CICS® Transaction Server for OS/390®

Release Guide

Release 3

GC34-5352-01

CICS® Transaction Server for OS/390®

Release Guide

Release 3

GC34-5352-01

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

Second edition (March 1999)

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

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

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

IBM United Kingdom Laboratories,
Information Development, Mail Point 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

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

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

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

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

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

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

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

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

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

Trademarks

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

AD/Cycle	C/370	CICS
CICS/ESA	CICSPlex	COBOL/370
DATABASE 2	DB2	DFSMS
Enterprise Systems Architecture/370	ES/3090	ESA/370
ESA/390	ES/9000	ESCON
IBM	IMS	IMS/ESA
Language Environment	MQSeries	MVS
MVS/ESA	Multiprise	OS/2
OS/390	Parallel Sysplex	PR/SM
RACF	S/390	System/390
SystemView	VisualAge	VTAM

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Tivoli and NetView are trademarks of Tivoli Systems Inc. in the United States, or other countries, or both.

UNIX is a trademark of X/Open Company Limited in the United States, or other countries, or both.

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

Preface

What this book is about

This book provides information about new and changed function in CICS® Transaction Server for OS/390® (CICS TS), Release 3. It gives an overview of the changes to reference information, and points you to the manuals where more detailed reference information is given.

The programming interface information given in this book is intended only to show what has changed from the previous release of CICS TS, and to help you plan your migration to the new release. For programming interface information, read the primary sources of programming interface and associated information in the following publications:

- *CICS Application Programming Reference*
- *CICS System Programming Reference*
- *CICS Customization Guide*
- *CICS External Interfaces Guide*
- *CICSPlex SM Application Programming Guide*
- *CICSPlex SM Application Programming Reference*

Who this book is for

This book is for those responsible for the following user tasks:

- Evaluation and planning
- System administration
- Programming
- Customization

It describes what is new, changed, and obsolete in the CICS and CICSPlex® SM elements of CICS Transaction Server for OS/390.

What you need to know to understand this book

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

How to use this book

This book is organised in seven parts:

- **Part 1** provides a brief summary of all new and changed function.
- **Part 2** describes new function that extends CICS support of Parallel Sysplex®.
- **Part 3** describes the new function designed to improve and simplify the task of CICS system management.
- **Part 4** describes the new function introduced to support application development and to help you to develop solutions that meet your business requirements.
- **Part 5** describes the new function introduced to support network computing over open computer networks.
- **Part 6** describes a number of miscellaneous changes.
- **Part 7** covers hardware and software requirement for CICS TS Release 3, and describes the publications that are available, in both hardcopy and softcopy.

Notes on terminology

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

“CICSPlex SM” is used for the CICSPlex System Manager element of IBM CICS Transaction Server for OS/39

“MVS™” is used for the operating system, which is a base element of OS/390.

Bibliography

CICS Transaction Server for OS/390

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

CICS books for CICS Transaction Server for OS/390

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

CICSPlex SM books for CICS Transaction Server for OS/390

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

Other CICS books

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

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

Part 1. Summary of CICS® TS Release 3

This Part provides a summary only of what is new and changed in Release 3 of CICS Transaction Server for OS/390.

Part 1 contains only the summary chapter:

Chapter 1. Summary of Release 3

This chapter summarizes what is new and changed in CICS Transaction Server for OS/390®, Release 3 under the following main topics:

Parallel Sysplex® support

CICS support for Parallel Sysplex environments is extended by the following new function:

CICSPlex SM is enhanced to include support for some of the functional changes to CICS for the Parallel Sysplex environment.

Sysplex enqueue and dequeue

The sysplex enqueue (ENQ) and dequeue (ENQ) function enables CICS transactions running in the same region, or in different regions within a sysplex, to serialize on a named resource using the existing CICS API. This extension to the scope of the CICS enqueue mechanism removes a major cause of inter-transaction affinity, enabling better exploitation of parallel sysplex environments, and thus providing better price/performance, capacity, and availability.

There are changes to CICSPlex SM in support of the CICS global enqueue/dequeue function.

See "Chapter 2. Sysplex enqueue and dequeue" on page 13 for details.

Coupling facility data tables

CICS coupling facility data tables enable user applications, running in different CICS regions that reside in one or more MVS images within a Parallel Sysplex, to share working data with update integrity.

Data in a coupling facility data table is accessed through the CICS file control application programming interface (API), enabling existing applications to use it, either without any modification, or with minimum modification, depending on the level of function required. Similar in some ways to user-maintained data tables, coupling facility data tables provides efficient sharing with update capability.

There are changes to some existing CICSPlex SM views, and a new view added, to provide support for CICS coupling facility data tables.

See “Chapter 3. Coupling facility data tables” on page 19 for details.

Dynamic routing of DPL and EXEC CICS START requests

CICS extends its dynamic routing facility to provide mechanisms for dynamically routing transactions started by distributed program link (DPL) requests, and a subset of START commands. Dynamic balancing for DPL includes:

- DPL requests from an external CICS interface (EXCI) client
- External Call Interface (ECI) requests from any of the CICS Client workstation products.

The routing mechanisms allow workload balancing to be managed by CICSPlex SM, making it possible to integrate workload balancing for EXCI clients, CICS Clients, and started tasks.

See “Chapter 4. Dynamic routing of DPL and EXEC CICS START requests” on page 41 for details.

System management

CICS system management is improved and made easier by the following new function:

- “Resource definition online for CICS temporary storage”
- Monitoring, statistics, and enterprise management changes
- Autoinstall for consoles.

CICSPlex SM is enhanced to include support for some of the functional changes to CICS system management.

Resource definition online for CICS temporary storage

CICS provides resource definition online (RDO) support for temporary storage queues. Instead of coding macros to define a temporary storage table (TST), you can define TSMODEL resource definitions for temporary storage queues in the CICS system definition (CSD) file. This RDO facility is provided by the DEFINE command on the CEDA transaction, and in the DFHCSDUP utility program. You can also discard TS queue resource definitions while CICS is running.

RDO for temporary storage eliminates the need to prepare a temporary storage table (TST) for batch assembly and link-edit. It also removes the need to shut down and restart CICS to make changes to TS queue definitions. RDO support for TS queues continues the CICS strategy of providing high availability and continuous operation.

CICSPlex SM provides several new operate views to support CICS RDO for TS queues.

See “Chapter 5. Resource definition online for CICS temporary storage” on page 59 for details.

Monitoring, statistics, and enterprise management changes

Many of the changes and new functions introduced in CICS are supported by additonal information provided by CICS monitoring and statistics domains. Remote monitoring of CICS regions is also provided by the addition of the CICSPlex SM Instrumentation component of Tivoli® Global Enterprise Manager.

See “Chapter 6. Monitoring, statistics, and enterprise management changes” on page 65 for details.

Autoinstall for consoles

Autoinstall for terminals is extended to include MVS consoles, making it unnecessary to define MVS consoles to CICS explicitly. Autoinstall for consoles uses the same autoinstall control program as for terminals.

See “Chapter 7. Autoinstall for MVS consoles” on page 73 for details.

Application support and solution enablement

CICS application support and solution enablement is extended by the following new function:

- “CICS business transaction services”
- “Open transaction environment”
- “Long temporary storage queue names” on page 6
- “EXCI enhancement for resource recovery” on page 6
- “Object-oriented interface to CICS services for C++” on page 6
- “JCICS interface to CICS services for Java” on page 7
- “VisualAge® for Java, Enterprise Edition for OS/390” on page 7
- “Support for the Java Virtual Machine” on page 7

CICS business transaction services

CICS business transaction services (BTS) provide an application programming interface (API) and support services that simplify the development of complex business transactions.

A real-world business transaction—for example, the booking of a holiday—may involve multiple actions that take place over an extended period. Traditionally, the individual actions that make up a complex business transaction have been mapped on to CICS transactions. CICS business transaction services provides a better way of modelling and managing complex business transactions.

See “Chapter 8. CICS business transaction services” on page 81 for details.

Open transaction environment

CICS enhances its internal architecture to enable specified user tasks to run under their own task control block (TCB). Initially, the main benefit of this change is for

Java application programs that run under a JVM. This will be followed by support for resource managers that, under the existing TCB structure, are forced to perform TCB switching to avoid unacceptable suspension (blocking) of the quasi-reentrant TCB (QR TCB). This is the TCB under which user tasks generally execute for most of their task lifetime.

The new TCBs under which tasks, optionally, can run are known as open TCBs. In the longer term, the introduction of these open TCBs will lead to CICS becoming an open transaction environment, in contrast to the closed, and somewhat restricted, environment of the past.

See “Chapter 9. Open transaction environment” on page 111 for details.

Long temporary storage queue names

The CICS temporary storage (TS) facility is enhanced to allow TS queues to have names up to 16-characters long, providing much greater flexibility in user application programs. The greater flexibility of 16-character names allows you to generate queue names in a variety of forms. For example, you could use the form *ttttSuuuuuuuu*, where *tttt* is the transaction identifier, *S* represents a sequence character (allowing you to have more than one queue for each user or transaction) and *uuuuuuuu* the userid. This example of name structure allows you to create temporary storage definitions with a prefix using only the *ttttS* part of the queue names.

Support for longer TS queue names removes many of the restrictions and difficulties that face application designers caused by the current 8-character limit.

See “Chapter 10. Long temporary storage queue names” on page 133 for details.

EXCI enhancement for resource recovery

CICS supports MVS recoverable resource management services (RRMS) for applications that use the external CICS interface (EXCI). In earlier releases, EXCI enforces a syncpoint by the CICS server region before returning control to the EXCI client program. With MVS recoverable resource management services (RRMS) support, there are the following enhancements:

- The unit of work within which the CICS server program changes recoverable resources can now become part of the MVS unit of recovery associated with the EXCI client program.
- The CICS server unit of work can be committed when the server program returns control to the client or can continue over multiple EXCI DPL calls, until the EXCI client decides to commit or backout the unit of recovery.

See “Chapter 11. EXCI enhancement for resource recovery” on page 139 for details.

Object-oriented interface to CICS services for C++

CICS introduces a new C++ object-oriented (OO) programming interface, which gives application programs access to the CICS services previously available only through the CICS command-level application programing interface (API). The CICS OO API, based on the CICS C++ foundation classes, gives a C++ programmer the choice of writing CICS application programs using either the traditional CICS command-level API or the CICS OO API classes.

See “Chapter 12. Object-oriented interface to CICS services for C++” on page 147 for details.

JCICS interface to CICS services for Java

CICS introduces a new programming interface for use in CICS application programs written in Java. Classes (known as JCICS) are provided to give access to a range of CICS services traditionally available through the CICS command-level API.

See “Chapter 14. VisualAge for Java, Enterprise Edition for OS/390” on page 151 for details.

VisualAge® for Java, Enterprise Edition for OS/390

This support uses the **VisualAge for Java, Enterprise Toolkit for OS/390 (ET/390)** to enable Java application programs to run under CICS control.

The Java language support is similar to CICS language support for COBOL or C++. The normal CICS program execution model is used, rather than a long-lived Java Virtual Machine (JVM).

The application program is developed and compiled, using a Java compiler (such as VisualAge for Java or javac) on a workstation or in the OS/390 UNIX System Services environment. The .class files produced are then processed by the **bytecode binder** component of ET/390, executing in the OS/390 UNIX System Services environment, to produce Java program objects stored in an MVS PDSE library.

CICS loads the files from the PDSE and executes the Java program in a Language Environment® (LE) run-unit, similar to C++, using run-time support in the CICS region provided by the Java run-time component of ET/390.

See “Chapter 14. VisualAge for Java, Enterprise Edition for OS/390” on page 151 for details.

Support for the Java Virtual Machine

CICS supports the Java Virtual Machine (JVM) to enable CICS application programs written in Java, and compiled to bytecode by any standard Java compiler, to run in the CICS address space under the control of a JVM. This support makes CICS fully Java compliant, and enables user application programs written in Java to use all the core Java classes.

See “Chapter 15. Support for the Java Virtual Machine” on page 155 for details.

e-Business enablement for network computing

CICS support for network computing is extended by the following new function:

- “Bridging to 3270 transactions” on page 8
- “Support for the secure sockets layer” on page 8
- “CORBA client support” on page 8

- "Enhancements to CICS Web support"

Bridging to 3270 transactions

CICS introduces the following enhancements to the 3270 bridge function:

- New options of the START command are provided to initiate a user transaction and establish the bridge environment. A **bridge transaction** is no longer needed for this purpose.
- Some restrictions on the CICS commands issued by a user transaction are removed. Support is added for:
  - START TRANSID TERMID commands, where TERMID is the bridge facility and TRANSID is local
  - RETURN IMMEDIATE
  - INPUTMSG on RETURN, XCTL and LINK
  - SET TERMINAL ATISTATUS

See "Chapter 16. Bridging to 3270 transactions" on page 167 for details.

Support for the secure sockets layer

The secure sockets layer (SSL) is a protocol for exchanging confidential information, in a secure form, across an insecure network such as the Internet. The OS/390 System Secure Sockets Layer (System SSL), introduced in OS/390 Release 7, is used by CICS to secure data for transmission through the CICS Web interface.

See "Chapter 17. Support for the secure sockets layer" on page 177 for details.

CORBA client support

CICS introduces support for IIOP requests inbound to Java application programs.

The Internet Inter-ORB protocol (IIOP) is a standard that can be used to provide communication between object-oriented application programs executing on different processors. It is part of the Common Object Request Broker Architecture (CORBA) specification, supporting distributed objects in a TCP/IP network.

CORBA is an architecture for distributed object middleware that separates client and server programs with a formal interface definition, and IIOP defines the message formats and protocols used in a CORBA distributed environment.

See "Chapter 18. CORBA client support" on page 183 for details.

Enhancements to CICS Web support

CICS Web support is enhanced by a number of major improvements, including: new resource definition types; a new set of API commands to handle HTTP requests and document templates; a new set of SPI commands; and the restructure of CICS Web support as a CICS domain providing improved reliability and servicebility.

See "Chapter 19. CICS Web support enhancements" on page 187 for details.

Miscellaneous enhancements

These include:

- The removal of runtime support for DB2® resource control tables (RCTs).
- A facility for generating sequence numbers that are unique across a Parallel Sysplex, using the services of a named counter server.
- The introduction of a new CICS system file, DFHDBFK, for storing IMS™ commands for use on the CDBM transaction.
- A change that enables CICSPlex SM API programs to run in USER key.
- A performance improvement for CICS application programs that run under Language Environment.
- A new option added to the INQUIRE TDQUEUE command.
- A new option, REMOVE, added to the CEDA and DFHCSDUP DELETE command.

See “Chapter 20. Miscellaneous changes” on page 197 for details.

Hardware and software requirements

Hardware and software requirements are described in “Chapter 21. Prerequisite hardware and software for CICS Transaction Server for OS/390” on page 207.

Part 2. Parallel Sysplex support

This Part describes the new function designed to extend CICS TS support for the System/390® Parallel Sysplex. It covers the following topics:

Chapter 2. Sysplex enqueue and dequeue

This chapter describes CICS sysplex enqueue and dequeue. It covers the following topics:

Overview

Changes to the CICS enqueue/dequeue function extend the CICS application programming interface to provide an enqueue mechanism that serializes access to a named resource across a specified set of CICS regions operating within a sysplex. This applies equally to a CICSplex within a single MVS image and to a CICSplex that resides in more than one MVS.

Local enqueues within a single CICS region are managed within the CICS address space. Sysplex-wide enqueues that affect more than one CICS region are managed by GRS. The main points of the changes to the CICS enqueue/dequeue mechanism are as follows:

- Sysplex enqueue and dequeue expands the scope of an EXEC CICS ENQ|DEQ command from region to sysplex, by introducing a new CICS resource definition type, ENQMODEL, to define resource names that are to be sysplex-wide.
- ENQSCOPE, an attribute of the ENQMODEL resource definition, defines the set of regions that share the same enqueue scope.
- When an EXEC CICS ENQ (or DEQ) command is issued for a resource whose name matches that of an installed ENQMODEL resource definition, CICS checks the value of the ENQSCOPE attribute to determine whether the scope is local or sysplex-wide, as follows:
  - If the ENQSCOPE attribute is left blank (the default value), CICS treats the ENQ|DEQ as local to the issuing CICS region.
  - If the ENQSCOPE is non-blank, CICS treats the ENQ|DEQ as sysplex-wide, and passes a queue name and the resource name to GRS to manage the enqueue. The resource name is as specified on the EXEC CICS ENQ|DEQ command, and the queue name is made up by prefixing the 4-character ENQSCOPE with the letters DFHE.
- The CICS regions that need to use sysplex-wide enqueue/dequeue function must all have the required ENQMODELs defined and installed.

  The recommended way to ensure this is for the CICS regions to share a CSD, and for the initialization group lists to include the same ENQMODEL groups.

Changes have been made to the CICS Affinity Utility to make it easier to create affinity groups for enqueues by address separately from enqueues by name.

Existing applications can use sysplex enqueues simply by defining appropriate ENQMODELs, without any change to the application programs.

Benefits

Sysplex enqueue provides the following benefits:

- Eliminates one of the most common causes of inter-transaction affinity.
- Enables better exploitation of a parallel sysplex providing better price/performance, capacity, and availability.
- Reduces the need for inter-transaction affinity rules in dynamic transaction routing programs thereby lowering the systems management cost of exploiting parallel sysplex.
- Enables serialization of concurrent updates to shared temporary storage queues, performed by multiple CICS tasks across the sysplex.
- Makes it possible to prevent interleaving of records written by concurrent tasks in different CICS regions to a remote transient data queue.
- Allows the single-threading and synchronization of tasks across the sysplex. It is not designed for the locking of recoverable resources.

Requirements

The hardware and software requirements for this function are the same as for CICS TS generally.

The CICS sysplex enqueue/dequeue function uses MVS GRS services. If you use the GRS star configuration, which is recommended for production CICS regions, you require a coupling facility.

Changes to CICS externals

There are changes to a number of CICS externals in support of the sysplex enqueue/dequeue facility. The externals affected are:

- "Changes to resource definition"
- "Changes to the system programming interface" on page 15
- "Changes to CICS-supplied transactions" on page 15
- "Changes to global user exits" on page 16
- "Changes to CICS Affinity Utility" on page 16

Changes to resource definition

A new resource type, ENQMODEL, is introduced to define the scope, name, and status of a CICS resource for use with the enqueue/dequeue function. The resource type is defined in the CSD, using the DEFINE ENQMODEL command in either the CEDA transaction or the DFHCSDUP utility program. The CEDA DEFINE panel for the ENQMODEL resource type is shown in Figure 1 on page 15.

```
ENQMODEL     ==> ........
Group        ==> ........
DEscription  ==> ..............................................
ENQSCope     ==> ....
STATUS       ==> ...            (ENABLED|DISABLED)
ENQNAME      ==> ..........................................
```

Figure 1. CEDA DEFINE panel for ENQMODEL resource definition

The ENQNAME attribute specifies a 1 - 255 character resource name. The permitted characters are A-Z 0-9 @ # and ¢, plus the asterisk as a wildcard character for generic names. Names ending with an * are treated as generic. The * must be used as the last character of the resource name.

The ENQSCOPE attribute specifies a group name for the CICS regions across which accesses to this resource name are to be serialized. For example, you could set ENQSCOPE to TEST for all test regions and PROD for all production regions. If you leave ENQSCOPE blank, CICS assumes that the ENQMODEL is local only, and that no other CICS regions are involved.

Changes to the system programming interface

The sysplex enqueue/dequeue facility is supported by the following changes to the system programming interface:

- The EXEC CICS INQUIRE UOWENQ command is enhanced with the addition of the ENQSCOPE and DURATION options.

  The EXEC CICS INQUIRE ENQ command is introduced as a synonym for EXEC CICS INQUIRE UOWENQ.
- There are four new SPI commands:
  - EXEC CICS CREATE ENQMODEL. This command enables you to define and install the specified ENQMODEL resource definition.
  - EXEC CICS DISCARD ENQMODEL. This command enables you to discard from the CICS region the specified ENQMODEL resource definition
  - EXEC CICS INQUIRE ENQMODEL. This command returns information about the specified ENQMODEL resource definition.
  - EXEC CICS SET ENQMODEL. This command enables you to change the value of the attributes of the specified ENQMODEL resource definition.

For details of all these commands, see the *CICS System Programming Reference* manual.

Changes to CICS-supplied transactions

The sysplex enqueue/dequeue facility is supported by the following changes to the CICS-supplied transactions:

- The CEMT INQUIRE UOWENQ command is enhanced with the addition of the ENQSCOPE and DURATION options.

  The CEMT INQUIRE ENQ command is introduced as a synonym for CEMT INQUIRE UOWENQ
- There are three new CEMT commands:
  - CEMT DISCARD ENQMODEL. This command enables you to discard from the CICS region the specified ENQMODEL resource definition.

- CEMT INQUIRE ENQMODEL. This command returns information about the specified ENQMODEL resource definition.
- CEMT SET ENQMODEL. This command enables you to change the value of the attributes of the specified ENQMODEL resource definition.

For details of all these commands, see the *CICS Supplied Transactions* manual.

Changes to global user exits

The XNQEREQ global user exit is enhanced by the addition of UEPSCOPE, an exit-specific parameter that addresses a 4-byte area containing the scope name from the ENQMODEL used for the ENQ/DEQ request. XNQEREQ is invoked before CICS processes an EXEC CICS ENQ or DEQ request, allowing you to change the ENQSCOPE name associated with the request.

CICS also provides a sample XNQEREQ global user exit program, DFH$XNQE. This sample program illustrates three methods of adding an ENQSCOPE name to ENQ and DEQ requests, so that they apply to multiple regions within a sysplex. These methods are:

- Prefix the resource name with a 4-character ENQSCOPE value, where the resource name is shorter than 250 characters, and matches an entry in an installed ENQMODEL resource definition.
- Replace the first 4-characters of the resource name with an ENQSCOPE value, where the resource name is longer 4 characters, and matches an entry in an installed ENQMODEL resource definition.
- Place a 4-character ENQSCOPE value in the area addressed by UEPSCOPE, and return UERCSCPE in R15. This method is not generally recommended.

For details of this sample exit program, see the source code supplied in CICSTS13.CICS.SDFHSAMP.

See the *CICS Customization Guide* for details of these new global user exit points.

The XRSINDI global user exit is invoked for the install and discard of ENQMODEL resource types.

Changes to CICS Affinity Utility

The CICS Affinity Utility is changed so that:

- For an ENQ or DEQ by name (that is, one for which the EXEC CICS ENQ or EXEC CICS DEQ command included a length parameter), if there is a non—blank ENQSCOPE name in the appropriate ENQMODEL definition the detector disregards any matching EXEC CICS ENQ|DEQ (because use of the ENQMODEL makes the ENQ|DEQ sysplex-wide and thus removes the implied affinity). The reporter does not report any such ENQs or DEQs because the detector does not detect them.
- For an ENQ or DEQ by address (that is, one for which the EXEC CICS ENQ or DEQ command does not include a length parameter), or if there is no matching ENQMODEL definition, or one with a blank ENQSCOPE name, the ENQ|DEQ is local and continues to be reported as an affinity.
- The scanner and the reporter now distinguish between ENQ NAME and ENQ ADDRESS in their reports, where previously only ENQ was shown. Similarly, DEQ is now reported as either DEQ NAME or DEQ ADDRESS.

CICSPlex SM support

CICSPlex SM supports the extended EXEC CICS ENQ and EXEC CICS DEQ commands by providing:

- A new BAS definition object ENQMDEF, which defines enqueue resources that are to be sysplex-wide.
- New operations views ENQMDL, ENQMDLD, and ENQMDLS, that allow you to display and manage information about enqueue model resource descriptions.
- A new field Enqscope on the UOWENQ view.
- New resource tables:
  - CRESENQM
  - ENQINGRP
  - ENQMDEF
  - ENQMODEL
  - ERMCENQM

Chapter 3. Coupling facility data tables

This chapter describes CICS coupling facility data tables. It covers the following topics:

Overview

CICS coupling facility data tables are designed to provide rapid sharing of working data within a sysplex, with update integrity. The data is held in a table that is similar in many ways to a shared user-maintained data table, and the API used to store and retrieve the data is based on the file control API used for user-maintained data tables.

Because read access and write access have similar performance, this new form of table is particularly useful for informal shared data. Informal shared data is characterised as:

- Data that is relatively short-term in nature (it is either created as the application is running, or is initially loaded from an external source)
- Data volumes that are not usually very large
- Data that needs to be accessed fast
- Data of which the occasional loss can be tolerated by user applications
- Data that commonly requires update integrity.

Informal data can include scratchpad data, shared queues, keyed data, and application control records.

Typical uses might include sharing scratchpad data between CICS regions across a sysplex, or sharing of files for which changes do not have to be permanently saved. There are many different ways in which applications use informal shared data, and most of these could be implemented using coupling facility data tables. Coupling facility data tables are particularly useful for grouping data into different tables, where the items can be identified and retrieved by their keys. For example, you could use a field in a coupling facility data table to maintain the next free order number for use by an order processing application, or you could maintain a list of the numbers of lost credit cards in a coupling facility data table. Other examples are:

- Look-up tables of telephone numbers
- Creating a subset of customers from a customer list
- Information that is specific to the user of the application, or that relates to the terminal from which the application is being run.

For many purposes, because it is global in scope, coupling facility data tables can offer significant advantages over resources such as the CICS common work area (CWA).

Comparison with user-maintained data tables

To an application, a coupling facility data table (CFDT) appears much like a sysplex-wide user-maintained data table, because it is accessed in the same way using the file control API. However, in a CFDT there is a maximum key-length restriction of 16 bytes.

See the *CICS Shared Data Tables Guide* for information about shared data tables support.

Coupling facility data table models

There are two models of coupling facility data table:

- The contention model, which gives optimal performance but normally requires new programs to exploit it. This is because there is a new API condition code to indicate to the application program that the data has been changed since it issued a read-for-update request. The new CHANGED response can occur on a REWRITE or DELETE command. There is also a new use for the existing NOTFND response, which may be returned to indicate to the application program that the record has been deleted since the program issued the read-for-update request.

  Note: It might be possible to use existing programs with the contention model if you are sure they cannot receive the CHANGED or NOTFND exceptions on a REWRITE or DELETE. An example of this could be where an application program operates only on records that relate to the user of the program, and therefore no other user could be updating the same records.

  This model is non-recoverable only: CFDT updates are not backed out if a unit of work fails.
- The locking model, which is API-compatible with existing programs that conform to the UMT subset of the file control API. (This subset is nearly, but not quite, the full file control API.)

  This model can either be:

  – **Non-recoverable**: locks do not last until syncpoint, and CFDT updates are not backed out if a unit of work fails, or

  – **Recoverable**: coupling facility data tables are recoverable in the event of a unit of work failure, and in the event of a CICS region failure, a CFDT server failure, and an MVS failure (that is, updates made by units of work that were in-flight at the time of the failure are backed out).

    The recoverable locking model supports in-doubt and backout failures: if a unit of work fails when backing out an update to the CFDT or if it fails in-doubt during syncpoint processing the locks are converted to retained locks and the unit of work is shunted.

    CFDTs cannot be forward recoverable. A CFDT does not survive the loss of the CF structure in which it resides.

You specify the model you want for each table on its file resource definition, enabling different tables to use different models.

Coupling facility data table structures and servers

Coupling facility data tables are held in coupling facility structures. Access to a coupling facility data table is through a CFDT server. Coupling facility data tables

allow you to separate related groups of coupling facility data tables by storing them in separate pools. For example, you might want to have one pool for production and another for test.

A coupling facility data table pool is a coupling facility list structure, and access to it is provided by a coupling facility data table server. Within each MVS image, there must be one CFDT server for each CFDT pool; in other words, for any given pool, there must be a server in each MVS that has CICS regions that need access to it. Coupling facility data table pools are defined in the coupling facility resource management (CFRM) policy. The pool name is then specified in the start-up JCL for the table server.

Coupling facility data table pools can be used almost continuously and permanently. There are utility commands that you can use to minimize the impact of maintenance.

Access using file control API

A coupling facility data table is accessed from CICS through file control commands. The file name specified on the command indicates the name of the table and pool in which it resides. The table name is either specified on the file definition or is the same as the file name, and the pool name is specified on the file definition. The table is uniquely identified by the pool name and table name, so that two tables with the same name may exist in different pools, and will be entirely separate entities.

Automatic connection to coupling facility data table pools

CICS automatically connects to the coupling facility data table server for a given pool the first time that a coupling facility data table within that pool is referenced. CICS also automatically reconnects to the coupling facility data table server when the server has restarted after a failure.

Creating coupling facility data tables

A coupling facility data table that does not already exist is created automatically by the first server to receive a request from a CICS region that requires the coupling facility data table to be opened. The CFDT is then used by the file that referenced it, and also by other CICS regions that issue subsequent open requests for other files that name the same coupling facility data table.

CICS can optionally load the coupling facility data table automatically from a source VSAM (KSDS) data set when it is first opened. Unlike user-maintained data tables, with coupling facility data tables you can specify that there is no associated source data set, allowing you to create an empty CFDT. Like a user-maintained data table, updates to a CFDT are not reflected in the source VSAM data set, even when it is defined as recoverable

Your application programs have access to a coupling facility data table as soon as it is created, although there are some restrictions on the keys that can be accessed while data is being loaded.

Administering coupling facility data tables

CICS provides some utility functions that allow you to obtain, from a coupling facility data table server, summary information and periodic statistics on coupling facility data tables defined within a pool. This information is designed to help you administer coupling facility data table pools, and to help you evaluate capacity.

Benefits

The major benefits of coupling facility data tables are:

Sysplex-wide sharing of data
You can share data across a sysplex with update integrity. This means that changed data cannot be accidentally overwritten by other tasks running concurrently. This is in contrast to facilities such as CICS temporary storage data sharing, which does not provide serialized access.

Coupling facility data tables is useful for data that:
- Consists of small items
- Is very volatile
- Needs updating frequently
- Requires high availability.

All connected CICS regions can update information held in a coupling facility data table on an equal basis.

Performance
Coupling facility data tables offers good performance for file control API operations (such as READ, WRITE, REWRITE, and DELETE) for reasonable record lengths (up to around 4K), compared with other means of sharing data in a sysplex.

Ease of use through existing API
Access to coupling facility data tables is through the existing file control API, using the same subset as that supported for user-maintained data tables. This makes coupling facility data tables easy to use, allows compatibility with existing applications where required, and means that applications written to use coupling facility data tables will be compatible with existing types of VSAM file access supported by CICS (CMTs, UMTs, and non-data-table files). This allows vendor programs to run both on CICS regions that support coupling facility data tables and on those that do not.

Server security
A security check is carried out when a CICS region connects to a coupling facility data table pool. An additional security check can optionally be carried out when a CICS region connects to a given coupling facility data table.

Availiablity
If a CICS region fails, most access from other regions can continue without impact.

With non-recoverable coupling facility data tables (using either contention or non-recoverable locking), all access can continue.

With recoverable coupling facility data tables (using the recoverable locking model), the only records that remain locked and unavailable are records updated by failed units of work (UOWs), such as indoubt-failed. In-flight UOWs are backed out by the server at the time of the CICS failure. In-doubt UOWs could either have failed in-doubt before the CICS region failed, or could be in-doubt failures caused in other CICS regions, because the failed CICS was a co-ordinator for the UOWs.

Requirements

The software requirements for coupling facility data tables are the same as for CICS TS generally, but in addition to the same hardware as for CICS TS, you also require a coupling facility.

Storage usage

Almost all the storage (other than that required by any file control request) is allocated in the server address space and in the coupling facility. The amount of server storage used depends on the number of concurrent requests.

Changes to CICS externals

There are changes in a number of CICS externals to support the coupling facility data tables. These are:

Changes to system definition

Before you can begin to use coupling facility data tables, you need to define the required coupling facility data table pools, write some JCL to start a coupling facility data table server, and understand how to use coupling facility data table server commands.

These tasks are summarized briefly in the following sections. For detailed information, see the *CICS System Definition Guide*.

Defining a coupling facility data table pool

A coupling facility data table pool is a coupling facility list structure, which you define in a coupling facility resource manager (CFRM) policy in a sysplex couple data set. A coupling facility data table pool, containing one or more coupling facility data tables, is accessed by CICS through a server region using cross-memory services. From the application point of view, a pool and its server are similar to a file-owning region, and the pool can contain any number of tables provided that each one has a unique table name.

Before a coupling facility data table server can use its pool, the active CFRM policy must contain a definition of the list structure to be used for the pool. To achieve this, add a statement that specifies the list structure to a selected CFRM policy, and then activate the policy.

For example, Figure 2 shows a policy definition that you can add using the administrative data utility, IXCMIAPU. You create the name of the list structure for a coupling facility data table pool by adding the prefix DFHCFLS_ to your chosen pool name, giving DFHCFLS_*poolname*.

```
STRUCTURE  NAME(DFHCFLS_PRODCFT1)
   SIZE(1000)
   INITSIZE(500)
   PREFLIST(FACIL01,FACIL02)
```

Figure 2. Example of defining a coupling facility data table structure

Activating a CFRM policy: When you have defined the list structure in a CFRM policy, activate the policy using the MVS command:

```
SETXCF START,POLICY,POLNAME=policy_name,TYPE=CFRM
```

Starting up AXM system services

The execution environment for coupling facility data table server regions is provided by a run-time environment package called the authorized cross-memory (AXM) server environment (which is also used for CICS temporary storage data sharing). To establish AXM cross-memory connections for an MVS image, initialize the AXM system services by defining a subsystem called AXM. For example, add the following entry to the appropriate IEFSSN*xx* member of SYS1.PARMLIB:

```
SUBSYS SUBNAME(AXM) INITRTN(AXMSI)
```

Starting coupling facility data table server regions

You activate a coupling facility data table pool in an MVS image by starting up a coupling facility data table server region for that pool. You can start the server program, DFHCFMN, as a started task, started job, or as a batch job. The server program must be run from an APF-authorized library, and is supplied in CICSTS13.CICS.SDFHAUTH. An example of the JCL needed to start a coupling facility data table server is shown in Figure 3.

```
//PRODCFD1 JOB  ...
//CFSERVER EXEC PGM=DFHCFMN,REGION=40M          CICS CFDT Server
//STEPLIB  DD   DSN=CICSTS13.CICS.SDFHAUTH,DISP=SHR  Authorized library
//SYSPRINT DD   SYSOUT=*                        Messages and statistics
//SYSIN    DD   *
POOLNAME=PRODCFD1                               Pool name
MAXTABLES=100                                   Allow up to 100 tables
/*
```

Figure 3. Sample job to start a coupling facility data table server

Coupling facility data table server parameters

The POOLNAME parameter, shown in Figure 3 is required when you start a coupling facility data table server. Other parameters, some of which are used only for initial allocation of a new pool list structure, are optional. There are parameters to enable you to control:

- Server security
- Server statistics
- List structure attributes
- Debug trace options
- Tuning of lock structure ratios
- Lock wait intervals
- Warning message thresholds
- Automatic structure alter thresholds.

All the parameters are described in the *CICS System Definition Guide*.

Controlling coupling facility data table server regions

You can issue commands to control a coupling facility data table server, using the MVS MODIFY (F) command to specify the job or started task name of the server region, followed by the server command. The general form of a coupling facility data table server command, using the short form F, is as follows:

```
F server_job_name,command  parameters...  comments
```

The commands supported by a coupling facility data table server are:
- SET
- DISPLAY
- PRINT
- DELETE TABLE
- STOP
- CANCEL

These commands and their options are as follows:

SET *keyword=operand[,keyword=operand,...]*
Change one or more server parameter values. The command can be abbreviated to **T**, as for the MVS SET command.

DISPLAY *keyword[=operand][,keyword[=operand,]...]*
Display one or more parameter values, or statistics summary information, on the console. The valid keywords for DISPLAY are all the initialization parameters, plus some others.

The command can be abbreviated to **D**, as for the MVS DISPLAY command.

PRINT *keyword[=operand][,keyword[=operand,]...]*
Produces the same output as DISPLAY, supporting the same keywords, but on the print file only.

DELETE TABLE=*name*
Delete the named table. The table must not be in use for this command to succeed. The command can be abbreviated to **DEL**.

STOP
Terminate the server normally. The server waits for any active connections to terminate first, and prevents any new connections while it is waiting. The command can be abbreviated to **P**.

Note: You can also use the MVS STOP|P *[jobname.]identifier*[,A=*asid*] command, which is equivalent to issuing the server STOP command through the MVS MODIFY command.

CANCEL
Terminate the server immediately.

See the *CICS System Definition Guide* for details of these commands.

Deleting or emptying coupling facility data table pools

You can delete a coupling facility data table pool using the MVS **SETXCF** command to delete its coupling facility list structure.

For example:

```
SETXCF FORCE,STRUCTURE,STRNAME=DFHCFLS_poolname
```

You can delete a structure only when there are no servers connected to the pool, otherwise MVS rejects the command.

When you attempt to start a server for a pool that has been deleted (or attempt to reload the pool), it is allocated as a new structure. The newly allocated structure uses size and location attributes specified by the currently active CFRM policy, and other values determined by the server initialization parameters (in particular, **MAXTABLES**).

Unloading and reloading coupling facility data table pools

You can unload, and reload, the complete contents of a coupling facility data table pool to and from a sequential data set by invoking the server program with the **FUNCTION** parameter, using the **UNLOAD** and **RELOAD** options. The unload and reload process preserves not only the table data, but also all recovery information such as unit of work status and record locks for recoverable updates.

You can use this function, for example, to:

- Preserve the coupling facility data table pool during planned coupling facility maintenance, or
- Move the pool to a different coupling facility.
- Increase the size of the pool's list structure.

  If the maximum number of tables specified in the original pool was too small, or the pool has reached its maximum size and needs to be expanded further, unload the pool, then delete the structure so that the reload process can reallocate it with more space.

Changes to resource definition

New options are added to the FILE resource definition and some existing options are used in new ways for coupling facility data tables.

File resource definition changes

New attributes are added to the DEFINE FILE command, and on the CEDA panel, the "REMOTE ATTRIBUTES" group has been split into two groups. The RECORDSIZE and KEYLENGTH attributes are now shown under a new group titled "REMOTE AND CFDATATABLE PARAMETERS" as shown in Figure 4 on page 27.

```
REMOTE ATTRIBUTES
 REMOTESystem   :
 REMOTEName     :

REMOTE AND CFDATATABLE PARAMETERS
 RECORDSize   ==>                    1 - 32767
 Keylength    ==>                    1 - 255 (1-16 for CF Datatable)
```

Figure 4. Changes to remote attributes on the CEDA DEFINE FILE panel

There are new options for the TABLE and MAXNUMRECS attributes in the "DATATABLE PARAMETERS" group, as shown in Figure 5.

```
DATATABLE PARAMETERS
 Table        ==> No                 No | CIcs | User | CF
 Maxnumrecs   ==> Nolimit            Nolimit | 1 - 99999999
```

Figure 5. New options for data table attributes on the DEFINE FILE command

There is a new group of attributes for coupling facility data tables, "CFDATATABLE PARAMETERS" as shown in Figure 6.

```
CFDATATABLE PARAMETERS
 Cfdtpool     ==>
 TABLEName    ==>
 UPDateModel  ==> Locking            Contention | Locking
 LOad         ==> No                 No | Yes
```

Figure 6. New attributes for DEFINE FILE command

The new and changed parameters for coupling facility data tables are summarized as follows:

TABLE(NO|CICS|USER|CF)
The CF option, for a coupling facility data table, is added to this existing parameter.

MAXNUMRECS(NOLIMIT|*number***)**
The NOLIMIT option is added, and the range is changed to 1 through 99 999 999. The new values apply to all forms of data table.

CFDTPOOL(*name***)**
This new parameter specifies the name of the coupling facility data table pool that contains the coupling facility data table to which the file definition refers.

TABLENAME(*name***)**
This new parameter specifies the name of the coupling facility data table to which the file definition refers. If you omit this attribute when TABLE(CF) is specified, it defaults to the name specified for the FILE.

UPDATEMODEL(LOCKING|CONTENTION)
This new parameter specifies the type of update model to be used for a coupling facility data table.

RECORDSIZE*(number)*
Specifies the maximum record size in bytes for remote files and coupling facility

data tables, in the range 1 through 32767. For a coupling facility data table, the record size is required only if the file definition also specifies LOAD(NO).

KEYLENGTH*(value)*
Specifies the key length for remote files and coupling facility data tables. For a coupling facility data table, KEYLENGTH is required only if the table specifies LOAD(NO). For a coupling facility data table, the key length is restricted to a maximum of 16 bytes.

LOAD(NO|YES)
This new parameter specifies whether the coupling facility data table is to be loaded from a source data set when first opened.

DSNAME(*name***)**
Specifies the name of the source data set from which a coupling facility data table is to be loaded when this file is opened with the LOAD(YES) attribute, and the data table is being created. You can also specify the name of the source data set on a DD statement for the file.

RECORDFORMAT(V|F)
The record format must specify V (for variable) if TABLE(CF) is specified.

JOURNAL(NO|*number***)**
Specify NO for this parameter because file operations to a coupling facility data table are not journaled. If you do specify a journal number for a CFDT, CICS attempts to open the log stream specified for the journal when opening the file.

RECOVERY(NONE|BACKOUTONLY|ALL)
Coupling facility data tables defined with UPDATEMODEL(CONTENTION) cannot be recoverable, and must specify RECOVERY(NONE).

Coupling facility data tables defined with UPDATEMODEL(LOCKING) can specify RECOVERY(NONE) or RECOVERY(BACKOUTONLY).

FWDRECOVLOG(NO|*number***)**
This attribute is ignored for a coupling facility data table. A coupling facility data table is not forward recoverable.

BACKUPTYPE(STATIC|DYNAMIC)
This attribute is ignored for a coupling facility data table. A CFDT is not eligible for dynamic (BWO) backup.

Changes to the application programming interface

In general, the API commands that operate on a coupling facility data table that is updated using the locking model (either non-recoverable or recoverable) are upwards compatible with the commands for a user-maintained data table; that is, any differences from the UMT API do not require existing applications to be rewritten.

The API for a coupling facility data table that uses the contention model, however, introduces a new condition, which affects the following file control API commands:
- REWRITE
- DELETE

Although you are advised to write new application programs to use a coupling facility data table that uses the contention update model, such new applications would be compatible with other types of file. Note that to be fully compatible, such

new applications should not be written in such a way that deadlocks become possible when they access a coupling facility data table that uses the locking update model.

The changes to the file control API to support coupling facility data tables are as follows:

New condition on the REWRITE and DELETE command

For a coupling facility data table using the contention update model, the REWRITE and DELETE commands succeed only if the record is unchanged since it was read for update. The new exception condition is:

CHANGED
A REWRITE or DELETE command is issued to a coupling facility data table defined with the contention update model and the record has been changed since it was read for update. To complete the operation, repeat the read for update to get the latest version of the record, and try again.

Additional meanings for some existing conditions

There are additional meanings for some existing exception conditions:

SYSIDERR
There are two additional RESP2 values that can be returned with the SYSIDERR condition on coupling facility data tables:

131 For a coupling facility data table, the connection to the coupling facility data table server has failed. This could be because the server itself has failed, or the server is available, but CICS has failed to connect to it.

132 The command is issued against a coupling facility data table that no longer exists, probably because of a coupling facility failure, in which case the coupling facility data table server also fails.

IOERR
For those file control commands that can return IOERR, the reason could be a bad response returned from a coupling facility access.

NOSPACE
For those file control commands that can return NOSPACE, the RESP2 value can be 102 for a coupling facility data table and there is a new RESP2 value that can be returned for operations on a coupling facility data table:

102 Can occur if the maximum number of records for a coupling facility data table is exceeded.

For a recoverable coupling facility data table, CICS can return this condition on a WRITE request when the CFDT apparently contains fewer than the maximum number of records specified, if there are uncommitted updates outstanding. The NOSPACE condition with a RESP2 of 102 can also be returned on a REWRITE command to a recoverable coupling facility data table, because this requires an extra record in the CFDT for recovery purposes, until the update has been committed.

108 There is insufficient storage in the coupling facility data table pool to perform the request. This can occur on a WRITE or a REWRITE command.

NOTFND
The NOTFND condition, when raised on a REWRITE or DELETE request to a CFDT using the contention model, means that the record has been deleted since it was read for update.

LOCKED
The LOCKED condition is possible for a READ UPDATE request to a recoverable CFDT if the record to be read is locked by a retained lock.

API restrictions during loading

There are restrictions on which API commands you can use while a server is loading a coupling facility data table. You can issue any file control request that is supported for a coupling facility data table, but it receives a LOADING condition if the loading is still in progress and the request is for a key that is outside the range of keys loaded. A REWRITE request for a record key outside the range of the records loaded so far receives an INVREQ, because such a request must have been preceded by a READ UPDATE, which must have failed with the LOADING condition before the REWRITE could be issued.

This behavior is different from that of a user-maintained data table during loading, for which only non-update reads with precise keys are allowed. In the case of a UMT, reads with precise keys succeed both for keys that are already loaded and those that are not.

Function shipping

If a release of CICS that does not support coupling facility data tables function ships a file control request to a file-owning region where the remote file refers to a coupling facility data table, and the FOR returns the new exception condition CHANGED, the local CICS returns an ERROR exception condition to the application program. The default action for ERROR is to abend the task.

If one of the existing conditions is returned on a request for which it is not possible without coupling facility data tables, the condition is correctly raised.

Changes to the system programming interface

There are changes to the CICS system programming interface to support coupling facility data tables:

- The EXEC CICS CREATE FILE command supports the changes to file resource definitions described under “Changes to resource definition” on page 26.
- The EXEC CICS INQUIRE FILE and EXEC CICS SET FILE commands are changed to support files that refer to coupling facility data tables.
- There is a new EXEC CICS INQUIRE CFDTPOOL command.
- The EXEC CICS INQUIRE UOWLINK command is enhanced to provide additional information for units of work involving coupling facility data tables.

Changed INQUIRE FILE command

The following options are added to the INQUIRE FILE command to return information about a coupling facility data table:

LOADTYPE(*cvda***)**
Returns a CVDA indicating whether the coupling facility data table is preloaded by the server from a source data set. The CVDA values returned are LOAD, NOLOAD, and NOTAPPLIC.

CFDTPOOL(*data-area***)**
Returns the 8-character name of the coupling facility data table pool in which the coupling facility data table resides.

TABLENAME(*data-area***)**
Returns the 8-character name of the table. If a specific table name is not defined on the file resource definition CICS returns the filename as the table name.

UPDATEMODEL(*cvda***)**
Returns a CVDA indicating the update model in use for the coupling facility data table. The CVDA values are CONTENTION, LOCKING, and NOTAPPLIC.

There are some minor changes to the descriptions of some existing options, as follows:

ACCESSMETHOD
Returns a CVDA value of VSAM for a coupling facility data table because the coupling facility data table API is compatible with the API used for VSAM files.

BASEDSNAME and **DSNAME**
Returns the 44-character data set name of the source data set if one is specified for the coupling facility data table.

EMPTYSTATUS
Returns a CVDA value of NOEMPTYREQ for a coupling facility data table.

FWDRECSTATUS
Returns a CVDA value of NOTFWDRECVBLE for a coupling facility data table.

KEYLENGTH
Returns a fullword binary field containing the keylength of the records in a coupling facility data table.

KEYPOSITION
Returns the key position of the records in the source data set if one is specified for a coupling facility data table.

MAXNUMRECS
The maximum number of records can now be in the range 1 through 99 999 999. Also, the special value of 0 is returned for a coupling facility data table that has no limit on its number of records.

OBJECT
Returns a CVDA value of BASE for a coupling facility data table.

CFDTPOOL
Returns the 8-character name of the coupling facility data table pool specified on the file definition (or blanks if the file does not refer to a coupling facility data table).

READINTEG
Returns a CVDA value of NOTAPPLIC for a coupling facility data table.

RECORDSIZE
Returns a fullword binary field containing the maximum record size for a coupling facility data table, taken either from the file resource definition or from VSAM if the CFDT is preloaded from a source data set and the file is open.

TABLE(*cvda***)**
A new CVDA value, CFTABLE, for a coupling facility data table, is added to CICSTABLE, NOTTABLE, and USERTABLE.

There are no new or changed conditions for coupling facility data tables.

See the *CICS System Programming Reference* for more information.

Changed SET FILE command

You can use the EXEC CICS SET FILE command to alter the file attributes that refer to a coupling facility data table, providing the file is closed. This includes changing a definition from TABLE(NO) to TABLE(CF), or from TABLE(CF) to TABLE(NO). In the latter case, the CFDT continues to exist in its coupling facility list structure—it is not deleted.

If a coupling facility data table already exists, and some table attributes specified on an EXEC CICS SET FILE command are inconsistent with those used when it was created, an attempt to open the file fails.

The new options are:

CFDTPOOL(*data-value***)**
Specify the name of the coupling facility data table pool in which the CFDT resides.

KEYLENGTH(*data-value***)**
Specify a key length for records in a coupling facility data table.

LOADTYPE(*cvda***)**
Specify whether the CF data table is to be preloaded from a data set. The values you can use are LOAD and NOLOAD.

RECORDSIZE(*data-value***)**
Specify a record size for records in a coupling facility data table.

TABLENAME(*data-value***)**
Specify the table name for a coupling facility data table.

UPDATEMODEL(*cvda***)**
Specify the update model to be used for the coupling facility data table. The CVDA values are CONTENTION and LOCKING.

The following existing options also apply to coupling facility data tables:

DSNAME
Specifies the name of the source data set from which a coupling facility data table is to be loaded.

MAXNUMRECS
Specifies the maximum number of records, for which the range is extended to an upper limit of 99 999 999.

OPENSTATUS
Specifies the open status for a file. If the file refers to a coupling facility data table, setting a file open for the first time in the sysplex causes the server to create the table, and if specified, load it from its source data set.

TABLE
Specifies the type of data table the file is to use, and a new CVDA value is added for a coupling facility data tables (CFTABLE).

Although there are no new exception conditions (RESP) returned on an EXEC CICS SET FILE command, there are new RESP2 values that can be returned with the INVREQ condition, and RESP2 value 38 is removed. The new values are:

INVREQ

New RESP2 values:

55 LOADTYPE has an invalid CVDA value.

56 UPDATEMODEL has an invalid CVDA value.

57 EMPTYSTATUS is invalid for a coupling facility data table—it must be NOEMPTYREQ.

58 CFDTPOOL is not specified for a file that refers to a coupling facility data table.

59 KEYLENGTH is not specified for a file that refers to a coupling facility data table specified with LOAD(NO).

60 An invalid KEYLENGTH is specified. The KEYLENGTH must be in the range 1 through 16 for a coupling facility data table.

