IBM

CICS Transaction Server from Start to Finish

- CICS Transaction Server introduction and history
- CICS Transaction server facilities, interfaces, and data
- CICS Explorer and the CICS Tools

Chris Rayns
Sarah Bertram
George Bogner
Chris Carlin
Andre Clark
Amy Ferrell
Gordon Keehn
Peter Klein
Ronald Lee
Erhard Woerner

Redbooks

ibm.com/redbooks

International Technical Support Organization

CICS Transaction Server from Start to Finish

December 2011

SG24-7952-00

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

First Edition (December 2011)

This edition applies to Version 4, Release 2, CICS Transaction Server.

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.

The following paragraph does not apply to 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 information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

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

AIX®	MVS™	RMF™
CICS Explorer™	NetView®	S/390®
CICSPlex®	OMEGAMON®	System z®
CICS®	OS/390®	Tivoli®
DataPower®	Parallel Sysplex®	VTAM®
DB2 Universal Database™	POWER®	WebSphere®
DB2®	RACF®	z/OS®
IBM®	Rational®	zSeries®
IMS™	Redbooks®	
Language Environment®	Redbooks (logo) ®	

The following terms are trademarks of other companies:

Intel, Itanium, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Preface

In this IBM® Redbooks® publication, we discuss CICS®, which stands for Customer Information Control System. It is a general-purpose transaction processing subsystem for the z/OS® operating system. CICS provides services for running an application online where, users submit requests to run applications simultaneously.

CICS manages sharing resources, the integrity of data, and prioritizes execution with fast response. CICS authorizes users, allocates resources (real storage and cycles), and passes on database requests by the application to the appropriate database manager, such as DB2®.

We review the history of CICS and why it was created. We review the CICS architecture and discuss how to create an application in CICS. CICS provides a secure, transactional environment for applications that are written in several languages. We discuss the CICS-supported languages and each language's advantages in this Redbooks publication.

We analyze situations from a system programmer's viewpoint, including how the systems programmer can use CICS facilities and services to customize the system, design CICS for recovery, and manage performance. CICS Data access and where the data is stored, including Temporary storage queues, VSAM RLS, DB2, IMS™, and many others are also discussed.

We review some of the interfaces that are available in CICS including:

- CICS web services
- CICS Transaction Gateway
- CICS web support
- WebSphere® MQ
- CICS sockets

We explain what CICSPlex® SM is and why you would want to use CICSPlex SM. We also discuss the CICS Explorer™ and the benefits that are gained from using the CICS Explorer.

We also examine the CICS Tools that can help you optimize and debug your CICS-based applications. These tools include:

- CICS Performance Analyzer for optimizing application performance and monitoring application resource utilization

- CICS Interdependency Analyzer to understand application dependencies and understand what CICS resources are used by your application
- CICS Deployment Assistant, which helps you discover, model, visualize, create, and deploy CICS regions using the CICS DA plug-in interface in CICS Explorer.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

Chris Rayns is an IT Specialist and Project Leader at the ITSO, Poughkeepsie Center in New York. Chris writes extensively about all areas of CICS TS and CICS TG. Before joining the ITSO, he worked in IBM Global Services in the United Kingdom as a CICS IT Specialist.

Sarah Bertram is an Advisory Software Engineer in the United States. She has 15 years of experience as a CICS Level II support technician. Her focus is with storage and communications. She has worked for IBM for 27 years, spending the first 12 years in support and development of Remote Spooling Communications Subsystem (RSCS), a VM-related product. She has a Bachelor of Science degree in Computer Science with a minor in mathematics from the State University of New York at Oswego.

George Bogner is a Software IT Specialist working in IBM Sales and Distribution supporting the CICS Transaction Server product suite. George worked at IBM for 25 years, specializing in the change to data management and transaction management area, working with CICS, IMS, and DB2, supporting client accounts in IBM Global Services. He currently works out of Raleigh, North Carolina, supporting North American clients by providing CICS seminars, proofs of technology (POT), proofs of concept (POC), and consulting services for CICS-related topics.

Chris Carlin is a Staff Software Engineer in the United States. He has six years of experience as a CICS Level II Technical Support representative in Research Triangle Park, North Carolina. He has worked at IBM for 15 years, spending nine years as an Application Developer at various IBM clients. He has a Bachelor of Arts degree in Computer Science and a minor in Management from New Jersey Institute of Technology (NJIT).

Andre Clark is a CICS Level 2 Support person working in Raleigh, NC. Prior to working at CICS Level 2, he was an IT Specialist dealing with WebSphere and UNIX/AIX®. Andre has been with IBM for five years. He has a Bachelor of Science degree in Computer and Information Systems from North Carolina Central University.

Amy Ferrell is a CICS Information Developer in Hursley, United Kingdom. She has a Bachelor of Science degree in Computing and Information Systems from the University of the West of England, Bristol. Her areas of expertise are human and computer inteaction, technical writing, and information system design. She has written extensively about system usability and accessibility and the adoption of precision technologies in the UK agricultural industry.

Gordon Keehn is an Advisory Software Engineer in the United States. He has 18 years of experience supporting CICSPlex System manager. Before becoming involved with CICSPlex SM, he worked as an MVS™ System Programmer for several large corporations. He has a Bachelor of Arts degree in Chemistry from Hartwick College and a Master of Science degree in Organic Chemistry from the University of Pennsylvania.

Peter Klein is a CICS Team Leader at the IBM Germany Customer Support Center. He has 19 years of experience working as a Technical Support Specialist with IBM software products. His expertise includes WebSphere MQ, CICSPlex System Manager, and distributed transaction systems. Peter contributed to several other IBM Redbooks publications and ITSO projects sponsored by IBM Learning Services.

Ronald Lee is a Staff Software Engineer in the United States. He has 11 years of experience as a CICS Level II Technical Support representative in Research Triangle Park, North Carolina. He has been with IBM for 16 years, spending five years as a Systems Programmer for IBM Global Services in St. Louis, Missouri for various clients. He has a Bachelor of Science Degree in Management Information Systems from Maryville University.

Erhard Woerner is a Software Support Specialist in Germany. He has worked for IBM in CICS Level 2 Support for eleven years. He has 25 years of experience working with CICS in several roles. Before Erhard joined IBM, he worked as a CICS Systems Programmer at Deutsche Bank AG. He has excellent knowledge in the CICS programming languages COBOL, PL/I, and Assembler. He is an IBM Certifed Application Developer with IBM Enterprise PL/I. His areas of expertise are Virtual Storage Access Method (VSAM), CICS VSAM Recovery (CICSVR), CICS Web Services, and the z/OS System Logger. He is an IBM Certifed Application Developer with IBM Enterprise PL/I. He is certified for System z® COBOL Application Programming at Marist College Institue for Data Center Professionals (IDCP) in Poughkeepsie, New York.

Thanks to the following people for their contributions to this project:

Richard M Conway
International Technical Support Organization, Raleigh Center

Timothy J Wuthenow
IBM Raleigh

Kaamil Baba kamara
IBM New York

Matthew Whitbourne
IBM Hursley

Nick Garrod
IBM Hursley

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Part 1

CICS introduction and history

In this Part of the book, we give an introdcution to CICS TS and review its history.

1

Introducing CICS

This chapter reviews what the Customer Information Control System (CICS) is and what it does. It explains how CICS evolved to shape world commerce and meet our ever changing needs as an information society.

1.1 What is CICS

Customer Information Control System (CICS) is the IBM transaction processing software that primarily runs on IBM mainframes under z/OS. Transaction processors, such as CICS are purpose-built to efficiently run business programs in a secure, transactional, multi-user environment.

CICS controls the interaction between applications and users. Depending on the size of the organization, CICS must support a small number of users to thousands every day, and each user might need to perform a number of transactions repetitively. Almost any computer system can be designed to process transactions, but only a purpose-built system can handle extremely large volumes of transactions efficiently.

CICS Transaction Server provides the following services:

- A highly efficient and optimized environment for running transactions. CICS transaction server manages concurrency, sharing of resources, integrity of data, and task prioritization securely.
- Support for business applications written in COBOL, C, C++, Java, Assembler, PL/I REXX, PHP using an application programming interface that provides access to CICS services.
- Access by applications to data stored in DB2 and IMS, and VSAM and BDAM data sets.
- Inter-operation with WebSphere MQ and access to Message Queue Interface from CICS application programs.
- Distribution of work between multiple CICS regions in both a z/OS sysplex and non-sysplex environment.
- Connectivity with non-CICS systems in client/server and peer-to-peer configurations.
- Interfaces for configuring and managing CICS regions.
- Aids for debugging application programs and for diagnosing system problems.

CICS builds on z/OS and System z facilities to provide high availability, integrity, scalability, performance, and security at a low cost per transaction. These services are explained in further detail in later chapters.

1.2 Transaction processing

Transaction processing is a style of computing, typically performed by large server computers, that supports interactive applications. In transaction processing, work is divided into individual, indivisible operations, called transactions. By contrast, batch processing is a style of computing in which one or more programs process a series of records (a batch) with little or no action from the user or operator.

A transaction processing system allows application programmers to concentrate on writing code that supports the business by shielding application programs from the details of transaction management:

- It manages the concurrent processing of transactions.
- It enables the sharing of data.
- It ensures the integrity of data.
- It manages the prioritization of transaction execution.

When a transaction starts processing, CICS runs a program that is associated with the transaction. That program can transfer control to other programs in the course of the transaction, making it possible to assemble modular applications consisting of many CICS programs.

At any time, in a CICS system, many instances of a transaction can run at the same time. A single instance of a running transaction is known as a *task*.

During the time that a task is running, it has exclusive use of (or holds a lock for) each data resource that it changes, ensuring the isolation property of the transaction. Other tasks that require access to the resources must wait until the lock is released. To ensure overall responsiveness of the transaction processing system, design your CICS application programs so that they hold locks for the shortest possible time. For example, you can design your transactions to be short-lived because CICS releases locks when a task ends.

1.2.1 Why CICS

CICS handles billions of transactions a week. In Martin Campbell-Kelly's history of the software industry, *From Airline Reservations to Sonic the Hedgehog*[1], he states that, "Although most people are blissfully unaware of CICS, they probably make use of it several times a week for almost every commercial electronic transaction they make."

[1] *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry*, Martin Campbell-Kelly

CICS Transaction Server continues to be one of the most scalable, secure, and reliable application environments because of its deep integration with the underlying z/OS operating system. This ability to execute large numbers of concurrent transactions, in a consistent and efficient manner, is at the center of the past success of CICS. It is equally important as an enabler of the horizontal business integration requirements that the banking industry faces today.

The value of CICS is discussed in depth in the IBM white paper *Why to chose CICS Transaction Server for new IT projects*[2]. In this paper, the authors outline why, even after forty years of existence, CICS remains an excellent choice for the IBM customers in the banking sector and in other sectors.

Business transactions (particularly banking transactions) tend to be short and repetitive. Common characteristics include concurrent access by many users, through multiple interfaces, to shared information without compromising data integrity. Some examples of such financial transactions include a customer information inquiry, a financial transfer, a Point of Sale (PoS) transaction, an insurance claim, stock trading, a credit check, or any number of similar operations. Because processing these transactions represents a direct cost to the business, the cost for each transaction must be minimized.

It is possible to write stand-alone applications to perform these tasks in shared, multiple-user environments; however, it is complex to do so because application developers must address such issues as the communications with the user interface, ensuring that only authorized users access the system, preventing concurrent changes, handling failures and backing out partial updates, managing the database connectivity, and numerous other complicated tasks, each of which must be rewritten for each new application. All of this is extra and unnecessary work in addition to writing the actual business logic that is required to process the business transaction.