61 RECORDSIZE is not specified for a file that refers to a coupling facility data table specified with LOAD(NO).

62 An invalid RECORDSIZE is specified. RECORDSIZE must be in the range 1 through 32767 bytes.

63 OPEN has been specified for a file that refers to a coupling facility data table, but open processing has failed because the file attributes of the file do not match those specified when the CFDT was created.

64 OPEN is specified for a file that refers to a coupling facility data table, but OPEN processing has failed because the server is not available.

65 An invalid CFDTPOOL name is specified.

66 An invalid TABLE name is specified.

67 An UPDATEMODEL of CONTENTION is specified for a recoverable coupling facility data table. The update model must be LOCKING for a coupling facility data table that is recoverable.

See the *CICS System Programming Reference* for more details, especially for information about how CICS responds if a CFDT-specific value is specified for a non-CFDT file, or a non-CFDT value for a CFDT file.

New INQUIRE CFDTPOOL command

The INQUIRE CFDTPOOL command allows you to discover whether CICS is connected to a named coupling facility data table pool, and to browse through all the pools for which the CICS region has installed file definitions.

Syntax:

Options:

START|NEXT|END

Pool browse operations. START is required to commence a browse and will not return values; likewise with END, which ends a browse. NEXT returns the next POOL.

CONNSTATUS

returns a CVDA value indicating the connection status of the pool. CVDA values are:

CONNECTED
The server for the coupling facility data table pool is available, and this CICS is currently connected to it.

NOTCONNECTED
The server for the coupling facility data table pool is available, but this CICS is not currently connected to it.

UNAVAILABLE
The server for the coupling facility data table pool is currently unavailable.

Conditions:

POOLERR
RESP2 values:
1 The named CFDT pool was not found (that is, CICS has not connected to it)
2 The CFDT pool was removed whilst the pool chain was being browsed

ILLOGIC
RESP2 values:
1 A START has not been given before a NEXT or END, or a START has been given before ending a previous START

END
RESP2 values:
2 There are no more coupling facility data table pools to be browsed

This is returned on the NEXT browse option.

NOTAUTH
RESP2 values:
100 Use of the command is not authorized

Changes to INQUIRE UOWLINK

The LINK and TYPE options are enhanced to provide information about a unit of work that involves a connection to a CFDT server. The full descriptions of these options are as follows:

LINK(*data-area***)**
returns the name associated with the link, depending on the type of connection, as follows:

- For a TYPE value of CONNECTION, CICS returns the 8-character netname of the remote system.
- For a TYPE value of RMI, CICS returns the entry name of the task-related user exit
- For a TYPE value of CFTABLE, CICS returns the the 8-character name of the coupling facility data table pool.

TYPE
returns a CVDA value indicating the type of connection. CVDA values are:

CFTABLE
A connection to a CFDT server.

CONNECTION
A connection defined in a CONNECTION resource definition.

RMI A connection to an external resource manager using the resource manager interface (RMI).

Changes to global user exits

The existing shared data tables global user exit points XDTRD, XDTLC, and XDTAD are invoked by CICS for coupling facility data tables.

XDTRD is invoked when a coupling facility data table is being loaded from its source data set. You can use an XDTRD global user exit program to select records for inclusion in a coupling facility data table, and to modify records before they are written to the CFDT. XDTRD is not invoked for a coupling facility data table that is not loaded from a source data set.

XDTLC is invoked on completion of loading of a coupling facility data table, whether successful or otherwise. You can use an XDTLC global user exit program to decide whether to accept an unsuccessfully loaded coupling facility data table. If the exit decides to accept the table, it remains open and available for access, but CICS does not mark it as loading completed. This is also the default action if no XDTLC exit is enabled. This means that application programs continue to receive the LOADING condition for any records that are beyond the key range of records successfully loaded into the table. This ensures that application programs are aware that not all the expected data is available. It also allows you to retry the load, when the cause of the failure has been corrected, by closing the file that initiated the load and reopening it. Alternatively, you could open another load-capable file that refers to the same data table. If your exit program decides to reject the table, it is closed and the records already loaded remain in the table. If the cause of the failure is corrected, a subsequent open for the data table allows the load to complete.

XDTLC is not invoked for a coupling facility data table that is not loaded from a source data set.

XDTAD is invoked when a record is being written to a coupling facility data table by an application program request. You can use an XDTAD global user exit program to decide whether to accept or reject the record being written.

The data tables user exit parameter list, DT_UE_PLIST, has a new indicator added to the flags byte to indicate that the exit has been invoked for a coupling facility data table request. Note that the table name parameter contains the name of the coupling facility data table and not the name of the file (unless they happen to be the same).

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

Changes to security

There are additional security checks introduced to control access to coupling facility data table servers and their pools of coupling facility data tables. These security checks are for:

- Controlling access to coupling facility data table pools
- Controlling AOR access to coupling facility data tables

Controlling access to coupling facility data table pools

You can control access to coupling facility data table pools using RACF® (or an equivalent external security manager).

Authorizing access to a coupling facility structure: Each coupling facility data table server region must be given access to its associated coupling facility list structure in accordance with the CFRM security rules. This means giving the user ID of the server region ALTER access to a FACILITY class general resource profile called **IXLSTR.***structure_name*. For example, use the RDEFINE command to define the profile, and the PERMIT command to grant access, as follows:

```
RDEFINE FACILITY IXLSTR.struct_name UACC(NONE)
PERMIT IXLSTR.struct_name CLASS(FACILITY) ID(server_userid) ACCESS(ALTER)
```

Authenticating servers: To enable the server to establish itself as a server for the given pool name, give the server region CONTROL access to a FACILITY class general resource profile called **DFHCF.***poolname*. For example, use the RDEFINE command to define the profile, and the PERMIT command to grant access, as follows:

```
RDEFINE FACILITY DFHCF.pool_name UACC(NONE)
PERMIT DFHCF.pool_name CLASS(FACILITY) ID(server_userid) ACCESS(CONTROL)
```

Authorizing CICS regions to access the server and its pool: To enable a CICS region to access a server and its pool, give the CICS region UPDATE access to the server's FACILITY class profile for the pool. For example, use the PERMIT command to grant access, as follows:

```
PERMIT DFHCF.pool_name CLASS(FACILITY) ID(CICS_region_userid) ACCESS(UPDATE)
```

Controlling AOR access to coupling facility data tables

In addition to controlling a CICS region's access to a coupling facility data table pool, you can optionally control access to each CFDT in the pool. This security check, if active, is performed by the server each time a CICS region connects to a data table for the first time. The resource security check is done as if for a CICS file owned by the coupling facility data table server region, using profiles defined in the default FCICSFCT class, with the table name as the name of the profile.

You can optionally prefix the profile name using the server region user ID as the prefix by specifying SECURITYPREFIX=YES as a server initialization parameter. You can customize the security class name and prefix for this level of security checking, using server initialization parameters SECURITYCLASS and SECURITYPREFIXID respectively.

The server performs the security check by issuing a cross-memory mode FASTAUTH check, which requires the use of global in-storage security profiles. If the external security manager does not support cross-memory mode FASTAUTH or global in-storage profiles, coupling facility data table security checks are not possible and an error message is issued at server initialization time if security checking is specified.

Resource security checking for files

Normal CICS resource security for files is unchanged and, if specified, CICS performs the usual file resource security checks against signed-on users of transactions that access coupling facility data tables.

Changes to CICS-supplied transactions

There are changes to the CICS-supplied transaction, CEMT, to support coupling facility data tables:

- There is a new CEMT INQUIRE CFDTPOOL command

- The CEMT INQUIRE FILE and CEMT SET FILE commands are changed to support files that refer to coupling facility data tables.

The CEMT INQUIRE CFDTPOOL command is the master terminal operator equivalent of the EXEC CICS INQUIRE CFDTPOOL SPI command. See “Changes to the system programming interface” on page 30 for more details.

The changes to the CEMT INQUIRE FILE and SET FILE commands are the same as those made to the equivalent SPI commands. That is, there are new options (CFDTPOOL, LOAD|NOLOAD, KEYLENGTH, TABLENAME, RECORDSIZE, and CONTENTION|LOCKING) added to support coupling facility data tables. See “Changed INQUIRE FILE command” on page 30 and “Changed SET FILE command” on page 32 for more details.

Changes to monitoring and statistics

There are changes to CICS performance class data and statistics for coupling facility data tables. See “Chapter 6. Monitoring, statistics, and enterprise management changes” on page 65 for details of these changes.

Changes to sample programs

The sample global user exit programs DFH$DTRD, DFH$DTLC, and DFH$DTAD are updated to work for coupling facility data tables.

Changes to problem determination

There are new messages and codes to aid problem determination, and enhancements to CICS dump and CICS trace.

Messages

There are new DFHFC*nnnn* messages for CICS coupling facility data table support, and DFHCF*nnnn* messages are introduced for use by the coupling facility data table server.

Abend codes

There are new abend codes that can occur in connection with coupling facility data table processing. These are:

ACFA The CFCL transaction has detected an abend, or an error response from a domain call, while loading a coupling facility data table, after which normal processing cannot continue.

ACFB A transaction issued a request for a coupling facility data table record for which it holds an active lock, but after the lock was acquired, the coupling facility data table server for the CFDT failed and was restarted. This request is of a type which is reliant on a lock that was acquired before the server failed; for example, a rewrite to a locking CFDT is dependent on the lock acquired by the previous read for update, and if the server has failed since the read for update, the rewrite cannot be allowed to proceed.

ACFC A transaction has issued a request to a coupling facility data table that was last accessed using a coupling facility data table server that has failed, and the server has been restarted one or more times since the last access.

Access between this CICS file and the CFDT, therefore, needs to be reopened before the request can be processed, but the reopen attempt has failed.

ACFD A call to the CICS transaction manager by the CFCL transaction during the loading of a coupling facility data table has received a response (such as DISASTER) after which normal processing could not continue. CICS issues message DFHFC7121, loading of the data table is terminated, and CFCL abends.

ACFE A transaction not internally attached by CICS has made an attempt to attach a transaction that specifies DFHFCDL as the program to be given control, which is not allowed. DFHFCDL is for use by the CICS system transaction, CFCL, to load a coupling facility data table.

New abend codes are added for cases where an application program fails to handle one of the new exceptional conditions:

AEZE CHANGED condition not handled

AEZN POOLERR condition not handled

Dump

The CICS IPCS dump formatting exit handles the new chain of connected pool elements, new fields in existing control blocks, and new control blocks associated with coupling facility data table support.

Trace

New trace points are added to write trace on entry to and exit from new modules that support coupling facility data tables, and before and after all calls to a coupling facility data table server.

The new trace point IDs used for coupling facility data tables are in the range 2440-24FF.

CICSPlex SM support

CICSPlex SM supports the file inquiry facility by providing a new view:

- CFDTPOOL, for coupling facility data table pools associated with a file. Associated with this general view is a detailed view, CFDTPOOD, and a summary view, CFDTPOOS.

The existing operations view, CMDT, and its associated detail view, CMDTD, and summary view, CMDTS, have been amended to reflect support for coupling facility data tables.

Additionally, there are two new detail views for CMDT:

- CMDT2, for detailed information relating to a CICS- or user-maintained data table, or a coupling facility data table. You can hyperlink to this view from the Table Info field of the CMDTD view.
- CMDT3, for statistical information relating to a data table file. You can hyperlink to this view from the Data Set Info field of the CMDT2 view.

The FILE OPERATE view is amended to reflect support of coupling facility data tables.

The BAS file definition view, FILEDEF, is extended to incorporate consideration for coupling facility data table files.

A new resource table, CFDTPOOL, is introduced.

Chapter 4. Dynamic routing of DPL and EXEC CICS START requests

The CICS dynamic routing facility is enhanced as follows:

- The dynamic routing interface is redesigned.
- The dynamic routing facility is extended to provide mechanisms for dynamically routing:
  - CICS-to-CICS distributed program link (DPL) requests.
  - Program-link requests received from outside CICS.
  - Transactions invoked by a subset of EXEC CICS START commands.
  - CICS business transaction services (BTS) processes and activities. (BTS is described in “Chapter 8. CICS business transaction services” on page 81.)

  For eligible requests:

  - A routing program is invoked, and can be used to select the target region
  - Workload balancing can be managed by CICSPlex SM.
- There is an enhanced method for statically routing transactions that are initiated by EXEC CICS START commands.

The chapter covers the following topics:

Changes to the dynamic routing interface

This section gives an overview of the enhanced dynamic routing interface.

First, some definitions are necessary:

Requesting region
: The region in which a transaction or program-link request is issued. Here are some examples of what is meant by “requesting region”:
  - For transactions started from terminals, it is the terminal-owning region (TOR).
  - For transactions started by EXEC CICS START commands, it is the region in which the START command is issued.
  - For traditional CICS-to-CICS DPL calls, it is the region in which the EXEC CICS LINK PROGRAM command is issued.
  - For program-link calls received from outside CICS—for example, ECI calls received from CICS Clients—it is the CICS region which receives the call.

Routing region
: The region in which the routing program runs. With one exception, the

requesting region and the routing region are always the same region. For terminal-related START commands only:

- Because the START command is always executed in the terminal-owning region (TOR), the requesting region and the routing region may or may not be the same. (This is more fully explained in “Routing transactions invoked by START commands” on page 46.)
- The routing region is always the TOR.

Target region
The region in which the routed transaction or program-link request executes.

Two routing models

There are two possible dynamic routing models:

- The hub model
- The distributed model.

The hub model

The hub is the model that has traditionally been used with CICS dynamic transaction routing. A routing program running in a TOR routes transactions between several AORs. Usually, the AORs (unless they are AOR/TORs) do no dynamic routing. Figure 7 shows a hub routing model.

Figure 7. Dynamic routing using a hub routing model. One routing region (the TOR) selects between several target regions.

The hub model applies to the routing of:

- Transactions started from terminals.
- Transactions started by terminal-related START commands.
- CICS-to-CICS DPL requests. (The requesting region acts as a hub because it routes requests among a set of AORs.)
- Program-link requests received from outside CICS. (The receiving region acts as a hub or TOR because it routes the requests among a set of back-end server regions.)

The hub model is a *hierarchical* system—routing is controlled by one region (the TOR); normally, a routing program runs only in the TOR.

Advantage of the hub model: It is a relatively simple model to implement. For example, compared to the distributed model, there are few inter-region connections to maintain.

Disadvantages of the hub model:

- If you use only one hub to route transactions and program-link requests across your AORs, the hub TOR is a single point-of-failure.
- If you use more than one hub to route transactions and program-link requests across the same set of AORs, you may have problems with distributed data. For example, if the routing program keeps a count of routed transactions for load-balancing purposes, each hub TOR will need access to this data.

The distributed model

In the distributed model, each region may be both a routing region and a target region. A routing program runs in each region. Figure 8 shows a distributed routing model.

Figure 8. Dynamic routing using a distributed routing model. Each region may be both a routing region and a target region.

The distributed model applies to the routing of:

- CICS business transaction services processes and activities
- Non-terminal-related START requests.

The distributed model is a *peer-to-peer* system—each participating CICS region may be both a routing region and a target region. A routing program runs in each region.

Advantage of the distributed model: There is no single point-of-failure.

Disadvantages of the distributed model:

- Compared with the hub model, there are a great many inter-region connections to maintain.
- You may have problems with distributed data. For example, any data used to make routing decisions must be available to all the regions. (CICSPlex SM solves this problem by using dataspaces and other mechanisms.)

Two routing programs

There are two CICS-supplied user-replaceable programs for dynamic routing:

The dynamic routing program, DFHDYP[1]
Can be used to route:
- Transactions started from terminals
- Transactions started by terminal-related START commands
- CICS-to-CICS DPL requests
- Program-link requests received from outside CICS.

The distributed routing program, DFHDSRP
Can be used to route:
- CICS business transaction services processes and activities
- Non-terminal-related START requests.

The two routing programs:

1. Are specified on separate system initialization parameters. You specify the name of your dynamic routing program on the DTRPGM system initialization parameter. You specify the name of your distributed routing program on the DSRTPGM system initialization parameter.
2. Are passed the same communications area. (Certain fields that are meaningful to one program are not meaningful to the other.)

1. In previous CICS releases, the dynamic routing program was known as the “dynamic *transaction* routing program”, because it could be used only to route transactions.

3. Are invoked at similar points—for example, for route selection, route selection error, and, optionally, at termination of the routed transaction or program-link request.

Together, these three factors give you a great deal of flexibility. You could, for example, do any of the following:

- Use different user-written programs for dynamic routing and distributed routing.
- Use the same user-written program for both dynamic routing and distributed routing.
- Use a user-written program for dynamic routing and the CICSPlex SM routing program for distributed routing, or vice versa.

It is worth noting two important differences between the dynamic and distributed routing programs:

1. The dynamic routing program is only invoked if the resource (the transaction or program) is defined as DYNAMIC(YES). The distributed routing program, on the other hand, is invoked (for eligible non-terminal-related START requests and BTS activities) even if the associated transaction is defined as DYNAMIC(NO); though it cannot route the request. What this means is that the distributed routing program is better able to monitor the effect of statically-routed transactions on the relative workloads of the target regions.
2. Because the dynamic routing program uses the hub routing model—one routing program controls access to resources on several target regions—*the routing program that is invoked at termination of a routed request is the same program that was invoked for route selection*.

   The distributed routing program, on the other hand, uses the distributed model, which is a “peer-to-peer” system; the routing program itself is distributed. *The routing program that is invoked at termination of a routed request is not the same program that was invoked for route selection*—it is the routing program on the target region.

Dynamic routing of DPL requests

Not all DPL requests can be dynamically routed. For a DPL request to be eligible for dynamic routing, the remote program must either:

- Be defined to the local system as DYNAMIC(YES), or
- Not be defined to the local system.

Note: If the program specified on an EXEC CICS LINK command is not currently defined, what happens next depends on whether program autoinstall is active:

- If program autoinstall is inactive, the dynamic routing program is invoked.
- If program autoinstall is active, the autoinstall user program is invoked. The dynamic routing program is then invoked only if the autoinstall user program:
  - Installs a program definition that specifies DYNAMIC(YES), or
  - Does not install a program definition.

As well as traditional CICS-to-CICS DPL calls initiated by EXEC CICS LINK PROGRAM commands, program-link requests received from outside CICS can also be dynamically routed. For example, all the following types of program-link request can be dynamically routed:

- Calls received from:
  - The CICS Web interface
  - The CICS Gateway for Java
- Calls from external CICS interface (EXCI) client programs
- External call interface (ECI) calls from any of the CICS Client workstation products
- Distributed Computing Environment (DCE) remote procedure calls (RPCs)
- ONC RPC calls.

A program-link request received from outside CICS can be dynamically routed by:
- Defining the program to CICS as DYNAMIC(YES)
- Coding your dynamic routing program to route the request.

How CICS obtains the transaction ID

A transaction identifier is always associated with each dynamic program-link request. CICS obtains the transaction ID using the following sequence:

1. From the TRANSID option on the LINK command
2. From the TRANSID option on the program definition
3. CSMI, the generic mirror transaction. This is the default if neither of the TRANSID options are specified.

If you write your own dynamic routing program, perhaps based on the CICS-supplied routing program DFHDYP, the transaction ID associated with the request may not be significant—you could, for example, code your program to route requests based simply on program name and available AORs.

However, if you use CICSPlex SM to route your program-link requests, the transaction ID becomes much more significant, because CICSPlex SM's routing logic is transaction-based. CICSPlex SM routes each DPL request according to the rules specified for its associated transaction.

Note: The CICSPlex SM system programmer can use the EYU9WRAM user-replaceable module to change the transaction ID associated with a DPL request.

When the dynamic routing program is invoked

For eligible program-link requests, the dynamic routing program is invoked at the following points:

- Before the linked-to program is executed, to either:
  - Obtain the SYSID of the region to which the link should be routed.

    Note: The address of the caller's communication area (COMMAREA) is passed to the routing program, which can therefore route requests by COMMAREA contents if this is appropriate.
  - Notify the routing program of a statically-routed request. This occurs if the program is defined as DYNAMIC(YES)—or is not defined—but the caller uses the SYSID option of the LINK command to specify the name of a remote server region explicitly.
- If an error occurs in route selection—for example, if the SYSID returned by the dynamic routing program is unavailable or unknown, or the link fails on the

specified target region—to provide an alternate SYSID. This process iterates until either the program-link is successful or the return code from the dynamic routing program is not equal to zero.

- After the link request has completed, if reinvocation was requested by the routing program.
- If an abend is detected after the link request has been shipped to the specified remote system, if reinvocation was requested by the routing program.

Routing transactions invoked by START commands

This section describes a new method of routing transactions that are invoked by EXEC CICS START commands. For convenience, we shall call the method described in this section the *enhanced* method. The enhanced method supersedes the traditional method described in the *CICS Intercommunication Guide*. Note, however, that the enhanced method cannot be used to route:

- Some transactions that are invoked by EXEC CICS START commands
- Transactions invoked by the trigger-level on a transient data queue.

In these cases, the traditional method must be used.

To specify that a transaction, if it is invoked by an EXEC CICS START command, is to be routed by the enhanced method described in this section, *define the transaction as ROUTABLE(YES) in the requesting region* (the region in which the START command is issued).

Advantages of the enhanced method

There are several advantages in using the enhanced method, where possible, rather than the traditional method:

Dynamic routing

Using the traditional method, you cannot route the started transaction dynamically. (For example, if the transaction on a terminal-related START command is defined as DYNAMIC(YES) in the terminal-owning region, your dynamic routing program is invoked for notification only—it cannot route the transaction.)

Using the enhanced method, you can route the started transaction dynamically.

Efficiency

Using the traditional method, a terminal-related START command issued in a TOR is function-shipped to the AOR that owns the transaction. The request is then shipped back again, for routing from the TOR.

Using the enhanced method, the two hops to the AOR and back are missed out. A START command issued in a TOR executes directly in the TOR, and the transaction is routed without delay.

Simplicity

Using the traditional method, when a terminal-related START command issued in a TOR is function-shipped to the AOR that owns the transaction the “terminal-not-known” condition may occur if the terminal is not defined in the AOR.

Using the enhanced method, because a START command issued in a TOR is not function-shipped to the AOR, the “terminal-not-known” condition does not

occur. The START command executes in the TOR directly, and the transaction is routed just as if it had been initiated from a terminal. If the terminal is not defined in the AOR, a definition is shipped from the TOR.

Terminal-related START commands

For a transaction invoked by a terminal-related START command to be eligible for the enhanced routing method, *all* of the following conditions must be met:

- The START command is a member of the subset of eligible START commands—that is, it meets all the following conditions:
  - The START command specifies the TERMID option, which names the principal facility of the task that issues the command. That is, the transaction to be started must be terminal-related, and associated with the principal facility of the starting task.
  - The principal facility of the task that issues the START command is *not* a surrogate Client virtual terminal.
  - The SYSID option of the START command does not specify the name of a remote region. (That is, the remote region on which the transaction is to be started must not be specified explicitly.)
- The requesting region, the TOR, and the target region are all CICS TS Release 3

  Note: The requesting region and the TOR may be the same region.
- The requesting region and the TOR (if they are different) are connected by either of the following:
  - An MRO link
  - An APPC parallel-session link.
- The TOR and the target region are connected by either of the following:
  - An MRO link.
  - An APPC single- or parallel-session link. If an APPC link is used, at least one of the following must be true:
    1. Terminal-initiated transaction routing has previously taken place over the link. (The terminal-initiated transaction routing enables the TOR to determine whether or not the target region is a CICS TS Release 3 system, and therefore eligible for enhanced routing.)
    2. CICSPlex SM is being used for routing.
- The transaction definition in the *requesting* region specifies ROUTABLE(YES).
- If the transaction is to be routed dynamically, the transaction definition in the TOR specifies DYNAMIC(YES).

  Important: When considering which START-initiated transactions are candidates for dynamic routing, you need to take particular care if the START command specifies any of the following options:
  - AT, AFTER, INTERVAL, or TIME (that is, there is a delay before the START is executed)
  - QUEUE
  - REQID
  - RTERMID
  - RTRANID

You need to understand how each of the options of the START command is being used; whether, for example, it affects the set of regions to which the transaction can be routed.

START commands issued in an AOR

If a terminal-related START command is issued in an AOR, it is function-shipped to the TOR that owns the terminal named in the TERMID option. The START executes in the TOR.

Static routing: The transaction definition in the AOR specifies ROUTABLE(YES). The transaction definition in the TOR specifies DYNAMIC(NO). The dynamic routing program is not invoked. If the transaction is eligible for enhanced routing, [2] it is routed to the AOR named in the REMOTESYSTEM option of the transaction definition in the TOR. If REMOTESYSTEM is not specified, the transaction executes locally, in the TOR.

Note: If the transaction is ineligible for enhanced routing, it is handled in the traditional way described in the *CICS Intercommunication Guide*—that is, CICS tries to route it back to the originating AOR for execution. If the REMOTESYSTEM option of the transaction definition in the TOR names a region other than the originating AOR, the request fails.

Figure 9 shows the requirements for using the enhanced method to statically route a transaction that is initiated by a terminal-related START command issued in an AOR.

Figure 9. Static routing of a terminal-related START command issued in an AOR, using the enhanced method. The requesting region, the TOR, and the target region are all CICS TS Release 3. The requesting region and the TOR are connected by an MRO or APPC parallel-session link. The TOR and the target region are connected by an MRO or APPC (single- or parallel-session) link. The transaction definition in the requesting region specifies ROUTABLE(YES). The transaction definition in the TOR specifies DYNAMIC(NO). The REMOTESYSTEM option names the AOR to which the transaction is to be routed.

Dynamic routing: The transaction definition in the AOR specifies ROUTABLE(YES). The transaction definition in the TOR specifies DYNAMIC(YES). The dynamic routing program is invoked in the TOR. If the transaction is eligible for enhanced routing, the routing program can reroute the transaction to an alternative AOR—that is, to an AOR other than that in which the START was issued.

Note: If the transaction is ineligible for enhanced routing, the dynamic routing program is invoked for notification only—it cannot reroute the transaction. The transaction is handled in the traditional way—that is, it is routed back to the originating AOR for execution.

Figure 10 on page 49 shows the requirements for dynamically routing a transaction that is initiated by a terminal-related START command issued in an AOR.

2. See the list of conditions on page 47.

Figure 10. Dynamic routing of a terminal-related START command issued in an AOR. The requesting region, the TOR, and the target region are all CICS TS Release 3. The requesting region and the TOR are connected by an MRO or APPC parallel-session link. The TOR and the target region are connected by an MRO or APPC (single- or parallel-session) link. The transaction definition in the requesting region specifies ROUTABLE(YES). The transaction definition in the TOR specifies DYNAMIC(YES).

START commands issued in a TOR

Static routing: The transaction definition in the TOR specifies ROUTABLE(YES) and DYNAMIC(NO). The dynamic routing program is not invoked. If the transaction is eligible for enhanced routing (see the list of conditions on page 47):

1. The START executes in the TOR.
2. The transaction is routed to the AOR named in the REMOTESYSYEM option of the transaction definition. If REMOTESYSTEM is not specified, the transaction executes locally, in the TOR.

Note: If the transaction is ineligible for enhanced routing, the START request is handled in the traditional way described in the *CICS Intercommunication Guide*—that is, it is function-shipped to the AOR named in the REMOTESYSTEM option of the transaction definition. If REMOTESYSTEM is not specified, the START executes locally, in the TOR.

Figure 11 shows the requirements for using the enhanced method to statically route a transaction that is initiated by a terminal-related START command issued in a TOR.

Figure 11. Static routing of a terminal-related START command issued in a TOR, using the enhanced method. The TOR and the target region are both CICS TS Release 3. The TOR and the target region are connected by an MRO or APPC (single- or parallel-session) link. The transaction definition in the TOR specifies DYNAMIC(NO) and ROUTABLE(YES). The REMOTESYSTEM option names the AOR to which the transaction is to be routed.

Dynamic routing: The transaction definition in the TOR specifies ROUTABLE(YES) and DYNAMIC(YES). The dynamic routing program is invoked. If the transaction is eligible for enhanced routing, the START is executed in the TOR, and the routing program can route the transaction.

Note: If the transaction is ineligible for enhanced routing, the dynamic routing program is invoked for notification only—it cannot route the transaction. The START request is handled in the traditional way—that is, it is function-shipped to the AOR named in the REMOTESYSTEM option of the transaction definition in the TOR. If REMOTESYSTEM is not specified, the START executes locally, in the TOR.

Figure 12 shows the requirements for dynamically routing a transaction that is initiated by a terminal-related START command issued in a TOR.

Figure 12. Dynamic routing of a terminal-related START command issued in a TOR. The TOR and the target region are both CICS TS Release 3. The TOR and the target region are connected by an MRO or APPC (single- or parallel-session) link. The transaction definition in the TOR specifies both DYNAMIC(YES) and ROUTABLE(YES).

Non-terminal-related START commands

For a non-terminal-related START request to be eligible for enhanced routing, *all* of the following conditions must be met:

- Both the requesting region and the target region are CICS Transaction Server for OS/390 Release 3.
- The requesting region and the target region are connected by either of the following:
  - An MRO link.
  - An APPC single- or parallel-session link. If an APPC link is used, at least one of the following must be true:
    1. Terminal-initiated transaction routing has previously taken place over the link. (The terminal-initiated transaction routing enables the requesting region to determine whether or not the target region is a CICS Transaction Server for OS/390 Release 3 system, and therefore eligible for enhanced routing.)
    2. CICSPlex SM is being used for routing.
- The transaction definition in the requesting region specifies ROUTABLE(YES).

In addition, if the request is to be routed dynamically:

- The transaction definition in the requesting region must specify DYNAMIC(YES).
- The SYSID option of the START command must not specify the name of a remote region. (That is, the remote region on which the transaction is to be started must not be specified explicitly.)

Important: When considering which START requests are candidates for dynamic routing, you need to take particular care if the START specifies any of the following options:

- AT, AFTER, INTERVAL(non-zero), or TIME. That is, there is a delay before the START is executed.

  If there is a delay, the interval control element (ICE) created by the START request is kept in the requesting region with a transaction ID of CDFS. The CDFS transaction retrieves any data specified by the user and reissues the START request without an interval. The request is routed when the ICE expires, based on the state of the transaction definition and the sysplex at that moment.
- QUEUE.
- REQID.
- RTERMID.
- RTRANID.

You need to understand how these options are being used; whether, for example, they affect the set of regions to which the request can be routed.

Static routing

The transaction definition in the requesting region specifies ROUTABLE(YES) and DYNAMIC(NO). If the START request is eligible for enhanced routing (see the list of conditions on page 50), the distributed routing program—that is, the program specified on the DSRTPGM system initialization parameter—is invoked for notification of the statically-routed request.

Notes:

1. The distributed routing program differs from the dynamic routing program, in that it is invoked—for eligible non-terminal-related START requests—even when the transaction is defined as DYNAMIC(NO). The dynamic routing program is never invoked for transactions defined as DYNAMIC(NO). This difference in design means that you can use the distributed routing program to assess the effect of statically-routed requests on the overall workload.
2. If the request is ineligible for enhanced routing, the distributed routing program is not invoked.

Dynamic routing

The transaction definition in the requesting region specifies ROUTABLE(YES) and DYNAMIC(YES). If the request is eligible for enhanced routing, the distributed routing program is invoked for routing. The START request is function-shipped to the target region returned by the routing program.

Notes:

1. If the request is ineligible for enhanced routing, the distributed routing program is not invoked. Unless the SYSID option specifies a remote region explicitly, the START request is function-shipped to the AOR named in the REMOTESYSTEM option of the transaction definition in the requesting region; if REMOTESYSTEM is not specified, the START executes locally, in the requesting region.
2. If the request is eligible for enhanced routing, but the SYSID option of the START command names a remote region, the distributed routing program is invoked for notification only—it cannot route the request. The START executes on the remote region named on the SYSID option.

Benefits

Here are some of the benefits of the new routing functions:

- The ability to route all types of program link request dynamically can be used to improve the performance and reliability of:
  - The CICS Web interface
  - The CICS Gateway for Java
  - EXCI calls
  - CICS Client ECI calls
  - DCE/RPC
  - ONC RPC
  - Any function that issues an EXEC CICS LINK PROGRAM request.
- The ability to route a subset of EXEC CICS START commands dynamically can be used to improve the performance and reliability of applications that use those commands.
- Using CICSPlex SM, you could, for example, integrate workload balancing for terminal-initiated transactions, EXCI clients, CICS Clients, and started tasks.

Requirements

For dynamic routing of DPL requests, no special hardware or software is required.

For dynamic routing of transactions started by terminal-related START requests, the requesting region, the TOR, and the target region must be at the CICS TS Release 3 level.

For dynamic routing of transactions started by non-terminal-related START requests, the requesting region and the target region must both be at the CICS TS Release 3 level.

Changes to CICS externals

This section gives an overview, in the following topics, of the changes to CICS externals introduced by dynamic routing of DPL and START requests:

Changes to system definition

A new system initialization parameter, DSRTPGM, is introduced; it names the distributed routing program.

Table 1. The DSRTPGM system initialization parameter

	DFHSIT	[TYPE={**CSECT**|DSECT}] [,DSRTPGM={**NONE**|DFHDSRP|*program-name*|EYU9XLOP}] : :

DSRTPGM={NONE|DFHDSRP|*program-name*|EYU9XLOP}
Specifies the name of the distributed routing program to be used for dynamically routing:

- Eligible CICS business transaction services (BTS) processes and activities
- Eligible non-terminal-related EXEC CICS START requests.

DFHDSRP
The CICS sample distributed routing program.

EYU9XLOP
The CICSPlex SM routing program.

NONE
For eligible BTS processes and activities, no routing program is invoked. BTS processes and activities cannot be dynamically routed.

For eligible non-terminal-related START requests, the CICS sample distributed routing program, DFHDSRP, is invoked.

program-name
The name of a user-written program.

Note: See also the DTRPGM parameter, used to name the dynamic routing program.

Changes to resource definition

The following RDO commands have been modified:

CEDA DEFINE PROGRAM
A new DYNAMIC option is added. It specifies whether, if the program is the subject of a program-link request, the request can be dynamically routed.

CEDA DEFINE TRANSACTION
A new ROUTABLE option is added. It specifies whether, if the transaction is the subject of an eligible EXEC CICS START command, it can be dynamically routed. This option applies to transaction definitions in regions in which START commands are issued.

Changes to system programming

The following system programming interface (SPI) commands have been modified:

EXEC CICS INQUIRE PROGRAM
A new DYNAMSTATUS option is added. This returns a CVDA value indicating whether, if the program is the subject of a program-link request, the request can be dynamically routed.

EXEC CICS INQUIRE SYSTEM
A new DSRTPROGRAM option is added. This returns the name of the distributed routing program currently identified to the system.

EXEC CICS INQUIRE TRANSACTION
A new ROUTESTATUS option is added. This returns a CVDA value indicating whether, if the transaction is the subject of an eligible EXEC CICS START command, it can be dynamically routed.

EXEC CICS SET SYSTEM
A new DSRTPROGRAM option is added. This specifies the name of the distributed routing program.

Changes to CICS-supplied transactions

There are changes to the CICS master terminal transaction, CEMT. The following CEMT commands have been modified:

CEMT INQUIRE PROGRAM
A new DYNAMSTATUS option is added. This indicates whether, if the program is the subject of a program-link request, the request can be dynamically routed.

CEMT INQUIRE SYSTEM
A new DSRTPROGRAM option is added. This displays the name of the distributed routing program currently identified to the system.

CEMT INQUIRE TRANSACTION
A new ROUTSTATUS option is added. This indicates whether, if the transaction is the subject of an eligible EXEC CICS START command, it can be dynamically routed.

CEMT SET SYSTEM
A new DSRTPROGRAM option is added. This specifies the name of the distributed routing program.

Changes to user-replaceable programs

A new user-replaceable program, DFHDSRP, is introduced. This is the default distributed routing program, which handles the dynamic routing of transactions:
- That implement BTS activities
- That are started by non-terminal-related START commands.

To support the new dynamic routing functions, several new fields are added to the DFHDYPDS communication area. This communication area is now passed to the distributed routing program as well as to the dynamic routing program.

Changes to the exit programming interface (XPI)

The following exit programming interface (XPI) commands have been modified:

INQUIRE_CURRENT_PROGRAM
A new DYNAMIC_STATUS option is added. This returns a value indicating whether, if the program that is currently running is the subject of a program-link request, the request can be dynamically routed.

INQUIRE_PROGRAM
A new DYNAMIC_STATUS option is added. This returns a value indicating whether, if the program is the subject of a program-link request, the request can be dynamically routed.

INQUIRE_TRANDEF
A new ROUTABLE_STATUS option is added. This returns a value indicating whether, if the transaction is the subject of an eligible EXEC CICS START command, it can be dynamically routed.

Changes to sample programs

There is a new CICS-supplied program—the default distributed routing program, DFHDSRP.

The existing default dynamic routing program, DFHDYP, is updated to handle its revised communication area.

Changes to monitoring and statistics

A new monitoring field, 073, is added to the DFHPROG group in performance class monitoring records. It contains the number of DPL requests issued by the user task.

A new statistics field, TCSESTPC, is added to the TCT system entry; it contains the number of DPL requests that have been successfully routed.

Changes to trace points

A number of new trace points are introduced. These are described in the *CICS User's Handbook*.

Changes to messages and abend codes

A number of new messages and abend codes are introduced, and some existing messages are modified.

All new and changed messages and abend codes are described in the *CICS Messages and Codes* manual.

CICSPlex SM support

CICSPlex SM dynamic routing program EYU9XLOP is extended to include the new request types. You can integrate all the new request types into workload management. CICSPlex SM support is provided by:

- A new view, WLMAWAOS, which shows summarized information about all routing regions that are associated with a workload that is within the CICSplex identified as the context. WLMAWTOS is a summary form of the WLMAWTOR view.
- A new view, WLMAWTOS, which shows summarized information about all target regions that are associated with a workload that is within the CICSplex identified as the context. WLMAWAOS is a summary form of the WLMAWTOR view.
- A new field **Routstatus** on the LOCTRAND view. This field indicates whether or not the current transaction is eligible for dynamic routing.
- A new field **Dynam Status** on the PROGRAMD view. This field indicates whether or not the current program is eligible for dynamic routing.
- A new field **Dynamic** on the PROGDEF view now specifies whether or not an EXEC CICS LINK to the named program may invoke dynamic routing.

Part 3. System management

This Part describes the new function designed to improve and simplify the task of CICS system management. It covers the following topics:

Chapter 5. Resource definition online for CICS temporary storage

This chapter describes resource definition online (RDO) for CICS temporary storage queues. It covers the following topics:

Overview

Continuing its policy of removing the need for macro-defined control tables, CICS extends support for resource definition online (RDO) by enabling resource definitions for temporary storage tables (TSTs) to be defined in the CSD. The new resource definitions support all those functions provided by the DFHTST macros, plus a small number of extra functions intended to improve usability.

All the function provided by the DFHTST macros is supported by a single new resource definition type, the TSMODEL resource definition. The TSMODEL resource definition and its attributes provide the function currently available in the following DFHTST macro definitions:

```
DFHTST TYPE=INITIAL
DFHTST TYPE=RECOVERY
DFHTST TYPE=SECURITY
DFHTST TYPE=REMOTE
DFHTST TYPE=LOCAL
DFHTST TYPE=SHARED
```

The TSMODEL enables you to specify a TS queue name prefix, and associate all the required attributes with that prefix. The TSMODEL also enables you to map TS queue names directly to a shared TS pool, without the need to specify a DFHTST TYPE=SHARED macro.

Benefits

Providing RDO for temporary storage queues extends CICS support for continuous availability, making it unnecessary to stop and restart CICS regions in order to introduce new or changed definitions.

It also simplifies system management, RDO making it much simpler to implement and maintain resource definitions.

Requirements

There are no specific hardware or software requirements for RDO support for TSMODELs.

Changes to CICS externals

There are changes to a number of CICS externals as a result of adding RDO support for TS queues. These are:

- "Changes to resource definition"
- "Changes to the system programming interface" on page 61
- "Changes to CICS-supplied transactions" on page 62
- "Changes to global user-exits" on page 62

Changes to resource definition

Adding TSMODEL resource definitions affects both macro and RDO support.

Changes to DFHTST macros

There is a change to the DFHTST TYPE=INITIAL macro, with the addition of the MIGRATE=YES option. This enables you to migrate your temporary storage control table definitions to the CSD as TSMODEL definitions.

The MIGRATE option causes a flag to be set in the assembled table to indicate that the TST can be used for migration purposes. The flag serves two purposes:

- If you are migrating a TST to the CSD, the CSD utility program, DFHCSDUP, tests whether this flag is set in the table load module. If it is not, the MIGRATE command fails.
- If you are bringing up a CICS region with a TST specified that has the MIGRATE flag set, the use of the TST is restricted by CICS to the TSAGE option on the TYPE=INITIAL macro, and to TYPE=SHARED entries only. For all other purposes, CICS uses TSMODELs only.

Changes to RDO

The CSD DEFINE command is extended to enable you to define TSMODEL resource definitions in the CSD. You can use the DEFINE TSMODEL command with either the CEDA or CEDB RDO transactions, or with the DFHCSDUP utility program.