CICS takes the load off application programmers by taking care of the application's non-functional requirements, such as transactionality, security, and so on. It does this for applications that are written in a variety of languages including COBOL, PL/I, C/C++, and Java. It can process complex and demanding application workloads with optimal performance. It provides a rich set of functions for application developers and system administrators that enable them to build and manage highly efficient applications that can use standard interfaces, such as database queries, Web services, Java connectivity, and WebSphere MQ messaging.

[2] *Why to choose CICS Transaction Server for new IT projects*, Andrew Bates and Timothy Sipples, September 2008.

CICS has sophisticated virtualization capabilities that integrate with the System z Parallel Sysplex® technology. These capabilities deliver the almost unlimited levels of scalability that are required to cost-effectively manage business growth.

A key question today is whether JEE servers are better placed than traditional TP monitors, such as CICS, for running banking workloads. For various banks this is an attractive proposition because it allows them to create a common processing environment for all applications, whether they are mission critical core banking applications or less critical applications. However, most medium-to-large sized banks are choosing to maintain their TP monitors and are opting for mixed IT environments that integrate across their existing CICS-based banking systems and new JEE implementations.

CICS usage continues to grow across all industries, including the finance industry, and IBM continues to invest in CICS to ensure that it takes advantage of new technologies when they become available and when the technology fits well with CICS.

1.3 CICS applications

An *application* is a collection of programs that together perform a business operation. Each program runs as part of a transaction under CICS control using CICS services and interfaces to access resources. You can write CICS application programs in COBOL, C, C++, PL/I, Java, Assembler, REXX, and PHP language. For more details about these languages in CICS see Chapter 9, “Application development” on page 281.

All CICS programs run as part of a transaction, and all work that the applications perform is associated with a transaction identifier (transaction ID).

A CICS transaction can be started in several ways, for example:

- An individual can enter a transaction identifier (transaction ID) from one of a variety of terminals and network-connected devices, including the IBM 3270 Display System and emulator programs.
- Another system connected in a peer-to-peer configuration or as the client in a client/server configuration can request a named transaction or a program.
- A CICS program can start another transaction to be run immediately or some time later.

The CICS application programming interface

In CICS programs, most of the processing logic is expressed in standard language statements, but to access program resources that are managed by

CICS and to request CICS services you use the CICS application programming interface (API).

CICS services for application programs

The CICS application programming interface (API) provides access to a rich set of CICS services that you can use to construct sophisticated applications that consist of many interacting programs.

Event processing

Business events that occur in CICS applications can be captured and consumed by other CICS applications or processed by a complex event processing engine, such as IBM WebSphere Business Events. You can use event processing in many ways, such as for detecting customer trends or for detecting abnormalities in patterns of customer behavior that might indicate irregular or fraudulent situations. For more details about events in CICS see 6.4.1, “Introduction to Event Processing” on page 216.

CICS data objects

CICS provides a number of objects that CICS programs and transactions can use to exchange data with one another.

Application architecture

You can construct CICS applications using a modular architecture in which the main functions of the application are preformed by distinct components. You can develop applications that conform to the Service Component Architecture (SCA) specifications, enabling rapid deployment of new applications to meet changing business requirements.

1.4 Access to external data

CICS supports access to external data in DB2 and DL/I databases and in VSAM and BDAM data sets. For more details about CICS Data, see Chapter 3, “CICS data” on page 103.

Access to DB2 databases

CICS programs can use SQL statements to access data in DB2 databases. Access to DB2 uses the CICS DB2 attachment facility.

Access to VSAM and BDAM data sets

CICS programs can use the CICS application programming interface to read and write data stored in data sets that are managed by VSAM (virtual storage access method) and BDAM (basic direct access method).

An application program does not refer directly to a data set; it refers to a file, which is a CICS resource that is mapped to a data set by a resource definition.

You can use shared data tables to provide efficient access to VSAM key-sequenced data sets. A data table is a file that has records that are held in main storage. A shared data table is a data table that is accessible to more than one CICS region.

Access to DL/I databases

CICS programs can use the DL/I command-level interface and the call level-interface to access data in DL/I databases. Access to DL/I databases uses Database Control (DBCTL).

You can distribute CICS applications and the resources they use between interconnected CICS regions. You can group CICS regions into CICS system groups and CICSplexes and distribute regions across the z/OS systems in a sysplex.

1.5 CICS topology

You can distribute CICS applications and the resources they use between interconnected CICS regions. You can group CICS regions into CICS system groups and CICSplexes and distribute regions across the z/OS systems in a sysplex. For more details about topologies, see Chapter 10, “CICS topology” on page 357

The terms that are used when discussing CICS topology are:

Sysplex A set of z/OS systems that communicate and cooperate with each other through multisystem hardware components and software services.

CICS region A named instance of CICS Transaction Server that runs in its own z/OS address space. A CICS region can be started and stopped independently of other CICS regions.

CICSplex A grouping of CICS regions that is managed as a single entity. Each CICS region can belong to one CICSplex

only. A CICSplex can include CICS regions running on several z/OS systems in a sysplex.

CICS system group A grouping of CICS regions in a CICSplex that can be managed as a single entity. A system group can include CICS regions running on several z/OS systems. In a CICSplex, each CICS region can belong to more than one system group, and system groups can be contained in other system groups.

In a CICSplex that contains many CICS regions, it is convenient to assign regions to perform particular roles for the applications that they support.

CICS provides a number of methods for communicating between CICS regions in the same z/OS system, between regions in the same z/OS sysplex, and between regions in different sysplexes.

1.6 CICS configuration interfaces

You can configure your CICS region and control its behavior by specifying system initialization parameters, by creating and installing resource definitions, and by writing user exits and user-replaceable programs.

1.6.1 System initialization parameters

System initialization parameters specify the initial configuration and capabilities of a CICS region. The things you can specify in system initialization parameters are:

- The unique identity of the CICS region
- The major CICS functions that you want to include in your initial configuration
- The names of user-replaceable programs that you want to include in your configuration
- The upper limits on the amounts of system storage that the CICS region can use

You can change some, but not all, of the values that are specified in system initialization parameters when the CICS region completes its initialization.