If you are using RDO, the CEDA DEFINE TSMODEL panel is as shown in Figure 13 on page 61.

```
OBJECT CHARACTERISTICS                                  CICS RELEASE = 0530
 CEDA  View TSmodel( testtsq  )
  TSmodel        : testtsq
  Group          : TEST
  Description    :
  PRefix         :
  Location       : Auxiliary          Auxiliary | Main
 RECOVERY ATTRIBUTES
  RECovery       : No                 No | Yes
 SECURITY ATTRIBUTES
  Security       : No                 No | Yes
 SHARED ATTRIBUTES
  POolname       :
 REMOTE ATTRIBUTES
  REMOTESystem   :
  REMOTEPrefix   :




                                                SYSID=HT61 APPLID=CICSHT61

PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 13. The CEDA DEFINE panel for the TSMODEL resource definition

You can define as many TSMODEL resource definitions as you need, specifying appropriate prefixes to match the TS queue names used by your applications.

Changes to the system programming interface

New and changed EXEC CICS commands support the use of TSMODEL resource definitions:

- The EXEC CICS INQUIRE TSQUEUE command is enhanced with the addition of two new options:
  - POOLNAME is added as an alternative to the SYSID option.
  - RECOVSTATUS is added to return a CVDA to indicate the recoverability of the specified queue. The CVDA values are RECOVERABLE and NOTRECOVERABLE.
- There are five new SPI commands:
  - EXEC CICS CREATE TSMODEL. This command enables you to define and install the specified TSMODEL resource definition.
  - EXEC CICS DISCARD TSMODEL. This command enables you to discard from the CICS region the specified TSMODEL resource definition.
  - EXEC CICS INQUIRE TSMODEL. This command returns information about the specified TSMODEL resource definition.
  - EXEC CICS INQUIRE TSPOOL. This command returns information about the specified TSPOOL resource definition.
  - EXEC CICS SET TSQUEUE. This command enables you to delete a specified TSQUEUE. You can use the LASTUSEDINT option to ensure that the queue to be deleted has not been referenced within a given interval.

For details of all these commands, see the *CICS System Programming Reference* manual.

Changes to CICS-supplied transactions

New and changed CEMT commands support the creation of TSMODEL resource definitions dynamically.

- The CEMT INQUIRE TSQUEUE COMMAND is enhanced with the addition of the ACTION and RECOVSTATUS options.
- There are four new CEMT commands:
  - CEMT DISCARD TSMODEL. This command enables you to discard from the CICS region the specified TSMODEL resource definition.
  - CEMT INQUIRE TSPOOL. This command returns information about the specified TSPOOL resource definition.
  - CEMT INQUIRE TSMODEL. This command returns information about the specified TSMODEL resource definition.
  - CEMT SET TSQUEUE. This command enables you to delete a temporary storage queue (TS queue). The LASTUSEDINT option may be used to ensure that the queue to be deleted has not been referenced since a previous INQUIRE was issued. It may also be used to delete queues which have not been referenced within a given interval.

For details of all these commands, see the *CICS Supplied Transactions* manual.

Changes to global user-exits

The XRSINDI global user exit is invoked when TSMODEL resource types are installed or discarded.

CICSPlex SM support

CICSPlex SM support for temporary storage queue models is provided by:

- A new BAS definition object TSMDEF, which allows you to specify a temporary storage queue name prefix and associate attributes with that name.
- New operate views:

  TSMODEL General view of temporary storage queue models.

  TSMODELD Detailed view of a temporary storage queue model.

  TSMODELS Summary view of temporary storage queue models.

  TSPOOL General view of temporary storage shared pools.

  TSQSHR General view of shared temporary storage queues.

  TSQSHRD Detailed view of a shared temporary storage queue.

  TSQSHRS Summary view of shared temporary storage queues.
- A Delete TSQ function, that allows you to delete temporary storage queues from the TSQ view. A TSQ Deletion panel asks you to confirm the deletion.

  Note: The existing temporary storage operate views (TSQ, TSQS, TSQGBL, and TSQGBLS) remain unchanged.
- New resource tables:
  - CRESTSMD
  - ERMCTSMD
  - TSMDEF
  - TSMINGRP
  - TSMODEL
  - TSPOOL
  - TSQSHR

Chapter 6. Monitoring, statistics, and enterprise management changes

This chapter describes the monitoring, statistics, and enterprise management changes. It contains the following topics:

Overview

There are numerous changes and additions to CICS monitoring data and statistics:

- A number of additions and improvements have been made to both the performance and exception class records. They are aimed primarily at improving the way that the monitoring data can be used for offline performance analysis and tuning.
- There are new CICS statistics for the TCP/IP service resources in support of the CSD TCPIPSERVICE resource definition.
- The statistics utility program (DFHSTUP) is enhanced to provide reports for the new TCP/IP service statistics data.
- The statistics sample program (DFH0STAT) is enhanced to provide reports for the new TCP/IP service statistics data as well as enhancements to some existing reports in support of new function provided in this release.
- New CICS SMF Type 110 statistics records are provided for the coupling facility data table server (subtype 4) and the named counter server (subtype 5).

  Note: These statistics records are not printed by the statistics utility program (DFHSTUP), they are only printed by the server jobs.

Enterprise management has been enhanced with support for Tivoli Global Enterprise Manager.

Monitoring

Additions and changes to CICS monitoring data are summarized in the following sections.

Additions and changes to monitoring data

There are many new performance class data fields in support of the following:

- CICS business transactions services
- Open transaction environment
- Java Virtual Machine (JVM)
- Coupling facility data tables
- Dynamic routing of DPL and EXEC CICS START requests
- CICS Web interface enhancements
- 3270 bridge enhancements
- Secure sockets layer
- MVS resource recovery services for EXCI

- Global enqueue and dequeue
- Dynamic routing enhancements
- Object-oriented interface to CICS services for C++.

Revised 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 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
- 456 bytes for a TOR
- 92 bytes for an AOR
- 496 bytes for an FOR.

There are additions to the exception class data in support of the following:
- MVS resource recovery services for EXCI
- Long temporary storage queue names.

Statistics

There are additional statistics for the new TCP/IP service resources, known by the resource type name TCPIPSERVICE, and mapped by a new copybook, DFHSORDS.

There are changes to a number of CICS statistics copybooks to provide additional information about resources. The changed copybooks are:
- Connection statistics (DFHA14DS)
- Dispatcher statistics (DFHDSGDS)
- Enqueue statistics (DFHNQGDS)
- File statistics (DFHA17DS)
- Transient data queue statistics (DFHTQRDS)

For more information, see the *CICS Performance Guide*.

Changes to CICS externals

The general changes to CICS monitoring and statistics result in a number of changes to CICS externals:
- “Changes to the system programming interface”
- “Changes to CICS-supplied transactions” on page 68
- “Changes to sample programs” on page 68
- “Changes to utility programs” on page 68
- “Changes to monitoring data” on page 69.

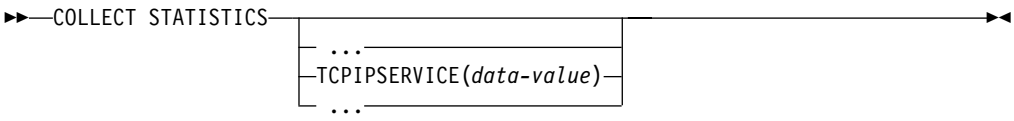
Changes to the system programming interface

The CICS system programming interface (SPI) is enhanced for monitoring and statistics with additional options on the EXEC CICS COLLECT STATISTICS and the EXEC CICS PERFORM STATISTICS RECORD commands.

EXEC CICS COLLECT STATISTICS

The EXEC CICS COLLECT STATISTICS command returns the current statistics for a named resource or resource type.

COLLECT STATISTICS



You can request only specific statistics for the TCPIPSERVICE resource type, using the option TCPIPSERVICE(*data-value*), where *data-value* is the 8-character name of the TCP/IP service.

Copybooks are provided in ASSEMBLER, COBOL, and PL/I, that map the returned statistics. The copybooks are supplied in the following libraries:

ASSEMBLER	CICSTS13.CICS.SDFHMAC
COBOL	CICSTS13.CICS.SDFHCOB
PL/1	CICSTS13.CICS.SDFHPL1

A full list of the statistics data copybooks that you can use is provided in the *CICS Performance Guide*. The new copybook for the TCP/IP service statistics is DFHSORDS.

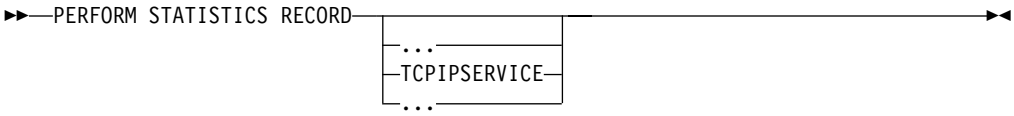
COLLECT STATISTICS exception conditions

There are no new exception conditions for the COLLECT STATISTICS command. The existing IOERR, NOTAUTH, and NOTFND can occur for the new TCPIPSERVICE resource names.

EXEC CICS PERFORM STATISTICS RECORD

The PERFORM STATISTICS RECORD command causes the statistics for a named resource to be written immediately to the SMF data set. The TCPIPSERVICE resource type is added to record the TCP/IP service resource statistics.

PERFORM STATISTICS RECORD



Specify TCPIPSERVICE for the TCP/IP service resource statistics to be written immediately to the SMF data set.

PERFORM STATISTICS exception conditions

There are no new exception conditions for the PERFORM STATISTICS command. The existing IOERR, NOTAUTH, and NOTFND can occur for the new TCPIPSERVICE resource name.

Changes to CICS-supplied transactions

The TCPIPSERVICE option is added to the CEMT PERFORM STATISTICS RECORD command. This option causes TCP/IP service resource statistics to be written immediately to the SMF data set.

Changes to sample programs

There are changes to the existing monitoring and statistics sample programs, DFH£MOLS and DFH0STAT.

DFH£MOLS sample monitoring program

DFH£MOLS is enhanced to:

- Handle SMF 110 monitoring data records for CICS TS Releases 1, 2, and 3, in addition to CICS/ESA® Version 3 and Version 4.
- Unload performance class records into a flat file (CICS TS (Releases 1, 2 and 3) and CICS/ESA Version 4 performance class records only).
- Extend record selection by the addition of two new options to the SELECT and IGNORE control statements. These options are TASKNO and PRCSTYPE, for task numbers and CICS BTS process types, respectively.
- Print the new data from the exception data record
- Provide a new control statement to control the conversion of the monitoring timestamp data fields to local time.

DFH0STAT sample statistics program

DFH0STAT, the statistics sample program, is enhanced to produce the following additional statistics reports:

- TCP/IP services
- Temporary storage queues, by shared TS pool
- Coupling facility data table pools

Various improvements are made to the following statistics reports:

- System status
- Transaction manager and dispatcher
- Files and data tables
- Enqueue
- Exit programs
- Currently loaded programs by DSA and LPA
- Connections and modenames
- Program totals.

Changes to utility programs

DFHSTUP, the statistics utility program, is enhanced to

- Produce additional statistics reports for TCP/IP services
- Produce improved statistics reports for:
  - Connections
  - Dispatcher
  - Enqueue
  - Files
  - Transient data queues

Changes to monitoring data

The following sections describe the changes to monitoring data fields.

Additional performance class data fields

The following table shows the additional performance class data fields that are added in this release.

Table 2. Additional performance class data fields

Group Name	Field-Id	Description
DFHCBTS	200	CICS BTS process name
DFHCBTS	201	CICS BTS process type
DFHCBTS	202	CICS BTS process id
DFHCBTS	203	CICS BTS activity id
DFHCBTS	204	CICS BTS activity name
DFHCBTS	205	CICS BTS run process/activity synchronous count
DFHCBTS	206	CICS BTS run process/activity asynchronous count
DFHCBTS	207	CICS BTS link process/activity count
DFHCBTS	208	CICS BTS define process count
DFHCBTS	209	CICS BTS define activity count
DFHCBTS	210	CICS BTS reset process/activity count
DFHCBTS	211	CICS BTS suspend process/activity count
DFHCBTS	212	CICS BTS resume process/activity count
DFHCBTS	213	CICS BTS delete activity, or cancel process/activity, request count
DFHCBTS	214	CICS BTS acquire process/activity request count
DFHCBTS	215	CICS BTS total process/activity request count
DFHCBTS	216	CICS BTS delete/get/put process container count
DFHCBTS	217	CICS BTS delete/get/put activity container count
DFHCBTS	218	CICS BTS total process/activity container request count
DFHCBTS	219	CICS BTS retrieve reattach request count
DFHCBTS	220	CICS BTS define input event request count
DFHCBTS	221	CICS BTS timer associated event request count
DFHCBTS	222	CICS BTS total event related request count
DFHCICS	25	CICS OO foundation class request count
DFHDATA	179	IMS (DBCTL) request count
DFHDATA	180	DB2 request count
DFHDATA	186	IMS (DBCTL) wait time
DFHDATA	187	DB2 Readyq wait time
DFHDATA	188	DB2 Connection wait time
DFHDATA	189	DB2 wait time
DFHDOCH	226	Document handler Create count
DFHDOCH	227	Document handler Insert count
DFHDOCH	228	Document handler Set count

Table 2. Additional performance class data fields (continued)

Group Name	Field-Id	Description
DFHDOCH	229	Document handler Retrieve count
DFHDOCH	230	Document handler Total count
DFHDOCH	240	Document handler total created document length
DFHFILE	176	CFDT I/O wait time
DFHPROG	73	Program DPL count
DFHSOCK	241	Socket (SO) I/O wait time
DFHSOCK	242	Bytes encrypted for secure socket
DFHSOCK	243	Bytes decrypted for secure socket
DFHSOCK	244	Client IP address
DFHSYNC	177	CFDT server syncpoint wait time
DFHSYNC	196	Syncpoint delay time
DFHTASK	82	Transaction group ID
DFHTASK	123	Task global ENQ delay time
DFHTASK	190	RRMS/MVS unit-of-recovery id (URID)
DFHTASK	191	RRMS/MVS wait time
DFHTASK	195	CICS BTS run process/activity synchronous wait time
DFHTASK	248	CICS TCB change modes
DFHTASK	249	User-task QR TCB wait-for-dispatch time
DFHTASK	250	CICS MAXOPENTCBS delay time
DFHTASK	251	CICS TCB attach count
DFHTASK	253	CICS JVM elapsed time
DFHTASK	254	CICS JVM suspend time
DFHTASK	255	User-task QR TCB dispatch time
DFHTASK	256	User-task QR TCB CPU time
DFHTASK	257	User-task MS TCB dispatch time
DFHTASK	258	User-task MS TCB CPU time
DFHTASK	259	User-task L8 TCB CPU time
DFHTASK	260	User-task J8 TCB CPU time
DFHTASK	261	User-task S8 TCB CPU time
DFHWEBB	231	WEB Receive request count
DFHWEBB	232	WEB Characters received
DFHWEBB	233	WEB Send request count
DFHWEBB	234	WEB Characters sent
DFHWEBB	235	WEB Total request count
DFHWEBB	236	WEB Repository read request count
DFHWEBB	237	WEB Repository write request count

Changed performance class data fields

The following table shows the changed performance class data fields.

Table 3. Changed performance class data fields

Group Name	Field-Id	Description
DFHCICS	130	Transaction routing sysid
DFHPROG	71	Initial program name
DFHTASK	124	3270 Bridge transaction ID
DFHTASK	129	Task local ENQ delay time

Additional exception records

There are two new exceptional records produced. These are written after the following conditions have been resolved:

- A transaction wait for coupling facility data table locking request slot
- A transaction wait for coupling facility data table non-locking request slot

New exception records data fields

The fixed format of exception records is extended with the addition of the following three fields:

EXCMNURI (TYPE-C, 16 BYTES)
RRMS/MVS unit-of-recovery ID (URID).

EXCMNRIL (TYPE-A, 4 BYTES)
Exception resource ID length.

EXCMNRIX (TYPE-C, 256 BYTES)
Exception resource ID (extended).

For full details of CICS montoring data that is affected by the changes, see “Appendix. Details of changed monitoring records” on page 215.

Support for Tivoli Global Enterprise Manager

Tivoli Global Enterprise Manager (GEM) provides a consistent systems management view of an enterprise. Most enterprises have a set of business objectives. Those business objectives can be achieved using various business systems. A business system might comprise various hardware and software packages from multiple vendors, as well as user-written applications. One or more of the applications within a business system can make use of various middleware components such as database, messaging services, or a transaction monitor.

If a business system uses CICS as part of a CICSplex, Tivoli GEM instrumentation is provided for CICSPlex SM systems management.

The instrumentation for CICSPlex SM provided by Tivoli GEM enables an individual using the Tivoli GEM client to view status information for the CMAS environment and the MAS environment of the CICSplex. The CMAS and MAS environments, when grouped together, form the transaction system component at the middleware layer of a business system.

The instrumentation always discovers the status of the CMAS components without any changes to the existing CICSPlex SM environment. This feature provides the immediate benefit of a graphical status display of the CMAS subsystem for the GEM client user.

The instrumentation also automatically discovers the MAS environment within the CICSplex. The instrumentation discovers any MAS which makes use of the RTA MRM and/or SAM functions.

The benefit of discovering only those MASs that use RTA and SAM is that the user has the ability to have only a subset of the CICSplex discovered. This method of discovery can be helpful in reporting a specific subset of MASs, such as those supporting an enterprise's e-mail application, where some of the CICS regions are used as e-mail servers. If the e-mail application had any RTADEFs or STATDEFs analyzing its operation, these would also be discovered and displayed at the client. This would provide the client user at a help desk or operations center with important status indicators for the CICS layer of the e-mail system.

For details of CICSPlex SM instrumentation in Tivoli GEM, see the *Tivoli GEM CICSPlex SM Instrumentation Guide User's Guide*, GC31-5156.

Chapter 7. Autoinstall for MVS consoles

This chapter describes enhancements to CICS autoinstall for terminals, which is extended to include MVS consoles. It covers the following topics:

Overview

The range of devices supported by the CICS autoinstall for terminals function is extended to include MVS consoles.

Commands issued at an MVS console (or in a job stream) can be directed to a CICS region running as a started task, or job, using the MVS MODIFY command. Before CICS can accept the MVS command, it needs an entry in the terminal control table for the console issuing the command, so that the entry can be used by CICS terminal control to direct the response to the console.

Using pre-installed console definitions

When MVS receives your request, it identifies the CICS region from the task or job name, and passes your request to CICS. CICS extracts the console's name from the MODIFY data, and searches the terminal control table (TCT) for a CONSNAME entry that matches the console name. If CICS cannot find a matching name, but the console has a migration ID (range 100 - 250) or a non-extended ID (range 0-99), CICS tries to find a matching CONSOLE ID. If CICS finds a matching entry, it starts the transaction you specified on the MODIFY command, and the transaction can send the results to the console using the termid of the console's entry in the terminal control table.

Using autoinstalled console definitions

If CICS fails to find a matching entry, it checks the autoinstall status for consoles to determine whether it can perform an autoinstall for the console.

If autoinstall for consoles is active, CICS searches the autoinstall model table (AMT) and initiates the autoinstall process for the console. CICS either:

- Passes the suitable autoinstall model definitions to the autoinstall control program, together with information about the console, or
- Automatically uses the first console model definition it finds, or a console model with the same name as the console, and autoinstalls the console using a CICS-generated termid, without calling your autoinstall control program.

Which of these options CICS takes is determined by the autoinstall status for consoles. The autoinstall status for consoles is either set at startup by the AICONS system initialization parameter, or dynamically by a CEMT (or EXEC CICS) SET AUTOINSTALL CONSOLES command.

The terminal autoinstall control program

You use the same autoinstall control program for console autoinstall as for VTAM® terminals and APPC connections.

If the autoinstall control program is invoked (either the CICS-supplied program or your own) it selects one of the models and provides the rest of the information necessary to complete a TCT terminal entry for the console. When the autoinstall control program returns control, CICS builds a terminal control table terminal entry (TCTTE) for the console using the autoinstall model, the termid and other data returned by the autoinstall control program, and MVS data for the console. CICS then adds the new entry to the TCT and starts the transaction specified on the MODIFY command.

Preset security for autoinstalled consoles

If the model terminal specifies USERID(*FIRST) or USERID(*EVERY), CICS uses the user ID passed by MVS on the MODIFY command to sign on the console, in effect using the MVS-passed user ID as the preset user ID for the new console.

See “Changes to resource definition” on page 75 for details.

Automatic deletion of autoinstalled consoles

CICS automatically deletes autoinstalled consoles if they are unused for a specified delay period (the default is 60 minutes). For the install function, the autoinstall control program can set a ‘delete-delay’ value for the console. The delete-delay period is the length of time (in minutes, since the autoinstalled console was last used) that a console remains installed before CICS deletes it. Setting this value to 0 inhibits automatic deletion. Autoinstalled consoles are not recorded on the catalog and not recovered at restart. Note that a console is deleted even if there is a currently signed-on user.

Benefits

With multiple MVS console support, and CICS console support for TSO users, CICS regions often need to support many console devices, with numbers ranging from tens to hundreds. Enabling CICS to autoinstall consoles saves a considerable amount of system programmer effort in defining and maintaining individual console resource definitions.

Requirements

There are no dependencies on other software products, but you need a terminal autoinstall control program that supports autoinstall for consoles. See “Changes to the user replaceable modules” on page 78.

Changes to CICS externals

There are changes in a number of CICS externals to support the introduction of autoinstall support for MVS consoles. These are:

- "Changes to system definition"
- "Changes to resource definition"
- "Changes to the system programming interface" on page 76
- "Changes to CICS-supplied transactions" on page 78
- "Changes to the user replaceable modules" on page 78

Changes to system definition

A new system initialization parameter is added to support autoinstall for consoles:

AICONS={NO|YES|AUTO**}**
Specifies whether you want CICS to autoinstall console devices that are not predefined by explicit TERMINAL and TYPETERM definitions.

NO This is the default, and specifies that CICS is not to autoinstall consoles for which there are no predefined entries in the terminal control table.

YES Specifies that autoinstall for consoles is active, and CICS is to autoinstall an undefined console that issues a MODIFY command to the CICS region. CICS calls the autoinstall control program to supply the required information.

AUTO Specifies that autoinstall for consoles is active, but CICS is not to call the autoinstall control program. CICS is to perform the autoinstall without any input from the autoinstall control program, using information from suitable model definitions and a CICS-generated termid.

Changes to resource definition

The USERID attribute of the TERMINAL resource definition is extended to support some special attribute values. The syntax of this attribute is as follows:

USERID(*name***|*EVERY|*FIRST)**
Specifies a user identifier used for signon and referred to in security error messages, security violation messages, and the audit trail.

name This can be up to eight characters in length. The acceptable characters are: A-Z 0-9 ¢ @ and #. Lowercase characters are converted to uppercase.

***EVERY (**autoinstalled consoles only**)**
Specifes that CICS is to use the user ID passed on the MVS MODIFY command every time a MODIFY command is received. The console is signed on using the MVS user ID as the preset user ID for the console being autoinstalled. The console remains signed on with this user ID until the console is deleted or another MODIFY command is received with another user ID. If a MODIFY command is received without a user ID, CICS signs on the default CICS user ID until a MODIFY command is received that has a valid user ID. For non-console terminals, or if security is not enabled, this value is ignored.

***FIRST (**autoinstalled consoles only**)**
Specifes that CICS is to use the user ID passed on the first MVS MODIFY command that requires the console to be autoinstalled. The console is signed on with the MVS user ID as the preset user ID. The console remains signed on with this user ID until the console is deleted. If a MODIFY command is received without a user ID, CICS signs on the default CICS user ID. For non-console terminals, or if security is not enabled, this value is ignored.

See the *CICS RACF Security Guide* for information about preset security on consoles and terminals.

Changes to the system programming interface

There are changes to the following SPI commands in support of autoinstall for consoles:

- EXEC CICS INQUIRE AUTOINSTALL
- EXEC CICS INQUIRE TERMINAL|NETNAME
- EXEC CICS SET AUTOINSTALL

Changes to the INQUIRE AUTOINSTALL command

A new option, CONSOLES, is added to this command, and the meanings of the ENABLESTATUS CVDAs are changed, as follows:

CONSOLES(*cvda***)**
Returns a CVDA value indicating the status of console autoinstall in CICS. The CVDA values are:

PROGAUTO
Consoles can be autoinstalled if ENABLESTATUS returns a CVDA of ENABLED. The autoinstall control program is called for the install and delete functions.

FULLAUTO
Consoles can be autoinstalled if ENABLESTATUS returns a CVDA of ENABLED. The autoinstall control program is not called for the install and delete functions.

NOAUTO
Consoles cannot be autoinstalled.

ENABLESTATUS
Returns a CVDA value indicating the status of the CICS autoinstall facility. The CVDA values are:

ENABLED
Either consoles or terminals or both can be autoinstalled in CICS. If you want to check whether ENABLED applies to consoles, terminals, or both, check the values returned on other options. ENABLED is returned for the following conditions:

Terminals
MAXREQS not equal 0 and autoinstall control program is enabled.

Consoles
1. CONSOLES CVDA returns PROGAUTO and autoinstall control program is enabled.

2. CONSOLES CVDA returns FULLAUTO.

DISABLED
Neither consoles nor terminals can be autoinstalled in CICS. DISABLED is returned for the following conditions:

Terminals
MAXREQS equal 0, or autoinstall control program is disabled.

Consoles
1. CONSOLES CVDA returns PROGAUTO but autoinstall control program is disabled.
2. CONSOLES CVDA returns NOAUTO.

Note: INVREQ (RESP2=1) is no longer returned.

Changes to INQUIRE TERMINAL|NETNAME

A new option, CONSOLE, is added to this command, as follows:

CONSOLE(*name_and_id***)**
Returns, for an MVS console only, a 12-byte string that contains the identifier of the console, in two sub-fields. If the device is not a console, CICS returns 12 blanks.

If the console is autoinstalled, or is defined with a console name, the name is returned in the first 8 bytes, and the last four bytes are blank.

If the console is defined by a numeric identifier, the string is divided into two sub-fields, separated by a period (.) in the ninth byte position. The sub-fields contain the following information:

- The first 8 bytes contain the MVS console name, if it is known, or the string '*UNKNOWN' if it isn't.
- The last 3 bytes contain the numeric console ID.

Changes to the SET AUTOINSTALL command

A new option, CONSOLES, is added to this command, as follows:

CONSOLES(*cvda***)**
Specifies whether CICS is to autoinstall console devices that are not predefined. The CVDA values are:

PROGAUTO
MVS consoles are to be autoinstalled, and CICS is to call the user autoinstall control program to obtain the termid and other user-specified information.

FULLAUTO
MVS consoles are to be autoinstalled by CICS automatically, without calling the user autoinstall control program. CICS is to assign the termid for the console automatically.

NOAUTO
Autoinstall for consoles is not allowed.

Changes to CICS-supplied transactions

There are changes to the following CEMT commands in support of autoinstall for consoles:

- CEMT INQUIRE AUTOINSTALL
- CEMT INQUIRE TERMINAL|NETNAME
- CEMT SET AUTOINSTALL

The CONSOLES and ENABLESTATUS options are added to the CEMT INQUIRE AUTOINSTALL command, the meanings of which are the same as for the EXEC CICS INQUIRE AUTOINSTALL command.

The CONSOLE option is added to the CEMT INQUIRE TERMINAL command, the meaning of which is the same as for the EXEC CICS INQUIRE TERMINAL command.

The PROGAUTO, FULLAUTO, and NOAUTO options are added to the CEMT SET AUTOINSTALL command, providing the same function as the equivalent options on the EXEC CICS SET AUTOINSTALL command.

See “Changes to the system programming interface” on page 76 for details of these options.

Changes to the user replaceable modules

The terminal autoinstall control program is a user-replaceable module (URM), the name of which you specify on the AIEXIT system initialization parameter. There are two new communication areas that CICS passes to this URM for an MVS console: (1) when the console is to be autoinstalled, and (2) when the console is to be deleted. If you use your own autoinstall control program, you need to modify your program to add support for autoinstalling MVS consoles.

If you use one of the IBM-supplied sample programs shown in Table 4, these are upgraded to include support for MVS consoles.

Table 4. Autoinstall programs and copy books

Language	Member name	Alias	Library
Programs: Assembler COBOL PL/I C/370®	DFHZATDX DFHZCTDX DFHZPTDX DFHZDTDX	— — — —	SDFHSAMP SDFHSAMP SDFHSAMP SDFHSAMP
Copy books: Assembler COBOL PL/I C/370	DFHTCUDS DFHTCUDO DFHTCUDP DFHTCUDH	— DFHTCUDS DFHTCUDS DFHTCUDS	SDFHMAC SDFHCOB SDFHPL1 SDFHC370

See the *CICS Customization Guide* for information about writing, or upgrading, an autoinstall control program that can handle MVS consoles.

Part 4. Application support and solution enablement

This Part describes the new function introduced to support application development and to help you to develop solutions that meet your business requirements. It covers the following topics:

Chapter 8. CICS business transaction services

This chapter describes CICS business transaction services (BTS). It covers the following topics:

For detailed information about CICS business transaction services, see the *CICS Business Transaction Services* manual.

Overview

CICS has always provided a robust transaction processing environment. For example, it:

- Allows you to create transactions with ACID properties [3] (atomicity, consistency, isolation, and durability)
- Allows transactions to continue to run under all sorts of conditions.

In recent years, much emphasis has been placed on continuous operation and high availability of CICS. Use of sophisticated technologies, such as the Parallel Sysplex, with resource managers sharing data across the sysplex, has led to improved system availablity through the elimination of single points-of-failure. CICS business transaction services (BTS) bring a similar sophistication to the CICS application programming interface (API), making it better able to model complex business transactions.

Business transactions and CICS transactions

This section examines the ways in which business transactions have traditionally been modelled by CICS transactions, and some of the shortcomings of the traditional approach.

Business transactions

A **business transaction** is a self-contained business deal—for example, buying a theatre ticket. Some business transactions—for example, buying a newspaper—are simple and short-lived. However, many are not. Many involve multiple actions that take place over an extended period. For example, selling a vacation may involve the travel agent in actions such as:

- Recording customer details
- Booking seats on an aircraft
- Booking a hotel
- Booking a hire car
- Invoicing the customer

3. Jim Gray and Andreas Reuter, *Transaction Processing: Concepts and Techniques*, 1993

- Checking for receipt of payment
- Processing the payment
- Arranging foreign currency.

Both the customer and the travel agent regard the purchase of the vacation as a single business transaction, as indeed it is, because each action only makes sense in the context of the whole. The example illustrates some typical properties of complex business transactions:

- They tend to be made up of a series of logical actions.
- Some actions may be taken days, weeks, or even months after the transaction was started—arranging foreign currency, in this example.
- Some of the actions may be optional—not everyone wants to hire a car, for example.
- At any point, an action could fail. For example, a communications failure could mean that it's not possible to book a hotel. In this case, the action must be retried. Or the customer might fail to meet his final payment; this would require a reminder to be sent. If the reminder produces no response, the vacation must be canceled—that is, the actions that have already been taken must be undone.
- Data—for example, a customer account number—must be passed between the individual actions that make up the business transaction.
- Some control logic is required, to "glue" the actions together. For example, there must be logic to deal with the conditional invocation of actions, and with failures.

CICS transactions

The basic building blocks used by CICS applications are the CICS transaction and the unit of work (UOW). Typically, a UOW is short-lived, because it is undesirable for it to hold locks for long periods, thus causing other UOWs to wait on resources and possibly abend. A CICS transaction consists of one or more UOWs. It provides the environment in which its associated UOWs will run—for example, the transid, program name, and userid. Typically, like the UOWs of which it consists, a CICS transaction is short-lived, because the aim should be for it to use CICS resources only while it is doing work—it should not spend long periods waiting for input, for example.

Before CICS Transaction Server for OS/390 Release 3, the largest transaction processing unit that CICS understood was the terminal-related pseudoconversation. A pseudoconversational application appears to a terminal user as a continuous conversation, but consists internally of multiple transactions.

The problems

Traditionally, application programmers have modelled business transactions using the basic CICS building blocks, transactions and units of work. However, there are problems. Here are some of them:

Application design: Typically, the individual actions that make up a complex business transaction are mapped on to CICS transactions. Usually, it is not practicable to map a whole business transaction on to a single, long-running CICS transaction (even if the transaction is divided into multiple units of work), because of resource constraints. The locks held by the UOWs would tend to be held for long periods; system performance would suffer, and transaction abends become frequent, due to deadlocks or contention for locked resources.

Mapping each individual action on to a CICS transaction is a more sensible option. However, this approach ignores the overall structure of the business transaction. Typically, the control logic necessary to glue the actions together ends up being spread between the various CICS transactions. Thus, the high-level logic required to control the overall progress of the business transaction and the low-level logic required to implement a specific business action become blurred. One effect is that the CICS transactions become less easy to reuse, because they are required to do more than implement a particular business action.

An even better option might be to separate the control logic in a single, top-level transaction that would be reinvoked whenever a new stage of the business transaction was ready to run. Each time it was invoked, the top-level transaction could run a transaction that implements a particular action of the business transaction. This would work similarly to a terminal-related pseudoconversation, in which terminal events cause successive transactions to be invoked. Unfortunately, in current CICS releases this is not possible. A pseudoconversational application can be used only to simulate a single conversation with a terminal.

Recovery and restart: Long-lived business transactions are much more likely than short-lived transactions to span restarts of CICS (which may or may not be planned). To survive restarts, state data relating to the business transaction's flow of control must be saved to a recoverable resource. Thought must also be given to how the business transaction is to be restarted after a restart of CICS.

What are CICS business transaction services?

CICS business transaction services extend the CICS API and provide support services that make it easier to model complex business transactions. As the vacation example on page 81 illustrates, business transactions are often made up of multiple actions, that may be spread over hours, days, or even months.

CICS business transaction services allow you to control the execution of complex business transactions. Using BTS, each action that makes up the business transaction is implemented as one or more CICS transactions, as in the traditional approach. However, a top-level program is used to control the overall progress of the business transaction. The top-level program manages the inter-relationship, ordering, parallel execution, commit scope, recovery, and restart of the actions that make up the business transaction.

What is a BTS application?

The components of an application written using the CICS business transaction services API are illustrated, in simplified form, in Figure 14. (For brevity, in the rest of this chapter we shall refer to an application that uses the CICS business transaction services API as "a BTS application".)

Figure 14. Components of a BTS application

The roles of the components are as follows:

Initial Request
A CICS transaction that starts a CICS business transaction services **process**.

Process

A collection of one or more BTS **activities**. It has a unique name by which it can be referenced and invoked. Typically, a process is an instance of a business transaction.

In the vacation example, an instance of the business transaction may be started to sell John Smith a vacation in Majorca. To identify this particular transaction as relating to John Smith, the process could be given the name of John Smith's account number.

Activity

The basic unit of BTS execution. Typically, it represents one of the actions of a business transaction—in the vacation example, booking a hire car, for instance.

A program that implements an activity differs from a traditional CICS application program only in its being designed to respond to BTS **events**. It can be written in any of the languages supported by CICS.

Activities can be hierarchically organized, in a tree structure. An activity that starts another activity is known as a **parent activity**. An activity that is started by another is known as a **child activity**.

Root activity

The activity at the top of the activity tree—it has no parent activity. A process always contains a root activity. When a process is started, the program that implements its root activity receives control. Typically, a root activity is a parent activity that:

- Creates and controls a set of child activities—that is, it manages their ordering, concurrent execution, and conditional execution
- Controls synchronization, parameter passing and saving of state data.

Data-container

A named area of storage, associated with a particular process or activity, and maintained by BTS. Each process or activity can have any number of data-containers. They are used to hold state data, and inputs and outputs for the activity.

Event (not shown in Figure 14)

A BTS event is a means by which CICS business transaction services signal progress in a process. It informs an activity that an action is required or has completed. "Event" is used in its ordinary sense of "something that happens". To define an event recognizable by CICS business transaction services, such a happening is given a name.

Timer (not shown in Figure 14)

A BTS object that expires when the system time becomes greater than a specified date and time, or after a specified period has elapsed. Each timer has an event associated with it. The event occurs ("fires") when the timer expires.

You can use a timer to, for example, cause an activity to be invoked at a particular time in the future.

The preceding components are managed by CICS, which:

- Manages many business transactions (processes)
- Records the current status of each business transaction
- Ensures that each activity is invoked at the appropriate times.

For detailed information about the components of a BTS application, and how they relate to each other, see the *CICS Business Transaction Services* manual.

Activation sequences: To complete its entire work, an activity may need to execute as a sequence of separate processing steps, or **activations**. For example, a parent activity typically needs to execute for a while, finish execution temporarily, then continue execution when one of its children has completed.

Each activation is "triggered" by a BTS event, and consists of a single transaction. An activity's first activation is triggered by the system event DFHINITIAL, supplied by BTS after the first RUN or LINK command is issued against the activity. (In the case of a root activity, DFHINITIAL occurs after the first RUN or LINK command is issued against the process.) When the last activation ends, the **activity completion event** is "fired", which may, in turn, trigger another activity's activation.

Figure 15 shows a BTS activity being reattached in a series of activations.

Figure 15. A sequence of activations

1 The first event that "wakes up" the activity is DFHINITIAL. The activity determines that the event which caused it to be activated was DFHINITIAL and therefore performs its first processing step. Typically, this involves defining further events for which it may be activated. The activity program issues an EXEC CICS RETURN command to relinquish control. The activity "sleeps".

2 The next event occurs and "wakes up" the activity. The activity program determines which event caused it to be activated and performs the processing step appropriate for that event. It issues an EXEC CICS RETURN command to relinquish control.

3 Eventually, no more processing steps are necessary. To confirm that its current activation is the last, and that it is not to be reactivated for any future events, the activity program issues an EXEC CICS RETURN ENDACTIVITY command. The activity completion event is fired.

Note: Root activities do not have completion events.

Figure 16 is a comparison between a terminal-related pseudoconversation and a BTS activity that is activated multiple times.

Figure 16. Comparison between a terminal-related pseudoconversation and a BTS activity that is activated multiple times

Note: The RETRIEVE REATTACH EVENT command issued by the activity retrieves the name of an event that caused the activity to be reactivated. The GET and PUT CONTAINER commands retrieve and store input and output data.

Control flow: The high-level control flow of a typical BTS business transaction is as follows:

1. A CICS transaction makes an initial request to start a process.
2. CICS initiates the appropriate root activity.

3. The root activity program, using the BTS API, creates a child activity—or several child activities. It provides the child activity with some input data (by placing the data in a data-container associated with the child), and requests CICS to start the child activity.

   If, as is often the case, the child activity is to run asynchronously with the root activity, the root activity program returns and becomes dormant.
4. The root activity is reactivated when one of its child activities completes. It determines which event caused it to be reactivated—that is, the completion of the activity that it started earlier. It retrieves, from the completed activity's output data-containers, any return data that the completed activity has placed there.
5. Steps 3 and 4 are repeated until all the child activities that make up the business transaction have completed.
6. The root activity program issues an EXEC CICS RETURN ENDACTIVITY command to indicate that it has completed all its processing steps. CICS terminates the root activity.

Recovery and restart

CICS maintains state data for BTS processes in a recoverable VSAM KSDS. This file can be RLS-enabled.

On an emergency restart, CICS automatically restarts any BTS activities that were in-flight at the time it failed.

Client/server support

CICS business transaction services support **client/server** processing. A server process is one that is typically waiting for work. When work arrives, BTS restarts the process, which retrieves any state data that it has previously saved.

Web Interface support

The CICS Web Interface allows Internet users to run CICS transactions from a Web browser. CICS business transaction services extend CICS support for the Internet.

In a typical current scenario, a Web-based business transaction might be implemented as a pseudoconversational CICS application. The initial request from the browser invokes a CICS transaction that does some setup work, returns a page of HTML to the browser, and ends. Subsequent requests are handled by other CICS transactions (or by further invocations of the same transaction). The CICS application is responsible for maintaining state data between requests.

Using BTS, a Web-based business transaction could be implemented as a BTS process. A major advantage of this approach is that state data is now maintained by BTS. This is particularly useful if the business transaction is long-lived.

Support for existing code

BTS supports the 3270 bridge function. (The 3270 bridge is described in the *CICS External Interfaces Guide*.) This means that BTS applications can be integrated with, and make use of, existing 3270-based applications.

Even though BTS activities are not terminal-related (they are never started directly from a terminal), a BTS activity can be implemented by a 3270-based transaction.

Sysplex support

You can operate BTS in a single CICS region. However, BTS processes are sysplex-enabled. In a sysplex, you can create one or more **BTS-set**s. A BTS-set is a set of CICS regions across which related BTS processes and activities may execute. For example, within a single process:

- The activities that constitute the process may execute on several regions.
- Different activations of the same activity may execute on different regions.

Dynamic routing of BTS activities: In a BTS-set, your BTS activities can be routed dynamically across the participating regions. You can control the dynamic routing of your BTS activities by either of the following means:

1. Writing a **distributed routing program**.

   CICS introduces a new dynamic routing model, the **distributed routing model**, which complements the "hub" model traditionally used for CICS dynamic transaction routing.

   In the traditional "hub" routing model, a *dynamic routing program* running in a terminal-owning region routes transactions between a set of application-owning regions. The new distributed model, on the other hand, is a "peer-to-peer" system, in that a *distributed routing program* runs in each participating BTS region; each region can be both a requesting region and a target region.

   The two routing models and routing programs are more fully described in "Changes to the dynamic routing interface" on page 41.
2. Using the CICSPlex System Manager/ESA (CICSPlex SM) element of CICS TS to:
   - Optionally, specify workload separation for your BTS processes
   - Manage affinities
   - Control workload balancing of the transactions that implement BTS activities.

Notes:

1. Dynamic routing of BTS activities is at the *activation level*. When an event is signalled, an activity is activated in the most appropriate region in the BTS-set, based on one or more of the following:
   - Any workload separation specified by the system programmer
   - Any affinities its associated transaction has with a particular region
   - The availability of regions
   - The relative workload of regions.
2. Activities can be routed, either dynamically or statically, only when they are run asynchronously with the requestor. When an activity is run synchronously with the requestor, it cannot be routed to another region, neither dynamically nor statically.

Audit trails: You can create an audit trail for the BTS processes and activities that run in your CICS regions. Doing so allows you to, for example, track the progress of a complex business transaction across the sysplex.

The CICS code contains BTS audit points in much the same way as it contains trace points. However, because in a sysplex environment different parts of a process may execute on different regions, each audit record contains system, date, and time information. By sharing log streams across regions, you can gather audit information from different regions in the same log.

Benefits

Compared with traditional CICS methods of modelling complex business transactions, CICS business transaction services confer a number of advantages:

- Management and control is at the business transaction level, as well as at the action level.
- Control logic is separated from business logic. The individual CICS transactions that make up the business transaction no longer need to be concerned with "before and after" actions. This simplifies the development of such transactions and makes it easier to reuse them.
- Because CICS maintains monitoring information for BTS processes and activities, you can request information about a business transaction's use of resources without knowing the identifiers of all its constituent CICS transactions. Information is now available at the business transaction level, as well as at the CICS transaction level.
- In a sysplex, BTS processes and activities can take full advantage of CICSPlex SM's workload separation and workload balancing functions.
- BTS processes can be used as servers in a client/server environment.

Requirements

To operate CICS business transaction services in a single CICS region, there are no additional requirements beyond those for CICS TS generally.

To create a BTS-set, you require a coupling facility. All the regions in a BTS-set must be in the same MVS Parallel Sysplex. This is because, to support the necessary sharing of process and activity data between the regions, BTS uses VSAM record-level sharing (RLS). VSAM RLS requires a coupling facility.

Changes to CICS externals

This section gives an overview of the changes to CICS externals introduced by CICS business transaction services. For full details of these changes, see the *CICS Business Transaction Services* manual. The topics covered are:

Changes to the application programming interface

CICS business transaction services add a subset of new commands to the CICS application programming interface (API). Also, some existing commands have been modified.

New API commands

The new BTS API commands are:

EXEC CICS ACQUIRE
Gives a UOW executing outside a BTS process access to an activity within the process.

EXEC CICS ADD SUBEVENT
Adds a sub-event to a composite event.

EXEC CICS CANCEL
Forces a process or activity to complete.

EXEC CICS CHECK ACQPROCESS
Returns the completion status of the process that the requestor has acquired in the current unit of work.

EXEC CICS CHECK ACTIVITY
Returns the completion status of an activity.

EXEC CICS CHECK TIMER
Returns the status of a timer and, if the timer has expired, deletes the event associated with it.

EXEC CICS DEFINE ACTIVITY
Creates a new child activity.

EXEC CICS DEFINE COMPOSITE EVENT
Defines a composite event.

EXEC CICS DEFINE INPUT EVENT
Defines an input event.

EXEC CICS DEFINE PROCESS
Creates a new BTS process.

EXEC CICS DEFINE TIMER
Defines a timer, and associates an event with it.

EXEC CICS DELETE ACTIVITY
Removes a child activity from the BTS data set where it is defined.

EXEC CICS DELETE CONTAINER
Deletes a named data-container.

EXEC CICS DELETE EVENT
Deletes an event.

EXEC CICS DELETE TIMER
Deletes a timer and its associated event (if any).

EXEC CICS ENDBROWSE ACTIVITY
Ends a browse of the child activities of an activity, or of the descendent activities of a process.

EXEC CICS ENDBROWSE CONTAINER
Ends a browse of the data-containers associated with an activity or process.

EXEC CICS ENDBROWSE EVENT
Ends a browse of the events known to an activity.

EXEC CICS ENDBROWSE PROCESS
Ends a browse of processes of a specified type.

EXEC CICS FORCE TIMER
Forces the early expiry of a timer, and causes the timer's associated event to fire.

EXEC CICS GET CONTAINER
Retrieves data from a named data-container.

EXEC CICS GETNEXT ACTIVITY
Browses the child activities of an activity, or the descendent activities of a process.

EXEC CICS GETNEXT CONTAINER
Browses the data-containers associated with an activity or process.

EXEC CICS GETNEXT EVENT
Browses the events known to an activity.

EXEC CICS GETNEXT PROCESS
Browses processes of a specified type.

EXEC CICS INQUIRE ACTIVITYID
Returns information about an activity to an external caller.

EXEC CICS INQUIRE CONTAINER
Retrieves the attributes of a data-container.

EXEC CICS INQUIRE EVENT
Retrieves the attributes of an event.

EXEC CICS INQUIRE PROCESS
Returns information about a process to an external caller.

EXEC CICS INQUIRE TIMER
Retrieves the attributes of a timer.

EXEC CICS LINK ACQPROCESS
Invokes the program that implements the process that the requestor has acquired in the current unit of work. Runs the program synchronously with the requestor, in the same unit of work, and with the same transaction attributes as the requestor.

EXEC CICS LINK ACTIVITY
Invokes a program that implements an activity. Runs it synchronously with the requestor, in the same unit of work, and with the same transaction attributes as the requestor.

EXEC CICS PUT CONTAINER
Saves data in a named data-container, creating the container if it does not already exist.

EXEC CICS REMOVE SUBEVENT
Removes a sub-event from a composite event.

EXEC CICS RESET ACQPROCESS
Resets the process that the requestor has acquired in the current unit of work to its initial state—used before retrying the process.

EXEC CICS RESET ACTIVITY
Resets an activity to its initial state—used before retrying an activity.

EXEC CICS RESUME
Allows a suspended process or activity to be reattached if events in its event pool fire.

EXEC CICS RETRIEVE REATTACH EVENT
Retrieves the name of an event that caused the current activity to be reattached.

EXEC CICS RETRIEVE SUBEVENT
Retrieves the name of the next sub-event in a composite event's sub-event queue.

EXEC CICS RUN
Invokes a program that implements a process or activity. Runs it synchronously or asynchronously with the requestor, in a separate unit of work, and with the transaction attributes specified on the DEFINE PROCESS or DEFINE ACTIVITY command.

EXEC CICS STARTBROWSE ACTIVITY
Starts a browse of the child activities of an activity, or of the descendent activities of a process.

EXEC CICS STARTBROWSE CONTAINER
Starts a browse of the data-containers associated with an activity or process.

EXEC CICS STARTBROWSE EVENT
Starts a browse of the events known to an activity.

EXEC CICS STARTBROWSE PROCESS
Starts a browse of processes of a specified type.

EXEC CICS SUSPEND
Prevents a process or activity being reattached if events in its event pool fire.

EXEC CICS TEST EVENT
Test whether an event has fired.

Changed API commands

The following existing API commands have been modified for use with BTS:

EXEC CICS ASSIGN
The following new options have been added. They return information about the BTS activity and process that the current unit of work is acting for.
- ACTIVITY
- ACTIVITYID
- PROCESS
- PROCESSTYPE

EXEC CICS RETURN
An ENDACTIVITY option has been added. This indicates that a process or activity is complete.

Changes to the system programming interface

CICS business transaction services add a subset of new system programming commands to CICS. Also, some existing commands have been modified.

New system programming commands

The new BTS system programming commands are:

EXEC CICS CREATE PROCESSTYPE
Builds a PROCESSTYPE definition in the local CICS region, without reference to data in the CICS system definition (CSD) file.

EXEC CICS DISCARD PROCESSTYPE
Removes a PROCESSTYPE definition from the local CICS region.

EXEC CICS INQUIRE PROCESSTYPE
Retrieves the attributes of a process-type.

EXEC CICS SET PROCESSTYPE
Modifies the attributes of a PROCESSTYPE definition.

Changed system programming commands

The following existing system programming commands have been modified for use with BTS:

EXEC CICS INQUIRE SYSTEM
A new DSRTPROGRAM option has been added. This returns the name of the distributed routing program currently identified to the system.

EXEC CICS INQUIRE TASK
The following new options have been added. They return information about the BTS activity and process that a task is executing on behalf of.
- ACTIVITY
- ACTIVITYID
- PROCESS
- PROCESSTYPE

EXEC CICS SET SYSTEM
A new DSRTPROGRAM option has been added. This specifies the name of the distributed routing program.

Changes to resource definition

Most BTS resources (processes, activities, events, and containers) are defined at run time, using BTS API commands.

The only new BTS resource type that can be defined in the CSD is the process-type, using the DEFINE PROCESSTYPE command. The files and journals used by BTS are defined using the existing DEFINE FILE and DEFINE JOURNALMODEL commands, as described in the *CICS Resource Definition Guide*.

The XRSINDI global user exit is invoked when PROCESSTYPE resource types are installed or discarded.

Changes to system definition

A new system initialization parameter, DSRTPGM, is introduced; it names the distributed routing program.

Table 5. The DSRTPGM system initialization parameter

	DFHSIT	[TYPE={**CSECT**|DSECT}] [,DSRTPGM={**NONE**|DFHDSRP|*program-name*|EYU9XLOP}] . . .

DSRTPGM={NONE|DFHDSRP|program-name|EYU9XLOP}

Specifies the name of the distributed routing program to be used for dynamically routing:

- Eligible CICS business transaction services (BTS) processes and activities
- Eligible non-terminal-related EXEC CICS START requests.

DFHDSRP

The CICS sample distributed routing program.

EYU9XLOP

The CICSPlex SM routing program.

NONE

For eligible BTS processes and activities, no routing program is invoked. BTS processes and activities cannot be dynamically routed.

For eligible non-terminal-related START requests, the CICS sample distributed routing program, DFHDSRP, is invoked.

program-name

The name of a user-written program.

Note: See also the DTRPGM parameter, used to name the dynamic routing program.

BTS introduces a new mandatory CICS data set—the local request queue (LRQ). You must define an LRQ even if you don't use BTS facilities.

Changes to CICS-supplied transactions

These are the CICS-supplied transactions that have been added or changed to support BTS:

CBAM Browses the BTS objects (process-types, processes, activities, containers, events and timers) known to this region. CBAM is menu-driven and is a "readonly" transaction—you cannot update any of the displayed attributes by overtyping their values.

CEDA DEFINE PROCESSTYPE

Defines a CICS business transaction services process-type. Process-types are used to categorize BTS processes and activities. This is useful for browsing and auditing purposes. It is possible to browse all processes of a specified type, for example.

CEMT INQUIRE PROCESSTYPE

Retrieves information about a CICS business transaction services process-type.

CEMT INQUIRE SYSTEM
A new DSRTPROGRAM option has been added. This displays the name of the distributed routing program currently identified to the system.

CEMT INQUIRE TASK
New options have been added, that display details of the BTS activity and process that a task is executing on behalf of.

CEMT SET PROCESSTYPE
Changes the attributes of a CICS business transaction services process-type.

CEMT SET SYSTEM
A new DSRTPROGRAM option has been added. This specifies the name of the distributed routing program.

Changes to user-replaceable programs

A new user-replaceable program, DFHDSRP, is introduced. This is the default distributed routing program, which handles the dynamic routing of:
- BTS activities
- Non-terminal-related START commands.

To support the new dynamic routing functions, several new fields are added to the DFHDYPDS communication area. This communication area is now passed to the distributed routing program as well as to the dynamic routing program.

Changes to monitoring

CICS maintains monitoring information for BTS processes and activities. See “Chapter 6. Monitoring, statistics, and enterprise management changes” on page 65 for details of the monitoring support for CICS BTS.

Changes to problem determination

There are changes to CICS trace, messages, and abend codes to aid problem determination when you are using CICS BTS.

Trace points

A number of new trace points are introduced. BTS consists of three CICS domains, each of which has a 2-character component identifier, used to specify levels of standard and special tracing:

Domain name	CICS Component code
Business application manager	BA
Event manager	EM
Scheduler services	SH

Audit points

The CICS code contains BTS audit points in much the same way as it contains trace points. However, there are three main differences between audit records and trace entries:

1. Trace entries are written to an internal trace table within the CICS address space. In contrast, the audit trail of a process is written to a CICS journal, which resides on an MVS log stream.
2. Trace entries record the progress of tasks over a relatively short period of time, typically seconds, minutes, or hours. In contrast, the audit trail of a process can extend to days, weeks, or even months.
3. Trace entries relate to activity in a single CICS region. In contrast, in a sysplex environment the execution of different parts of a process may take place on different regions within the sysplex. Therefore, each audit record contains system, date, and time information. Typically, an audit record for a BTS activity also contains:
   - The identifier of the activity
   - The process to which the activity belongs
   - Information about the event which caused the activity to be invoked, canceled, suspended, or resumed; or that fired when it completed.

   Because log streams can be shared by more than one region, it is possible to write audit records from different regions to the same log, using the merge facility of the MVS system logger.

A number of levels of logging are available:
1. No logging
2. Primary logging of processes
3. Primary logging of both processes and activities
4. Full logging of both processes and activities.

Audit log records are written to an MVS log stream by the CICS log manager. You can read the records offline using the CICS audit trail utility program, DFHATUP. DFHATUP allows you to:
- Filter records for specific process-types, processes, and activities
- Interpret records into a readable format.

Messages and abend codes

A number of new messages and abend codes are introduced. These are described in the *CICS Messages and Codes* manual.

Changes to utility programs

Two new CICS utility programs are introduced:

- The audit trail utility program, DFHATUP, enables you to print selected BTS audit records from a logstream.
- The repository utility program, DFHBARUP, enables you to print selected records from a BTS repository data set.

Changes to sample programs

Besides the many fragments of example pseudocode in the *CICS Business Transaction Services* manual, CICS supplies a sample BTS application. The sample is a basic sales application, consisting of order, credit check, stock check, delivery note, invoice, and payment/reminder activities. It is implemented as a set of COBOL programs and copybooks. These are supplied, in source code only, in the SDFHSAMP library.

An example BTS application

The Sale example application is a set of programs that demonstrates how to use CICS business transaction services to manage business transactions.

Overview

The Sale example implements a *Sale* business transaction that is made up of four basic actions:

- Order entry
- Delivery
- Invoice
- Payment.

A Sale business transaction is started by a terminal-user selecting the `Sale` option from a menu of business transactions. This causes an instance of the transaction to be created and its root activity to be started. The root activity creates and runs, in sequence, four child activities that implement the four actions of the business transaction:

1. The *Order activity* obtains order data from the user, and validates it.
2. Successful completion of the Order activity causes the *Delivery activity* to be started.
3. Completion of the Delivery activity causes the *Invoice activity* to be started.
4. When payment is received and recorded by the *Payment activity*, the Sale business transaction is complete.

Data flows

Figure 17 shows, in simplified form, data flows in the Sale example application.

Figure 17. Data flow in the Sale example apllication. (The root activity is not shown.)

1. Customer data (for example, an account number) collected after the terminal user selects the `Sale` menu option is used as input to the Order activity.
2. Customer data collected by the Order activity is used as input to the Delivery activity.
3. The output data produced by the Delivery activity is used as input to the Invoice activity.
4. The output produced by the Invoice activity is used as input to the Payment activity.

Note: The first activity (Order) requires input from the terminal user. For the purposes of this simple example, subsequent activities (Delivery, Invoice and Payment) are assumed not to require any user involvement and are triggered serially in the background after the Order activity has completed successfully. For examples of activities that require user input, see the *CICS Business Transaction Services* manual.

CICS transactions and programs

Table 6 shows the CICS transactions and programs that make up the basic Sale application described in this chapter.

Table 6. Transactions and programs in the Sale application

Transid	Program	Comments
MENU	MNU001	Menu of business transactions
—	SAL001	Creates and starts the Sale business transaction
SALE	SAL002	BTS root activity, manages the child activities that comprise the Sale business transaction
SORD	ORD001	Order activity
SDEL	DEL001	Delivery activity
SINV	INV001	Invoice activity
SPAY	PAY001	Payment activity

Notes:

1. In the *CICS Business Transaction Services* manual, the Sale example application is extended to illustrate more advanced features of BTS, such as:
   - Parallel activities
   - User-related activities
   - Compensation actions.
2. For the sake of clarity, the basic example does not include any error handling code.

The initial request

The initial request to start a Sale business transaction is handled by the MNU001 and SAL001 programs. When a terminal user selects the `Sale` menu option, the menu program MNU001 links to the SAL001 program to service the request. SAL001 establishes a unique reference for this instance of the Sale business transaction and starts it.

Figure 18 on page 98 shows, in COBOL pseudocode, how SAL001 creates and starts an instance of the Sale business transaction.