Resource definitions

A resource definition is an object that specifies the attributes of resources that are used directly, and indirectly, by applications. Most resource attributes,

including their location in a CICSPlex, are hidden from the programs that use them, so you can reconfigure an application in a CICSPlex without changing the application programs. The things you can specify in resource definitions are:

- The attributes of multiregion operation (MRO), intersystem communication (ISC), and IP interconnectivity (IPIC) connections with other CICS regions
- The attributes of transactions and programs in the CICS region
- The attributes of data resources used by CICS programs, such as files, temporary storage queues, and transient data queues
- The attributes of terminals and network devices that initiate transactions in CICS
- The attributes of the connections between CICS and other resource managers, such as DB2

Creating resource definitions and then installing them so that they become active in a CICS region are two distinct operations:

- You can create a resource definition even when the CICS region that will use it is not running.
- You can install a resource definition when a region starts or when the region is running.
- You can remove resource definitions from a CICS region when they are no longer needed.

You can change some, but not all, attributes of a resource after it has been installed in a CICS region.

Initialization and shutdown programs

You can add your own processing to CICS initialization and shutdown by writing initialization and shutdown programs. Initialization and shutdown programs use the CICS application programming interface (API), but not all functions of the API are available.

Initialization programs run in two phases. During the first phase, because CICS initialization is not complete, the actions that you can perform in your initialization programs are limited to enabling user exit programs. During the second phase, CICS initialization is complete, and most CICS services are available.

Likewise, shutdown programs run in two phases. During the first phase, most CICS services are available. During the second phase the actions you can perform are more limited.

User exits and user-replaceable programs

You can control most aspects of the behavior of a CICS region using system initialization parameters and resource definitions. However, sometimes you need a greater degree of control, and for these situations, CICS provides user exits and user-replaceable programs.

Global user exit

A global user exit (GLUE) is a point in a CICS system program at which CICS can pass control to a user-written program (also called an exit program). You can enable and disable global user exits when CICS is running. When an exit is enabled, the user-written program is called each time the exit point is reached.

Task-related user exit

A task-related user exit (TRUE) is a point in a CICS application program at which the program passes control to a user-written program. Task-related user exits allow you to write your own program to access resource managers that are not directly supported by CICS. Task-related user exits are triggered by commands in the application program that request access to the resource. For more details, see 2.6.2, “Task-related user exits” on page 74.

User-replaceable program

A user-replaceable program is a program supplied as part of CICS that you can replace with a version that modifies the actions performed by the original version.

1.7 Security and CICS

CICS uses the services provided by an external security manager, such as RACF®, to control access to application and system resources. For more details about CICS and security, see 2.7, “CICS security” on page 76.

A transaction can be initiated in CICS by an individual, by a program running in a system connected to CICS through a communication network, or by another transaction running in CICS. In all cases, when CICS security is active, the transaction is associated with a user. Although a CICS user is, in many cases, an individual, in general a user is an entity that is identified by a user identifier (user ID).

When a user makes a request, CICS calls the external security manager to determine if the user has the authority to make the request. If the user does not have the correct authority, CICS denies the request.

CICS checks the user's authority at several levels:

Signon security Ensures that terminal users are entitled to connect to CICS.

Link security Ensures that remote systems are entitled to connect to CICS.

Transaction security Ensures that users that attempt to run a transaction are entitled to do so.

Resource security Ensures that users who use CICS resources are entitled to do so.

Command security Ensures that users who use the CICS system programming interface are entitled to do so.

CICS can use user IDs and passwords, password phrases, or Secure Sockets Layer (SSL) client certificates to perform identification and authentication of users.

1.8 Systems management programming interfaces

CICS Transaction Server provides a family of programming interfaces that you can use to develop programs that manage your CICS regions and resources.

The systems management programming interfaces are the:

- CICS management client interface
- CICSPlex SM application programming interface
- CICS system programming interface

1.8.1 The CICS management client interface

The CICS management client interface is designed for use by HTTP client applications (including the IBM CICS Explorer). You can use this interface to develop HTTP client applications that manage installed and definitional CICS resources on regions that are managed by CICSPlex SM. On stand-alone CICS regions, you can use the interface with operational resources.

1.8.2 The CICSPlex SM application programming interface

Using the CICSPlex SM application programming interface (API) you can write programs that monitor and control CICS regions and CICSPlex SM itself.

Programs that use this API can run in a z/OS batch or TSO environment, in a Tivoli® NetView® for z/OS environment, or in CICS itself.

1.8.3 The CICS system programming interface

The CICS system programming interface (SPI) is a set of commands that extends the CICS application programming interface. Using the CICS SPI you can work with resources in a CICS region from a CICS program.

1.9 The evolution of CICS

When CICS was first released in 1969, it could handle 50 Basic Telecommunications Access Method (BTAM) terminals, three file data sets, 100 programs, 50 transaction types, 50 queues, and required anywhere from 15,000 bytes up to 35,000 bytes (15 KB and 35 KB, respectively) of storage. CICS today can handle more than 900,000 concurrent users and an average CICS installation handles around 250 million transactions per day. The major milestones in the evolution of CICS are explained in the following sections. Later chapters describe some of the features that are listed in more detail.

Figure 1-1 on page 15 shows the CICS time line.

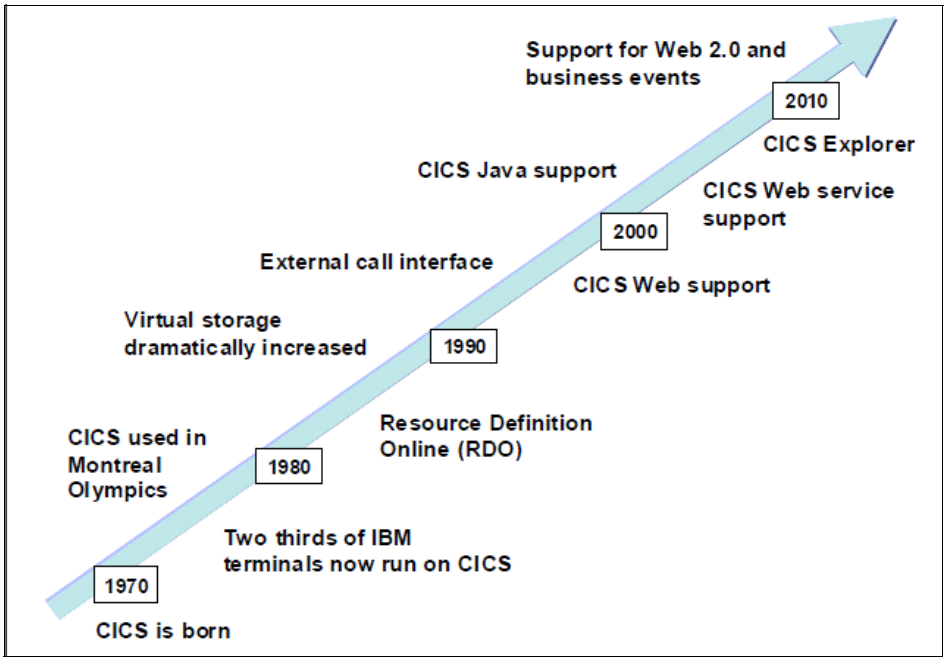


Figure 1-1 CICS time line

1.9.1 The birth of CICS

1969 was a great time for the world and for IBM. Events, such as Woodstock, the maiden flight of Concord, and the first man on the Moon, changed world culture and history. Most important, the IBM Customer Information Control System (CICS) was released, available at the low price of $600 dollars per month. The foundation of CICS however, started over a decade earlier in the 1950s with American Airlines and the IBM Semi Automatic Business Environment Research (SABER) project. The SABER project provided the first specialized terminals for airline reservations and was the foundation for transaction processing facility (TPF).

The major utility companies in the United States were well aware of the need to find a way to manage customer's data efficiently in a time of expansion. The public utility companies looked to the airlines who used TPF and central data repositories to manage their businesses and liked the idea of a central repository for all of their customer records. Various utility companies tried to solve this issue by developing their own solutions. Typically, they assembled teams of application programmers familiar with the industry or systems programmers familiar with access methods, such as BTAM and the operating system (which at the time was

OS/MFT). These in-house transaction processing systems were developed at significant resource cost without much success.

Someone within IBM who worked with the utility industry noticed the development of several programs like this and decided it would be better to have one application written for all of these companies. That someone was systems programmer Ben Riggins, the "Father of CICS". He was assigned to work with the Virginia Light and Power Company of Richmond, Virginia. The power companies had a vision to create an *online* information control system. Riggins was eager to help them make this vision a reality.

The initial purpose of the newly developed system was to be a transaction processor for the utility industry, which would help them avoid the duplication and the proliferation of repositories containing customer data. The American business emphasis on service to the customer meant that service levels and responsiveness were of prime importance. The vast amounts of data, which they already had, was typically being processed in long, sequential batches preventing it from being used to provide better service to their customers.

Ben Riggins, Jerry Hughes, and Ray Vander Vliet comprised the first development team for CICS, managed by Jerry Anderson. This small team moved to Palo Alto, California and developed the first operating CICS system, which provided terminal access, file control, communications access, and the ability to work with the different operating systems of the System/360.

In April 1968, IBM released Public Utility Customer Information Control System (PUCICS). It was free code, known as a Type II program, which was originally created to address the needs of a few companies; however the industry-wide, almost immediate demand for PUCICS showed that almost every other kind of organization across a multitude of industries had the same needs. The wide demand for the transaction processing capabilities provided by this new system prompted IBM to drop the “Public Utility” part of the name and officially release the product known as CICS on July 8th, 1969.

1.9.2 1970 to 1979

In the 1970s, CICS became a major force in the banking, insurance, and securities industries. It was used for applications focusing on check clearing, customer record database updates, and orders.

In 1974, CICS development was moved to Hursley in the United Kingdom. The Hursley programming center assumed the world wide CICS mission, taking global responsibility for CICS, which was at the time, and still is today, one of the leading program products in IBM. To keep CICS viable after the move, Hursley chose not to use the newest technologies on the market to support CICS but

rather to use the best technologies available at the time. This was a major factor in the long term success of CICS.

1974 was also the year that CICS Virtual Storage (CICS/VS) Version 1.0, was released, which introduced single address spacing and management functions. A year later, in 1975, IBM determined that over 30% of all terminals world wide were running CICS.

In 1976, CICS/VS Version 1.3 was released, which introduced the command level and recovery and restart functions. The command level API was an important new capability. It was added to CICS in the shape of High Level Programming Interface (HLPI), allowing access to CICS services and resources more easily, which meant CICS applications and IT infrastructure upgrades were completed in a fraction of the time.

In 1978, intersystem communication (ISC) was introduced when CICS/VS Version 1.4 was released.

1.9.3 1980 to 1989

CICS/VS Version 1.5 was released in 1980, which introduced multi-region operation (MRO).

Hursley changed the CICS architecture in 1983 and introduced Resource Definition Online (RDO) as part of the CICS/VS Version 1.6 release. Transactions (PCT entries) and programs (PPT entries) could then be defined in real-time without the need to recycle the CICS region to bring in changes. This was also the release of CICS, which provided support for XA functionality thereby utilizing storage above the 16MB line for the first time. This flexibility enabled a single CICS implementation to support multiple applications. The result was that users could, and still can, run CICS routinely for long periods without error.

In 1986, CICS/VS Version 1.7 was released, which added RDO support for VTAM® terminals. In addition, it automatically installed these terminals using the new AUTOINSTALL feature. This was also the version of CICS where IBM no longer shipped the source code for CICS; therefore, CICS became "OCO only".

There was a release of CICS Multiple Virtual Storage (CICS/MVS) in 1987, Version 2.1, which bought some performance improvements and introduced Extended Recovery Facility (XRF).

1989 was the year of the CICS Enterprise Systems Architecture (CICS/ESA) Version 3.1 release. This was the first time that CICS was largely rewritten. Many of the old functions were dropped in preference for a totally new architecture that

introduced the new CICS domain structure, which IBM continually built upon since then.