```
Identification Division.
Program-id. SAL001.
Environment Division.
Data Division.
Working-Storage Section.
01  Sales-Reference              pic x(36) value low-values.
    .
01  Process-Type                 pic x(8) value 'Sales'.
    .
Linkage Section.
01  DFHEIBLK.
    .
01  DFHCOMMAREA.
    .
Procedure Division using DFHEIBLK DFHCOMMAREA.
In-The-Beginning.
    .
    .. create unqiue sales reference ..
    .
    EXEC CICS DEFINE PROCESS(Sales-Reference) PROCESSTYPE(Process-Type)
                TRANSID('SALE')
                PROGRAM('SAL002')
             RESP(data-area) RESP2(data-area) END-EXEC
      .
    EXEC CICS RUN ACQPROCESS
                SYNCHRONOUS
             RESP(data-area) RESP2(data-area) END-EXEC
    .
    EXEC CICS RETURN END-EXEC
End Program.
```

Figure 18. Pseudocode for the SAL001 program. SAL001 creates and starts an instance of the Sale business transaction.

Creating the business transaction

To create an instance of the Sale business transaction, SAL001 issues a DEFINE PROCESS command. The PROGRAM option of DEFINE PROCESS defines a program to run under the control of CICS business transaction services—a root activity program that typically manages the ordering and execution of the child activities that make up a business transaction. In this case, the program is SAL002, which is the root activity program for the Sale business transaction.

The PROCESS option uniquely identifies this business transaction instance from others. (The creation of a unique reference is managed by the user. Typically, you might use a customer reference or account number.)

The PROCESSTYPE option categorizes the business transaction by assigning it a process-type of 'Sales'. Categorizing your processes (business transactions) in this way means that you can browse details of individual processes—and their constituent activities—more easily.

The TRANSID option serves a number of purposes:

Security
: If security is active, CICS performs a security check to see if the requestor has authority to use the specified transaction identifier (transid). Thus, in this example, there would be a check on whether the requestor is authorized to create a new instance of the Sale business transaction.

Externals
: When a business transaction is started, its root activity program begins

executing, and any external inquiry such as CEMT shows work being done under the root activity's transaction identifier.

In the Sale application, the Sale business transaction is started under the control of the MENU transaction; however, the actual start of an instance of the Sale transaction occurs when control is passed to the root activity program, SAL002. At this point, the transaction identifier changes from MENU to SALE.

Root activity

Later restarts of a root activity may be required to deal with child activities that are executed with the RUN ACTIVITY ASYNCHRONOUS command (the child activities are executed asynchronously with the root activity, are not included in its unit of work, and have different transaction identifiers).

In the Sale application, the SAL002 root activity program is attached under the SALE transaction identifier to deal with the Delivery, Invoice, and Payment activities, that all execute asynchronously, under separate UOW scope, and under different transaction identifiers.

Monitoring and statistics

The transaction identifier can be used to track resource usage for monitoring, statistics, and accounting purposes. It allows monitoring and statistics information to be related to a CICS business transaction services process.

DEFINE PROCESS is a synchronous request and control is returned to the requesting program when BTS has accepted the request and added the process to the set that it is currently managing.

The addition of the process is not committed until the current unit of work has taken a successful syncpoint. If the requesting task abends before the syncpoint is taken, the request to add the process is canceled.

Starting the business transaction

To start this instance of the Sale business transaction, on return from the DEFINE PROCESS request SAL001 issues a RUN ACQPROCESS command. RUN ACQPROCESS causes the process that has been "acquired" in the current unit of work to be activated. A program can "acquire" a process in two ways: by defining it, or by issuing an ACQUIRE PROCESS command. Here, SAL001 has acquired a process by defining it; thus the RUN ACQPROCESS command causes the SAL002 program specified on the DEFINE PROCESS command to be executed.

Using RUN causes the process to be activated in a separate unit of work from that of the requesting transaction, under the transaction identifier specified on the TRANSID option of the DEFINE PROCESS command. (A LINK ACQPROCESS command would have caused SAL002 to be executed in the same unit of work as MNU001 and SAL001, and under the same TRANSID, MENU.) The advantages of giving a process a separate TRANSID from that of its creator are explained in "Creating the business transaction" on page 98. The SYNCHRONOUS option on the RUN command causes SAL002 to be executed synchronously with SAL001.

Although a RUN ACQPROCESS command causes a process to be activated in a separate unit of work from that of its requestor, the start and finish of the activation are related to the requestor's syncpoints. In the example application, the SAL002 root activity runs its first child activity (Order) synchronously and as part of its own unit of work. If the Order activity is successfully completed (in the business sense

as well as the transactional sense), the Sale business transaction will be accepted. If not, it will be rejected. “Accepted” means committed—this instance of the Sale transaction will be ready to start its next activity. “Rejected” means rolled back—this instance of the Sale transaction will no longer exist.

The root activity

The SAL001 program starts a new instance of the Sale business transaction by starting the SAL002 program, running under the transid SALE. SAL002 implements a root activity that manages the inter-relationship, ordering, and execution of the child activities that make up the Sale business transaction.

A root activity program such as SAL002 is designed to be reattached by CICS business transaction services when events in which it is interested are triggered. The activity program determines which of the possible events caused it to be attached and what to do as a result. A typical sequence (somewhat simplified) is:

1. The root activity requests BTS to run a child activity (possibly several child activities), and to notify it when the child has completed.
2. The root activity “sleeps” while waiting for the child activity to complete.
3. BTS reattaches the root activity because the child activity has completed.
4. The root activity requests the next child activity to run.
5. Steps 1 through 4 are repeated until the business transaction is complete.

Thus, even though the root activity is not initiated from a terminal, you could think of its style as being “pseudoconversational”.

Figure 19 on page 101 shows, in COBOL pseudocode, the Sale root activity program, SAL002.

```
Identification Division.
Program-id. SAL002.
Environment Division.
Data Division.
Working-Storage Section.
01  RC                          pic s9(8) comp.
01  Process-Name                pic x(36).
01  Event-Name                  pic x(16).
    88  DFH-Initial             value 'DFHINITIAL'.
    88  Delivery-Complete       value 'Delivry-Complete'.
    88  Invoice-Complete        value 'Invoice-Complete'.
    88  Payment-Complete        value 'Payment-Complete'.
01  Sale-Container               pic x(16) value 'Sale'.
01  Order-Container              pic x(16) value 'Order'.
01  Order-Buffer                pic x(..).
01  Delivery-Container           pic x(16) value 'Delivery'.
01  Delivery-Buffer              pic x(..).
01  Invoice-Container            pic x(16) value 'Invoice'.
01  Invoice-Buffer               pic x(..).
Linkage Section.
01  DFHEIBLK.
    .
Procedure Division.
Begin-Process.
     .
    EXEC CICS RETRIEVE REATTACH EVENT(Event-Type)
              RESP(RC) END-EXEC
      .
    If RC NOT = DFHRESP(NORMAL)
      .
    End-If.
      .
    Evaluate True
      When DFH-Initial
        Perform Initial-Activity
        Perform Order-Activity
        Perform Delivery-Activity
      When Delivery-Complete
        Perform Invoice-Activity
      When Invoice-Complete
        Perform Payment-Activity
      When Payment-Complete
        Perform End-Process
      When Other
        .
    End Evaluate.
        .
    EXEC CICS RETURN END-EXEC
    .
```

Figure 19. Pseudocode for SAL002, the root activity program for the Sale business transaction (Part 1 of 3)

```
Initial-Activity.
    .
    EXEC CICS ASSIGN PROCESS(Process-Name)
             RESP(data-area) RESP2(data-area) END-EXEC
    .
Order-Activity.
    .
    EXEC CICS DEFINE ACTIVITY('Order')
                 TRANSID('SORD')
                 PROGRAM('ORD001')
             RESP(data-area) RESP2(data-area) END-EXEC
    .
    EXEC CICS PUT CONTAINER(Sale-Container)
                 ACTIVITY('Order') FROM(Process-Name)
             RESP(data-area) RESP2(data-area) END-EXEC
    .
    EXEC CICS LINK ACTIVITY('Order')
             RESP(data-area) RESP2(data-area) END-EXEC
    .
Delivery-Activity.
    .
    EXEC CICS DEFINE ACTIVITY('Delivery')
                 TRANSID('SDEL')
                 EVENT('Delivry-Complete')
             RESP(data-area) RESP2(data-area) END-EXEC
    .
    EXEC CICS GET CONTAINER(Order-Container)
                 ACTIVITY('Order') INTO(Order-Buffer)
             RESP(data-area) RESP2(data-area) END-EXEC
    .
    EXEC CICS PUT CONTAINER(Order-Container)
                 ACTIVITY('Delivery') FROM(Order-Buffer)
             RESP(data-area) RESP2(data-area) END-EXEC
    .
    EXEC CICS RUN ACTIVITY('Delivery')
                 ASYNCHRONOUS
             RESP(data-area) RESP2(data-area) END-EXEC
    .
```

Figure 19. Pseudocode for SAL002, the root activity program for the Sale business transaction (Part 2 of 3)

```
Invoice-Activity.
    .
    EXEC CICS DEFINE ACTIVITY('Invoice')
                TRANSID('SINV')
                EVENT('Invoice-Complete')
             RESP(data-area) RESP2(data-area) END-EXEC
    .
    EXEC CICS GET CONTAINER(Delivery-Container)
                 ACTIVITY('Delivery') INTO(Delivery-Buffer)
             RESP(data-area) RESP2(data-area) END-EXEC
    .
    EXEC CICS PUT CONTAINER(Delivery-Container)
                 ACTIVITY('Invoice') FROM(Delivery-Buffer)
             RESP(data-area) RESP2(data-area) END-EXEC
    .
    EXEC CICS RUN ACTIVITY('Invoice')
                ASYNCHRONOUS
             RESP(data-area) RESP2(data-area) END-EXEC
    .
Payment-Activity.
    .
    EXEC CICS DEFINE ACTIVITY('Payment')
                TRANSID('SPAY')
                EVENT('Payment-Complete')
             RESP(data-area) RESP2(data-area) END-EXEC
    .
    EXEC CICS GET CONTAINER(Invoice-Container)
                 ACTIVITY('Invoice') INTO(Invoice-Buffer)
             RESP(data-area) RESP2(data-area) END-EXEC
    .
    EXEC CICS PUT CONTAINER(Invoice-Container)
                 ACTIVITY('Payment') FROM(Invoice-Buffer)
             RESP(data-area) RESP2(data-area) END-EXEC
    .
    EXEC CICS RUN ACTIVITY('Payment')
                ASYNCHRONOUS
             RESP(data-area) RESP2(data-area) END-EXEC
    .
End-Process.
    .
    EXEC CICS RETURN ENDACTIVITY
             RESP(data-area) RESP2(data-area) END-EXEC
End Program.
```

Figure 19. Pseudocode for SAL002, the root activity program for the Sale business transaction (Part 3 of 3)

The following discussion steps through the SAL002 pseudocode shown in Figure 19 on page 101:

1. The root activity determines what event caused it to be attached by issuing the following command:

   ```
   EXEC CICS RETRIEVE REATTACH EVENT(Event-Type)
             RESP(data-area) RESP2(data-area) END-EXEC
   ```

   The first time an activity is started during a process, the event returned is the system event DFHINITIAL. This tells the activity that it should perform any initial housekeeping.

   In this example, CICS initially invokes the SAL002 root activity as a result of the RUN ACQPROCESS command issued by the SAL001 program. As part of its initial housekeeping, SAL002 uses the EXEC CICS ASSIGN PROCESS command to discover the name of this instance of the business transaction

(process). (The name of the process instance was assigned by the DEFINE PROCESS command, and might be, for example, a customer reference or account number.)

2. The root activity creates its first child activity, which in this case is the Order activity:

```
EXEC CICS DEFINE ACTIVITY('Order')
             TRANSID('SORD')
             PROGRAM('ORD001')
          RESP(data-area) RESP2(data-area) END-EXEC
```

   The DEFINE ACTIVITY command requests CICS business transaction services to add an activity to a business transaction (process). In this example, SAL002 adds an activity called *Order* to the Sale business transaction. It is implemented by program ORD001. The TRANSID option specifies that, if the Order activity is run in its own unit of work, it will run under transaction identifier SORD.

3. When the Order activity has been added, SAL002 uses the PUT CONTAINER command to provide it with some input data.

```
EXEC CICS PUT CONTAINER(Sale-Container)
         ACTIVITY('Order') FROM(Process-Name)
         RESP(data-area) RESP2(data-area) END-EXEC
```

   The input data is placed in a data-container named *Sale* (the value of the variable *Sale-Container*). The ACTIVITY option of PUT CONTAINER associates the *Sale* data-container with the Order activity.

   Note: An activity can have many data-containers associated with it. A data-container is associated with an activity simply by being named on a command (such as PUT CONTAINER) that specifies the activity.

   Two or more activities can each have a data-container named, for example, *Order*.

   The data put into the *Sale* data-container is the process name—that is, the unique reference that identifies this instance of the Sale business transaction. The process name in this case is the customer reference or account number specified on the DEFINE PROCESS command in SAL001.

4. SAL002 requests BTS to start the Order activity:

```
EXEC CICS LINK ACTIVITY('Order')
         RESP(data-area) RESP2(data-area) END-EXEC
```

   The LINK ACTIVITY command causes the ORD001 program to be executed synchronously with SAL002 and to be included as part of the current unit of work. The TRANSID option of the DEFINE ACTIVITY command is ignored—LINK ACTIVITY causes the Order activity to run under the requestor's transaction identifier, SALE.

   The Order activity collects order details from the terminal operator and validates them. The ORD001 program converses with the terminal operator until the order is accepted. It then returns the validated details in an output data-container.

5. When the Order activity completes, SAL002 creates the Delivery activity:

```
EXEC CICS DEFINE ACTIVITY('Delivery')
             TRANSID('SDEL')
             EVENT('Delivry-Complete')
          RESP(data-area) RESP2(data-area) END-EXEC
```

The Delivery activity is to be executed asynchronously with the root activity. When an activity completes, its completion event fires. The EVENT option *names* the Delivery activity's completion event as *Delivry-Complete*, and thus defines it. Defining the event allows it to be referenced and checked for.

CICS reattaches an activity on the firing of any event, other than a sub-event, that is in its event pool. (An activity's event pool contains events that have been defined to the activity, plus the DFHINITIAL system event.) Thus, the SAL002 root activity will be reattached when the Delivery activity's completion event (*Delivry-Complete*) fires.

Note: All child activities have completion events that fire when the activities complete. If the EVENT option of DEFINE ACTIVITY is not used, CICS gives the completion event the same name as the activity itself.

For child activities like the *Order* activity, that will always be executed *synchronously* with the parent, the EVENT option is not often used. Normally, the firing of a synchronous activity's completion event does not cause the parent to be reattached, because the event is deleted (by a CHECK ACTIVITY command) during the parent's current activation. Therefore the event never needs to be tested for by name, among several other possible reattachment events.

The CHECK ACTIVITY command is described in the *CICS Business Transaction Services* manual.

6. SAL002 makes the data returned by the Order activity available to the Delivery activity:

```
EXEC CICS GET CONTAINER(Order-Container)
              ACTIVITY('Order') INTO(Order-Buffer)
          RESP(data-area) RESP2(data-area) END-EXEC

EXEC CICS PUT CONTAINER(Order-Container)
              ACTIVITY('Delivery') FROM(Order-Buffer)
          RESP(data-area) RESP2(data-area) END-EXEC
```

   Here, the GET and PUT commands are used to transfer data from the Order activity's output data-container to the Delivery activity's input data-container (both of which are named *Order*). Note that these are different data-containers—although they share the same name, they are associated with different activities.

7. SAL002 requests BTS to start the Delivery activity:

```
EXEC CICS RUN ACTIVITY('Delivery')
              ASYNCHRONOUS
          RESP(data-area) RESP2(data-area) END-EXEC
```

   Because RUN rather than LINK is used, the Delivery activity will be executed as a separate unit of work, and under the transaction identifier specified on the TRANSID option of the DEFINE ACTIVITY command. (The RUN command always activates the specified process or activity in a new unit of work.) Because the ASYNCHRONOUS option is used, the Delivery activity will be executed asynchronously with SAL002, and will start only if the current unit of work completes successfully.

8. SAL002 issues an EXEC CICS RETURN command. Control is returned to SAL001, then to MNU001, and finally to CICS. CICS takes a syncpoint and commits the following:
   - The creation of a new Sale business transaction

- Work done by the Order activity, and its input and output data-containers
- The request to run the Delivery activity, and its input data-container
- The condition under which the SAL002 root activity is to be "pseudoconversationally" reattached.

After the CICS syncpoint, the menu of business transactions is redisplayed on the user's terminal, ready for further selection. The remaining activities will be completed, without reference to the terminal user, under the control of CICS business transaction services. The SAL002 program no longer exists in memory, and the existence of this instance of the Sale business transaction is known only to BTS.

CICS business transaction services start the Delivery activity (SDEL) as requested. (BTS participates as a resource manager for the transaction.) On completion of the Delivery activity, BTS reattaches the Sale root activity—that is, the SAL002 program under the transaction identifier SALE.

9. The SAL002 program is entered at the top again, and so determines what event caused it to be reattached by issuing the RETRIEVE REATTACH EVENT command. This time, however, the event returned is *Delivry-Complete*. Having established which child activity has completed, SAL002 determines that the next activity to be started is the Invoice activity.

   As with the Order activity, SAL002 sets the Invoice activity's parameters, input data, and execution options before requesting the activity to be run. It then issues an EXEC CICS RETURN command and waits to be reattached for this instance of the Sale business transaction.
10. The pattern implied in step 9 is repeated until the Payment activity completes, at which point the Sale business transaction is complete. SAL002 issues an EXEC CICS RETURN command on which the ENDACTIVITY option is specified. This indicates to CICS that the root activity's processing is complete, and that it no longer wants to be reattached if defined or system events occur. The business transaction ends.

Transferring input and output data

This section illustrates how to transfer data between a parent and a child activity. It uses the Sale application's Delivery activity as an example.

The SAL002 root activity creates the Delivery child activity by issuing a DEFINE ACTIVITY command.

```
Delivery-Activity.
    .
    EXEC CICS DEFINE ACTIVITY('Delivery')
                 TRANSID('SDEL')
                 EVENT('Delivry-Complete')
              RESP(data-area) RESP2(data-area) END-EXEC
    .
    EXEC CICS GET CONTAINER(Order-Container)
                  ACTIVITY('Order') INTO(Order-Buffer)
              RESP(data-area) RESP2(data-area) END-EXEC

    EXEC CICS PUT CONTAINER(Order-Container)
                  ACTIVITY('Delivery') FROM(Order-Buffer)
              RESP(data-area) RESP2(data-area) END-EXEC
```

Figure 20. Creating the Delivery activity

The GET CONTAINER command retrieves the data returned by the Order activity, and places it in a storage buffer. The data is retrieved from the Order activity's output data-container, which is named *Order*.

The PUT CONTAINER command associates a data-container (also named *Order*) with the Delivery activity, and places the retrieved data in it.

The implementation of the Delivery activity is shown in Figure 21:

```
Identification Division.
Program-id. DEL001.
Environment Division.
Data Division.
Working-Storage Section.
01  Event-Name                  pic x(16).
    88  DFH-Initial             value 'DFHINITIAL'.
1  Order-Ptr               usage is pointer.
01  Order-Container         pic x(16) value 'Order'.
01  Delivery-Container      pic x(16) value 'Delivery'.
01  Deliver-Data.
    .
Linkage Section.
01  DFHEIBLK.
    .
01  Order-Details.
    05  Order-Number        pic 9(8).
    .
```

Figure 21. Pseudocode for the Delivery activity (Part 1 of 2)

```
Procedure Division.
Begin-Process.
    .
    EXEC CICS RETRIEVE REATTACH EVENT(Event-Type)
             RESP(RC) END-EXEC
    .
    If RC NOT = DFHRESP(NORMAL)
      .
    End-If.
    .
    Evaluate True
      When DFH-Initial
        Perform Delivery-Work
        Perform End-Activity
      When Other
        .
    End Evaluate.
    .
    EXEC CICS RETURN END-EXEC
    .
Delivery-Work.
    .
    EXEC CICS GET CONTAINER(Order-Container) SET(Order-Ptr)
             RESP(data-area) RESP2(data-area) END-EXEC
    .
    set address of Order-Details to Order-Ptr.
    .
    EXEC CICS READ FILE .....
              RESP(data-area) RESP2(data-area) END-EXEC
    .
    .  logic to print delivery details
    .
    .
    EXEC CICS PUT CONTAINER(Delivery-Container) FROM(Delivery-Data)
             RESP(data-area) RESP2(data-area) END-EXEC
    .
End-Activity.
    .
    EXEC CICS RETURN ENDACTIVITY
             RESP(data-area) RESP2(data-area) END-EXEC
```

Figure 21. Pseudocode for the Delivery activity (Part 2 of 2)

The Delivery activity issues a GET CONTAINER command to retrieve data from a data-container named *Order*. Because the command does not specify the ACTIVITY option, it references a data-container associated with the current activity; in other words, it references the same *Order* data-container as that referenced by the PUT CONTAINER command in Figure 20 on page 107.

The Delivery activity uses the input data to execute its logic. Then it issues a PUT CONTAINER command to store its output in a data-container named *Delivery*. Again, the ACTIVITY option is not specified, so the data-container is associated with the current (Delivery) activity.

CICSPlex SM support

CICSPlex SM supports BTS by providing:

Dynamic routing of CICS BTS activities

The workload management function of CICSPlex SM controls the dynamic routing of BTS processes and activities within a BTS-set. The distributed routing model is used.

Managing CICS BTS workloads

CICSPlex SM workload management allows you to:

- Separate workloads on the basis of process type. For example, you could specify that workload management is to route all processes of process type TRAVEL to one BTS-set, and all processes of process type PAYROLL to another BTS-set.
- Balance workloads by routing CICS BTS activities to the most appropriate region in the BTS-set, based on:
  - Any affinities its associated transaction has with a particular region.
  - Any workload separation specified by the system programmer.
  - The availability of regions.
  - The relative workload of regions.
- Handle transaction affinities.

  Workload management and the CICS Transaction Affinities Utility understand affinities between BTS processes and activities. CICS BTS itself does not introduce affinities, and discourages programming techniques that do, but it does support previous releases, which may introduce affinities. You must define such affinities to workload management, so that it can make sensible routing decisions. It is particularly important to specify each affinity's lifetime; failure to do so may restrict unnecessarily workload management's routing options.

You should be aware that:

- A single CICSPlex SM may control routing within multiple BTS-sets. It cannot route activities *between* BTS-sets.
- Workload separation may be performed at two levels:
  - By creating multiple BTS-sets
  - By CICSPlex SM within a BTS-set

Changes to workload management views to support BTS are:

- A new view, WLMATAFD, which provides a detailed display of the properties of a single active BTS affinity. You can use this view to display the contents of the BTS affinity key in hexadecimal format.
- The TRANGRP and WLMSPEC views have been enhanced to allow you to specify the BTS BAPPL affinity relation and the BTS ACTIVITY and PROCESS affinity lifetimes

Distributed data

If you write your own routing program, you may wish to keep track of the relative loads in the target regions and use this information in your routing decision. In the hub model, with a single routing region, such data may be maintained in a local temporary storage queue. However, in the distributed model, this data must be held on a shared resource, and distributed, as all the regions are potential routing regions and target regions. CICSPlex SM handles routing data by using MVS data spaces. If more than one MVS

image is involved in a workload, CICSPlex SM usually attempts to keep the data spaces in step. If you use the CICSPlex SM routing program EYU9XLOP, you will not have to be concerned with the distributed nature of the routing data, because CICSPlex SM manages this for you.

BAS support

The only CICS BTS resource you need to define to BAS is the *process type*, using the PROCDEF view. All other CICS BTS resources are created dynamically when you specify a name in your application. Each process type definition is associated with a VSAM KSDS file definition and a journal definition. The file is used to hold process activity state data. The journal is used to hold process activity audit data.

Connections between regions in a BTS-set are handled by BAS in the same way as connections between regions in the CICSplex. You define the connection and session, and the links are created dynamically when required. This reduces the number of link definitions that you need to define for the BTS-set, compared with the number required by CEDA.

Operations support

You can control the CICS BTS processes and activities running in your BTS-set through a set of operate views:

PROCTYP General view of all installed process types and their attributes.

PROCTYPD Detailed view of the selected process type and its attributes.

PROCTYPS Summary view of all installed process types and their attributes.

New resource tables

The following resource tables are introduced to support CICS BTS:

- CRESPRTY
- ERMCPRTY
- PRCINGRP
- PROCDEF
- PROCTYP

Benefits of using CICSPlex SM to manage CICS BTS

You do not have to use CICSPlex SM workload management to route BTS activities, but using CICSPlex SM can offer many benefits:

- Management of distributed data
- Workload separation and balancing functions of workload management
- Customizable dynamic routing program EYU9XLOP
- Reduction in link definitions if you use BAS and CICS BTS
- Cooperation between BAS and CICS BTS in the management of your business environment.

Chapter 9. Open transaction environment

This chapter describes changes to CICS internal architecture to provide an open transaction environment (OTE) that supports mutiple task control blocks (TCBs). It covers the following topics:

Overview

In earlier releases of CICS, user applications operate in a restricted, or "closed", environment. Although they can use the functionally-rich CICS programming interfaces (see "Permitted programming interfaces under the QR TCB" on page 114), direct invocation of other services is not supported. This is because CICS runs user transactions under a single MVS TCB, known as the CICS quasi-reentrant (QR) TCB. Direct invocation of other services outside the scope of the CICS permitted interfaces is not supported, because such actions could interfere with CICS own use of its QR TCB. In particular, services which result in the suspension ("blocking") of the QR TCB would cause all CICS tasks to wait.

Other subsystems, such as DB2, MQSeries®, and IMS DBCTL, can be invoked indirectly through the CICS resource manager interface, using task-related user exits (also referred to as adapters). These task-related user exits are themselves restricted to the closed CICS environment when running under the CICS QR TCB, and hence they are required to manage a private set of MVS TCBs in order to access non-CICS function (such as MVS services). However, having to switch to private TCBs incurs a performance overhead. This is because, when a QR program calls a task-related user exit (such as DB2 or MQSeries) that performs its processing under a non-QR TCB, two TCB switches are incurred: (1) from the QR TCB when the task-related user exit is invoked, and (2) back to the QR TCB on return from the task-related user exit. Also, switching between a private TCB and the QR TCB adds to the complexity of writing a task-related user exit.

The restrictions that affect RMI adapters would also prevent CICS from offering full support for all the core Java classes. Java application programs developed and compiled using a Java compiler (such as VisualAge for Java or javac) are restricted to the function provided by the CICS Java classes (JCICS). The normal CICS program execution model is used for these Java application programs, which means they are run by CICS under the QR TCB. For information about this type of CICS Java application program, see "Chapter 14. VisualAge for Java, Enterprise Edition for OS/390" on page 151.

The future

To remove the restrictions that allow user application programs to use only the programming interfaces that are permitted to run under the QR TCB, CICS introduces a new type of TCB (see "Open TCBs" on page 112). These open TCBs provide support for OS/390 Java Virtual Machines (JVMs) invoked by CICS, and

which run as part of the CICS address space. Open TCB support for JVMs is provided immediately; see “Chapter 15. Support for the Java Virtual Machine” on page 155 for information about CICS support for JVMs.

In the longer term, it is IBM’s intention to extend the use of this new type of TCB to provide a fully open environment in which user applications can operate without today’s restrictions. In particular, task-related user exits, will be able to exploit OTE support to reduce some of the overhead and to simplify the programming.

Open TCBs

The open transaction environment extends the TCBs used by CICS to include a new type of TCB called an **open** TCB. A task running under its own open TCB is allowed to invoke previously restricted services, such as blocking services, because it does not interfere with the QR TCB, or cause it to wait. For example, invoking an OS/390 JVM under a unique open TCB allows the JVM to run a user Java application program that uses Java classes such as **java.io**, **java.net**, and **java.awt** (including windowing support).

Similarly, the change to CICS internal architecture for multiple TCB support will allow task-related user exits to invoke non-CICS services. They will then benefit from the open transaction environment because they will not need to manage their own private set of TCBs. This will result in simplified systems management for the CICS system administrator, and provide performance improvement. A user application program that is able to run under an open TCB will benefit, because, when CICS returns control to it following an RMI call, CICS need not switch to the QR TCB. The associated user task can continue executing on the unique open TCB under which CICS invoked the task-related user exit and need not incur the overhead of a TCB switch.

The open TCB modes supported in CICS TS Release 3 are as follows:

J8 A pool of open TCBs, from which a TCB is allocated for each invocation of a JVM. The mode identification, J8, signifies a TCB mode reserved for JVMs running in key 8 (CICS key). J8 TCBs are fully supported in CICS TS Release 3, and you can see information about the use of these in CICS dispatcher statistics.

L8 A pool of open TCBs, from which one TCB is allocated for a task-related user exit program that is defined to support open TCBs. The mode identification, L8, signifies a TCB mode reserved for CICS user tasks that use LE services, running in key 8 (CICS key). The use of L8 TCBs is disabled in CICS TS Release 3, but will be enabled in a future release. CICS dispatcher statistics will always show a zero count until such time that L8s are enabled for use.

You specifiy the maximum number of all open TCBs that a CICS region can use on the MAXOPENTCBS system initialization parameter (see “Changes to system definition” on page 118 for details).

IBM plans to roll out full support for open TCBs over more than one release of CICS TS. The information provided now is intended to help you prepare your application programs, task-related user exits, and global user exit programs to exploit the CICS open transaction environment as more and more OTE support is enabled.

Using an open TCB

CICS will determine whether your application programs can take advantage of an open TCB according to whether they are defined as **threadsafe** (fully reentrant) or only **quasi-reentrant**.

Quasi-reentrant programs

In the CICS quasi-reentrant environment, all user programs (application programs, user-replaceable modules, global user exits, and task-related user exits), in general, run under a single TCB—the QR TCB. When a program is given control by the CICS dispatcher, it can be sure that no other user program can run until it relinquishes control during a CICS request, at which point the user task is suspended, leaving the program still"in use". The same program could then be reinvoked for another task, which means an application program can be in use concurrently by more than one task, although only one task at a time can actually be executing (see *quasi-reentrant* in "Definition of terms" on page 247).

To ensure that user programs cannot interfere with the working storage of other user programs, CICS obtains a separate copy of working storage for each execution of an application program. Thus, if a user application program is in use by 11 user tasks, there are 11 copies of working storage in the appropriate dynamic storage area (DSA).

Quasi-reentrancy allows programs to access shared virtual storage—for example, the CICS common work area (CWA)—without the need to protect that virtual storage from concurrent access by other programs. The shared virtual storage is dedicated exclusively to the running program, until it issues its next CICS request. Thus, for example, an application can read and subsequently update a field in the CWA between CICS commands without using compare and swap (CS) instructions or locking (enqueuing on) the resource.

Note: The CICS QR TCB provides protection through exclusive control of shared virtual storage only if *all* user tasks that access that virtual storage run under the QR TCB. It does not provide automatic protection from other tasks that execute concurrently under another (open) TCB.

Threadsafe programs

In an OTE environment, threadsafe application programs and open task-related user exits, global user exit programs, and user-replaceable modules cannot rely on quasi-reentrancy, because they may run concurrently on open TCBs. Furthermore, even quasi-reentrant programs are at risk if they access shared virtual storage that can also be accessed by a user task running concurrently under an open TCB. This means that the techniques used by user programs to access shared resources must take into account the possibility of simultaneous access by other programs. Programs that use appropriate serialization techniques when accessing shared virtual storage are described as **threadsafe**. (The term **fully reentrant** is also used sometimes, but this can be misunderstood, hence threadsafe is the preferred term.) For all CICS-managed resources, such as files, transient data queues, temporary storage queues, and DB2 tables, CICS processing automatically ensures access in a threadsafe manner. However, for any other resources, such as shared storage, which are accessed directly by user programs, it is the responsibility of the user program to ensure threadsafe processing. Typical examples of shared storage are the CICS CWA, global user exit global work areas, and storage acquired by EXEC CICS GETMAIN SHARED commands.

Note: When identifying programs that use shared resources, you should also include any program that modifies itself. Such a program is effectively sharing storage and should be considerd at risk.

Techniques that you can use to provide threadsafe processing when accessing a shared resource are as follows:

- Retry access, if the resource has been changed concurrently by another program, using the compare and swap instruction
- Enqueue on the resource, to obtain exclusive control and ensure that no other program can access the resource, using:
  - An EXEC CICS ENQ command, in an application program
  - An XPI ENQUEUE function call to the CICS enqueue (NQ) domain, in a global user exit program
- Always perform accesses to shared resources in a quasirent program, and linking to this using the EXEC CICS LINK command.

  This technique applies to threadsafe application programs and open API task-related user exits only. A linked-to program defined as quasi-reentrant runs under the QR TCB and can take advantage of the serialization provided by CICS quasi-reentrancy. Note that even in quasi-reentrant mode, serialization is provided only for as long as the program retains control and does not wait (see “Quasi-reentrant programs” on page 113 for more information).
- Place all transactions that access the shared resource into a restricted transaction class (TRANCLASS), one that is defined with the number of active tasks specified as MAXACTIVE(1).

  This last approach effectively provides a very coarse locking mechanism, and may have a severe impact on performance.

Note: Although the term threadsafe is defined in the context of individual programs, a user application as a whole can only be considered threadsafe if *all* the application programs that access shared resources obey the rules. A program that is written correctly to threadsafe standards cannot safely update shared resources if another program that accesses the same resources does not obey the threadsafe rules.

Permitted programming interfaces under the QR TCB

The services available to user-written programs that run under the CICS QR TCB (application programs, user-replaceable modules (URMs), PLT programs, task-related user exits, and high-level global user exits) are those provided by:

- The **CICS command-level application programming interface (API)**, which is documented in the *CICS Application Programming Reference*.
- The **CICS system programming interface (SPI)**, which is documented in the *CICS System Programming Reference*.
- The **CICS resource manager interface (RMI)**, which is documented in the *CICS Customization Guide*. in the chapter entitled “Task-related user exit programs”. The RMI allows other products to provide a task-related user exit that interfaces with their code running in another address space. User applications can use the RMI to access resource managers such as DB2, DBCTL, and MQSeries.
- The **CICS exit programming interface (XPI)**, which is documented in the *CICS Customization Guide*. The XPI, which is available only within global user exit programs, allows user-written exit programs to have access to certain CICS services.

- **Systems application architecture (SAA) Common Programming Interfaces (CPI)**. The SAA CPIs supported by CICS TS are the common programming interface for communications (CPI-C) and the common programming interface for resource recovery (CPI-RR).
- **LE callable services**, which are listed in the *Language Environment for OS/390 & VM Concepts Guide*, GC28-1945.

Program attributes for the open transaction environment

The open transaction environment provides a new attribute for user application programs, global user exit programs, user-replaceable modules (URMs), and PLT programs. This new attribute, CONCURRENCY, indicates the standard, or level, of reentrancy to which the program is written. The value you define for CONCURRENCY (QUASIRENT or THREADSAFE) indicates to CICS whether the program must run under the QR TCB, or whether it is written to threadsafe standards and can run under an open TCB. For details of the program CONCURRENCY attribute, see “Changes to resource definition” on page 119.

Non-threadsafe programs will require the QUASIRENT attribute (for an explanation of what makes a program threadsafe, see “Quasi-reentrant programs” on page 113). Existing programs, including existing task-related user exits, will run unchanged as quasi-reentrant programs, which is the default.

A program defined as THREADSAFE will run under either the QR TCB, or an open TCB, whichever the user task is running under at the time the program is given control. In CICS TS Release 3, threadsafe application programs will continue to be restricted to the CICS programming interfaces (see “Permitted programming interfaces under the QR TCB” on page 114). As well as being threadsafe, they must not have any TCB affinity[4], but this is assured because compliance with the CICS permitted programming interfaces avoids TCB affinity. A threadsafe program must be Language Environment-conforming[5], or an assembler program.

Adherence to these restrictions will enable CICS to invoke, or return control to, a threadsafe program under whichever TCB the user task is running. CICS will not first need to switch to the QR TCB as for a quasi-reentrant program. This will provide a performance benefit in the future when using task-related user exits that exploit OTE.

Program attributes for task-related user exits

Table 7 on page 116 shows the concurrency attributes that you can specify for a task-related user exit program, depending on whether the program:

- Is threadsafe

4. TCB affinity occurs when a program has a dependency on a specific TCB, and is only able to execute correctly on that TCB. This can occur when a program issues calls to non-CICS API services that leave the TCB, or dependent control blocks, in a state on which the program relies whenever control it is returned to it after an interrupt.

5. Language Environment-conforming program. An application program that has been compiled by a Language Environment-conforming compiler. These compilers are: (1) IBM OS/390 C/C++ (2) IBM SAA AD/Cycle® COBOL/370™ (3) IBM COBOL for OS/390 & VM (4) IBM COBOL for MVS & VM (formerly COBOL/370) (5) IBM SAA AD/Cycle PL/I MVS & VM (6) IBM PL/I for MVS & VM (formerly PL/I MVS & VM) and, (7) IBM OS/390 VisualAge for Java, Enterprise ToolKit for OS/390. LE-conforming programs cannot run without Language Environment and can take advantage of all Language Environment facilities. In contrast, compatibility-mode application programs are compiled with older compilers, whose executable output is only partially supported by LE/370.

- Uses non-CICS programming interfaces (apart from the MVS ATTACH and POST that it has to use, under the QR TCB, to manage its own TCBs)

Table 7. Defining attributes for task-related user exit programs

Threadsafe?	Uses non-CICS interfaces?	Description of the task-related user exit program
No	No	It must be defined as quasi-reentrant. It is called under the CICS-managed QR TCB, and does not use any private TCBs.
No	Yes	It must be defined as quasi-reentrant. It is called under the CICS-managed QR TCB, and must provide private TCBs in order to use non-CICS interfaces.
Yes	No	It can be defined as threadsafe. It is called under a CICS-managed TCB (not necessarily the QR TCB) and does not use any private TCBs.
Yes	Yes	It can be defined as: • **Threadsafe:**It provides private TCBs in order to use non-CICS interfaces. It is called under a CICS-managed TCB, which could be an open TCB or the QR TCB. • **Threadsafe and open API:** It does not need private TCBs. Will be called by CICS under a CICS-managed open TCB, mode L8.

If the task-related user exit does not support the OPENAPI option on the enable command, it should switch to a privately-managed TCB if it needs to use non-CICS API services. Thus, through a combination of the program resource definition and options on the ENABLE command, a task-related user exit will be:

- Quasi-reentrant, in which case it must switch to a privately-owned TCB in order to use services other than those provided by the CICS permitted programming interfaces
- Threadsafe only, in which case it will run under the invoking TCB
- Threadsafe and also an open API task-related user exit, in which case CICS will invoke the task-related user exit under an open TCB.

The use of these attributes

To enable a task-related user exit as an open API task-related user exit, you will specify the OPENAPI option on the enhanced EXEC CICS ENABLE command (see “Changes to the system programming interface (SPI)” on page 121). If a task-related user exit is enabled in this way, CICS will invoke it under an open TCB obtained from a CICS-managed pool of open TCBs. Running under its own open TCB means that the task-related user exit will not need to perform a TCB switch in order to invoke non-CICS API services. It will therefore be able to take advantage of reduced TCB switching and gain a performance advantage when invoked by threadsafe programs.

Task-related user exits that do not support the OPENAPI option will manage their own pools of TCBs, as currently, switching to one of their own TCBs to use function outside the permitted programming interfaces.

A user task invoking an open API task-related user exit will be given its own dedicated open TCB for the lifetime of the task. This open TCB will be shared by:

- All open API task-related user exits invoked under the user task
- All the user task's threadsafe programs, unless they are given control under the QR TCB.

When a user task first invokes an open API task-related user exit and acquires its open TCB, it becomes an "open task".

An open API task-related user exit must be threadsafe and must be able to run its requests directly under the open TCB. It may have TCB affinity to the open TCB under which it runs.

IBM plan to enable the DB2 adapter to exploit the open transaction environment in a future deliverable.

Benefits

The benefits provided by the CICS open transaction environment facility will be as follows:

Performance improvements
When the DB2 adapter is enabled for OTE, user transactions that access DB2 resources, and which are threadsafe and Language Environment-conforming (or written in assembler) should achieve improved performance when measured against earlier releases of CICS and DB2.

Improved RMI
New options on the EXEC CICS ENABLE command will allow a task-related user exit to notify CICS whether to invoke it under the CICS QR TCB or an open TCB, obtained from a CICS-managed pool of open TCBs, dedicated to the user task. When task-related user exits are able to request usage of an open TCB, the following benefits will apply:

Performance improvements
Task-related user exit programs will be able to avoid the performance overhead of switching to their own privately-managed TCBs. To do this, they will have to be threadsafe and will have to perform their function on the TCB under which CICS calls them (without the need to perform some extra function that negates the performance gains made by not switching TCBs).

Easier to write new task-related user exits in future releases
It is planned that the open TCB attribute will become available to *all* task-related user exits. It will then be possible to write threadsafe task-related user exits that do not need to manage their own set of TCBs, in which case they will benefit from the above performance advantages.

Simpler systems management
Task-related user exits that manage their own TCBs can require their users to become involved in the systems management of these TCBs. This requirement could be removed when task-related user exits are enabled to exploit open TCBs.

Additional function available
Task-related user exits will be able to use the CICS permitted programming interfaces and additional function supported for application use while running under a CICS-managed open TCB.

Task-related user exits that are able to run under the caller's TCB will benefit by avoiding TCB switches to and from the QR TCB, when invoked by the CICS RMI running under an open TCB.

Improved problem determination
Information relating to open TCBs appears in CICS trace entries, messages, statistics, monitoring data, and dumps. Without OTE support, the equivalent TCBs are in private pools owned by individual task-related user exits, and are therefore not so visible.

These benefits will be available with the full support of existing CICS functions such as security, storage protection, and transaction isolation.

Requirements

The hardware and software required for the first phase of OTE support enabled in CICS TS Release 3 is the same as for CICS TS 3 generally.

Changes to CICS externals

There are changes to a number of CICS externals in readiness for the enablement of the open transaction environment. These are:

- "Changes to system definition"
- "Changes to resource definition" on page 119
- "Changes to the system programming interface (SPI)" on page 121
- "Changes to global user exits" on page 127
- "Changes to task-related user exits" on page 128
- "Changes to the exit programming interface (XPI)" on page 130
- "User-replaceable modules" on page 132

Changes to system definition

The additions to CICS system initialization parameters to support OTE are shown in Table 8.

Table 8. DFHSIT macro parameters

	DFHSIT	[TYPE={**CSECT**]DSECT}]
		,...
		,MAXOPENTCBS={**5**]number}
		,FORCEQR={**NO**]YES}
		,...
	END	DFHSITBA

MAXOPENTCBS=5|*number*
Specifies the maximum number, in the range 1 to 999, of open TCBs (J8s only initially) that can exist concurrently in the CICS region. When specifying this value, take into account TCB storage requirements: TCBs use real storage, and virtual storage below 16MB.

The default value is **5** open TCBs.

CICS manages a pool of open TCBs up to the limit set by MAXOPENTCBS. At any one time, the pool can consist of some TCBs that are allocated, and others that are free. CICS attaches a new TCB only when there isn't a free TCB available in the pool. For example, if the maximum number of open TCBs is set at 100, the pool could consist of 50 open TCBs, not all of which are allocated.

FORCEQR={NO]YES}
Specifies whether you want CICS to force all user application programs specified as CONCURRENCY(THREADSAFE) to run under the CICS QR TCB, as if they were specified as CONCURRENCY(QUASIRENT) programs. This parameter applies to application programs that are restricted to the current CICS programming interfaces, and therefore does not apply to Java programs that are run in a JVM.

FORCEQR=YES will allow you, in a test environment, to run incompletely tested threadsafe application programs that have proved to be non-threadsafe.

FORCEQR will apply to all programs defined as threadsafe that are not invoked as task-related user exits, global user exits, or user-replaceable modules.

Changes to resource definition

A CONCURRENCY option is added to the program resource definition, as follows:

CONCURRENCY(QUASIRENT]THREADSAFE)
specifies whether the program is written to threadsafe or quasi-reentrancy standards.

The CONCURRENCY option is applicable to all CICS executable program objects:
- User application programs
- PLT programs
- User replaceable modules (URMs)
- Global user exits
- Task-related user exits

However, there are special considerations for URMs, task-related user exits and global user exits. See "User-replaceable modules" on page 132, "Changes to task-related user exits" on page 128 and "Changes to global user exits" on page 127 for details of these.

QUASIRENT
specifies that the program is quasi-reentrant only, and relies on the serialization provided by CICS single threading when accessing shared resources. QUASIRENT is the default.

The program is restricted to the permitted programming interfaces, and must comply with the CICS quasi-reentrancy rules. For details of these, see the multithreading topic in the *CICS Application Programming Guide*.

This option is supported for all executable programs.

CICS will ensure that the program always executes under the QR TCB, even when control is returned after it has invoked an open API task-related user exit, or when it interacts with threadsafe programs.

To ensure compatibility when migrating from an earlier release of CICS, CONCURRENCY(QUASIRENT) is the default, and should be specified for traditional CICS application programs.

THREADSAFE
specifies that the program is threadsafe, and when it accesses shared non-CICS-managed resources it takes into account the possibility that other programs may be executing concurrently and attempting to modify the same resources. It uses appropriate serialization techniques when accessing any non-CICS-managed shared resources.

A threadsafe program will be able to run under whichever TCB CICS invokes it, either the QR TCB or an open TCB, which could be:

- An open TCB of mode L8, in which case it will continue to be restricted to the CICS permitted programming interfaces. Compliance with these rules ensures that the program has no TCB affinity.
- An open TCB of mode J8, allocated for Java programs defined to run under a JVM, in which case the CICS restrictions do not apply.

This option is supported for all executable programs. Threadsafe programs must be Language Environment-conforming, or be assembler programs.

In practice, CICS will not switch TCBs for threadsafe programs between invocations of the permitted programming interfaces. However, on return from invoking a service through the permitted programming interfaces, a user task could receive control under a different TCB, as follows:

- An EXEC CICS request will generally return control under the QR TCB.

  Exceptions to this general rule will be where the CICS internal request processing has been enhanced to be threadsafe (in which case CICS will return control under the invoker's TCB, thus possibly avoiding a TCB switch).
- An invocation of a CONCURRENCY(QUASIRENT) task-related user exit will return control under the QR TCB (possibly after switching to and from the task-related user exit's privately-owned TCB to use services outside the scope of the permitted programming interfaces).
- An invocation of an open API task-related user exit will return control under the task-related user exit's open TCB.
- If a threadsafe program issues an EXEC CICS HANDLE command that specifies the LABEL option, and an event causes a branch to the label, the user code could be resumed at the specified label under a different TCB from that under which the program was running when the event was triggered. The CICS HANDLE table includes the ID of the TCB under which the EXEC CICS HANDLE was issued, therefore it will be possible to identify in a dump the TCB in use prior to label processing.

For an executing CONCURRENCY(THREADSAFE) program, floating point registers, access registers (other than 0, 1, 14, 15), and program mask will remain unchanged by CICS between invocations of the permitted programming interfaces, as for QUASIRENT-defined programs.

CONCURRENCY(THREADSAFE) can be regarded as a performance option, to be used by application programs to reduce TCB switching, in conjunction with open API task-related user exits. You will be able to specify THREADSAFE for existing CICS applications that you know are threadsafe. You are recommended to comply with the rules for threadsafe programs

when developing new application programs, so that they can take advantage of the benefits of OTE as the support becomes available.

You can specify the program CONCURRENCY option in the following ways:

- On a CSD DEFINE PROGRAM command, using either a CED*x* transaction or the DFHCSDUP utility program.
- On an EXEC CICS CREATE command.
- Using a program autoinstall exit, if program autoinstall is active.

If CONCURRENCY is not specified by any method, it defaults to QUASIRENT.

Changes to the system programming interface (SPI)

The open transaction environment introduces changes to the following SPI commands:

- CREATE PROGRAM
- ENABLE
- INQUIRE EXITPROGRAM
- INQUIRE PROGRAM
- INQUIRE SYSTEM
- INQUIRE TASK
- SET SYSTEM

New options supported by EXEC CICS CREATE PROGRAM

The EXEC CICS CREATE command is extended to enable you to define the CONCURRENCY attribute for the program definition you are installing. The ATTRIBUTES string on the CREATE PROGRAM command can now include one of the following options:

- CONCURRENCY(QUASIRENT)
- CONCURRENCY(THREADSAFE)

For information about these program attributes, see “Changes to resource definition” on page 119.

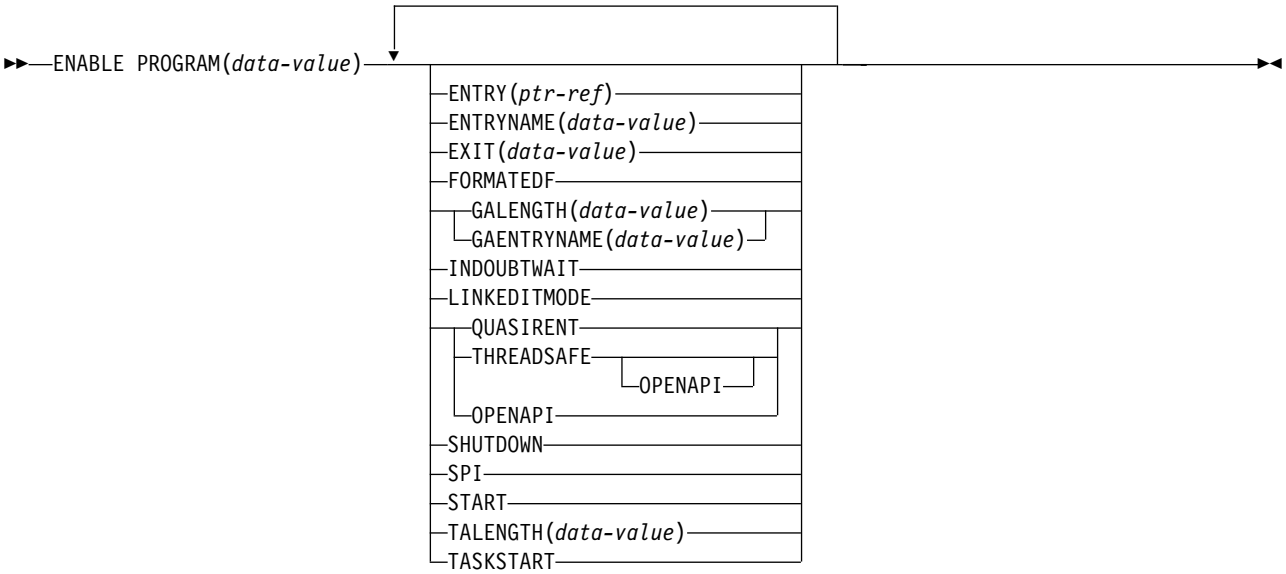
New options on EXEC CICS ENABLE command

The EXEC CICS ENABLE command is extended by the addition of new options that will allow you to:

- Override the CONCURRENCY attribute defined on the PROGRAM resource definition for a task-related user exit program.
- Specify whether the task-related user exit is restricted to the CICS permitted programming interfaces, or can also use non-CICS API services.

The full syntax of the enhanced ENABLE command is as follows:

ENABLE PROGRAM



Conditions: INVEXITREQ, NOTAUTH

The new options on the EXEC CICS ENABLE command are as follows:

QUASIRENT]THREADSAFE(task-related user exits only)
specifies whether the task-related user exit program is written to threadsafe standards, or is only quasi-reentrant. If specified, these options override the CONCURRENCY attribute set by the PROGRAM resource definition for the task-related user exit program (see “Changes to resource definition” on page 119).

Note: The concurrency attribute on the installed program resource definition remains unchanged, because the attributes for a task-related user exit are held in its own exit program control block, the information for which is derived from a combination of the program resource definition and the ENABLE command.

These options remain in force until changed by another ENABLE command—there are no equivalent options on the DISABLE command. If, having enabled a task-related user exit, the adapter component that starts the connection finds that its resource manager does not support the options specified, the adapter must issue another ENABLE command to re-specify the concurrency attribute. There is no SPI option to specify that the CONCURRENCY attribute defined on the task-related user exit's program resource definition is to be reinstated.

If none of a task-related user exit's ENABLE commands specify a THREADSAFE or QUASIRENT option, the concurrency is derived from the program resource definition. Note that the value is not affected by the FORCEQR system initialization parameter, which does not apply to task-related user exits.

The CONCURRENCY values that can be specified on the ENABLE command are as follows:

QUASIRENT specifies that the task-related user exit program is quasi-reentrant, and relies on the serialization provided by CICS when accessing shared resources.

The task-related user exit program is restricted to the permitted programming interfaces, and must comply with CICS quasi-reentrancy rules. CICS always invokes a quasi-reentrant task-related user exit under the QR TCB. If the task-related user exit uses non-CICS API services, it must switch to its own TCB before issuing calls to these services, and switch back again before returning to its caller.

THREADSAFE
specifies that the task-related user exit program is threadsafe, and will take into account the possibility that, when accessing shared resources, other programs may be executing concurrently and attempting to modify the same resources. It will use appropriate serialization techniques when accessing any shared resources.

A threadsafe task-related user exit must be able to run under whichever TCB CICS invokes it. If OPENAPI is also specified, CICS will generally invoke the task-related user exit under an L8 open TCB.

OPENAPI(task-related user exits only)
specifies that the task-related user exit is using non-CICS APIs and wishes to do so under a CICS open TCB. CICS will always call the task-related user exit under an open TCB.

Note: If OPENAPI is specified without THREADSAFE, CICS enforces THREADSAFE by default.

These new options are supported by EDF and CECI.

New options on EXEC CICS INQUIRE EXITPROGRAM command

The EXEC CICS INQUIRE EXITPROGRAM command is extended to allow you to obtain additional information about a task-related user exit. The new options provided for this purpose are:

CONCURRENST(*cvda***) (**task-related user exit only**)**
returns a CVDA indicating the concurrency status of the task-related user exit program, specified by the latest ENABLE command for this program.

CVDA values are:

QUASIRENT
The task-related user exit program is defined as being quasi-reentrant, and is able to run only under the CICS QR TCB when invoking CICS services through the CICS API. To use any non-CICS API services, this task-related user exit must switch to a privately-managed TCB.

THREADSAFE
The program is defined as threadsafe, and is capable of running under any TCB. If the APIST option returns OPENAPI, it will always be invoked under an open TCB. If the APIST option returns BASEAPI, it is

invoked under whichever TCB is in use by its user task when the program is given control, which could be either an open TCB or the CICS QR TCB.

APIST(*cvda***) (**task-related user exit only**)**
returns a CVDA indicating which APIs the task-related user exit program uses.

CVDA values are:

BASEAPI
The task-related user exit program is enabled as either QUASIRENT or THREADSAFE, but without the OPENAPI option. This means it is restricted to the CICS permitted programming interfaces

OPENAPI
The task-related user exit program is enabled with the OPENAPI option. This means it will be permitted to use non-CICS API in a threadsafe manner, for which purpose CICS gives control to the task-related user exit under an open TCB.

The above new options are supported by EDF and CECI. CEMT does not support INQUIRE EXITPROGRAM.

New options on EXEC CICS INQUIRE PROGRAM command

The EXEC CICS INQUIRE PROGRAM command is extended to allow you to obtain information about a program's concurrency as defined in the program resource definition. The new option provided for this purpose is:

CONCURRENCY(*cvda***)**
returns a CVDA indicating the concurrency status of the application program. The CVDA values are:

QUASIRENT
The program is defined as being quasi-reentrant, and is able to run only under the CICS QR TCB.

THREADSAFE
The program is defined as threadsafe, and is able to run under whichever TCB is in use by its user task when the program is given control. This could be either an open TCB or the CICS QR TCB.

Notes:

1. If the program is not yet loaded (or is waiting to be reloaded following a NEWCOPY or PHASEIN request), the concurrency attribute is derived from the installed program resource definition. Note that the default for the program definition is QUASIRENT. However, in the case of a Language Environment-conforming program, the concurrency as originally defined can be overridden when the program is subsequently loaded. If CICS finds that the program itself contains a CONCURRENCY value defined by LE run-time options, the installed program resource definition is updated by the LE run-time option.
2. The CONCURRENCY attribute on the installed program resource definition is not changed by the FORCEQR system initialization parameter. CICS returns a CVDA of THREADSAFE for a threadsafe-defined program, even if FORCEQR=YES is specified.
3. The CONCURRENCY attribute on the installed program resource definition for a task-related user exit program is not changed by any attributes specified on an ENABLE command. The CONCURRENCY attribute on an

ENABLE command affects only the use of the program established by the ENABLE. For a task-related user exit program, CICS always returns a CVDA using the values defined in the program resource definition.

You cannot modify a program's concurrency attribute using the SPI—the CONCURRENCY option is not supported on the EXEC CICS SET PROGRAM command. You can only change the concurrency by on of the following methods:

- Redefine the program's CONCURRENCY option in the CICS program resource definition, or in the program autoinstall model, and reinstall the definition
- Recompile and link-edit the program with new LE run-time options, and issue a CEMT, or EXEC CICS, SET PROGRAM(...) NEWCOPY (or PHASEIN) command.

The above new options are supported by EDF, CECI and CEMT.

New options on EXEC CICS INQUIRE SYSTEM command

The EXEC CICS INQUIRE SYSTEM command is extended to allow you to obtain information about open TCBs, and whether quasi-reentrancy is being forced for all application programs. The new options are:

MAXOPENTCBS(*data-area***)**
returns, as a fullword binary value, the maximum number of open TCBs currently allowed. For information about the MAXOPENTCBS system initialization parameter, see "Changes to system definition" on page 118.

ACTOPENTCBS(*data-area***)**
returns, as a fullword BINARY value, the number of open TCBs currently allocated.

FORCEQR(*cvda***)**
returns a CVDA value indicating whether quasi-reentrancy is currently being forced for all user application programs defined as threadsafe. The CVDA values are:

FORCE
All user application programs are being forced to execute under the QR TCB, as if they were defined with the CONCURRENCY(QUASIRENT) attribute, even if were defined with CONCURRENCY(THREADSAFE).

NOFORCE
Quasi-reentrancy is not being enforced for all user application programs, and the threadsafe attribute on program definitions is being honored.

The new INQUIRE SYSTEM options are supported by EDF, CECI and CEMT.

New options on EXEC CICS INQUIRE TASK command

The EXEC CICS INQUIRE TASK command has a new option to allow you to obtain information about the TCB under which the user task is running. The new option is:

TCB(*cvda***)**
returns a CVDA value indicating the type of TCB under which the user task is running.

CVDA values are:

CKOPEN
The user task is running under a CICS key open TCB, either mode L8 or J8.

QR The user task is running under the CICS quasi-reentrant TCB.

INTERNAL
The user task is running under one of the other CICS-managed TCBs. This could be the file-owning TCB, resource-owning TCB, the concurrent TCB, the FEPI TCB, the ONC/RPC TCB, or a sockets domain TCB.

The TCB option is supported by EDF, CECI, and CEMT.

New options on EXEC CICS SET SYSTEM command

The EXEC CICS SET SYSTEM command is extended to allow you to modify information relating to the open transaction environment while CICS is running. The new options provided on the SET SYSTEM command are:

MAXOPENTCBS(*value*)
specifies, as a fullword binary value, the maximum number of open TCBs that can exist concurrently in the CICS region. The value specified can be in the range 1 to 999.

If you reduce MAXOPENTCBS from its previously defined value, and the new value is less than the number of open TCBs currently allocated, CICS detaches TCBs to achieve the new limit only when they are freed by user tasks. Transactions are not abended to allow TCBs to be detached to achieve the new limit.

If there are tasks queued waiting for an open TCB, and you increase MAXOPENTCBS from its previously defined value, they will be resumed when CICS attaches new TCBs up to the new limit.

FORCEQR(*cvda*)
specifies whether you want CICS to force all user application programs specified as CONCURRENCY(THREADSAFE) to run under the CICS QR TCB, as if they were specified as CONCURRENCY(QUASIRENT) programs.

This allows you, in a test environment, to run incompletely tested threadsafe application programs that have proved to be non-threadsafe.

FORCEQR applies to all programs defined as threadsafe that are not invoked as task-related user exits, global user exits, or user-replaceable modules.

CVDA values are:

FORCE
All user programs defined as threadsafe are to be forced to execute as quasi-reentrant programs.

NOFORCE
CICS is to honor the CONCURRENCY(THREADSAFE) attribute defined on program resource definitions, and invoke them under either the QR TCB or an open TCB.

The FORCEQR(FORCE]NOFORCE) option allows you to change dynamically the option specified by the FORCEQR system initialization parameter (see “Changes to system definition” on page 118).

Specifying FORCEQR(FORCE) is not applied to currently invoked programs, and applies only to programs invoked for the first time after the change to the FORCEQR status.

The above new options are supported by EDF, CECI and CEMT.

Changes to global user exits

There are changes to the DFHUEPAR standard parameter list:

- The amount of storage addressed by UEPXSTOR is increased.
- Additional task indicators are provided.

UEPXSTOR now points to a 320-byte area of DFHUEH-owned storage that a global user exit program should use when invoking the XPI. This provides up to 256 bytes for the XPI services parameter list, plus an extra 64 bytes for your own purpose.

The DFHUEPAR standard parameter list includes some additional task indicators. The global user exit task indicator field, addressed by UEPGIND, is extended from one byte to three bytes, the second and third bytes containing a value indicating the TCB mode of the global user exit program's caller. This is represented in DFHUEPAR as both a two-character code and a symbolic value, as follows:

Table 9. TCB indicators in DFHUEPAR. Description

Symbolic value	2-byte code	Description
UEPTQR	QR	The quasi-reentrant mode TCB
UEPTCO	CO	The concurrent mode TCB
UEPTFO	FO	The file-owning mode TCB
UEPTRO	RO	The resource-owning mode TCB
UEPTRP	RP	The ONC/RPC mode TCB
UEPTSZ	SZ	The FEPI mode TCB
UEPTJ8	J8	The JVM mode TCB
UEPTL8	L8	An open mode TCB
UEPTSL	SL	The sockets listener mode TCB
UEPTSO	SO	The sockets mode TCB
UEPTS8	S8	The secure sockets layer mode TCB

If you have global user exit programs that are frequently invoked by user tasks that run under an open TCB, you are recommended for performance reasons to ensure that the global user exit programs are fully threadsafe. When you are sure that a global user exit program is fully threadsafe, define it as such by specifying CONCURRENCY(THREADSAFE) on the global user exit's program resource definition (see “Changes to resource definition” on page 119). This indicates that it it can be invoked under its caller's TCB (QR, an open TCB, or any other CICS TCB). Define global user exit programs that are not threadsafe as quasi-reentrant. This causes CICS to switch to the QR TCB before invoking the global user exit program, as in earlier releases.

The XPI is updated to provide serialisation function to make it easier for global user exits to be made threadsafe. See “Changes to the exit programming interface (XPI)” on page 130 for details.

Global user exit programs are restricted to the CICS permitted programming interfaces.

Global user exit recovery

There are no new recovery mechanisms for global user exits. API-capable global user exits can use EXEC CICS HANDLE.

Note that if a global user exit fails while holding an enqueue acquired through the enqueue/dequeue XPI, the enqueue is freed automatically when the acquiring task terminates. See “Changes to the exit programming interface (XPI)” on page 130 for more details.

XPCTA global user exit

An XPCTA global user exit program, invoked when a transaction abend occurs, is always invoked under the QR TCB. It allows you to attempt to resume the transaction at a given address, under a given execution key. If that execution key is key 8, CICS switches to base space before resumption.

If a threadsafe program is in control when the transaction abend occurs, and the exit requests that the task be resumed, the XPCTA exit is driven under the QR TCB, but the task is resumed under the TCB under which the failure occurred.

Changes to task-related user exits

An additional standard parameter is passed to all task-related user exits. The task indicator bits, addressed by UEPTIND, are extended. The UEPTIND parameter now addresses a three byte area instead of one. The first byte contains an additional bit setting, X'20' (equated value UEPTUTCB), to indicate an unexpected TCB. This is set after a failure to switch to the TCB expected by the task-related user exit on a syncpoint or end-of-task call only. In these two cases, the task-related user exit is called on the QR TCB with the UEPTUTCB bit set. For all other calls, CICS abends the transaction without invoking the task-related user exit.

The new second and third bytes addressed by UEPTIND contain a value indicating the TCB mode of the task-related user exit program’s caller. The symbolic values representing the modes are the same as those defined for the UEPGIND parameter (see “Changes to global user exits” on page 127).

Task-related user exit programs can use the new CONCURRENCY option on the PROGRAM resource definition, (see “Changes to resource definition” on page 119). Also, the new OPENAPI option on the EXEC CICS ENABLE command (see “Changes to the system programming interface (SPI)” on page 121) will be enabled in a future deliverable for use by task-related user exit programs.

The following considerations apply to threadsafe and open API task-related user exit programs:

- **CONCURRENCY(QUASIRENT)**

  Task-related user exit programs can remain unchanged, in which case they continue to work as before; that is, they are invoked by CICS under the QR TCB and have to switch to a private TCB if necessary.

- **CONCURRENCY(THREADSAFE)**

  This allows a task-related user exit program to run under the caller's TCB, whether it is QR or open, thus avoiding any TCB switching. This could be useful for a simple task-related user exit which can achieve its function without recourse to a private or open TCB, confining itself to permitted programming interfaces.
- **OPENAPI**

  OPENAPI, currently disabled, will be available for task-related user exit programs.

  An open API task-related user exit will be invoked by CICS under a CICS-managed TCB known as an open TCB. A unique open TCB will be dedicated for the lifetime of the user task that calls the task-related user exit.

  An open API task-related user exit program must be threadsafe.

  An open API task-related user exit program will be able to use the CICS permitted programming interfaces (see "Permitted programming interfaces under the QR TCB" on page 114) and also additional interfaces supported by OTE.

  An open API task-related user exit is generally guaranteed to run under its associated open TCB while it remains in control. If the task-related user exit relinquishes control (by issuing a request for one of permitted programming interfaces), the task may switch to a different TCB for the duration of the request.

  An open API task-related user exit program can be invoked by quasi-reentrant or threadsafe PLT programs executing during the final stage of initialization, and during the first and second stages of quiesce.

General considerations for task-related user exits

There are several factors to consider as a result of the changes that are being made to the task-related user exit interface.

The design of open API task-related user exit programs can be much simpler than that for programs that are always invoked under the QR TCB. For example, they can be designed without the need for privately managed TCBs, and existing task-related user exits can be reworked to remove such TCBs.

Generally, calls to an open API task-related user exit program will be made under an L8 mode open TCB, except for the following calls, which will continue to made under the QR TCB:

- The start of task call
- The FORMAT EDF call
- The CICS termination call

The SPI call will be invoked under the caller's TCB, because the SPI function is simple, and can be performed without the need to switch TCB.

An open API task-related user exit will be able to use non-CICS programming interfaces. If it issues blocking calls, only the issuing task will be held up. As the open TCB is allocated for the lifetime of a user task, it could also be used to run threadsafe programs and other open API task-related user exits in the same task. Therefore, a task-related user exit program must not do anything that could interfere with the other code running under the same open TCB (for example, it must not issue an MVS STIMER macro call). Also, an open API task-related user exit program must leave the TCB in a clean state when returning control to CICS. Unlike a privately-managed TCB, it cannot rely on the state of the TCB being the same on a subsequent invocation.

An open API task-related user exit program will continue to be invoked in key 8. It must be threadsafe, and able to work within the user task's subspace.

Programming considerations

If an open API task-related user exit program uses the CICS programming interfaces, it must ensure that it first restores at least the following aspects of the programming environment before invoking any of permitted programming interfaces (see "Permitted programming interfaces under the QR TCB" on page 114).

- Cross-memory mode (PASN=HASN=SASN)
- ASC mode (Primary)
- Request block (RB) level
- Linkage stack level
- The TCB's dispatching priority
- Any added ESTAEs cancelled.

In addition, an open API task-related user exit must ensure that at CICS task termination, the open TCB is in a suitable state to be reused by another transaction; that is, the TCB must be in a "clean" state). In particular, the task-related user exit must ensure that all non-CICS resources acquired specifically on behalf of the terminating task are freed. Such resources could include:

- Dynamically allocated data sets
- OPEN ACBs or DCBs
- STIMERM requests
- MVS-managed storage
- ENQ requests
- Subtasks (by MVS ATTACH requests)
- Explicitly loaded modules
- Owned data spaces
- Added access list entries
- Name/token pairs
- Fixed pages
- Security environment (that is, TCBSENV must be set to zero)

If an open API task-related user exit abends, and does not recover (that is, it allows the abend to percolate), CICS treats the TCB as unusable, and detaches it when the task completes. Even so, it is the responsibility of the failing task-related user exit to ensure that it does not leave the TCB in a state that precludes the task from invoking all relevant open API task-related user exits for syncpoint and end-of-task.

Note that an open API task-related user exit must be able to share its open TCB with other open API task-related user exits.

Changes to the exit programming interface (XPI)

The XPI is extended with addition of the DFHNQEDX macro function call. This provides a serialisation service using CICS enqueue/dequeue (NQ) domain services. Two request types are supported, ENQUEUE and DEQUEUE. The syntax of the ENQUEUE request is as follows:

ENQUEUE