1.9.4 1990 to 1999

CICS/ESA Version 3.2, released in 1991, dropped more of the functionality of earlier releases, including Macro Level support for all languages, and the inbuilt CICS security feature was eliminated. All CICS security functions were then handled by an External Security Manager, such as RACF.

In 1992, CICS/ESA Version 3.3 was released providing a storage protection feature for CICS/ESA domain storage. It also introduced the Dynamic Link Function, Shared Data Tables, Distributed Program Link (DPL), and the front-end programming interface (FEPI).

CICS/ESA Version 4.1 was released in 1994. This new release introduced dynamic transaction routing among parallel transaction server machines, and a new external call interface, allowing a CICS program to be called from platforms other than the mainframe. Storage protection was enhanced further in this release with the addition of Transaction Isolation.

On the VSE operating system, in 1995, CICS/VSE Version 2.3 was released, providing enhanced programmer interfaces that allowed for the replacement of macro level calls. Support for Language Environment® for VSE/ESA and associated languages were made available. There was also the introduction of a tool to aid customers in migrating from the use of internal CICS security to an external security manager (ESM).

CICS Transaction Server (TS) for OS/390® Version 1.1 was released in 1996, which created a new domain for logging and journaling, recovery, and temporary storage. Resource Definition Online (RDO) for transient data destinations was introduced, and there was also the addition of the EXEC CICS CREATE command for the creation of CICS resource definitions. This feature enabled the sysplex-wide management of CICS regions using CICSPlex SM (CPSM). The web service capabilities introduced by CICS TS started a shift from traditional green screen applications to web-based applications. This transition allowed e-business and e-commerce to be a reality. Stocks and shares were now being traded over the Internet with connections to CICS at the back end.

In 1997, CICS TS for OS/390 Version 1.2 was released, introducing DASDONLY logging.

CICS TS for OS/390 Version 1.3 was released in 1998 providing a number of improvements to the functionality of CICS. RDO for temporary storage, parallel sysplex support, coupling facility data tables, the Open Transaction Environment

(OTE), and CICS Business Transaction Services (BTS) were just a few of the new features added to CICS. Support for the Java Virtual Machine (JVM) was also added, as was support for inbound IIOP requests to Java application programs.

IBM products, such as CICS Internet Gateway and the CICS Gateway for Java were part of the CICS Universal Client, and later the CICS Common Client Version 1. These products were available under the CICS Transaction Server umbrella since it was released in the mid 1990s. As these smaller products continued to grow, in 1998, CICS Transaction Gateway was released as a technology preview with Version 2 of the CICS Common Client. CICS Transaction Gateway Version 3.0 was officially released in 1998 as a separate product to CICS Transaction Server, and Version 3.1, released in 1999, was the first release to support z/OS.

1.9.5 2000 to 2011

In 2000, CICS TS for VSE/ESA Version 1.1.1 was released to give CICS new and improved features, including:

- Expanded application programing support, including a new External CICS Interface (EXCI) and a Front End Programming Interface (FEPI).
- System management enhancements, including RDO for files.
- The CICS Web Interface (CWI), which provided direct access to CICS applications from the world wide web.
- Shared data tables that improved performance and availability.
- The CICS 3270 Bridge, which provides an interface to run 3270-based CICS transactions without a 3270 terminal.
- Extensive virtual storage constraint release.

In 2001, CICS TS for z/OS Version 2.1 provided the following features and enhancements:

- Support for the industry-standard Enterprise Java Beans (EJB) architecture, enabling CICS to act as an EJB server.
- Facilities for the generation of new EJB application, or for the reuse or incorporation of existing applications and data in an EJB solution.
- Enhancements to the facilities for network connectivity in support of e-business, and to Java programming under CICS.
- Extensions to facilities for applications based on procedural programming models.
- Significant extensions to CICSPlex SM.

2003 saw the release of CICS TS for z/OS Version 2.3, which provided a number of new and improved features, such as:

- Enhanced Java capability, including performance improvements, support for Java SDK 1.4, the Common Client Interface (CCI) connector for CICS TS, and the addition of web support to the JCICS classes.
- Improved support for development of applications, including interactive debugging.
- Further connectivity options, including new security functions and the recently introduced SOAP.
- Enhancements in the area of availability, including workload balancing of the 3270 Bridge interface using CICSPlex SM.
- Important performance improvements to the CICS-DB2 attachment.

In 2005, CICS TS for z/OS Version 3.1 was released and provided the following enhancements and new features:

- Major new capabilities in the areas of web services, HTTP function, security,
- Extension of the OTE.
- Eclipse based Information Centers.
- Support for Language Environment-enabled Assembler applications.
- Threadsafe CICS web API commands.
- Enhanced C and C++ capability by support for XPLink

CICS TS for z/OS Version 3.2, released in 2007, fulfilled more than 80 customer requirements and provided the following enhancements and new features:

- Improved application connectivity, web service standards, interoperability profiles, and data mapping are introduced along with intercommunications over TCP/IP.
- Application reuse through Java enhancements, CICS integrated translator support for C, and C++, and 64-bit storage for CONTAINER data.
- Architectural enhancements that brought threadsafe core APIs for accessing VSAM files, journals, WebSphere MQ, and enhancements to trace, monitoring, and statistic capabilities.

In 2009, CICS TS for z/OS Version 4.1 was released and provided the following enhancements and new features in response to more than 60 customer requirements:

- Atom-feeds and RESTful interfaces that enable existing CICS programs and resources to participate in mashups, Web 2.0, and other situational applications.
- The CICS Explorer, a graphical user interface, is launched to simplify the development and management of traditional and modern applications. The

main benefit of the CICS Explorer is that it provides a platform for integration of CICS tools and applications that a user needs to perform business tasks. CICS Explorer represents a shift from the traditional green screen mainframe interface to the point-and-click GUI that the new generation of computer programmers have become accustomed to.

- Non-invasive detection and emission of business events from CICS applications providing a whole new insight into business behavior.

In 2011, CICS TS for z/OS Version 4.2 was released providing the following new features and improvements:

- System events and improvements to existing event processing capabilities, such as assured event emission, that specify that certain CICS events are critical and that their emission is assured. EPADAPTER resource enhancements providing the ability to choose between two predefined EPADAPTER resources making it easier to share configurations between event bindings and less complicated to manage. Other improvements to event processing are improved search facility, additional data types, and support for the temporary storage queue (TSQ) EP adapter for XML event formats, common base event REST, and WebSphere Business Events (WBE).
- A new Java runtime environment, multi-threaded Java applications, software developer toolkit for Java and OSGi deployment and management of Java programs.
- A new CICS transaction tracking facility to simplify system operation tasks, 100-character password phrases to improve CICS security and enhance usability, workload management improvements to control routing behavior at the transaction level, and the use of 64-bit storage for main temporary storage.
- Support for web services discovery, enhancements to function shipping to enable the shipping of file control, transient data, and temporary storage requests over IP interconnectivity (IPIC), enhancements to CICS DB2 thread reuse, and enhancements to Atom feeds.
- Enhancements to scalability with major extensions to multiprocessor exploitation, the use of 64-bit storage increasing the virtual storage available, provision of an interface to enhance CICS IMS database control (DBCTL), and more areas of CICS being made threadsafe to increase throughput and reduce processor usage.
- Improvements to the CICS Explorer improvements, including:
    - Improved support to transaction tracking
    - New workload management views
    - CICS Explorer SDK support for creating Java OSGi applications
    - Support for system events in the CICS event binding editor

Many more new features and improvements are available with CICS Transaction Server Version 4.2 and the CICS Explorer. The capabilities we previously listed are a highlight of functions that were introduced to improve the performance, usability, and scalability of CICS.

1.10 CICS today

With each release of CICS, the features included with it increased greatly. CICS now offers support for SOA, SOAP, and Extensible Markup Language (XML). Since 2001, CICS has been able to use Java programs in the production environment, which helped open it to a whole new generation of programmers. Today, CICS can generate XML atom feeds to populate Web 2.0 mashups quickly and efficiently. The use of CICS-WebSphere MQ interface was enhanced to enable the use of queue-sharing groups rather than single queue managers to increase reliability and standardization.

In the past, management and administration of CICS systems was achieved by issuing CICS commands from within a CICS region or by utilizing the Interactive System Productivity Facility (ISPF) based panels provided by CICSPlex SM. In 2009, CICS introduced the first version of The CICS Explorer, "The New Face of CICS", which was designed to simplify the development and management of traditional and modern applications within CICS. It also provided an integration point for CICS Transaction Server, CICS Tools, CICS Transaction Gateway, PD Tools, and Rational® Tools. The CICS Explorer provides a common, intuitive, Eclipse-based environment for architects, developers, administrators, system programmers, and operators to manage their CICS environment. Some of these tools are explained later in Chapter 6, “CICS Explorer” on page 205, Chapter 7, “CICS tools” on page 241, and Chapter 8, “Introduction to the IBM Problem Determination tools” on page 259.

1.10.1 What is new

CICS Transaction Server for z/OS, Version 4 Release 2 delivers a set of capabilities that provide customer value by enabling business flexibility through IT simplification. These capabilities are represented in several themes: Events, Java, Connectivity, Management, and Scalability.

Events

The Events theme provides the capability for system events to be emitted using the CICS business event infrastructure, providing monitoring information about the CICS system. Enhanced CICS bundle support reduces set up time and complexity by separating system programmer and application developer

activities with common technology for both system and application-related events.

Java

The Java theme provides a new 64-bit Java runtime environment with enhancements to the multithreaded Java Virtual Machine (JVM) server infrastructure. This environment allows CICS to run a greater number of larger Java applications in a single region with better scalability for Java workloads in CICS. Combined with new support for the deployment of OSGi bundles, you can significantly simplify system administration and portability for Java applications. The provision of a new Java-based Axis2 web services engine improves portability of web services applications. All Java workloads are eligible to run on a zAAP.

Connectivity

The Connectivity theme provides an extension to System z IP networking and supports the function shipping of file control, transient data, and temporary storage requests between CICS regions over a TCP/IP network using an IP interconnectivity (IPIC) connection. Potential benefits include improved performance and the ability to simplify network administration when moving from SNA to an IP network.

Management

The Management theme provides enhancements to simplify the management of CICS, including a new transaction tracking facility, improvements to workload management, support for password phrases to enhance log-on security, enhancements to the CICS Explorer, and the ability to view system initialization parameters.

Scalability

The Scalability theme provides enhanced scalability of the CICS environment, with major extensions to multiprocessor exploitation, including threadsafe access through the CICS-DBCTL interface to IBM Information Management System (IMS) Databases and function shipping requests between CICS systems over IPIC connections.

1.10.2 CICS and world commerce

At over 40 years old, CICS is still maturing but retains the unique architecture that made it the best purpose built transaction processor in the world. CICS and CICS Tools are developed primarily in the United Kingdom but are also

developed in the United States of America, Australia, China, Germany, India, Israel, and Russia. Development sites are shown in red on the CICS world map in Figure 1-2. Although there are relatively few development sites, CICS is used in over 80 countries world wide including those in the Americas, Europe, Africa, Asia, and Australia, which are shown in black on the CICS world map.

Figure 1-2 The world of CICS

The creators of CICS built it with the most advanced technology available at that time and their successors followed this example, evolving CICS alongside ground breaking and innovative technologies for over 40 years.

With thousands of CICS mainframe licenses world wide that process 30 billion transactions per day, it provides a foundation for world commerce. Organizations from every area of industry and commerce use and rely on CICS transaction processing. Some of the industries that use CICS are shown in Table 1-1.