```
DFHNQEDX [CALL,]
                                [CLEAR,
                                [IN,
                                 FUNCTION(ENQUEUE),
                                 ENQUEUE_NAME1(address,length),
                                 [ENQUEUE_NAME2(address,length),]
                                  MAX_LIFETIME(DISPATCHER_TASK),
                                 [WAIT(YES|NO),]
                                 [PURGEABLE(YES|NO),]
                                  [OUT,
                                  ENQUEUE_TOKEN,
                                  DUPLICATE_REQUEST,
                                  RESPONSE (name1 | *),
                                  REASON(name1 | *)]
```

ENQUEUE_NAME1(address,length)
High-order part of name to be enqueued.

ENQUEUE_NAME2(address,length)
Low-order part (if any) of name to be enqueued.

MAX_LIFETIME(DISPATCHER_TASK)
MAX_LIFETIME(DISPATCHER_TASK) is required and indicates that all XPI enqueues are owned by the requesting dispatcher task.

If you use the enqueue/dequeue XPI to achieve threadsafety in your global user exit progams, you are recommended to free (dequeue) resources during the invocation of the global user exit program in which they were enqueued. However, as no recovery services are provided for abending global user exits, CICS will ensure that any outstanding XPI enqueues are dequeued automatically when the dispatcher task terminates. Note that if the dispatcher task is running a CICS transaction, the dispatcher task terminates when the CICS transaction terminates (whether normally or abnormally).

Today, all enqueues are owned by the requesting transaction. All transactions contain units of work (UOWs), and these are used to anchor the enqueue control blocks. The XPI, however, does not require a transaction environment; indeed, global user exits may be invoked under dispatcher tasks which have no transactions or UOWs.

WAIT(YES|NO)
Indicates whether the dispatcher task is to wait if the resource is currently enqueued to another dispatcher task.

PURGEABLE(YES|NO)
The PURGEABLE option indicates whether a purge (or timeout) request against the task is to be honored if the requesting dispatcher task has to wait for the enqueue.

ENQUEUE_TOKEN
Enables a subsequent DEQUEUE request to identify the resource by a token rather than enqueue name, allowing the NQ domain to locate the enqueue control block directly, and hence more efficiently.

DUPLICATE_REQUEST
Indicates that the requesting dispatcher task already owns the resource being enqueued.

DEQUEUE

```
DFHNQEDX  [CALL,]
          [CLEAR,
          [IN,
          FUNCTION(DEQUEUE),
          {ENQUEUE_TOKEN,|
          ENQUEUE_NAME1(address,length)[ENQUEUE_NAME2(address,length)],}
          [OUT,
          RESPONSE (name1 | *),
          REASON(name1 | *)]
```

The syntax of the DEQUEUE request is as follows:

The DEQUEUE request must be issued under the dispatcher task which issued the ENQUEUE.

The parameters ENQUEUE_TOKEN, ENQUEUE_NAME1, and ENQUEUE_NAME2 are the same as in the ENQUEUE function call.

User-replaceable modules

The CONCURRENCY attribute for URMs can be specified as QUASIRENT or THREADSAFE and is treated in the same way as for normal user application programs.

CICSPlex SM support

CICSPlex SM supports the open transaction environment by extending its inquiry function for the following resource types:

- New fields, Force QR and Max open TCBs, added to the CICSRGN2 view.
- A new field, Concurrency, added to the PROGRAMD view.
- Amendments to the PROGRAM view.
- Amendments to the EXITGLUE and EXITTRUE views.
- Amendments to the TASK and TASKD views.

Chapter 10. Long temporary storage queue names

This chapter describes enhancements to the CICS temporary storage facility that allow you to use up to 16 characters for TS queue names. It covers the following topics:

Overview

In earlier releases of CICS it is sometimes difficult to generate unique names using the conventional 8-character TS queue names. Typically, the generated queue name would be of the form *xxxxtttt*, where *xxxx* is the transaction ID and *tttt* is the terminal ID (termid). This convention succeeds only if the termid is unique and one queue only is required by the transaction.

The CICS temporary storage facility is enhanced to allow TS queue names to be 16 characters long, providing greater flexibility for generating TS queue names that are unique across a CICSplex.

A longer TS queue name removes existing restrictions, and provides new design options, such as TS queues based on the user ID. In this way, scratchpad data can be kept for longer periods.

16-character queue names provide much greater flexibility in user application programs because they can be generated of the form *ttttSSSSuuuuuuuu*, where *tttt* is the transaction identifier, *SSSS* is a suffix allowing the use of more than one queue per transaction, and *uuuuuuuu* represents the user ID. The TSMODEL prefix could then be *tttt* or *ttttSSSS*. Alternatively, with the increased length of queue names to 16 characters, you could incorporate the 4-character termid or the 8-character network name instead of the user ID.

Note: Increasing TS queue names to only 16 characters is governed by the use of the coupling facility for TS data sharing, which imposes a maximum key length of 16 characters.

Effect on application programs

There are some changes to the application programming interface to accommodate 16-character TS queue names (see "Changes to resource definition" on page 135 for details). However, there are some API commands that use temporary storage control that are unaffected by the change.

EIB resource name field EIBRSRCE unchanged

There are no changes to the EXEC interface block for long TS queue names. When a temporary storage control API command is issued, EIBRSRCE, the EXEC

interface block that contains the symbolic identifier of the temporary storage queue name being accessed, continues to contain the first 8 characters of the queue name.

REQID on BMS and interval control requests unchanged

The REQID provided on the START command for BMS and interval control requests remains unchanged at 8 characters.

QUEUE option on START requests unchanged

The QUEUE option on START requests remains unchanged at 8 characters.

Shared TS unchanged

There are no changes for shared temporary storage because 16-character queue names are already supported by this function.

Benefits of long temporary storage queue names

Long temporary storage queue names make it easier to generate unique queue names to provide the required flexibility in user application programs.

Requirements for long temporary storage queue names

The hardware requirements for this function are the same as for CICS TS generally.

If you use RACF as your external security manager, and your RACF profiles for TS queues are defined with the CICS region userid as a prefix, you need RACF OS/390 Release 6, plus PTF UW90545 (for APAR OW35612). This PTF enables you to define temporary storage resource class profiles with names up to a maximum length of 25 characters. See “Changes to security” on page 138 for more information.

If you do not use RACF security for temporary storage queues, or if you use RACF security without prefixing, you need OS/390 Release 5, the same as for CICS TS generally.

Changes to CICS externals

This section provides an overview of the changes to CICS externals for temporary storage queues to be given names up to 16 characters long:

- “Changes to problem determination” on page 138
- “Changes to security” on page 138.

Changes to resource definition

Extending temporary storage queue names to 16 characters is implemented in the new TSMODEL resource definition panel (see “Chapter 5. Resource definition online for CICS temporary storage” on page 59 for details). This resource definition provides the only method for defining recoverable, secure, or remote TS queues with up to 16-character queue names, using the PREFIX attribute.

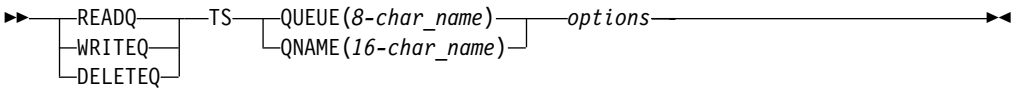
The DATAID equivalent of the DFHTST macro remains an 8-character field.

Changes to the application programming interface (API)

A new parameter on the temporary storage API makes it easier for CICS applications to generate TS queue names that are unique across a CICSplex.

Commands READQ, WRITEQ, and DELETEQ TS

The QNAME option, for specifying a 16-character queue name, is added to the following commands as an alternative to the QUEUE option:
- WRITEQ TS
- READQ TS
- DELETEQ TS



To exploit 16-character queue names, change your applications to use the QNAME option, otherwise you can continue to use your existing applications unchanged using the 8-character QUEUE option.

Function shipping

Function shipping enables application programs to send data to, or retrieve data from, temporary storage queues located on remote systems.

If an application program issues a request to access a remote queue over an MRO link using a 16-character queue name, the 8-character queue name is set to binary zeros and the long queue name is shipped in a separate 16-character field. CICS returns an INVREQ if the remote region is an earlier release of CICS that cannot handle the 16-character name.

For non-MRO remote requests, CICS ships the request with a 16-character name. If the remote region cannot handle the full 16 characters, the first 8 characters only are used.

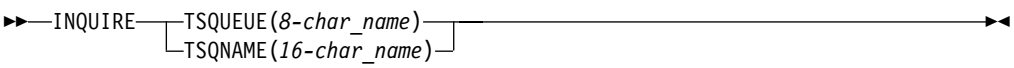
Changes to the system programming interface (SPI)

The SPI commands EXEC CICS INQUIRE TSQUEUE and EXEC CICS INQUIRE TASK are enhanced to browse the temporary storage table using a long queue name, and a new command EXEC CICS SET TSQUEUE is added.

INQUIRE TSQUEUE

On the INQUIRE TSQUEUE command, which retrieves information about a temporary storage queue, the option TSQNAME has been added as an alternative to TSQUEUE.

This change is shown in the following fragment of the syntax:



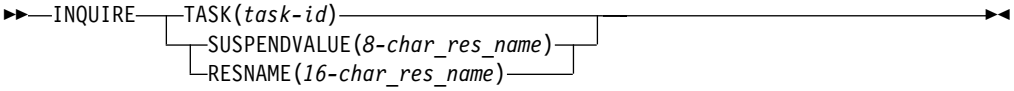
Browsing TS queues: You can use the INQUIRE TSQUEUE command with START AT(*data-value*) to browse TS queues. The *data-value* operand can specify queue names greater than 16 characters. Use the TSQNAME option with the NEXT option on the INQUIRE command if you have application programs using TS queues with long names.

If you use the existing TSQUEUE option on the INQUIRE NEXT command to browse a queue, and the next queue name being returned exceeds 8 significant characters, CICS truncates the queue name to 8 characters, and returns an INVREQ condition with a RESP2 value of 2.

INQUIRE TASK

A new option, RESNAME, is added to the INQUIRE TASK command as an alternative to SUSPENVALUE. RESNAME is added to handle 16-character TS queue names.

If you specify SUSPENDVALUE, and the task you are inquiring about is waiting on a TSQUEUE with a long queue name, CICS truncates the name to 8 characters and returns an INVREQ condition with a RESP2 value of 1.



Changes to global user exits

The introduction of 16-character TS queue names affects the following temporary storage global user exits:

- XTSEREQ and XTSEREQC
- XTSQRIN and XTSQROUT
- XTSPTIN and XTSPTOUT

TS EXEC interface program exits, XTSEREQ and XTSEREQC

The information passed in the command-level parameter structure is modified to indicate whether the application program has specified QUEUE or QNAME on the TS request. If QNAME is used, TS_ADDR1 is the address of a 16-byte area containing the name from QNAME. This is indicated by the TS_EIDOPT5 field in the EID, which is set to X'80'.

TS storage domain exits XTSQRIN and XTSQROUT

The parameter list for the temporary storage domain exits, XTSQRIN and XTSQROUT, contains the address of the queue name, which now refers a 16-character field.

TS storage domain exits XTSPTIN and XTSPTOUT

TS storage domain exits XTSPTIN and XTSPTOUT remain unchanged because the TSPT interface does not support long queue names.

Changes to CICS-supplied transactions

There are changes to CEMT and CEBR in support of long TS queue names.

Changes to CEMT

TSQNAME is added as an alias for TSQUEUE on the CEMT INQUIRE TSQUEUE command. CEMT INQUIRE TSQUEUE|TSQNAME retrieves information about temporary storage queues, the names of which can be up to 16 characters. Unlike the TSQUEUE option, TSQNAME cannot be abbreviated to the minimum number of unique characters. For example, TSQN returns an error as command not valid.

The HVALUE parameter of the CEMT INQUIRE TASK command is modified to allow for 16-character resource names for tasks suspended for any reason on TS queues. 8-character names are padded with blanks up to 16 characters.

Changes to CEBR

The CICS-supplied transaction CEBR, which allows you to browse temporary storage queues and to delete them, is enhanced to support long queue names.

Changes to monitoring

The monitoring interfaces are updated to allow for 16-character resource ID fields and the monitoring DSECTS are changed.

The sample monitoring utility program DFHœMOLS is changed to allow for 16-character names appearing in exception records.

See “Chapter 6. Monitoring, statistics, and enterprise management changes” on page 65 for details of all these monitoring changes for long TS queue names.

Changes to samples

The sample program DFH0STAT has been changed to include INQUIRE TSQNAME statistics. The page layout is altered to accommodate 16-character queue names.

Changes to utilities

The CICS Affinities Utility is updated to allow for 16-character queue names.

Changes to problem determination

There are changes to CICS messages.

Those messages that have TS queue names appearing as inserts are changed, where appropriate, to include the 16-character names.

Changes to security

With releases of RACF earlier than RACF OS/390 Release 5, the maximum length of profile names in the CICS temporary storage resource class, SCICSTST, is 17 characters. This allows for an 8-byte TS queue name with, optionally, an 8-byte prefix followed by a period. To enable you to define longer profile names (up to 25 bytes to allow for 16-character TS queue names), you need to apply the required RACF service PTF to OS/390 Release 6.

Alternatively, you can create your own resource classes by adding new class descriptors to the installation-defined part (module ICHRRCDE) of the RACF class descriptor table (CDT), and specify the length on the MAXLNTH parameter of the ICHERCDE macro.

CICSPlex SM support

CICSPlex SM support for 16-character, non-shared temporary storage queue names is provided by the new operations views:

- TSQNAME, which shows general information about all 16-character, non-shared temporary storage queues.
- TSQNAMED, which shows detailed information about a 16-character, non-shared temporary storage queue.
- TSQNAMES, which shows summary information about 16-character, non-shared temporary storage queues.

The existing TSQ view remains unchanged and continues to support 8-character names only.

A new resource table, TSQNAME, is introduced.

Chapter 11. EXCI enhancement for resource recovery

This chapter describes enhancements to the external CICS interface (EXCI) to support OS/390 resource recovery services. It covers the following topics:

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 region and to pass and receive data by means of a communications area. The CICS application program is invoked as if linked-to by another CICS application program.

In earlier releases of CICS, the external CICS interface forces the SYNCONRETURN option on the DPL_Request call, which means that the server region always takes a syncpoint on successful completion of the server program. Issuing a DPL_request call with SYNCONRETURN means there is no coordination of syncpoints between the EXCI client program and the server program. Also, with SYNCONRETURN, the server application program is permitted to take explicit syncpoints during its execution.

Adding support for OS/390 recoverable resource management services (RRMS) changes the EXCI recovery environment. It introduces the concept of a unit of recovery (UR). A UR represents an application program's changes to resources since the last commit or backout, or, for the first UR, since the beginning of the application. Support for RRMS enables CICS server regions to register with resource recovery services (RRS), the MVS syncpoint manager. This support then enables EXCI client programs to coordinate changes to protected resources within a unit of recovery, thus ensuring data integrity. See Figure 22 on page 140 for an illustration of EXCI client and CICS server regions with RRMS.

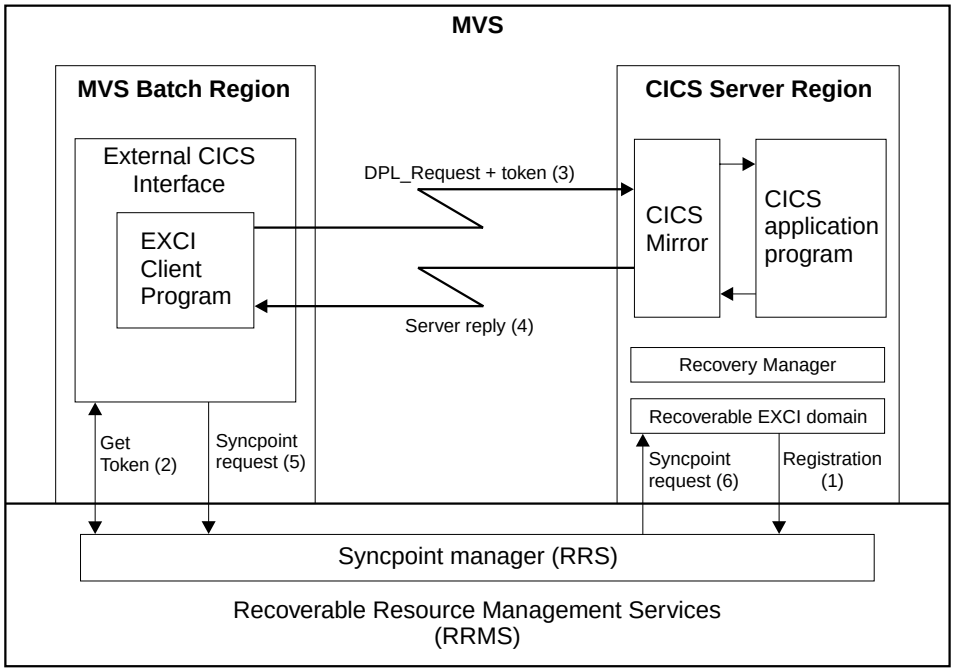


Figure 22. Conceptual view of EXCI client and CICS server region using RRMS. The main (numbered) steps within a unit of recovery are described in the list following the diagram

1. If the CICS system initialization parameter RRMS=YES is specified, CICS registers with RRMS as a resource manager. This registration occurs during CICS initialization.
2. When the EXCI client program issues a DPL_Request call in 2-phase commit mode (a call that omits the SYNCONRETURN option), it receives from RRMS:
   - A unit of recovery identifier (URID)
   - A context token
   - A pass token
3. The URID and the tokens obtained by EXCI on behalf of the client progam are included on the DPL request passed to the CICS server region. If the DPL request is the first one within the URID, CICS calls RRS to express an interest in the UR, attaches a new mirror transaction, and validates the tokens. If the request is valid, the mirror program links to the specified server application program. The server program completes its work, which is all performed within the unit of recovery. This work can include updating recoverable resources in the local server region, or daisy-chaining to other CICS regions.
4. When the server program completes, it returns the communications area (COMMAREA) and return codes to the client program.

   Note: Steps 3 and 4 can repeated many times for the same UR.
5. When the EXCI client program is ready to commit or back out its changes, the program invokes RRS to begin the 2-phase commit protocol.
6. RRS acts as coordinator and either:
   - Asks the resource managers to prepare to commit all updates within the UR. (Note that resource managers other than the CICS server region may also have expressed an interest in the UR.) If all vote yes, RRS tells them to go ahead and commit the changes. If any vote no, all resource managers are told to perform backout.

- Tells all the resource managers that expressed an interest in the UR to perform backout of all the changes made with the UR.

The UR is now complete and CICS detaches the mirror task. If the EXCI client sends any new DPL requests after this point, EXCI starts a new unit of recovery and CICS attaches a new mirror transaction.

Resource recovery consists of the protocols and program interfaces that allow an application program to make consistent changes to multiple protected resources. The RRMS syncpoint manager , when requested, can coordinate changes to one or more protected resources, which can be accessed through different resource managers. RRMS, in conjunction with CICS recovery manager, ensures that all changes are made or no changes are made. Resources that OS/390 RRMS can protect include:

- A hierarchical database
- A relational database
- A product-specific resource (such as a CICS file or CICS temporary storage queue).

Note: RRMS consists of three separate MVS components. These are registration services, context services, and resource recovery services (RRS), that collectively are referred to as recoverable resource management services (RRMS). For information about all these services, see the*OS/390 MVS Programming: Resource Recovery* manual.

With RRMS support, EXCI clients can issue DPL_Request calls without the SYNCONRETURN option, allowing the EXCI client program to control when updates to recoverable resources are committed. This enhancement to EXCI for resource recovery means that there are now two modes of operation:

- Without RRMS support, as in earlier releases (the sync-on-return mode). On completion of each DPL request, updates made by the server program for each DPL request are committed independently of the client program or any other server.
- With RRMS transactional support (the 2-phase commit mode). The client program can issue successive DPL requests, to more than one server program, in more than one CICS region. All updates initiated by these DPL requests are part of the same UR. When the client program requests it, the commitment of updates within the unit of recovery is coordinated by RRMS.

  Thus, a client application program can issue EXCI API calls in 2-phase commit mode, which ensures that all changes to recoverable resources are treated as part of the same MVS unit of recovery and the same CICS distributed UOW (identified by its network UOW ID).

For more information about how CICS uses RRMS and about the UR as the scope of recoverable work, see the *CICS Recovery and Restart Guide*.

RRMS provides some ISPF panels that allow you view RRS information, an example of which is shown in Figure 23 on page 142.

```
                                    RRS
Option ===>

Select an option and press ENTER:

1 Browse an RRS log stream
2 Display/Update RRS related Resource Manager information
3 Display/Update RRS Unit of Recovery information
4 Exit
```

Figure 23. The main RRS ISPF panel

Adding RRMS support to CICS regions

To enable a CICS server region to support DPL_Request calls from an EXCI client in 2-phase commit mode, the CICS region must register with RRS during CICS initialization. This is controlled by a system initialization parameter, RRMS. See “Changes to system definition” on page 143 for information about this parameter.

When the first EXCI call in a unit of recovery is received by CICS, there is additional processing involved notifying RRS of the CICS region’s interest in the unit of recovery.

Locks acquired by a CICS server region on behalf of a server application program unit of work are not released when the server program ends. They are retained until the MVS client application commits the unit of recovery.

RRS

An EXCI client program can issue DPL_Request calls that specify SYNCONRETURN when the target CICS server resides in another MVS image within the sysplex. This is not the case when the EXCI client is using 2-phase commit mode. In this mode, the EXCI client program and its initial CICS server region *must* run in the same MVS image as RRS, the syncpoint manager component of OS/390 RRMS. However, the CICS server application program can issue other DPL requests to remote CICS regions that reside in other MVS images within the sysplex.

RRS provides an application programming interface (API) that an EXCI client program can use. This supports two callable services:

- Application_Commit_UR
- Application_Backout_UR

These callable service and the API are described in the *OS/390 MVS Programming: Resource Recovery* manual.

Benefits

All the previous benefits of the EXCI are preserved and, in addition, programs executing in an MVS environment are able to:

- Coordinate and use CICS with other RRS-enabled resource managers
- Provide a single unit of work capability to clients over multiple calls to CICS programs and over multiple CICS regions.

Requirements

The hardware requirements for this function are the same as for CICS TS generally.

The software requirement is that you need either OS/390 Version 2 Release 6, or OS/390 Version 2 Release 5 with a service PTF for APAR UW46914.

Changes to CICS externals

There are changes to a number of CICS externals in support of the RRMS enhancements to the external CICS interface. The externals affected are:

- "Changes to system definition"
- "Changes to the system programming interface (SPI)"
- "Changes to the external CICS interface" on page 144
- "Changes to CICS-supplied transactions" on page 145
- "Changes to problem determination" on page 145

Changes to system definition

A new system initialization parameter is added to suppport the EXCI enhancements for resource recovery:

RRMS={NO|YES}
Specifies whether you want CICS to register with MVS resource recovery services.

Changes to the system programming interface (SPI)

The following SPI commands are new for use with MVS RRS:

EXEC CICS INQUIRE EXCI
Provides an INQUIRE EXCI function equivalent to the existing CEMT INQUIRE EXCI with the addition of the following option:

URID(data value)
This field contains, when EXCI is using RRMS to coordinate updates, a 32–character string containing the hexadecimal representation of the RRMS unit of recovery identifier. Note that the absence of a URID (that is, blanks are returned) indicates that the EXCI DPL_Request call specified SYNCONRETURN.

EXEC CICS INQUIRE RRMS
Retrieves information about the state of a CICS region's registration with MVS RRMS, and the status of RRS in the MVS image. The OPENSTATUS option returns one of the following CVDA values:

OPEN CICS is registered with RRMS and RRS is active. CICS is able to accept inbound transactional EXCI work.

CLOSED
CICS is registered with RRMS, but RRS is not active. CICS cannot accept inbound transactional EXCI work.

NOTAPPLIC
The CICS region does not require RRMS (RRMS=NO is specified as a system initialization parameter).

There are also changes to some existing SPI commands for use with MVS RRS, as follows:

- A new CVDA value is added for the WAITCAUSE option of the EXEC CICS INQUIRE UOW command. RRMS indicates that the UOW is waiting, or has been shunted because communication has been lost with RRMS.
- A new CVDA value is added for the PROTOCOL option of the EXEC CICS INQUIRE UOWLINK command. RRMS indicates that the UOW is coordinated by RRMS. Note that when the PROTOCOL option returns RRMS, the SYSID option returns blanks.
- The function of EXEC CICS SET UOWLINK DELETE command is extended so that UOWLINKs associated with RRMS can be deleted when RRS has been cold started. The syntax is unchanged.

Changes to the external CICS interface

There are changes to the EXCI CALL interface and to the EXEC CICS interface.

The EXCI CALL interface changes

There are changes affecting a number of parameters of the DPL_Request call of the CICS external interface. These are:

DPL_opts
This parameter is extended to permit DPL_Request calls without the SYNCONRETURN option. The one-byte input area of the DPL_opts parameter can have one of the following values:

- X'00'to indicate that the CICS server region must not perform a syncpoint when the server program returns control to CICS. Furthermore, the CICS server application program must not take any explicit syncpoints, otherwise it is abended by CICS.
- X'80' to indicate that the CICS server region is to perform a syncpoint when the server program returns control to CICS (enforced SYNCONRETURN). In addition to the implicit syncpoint taken by CICS, the server application program can take explicit syncpoints.

transid
There are new rules about the use of the *transid* parameter when *DPL_opts* specifies X'00'.

The *transid* specification must be the same for all EXCI calls in the same MVS unit of recovery. If you specify *transid* on the first call, you must specify the same value for *transid* on all calls. If you omit *transid* from the first call, you must omit it from all calls.

uowid
This parameter must not be specified when *DPL_opts* specifies X'00'. This is because EXCI generates one for you.

When *DPL_opts* specifies X'80' (SYNCONRETURN), the *uowid* is the APPC unit-of-work identifier used by the CICS server region and any subordinate CICS regions.

userid

There are new rules about the use of the *userid* parameter when *DPL_opts* specifies X'00'.

The *userid* specification must be the same for all EXCI calls in the same MVS unit of recovery. If you specify *userid* on the first call, you must specify the same value for *userid* on all calls. If you omit *userid* from the first call, you must omit it from all calls.

There are new reason codes for DPL_Request calls that specify X'00' (no SYNCONRETURN) on the *DPL_opts* parameter. These are described in the *CICS External Interfaces Guide*.

There are also some new abend codes associated with transactional EXCI calls.

The EXEC CICS interface changes

The SYNCONRETURN option of the EXEC CICS LINK PROGRAM(*name*) command is now optional. It is a required option in earlier releases. Omitting the SYNCONRETURN option allows the client program to control syncpoints taken by the CICS server region on behalf of the server application program.

Also:

- The INVREQ response is no longer returned on an EXEC CICS LINK issued by an EXCI client program.
- There are some new RESP2 values for EXEC CICS LINK commands associated with commands that omit SYNCONRETURN.
- There are new abend codes associated with commands that omit SYNCONRETURN.

The new codes are described in the *CICS External Interfaces Guide*.

Changes to CICS-supplied transactions

A new CEMT INQUIRE RRMS command is provided to enable you to obtain information about the state of a CICS region's registration with the MVS syncpooint manager, RRS. This is the CEMT equivalent of the EXEC CICS INQUIRE RRMS command.

The following CEMT commands are enhanced to provide the same function as their EXEC CICS equivalent commands:

- CEMT INQUIRE EXCI is enhanced with the addition of the URID option.
- CEMT INQUIRE UOW is enhanced with the addition of a new wait cause, RRMS.
- INQ UOWLINK is enhanced with the addition of a new protocol option, RRMS

The function of CEMT SET UOWLINK DELETE is extended so that UOWLINKs associated with RRS can be deleted if required. The syntax is unchanged.

Changes to problem determination

To aid problem determination, there are some new messages and abend codes. These are described in the *CICS Messages and Codes*, and further information is provided in the *CICS Problem Determination Guide*.

CICSPlex SM support

CICSPlex SM support for RRS is provided by:

- A new value of RRMS for the Wait Cause field of the UOWORKD view.
- A new field, RRMS Status, on the CICSRGND view. The RRMS Status may have the values OPEN, CLOSED, and N/A.
- A new field Protocol on the UOWLINKD view. The Protocol field can either have the value RRMS or be blank. If the Protocol field has the value RRMS, the Linked Sysid field is blank.

Chapter 12. Object-oriented interface to CICS services for C++

This chapter describes the CICS object-oriented programming interface for the C++ programming language. It covers the following topics:

Overview

The CICS family of products provides robust transaction processing capabilities across the major hardware platforms that IBM offers, and also across key non-IBM platforms. It offers a wide range of features for supporting client/server applications, and allows the use of modern graphical interfaces for presenting information to the end-user. CICS now supports object-oriented programming and offers CICS users a way of capitalizing on many of the benefits of object technology while making use of their investment in CICS skills, data, and applications.

The CICS C++ foundation classes allow an application programmer to access many of the CICS services that are available through the EXEC CICS procedural application programming interface (API).

The CICS foundation classes package is supplied in a number of data sets that contain:

- The header files
- The dynamic link library
- The sample programs and some JCL procedures
- The CICS-supplied side-deck.

The header files

The header files are the C++ class definitions you need to compile CICS C++ foundation class programs. A single header file, **icceh**, that includes all the other header files, is supplied in the CICSTS13.CICS.SDFHC370 library.

The file **iccmain** is also supplied with the C++ header files. This contains the main function stub that you use when building a foundation class program, and is also supplied in the CICSTS13.CICS.SDFHC370 library.

The dynamic link library

The dynamic link library (DLL) provides the runtime support that you need to run a CICS C++ foundation class program. The DLL is supplied as ICCFCDLL in the CICSTS13.CICS.SDFHLOAD library.

The sample programs

A number of sample programs are provided to help you understand how to use the C++ foundation classes to create object-oriented application programs. These are provided in both source and executable form:

- The source program are supplied as members of the CICSTS13.CICS.SDHSAMP library.

- The executable load modules are supplied as members of the CICSTS13.CICS.SDHLOAD library.

CICS also provides three JCL procedures that enable you to compile, prelink, and link a C++ OO program.

The CICS-supplied side-deck

A new CICS library, CICSTS13.CICS.SDFHSDCK, is created when you install CICS TS. This PDS holds the definition side-deck generated in SYSDEFSD by the prelinker process run by CICS, and supplied as member, ICCFCIMP. This member contains the import control statements required by the linkage-editor step when linking your C++ OO programs.

For more informaiton about the definition side-decks required by your C++ programs, see the *OS/390 C/C++ User's Guide*.

Benefits

Object oriented programming allows more realistic models to be built in flexible programming languages that allow you to define new types or classes of objects, as well as employing a variety of structures to represent these objects.

Object oriented programming also allows you to create methods (member functions) that define the behavior associated with objects of a certain type, capturing more of the meaning of the underlying data.

Chapter 13. JCICS interface to CICS services for Java

This chapter describes the CICS object oriented interface that provides access to CICS services from Java application programs. It covers the following topics:

Overview

CICS provides a new programming interface for use in CICS application programs written in Java. CICS Java classes, known as JCICS, are provided to give Java access to CICS services traditionally available through the CICS command-level procedural API.

See the *CICS Application Programming Guide* for a full description of the Java language support.

You can develop CICS Java programs on a workstation, or in the OS/390 UNIX System Services shell, using an editor of your choice, or in a visual composition environment such as VisualAge.

The Java language is designed to be portable and architecture-neutral, and the bytecode generated by compilation requires a machine-specific interpreter for execution. CICS provides this execution environment in two different ways:

1. Using VisualAge for Java, Enterprise ToolKit for OS/390 to bind the Java bytecode into OS/390 executable files that are stored in MVS PDSE libraries and executed by CICS in a Language Environment (LE) run-unit, similarly to C++.
2. Using an MVS Java Virtual Machine that is executing under CICS control.

There is no CICS translator for CICS Java programs.

JavaBeans

Some of the classes in JCICS may be used as JavaBeans, which means that they can be customized in an application development tool such as VisualAge for Java, serialized, and manipulated using the JavaBeans API. The beans in the CICS Java API are currently:

- Program
- ESDS
- KSDS
- RRDS
- TDQ
- TSQ
- AttachInitiator
- EnterRequest

These beans do not define any events; they consist of properties and methods. They can be instantiated at run-time in one of three ways:

1. By calling `new()` for the class itself. (This is the recommended way)
2. By calling `Beans.instantiate()` for the name of the class, with property values set manually
3. By calling `Beans.instantiate()` of a `.ser` file, with property values set at design time

If either of the first two options are chosen, then the property values, including the name of the CICS resource, must be set by invoking the appropriate setter methods at run-time.

Supplied components

The following jar files are stored in the OS/390 UNIX System Services HFS in the directory **$CICS_HOME/classes**:

dfjcics.jar
The JCICS API classes, required for compilation of a Java application program that uses JCICS to access CICS services.

Note: The JCICS classes can also be used at execution time by a JVM program to access CICS services.

dfjwrap.jar
Used internally by CICS to support the JCICS interface in a JVM environment.

Sample programs

Sample programs to demonstrate the use of the JCICS classes are stored in the HFS in the $CICS_HOME/samples directory and in SDFHSAMP.

JCICS reference documentation

The JCICS classes are documented in JAVADOC HTML. This is stored in the OS/390 UNIX System Services HFS in the directory **$CICS_HOME/docs**. You can read this file using a web browser. The following file is supplied:

dfjcics_docs.zip

Benefits of Java language support

The Java programming language is rapidly growing in popularity. Support for CICS Java applications allows CICS users to remain at the leading edge of technology, and to exploit future business opportunities using the Internet, where Java is heavily used.

Chapter 14. VisualAge for Java, Enterprise Edition for OS/390

This chapter describes CICS support for Java application programs. It covers the following topics:

Overview

This support uses the **VisualAge for Java, Enterprise Toolkit for OS/390 (ET/390)** to enable Java application programs to run under CICS control.

The Java language support is similar to CICS language support for COBOL or C++. The normal CICS program execution model is used, rather than a Java Virtual Machine (JVM).

The application program is developed and compiled using a Java compiler (such as VisualAge for Java or javac) on a workstation or in the OS/390 UNIX System Services environment. The .class files produced are then processed by the **binder** component of ET/390, executing in the OS/390 UNIX System Services environment, to produce **Java program objects** that can be loaded by CICS and executed in a Language Environment (LE) run-unit, similar to C++.

CICS also introduces a new programming interface for use in CICS application programs written in Java. Classes (known as JCICS) are provided to give Java access to CICS services traditionally available through the CICS command-level procedural API.

See the *CICS Application Programming Guide* for a full description of the Java language support.

In addition, VisualAge for Java, Enterprise Edition for OS/390 enables you to develop CICS applications using a workstation Integrated Development Environment (IDE); export them to the host CICS environment via OS/390 UNIX System Services, and debug them interactively from the workstation.

Benefits of Java language support

The Java programming language is rapidly growing in popularity. Support for CICS Java applications allows CICS users to remain at the leading edge of technology, and to exploit future business opportunities using the Internet, where Java is heavily used.

Java language support in CICS provided by VisualAge for Java, Enterprise Edition for OS/390, has the following advantages:

- Object Oriented programming with ease of use

- Application development using interfaces of sufficient standardization and simplicity to allow the use of visual AD technology, such as Java Beans
- Access to existing CICS applications and data from Java programs
- A state-of-the-art development and debug environment for CICS/TS applications

Requirements

The hardware and software requirements for this function are the same as for CICS TS generally. In addition:

- VisualAge for Java, Enterprise ToolKit for OS/390 is required to bind and run CICS Java programs.
- OS/390 UNIX System Services is required to run the VisualAge for Java, Enterprise ToolKit for OS/390 byte-code binder.

Service PTFs for the following APARs are required before using Java language support:

- OW31036 (Bind with long object names)
- OW31718 (DFSMS™ 1.4 Invalid loader storage check)
- OW31924 (IEW2333E Invalid syntax in IMPORT control statement)
- OW32111, OW32261, and OW32334 (IEW2900T E913 Binder abnormal termination)
- OW33782 (DFSMS 1.4 DESERV to set output buffer length for PDSE access)
- OW34052 (Load optimization for C_WSA for DLLs in dynamic LPA)
- PQ08747 (For LE to support double precision floating-point in single thread)
- PQ17512 (0C4 when signal occurs in stack extension boundary)
- PQ19340 (CICS ABEND failure caused by LE condition handler)

Check the *CICS Transaction Server for OS/390 Program Directory* for later information about other required service. See also Info APAR II11025 on RETAIN (which lists the CICS TS and OS/390 Release 5 LE compatibility fixes) and Info APAR II11597 on RETAIN (for other product required PTFs).

Changes to CICS externals

This section describes the following changes to CICS externals to provide Java language support using VisualAge for Java, Enterprise Edition for OS/390:

- "Changes to installation"
- "Changes to the application programming interface (API)" on page 153
- "Changes to samples" on page 153
- "Changes to problem determination" on page 153

Changes to installation

- **Full function** OS/390 UNIX System Services must be active during installation so that supporting libraries and samples can be stored in the HFS.
- MVS PDSE libraries must be created to hold CICS and user Java program objects. These libraries must be defined in the CICS DFHRPL concatenation in the CICS start-up JCL.

Changes to the application programming interface (API)

There are no changes to the EXEC CICS API. A class library, similar to the CICS C++ foundation class library is provided to access existing CICS services. See the *CICS Application Programming Guide* for more information about using JCICS.

This class library, known as JCICS, is documented in HTML format only and distributed in **dfjcics_docs.zip**, which is stored in the HFS during installation.

Changes to samples

Sample programs are provided. See the *CICS Application Programming Guide* for more information about running the sample Java programs.

Changes to problem determination

Changes are made to CICS problem determination services to support CICS Java applications.

Messages

New CICS messages in the DFHCZxxxx range are generated by the Java language support.

Trace

New CICS trace entries are generated by the Java language support.

Abend codes

New CICS abend codes, AJ01-09 and AJ99 are generated by the Java language support.

Changes to user tasks

Application development

To build a CICS Java program, you need to:

- Prepare the prerequisite environment. This includes installing the VisualAge for Java, Enterprise ToolKit for OS/390 and configuring the OS/390 UNIX System Services shell on your OS/390 system. Note that OS/390 UNIX System Services must be configured with **full function**.
- Compile your program. You can develop and compile your Java program remotely on a workstation or in the OS/390 UNIX System Services shell on your OS/390 system. The CICS Java API .jar files must be in the CLASSPATH for your compiler.
- Transfer the compiler output (Java byte-code) to OS/390 UNIX System Servicesl, if you compiled on a workstation. If you use VisualAge for Java to develop your application on a workstation, you can use the supplied export function to transfer your Java byte-code to the OS/390 UNIX System Services HFS. If you use another compiler, you must transfer your files using other means, such as **ftp** or **nfs**. The files must be transferred in binary mode.
- Bind the Java bytecode using the VisualAge for Java, Enterprise ToolKit for OS/390 bytecode binder to produce Java program objects that can be loaded into CICS and executed as CICS programs. These program objects are

stored in MVS PDSE libraries that must be concatenated to the CICS DFHRPL at run-time. The CICS Java API .jar files must be in the CLASSPATH for this step.

You can use any suitable compiler, such as javac or VisualAge for Java. The VisualAge for Java product contains support for the CICS environment, providing access to the CICS Java classes and commands to export and run the ET/390 binder.

An integrated development environment (IDE) such as VisualAge for Java allows control of the process from the workstation.

The *CICS Application Programming Guide* gives further details.

Chapter 15. Support for the Java Virtual Machine

This chapter describes CICS support for the Java Virtual Machine. It covers the following topics:

Overview

CICS provides the support you need to run a Java transaction in a CICS region under the control of an OS/390 Java Virtual Machine (JVM). The CICS JVM support is complementary to the support for VisualAge for Java, Enterprise ToolKit for OS/390. However, whereas the support for VisualAge for Java, Enterprise ToolKit for OS/390 restricts application programs to a subset of the core Java classes and the CICS API for Java (JCICS), the CICS JVM provides full Java support without any restrictions. To distinguish between the two types of Java program supported by CICS, they are referred to in this chapter in the following terms:

JVM program
: This is a Java program compiled to bytecode by a standard Java compiler, and stored in an HFS data set, from which it can be executed by a Java virtual machine. For use by CICS, the HFS data set must be in a directory identified by a CLASSPATH statement in the DFHJVM member of the SDFHJENV environment variables data set.

Java program object
: This is a Java program compiled to bytecode by a standard Java compiler, bound into an executable program object using the VisualAge for Java, Enterprise ToolKit for OS/390 (ET/390), and stored in a partitioned data set extended (PDSE) library. For use by CICS, the PDSE must be included in the DFHRPL library concatenation, the same as CICS application programs written in the other supported languages.

The JVM support is provided for CICS application programs written in Java and compiled to bytecode by any standard Java compiler.

In addition to the full set of Java classes, JVM programs executing in the CICS address space can also use the following Java packages (set of classes):

- java_io, which includes access to HFS files
- java_net, which includes access to OE sockets
- java_rmi, which includes bytecode interpretation
- java_lang, which includes application level threading
- java_awt, which includes windowing support.

With CICS JVM support a JVM program can use the JCICS API with some restrictions. For example, the JVM program can issue JCICS API calls under the initial process thread (IPT), but not under any pthreads it has created.

The OS/390 JVM supported by CICS uses services provided by the CICS open transaction environment (OTE), which is described in “Chapter 9. Open transaction environment” on page 111. The multiple open TCB support provided by CICS OTE introduces the J8 mode TCB, which is reserved for use by JVMs. Each CICS transaction invoking a JVM program runs under its own J8 TCB, with its own JVM. A new JVM is created for each JVM program invoked by CICS, and when execution of the user class ends, the JVM is destroyed. Each J8 TCB is set up to use MVS LE services rather than CICS LE services. Hence programs executing under a J8 mode TCB, including the JVM, have access to standard OS/390 UNIX system services and MVS LE services.

MVS JVM integration with CICS

CICS program manager treats JVM programs much the same as any other user application program that is compiled by a language compiler. Program manager looks after installed program resource definitions, and the program attributes that can be both set and inquired upon. It is also responsible for program control API commands, such as LINK, LOAD, XCTL and so on.

JVM programs are loaded from HFS by the JVM itself, and are not loaded directly from the RPL. The JVM loads the program from HFS using a qualified class name which is greater than the 8 character program name by which it is referenced by CICS. Program manager provides the mapping from 8-character name to fully qualified classname, enabling this association to be made as part of the process of invoking the program through the JVM.

A JVM program can be used as a CICS application program in the following situations:

- As the initial program in a transaction, defined in the transaction’s resource definition
- As a program linked by an EXEC CICS LINK command
- As a program named in an initialization and shutdown program list table (PLT)
- As a program given control by an EXEC CICS XCTL command.

A JVM program could also be used as a CICS program in the following situations, although these are much less practical than the uses listed above, and are not recommended:

- As a program named in an EXEC CICS HANDLE ABEND command
- As a user replaceable module (URM).

A JVM program cannot be specified on EXEC CICS LOAD and EXEC CICS RELEASE commands, because there is no program for CICS to load (from the DFHRPL library) or to release. These requests fail with PGMIDERR with a new RESP2 value.

A JVM program cannot be invoked by a COBOL dynamic call, and cannot be linked-edited with a program compiled in another language.

There are new attributes on the CICS program resource definition for a JVM program; see “Changes to CICS externals” on page 158 for details.

Note: A Java program object compiled and link-edited into a DFHRPL library is not defined with the new JVM program attributes. A Java program object is

treated like any CICS LE program, and you can leave the LANGUAGE attribute blank on the program resource definition. The actual program language is deduced at load time.

An illustration of the OS/390 JVM running under CICS is shown in Figure 24 .

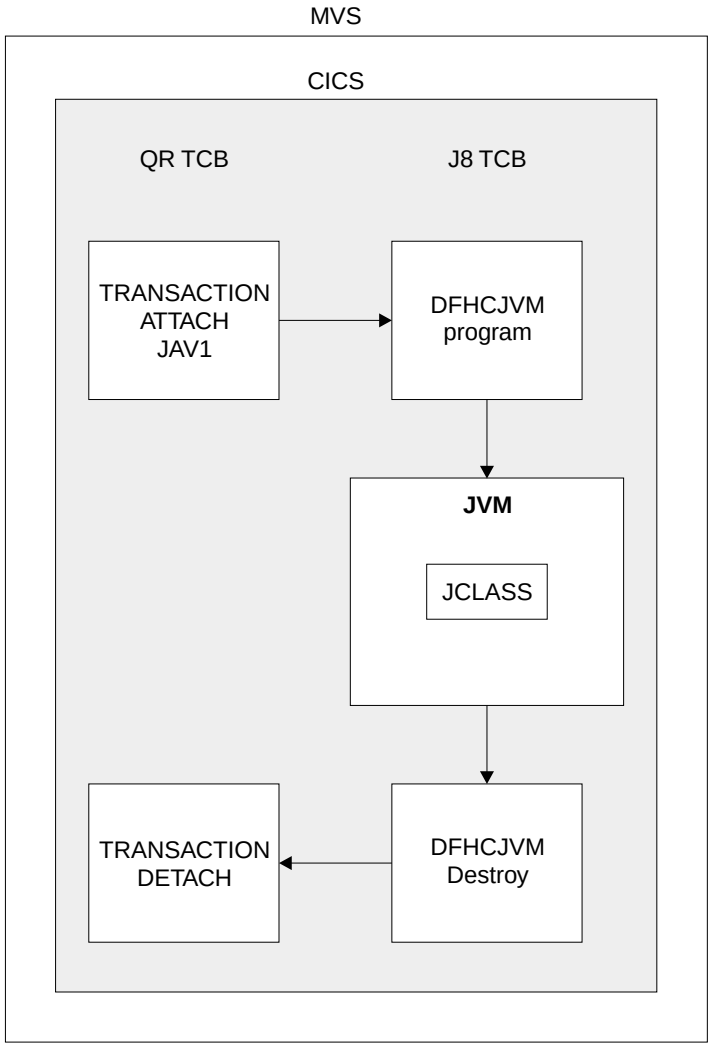


Figure 24. The OS/390 JVM and CICS

Benefits

CICS JVM support:

- Allows you to write Java application programs that use core Java classes that are not supported by ET/390 Java program objects.
- Enables CICS Transaction Server as a Java application server, allowing portability of Java applications between platforms

Requirements

The hardware and software requirements for CICS JVM are the same as for CICS TS generally.

To enable a JVM program to use abstract windows toolkit classes, you need one of the following:

- The X-windows capability provided by OS/390 Unix system services and TCP/IP
- The Remote Abstract Window Toolkit for Java.

Changes to CICS externals

There are changes to a number of CICS externals in support of the OS/390 JVM. These are described in:

- "Changes to system definition"
- "Changes to resource definition"
- "Changes to the application programming interface" on page 159
- "Changes to the system programming interface" on page 160
- "Changes to global user exits" on page 161
- "Changes to the exit programming interface" on page 161
- "Changes to user-replaceable modules" on page 162
- "Changes to CICS-supplied transactions" on page 163
- "Changes to monitoring and statistics" on page 163
- "Changes to problem determination" on page 163

Changes to system definition

There are no changes to CICS system initialization parameters. Instead, a new system dataset is introduced, from which CICS obtains the necessary JVM default options and environment variables. This has a DDNAME of DFHJVM, and refers to the DFHJVM member of the CICSTS13.CICS.SDFHENV partitioned data set (PDS). This contains parameters such as CLASSPATH, JAVASTACKSIZE, JAVA_COMPILER, and LIBPATH. The startup CICS JCL also requires a dummy DD statement for DFHCJVM.

The DD statements for the environment variables data set and the dummy data set must be included in the CICS startup JCL as follows:

```
DFHJVM    DD  DSN=CICSTS13.CICS.SDFHENV(DFHJVM),DISP=SHR
DFHCJVM   DD  DUMMY
```

You can override the environment parameters using the user-replaceable module, DFHJVMAT.

Changes to resource definition

There are new attributes added to the PROGRAM resource definition type. These are in the JVM attributes section of the CEDA DEFINE panel, part of which is shown in Figure 25 on page 159

```
 OBJECT CHARACTERISTICS                                  CICS RELEASE = 0530
  CEDA  View PROGram( BILL     )
   PROGram        : BILL
   Group          : TEST
   DEscription    :
   Language       :               CObol | Assembler | Le370 | C | Pli
⋮
   REMOTE ATTRIBUTES
⋮
   JVM ATTRIBUTES
   JVM            : No            No | Yes | Debug
   JVMClass       :
                  :
                  :
                  :
                  :
                                                    SYSID=HT61 APPLID=CICSHT61
PF 1 HELP 2 COM 3 END          6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 25. Changes to the CEDA DEFINE PROGRAM panel for JVM programs

The JVM attribute specifies whether the program is to run under the control of a CICS JVM. Specify YES (or DEBUG) only if the program is a Java bytecode program. You specify NO if the application program is a Java program object that is compiled and fully-bound in the CICS DFHRPL library concatenation. For both types of Java program, leave the program language blank. If you specify JVM(YES) or JVM(DEBUG) a JVMCLASS name is required, and the program's concurrency attribute must be specified as THREADSAFE.

Note: On an INQUIRE PROGRAM command, CICS returns C as the deduced language for Java program objects, and JAVA for as the CVDA for a JVM program

The JVMCLASS attribute specifies the main class in the CICS Java program to be executed by a JVM. On the CICS program resource definition, the JVM class name is restricted to 255 bytes. If you want to use larger class names, you can override the class name specified on the program definition, using the DFHJVMAT user replaceable module.

Changes to the application programming interface

A JVM program can invoke CICS services through the CICS API using the JCICS Java classes. There are some restrictions when using invoking JCICS classes from a JVM program. The restrictions affect the following EXEC CICS API commands:

ASSIGN
The PROGRAM, INVOKINGPROG, and RETURNPROG options are not available in the JCICS classes. On the other hand, if a JVM program calls a DFHRPL program (through the JCICS LINK or XCTL class) ASSIGN INVOKINGPROG and ASSIGN RETURNPROG issued by the DFHRPL program return the name of the calling JVM program.

LINK
The LINK command is supported by the JCICS classes, but a JVM program

cannot link to another JVM in the same CICS region because of the restriction that there can only be one JVM active for a user task. A link request to another local JVM program fails with an INVREQ response, RESP2 value 41.

A JVM program can link to another JVM program in another CICS region, where the JVM runs under another task.

A DFHRPL program can invoke a JVM program by issuing an EXEC CICS LINK command, and can pass COMMAREA.

Accessing the EXEC interface block

Fields in the CICS EXEC interface block (EIB) can be accessed in a JVM program in the same way as in a compiled and fully-bound JAVA program from the DFHRPL.

Changes to the system programming interface

There are changes to the system programming interface to support JVM programs. These changes affect the EXEC CICS INQUIRE and SET PROGRAM commands.

INQUIRE PROGRAM

The EXEC CICS INQUIRE PROGRAM command is not supported by the JCICS classes, and cannot by issued by a JVM program or a Java program object. For the other supported languages that can issue the INQUIRE PROGRAM command, the options are extended to include the following:

JVMCLASS
Returns the 255-character class name specified on the program resource definition.

JVMDEBUG(*cvda***)**
Returns a CVDA value indicating whether or not the JVM is to operate in debugging mode. The CVDA values are DEBUG and NODEBUG.

LANGDEDUCED(*cvda***)**
This existing option is enhanced to return an additional CVDA—JAVA for JVM programs. This CVDA does not apply to Java program objects, which CICS treats as C programs, because in this release of CICS TS with the current level of LE, CICS is unable to distinguish between C and Java program objects.

Note: It is intended in a future release to report C++ separately, and for Java program objects to return a CVDA of JAVA.

RUNTIME(*cvda***)**
Returns a CVDA value indicating the run-time environment of the program. The CVDA values are: JVM for a JVM program; LE370 for a program that uses the LE run-time libraries; NONLE370 for a program that uses a language-specific runtime environment; and UNKNOWN for a program that is not yet loaded.

For JVM programs, program options on the INQUIRE PROGRAM command are affected as follows:

- CEDFSTATUS, DATALOCATION, DYNAMSTATUS, EXECUTIONSET, REMOTENAME, REMOTESYSTEM, STATUS, and TRANSID are all supported for JVM programs.
- COBOLTYPE, SHARESTATUS, and LPASTATUS return NOTAPPLIC.
- COPY returns NOTREQUIRED.

- EXECKEY returns CICSEXECKEY.
- ENTRYPOINT, LOADPOINT, LENGTH, RESOUNT, and USECOUNT return a null address or zero, as appropriate.
- LANGUAGE returns the language defined on the program resource definition. Normally this is NOTDEFINED, because CICS determines the language from the load module when it loads the application program.
- PROGTYPE returns PROGRAM.

SET PROGRAM

The EXEC CICS SET PROGRAM command is not supported by the JCICS classes, and cannot by issued by a JVM program or a Java program object. For the other supported languages that can issue the SET PROGRAM command, the options are extended to include the following:

JVMCLASS(*name***)**
Specifies the 255-character class name of the Java program to be given control by CICS.

JVMDEBUG(*cvda***)**
Specifies a CVDA value to indicate whether or not the JVM is to operate in debugging mode.

RUNTIME(*cvda***)**
Specifies a CVDA value to indicate the runtime environment of the program. If you specify JVM (for a JVM program) you must also specify the JVM class name. If you specify NOJVM, JVMCLASS is ignored, and the runtime environment is unknown until CICS loads the program. CICS determines the runtime environment for a non-JVM program when it establishes the program language at load-time.

For JVM programs, program options on the SET PROGRAM command are affected as follows:

- CEDFSTATUS, EXECUTIONSET, and STATUS are supported.
- COPY is not applicable. JVM programs are loaded from HFS by the JVM, and not by the CICS loader.
- VERSION always returns OLDCOPY.

CREATE and DISCARD commands

The CREATE and DISCARD commands are not supported by the JCICS classes and cannot be used by any type of Java program. However, you can use these commands in an application program written in other supported languages to create and discard program resource definitions for Java programs.

Changes to global user exits

There are no new or changed global user exits for JVM programs, but note that the XPCFTCH global user exit is not invoked for a JVM program.

Changes to the exit programming interface

The CICS loader domain has nothing to do with JVM programs, which are loaded from HFS by the JVM, hence XPI loader functions are not supported for JVM programs.

The program management functions of the XPI are supported for JVM programs, but those parameters that relate the loader attributes of a program are not applicable. For example, CURRENT_LOAD_POINT, CURRENT_ENTRY_POINT, and AND LOAD_STATUS are ignored on the SET function and return null values on INQUIRE.

Changes to user-replaceable modules

There changes to the user-replaceable module (URM) for program autoinstall in support of JVM programs, and a new URM, DFHJVMAT, is introduced.

Program autoinstall

The parameter list passed to the user-replaceable module for program autoinstall is extended to include new parameters for JVM programs. The IBM-supplied copybooks for all the supported languages are updated to inclued these new parameters. These are DFHPGACD (assembler), DFHPGACO (COBOL), DFHPGACL (PL/I), and DFHPGACH (C). The added parameters are:

```
PGAC_JVM
PGAC_JVM_CLASS_LEN
PGAC_JVM_CLASS_DATA
PGAC_JVM_DEBUG
```

The JVM user-replaceable module, DFHJVMAT

This new URM enables you to modify the execution attributes of the JVM. The IBM-supplied default environment variables for the CICS JVM are supplied in the SDFHENV library, in the DFHJVM member, which contains the following:

```
CHECKSOURCE=NO
CICS_HOME=.
CLASSPATH=/usr/lpp/cicsts///cicsts13///classes/dfjwrap.jar:
          /usr/lpp/cicsts///cicsts13///classes/dfjcics.jar:
          /usr/lpp/java116/J1.1/lib/classes.zip:.
DISABLEASYNCGC=NO
ENABLECLASSGC=YES
ENABLEVERBOSEGC=NO
INVOKE_DFHJVMAT=YES
JAVASTACKSIZE=409600
JAVA_COMPILER=OFF
JAVA_HOME=/usr/lpp/java116/J1.1
LIBPATH=/usr/lpp/cicsts///cicsts13///lib:
        /usr/lpp/java116/J1.1/lib/mvs/native_threads
MAXHEAPSIZE=8000000
MINHEAPSIZE=1000000
NATIVESTACKSIZE=262144
STDERR=dfhjvmerr
STDIN=dfhjvmin
STDOUT=dfhjvmout
VERBOSE=NO
VERIFYMODE=NO
```

The environment variables are passed to the JVM when it is initialized. You can edit the DFHJVM member using TSO to modify these global defaults, and add new parameters.

When a user transaction invokes a JVM program, CICS:

1. Obtains the default values from the DFHJVM member of the SDFHENV data set (DD name DFHJVM).

2. Applies the JVMCLASS and JVMDEBUG attributes from the CICS program definition.
3. Invokes the user replaceable module DFHJVMAT, which can use getenv and setenv services to read and set all options.
4. Passes the options to the JVM.

For full details of the environment variables used by the CICS JVM, and for information writing about writing a DFHJVMAT URM, see the *CICS Customization Guide*.

Changes to CICS-supplied transactions

There are additional options on the CEMT INQUIRE and SET PROGRAM commands. These options, JVMCLASS, JVMDEBUG, and RUNTIME, are the same as the options added to the EXEC CICS INQUIRE and SET PROGRAM commands.

You can use the CEMT DISCARD command to remove unwanted resource definitions of Java programs from a CICS region.

Changes to monitoring and statistics

Monitoring data gathered from the program manager domain is also gathered for JVM programs, but program storage monitoring records are not incremented because the JVM program does not use CICS storage. Also, the program fetch time is not recorded, and there is no monitoring of JVM programs fetched from HFS by the JVM.

The CICS JVM runs on a J8 mode open TCB, and CICS monitoring domain records the CPU and dispatch time for each J8 mode TCB. This monitoring data should give some indication of the resource consumed by the task while running under the J8 TCB, most of which is used by the JVM. This J8 time includes any time spent processing CICS threadsafe API commands through the JCICS interface (all threadsafe activity remains on the J8 TCB).

For details of the new monitoring records created for JVM programs, see “Chapter 6. Monitoring, statistics, and enterprise management changes” on page 65.

Program statistics gathered from the program manager domain are also gathered for JVM programs. However, the only value recorded for a JVM program is the number of times used. The other statistics fields are either zero or blank.

The CICS sample statistics program, DFH0STAT, is updated to indicate which programs are JVM programs.

Changes to problem determination

There are new trace points at initialization and termination of the JVM environment, and when control passes to the JVM. However, once control passes to the JVM, it executes as an MVS JVM, and no CICS tracing takes place (unless the JVM program issues a JCICS call, which is traced just like any other CICS API command).

CICSPlex SM support

CICSPlex SM support for the JVM is provided by:

- New fields, JVM and JVMClass, on the BAS PROGDEF view.
- New fields, Runtime, JVM Class, and JVM Debug, on the operations PROGRAMD view.
- A new view, PROGRAMJ, which shows the JVM Class value for the named program.

Part 5. e-Business enablement for network computing

This Part describes the new function introduced to support network computing over open computer networks. It covers the following topics:

Chapter 16. Bridging to 3270 transactions

This chapter describes the enhancements to CICS support for bridging to 3270 transactions. It covers the following topics:

Overview

The 3270 bridge provides an interface so that you can run 3270-based CICS transactions without a 3270 terminal. Commands for the 3270 terminal are intercepted by CICS and replaced by a messaging mechanism that provides a bridge between the end-user and the CICS transaction.

With the bridge feature, an end-user or client application that may be executing outside the CICS environment can use transport mechanisms such as MQSeries or the Internet to access and run a CICS 3270-based user transaction.

The following diagram shows a CICS user transaction executing in a 3270 bridge environment.

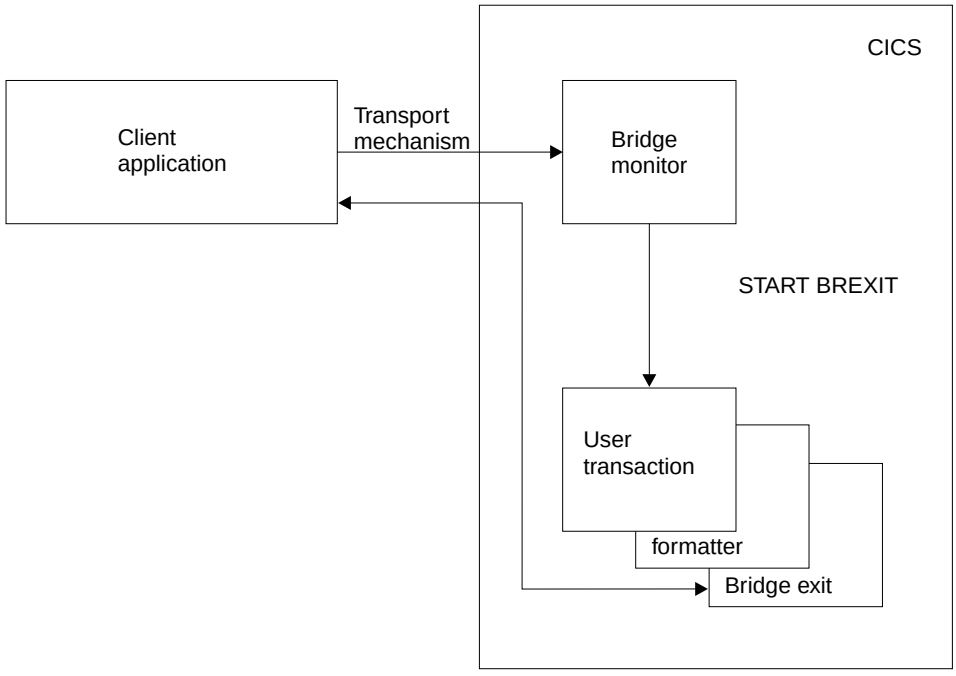


Figure 26. The CICS 3270 bridge environment

The client application can also be a CICS transaction using, for example, a temporary storage queue to pass 3270 requests and data to a user transaction executing in the same CICS region. This provides a similar function to the FEPI interface.

The user transaction can be an existing 3270 or BMS-based CICS transaction. It runs unchanged as if it were being driven by a real terminal.

Changes to the 3270 bridge

CICS introduces the following enhancements to the 3270 bridge function currently provided:

- New options of the START command are provided to initiate a user transaction and establish the bridge environment. A bridge transaction is no longer needed for this purpose.
- Some restrictions on the CICS commands issued by a user transaction are removed. Support is added for:
  - START TRANSID TERMID commands, where TERMID is the bridge facility. Some options are not supported and TRANSID must be local.
  - RETURN IMMEDIATE
  - INPUTMSG on RETURN, XCTL and LINK
  - SET TERMINAL ATISTATUS
- Writing a bridge exit has been simplified by delegating the handling of API commands to another user-replaceable program, called a formatter. This means that the bridge exit only needs to handle message input and output.

Components of the new 3270 bridge

User transaction
A user transaction is a 3270 CICS transaction.

CICS 3270 bridge mechanism
The CICS 3270 bridge mechanism is the CICS function that allows a user transaction to be run without a VTAM 3270 terminal. Terminal input/output commands are intercepted by a bridge exit that emulates the real terminal by passing the commands packaged as messages to the end-user or client application.

Client application
A client application is a program, usually executing outside CICS, and possibly outside MVS, that uses the CICS 3270 bridge mechanism to run a user transaction. If a client application runs inside CICS, it can also perform the functions of a bridge monitor.

Transport mechanism
A transport mechanism is used by the client application to pass messages to CICS. MQSeries, the Web, CICS temporary storage, and CICS transient data are all examples of transport mechanisms. Some transport mechanisms have separately definable queues.

Messages
A message contains information that provides all or part of the data needed to run a 3270 user transaction. Data originally written to the 3270 screen by the user transaction is packaged into messages and sent to the client application. Data originally read from the 3270 screen by the user transaction is obtained from messages sent by the client application.

Bridge monitor

A bridge monitor is usually a long-running CICS task that is associated with a specific transport mechanism, or a specific queue in a transport mechanism. When a message arrives requesting a CICS user transaction, the bridge monitor issues a START BREXIT TRANSID command to start the requested user transaction in a bridge environment where it will execute in association with the specified bridge exit.

The bridge monitor must ensure that the requested transaction is started, and started only once. It can receive confirmation messages from the bridge exit after a successful start.

A bridge monitor can be designed as a long running CICS task, to start any user transaction, or it can be a short running task that is designed to start only a single transaction. The CICS Web interface implementation of the 3270 bridge uses a short running task.

Bridge environment

The bridge environment includes the CICS components that are needed to run a user transaction using the CICS 3270 bridge mechanism. A bridge exit and a bridge facility are essential components of the bridge environment.

Bridge exit

A bridge exit is a user replaceable program that emulates the 3270 terminal API. It usually does this by passing messages to the transport mechanism, so there is usually a different bridge exit for each transport mechanism.

A bridge exit can be generic if it is designed to run any user transaction, or specific if it is designed to work with a single user transaction.

In CICS TS Release 2, all 3270 terminal API requests were passed to the bridge exit. This provides a simple interface for specific bridge exits, but is very complicated in the generic case.

To reduce the complexity, the bridge exit can be designed to handle some of the requests, with all the terminal API requests being passed to a formatter, which is another user-replaceable program.

If used, the name of the formatter is obtained from the BRXA_FORMATTER field in the bridge exit communication area (BRXA), and the bridge exit is only called for requests that require input or output of data. If a formatter is not specified, the bridge exit is called for all requests.

A bridge exit is always called for the following requests:

- User transaction initialization
- User transaction bind
- User transaction termination
- User transaction abnormal termination
- Read and Write message
- Syncpoint requests (optional)

A formatter is called for the following requests:

- SEND (Terminal Control and BMS)
- RECEIVE (Terminal Control and BMS)
- CONVERSE
- FREE
- ISSUE DISCONNECT

- ISSUE ERASEAUP
- RETRIEVE (in some cases)

All requests made in a bridge exit execute in the same unit of work as the user transaction. Therefore, any recoverable requests made by the bridge exit are committed or rolled back at the same time as the user transaction resources.

The bridge exit area, which forms an interface between the bridge exit and CICS, is extended, and is described fully in *CICS External Interfaces Guide*. This interface is defined by CICS and must be used by all bridge exits.

The messaging interface between the bridge exit and the remote resource or the end user is not defined by CICS. You may define this interface to fit your own environment.

However, we have defined an interface that is used by both the MQSeries and the CICS TS/TD sample exits. This interface uses a message header (MQCIH) and message vectors for each API request. It is described in *CICS External Interfaces Guide*. You can use this interface definition as a basis for your own implementation using other transport mechanisms.

Bridge exit area
The bridge exit area (BRXA) is the communication area between the bridge exit and CICS. It is a CICS COMMAREA, and is subject to normal length constraints for a COMMAREA..

Bridge facility
The bridge facility is an emulated 3270 terminal, replacing a real 3270, that is visible only to the user transaction and does not appear in response to CEMT I TERM or CEMT I TASK.

It has a dynamically created TERMID that can be used, for example, as the basis of a unique name for a TD or TS queue.

Sample code provided

The following sample code is available:

DFH0CBRE
A COBOL bridge exit program that uses CICS temporary storage or transient data queues to pass messages (in MQCIH format) to the end-user application (another CICS application).

DFH0CBRF
A COBOL bridge exit formatter designed to work with DFH0CBRE. This builds and interprets BRMQ message vectors.

DFHWBLT
An object code exit that allows you to access a CICS transaction from the World Wide Web. This exit uses the CICS Web interface support described in the *CICS Internet Guide*.

Copybooks
CICS provides a number of copybooks for use with the sample programs:

DFH0CBRD
COBOL copybook used by DFH0CBRE.

DFH0CBRU
COBOL copybook used by DFH0CBRF.

DFHBRSDx
Copybooks in all supported languages defining the interface between the bridge monitor and the bridge exit.

DFHBRMHx
Copybooks in all supported languages defining the message header included in all messages passed between the sample bridge exit and the client application. Constants are also supplied for these copybooks.

DFHBRMQx
Copybooks in all supported languages defining the command vectors in the messages passed between the sample bridge exit and the client application. Constants are also supplied for these copybooks.

See the *CICS External Interfaces Guide* for more information about the formats of message headers and vectors.

Security considerations

Bridge monitor transaction
The bridge monitor issuing the EXEC CICS START BREXIT command must have authority for the following:

1. Surrogate authority to issue a START for the specified USERID.
2. Authority to load the bridge exit specified by the BREXIT parameter. If the default bridge exit specified in the TRANSACTION resource definition is used, CICS does not perform a security check.

User transaction
The user transaction, running under the specified USERID, must have the authority to run the transaction.

Bridge exit
The bridge exit responsible for reading the messages can obtain a password from the message (note that the user ID is not in the message). The bridge exit can then issue a VERIFY command with the password. This can be done for all messages, or just the first.

Running 3270 transactions in a bridge environment

A user transaction is started directly by a bridge monitor transaction using the START BREXIT TRANSID command.

The user transaction is initialized in a bridge environment; all 3270 terminal commands are intercepted and passed to the named bridge exit that emulates the 3270 terminal by packaging the commands into messages and passing them to the transport mechanism for delivery to a client application.

A formatter may be used to package the commands into messages, to simplify the role of the bridge exit as a message handler only.

The bridge monitor transaction is normally a long running task associated with a message queue. It looks at the contents of each message to search for requests for new work, and identifies the name of the user transaction to run. It then starts the requested transaction and checks that it has started successfully.

The client application, which may be executing anywhere accessible by the transport mechanism, extracts the 3270 data from the messages and constructs reply messages, containing 3270 data and commands, to pass to the bridge exit to satisfy the terminal commands.

Migration considerations

Bridge exits and transport mechanism monitors written to run on the earlier release require modification as a result of the following changes.

Changes to the monitor
The monitor (now called a bridge monitor) does not need to start a bridge transaction, but must use the new START BREXIT TRANSID command to start the user transaction directly.

Changes to the bridge exit
The following changes affect the bridge exit:

- The initialization call of the bridge exit is now divided into two parts: non-recoverable (Init) and recoverable (Bind) initialization.
- The retrieval of data in the bridge exit (using EXEC CICS RETRIEVE) is now replaced by data passed in the BRXA from the BRDATA specified by the EXEC CICS START BREXIT command.
- The bridge exit can now issue an EXEC CICS VERIFY command for the password.
- The BRXA-PASSWORD option is withdrawn.

Changes to the bridge transaction
Bridge transaction resource definitions are no longer valid.

CSD migration
The bridge transaction definitions provided in CICS TS Release 2 are moved to the CSD compatibility group, DFHCOMP7.

Bridge transaction resource definitions are no longer required.

Benefits of bridging to 3270 transactions

The 3270 bridge allows you to introduce new GUI front ends to access existing 3270-based CICS applications without modifying them. This means that you can concentrate your efforts on the new user interfaces, and avoid, or at least postpone, rewriting stable mainframe applications.

You don’t need to restructure your applications to separate the business logic from the presentation logic; the bridge effectively does this for you.

The same existing applications can be used both by 3270 terminals, and by the new end-user applications. This will, therefore, allow a phased migration of users from the 3270 applications to the new end-user applications.

Applications written for 3270 terminals can be run on CICS regions without using VTAM support.

The bridge can process commands faster than existing front end methods, such as FEPI and EPI, because the terminal emulation is part of the same CICS transaction.

Unlike other front-end methods, there is only a single unit of work. This means that the bridge can use a recoverable MQSeries queue. This greatly simplifies recovery.

Requirements for the 3270 bridge

The hardware and software requirements for bridging to 3270 transactions are the same as for CICS generally plus, if you intend to use the MQSeries supplied bridge exit, MQSeries for MVS/ESA™ Version 2.1.

Resource usage

The 3270 bridge mechanism allows you to drive 3270 applications without a real 3270 terminal. Although overall resource usage may not change significantly, the resources involved may be different.

For example, the real terminal is replaced by a bridge facility, but the storage required for the definition is the same. A saving occurs if the bridge facility is not retained for the same length of time as the equivalent real terminal.

The 3270 bridge has some similarity in function to the FEPI interface, but has no separate controlling task in CICS, and a correspondingly lower task-attach overhead. The controlling task is now outside CICS, transferring this resource requirement to another platform. Terminal requests do not need to be processed by VTAM, as with FEPI, but are passed directly to the bridge exit, saving about 60% pathlength.

Some additional resource requirements are:

- RACF validation is required each time the bridge exit issues a VERIFY command, which may occur on each request. This increases the overall pathlength compared to terminal usage, especially if PassTickets are used.
- Storage is required for the bridge exit, terminal, and state data, including the bridge exit communication area (BRXA). This remains in storage for the duration of the transaction, increasing 31-bit task storage usage for the user transaction.

Changes to CICS externals

This section describes the following changes to CICS externals caused by the enhancements to the 3270 bridge.

- “Changes to resource definition”
- “Changes to the application programming interface (API)” on page 174
- “Changes to the system programming interface (SPI)” on page 175
- “Changes to the exit programming interface (XPI)” on page 176
- “Changes to user-replaceable modules” on page 176
- “Changes to CICS-supplied transactions” on page 176

Changes to resource definition

Changes are made to the TRANSACTION resource definition, as follows:

- The BREXIT parameter is now defined on the user transaction definition. The bridge transaction introduced in CICS TS Release 2 is no longer required to run

a transaction in the 3270 bridge environment. Existing definitions are moved to the CSD compatibility group, DFHCOMP7.

- The BREXIT parameter now defines the default bridge exit to be used for a transaction started in the bridge environment. This is intended to define a specific bridge exit.

Changes to the application programming interface (API)

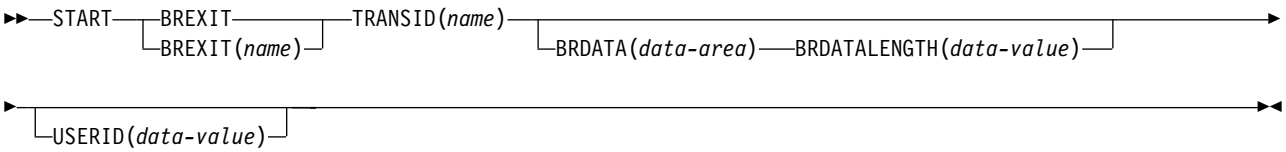
Changes are made to the following interfaces to support the 3270 bridge:

- START BREXIT
- ASSIGN

START BREXIT command

The EXEC CICS START BREXIT command starts a task immediately in the local CICS region, and initializes the specified transaction (TRANSID) and bridge exit (BREXIT).

START BREXIT



Conditions: INVREQ,LENGERR, NOTAUTH, TRANSIDERR, USERIDERR

The options available on the EXEC CICS START BREXT command are:

BREXIT(*name***)**
Specifies the name (1-8 characters) of the bridge exit to be associated with the started task. If no name is specified, the value of the BREXIT parameter on the TRANSACTION resource definition for TRANSID is used.

BRDATA(*data-area***)**
Specifies the data to be passed to the bridge exit specified by BREXIT when the task is started.

BRDATALENGTH(*data-value***)**
Specifies a halfword binary data value that is the length of the BRDATA to be passed to the bridge exit specified by BREXIT when the task is started.

TRANSID(*name***)**
Specifies the symbolic identifier (1–4 characters) of the transaction to be executed by a task started as the result of a START BREXIT command. The transaction starts in the 3270 bridge environment, and executes in association with the bridge exit specified in BREXIT.

USERID(*data-value***)**
Specifies the user ID under whose authority the started transaction is to run.

The exception conditions are:

INVREQ
RESP2 values:

11 An attempt was made to ship a START BREXIT request.

12 A START BREXIT request has failed.

18 A USERID is specified and the CICS external security manager interface is not initialized.

INVREQ can also occur (but RESP2 is not set) if the START command is not valid for processing by CICS.

Default action: terminate the task abnormally.

LENGERR
Occurs if BRDATALENGTH is not greater than zero.

Default action: terminate the task abnormally.

NOTAUTH
RESP2 values:

7 A resource security check fails on TRANSID (*name*).

9 A surrogate user security check fails on USERID (*name*).

The security access capabilities of the transaction that issued the command do not allow the command to be performed with the value specified in the USERID option.

Default action: terminate the task abnormally.

PGMIDERR
occurs if no name is supplied by the BREXIT option and the transaction definition for TRANSID does not provide a default BREXIT name.

Default action: terminate the task abnormally.

TRANSIDERR
occurs if the transaction identifier specified in a START BREXIT command has not been defined to CICS.

Default action: terminate the task abnormally.

USERIDERR
RESP2 values:

8 The specified USERID is not known to the external security manager.

10 The external security manager is in a state such that CICS cannot determine whether a specified USERID is valid.

ASSIGN command

Changes to the ASSIGN command

The BRIDGE option introduced in CICS TS Release 2 is retained but the value returned has a different meaning. In the earlier release, BRIDGE returns the name of the bridge transaction. The BRIDGE option now returns the name of the bridge *monitor* transaction. The bridge transaction is no longer required to run a transaction in the 3270 bridge environment.

Changes to the system programming interface (SPI)

The INQUIRE TASK command is changed to support the enhancements to the 3270 bridge.

Changes to the INQUIRE TASK command

The INQUIRE TASK BRIDGE option introduced in CICS TS Release 2 is retained, but the value returned has a different meaning. In the earlier rlease, BRIDGE returns the name of the bridge transaction. The BRIDGE option now returns the name of the bridge *monito*r transaction. The bridge transaction is no longer required to run a transaction in the 3270 bridge environment.

Changes to the exit programming interface (XPI)

The INQUIRE_CONTEXT function is modified to support the enhancements to the 3270 bridge.

Changes to the INQUIRE_CONTEXT command

The value returned by the BRIDGE_TRANSACTION_ID parameter now has a different meaning. In CICS TS Release 2, BRIDGE_TRANSACTION_ID returns the name of the bridge transaction. The BRIDGE_TRANSACTION_ID option now returns the name of the bridge *monitor* transaction. The bridge transaction is no longer required to run a transaction in the 3270 bridge environment.

Other parameters of INQUIRE_CONTEXT are unchanged.

Changes to user-replaceable modules

The following changes are made to user-replaceable modules:

- The interface to the existing bridge exit user-replaceable module is extended to support the enhancements to the 3270 bridge. See *CICS External Interfaces Guide* for detailed information about the BRXA interface.
- A formatter user-replaceable module can, optionally, be used to handle all the API commands, leaving the bridge exit to handle message input and output only.

Changes to CICS-supplied transactions

The CEMT INQUIRE TASK command is changed to support the enhancements to the 3270 bridge.

Changes to the CEMT INQUIRE TASK command

The INQUIRE TASK BRIDGE option introduced in CICS TS Release 2 is retained, but the value returned for the BRIDGE option has a different meaning. In the earlier release, BRIDGE returns the name of the bridge transaction, but now it returns the name of the bridge monitor transaction. The bridge transaction is no longer required to run a transaction in the 3270 bridge environment.

Chapter 17. Support for the secure sockets layer

This chapter describes CICS Web interface support for the secure sockets layer. It covers the following topics:

Overview of SSL

The secure sockets layer (SSL) is a protocol for exchanging secure information across an insecure network such as the Internet. It was invented by the Netscape Communications Corporation and is described in some detail at their website.

Caution

These URLs are subject to change.

(See `http://www.netscape.com/newsref/ref/rsa.html` for an overview, or `http://home.netscape.com/eng/ssl3/` for a detailed specification).

CICS supports SSL directly using the system SSL function of OS/390 Release 7, so you do not need an intermediate SSL server between the CICS application and the Internet.

SSL authentication

When client and server programs communicate using the SSL protocol, an initial handshake occurs in which the server and client authenticate each other and then negotiate an encryption technique. After the handshake, all the data that flows between the client and server is encrypted using the negotiated technique.

The authentication is performed by an exchange of **certificates**, which are blocks of data in a format described in ITU-T Standard X.509. The server certificate is mandatory, but the client certificate is optional—it is up to the server to decide whether to accept a connection from a client without a certificate.

The X.509 certificates are digitally signed by an external authority known as the **certificate authority**. Signing is done by partially encrypting the certificate with the certificate authority's private key. A user of the certificate is assured of the origin of the certificate when it is successfully decrypted by the certificate authority's public key. To obtain a certificate, search the Web using a search string such as **certificate authority**, and apply to one of the authorities listed.

In CICS, the required server certificate and related information about certificate authorities are kept in a **key database file** that is stored within the hierarchical file system (HFS) of OS/390. The keyring file is created and maintained by the **gskkyman** utility shipped with OS/390.

Benefits of secure sockets layer

Using the secure sockets layer when communicating with a client over the Internet allows you to transfer secure data such as credit card numbers, PIN numbers, or sensitive financial data without the risk that the data can be intercepted or changed while in transit.

Furthermore, you can force your clients to identify themselves by requiring that they have a client certificate, which assures you that they are a particular client who has obtained a certificate from a recognized certificate authority. You can also configure your external security manager, such as the Security Server for OS/390 (RACF), to map requests from certificate holders to specific user IDs, so that you can grant or deny them access to particular CICS resources, through the normal CICS security mechanisms.

Using SSL within CICS means that you do not have to purchase, install, or configure an external Internet server solely for the purpose of providing secure CICS transactions across the Internet.

Requirements

Secure sockets layer does not require any additional hardware, but performance is greatly improved if appropriate cryptographic hardware is installed.

The software required for secure sockets layer is OS/390 Release 7, plus the required PTF for APAR PQ23421 to enable the CICS SSL support.

Changes to CICS externals

There are a number of changes to CICS externals to support the secure sockets layer:

- "Changes to installation"
- "Changes to system definition" on page 179
- "Changes to application programming interface" on page 180
- "Changes to user-replaceable modules" on page 180
- "Changes to samples" on page 180
- "Changes to CICS-supplied transactions" on page 180
- "Changes to problem determination" on page 181

Changes to installation

To use secure sockets layer, install a key database in the OS/390 UNIX System Services hierarchical file system (HFS). This file contains your system's private and public key pair, together with your server certificate and the certificates for all the certificate authorities that might have signed the certificates you receive from your clients.

You create the keyring file using the key management utility **gskkyman**, which you execute under the UNIX System Services shell in TSO.

For further information about gskkyman, see the *OS/390 Cryptographic Services System SSL Programming Guide and Reference*, SC24-5877.

Changes to system definition

There are four new system initialization parameters to support secure sockets layer. These are:

KEYFILE=*key-database-path-name*
specifies the fully-qualified HFS pathname of the key database file created by the gskkyman utility program for this CICS region. When you specify this parameter, the CICS region user ID must be authorized to read the specified HFS file.

ENCRYPTION={WEAK|NORMAL|STRONG}
specifies the level of encryption you want to use for TCP/IP connections using the secure sockets layer. The parameter selects the list of ciphers that are negotiated with the client program to choose the SSL encryption technique, keysize, and message authentication code (MAC) . You can specify an option only if you have the underlying encryption support in the OS/390 operating system. Possible values are:

WEAK
Specifies the following list of ciphers:

- RC4 encryption with a 40-bit key and an MD5 MAC
- RC2 encryption with a 40-bit key and an MD5 MAC
- No encryption with an MD5 MAC
- No encryption with an SHA MAC.

This option is available in all countries with OS/390 Release 7.

NORMAL
Specifies the following list of ciphers:

- DES encryption with a 56-bit key and an SHA MAC
- RC4 encryption with a 40-bit key and an MD5 MAC
- RC2 encryption with a 40-bit key and an MD5 MAC
- No encryption with an MD5 MAC
- No encryption with an SHA MAC.

This option is available with OS/390 Release 8 in all countries except France.

NORMAL is the default value, but if you specify NORMAL under OS/390 V2R7, and DES encryption is not available, CICS automatically resets it to WEAK.

STRONG
Specifies the following list of ciphers:

- Triple DES encryption with a 168-bit key and an SHA MAC
- RC4 encryption with a 128-bit key and an MD5 MAC
- RC4 encryption with a 128-bit key and an SHA MAC
- DES encryption with a 56-bit key and an SHA MAC
- RC4 encryption with a 40-bit key and an MD5 MAC
- RC2 encryption with a 40-bit key and an MD5 MAC
- No encryption with an MD5 MAC

- No encryption with an SHA MAC.

STRONG is available with OS/390 Release 7 in the USA and Canada only.

SSLDELAY={600|*number*}
Specifies the length of time in seconds for which CICS retains session IDs for secure socket connections. Session IDs are tokens that represent a secure connection between a client and an SSL server.

SSLTCBS={8|*number*
Specifies the number of CICS subtask TCBs to be dedicated to processing secure sockets layer connections, in the range 0 to 255. The parameter controls the number of simultaneous SSL connections that CICS can establish. A value of 0 means that no SSL connections are to be established.

This number is independent of and in addition to the TCBs specified on the MAXOPENTCBS system initialization parameter.

Changes to application programming interface

There is a new API command for SSL:
- EXTRACT CERTIFICATE

EXEC CICS EXTRACT CERTIFICATE allows an application program to obtain information about an inbound client request. The options allow you to obtain details of any certificates being used for SSL authentication. See the *CICS Application Programming Reference* for a full description of this command.

Changes to user-replaceable modules

The userid field (`wbra_userid`) in the COMMAREA passed to the CICS Web Interface analyzer user-replaceable module is changed from an output parameter to an input and output parameter. On input to the Analyzer, if the client has provided a client certificate, `wbra_userid` is set to is the userid associated with the certificate, if one is available. Otherwise, the field is set to binary zeroes. The analyzer may leave the field unchanged, or may replace it with a different value. The analyzer program can use the `EXEC CICS EXTRACT CERTIFICATE` command to extract details about the client certificate to help choose a different user ID.

If `wbra_userid` contains a valid userid on return from the analyzer (which may be the one deduced from the client certificate if the field is left unchanged) that is the user ID under which the alias transaction will run.

Changes to samples

A new sample COBOL program, DFH0WBCA, is available to demonstrate how to use client certificates. This uses the `EXEC CICS EXTRACT CERTIFICATE` command.

Changes to CICS-supplied transactions

A new function of the CEMT SET command CEMT SET TCPIPSERVICE is introduced for the CICS Web Interface enhancements. It is used to enable a TCP/IP service for SSL, either with or without client authentication. See the *CICS Supplied Transactions* for a full explanation of this function.

Changes to problem determination

There are new messages and trace entries to aid problem determination.

Messages

The following messages are introduced in support of secure sockets layer:

- A message when the KEYFILE system initialization parameter is specified, but the specified key database file cannot be opened, for one of the following reasons:
  - The specified key database file does not exist
  - The CICS region is not authorized to read the key database file
  - The key database file's password has expired.
- A message when the ENCRYPTION=STRONG system initialization parameter is specified, but the North American encryption feature is not installed.
- A message when a TCP/IP port is enabled for SSL, but the KEYFILE system initialization parameter is not specified.

Trace

SSL-related parameters are traced on entry and exit from the sockets domain, when relevant.

New trace entries in the security domain trace entry to, and exit from, the `ADD_USER_WITH_CERTIFICATE` function call, in addition to relevant calls to the external security manager.

Chapter 18. CORBA client support

This chapter describes CICS support for IIOP inbound to Java applications. It covers the following topics:

Overview

CICS introduces support for CORBA clients using the IIOP protocols to access Java server programs.

The Internet Inter-ORB protocol (IIOP), is an industry standard that can be used to provide communication between object-oriented application programs executing on different processors. It is part of the Common Object Request Broker Architecture (CORBA) specification, supporting distributed objects in a TCP/IP network.

CORBA is an architecture for distributed object middleware that separates client and server objects with a formal interface definition, and IIOP defines the message formats and protocols used in a CORBA distributed environment.

CICS supports inbound requests to Java programs, using the IIOP protocol. You can read about building CICS Java applications using the CICS Java classes in the *CICS Application Programming Guide*.

A subset of the full CORBA function is provided, suitable for distributed objects that have evolved from existing CICS applications and therefore have the following characteristics:

- State by virtue of their explicit use of CICS resources, rather than state that is managed by the Object Request Broker(ORB). State is initialized at the start of each method call and referenced by explicit method parameters.
- Transaction and security contexts managed by CICS facilities, so CORBA services are not provided.
- CICS services used to reference distributed applications, so outbound object references are not supported.
- Applications and their interfaces predefined, so the Dynamic Skeleton Interface (DSI) is not supported.

Each method call is implemented as a CICS transaction, and the existing TOR/AOR structure is retained. CICS can select the best server to meet the client's request, or balance the loading of requests between servers.

For a full description of the IIOP support, see the *CICS Internet Guide*.

Benefits

This function provides:

- Improved application development productivity. Use of a distributed object model gives well-defined (strongly typed) interfaces for applications with inheritance and polymorphism characteristics.
- Stronger type checking at compile time and reduction in application error compared with the use of untyped COMMAREAS used in CICS ECI calls. Run-time type checking is also enabled for use in polymorphic implementations.
- The use of vendor-independent client platforms, giving true separation of server and client environments.

Requirements

The hardware and software requirements for this function are the same as for CICS TS with:

- VisualAge for Java, Enterprise Edition for OS/390 (ET/390) to bind Java byte-code into Java program objects that can be run in CICS, or the MVS Java Virtual Machine (JVM) running in CICS.
- **Full function** OS/390 UNIX System Services configured to run ET/390.

Service PTFs for the following APARs are required before using Java language support:

- OW31036 (Bind with long object names)
- OW31718 (DFSMS 1.4 Invalid loader storage check)
- OW31924 (IEW2333E Invalid syntax in IMPORT control statement)
- OW32111, OW32261, and OW32334 (IEW2900T E913 Binder abnormal termination)
- OW33782 (DFSMS 1.4 DESERV to set output buffer length for PDSE access)
- OW34052 (Load optimization for C_WSA for DLLs in dynamic LPA)
- PQ08747 (For LE to support double precision floating-point in single thread)
- PQ17512 (0C4 when signal occurs in stack extension boundary)
- PQ19340 (CICS ABEND failure caused by LE condition handler)

Changes to CICS externals

This section describes the following changes to CICS externals caused by the IIOP inbound function.

Changes to resource definition

Two new CICS resource definitions are used to support IIOP:.

REQUESTMODEL
provides the connection between an IIOP inbound request and the name of the CICS transaction that is to be initiated.

TCPIPSERVICE
defines the TCP/IP attributes of the CICS IIOP service. A TCPIPSERVICE definition is required for each port that CICS will listen on for IIOP requests.

For details of the TCPIPSERVICE and REQUESTMODEL resource definitions, see the *CICS Resource Definition Guide*.

Changes to the system programming interface (SPI)

The following new SPI commands are added to support the IIOP inbound function:
- INQUIRE REQUESTMODEL
- CREATE REQUESTMODEL
- DISCARD REQUESTMODEL

For further details of these SPI commands, see the *CICS System Programming Reference*

Changes to user-replaceable modules

A new user-replaceable module is called before the selected TRANSID is executed. This program allows you to examine the incoming IIOP message and assign a CICS USERID under which the request is run. The program name is defined in the TCPIPSERVICE definition, defaulting to DFHXOPUS. It is invoked for all IIOP messages.

Changes to CICS-supplied transactions

CEMT is changed to support the new REQUESTMODEL and TCPIPSERVICE resources.

The following new transactions are supplied:

CIOD The default transaction used for a method request if no match is found with any defined REQUESTMODELs.

CIOF The default transaction used for a GenericFactory request.

CIOR This is the initial transaction, attached to process an incoming message.

Changes to samples

Sample Java server and client programs are supplied.

Changes to CICS-supplied utilities

- An IDL to the Java compiler utility is provided in dfjcidl.jar.
- The GenFacIOR utility is provided in dfhcorb.jar for use by client applications.

See the *CICS Internet Guide* for more information about these utilities.

Changes to problem determination

Changes are made to CICS problem determination services to support the inbound IIOP function.

Messages

New CICS messages in the DFHCZxxxx range are issued by the inbound IIOP function.

Trace

New CICS trace entries are written by the inbound IIOP function.

Abend Codes

New Abend codes, AIOx, are generated by the inbound IIOP function.

Security

The IIOP inbound function invokes the user replaceable module DFHXOPUS to supply a USERID under which the request will be run, providing powerful attributes not available to normal end-users. No application code will be executed under the powerful USERID.

CICSPlex SM support

CICSPlex SM supports IIOP inbound functions for Java by providing:

- A new BAS definition object **RQMDEF** that associates an inbound IIOP request with a set of execution characteristics such as security or priority, and with monitoring and accounting data
- New operate views:

  RQMODEL General view of request models.

  RQMODELD A detailed view of a specific request model.

  RQMODELS A summary view of request models.
- New resource tables:
  - CRESRQMD
  - ERMCRQMD
  - RQMDEF
  - RQMINGRP
  - RQMODEL

Chapter 19. CICS Web support enhancements

This chapter describes the enhancements to CICS Web support. It covers the following topics:

Overview

CICS Web support is restructured as a new CICS domain, conforming to the standard CICS domain architecture. In addition to the restructure, it is further enhanced by:

- The addition of new EXEC CICS application programming interface commands for the manipulation of Web entities
- Improvements to the definition and management of HTML templates
- Using the 3270 bridge enhancements in a Web 3270 environment
- Enabling CICS Web support to handle more than 32K of data, inbound and outbound
- Simplified administration through new resource type, TCPIPSERVICE.

EXEC CICS API for the CICS Web interface

The current interface to the template manager is by means of an EXEC CICS LINK to program DFHWBTL. A new suite of EXEC CICS commands is provided to allow user application programs to manipulate templates without the need to link to DFHWBTL. EXEC CICS commands are also provided to retrieve information from inbound HTTP requests, and to build HTTP responses.

HTML templates

- HTML templates are now called document templates, and can be stored in the following places:
  - A CICS file
  - An extrapartition transient data (TD) queue
  - A recoverable auxiliary TS queue
  - CICS program storage
  - A URM-managed repository (for example, DB2)
  - An MVS partioned data set (PDS).

  Document templates are defined to CICS in a DOCTEMPLATE resource definition type. EXEC CICS CREATE and DISCARD commands are extended to allow installed DOCTEMPLATE resource definitions to be managed dynamically.

Improvements to 3270 support on the Web

Web 3270 transactions can now use EXEC CICS START against the same facility.

Removal of 32KB restriction

In earlier releases, the CWI uses the EXEC CICS LINK command to pass data between CICS code and user-written programs. EXEC CICS LINK communication areas (COMMAREAs) have a maximum size of 32KB, so the CWI is also restricted to 32KB for inbound data (some CWI applications can use a work-around to exceed 32KB of outbound data).

The use of a CICS temporary storage queue to store inbound requests removes the 32KB restriction for inbound data.

TS queues are used to store inbound data and Web documents created by applications; a TS queue generic prefix is required for each Web TCPIPSERVICE. The TS queue prefix is defined in the TCPIPSERVICE resource type. Each TS queue prefix must have a corresponding TSMODEL definition to meet your system and application requirements. Most Web transactions exchange relatively small amounts of data Therefore, a TSMODEL defining the TS queues as main TS might be appropriate. For applications handling large amounts of data, a model specifying main TS might be unacceptable due to storage constraints.

Support for the HTTP 1.0 Keep-Alive header

HTTP 1.0 persistent connection (Keep-Alive) support improves network performance (particularly when SSL is being used) by allowing one sockets connection between CICS and an HTTP client to process multiple HTTP requests. CICS supports only the HTTP 1.0 Keep-Alive implementation of persistent connections, not the HTTP 1.1 connection.

To enable HTTP Version 1.0 persistent connection support, set the Socketclose keyword on the associated TCPIPSERVICE definition either to NO (that is, CICS does not close the socket), or to a specific interval, after which CICS closes the socket if no new HTTP request arrives.

When “OK HTTP” responses are returned, this tells the browser that further HTTP requests can be submitted over the same socket connection.

Simplified administration

Administration is simplified by the introduction of a new resource type, TCPIPSERVICE, to replace the CWBC transaction and its associated data set, DFHWBCD, which are now obsolete. See the *CICS Transaction Server for OS/390 Migration Guide* for more information about migration implications, and the *CICS Resource Definition Guide* for information about defining TCPIPSERVICE resource definitions.

Benefits

The benefits provided by CICS Web support enhancements are:

- CICS Web support now handles more than 32KB of data, inbound and outbound (see “Removal of 32KB restriction”).
- System administration is simplified by the removal of transaction CWBC, and its associated VSAM data set, DFHWBCD.
- HTML templates are replaced by document templates, which are a CICS resource, definable using RDO or the EXEC CICS CREATE command.

- Document templates can be stored in CICS TS and TD queues, load modules, VSAM files, and other user-managed storage.
- Support of the HTTP 1.0 Keep-Alive header allows CICS Web applications to take advantage of the significant reduction in network traffic which this header allows. The improved performance is particularly marked when SSL is used to secure the HTTP flows.
- Improvements are made to 3270 support on the Web (see “Improvements to 3270 support on the Web” on page 187).
- The new CWI API simplifies the retrieval and building of HTTP requests and responses (see “EXEC CICS API for the CICS Web interface” on page 187).

Requirements

The hardware requirements for CICS Web support are the same as for CICS TS generally.

The software required is OS/390 Release 5 or later, plus a required PTF for APAR PQ21197, to enable you to run CICS with TCPIP=YES. The required PTFs are:

- UQ23629 for OS/390 Version 2 Release 5
- UQ23630 for OS/390 Version 2 Release 6
- UQ23628 for OS/390 Version 2 Release 7

Changes to CICS externals

This section describes the following changes to CICS externals introduced by CICS Web support enhancements. It consists of:

Changes to system definition

Two new system initialization parameters, TCPIP and DOCCODEPAGE, are introduced for the CICS Web interface.

TCPIP={YES|NO}
TCPIP=YES, which is the default, specifies that CICS TCPIP services, (HTTP and IIOP) are to be activated at CICS startup. If you specify TCPIP=NO, these services cannot be enabled.

DOCCDEPAGE={037|number}
Specifies the default host code page to be used by the CICS document domain if the HOSTCODEPAGE option is omitted from CICS API DOCUMENT commands.

DFHWBCD data set obsolete

The VSAM dataset DFHWBCD, used in earlier releases to store information relating to the CICS Web interface, is obsolete. It is replaced by the TCPIPSERVICE resource type (see“Changes to resource definition” for further information). Sample TCPIPSERVICE definitions are supplied in the CSD group, DFH$SOT.

Changes to resource definition

Two new resource definitions, DOCTEMPLATE and TCPIPSERVICE, have been introduced.

DOCTEMPLATE

Use this resource definition to define document templates to CICS. Document templates allow you to perform variable substitution on documents in a manner similar to that done by BMS for 3270 screens. The template can be retrieved by a user-replaceable module, or it can reside in any one of the following places:

- An MVS partitioned data set
- A CICS temporary storage queue
- A CICS extrapartition transient data destination
- A CICS load library as a load module

TCPIPSERVICE

Use this resource definition to define which TCP/IP services are to use CICS internal sockets support. The internal CICS services that can be defined are IIOP and the CICS Web interface.

The TCPIPSERVICE definition allows you to manage these internal CICS interfaces, with CICS listening on multiple ports, and with different flavors of CICS Web or IIOP support on different ports.

Specifying a value other than YES for the Socketclose keyword allows CICS to use HTTP Version 1.0 persistent connection support.

The XRSINDI global user exit is invoked when DOCTEMPLATE and TCPIPSERVICE resource types are installed or discarded.

DFHWEB TSMODEL definition

A new TSMODEL resource definition, DFHWEB, is provided in group DFHWEB to store inbound and outbound data from HTTP requests and responses. This defines a default TS queue prefix value of DFHWEB. If you use your own TCPIPSERVICE resource definition that specifies a different TS queue prefix, which is defined in your own TSMODEL definition, CICS uses that instead.

Changes to the application programming interface (API)

A new set of commands is added to the EXEC CICS API for the CICS Web interface, generally of the form EXEC CICS WEB *verb*, plus one EXEC CICS TCPIP command.

This set of CICS Web support API consists of the following commands:

- EXEC CICS TCPIP EXTRACT allows the application to obtain information about the TCPIP characteristics of the transaction.
- EXEC CICS WEB EXTRACT allows the application to extract information from the HTTP request sent by the client.
- EXEC CICS WEB ENDBROWSE HTTPHEADER terminates a browse started by the EXEC CICS WEB STARTBROWSE HTTPHEADER command.
- EXEC CICS WEB READ is used to extract HTTP header information.
- EXEC CICS WEB READNEXT retrieves the next HTTP header in the list of HTTP headers received on an inbound request.
- EXEC CICS WEB RECEIVE receives data from CICS Web support or business logic interface into an application-supplied buffer.
- EXEC CICS WEB SEND selects a document or application-supplied buffer for delivery by CICS Web support or the business logic interface.
- EXEC CICS WEB STARTBROWSE HTTPHEADER signals the start of a browse.
- EXEC CICS WEB WRITE HTTPHEADER allows the application to add HTTP header information to the HTTP response.

The following commands have been introduced for the creation and management of CICS documents:

- EXEC CICS DOCUMENT CREATE signals the start of the document creation process. The document being created can be an empty document or it can be based on an existing document, a template, or data contained in an application buffer.
- EXEC CICS DOCUMENT INSERT allows the application to insert document objects at insertion points (known as bookmarks) within the document.
- EXEC CICS DOCUMENT RETRIEVE allows the application to obtain a copy of the document in its own buffer, which it can then manipulate directly.
- EXEC CICS DOCUMENT SET allows the application to add symbols and their associated values to the symbol table.

Changes to the system programming interface (SPI)

There are new system programming interface commands in support of the CICS Web interface, as follows:

- INQUIRE TCPIP, to retrieve information about CICS internal sockets support.
- SET TCPIP, to open or close CICS internal sockets support.
- INQUIRE TCPIPSERVICE, to retrieve information about TCPIP ports on which CICS internal TCPIP support is currently listening on behalf of other CICS services.
- SET TCPIPSERVICE, to update the information relating to a service using CICS internal TCPIP support.
- CREATE TCPIPSERVICE, to define a TCP/IP service in the local CICS region.
- DISCARD TCPIPSERVICE, to remove a TCP/IP service from the local CICS region.
- CREATE DOCTEMPLATE, to define a document template in the local CICS region.
- DISCARD DOCTEMPLATE, to remove a document template from the local CICS region.
- INQUIRE DOCTEMPLATE, to retrieve information about a document template.
- INQUIRE WEB, to retrieve information about CICS Web support.

- SET WEB, to open or close CICS internal sockets support, change Web garbage collection settings, or change Web 3270 terminal timeout settings.

Changes to user-replaceable programs

There is a new user-replaceable module, the CICS Web error program (DFHWBEP), which is driven by CICS Web support in the event of a failure in the course of processing a Web request, and a change to the invocation of the analyzer program, DFHWBADX.

Web error program (DFHWBEP)

DFHWBEP allows you to modify the HTTP response issued by the CICS Web interface, or to put out an alternative message. DFHWBEP can overwrite CICS Web support HTTP response with its own HTTP response, which may be more meaningful to a user. The maximum length of the response is 32K. If the response needs to be longer, DFHWBEP must GETMAIN a new area and use that for the HTTP response. The resultant HTTP response is subject to the same DFHCCNV conversion as would have been the case for the target program.

CICS Web support analyzer program (DFHWBADX)

The context in which this program is run changes. Previously, the analyzer was only ever invoked by the one long-running server controller task. It can now be invoked concurrently by multiple transactions, which perform the function previously performed by the server Controller.

Changes to CICS-supplied transactions

The following options have been added to the CEMT transaction:

- INQUIRE TCPIP, to inquire on the status of CICS internal TCP/IP support.
- INQUIRE TCPIPSERVICE, to retrieve information about TCP/IP ports on which CICS internal TCP/IP support is currently listening on behalf of other CICS services.
- SET TCPIP, to open or close CICS internal sockets support.
- SET TCPIPSERVICE, to modify the status of a service using CICS internal TCP/IP support.
- INQUIRE WEB, to retrieve information about CICS Web support.
- SET WEB, to open or close CICS internal sockets support, change Web garbage collection settings, or change Web 3270 terminal timeout settings.

Problem determination

There are changes to the following diagnostic information to aid problem determination:

- CICS Web support dump formatting is changed for the new Web domain. New sockets domain dump formatting is added, with dump exit component code SO, and new document domain dump formatting, with dump exit component code DH.
- CICS Web support trace points are replaced with new Web domain trace points, and sockets domain and document domain trace points are added.
- There are new and changed messages and abend codes.

CICSPlex SM support

CICSPlex SM provides a new inquiry facility for CICS Web support in the form of two new views:

- **DOCTEMP**, for HTML document templates. Associated with this general view are a detail view, **DOCTEMPD**, and a summary view, **DOCTEMPS**.
- **TCPIPS** , for TCP/IP services using CICS internal sockets support. Associated with this general view are a detail view, **TCPIPSD**, and a summary view, **TCPIPSS**.

There are two new BAS resource definition views:

- **DOCDEF**, for defining HTML document templates to CICS.
- **TCPDEF**, for defining which TCP/IP services are to use CICS internal sockets support.

There are new resource tables:

- CRESDOCT
- CRESTCPS
- DOCDEF
- DOCINGRP
- DOCTEMP
- ERMCDOCT
- ERMCTCPS
- TCPDEF
- TCPINGRP
- TCPIPS

Part 6. Miscellaneous changes

This Part describes a number of miscellaneous changes to CICS TS. They are described in the following chapter:

Chapter 20. Miscellaneous changes

In addition to the changes described in earlier chapters of this book, CICS provides the following enhancements:

Removal of runtime support for RCTs

CICS TS introduced resource definition online support for DB2 resources in Release 2 while maintaining runtime support for RCTs generated using the DSNCRCT macro. The RCT runtime support is now removed. This leaves the CSD definitions and the EXEC CICS CREATE command as the only means of defining and installing DB2 resources to CICS, using resource types DB2CONN, DB2ENTRY, and DB2TRAN.

CICS continues to support the use of the DSNCRCT macro for migration purposes, allowing you to assemble an RCT ready for migration to your CSD.

To start the CICS DB2 attachment facility, install a DB2CONN resource definition to define the connection, and DB2ENTRY and DB2TRAN definitions for transactions. The minimum requirement when starting the CICS DB2 attachment facility is that the required DB2CONN definition is installed. Associated DB2ENTRY and DB2TRAN definitions can be installed later.

Effect of change on DSNC and INITPARM commands

The removal of runtime support for RCTs changes the syntax of the DSNC STRT command, and the syntax of the INITPARM command for DB2, as follows:

DSNC STRT {*name***}**
: The RCT suffix option is removed from this command, which now allows you to specify `DSNC STRT name`, where *name* is the name of the DB2 subsystem, or `DSNC STRT` only. The command requires a DB2CONN connection definition to be installed first.

INITPARM=(DFHD2INI='*name***')**
: The RCT suffix is removed from the syntax of the DFHD2INI parameter. The only purpose of the INITPARM parameter for DB2 is to provide the name of the DB2 subsystem when DB2ID is left blank in the DB2CONN resource definition.

Named counter sequence number facility

This section describes the facility for generating unique sequence numbers using the services of a named counter server. It covers the following topics:

- "Overview"
- "The named counter application programming interface" on page 199

Overview

CICS provides a facility for generating unique sequence numbers for use by applications in a Parallel Sysplex environment (for example, to allocate a unique number for orders or invoices). This facility is provided by a named counter server, which maintains each sequence of numbers as a named counter. Each time a sequence number is assigned, the corresponding named counter is incremented automatically so that the next request gets the next number in sequence.

The named counter server is modeled on the other coupling facility servers used by CICS, and has many features in common with the coupling facility data table server.

A named counter server provides a full set of functions to define and use named counters. Each named counter consists of:

- A 16-byte name
- A current value
- A minimum value
- A maximum value.

The values are internally stored as 8-byte (doubleword) unsigned binary numbers. The CICS API provides two sets of named counter commands, one that treats values as fullword signed binary numbers, and the other that treats values as doubleword unsigned binary numbers. The callable programming interface allows the numbers to be treated as any length from 1 to 8 bytes, typically 4 bytes.

Named counters are stored in a pool of named counters, where each pool is a small coupling facility list structure, with keys but no data. The pool name forms part of the list structure name. Each named counter is stored as a list structure entry keyed on the specified name, and each request for the next value requires only a single coupling facility access.

For information on how to create a list structure for use as a named counter pool, see the *CICS System Definition Guide*.

Selecting a named counter server

To reference a named counter, an application program can specify either the actual name of the pool in which the named counter is stored, or it can specify a pool selection parameter, which is mapped to the actual pool name by the POOL parameter specified in the options table, DFHNCOPT. This makes it easy to use a different pool (for example, to isolate test pools from production pools) without having to change the pool selection parameter in the application program. To vary the pool used by a CICS region, either load a different copy of the options table from STEPLIB, or use a common options table where the pool name selection is conditional on the job name and CICS APPLID, in addition to the pool name selection parameter. The options table also supports invocation of a user-specified program to select the appropriate pool given the pool selection parameter.

If the named counter facility can't find a match in the options table, it uses the default pool name DFHNC001. You can specify a different default pool name using the NCPLDFT system initialization parameter.

For information about creating a loadable options table, see the *CICS System Definition Guide*.

The named counter application programming interface

You access the named counter through the CICS API, or through a callable interface which can be used in batch jobs.

Although all named counter values are held internally as double word unsigned binary numbers, the CICS API provides both a fullword (COUNTER) and doubleword (DCOUNTER) set of commands, which you should not mix. These EXEC CICS commandsm allow you to perform the following operations on named counters:

DEFINE
Defines a new named counter, setting minimum and maximum values, and specifying the current number at which the counter is to start.

DELETE
Deletes a named counter from its named counter pool.

GET
Gets the current number from the named counter, provided the maximum number has not already been allocated. The GET command provides two comparison parameters that allow you to make the result of the command conditional upon the current number being within a specified range, or being greater than, or less than, one of the specified comparison values.

QUERY
Queries the named counter to obtain the current, minimum, and maximum values.

REWIND
Rewinds a named counter that is in the counter-at-limit condition back to its defined minimum value.

UPDATE
Updates the current value of a named counter to a new current value.

You can use the programming interfaces to a named counter server in all the supported programming languages—assembler, COBOL, PL/I, and C/C++.

Establishing contact with a named counter server

The first request by a CICS region that addresses a particular pool automatically establishes a connection to the server for that pool. This connection is associated with the current MVS TCB (which for CICS is the quasi-reentrant (QR) TCB) and normally lasts until the TCB terminates at end of job. Multiple TCBs in the same region can establish independent connections to the same named counter pool.

For more information about the named counter facility, see the *CICS Application Programming Guide* and the *CICS System Definition Guide*.

CDBM command file for storing IMS commands

The CICS-DBCTL operator transaction, CDBM, is enhanced to enable you to store IMS commands in a file and to issue commands from that file, or by screen input as before. A mechanism to update and maintain the file is provided.

Note: The CDBM transaction uses basic mapping support (BMS) and therefore this function is available only through those terminal devices that are supported by BMS.

Background

CICS regions generally have many IMS databases, and often the same operations are performed repeatedly, such as starting a group of databases in the morning, and stopping them at some other time for batch, and then repeating the cycle. Currently, you either have to type in the whole command each time, and this may be quite long, or if your databases are named in a generic way, you can abbreviate the command using the generic form (for example: `/STA DB DB*` to start all databases beginning with the letters DB). Many existing IMS systems, however, have databases with completely variable names, preventing the use of generic naming.

Overview

A new CICS system file, DFHDBFK, (the CDBM GROUP command file) is introduced, in which you can store IMS commands for subsequent use. A new function on the CICS-DBCTL Operator Transaction panel (PF2) brings up a new panel, the CICS/DBCTL COMMAND GROUP MAINTENANCE panel, which you can use to add and modify commands. You can also your own methods outside CICS to maintain the contents of the DFHDBFK file, which is required only if you intend to use the new facility.

DFHDBFK is a VSAM key-sequenced data set (KSDS), with a key length of 22 bytes, and a maximum record length of 1428 bytes. See Figure 27 for an example of the DEFINE CLUSTER statements you can use to define this data set.

```
DEFINE  CLUSTER ( NAME(CICSTS13.CICS.DFHDBFK ) -
           INDEXED                             -
           RECORDS(100 20)                     -
           RECORDSIZE(1428 1428)               -
           KEYS(22,0))                         -
         DATA (                                -
           NAME(CICSTS13.CICS.DFHDBFK)         -
           CONTROLINTERVALSIZE(2048))          -
         INDEX (                               -
           NAME(CICSTS13.CICS.DFHDBFK)         -
           CONTROLINTERVALSIZE(512))
```

Figure 27. IDCAMS definition statements for the DFHDBFK data set

For more information about creating this data set, see the *CICS System Definition Guide*.

The CICS-DBCTL Operator Transaction panel now accepts input of the form:

```
/GROUP SAMPLE STO
```

where SAMPLE STO is the key to a record in DFHDBFK. In this case it is a STOP command, probably for the same list of databases that can be started using:

```
/GROUP SAMPLE STA
```

Using this command facility allows greater numbers of databases to be actioned in a single command, because the CDBM Operator Transaction screen allows only 4 lines for command input, whereas DFHDBFK has 1406 bytes for IMS command parameters after the 22 byte record key.

For more information about the CDBM transaction, see the *CICS Supplied Transactions* or the *CICS IMS Database Control Guide*.

Enabling USER KEY CICSPlex SM API applications

CICSPlex SM API programs can now be run under transactions defined with TASKDATAKEY(USER). The CICSPlex SM program can be defined with an EXECKEY value of either USER or CICS. The associated transaction can be defined with a TASKDATAKEY value of either USER or CICS. The TASKDATAKEY reason of the ENVIRONERROR response is no longer returned.

Performance improvement for EXEC CICS LINK under LE

Migrating your CICS applications to Language Environment (LE) in CICS can significantly increase system pathlength if your application programs issue large numbers of EXEC CICS LINK commands. Much of the increase in pathlength is associated with obtaining and releasing MVS storage for run-unit work areas.

You can minimize the increase in pathlength by enabling CICS and LE to manage more effectively the storage requirements for LE-conforming application programs. You do this in CICS by specifying the RUWAPOOL system initialization parameter, indicating that you want CICS to create a run-unit work areas pool for each task. If you specifiy RUWAPOOL=YES, CICS creates a pool of storage at task initialization that can be reused by LE-conforming application programs.

For information about the RUWAPOOL system initialization parameter, see the *CICS System Definition Guide*. For information about the impact of LE-conforming CICS applications that issue EXEC CICS LINK commands, see the *CICS Performance Guide*.

MEMBER option added to INQUIRE TDQUEUE command

The option MEMBER(*data-area*) is added to EXEC CICS INQUIRE TDQUEUE command. If MEMBER is specified, CICS returns the 1– to 8–character member name of the partitioned data set used for the named extrapartition transient data queue. CICS returns blanks if the QSAM data set is not a partitioned data set. The option is also added to the CEMT INQUIRE TDQUEUE command.

Remove option added to CEDA and DFHCSDUP Commands

Extending the scope of the DELETE command, in an earlier release, to remove automatically a CSD group from group lists when the group itself was deleted caused problems for some users. To avoid such problems, a new option, REMOVE, is added to CEDA DELETE, CEDA MOVE, and to DFHCSDUP DELETE. Only

when REMOVE is specified does CICS remove a deleted group from all lists that contain the group, at the point when the group is itself deleted.

USERDEFINE added to DFHCSDUP Commands

The USERDEFINE command, previously only offered from CEDA, is made available from DFHCSDUP.

Euro support

CICS supports the new European currency unit, the euro.

Data conversion

When data is transferred between CICS systems—by, for example, transaction routing or function shipping—some or all of the data may need to be converted from one form of encoding to another. For example, when character data is transferred from CICS for OS/2® to CICS System/390, it has to be converted from the ASCII format used by CICS for OS/2 to the EBCDIC format used by System/390. The same consideration applies whenever character data is transferred between any ASCII system and CICS System/390 (typically, non-System/390 systems hold character data in ASCII format).

CICS TS, in common with other CICS System/390 products, supports new client (ASCII) and server (EBCDIC) code pages that include the euro symbol. All the client and server code pages supported by CICS System/390 are listed in the *CICS Family: Communicating from CICS on System/390*.

BMS support for the Euro

Extensions to the COBOL standard now allow multiple CURRENCY SIGN clauses, with a new PICTURE SYMBOL phrase that can define a symbol of one or more characters. For example:

```
SPECIAL NAMES.
CURRENCY SIGN IS '$' WITH PICTURE SYMBOL '$'.
CURRENCY SIGN IS '£' WITH PICTURE SYMBOL '£'.
CURRENCY SIGN IS 'EUR' WITH PICTURE SYMBOL '#'.


WORKING STORAGE SECTION.
01 USPRICE PIC $.99.99
01 UKPRICE PIC £.99.99
01 ECPRICE PIC #.99.99


PROCEDURE DIVISION
MOVE 12.34 to UKPRICE.       value is £12.34
MOVE 12.34 to USPRICE.       value is $12.34
MOVE 12.34 to ECPRICE.       value is EUR12.34
```

The DFHMDF macro now supports PICIN and PICOUT picture specifications with currency symbols other than $, which previously defaulted to the national currency symbol. LENGTH must be specified when PICOUT specifies a COBOL picture

containing a currency symbol that will be replaced by a currency sign of length greater than 1. See *CICS Application Programming Reference* for details of the DFHMDF macro.

Support for connection quiesce protocol

CICS supports the connection quiesce protocol (CQP), which CICS invokes automatically when APPC sessions are being closed. CQP is an exchange of information that takes place between connected systems to determine whether there is any outstanding resynchronization to be done. If the exchange shows that there is no outstanding work on the connection, both systems can:

- Purge its record of the partner's logname
- Delete any affinity that the connection has.

Successful completion of connection quiesce assures you that a cold start can take place safely.

There are changes to the CEMT INQUIRE CONNECTION and EXEC CICS INQUIRE CONNECTION commands to return information about the status of CQP on connections. The new option is CQP:

CQP(*cvda***)**
returns a CVDA indicating the status of the connection quiesce protocal on the connection. The CVDA values are:

COMPLETE
The protocol completed successfully when the connection was released. This reverts to NOTATTEMPTED if the connection is reacquired.

FAILED
The protocol failed. This can occur for several reasons, such as a session failure during execution of the protocol, or because the partner receiving the CQP flow has outstanding work.

NOTATTEMPTED
The connection supports the protocol, but it has not yet been invoked because the connection status ACQUIRED.

NOTSUPPORTED
The connection does not support the protocol. This could be, for example, because the partner is a back-level CICS region that does not have CQP support.

There are new messages, DFHZC4950 and DFHZC4951, to indicate a failure of the CICS CQP transaction, and two new abend codes, ACQA and ACQB.

CICSPlex SM BAS support for FEPI resources

Changes have been made to the way in which FEPI resources are installed using CICSPlex SM. The installation function has been removed from the operations FEPI views. Instead, there are new BAS views for defining and installing FEPI resources:

- FENODDEF, which describes the physical and operational characteristics of FEPI nodes.

- FEPOODEF, which describes the physical and operational characteristics of FEPI pools.
- FEPRODEF, which describes the physical and operational characteristics of FEPI property sets.
- FETRGDEF, which describes the physical and operational characteristics of FEPI targets.

New resource tables are introduced:

- FENODDEF
- FEPOODEF
- FEPRODEF
- FETRGDEF
- FNOINGRP
- FPOINGRP
- FPRINGRP
- FTRINGRP

Part 7. Requirements

This Part describes the hardware and software requirements for CICS TS in the following chapter:

Chapter 21. Prerequisite hardware and software for CICS Transaction Server for OS/390

This chapter gives some information about related IBM program products that you need either to use CICS Transaction Server for OS/390, or exploit the new and changed function. It covers the following topics:

Hardware prerequisites

To run CICS Transaction Server you need a System/390 processor that supports OS/390 Version 2 Release 5 or later, and which has enough processor storage to meet the combined requirements of the host operating system, CICS TS, the access methods, and the application programs. Suitable processors include:

- All models of the S/390® Parallel Enterprise Servers or S/390 Parallel Transaction Servers (IBM 9672)
- All models of the Multiprise® 2000
- All models of the ES/9000® Processor Unit 9021, 9121, or 9221
- IBM ES/3090™–9000T processors (models 15T, 17T, 18T, 25T, 28T) that support IBM Enterprise Systems Architecture/370™(ESA/370) and which must have optional ESA/390™ facilities
- PC Server System/390 or RS/6000® and System/390 Server-on-Board.

Parallel Sysplex support

Each of the the data-sharing facilities supported by CICS, and the MVS system logger's log stream merging facility, all require a Parallel Sysplex environment. For this you need:

- One or more coupling facilities with their associated coupling links installed (see "Coupling facilities" on page 208.
- An IBM sysplex timer
- Sufficient DASD paths to support the number of CPCs in the sysplex.

You can use CICS support for data sharing to access the following forms of data:

- IMS databases
- DB2 databases
- VSAM data sets

- CICS temporary storage
- Coupling facility data tables
- Named number counters.

Coupling facilities

A coupling facility can be one of the following:

- A standalone IBM 9674.
- A PR/SM™ logical partition (LPAR) running the coupling facility control code. The processors that can enable the coupling facility function in an LPAR are:
  - ES/9000 711-based models
  - ES/9000 511-based models
  - S/390 Parallel Enterprise Servers (9672).

  The 9121 511-based models require the integrated coupling migration facility (ICMF) to provide coupling facility functions.
- A PR/SM logical partition (LPAR) with ICMF for both the 9021 711-based and 9121 511-based processors, or for the S/390 Parallel Enterprise Servers (9672). This latter configuration eliminates the need for coupling links.

In general, a standalone coupling facility is recommended for a production environment to eliminate a single point of failure, and two coupling facilities are recommended for high availability.

Sysplex timer

A Parallel Sysplex requires an IBM sysplex timer to provide a common external time source.

DASD paths

A Parallel Sysplex requires either DASD controllers with enough paths to dedicate one to each CPC in the sysplex, or an ESCON® director to provide the paths.

Operating system

CICS TS requires OS/390 Version 2 Release 5 (5647–A01) or later. Note that OS/390 includes, as base elements, many of the products required by CICS TS, therefore the sections that follow cover only products that are not supplied as part of OS/390. The following service is also required before you install CICS TS under OS/390 Release 5:

DFSMS/MVS Binder PTF for APAR OW36582
IEBCOPY PTFs UW49740 and UW54887

IBM database products

CICS supports IMS/ESA® Database Manager and IBM DATABASE 2™ (DB2) as described in this section.

IMS/ESA Database Manager

CICS application programs can access IMS databases, through the DBCTL interface only, using IMS/ESA Database Manager Version 5 Release 1 (5695–176) or later.

IBM DATABASE 2 (DB2)

CICS application programs can access DB2 databases using DB2 Version 4 (5695–DB2) or later.

IBM telecommunications access methods

VTAM and TCP/IP are both included as exclusive elements of OS/390 Release 5.

TCP/IP OS/390 Release 5 includes TCP/IP CICS Sockets, which enables network users access to CICS regions. CICS programs can use the TCP/IP “sockets” application programming interface (API) to communicate with TCP/IP devices. TCP/IP also enables access to CICS through:

- The CICS ONC RPC support, which enables CICS as a server for Remote Procedure Call (RPC) requests using the Open Network Communication (ONC) standard protocol
- DCE/MVS, which enables CICS as a server for Remote Procedure Call requests using the Distributed Computing Environment (DCE) standard protocol

You can access CICS Transaction Server for OS/390 using ACF/TCAM (DCB) Version 2.4 (5735–RC3) plus PTFs, or ACF/TCAM (DCB) Version 3.1 (5665–314) plus PTFs.

IBM external security manager (RACF)

RACF Security Server available with OS/390 Release 5 provides for all CICS TS security needs except for security of temporary storage queues that use long TS queue names. If you have application programs that use secure TS queues with 16-character queue names, you need RACF OS/390 Release 6 with a PTF to support 25-character profile names.

CICS VSAM Recovery

If you use CICS VSAM Recovery (CICSVR) as your VSAM forward recovery utility, CICSVR Version 2.3 (5695–010) is required.

Tivoli Performance Reporter for OS/390

CICS TS no longer supports earlier versions of the Performance Reporter products (IBM SystemView® Enterprise Performance Data Manager/MVS (EPDM) or IBM SystemView Performance Reporter for MVS) and in their place supports Tivoli Performance Reporter for OS/390 (5695–101). For CICS TS Release 3, you need Tivoli Performance Reporter Version 1.3 or 1.4, both of which require service PTFs to enable them to work with CICS TS.

Netview® for MVS/ESA

For a resource object data manager (RODM) repository that CICSPlex SM exploits through NetView MultiSystem Manager Version 2 Release 2 (5655–126), CICS TS supports NetView for MVS/ESA Version 3 Release 1 (5655–007).

Programming languages

CICS Transaction Server for OS/390 supports the following programming languages and environments:

- High Level Assembler/MVS (5696–234)
- IBM PL/I for MVS & VM (5688–235)
- OS PL/I Optimizing Compiler Version 2 Release 1 (5668–910)
- OS PL/I Optimizing Compiler Version 1 Release 5 (5724–PL1)
- IBM COBOL for MVS & VM (5688–197)
- VS COBOL II (5668–958 and 5668–023) (requires PTF for APAR 43097)
- C/370 (5688–040 and 5688–187)
- IBM C/C++ for MVS (5655–121)
- CSP Version 3 or later
- SAA AD/Cycle COBOL/370 (5688–197)
- SAA AD/Cycle PL/I (5688–235)
- SAA AD/Cycle C/370 (5688–216)
- VisualAge for Java, Enterprize Edition for OS/390

CICS components in object-code-only (OCO) form

Some of the functional areas of CICS are provided, either completely or partially, in object-code-only form (OCO), without licensed source materials. These areas include:

- Authorized cross-memory (AXM) server environment
- Autoinstall terminal model manager, AITM
- Business application manager domain
- Catalog domains
- Common Programming Interface functions
- Coupling facility data tables
- Coupling facility data table server
- Directory manager domain
- Dispatcher domain
- Document domain
- Dump domain
- Enqueue domain
- Event manager domain
- EXEC CICS system programming command support
- File control RLS support
- Kernel domain
- Loader domain
- Lock manager domain
- Log manager
- Message domain
- Monitoring domain
- Named counter server
- Offline statistics utility
- Offline system dump formatting routines
- Parameter manager domain

- Partner resource manager
- Program manager domain
- RDO for VSAM files and LSR pools
- Recovery manager
- Resource recovery services (RRS) interface
- SAA communications and resource recovery interfaces
- Scheduler services domain
- Security domain
- Shared data tables
- Sockets domain
- Statistics domain
- Storage manager domain
- Temporary storage data sharing server
- Temporary storage domain
- Timer domain
- Trace domain
- Transaction manager domain
- User domain

Part 8. Appendixes

Appendix. Details of changed monitoring records

This appendix contains full details of the CICS monitoring performance data affected by the changes described in “Chapter 6. Monitoring, statistics, and enterprise management changes” on page 65.

Selectivity of performance class data fields

You can control the performance data CICS selects by means of DFHMCT parameters. These parameters (EXCLUDE= and INCLUDE=) apply to the TYPE=RECORD parameter for performance class monitoring. Each parameter can specify one or more fields, either specifically by field ID, or generically by group name. The EXCLUDE parameter is honored before any INCLUDE parameter. A revised list of field IDs and group names that are eligible for exclusion or inclusion follows.

Note: Only the fields listed in Table 10 can be selected in this way.

Table 10. Data groups and fields that can be excluded/included

Group Name	Field Id	Description
DFHCBTS	200	CICS BTS process name
DFHCBTS	201	CICS BTS process type
DFHCBTS	202	CICS BTS process id
DFHCBTS	203	CICS BTS activity id
DFHCBTS	204	CICS BTS activity name
DFHCBTS	205	CICS BTS run process/activity synchronous count
DFHCBTS	206	CICS BTS run process/activity asynchronous count
DFHCBTS	207	CICS BTS link process/activity count
DFHCBTS	208	CICS BTS define process count
DFHCBTS	209	CICS BTS define activity count
DFHCBTS	210	CICS BTS reset process/activity count
DFHCBTS	211	CICS BTS suspend process/activity count
DFHCBTS	212	CICS BTS resume process/activity count
DFHCBTS	213	CICS BTS delete activity, or cancel process/activity, request count
DFHCBTS	214	CICS BTS acquire process/activity request count
DFHCBTS	215	CICS BTS total process/activity request count
DFHCBTS	216	CICS BTS delete/get/put process container count
DFHCBTS	217	CICS BTS delete/get/put activity container count
DFHCBTS	218	CICS BTS total process/activity container request count
DFHCBTS	219	CICS BTS retrieve reattach request count
DFHCBTS	220	CICS BTS define input event request count
DFHCBTS	221	CICS BTS timer associated event request count
DFHCBTS	222	CICS BTS total event related request count
DFHCICS	25	CICS OO foundation class request count
DFHCICS	103	Transaction exception wait time
DFHCICS	112	Performance record type
DFHCICS	130	Transaction routing sysid
DFHCICS	131	Performance record count
DFHCICS	167	MVS Workload Manager Service Class name
DFHCICS	168	MVS Workload Manager Report Class name

Table 10. Data groups and fields that can be excluded/included (continued)

Group Name	Field Id	Description
DFHDATA	179	IMS (DBCTL) request count
DFHDATA	180	DB2 request count
DFHDATA	186	IMS (DBCTL) wait time
DFHDATA	187	DB2 Readyq wait time
DFHDATA	188	DB2 Connection wait time
DFHDATA	189	DB2 wait time
DFHDOCH	226	Document handler Create count
DFHDOCH	227	Document handler Insert count
DFHDOCH	228	Document handler Set count
DFHDOCH	229	Document handler Retrieve count
DFHDOCH	230	Document handler Total count
DFHDOCH	240	Document handler total created document length
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
DFHFILE	174	RLS FC I/O wait time
DFHFILE	175	RLS File request CPU (SRB) time
DFHFILE	176	CFDT I/O wait time
DFHJOUR	10	JC I/O wait time
DFHJOUR	58	Journal write count
DFHJOUR	172	CICS Logger 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

Table 10. Data groups and fields that can be excluded/included (continued)

Group Name	Field Id	Description
DFHPROG	56	Program XCTL count
DFHPROG	57	Program LOAD count
DFHPROG	71	Initial program name
DFHPROG	72	Program LINK URM count
DFHPROG	73	Program DPL count
DFHPROG	113	Original abend code
DFHPROG	114	Current abend code
DFHPROG	115	Program load time
DFHSOCK	241	Socket (SO) I/O wait time
DFHSOCK	242	Bytes encrypted for secure socket
DFHSOCK	243	Bytes decrypted for secure socket
DFHSOCK	244	Client IP Address
DFHSTOR	33	User-storage high-water-mark (UDSA)
DFHSTOR	54	User-storage getmain-count (UDSA)
DFHSTOR	87	Program-storage high-water-mark - total
DFHSTOR	95	User-storage-occupancy (bytes-ms) (UDSA)
DFHSTOR	105	User-storage getmain-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 getmain-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 getmain-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	144	Shared storage GETMAIN-count—below 16MB (CDSA and SDSA)
DFHSTOR	144	Shared storage bytes GETMAINed—below 16MB (CDSA and SDSA)
DFHSTOR	145	Shared storage bytes FREEMAINed—below 16MB (CDSA and SDSA)
DFHSTOR	146	Shared storage GETMAIN-count—above 16MB (ECDSA and ESDSA)
DFHSTOR	147	Shared storage bytes GETMAINed—above 16MB (ECDSA and ESDSA)
DFHSTOR	148	Shared storage bytes FREEMAINed—above 16MB (ECDSA and ESDSA)
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	Syncpoint count
DFHSYNC	173	Syncpoint elapsed time
DFHSYNC	177	CFDT server syncpoint wait time
DFHSYNC	196	Syncpoint delay time

Table 10. Data groups and fields that can be excluded/included (continued)

Group Name	Field Id	Description
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	66	IC total count
DFHTASK	64	Error flag field
DFHTASK	82	Transaction group ID
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	123	Task global ENQ delay time
DFHTASK	124	3270 Bridge transaction ID
DFHTASK	125	First dispatch delay time
DFHTASK	126	First dispatch delay time due to TRANCLASS
DFHTASK	127	First dispatch delay due to MXT
DFHTASK	128	Lock manager delay time
DFHTASK	129	Task local ENQ delay time
DFHTASK	132	Recovery manager unit-of-work ID
DFHTASK	163	Transaction facility name
DFHTASK	164	Transaction flags
DFHTASK	170	Resource manager interface—elapsed time
DFHTASK	171	Resource manager interface—suspend time
DFHTASK	181	EXEC CICS WAIT EXTERNAL wait time
DFHTASK	182	EXEC CICS WAITCICS and WAIT EVENT wait time
DFHTASK	183	Interval control delay time
DFHTASK	184	“Dispatchable Wait” wait time
DFHTASK	190	RRMS/MVS unit-of-recovery id (URID)
DFHTASK	191	RRMS/MVS wait time
DFHTASK	195	CICS BTS run process/activity synchronous wait time
DFHTASK	248	CICS TCB change modes
DFHTASK	249	User-task QR TCB wait-for-dispatch time
DFHTASK	250	CICS MAXOPENTCBS delay time
DFHTASK	251	CICS TCB attach count
DFHTASK	253	CICS JVM elapsed time
DFHTASK	254	CICS JVM suspend time
DFHTASK	255	User-task QR TCB dispatch time
DFHTASK	256	User-task QR TCB CPU time
DFHTASK	257	User-task MS TCB dispatch time
DFHTASK	258	User-task MS TCB CPU time
DFHTASK	259	User-task L8 TCB CPU time
DFHTASK	260	User-task J8 TCB CPU time
DFHTASK	261	User-task S8 TCB CPU 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
DFHTEMP	178	Shared TS I/O wait time
DFHTERM	9	TC I/O wait time

Table 10. Data groups and fields that can be excluded/included (continued)

Group Name	Field Id	Description
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
DFHTERM	165	Terminal information
DFHTERM	169	Terminal session connection name
DFHWEBB	231	WEB Receive request count
DFHWEBB	232	WEB Characters received
DFHWEBB	233	WEB Send request count
DFHWEBB	234	WEB Characters sent
DFHWEBB	235	WEB Total request count
DFHWEBB	236	WEB Repository read request count
DFHWEBB	237	WEB Repository write request count

Interpreting CICS monitoring

The exception class data and performance class data that has been added or changed is described in this section. Each of the data fields is presented as a field description, followed by an explanation of the contents. The field description has the format shown in Figure 28 on page 220, which is taken from the performance data group DFHTASK.

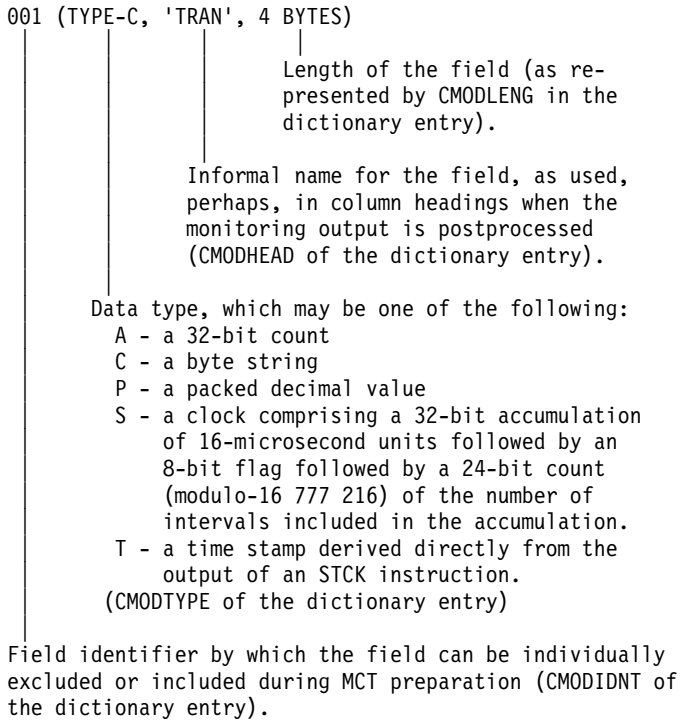


Figure 28. Format of the descriptions of the data fields

Note: References in Figure 28 to the associated dictionary entries apply only to the performance class data descriptions. Exception class data is not defined in the dictionary record.

Clocks and time stamps

In the descriptions that follow, the term **clock** is distinguished from the term **time stamp**.

A **clock** is a 32-bit value, expressed in units of 16 microseconds, accumulated during one or more measurement periods. The 32-bit value is followed by eight reserved bits, which are in turn followed by a 24-bit value indicating the number of such periods.

Neither the 32-bit timer component of a clock nor its 24-bit period count is protected against wraparound. The timer capacity is about 18 hours, and the period count runs modulo 16 777 216.

The eight reserved bits have the following significance:

Bits 0, 1, 2, and 3
Used for online control of the clock when it is running, and should always be zeros on output.

Bits 4 and 7
Not used.

Bits 5 and 6
Used to indicate, when set to 1, that the clock has suffered at least one out-of-phase start (bit 5) or stop (bit 6).

A **time stamp** is an 8-byte copy of the output of an STCK instruction.

Note: All times produced in the offline reports are in Greenwich Mean Time (GMT), not local time (assuming your TOD clock is set to GMT). Times produced by online reporting can be expressed in either GMT or local time.

Performance class data

The performance class data is described below in order of group name. The group name is always in field CMODNAME of the dictionary entry.

A user task can be represented by one or more performance class monitoring records, depending on whether the MCT event monitoring option DELIVER or the system initialization parameters MNCONV=YES or MNSYNC=YES have been selected. In the descriptions that follow, the term “user task” means that part or whole of a transaction that is represented by a performance class record, unless the description states otherwise.

Transaction timing fields

The CMF performance class record provides detailed timing information for each transaction as it is processed by CICS. A transaction can be represented by one or more performance class records, depending on the monitoring options selected. The key transaction timing data fields are described as follows:

- The “Transaction Start time” and “Transaction Stop time” represent the start and end of a transaction measurement interval. This is normally the period between transaction attach and detach but the performance class record could represent a part of a transaction depending on the monitoring options selected. The “Transaction Response Time” can be calculated by subtracting the transaction start time from the stop time.
- The “Transaction Dispatch time” is the time the transaction was dispatched.
- The “Transaction CPU time” is the portion of Dispatch time when the task is using processor cycles.
- The “Transaction Suspend time” is the total time the task was suspended and includes:
  - All task suspend (wait) time, which includes:
    - The wait time for redispatch (dispatch wait)
    - The wait time for first dispatch (first dispatch delay)
    - The total I/O wait and other wait times
- The First Dispatch Delay is then further broken down into:
  - First Dispatch Delay due to TRANCLASS limits
  - First Dispatch Delay due to MXT limits

The CMF performance class record also provides a more detailed breakdown of the transaction suspend (wait) time into separate data fields. These include:

- Terminal I/O wait time
- File I/O wait time
- RLS file I/O wait time
- Coupling facility data table (CFDT) time
- Journal I/O wait time
- Temporary storage I/O wait time
- Shared temporary Storage I/O wait time
- Interregion (MRO) I/O wait time
- RRMS/MVS wait time
- Socket I/O wait time

- Transient data I/O wait time
- LU 6.1 I/O wait time
- LU 6.2 I/O wait time
- FEPI suspend time
- Task local ENQ delay time
- Task global ENQ delay time
- RMI suspend time
- Lock manager delay time
- EXEC CICS WAIT EXTERNAL wait time
- EXEC CICS WAITCICS and WAIT EVENT wait time
- Interval control delay time
- “Dispatchable wait” wait time
- IMS (DBCTL) wait time
- DB2 ready queue wait time
- DB2 connection wait time
- DB2 wait time
- CFDT server syncpoint wait time
- Syncpoint delay time
- CICS BTS run process/activity synchronous wait time
- CICS MAXOPENTCBS delay time
- CICS JVM suspend time.

Response time

You can calculate the internal CICS response time by subtracting performance data field 005 (start time) from performance data field 006 (stop time).

Figure 29 shows the relationship of dispatch time, suspend time, and CPU time with the response time.

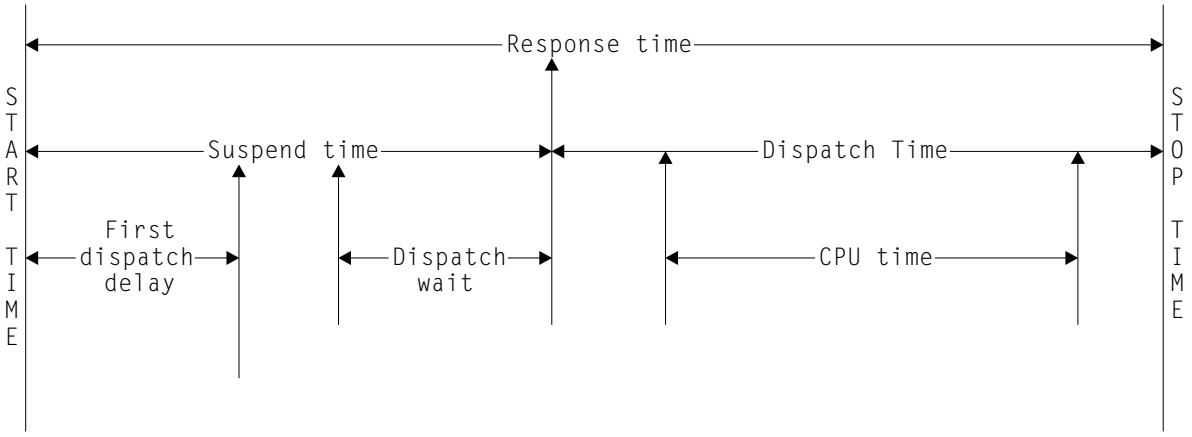


Figure 29. Response time relationships

Wait (suspend) times

Table 11 on page 223 lists the CICS performance class wait (suspend) fields.

The performance data fields 009, 010, 011, 63, 100, 101, 123, 128, 129, 133, 134, 156, 171, 174, 176, 177, 178, 181, 182, 183, 184, 186, 187, 188, 189, 191, 195, 196, 241, 250 and 254 all record the elapsed time spent waiting for a particular type of I/O operation. For example, field 009 records the elapsed time waiting for terminal I/O. The elapsed time includes not only that time during which the I/O

operation is actually taking place, but also the time during which the access method is completing the outstanding event control block, and the time subsequent to that until the waiting CICS transaction is redispatched.

Table 11. Performance class wait (suspend) fields

Field-Id	Group Name	Description
009	DFHTERM	TC I/O wait time
010	DFHJOUR	JC I/O wait time
011	DFHTEMP	TS I/O wait time
063	DFHFILE	FC I/O wait time
100	DFHTERM	IR I/O wait time
101	DFHDEST	TD I/O wait time
123	DFHTASK	Task global ENQ delay time
128	DFHTASK	Lock Manager delay time
129	DFHTASK	Task local ENQ delay time
133	DFHTERM	TC I/O wait time—LU6.1
134	DFHTERM	TC I/O wait time—LU6.2
156	DFHFEPI	FEPI Suspend time
171	DFHTASK	Resource manager interface—suspend time
174	DFHFILE	RLS FC I/O wait time
176	DFHFILE	CFDT I/O wait time
177	DFHSYNC	CFDT server syncpoint time
178	DFHTEMP	Shared TS I/O wait time
181	DFHTASK	EXEC CICS WAIT EXTERNAL wait time
182	DFHTASK	EXEC CICS WAITCICS and WAIT EVENT wait time
183	DFHTASK	Interval Control delay time
184	DFHTASK	“Dispatchable Wait” wait time
186	DFHDATA	IMS (DBCTL) wait time
187	DFHDATA	DB2 ready queue wait time
188	DFHDATA	DB2 connection time
189	DFHDATA	DB2 wait time
191	DFHTASK	RRMS/MVS wait time
195	DFHTASK	CICS BTS run process/activity synchronous wait time
196	DFHSYNC	Syncpoint delay time
241	DFHSOCK	Socket I/O wait time
250	DFHTASK	CICS MAXOPENTCBS delay time
254	DFHTASK	CICS JVM suspend time

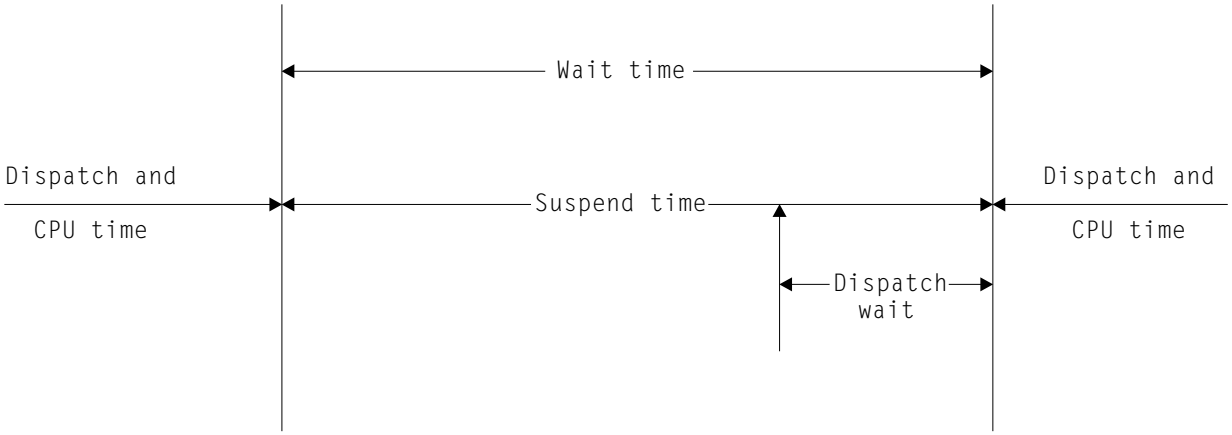


Figure 30. Wait (suspend) time relationships

As a result of the improvements to the CMF suspend time and wait time measurements in CICS, it is now possible to perform various calculations on the suspend time accurately. For example, the "Total I/O wait time" can be calculated as shown on page 224.

```
Total I/O wait time = ( Terminal control I/O wait        +
                        Temporary storage I/O wait    +
                        Shared temporary storage I/O wait +
                        Transient data I/O wait       +
                        Journal (MVS logger) I/O wait +
                        File control I/O wait         +
                        RLS file I/O wait             +
                        CF data table I/O wait        +
                        Socket I/O wait               +
                        Interregion (MRO) I/O wait    +
                        LU6.1 TC I/O wait             +
                        LU6.2 TC I/O wait             +
                        FEPI I/O wait )
```

The “Other wait time” (that is, uncaptured wait (suspend) time) can be calculated as:

```
Total other wait = ( First dispatch delay         +
                     Local ENQ delay              +
                     Global ENQ delay             +
                     Interval control delay       +
                     Lock manager delay           +
                     Wait external wait           +
                     Wait CICS/event wait         +
                     Dispatchable wait            +
                     RMI suspend                  +
                     CICS BTS run synchronous wait +
                     CFDT server synchronous wait  +
                     Syncpoint delay time         +
                     CICS MAXOPENTCBS delay        +
                     RRMS/MVS wait )
```

Uncaptured wait time can be calculated as:

```
Uncaptured wait time = (Suspend - (Total I/O wait time +
                                   Total other wait time) )
```

Note: The “First Dispatch Delay” performance class data field includes the MXT and TRANCLASS “First Dispatch Delay” fields.

In addition to the transaction “Suspend (wait) Time” breakdown, the CMF performance class data provides several other important transaction timing measurements:

- The program load time is the program fetch time (dispatch time) for programs invoked by the transaction.
- The exception wait time is the accumulated time from the exception conditions as measured by the CMF exception class records.
- The RMI elapsed time is the elapsed time the transaction spent in all resource managers invoked by the transaction using the resource manager interface (RMI).
- The JVM elapsed time is the elapsed time the transaction spent in the JVM.
- The syncpoint elapsed time is the elapsed time the transaction spent processing a syncpoint.

Performance data in group DFHCBTS

Group DFHCBTS contains the following performance data:

200 (TYPE-C, ‘PRCSNAME’, 36 BYTES)
The name of the CICS business transaction service (BTS) process of which the user task formed part.

201 (TYPE-C, ‘PRCSTYPE’, 8 BYTES)
The process-type of the CICS BTS process of which the user task formed part.

202 (TYPE-C, ‘PRCSID’, 52 BYTES)
The CICS-assigned identifier of the CICS BTS root activity that the user task implemented.

203 (TYPE-C, ‘ACTVTYID’, 52 BYTES)
The CICS-assigned identifier of the CICS BTS activity that the user task implemented.

204 (TYPE-C, ‘ACTVTYNM’, 16 BYTES)
The name of the CICS BTS activity that the user task implemented.

205 (TYPE-A, ‘BARSYNCT’, 4 BYTES)
The number of CICS BTS run process, or run activity, requests that the user task made in order to execute a child process or activity synchronously.

206 (TYPE-A, ‘BARASYCT’, 4 BYTES)
The number of CICS BTS run process, or run activity, requests that the user task made in order to execute a child process or activity asynchronously.

207 (TYPE-A, ‘BALKPACT’, 4 BYTES)
The number of CICS BTS link process, or link activity, requests that the user task issued.

208 (TYPE-A, ‘BADPROCT’, 4 BYTES)
The number of CICS BTS define process requests issued by the user task.

209 (TYPE-A, ‘BADACTCT’, 4 BYTES)
The number of CICS BTS define activity requests issued by the user task.

210 (TYPE-A, ‘BARSPACT’, 4 BYTES)
The number of CICS BTS reset process and reset activity requests issued by the user task.

211 (TYPE-A, ‘BASUPACT’, 4 BYTES)
The number of CICS BTS suspend process, or suspend activity, requests issued by the user task.

212 (TYPE-A, 'BARMPACT', 4 BYTES)
The number of CICS BTS resume process, or resume activity, requests issued by the user task.

213 (TYPE-A, 'BADCPACT', 4 BYTES)
The number of CICS BTS delete activity, cancel process, or cancel activity, requests issued by the user task.

214 (TYPE-A, 'BAACQPCT', 4 BYTES)
The number of CICS BTS acquire process, or acquire activity, requests issued by the user task.

215 (TYPE-A, 'BATOTPCT', 4 BYTES)
Total number of CICS BTS process and activity requests issued by the user task.

216 (TYPE-A, 'BAPRDCCT', 4 BYTES)
The number of CICS BTS delete, get, or put, container requests for process data containers issued by the user task.

217 (TYPE-A, 'BAACDCCT', 4 BYTES)
The number of CICS BTS delete, get, or put, container requests for activity data containers issued by the user task.

218 (TYPE-A, 'BATOTCCT', 4 BYTES)
Total number of CICS BTS process container and activity container requests issued by the user task.

219 (TYPE-A, 'BARATECT', 4 BYTES)
The number of CICS BTS retrieve-reattach event requests issued by the user task.

220 (TYPE-A, 'BADFIECT', 4 BYTES)
The number of CICS BTS define-input event requests issued by the user task.

221 (TYPE-A, 'BATIAECT', 4 BYTES)
The number of CICS BTS timer-associated event requests issued by the user task.

222 (TYPE-A, 'BATOTECT', 4 BYTES)
The total number of CICS BTS event requests issued by the user task.

Performance data in group DFHCICS

Group DFHCICS contains the following performance data:

005 (TYPE-T, 'START', 8 BYTES)
Start time of measurement interval. This is one of the following:

- The time at which the user task was attached
- The time at which data recording was most recently reset in support of the MCT user event monitoring point DELIVER option
- The monitoring options MNCONV, MNSYNC, or FREQUENCY.

For more information, see "Clocks and time stamps" on page 220.

Note: Response Time = STOP − START. For more information, see "Response time" on page 222.

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 MNCONV, MNSYNC or FREQUENCY. For more information, see “Clocks and time stamps” on page 220.

Note: Response Time = STOP − START. For more information, see “Response time” on page 222.

025 (TYPE-A, ‘CFCAPICT’, 4 BYTES)
The total number of CICS OO foundation class requests and CICS Java (JCICS) requests issued by the user task. CICS does not distinguish between OO class and JCICS requests.

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. For more information, see “Clocks and time stamps” on page 220.

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)
The name (sysid) of the remote system to which this transaction was routed either statically or dynamically.

This field also includes the connection name (sysid) of the remote system to which this transaction was routed when using the CRTE routing transaction. The field will be null for those CRTE transactions which establish or cancel the transaction routing session.

Note: If the transaction was not routed or was routed locally, this field is set to null. Also see the program name (field 71).

131 (TYPE-A, ‘PERRECNT’, 4 bytes)
The number of performance class records written by the CICS Transaction Server for OS/390 Monitoring Facility (CMF) for the user task.

167 (TYPE-C, ‘SRVCLASS’, 8 bytes)
The MVS Workload Manager (WLM) service class for this transaction. This field is null if the transaction was WLM-classified in another CICS region.

168 (TYPE-C, ‘RPTCLASS’, 8 bytes)
The MVS Workload Manager (WLM) report class for this transaction. This field is null if the transaction was WLM-classified in another CICS region.

Performance data in group DFHDATA

Group DFHDATA contains the following performance data:

179 (TYPE-A, 'IMSREQCT', 4 BYTES)
Number of IMS (DBCTL) requests issued by the user task.

180 (TYPE-A, 'DB2REQCT', 4 BYTES)
Number of DB2 (EXEC SQL and IFI) requests issued by the user task.

186 (TYPE-S, 'IMSWAIT', 8 BYTES)
The elapsed time in which the user task waited for DBCTL to service the IMS requests issued by the user task.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

187 (TYPE-S, 'DB2RDYQW', 8 BYTES)
The elapsed time in which the user task waited for a DB2 thread to become available.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

188 (TYPE-S, 'DB2CONWT', 8 BYTES)
The elapsed time in which the user task waited for a CICS DB2 subtask to become available.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

189 (TYPE-S, 'DB2WAIT', 8 BYTES)
The elapsed time in which the user task waited for DB2 to service the DB2 EXEC SQL and IFI requests issued by the user task.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

Performance data in group DFHDOCH

Group DFHDOCH contains the following performance data:

226 (TYPE-A, 'DHCRECT', 4 BYTES)
Number of document handler CREATE requests issued by the user task.

227 (TYPE-A, 'DHINSCT', 4 BYTES)
Number of document handler INSERT requests issued by the user task.

228 (TYPE-A, 'DHSETCT', 4 BYTES)
Number of document handler SET requests issued by the user task.

229 (TYPE-A, 'DHRETCT', 4 BYTES)
Number of document handler RETRIEVE requests issued by the user task.

230 (TYPE-A, 'DHTOTCT', 4 BYTES)
Total number of document handler requests issued by the user task.

240 (TYPE-A, 'DHTOTDCL', 4 BYTES)
Total length of all documents created by the user task.

Performance data in group DFHFILE

Group DFHFILE contains the following performance data:

036 (TYPE-A, 'FCGETCT', 4 BYTES)
Number of file GET requests issued by the user task.

037 (TYPE-A, 'FCPUTCT', 4 BYTES)
Number of file PUT requests issued by the user task.

038 (TYPE-A, 'FCBRWCT', 4 BYTES)
Number of file browse requests issued by the user task. This number excludes the START and END browse requests.

039 (TYPE-A, 'FCADDCT', 4 BYTES)
Number of file ADD requests issued by the user task.

040 (TYPE-A, 'FCDELCT', 4 BYTES)
Number of file DELETE requests issued by the user task.

063 (TYPE-S, 'FCIOWTT', 8 BYTES)
Elapsed time in which the user task waited for file I/O. For more information, see "Clocks and time stamps" on page 220.

070 (TYPE-A, 'FCAMCT', 4 BYTES)
Number of times the user task invoked file access-method interfaces. This number excludes requests for OPEN and CLOSE.

093 (TYPE-A, 'FCTOTCT', 4 BYTES)
Total number of file control requests issued by the user task. This number excludes any request for OPEN, CLOSE, ENABLE, or DISABLE of a file.

174 (TYPE-S, 'RLSWAIT', 8 BYTES)
Elapsed time in which the user task waited for RLS file I/O.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

175 (TYPE-S, 'RLSCPUT', 8 BYTES)
The RLS File Request CPU (SRB) time field (RLSCPUT) is the SRB CPU time this transaction spent processing RLS file requests. This field should be added to the transaction CPU time field (USRCPUT) when considering the measurement of the total CPU time consumed by a transaction. Also, this field cannot be considered a subset of any other single CMF field (including RLSWAIT). This is because the RLS field requests execute asynchronously under an MVS SRB which can be running in parallel with the requesting transaction. It is also possible for the SRB to complete its processing before the requesting transaction waits for the RLS file request to complete.

Note: This clock field could contain a CPU time of zero with a count of greater than zero. This is because the CMF timing granularity is measured in 16 microsecond units and the RLS file request(s) may complete in less than that time unit.

176 (TYPE-S, 'CFDTWAIT', 8 BYTES)
Elapsed time in which the user task waited for coupling facility data table I/O.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

How EXEC CICS file commands correspond to file control monitoring fields is shown in Table 12.

Table 12. EXEC CICS file commands related to file control monitoring fields

EXEC CICS command	Monitoring fields
READ	FCGETCT and FCTOTCT
READ UPDATE	FCGETCT and FCTOTCT
DELETE (after READ UPDATE)	FCDELCT and FCTOTCT
DELETE (with RIDFLD)	FCDELCT and FCTOTCT
REWRITE	FCPUTCT and FCTOTCT
WRITE	FCADDCT and FCTOTCT
STARTBR	FCTOTCT
READNEXT	FCBRWCT and FCTOTCT
READNEXT UPDATE	FCBRWCT and FCTOTCT
READPREV	FCBRWCT and FCTOTCT
READPREV UPDATE	FCBRWCT and FCTOTCT
ENDBR	FCTOTCT
RESETBR	FCTOTCT
UNLOCK	FCTOTCT

Note: The number of STARTBR, ENDBR, RESETBR, and UNLOCK file control requests can be calculated by subtracting the file request counts, FCGETCT, FCPUTCT, FCBRWCT, FCADDCT, and FCDELCT from the total file request count, FCTOTCT.

Performance data in group DFHPROG

Group DFHPROG contains the following performance data:

055 (TYPE-A, 'PCLINKCT', 4 BYTES)
Number of program LINK requests issued by the user task, including the link to the first program of the user task. This field does not include program LINK URM (user-replaceable module) requests.

056 (TYPE-A, 'PCXCTLCT', 4 BYTES)
Number of program XCTL requests issued by the user task.

057 (TYPE-A, 'PCLOADCT', 4 BYTES)
Number of program LOAD requests issued by the user task.

071 (TYPE-C, 'PGMNAME', 8 BYTES)
The name of the first program invoked at attach-time.

For a remote transaction:

- If this CICS definition of the remote transaction does not specify a program name, this field contains blanks.
- If this CICS definition of the remote transaction specifies a program name, this field contains the name of the specified program. (Note that this is not necessarily the program that is run on the remote system.)

For a dynamically-routed transaction, if the dynamic transaction routing program routes the transaction locally and specifies an alternate program name, this field contains the name of the alternate program.

For a dynamic program link (DPL) mirror transaction, this field contains the initial program name specified in the dynamic program LINK request. DPL mirror transactions can be identified using byte 1 of the transaction flags, TRANFLAG (164), field.

For an ONC RPC or WEB alias transaction, this field contains the initial application program name invoked by the alias transaction. ONC RPC or WEB alias transactions can be identified using byte 1 of the transaction flags, TRANFLAG (164), field.

072 (TYPE-A, 'PCLURMCT', 4 BYTES)
Number of program LINK URM (user-replaceable module) requests issued by, or on behalf of, the user task.

A user-replaceable module is a CICS-supplied program that is always invoked at a particular point in CICS processing, as if it were part of the CICS code. You can modify the supplied program by including your own logic, or replace it completely with a version that you write yourself.

The CICS-supplied user-replaceable modules are:

- Bridge exit program
- Program error program
- Transaction restart program
- Terminal error program
- Node error program
- Terminal autoinstall programs
- Program autoinstall program
- Dynamic routing program
- CICS-DBCTL interface status program
- CICS-DB2 dynamic plan exit program.

For detailed information on CICS user-replaceable programs, see the *CICS Customization Guide*.

073 (TYPE-A, 'PCDPLCT', 4 BYTES)
Number of distributed program LINK (DPL) requests issued by the user task.

113 (TYPE-C, 'ABCODEO', 4 BYTES)
Original abend code.

114 (TYPE-C, 'ABCODEC', 4 BYTES)
Current abend code.

115 (TYPE-S, 'PCLOADTM', 8 BYTES)
Elapsed time in which the user task waited for program library (DFHRPL) fetches. Only fetches for programs with installed program definitions or autoinstalled as a result of application requests are included in this figure. However, installed programs residing in the LPA are not included (because they do not incur a physical fetch from a library).

Performance data in group DFHSOCK

Group DFHSOCK contains the following performance data:

241 (TYPE-S, 'SOIOWTT', 8 BYTES)
The elapsed time in which the user task waited for socket I/O.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

242 (TYPE-A, 'SOBYENCT', 4 BYTES)
The number of bytes encrypted by the secure sockets layer for the user task.

243 (TYPE-A, 'SOBYDECT', 4 BYTES)
The number of bytes decrypted by the secure sockets layer for the user task.

244 (TYPE-C, 'CLIPADDR', 16 BYTES)
Client IP address (*nnn.nnn.nnn.nnn*).

Performance data in group DFHSYNC

Group DFHSYNC contains the following performance data:

060 (TYPE-A, 'SPSYNCCT', 4 BYTES)
Number of SYNCPOINT requests issued during the user task.

Notes:

1. A SYNCPOINT is implicitly issued as part of the task-detach processing.
2. A SYNCPOINT is issued at PSB termination for DBCTL.

173 (TYPE-S, 'SYNCTIME', 8 BYTES)
Total elapsed time for which the user task was dispatched and was processing syncpoint requests.

For more information about syncpoint time see "Clocks and time stamps" on page 220.

177 (TYPE-S, 'SRVSYWTT', 8 BYTES)
The elapsed time in which the user task waited for a coupling facility data table server to process syncpoint requests issued by the user task.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

196 (TYPE-S, 'SYNCDLY', 8 BYTES)
The elapsed time in which the user task waited for a syncpoint request to be issued by its parent transaction. The user task was executing as a result of a parent task issuing a CICS BTS run-process or run-activity request to execute a process or activity synchronously.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

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 Front End Programming Interface (FEPI)

007 (TYPE-S, 'USRDISPT', 8 BYTES)
Total elapsed time during which the user task was dispatched on each CICS TCB under which the task executed. This can include TCB modes QR, RO, CO, FO, SZ, RP, SL, SO, L8, J8, and S8. For more information, see "Clocks and time stamps" on page 220.

008 (TYPE-S, 'USRCPUT', 8 BYTES)
Total processor time during which the user task was dispatched on each CICS TCB under which the task executed. This can include TCB modes QR, RO, CO, FO, SZ, RP, SL, SO, L8, J8, and S8. For more information, see "Clocks and time stamps" on page 220.

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 redispatch after a suspended task has been resumed.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

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
- **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 32 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

066 (TYPE-A, 'ICTOTCT', 4 BYTES)
Total number of interval control START, CANCEL, DELAY, and RETRIEVE requests issued by the user task.

082(TYPE-C, 'TRNGRPID', 28 BYTES)
Transaction group id that is assigned at transaction attach time, and can be used to correlate the transactions that CICS executes for the same incoming work request (for example, the CWXN and CWBA transactions for Web requests). This transaction group ID applies to requests that originate through the CICS Web, IIOP, or 3270 bridge interface, as indicated by the tranasction origin in byte 4 of the transaction flags (164, TRANFLAG).

097 (TYPE-C, 'NETUOWPX', 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, the NETNAME is the *networkid.LUname*. If the terminal is non-VTAM, 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 an external CICS interface (EXCI) session, the name is a concatenation of:

'DFHEXCIU	.	MVS Id	Address Space Id (ASID)'
8 bytes	1 byte	4 bytes	4 bytes

derived from the originating system. That is, the name is a 17-byte LU name consisting of:

- An 8-byte eye-catcher set to 'DFHEXCIU'.
- A 1-byte field containing a period (.).
- A 4-byte field containing the MVSID, in characters, under which the client program is running.

- A 4-byte field containing the address space ID (ASID) in which the client program is running. This field contains the 4-character EBCDIC representation of the 2-byte hex address space ID.

098 (TYPE-C, 'NETUOWSX', 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 is a binary value derived from the clock of the originating system and wrapping round at intervals of several months.

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 NETUOWSX field must be combined with the NETUOWPX field (097) to uniquely identify a task, because the NETUOWSX field is unique only to the originating CICS system.

102 (TYPE-S, 'DISPWTT', 8 BYTES)
Elapsed time for which the user task waited for redispatch. This is the aggregate of the wait times between each event completion and user-task redispatch.

Note: This field does not include the elapsed time spent waiting for first dispatch. The DISPWTT field is a component 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).

123 (TYPE-S, 'GNQDELAY', 8 BYTES)
The elapsed time waiting for a CICS task control global enqueue.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

124 (TYPE-C, 'BRDGTRAN', 4 BYTES)
3270 Bridge transaction identification.

125 (TYPE-S, 'DSPDELAY', 8 BYTES)
The elapsed time waiting for first dispatch.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: This field is a component 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. For more information, see "Clocks and time stamps" on page 220.

Note: This field is a component 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 component of the first dispatch delay, DSPDELAY (125), field.

128 (TYPE-S, 'LMDELAY', 8 BYTES)
The elapsed time that the user task waited to acquire a lock on a resource. A user task cannot explicitly acquire a lock on a resource, but many CICS modules lock resources on behalf of user tasks using the CICS lock manager (LM) domain.

For more information about CICS lock manager waits, see the *CICS Problem Determination Guide*.

For information about times, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

129 (TYPE-S, 'ENQDELAY', 8 BYTES)
The elapsed time waiting for a CICS task control local enqueue.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

132 (TYPE-C, 'RMUOWID', 8 BYTES)
The identifier of the logical unit-of-work (unit-of-recovery) for this task. Unit-of recovery values are used to synchronize recovery operations among CICS and other resource managers, such as IMS and DB2.

163 (TYPE-C, 'FCTYNAME', 4 BYTES)
Transaction facility name. This field is null if the transaction is not associated with a facility. The transaction facility type (if any) can be identified using byte 0 of the transaction flags, TRANFLAG, (field id 164).

164 (TYPE-A, 'TRANFLAG', 8 BYTES)
Transaction flags, a string of 64 bits used for signaling transaction definition and status information:

Byte 0
Transaction facility identification

Bit 0 Transaction facility name = none X'80'

Bit 1 Transaction facility name = terminal X'40'. If this Bit is set, the FCTYNAME and TERM fields contain the same terminal id.

Bit 2 Transaction facility name = surrogate X'20'. If this Bit is set, the FCTYNAME field contains the name of the surrogate terminal id.

Bit 3 Transaction facility name = destination X'10'

Bit 4 Transaction facility name = 3270 bridge X'08'

Bits 5–7
Reserved

Byte 1

Transaction identification information

Bit 0 System transaction X'80'
Bit 1 Mirror transaction X'40'
Bit 2 DPL mirror transaction X'20'
Bit 3 ONC RPC alias transaction X'10'
Bit 4 WEB alias transaction X'08'
Bit 5 3270 bridge transaction X'04'
Bits 6 Reserved X'02'
Bits 7 CICS BTS run transaction X'01'

Byte 2

MVS workload manager request (transaction) completion information

Bit 0 Report the total response time for completed work request (transaction)
Bit 1 Notify that the entire execution phase of the work request is complete
Bit 2 Notify that a subset of the execution phase of the work request is complete
Bits 3–7
Reserved

Byte 3

Transaction definition information

Bit 0 Taskdataloc = below (x'80')
Bit 1 Taskdatakey = cics (x'40')
Bit 2 Isolate = no (x'20')
Bit 3 Dynamic = yes (x'10')
Bits 4–7
Reserved

Byte 4

Transaction origin type

Byte 5

Reserved

Byte 6

Reserved

Byte 7

Recovery manager information

Bit 0 Indoubt wait = no
Bit 1 Indoubt action = commit
Bit 2 Recovery manager - UOW resolved with indoubt action
Bit 3 Recovery manager - shunt
Bit 4 Recovery manager - unshunt
Bit 5 Recovery manager - indoubt failure
Bit 6 Recovery manager - resource owner failure
Bit 7 Reserved

Note: Bits 2 through 6 is reset on a SYNCPOINT request when the MNSYNC=YES option is specified.

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). For more information, see "Clocks and time stamps" on page 220.

171 (TYPE-S, 'RMISUSP', 8 BYTES)
Amount of elapsed time the task was suspended by the dispatcher while in the resource manager interface (RMI). For more information, see "Clocks and time stamps" on page 220, and "Wait (suspend) times" on page 222.

Note: The field is a component of the task suspend time, SUSPTIME (014), field and also the RMITIME (170) field.

181 (TYPE-S, 'WTEXWAIT', 8 BYTES)
The elapsed time that the user task waited for one or more ECBs, passed to CICS by the user task using the EXEC CICS WAIT EXTERNAL ECBLIST command, to be MVS POSTed. The user task can wait on one or more ECBs. If it waits on more than one, it is dispatchable as soon as one of the ECBs is posted.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

182 (TYPE-S, 'WTCEWAIT', 8 BYTES)
The elapsed time that the user task waited for:

1. One or more ECBs, passed to CICS by the user task using the EXEC CICS WAITCICS ECBLIST command, to be MVS POSTed. The user task can wait on one or more ECBs. If it waits on more than one, it is dispatchable as soon as one of the ECBs is posted.
2. Completion of an event initiated by the same or by another user task. The event would normally be the posting, at the expiration time, of a timer-event control area provided in response to an EXEC CICS POST command. The EXEC CICS WAIT EVENT command provides a method of directly relinquishing control to some other task until the event being waited on is completed.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

183 (TYPE-S, 'ICDELAY', 8 BYTES)
The elapsed time the user task waited as a result of issuing either:

- An interval control EXEC CICS DELAY command for a specified time interval, or
- A specified time of day to expire, or
- An interval control EXEC CICS RETRIEVE command with the WAIT option specified.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

184 (TYPE-S, 'GVUPWAIT', 8 BYTES)
The elapsed time that the user task waited as a result of relinquishing control to

another task. A user task can relinquish control in many ways. Some examples are application programs that use one or more of the following EXEC CICS API or SPI commands:

- Using the EXEC CICS SUSPEND command. This command causes the issuing task to relinquish control to another task of higher or equal dispatching priority. Control is returned to this task as soon as no other task of a higher or equal priority is ready to be dispatched.
- Using the EXEC CICS CHANGE TASK PRIORITY command. This command immediately changes the priority of the issuing task and causes the task to relinquish control in order for it to be redispatched at its new priority. The task is not redispatched until tasks of higher or equal priority, and that are also dispatchable, have been dispatched.
- Using the EXEC CICS DELAY command with INTERVAL(0). This command causes the issuing task to relinquish control to another task of higher or equal dispatching priority. Control is returned to this task as soon as no other task of a higher or equal priority is ready to be dispatched.
- Using the EXEC CICS POST command requesting notification that a specified time has expired. This command causes the issuing task to relinquish control to give CICS the opportunity to post the timer-event control area.
- Using the EXEC CICS PERFORM RESETTIME command to synchronize the CICS date and time with the MVS system date and time of day.
- Using the EXEC CICS START TRANSID command with the ATTACH option.

For more information, see “Clocks and time stamps” on page 220 and “Wait (suspend) times” on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

190 (TYPE-C, ‘RRMSURID’, 16 BYTES)
RRMS/MVS unit-of-recovery ID (URID).

191 (TYPE-S, ‘RRMSWAIT’, 8 BYTES)
The elapsed time in which the user task waited indoubt using resource recovery services for EXCI.

For more information, see “Clocks and time stamps” on page 220 and “Wait (suspend) times” on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

195 (TYPE-S, ‘RUNTRWTT’, 8 BYTES)
The elapsed time in which the user task waited for completion of a transaction that executed as a result of the user task issuing a CICS BTS run process, or run activity, request to execute a process, or activity, synchronously.

For more information, see “Clocks and time stamps” on page 220 and “Wait (suspend) times” on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

248 (TYPE-A, ‘CHMODECT’, 4 BYTES)
The number of CICS change-TCB modes issued by the user task.

249 (TYPE-S, 'QRMODDLY', 8 BYTES)
Elapsed time for which the user task waited for redispatch on the CICS QR TCB. This is the aggregate of the wait times between each event completion and user-task redispatch.

Note: This field does not include the elapsed time spent waiting for the first dispatch. The QRMODDLY field is a component of the task suspend time, SUSPTIME (014), field.

250 (TYPE-S, 'MAXOTDLY', 8 BYTES)
The elapsed time in which the user task waited to obtain a CICS open TCB, because the region had reached the limit set by the system parameter, MAXOPENTCBS.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: This field is a component of the task suspend time, SUSPTIME (014), field.

251 (TYPE-A, 'TCBATTCT', 4 BYTES)
Number of CICS TCBs attached by the user task.

253 (TYPE-S, 'JVMTIME', 8 BYTES)
Elapsed time spent in the CICS JVM by the user task.

254 (TYPE-S, 'JVMSUSP', 8 BYTES)
Elapsed time the task was suspended by the CICS dispatcher while running in the CICS JVM.

255 (TYPE-S, 'QRDISPT', 8 BYTES)
Elapsed time for which the user task was dispatched on the CICS QR TCB. For more information, see "Clocks and time stamps" on page 220.

256 (TYPE-S, 'QRCPUT', 8 BYTES)
Processor time for which the user task was dispatched on the CICS QR TCB. For more information, see "Clocks and time stamps" on page 220.

257 (TYPE-S, 'MSDISPT', 8 BYTES)
Total elapsed time for which the user task was dispatched on each CICS TCB, mode RO, CO, FO, SZ, RP, SL, or SO. Note that:

- Mode SZ is used only if FEPI is active
- Mode RP is used only if ONC RPC support is active, or the CICS WEB interface is active
- Modes SO and SL are used only if TCPIP=YES is specified as a system initialization parameter.

For more information, see "Clocks and time stamps" on page 220.

258 (TYPE-S, 'MSCPUT', 8 BYTES)
Total processor time for which the user task was dispatched on each CICS TCB, mode RO, CO, FO, SZ, RP, SL, or SO. Note that:

- Mode SZ is used only if FEPI is active
- Mode RP is used only if ONC RPC support is active, or the CICS WEB interface is active
- Modes SO and SL are used only if TCPIP=YES is specified as a system initialization parameter.

For more information, see "Clocks and time stamps" on page 220.

259 (TYPE-S, 'L8CPUT', 8 BYTES)
Processor time for which the user task was dispatched on the CICS L8 TCB. For more information, see "Clocks and time stamps" on page 220.

260 (TYPE-S, 'J8CPUT', 8 BYTES)
Processor time for which the user task was dispatched on the CICS J8 TCB. For more information, see "Clocks and time stamps" on page 220.

261 (TYPE-S, 'S8CPUT', 8 BYTES)
Processor time for which the user task was dispatched on the CICS S8 TCB. For more information, see "Clocks and time stamps" on page 220.

Performance data in group DFHTEMP

Group DFHTEMP contains the following performance data:

011 (TYPE-S, 'TSIOWTT', 8 BYTES)
Elapsed time for which the user task waited for VSAM temporary storage I/O. For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: The field is a component of the task suspend time, SUSPTIME (014), field.

044 (TYPE-A, 'TSGETCT', 4 BYTES)
Number of temporary storage GET requests issued by the user task.

046 (TYPE-A, 'TSPUTACT', 4 BYTES)
Number of PUT requests to auxiliary temporary storage issued by the user task.

047 (TYPE-A, 'TSPUTMCT', 4 BYTES)
Number of PUT requests to main temporary storage issued by the user task.

092 (TYPE-A, 'TSTOTCT', 4 BYTES)
Total number of temporary storage requests issued by the user task. This field is the sum of TS GET, PUT AUXILIARY, PUT MAIN, and DELETE requests issued by the user task.

178 (TYPE-S, 'TSSHWAIT', 8 BYTES)
Elapsed time that the user task waited for an asynchronous shared temporary storage request to a temporary storage data server to complete.

For more information, see "Clocks and time stamps" on page 220 and "Wait (suspend) times" on page 222.

Note: The field is a component of the task suspend time, SUSPTIME (014), field.

Performance data in group DFHWEBB

Group DFHWEBB contains the following performance data:

231 (TYPE-A, 'WBRCVCT', 4 BYTES)
Number of CICS Web interface RECEIVE requests issued by the user task.

232 (TYPE-A, 'WBCHRIN', 4 BYTES)
Number of characters received by the CICS Web interface RECEIVE requests issued by the user task.

233 (TYPE-A, 'WBSENDCT', 4 BYTES)
Number of CICS Web interface SEND requests issued by the user task.

234 (TYPE-A, 'WBCHROUT', 4 BYTES)
Number of characters sent by the CICS Web interface SEND requests issued by the user task.

235 (TYPE-A, 'WBTOTCT', 4 BYTES)
Total number of CICS Web interface requests issued by the user task.

236 (TYPE-A, 'WBREPRCT', 4 BYTES)
Number of reads from the repository in shared temporary storage issued by the user task.

237 (TYPE-A, 'WBREPWCT', 4 BYTES)
Number of writes to the repository in shared temporary storage issued by the user task.

Exception class data

CICS monitoring domain now produces exception records after each of the following transaction waits has been resolved:

- Wait for coupling facility data table locking request slot
- Wait for coupling facility data table non-locking request slot
- 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 LSRPOOL buffer
- Wait for LSRPOOL string

These records are fixed format. 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
EXCMNSRV DS    CL8               SERVICE CLASS NAME
EXCMNRPT DS    CL8               REPORT CLASS NAME
EXCMNNPX DS    CL20              NETWORK UNIT-OF-WORK PREFIX
EXCMNNSX DS    XL8               NETWORK UNIT-OF-WORK SUFFIX
```

```
EXCMNTRF DS    XL8               TRANSACTION FLAGS
EXCMNFCN DS    CL4               TRANSACTION FACILITY NAME
EXCMNCPN DS    CL8               CURRENT PROGRAM NAME
EXCMNBTR DS    CL4               BRIDGE TRANSACTION ID
EXCMNURI DS    XL16              MVS/RRMS Unit of Recovery Id
EXCMNRIL DS    F                 EXCEPTION RESOURCE ID LENGTH
EXCMNRIX DS    XL256             EXCEPTION RESOURCE ID (EXTENDED)
*        END OF EXCEPTION RECORD ...
```

Exception data field descriptions

The following is a full list of all the exception record fields, including the three new fields:

EXCMNTRN (TYPE-C, 4 BYTES)
Transaction identification.

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, 8 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.

EXCMNRID (TYPE-C, 8 BYTES)
Exception resource identification.

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.

EXCMNSRV (TYPE-C, 8 BYTES)
MVS workload manager service class name for this transaction. This field is null if the transaction was WLM-classified in another CICS region.

EXCMNRPT (TYPE-C, 8 BYTES)
MVS workload manager report class name for this transaction. This field is null if the transaction was WLM-classified in another CICS region.

EXCMNNPX (TYPE-C, 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 a VTAM device across an ISC APPC or IRC link, the NETNAME is the *networkid.LUname*. If the terminal is non-VTAM, 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 an external CICS interface (EXCI) session, the name is a concatenation of:

DFHEXCIU	.	MVS ID	Address Space Id (ASID)
8 bytes	1 byte	4 bytes	4 bytes

derived from the originating system. That is, the name is a 17-byte LU name consisting of:

- An 8-byte eye-catcher set to 'DFHEXCIU'.
- A 1-byte field containing a period (.).
- A 4-byte field containing the MVS ID, in characters, under which the client program is running.
- A 4-byte field containing the address space ID (ASID) in which the client program is running. This field contains the 4-character EBCDIC representation of the 2-byte hex address space ID.

EXCMNNSX (TYPE-C, 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 contain a binary value derived from the clock of the originating system and wrapping round at intervals of several months.

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 EXCMNNSX field must be combined with the EXCMNNPX field to uniquely identify a task, because the EXCMNNSX field is unique only to the originating CICS system.

EXCMNTRF (TYPE-C, 8 BYTES)

Transaction flags—a string of 64 bits used for signaling transaction definition and status information:

Byte 0

Transaction facility identification

Bit 0 Transaction facility name = none
Bit 1 Transaction facility name = terminal
Bit 2 Transaction facility name = surrogate
Bit 3 Transaction facility name = destination
Bit 4 Transaction facility name = 3270 bridge
Bits 5–7
Reserved

Byte 1

Transaction identification information

Bit 0 System transaction
Bit 1 Mirror transaction
Bit 2 DPL mirror transaction
Bit 3 ONC RPC alias transaction
Bit 4 WEB alias transaction
Bit 5 3270 bridge transaction
Bits 6–7
Reserved

Byte 2

MVS Workload Manager information

Bit 0 Workload Manager report
Bit 1 Workload Manager notify, completion = yes
Bit 2 Workload Manager notify
Bits 3–7
Reserved

Byte 3

Transaction definition information

Bit 0 Taskdataloc = below
Bit 1 Taskdatakey = cics
Bit 2 Isolate = no
Bit 3 Dynamic = yes
Bits 4–7
Reserved

Byte 4

Reserved

Byte 5

Reserved

Byte 6

Reserved

Byte 7

Recovery manager information

Bit 0 Indoubt wait = no

Bit 1 Indoubt action = commit
Bit 2 Recovery manager - UOW resolved with indoubt action
Bit 3 Recovery manager - shunt
Bit 4 Recovery manager - unshunt
Bit 5 Recovery manager - indoubt failure
Bit 6 Recovery manager - resource owner failure
Bit 7 Reserved

Note: Bits 2 through 6 will be reset on a SYNCPOINT request when the MNSYNC=YES option is specified.

EXCMNFCN (TYPE-C, 4 BYTES)
Transaction facility name. This field is null if the transaction is not associated with a facility. The transaction facility type (if any) can be identified using byte 0 of the transaction flags field, EXCMNTRF.

EXCMNCPN (TYPE-C, 8 BYTES)
The name of the currently running program for this user task when the exception condition occurred.

EXCMNBTR (TYPE-C, 4 BYTES)
3270 Bridge transaction identification.

EXCMNURI (TYPE-C, 16 BYTES)
RRMS/MVS unit-of-recovery ID (URID).

EXCMNRIL (TYPE-A, 4 BYTES)
Exception resource ID length.

EXCMNRIX (TYPE-C, 256 BYTES)
Exception resource ID (extended).

Definition of terms

reenterable. See reentrant

reentrant. 1. (*From the MVS Assembler Services Guide*). The attribute that describes a load module, of which only one copy is loaded into virtual storage to satisfy the requirements of any number of tasks. A single copy of a reentrant load module can be executed concurrently by any number of tasks. A reentrant load module is also one that does not modify itself, and must be link-edited with the RENT attribute.

2. Reenterable (reentrant). (*From the **DFSMS Program Management** manual*). The module is designed for concurrent execution by multiple tasks. If a reenterable module modifies its own data areas or other shared resources in any way, it must use appropriate serialization methods to prevent interference between using tasks.

serially reusable. 1. The attribute that describes a serially reusable load module. Only one copy of a serially reusable load module is loaded into virtual storage to satisfy the requirements of any number of tasks, but only one task can execute the module at any one time. If the copy is in use when a request is issued for the module, the task requiring the module is placed in a wait condition until the module is available.

2. Serially reusable. (*From the **DFSMS Program Management** manual*). The module is designed to be reused and therefore must contain the necessary logic to reset control variables and data areas at entry or exit. A second task may not enter the module until the first task has finished.

quasi-reentrant. The attribute used to describe CICS application programs that run under the CICS quasi-reentrant task control block (QR TCB). This means that:

- CICS obtains a separate copy of program working storage for each task that executes application program code.
- CICS allows only one task at a time to execute application program code. In this way, CICS ensures the necessary serialization of user application programs that access any kind of shared resources, whether CICS- or user-managed. This means that different tasks cannot interfere with each other. Thus the user application program need not be reenterable strictly according to the DFSMS program management definition (see *reentrant*).

Although only one user task can execute an application program at any one time, a second user task can enter the program before another task has finished with it, unlike a *serially reusable* module as defined by DFSMS. This is because user applications programs give up control part way through execution whenever they issue an EXEC CICS command that causes a wait. Thus a user program can be in use concurrently by more than one task, indicated by the use count maintained by CICS, which can be greater than one.

Whenever an application program receives control, it should be in the same state as when it relinquished control on a previous invocation.

Index

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

Sending your comments to IBM

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

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

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

To 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, to this address:

  Information Development Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  United Kingdom
- By fax:
  - From outside the U.K., after your international access code use 44–1962–870229
  - From within the U.K., use 01962–870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink™: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

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

Program Number: 5655-147

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

GC34-5352-01

Spine information:

CICS TS for OS/390 Release Guide Release 3