Table 1-1 Industries that use CICS

Agriculture	Architecture
Chemical	Construction
Education	Financial services
Government	Insurance
Manufacturing	Media and broadcasting
Medical and pharmaceutical	Military
Motoring	Oil, gas, and coal

Real estate	Research and development
Retail	Shipping and transport
Telecommunications	Travel

Almost all commercial organizations have the need for real-time transaction processing. Without the help of systems, such as CICS, these organizations would not be able to compete in today's market. Possibly its highest profile deployment is in the financial services industry. CICS is extensively used in organizations, including banks, credit card providers, insurance companies, stock brokerages, investment funds, government-sponsored enterprises, and consumer finance companies.

The application programming facilities of CICS make it broader than many alternate solutions and enable it to be adapted for use in a variety of business situations. The universal usage of CICS and its ability to securely manage so many transactions simultaneously led directly to fundamental improvements in information communication, the growth of industry and commerce, and globalization. Despite this, CICS is still relatively anonymous even though it is at the heart of our daily life. CICS is there when you buy your lunch, call your electricity provider to request an engineer visit, order a new book from an online store, or book a flight. It is for this reason that CICS is described as probably the most successful piece of software of all time with millions of users unknowingly running CICS transactions everyday, and that if it were to disappear the world economy would cease functioning.

1.11 The future of CICS

When looking to the future of CICS in the world of information technology, we must look at how the marketplace has evolved and is expected to evolve, and how the users of CICS will change.

CICS has always been associated with mainframe systems, and the demise of the mainframe computer was predicted for many years by industry watchers. IBM, along with many customers, believed that this way of computing—serious, secure, and industrial-strength—would always be in demand. The benefits of constant availability, ironclad security, and massive computing power stimulated increased sales for this type of computing in the early 1990s, which was a time when critics said that it would cease to exist. CICS grew into a family of application servers, spanning a variety of platforms and offering business solutions for all sizes of organizations. While introducing entry-level systems at the low end of the marketplace, it has kept pace with technological developments in the mainframe world.

Cloud computing is a relatively new concept, it was first discussed in the late 1990s, but in depth research was not carried out on cloud computing by the larger information technology companies until the mid 2000s. The term “cloud” refers to the resources, or services, provided on demand from a network. These resources, or services are often abstracted, that is, they are simplified so that much of the details and inner working are not shown to the users. The users can be anyone who requests services or resources, including developers. Cloud computing eliminates the need for these users to access information and create applications without requiring knowledge of the physical location and configuration of the system that delivers the services. CICS is remarkably similar to cloud computing in that the end users of CICS do not need to understand how CICS works to access the information that they require, in fact many end users are not even aware that they are using CICS at all. Similarly, to develop new applications, developers do not need to have a full understanding of the internal workings of CICS transaction server. CICS has had many characteristics similar to cloud computing for the last 40 years, but that is not to say that the developers of CICS Transaction Server are not looking at what advances in cloud computing can offer CICS.

The use of computerized transaction processing has grown steadily across the world and throughout all segments of industry. In the 1980s, hardware advances made powerful systems accessible to smaller businesses. Before this, only the large and medium-sized organizations could afford the large-scale computerized systems that could handle their processing and communication workloads. Now we see that tens of thousands of smaller CICS systems outnumber the established large scale-host based system.

It is not only the future direction of organizations that will shape the evolution of CICS. The nature and characteristics of the users of CICS will also play an important role in how this software is developed for the future. The end user has gone through an evolution. The increased use of telecommunications technology, such as satellite and cable communications, made the world “smaller”. Users can work from home, from the office, and from site locations around the world. The user can be at their workstation holding a video conference with shared use of data and access to tools, such as notebooks, that all users can add information to. This is the perfect setting for CICS, which can process participants’ inquiries and provide results that can be shared and discussed in real-time.

The past was shaped by communications technology and so the future will be too. The capability of CICS to communicate with a number of different systems brings a greater potential for customization, opening the door for more powerful applications to be built according to the needs of users.

CICS evolved over multiple decades, providing support on multiple operating system platforms, using a variety of traditional programming languages and all

the common forms of data management. CICS supports systems network architecture (SNA) terminal networks, TCP/IP sockets, and web input and output. It supports previously introduced capabilities, while expanding to support new technologies, such as the internet, Java/EJB, and increased performance through its open transaction environment (OTE) facilities.

Though CICS is continually evolving in this way, it is not a case of progress for the sake of progress. IBM has long recognized the need to engineer systems that offer greater interconnectivity and intelligence, yet the fundamental nature of CICS will not change. It will continue to support a large variety of terminal devices to provide controlled access to resources through a communication network. It will always be achieved under the control of application programs, and CICS will provide simplified programming interfaces to make things simpler for application programmers. It will continue to be a *CICS for everyone*; however many applications are added, an organization can still design, test, and run simple transactions on the largest or smallest systems. This is the purpose for which it was made and this ethos will stay with CICS into the future.

1.12 Example organizations

To help you better understand how CICS is used today, throughout the book we use two example organizations to demonstrate how and why features and functions of CICS are implemented.

The organizations used are a bank and a catalog sales store. Each organization was chosen because they are in different industry sectors, and, although they both use CICS to process business transactions, they use different CICS configurations according to their business needs.

1.12.1 Bank

The bank is a major multi-national financial institution with thousands of CICS users based at multiple sites. The bank has used CICS for 20 years and have multiple CICS systems.

The banking systems that are used in the bank run a mixed workload of real-world financial transactions, including cash withdrawals, deposits, mortgages, and car loans. The COBOL core banking application runs on CICS and stores customer and account records in DB2 for z/OS.

The bank environment:

- The bank is in the late stages of upgrading from CICS Transaction Server Version 3.2 to Version 4.1.
- CICS Transaction Server Version 4.1 is configured with:
  - Two terminal-owning regions (TOR), one per z/OS LPAR
  - Six application-owning regions (AOR), three per z/OS LPAR

The CICSPlex SM component of CICS TS Version 4.1 is used for CICSPlex Single System Image management, providing a single point-of-control and dynamic workload transaction balancing over the core banking CICSplex.

CICS Explorer is used throughout the bank.

They use IBM DB2 V9 for the operational core banking application data store and BDW, and it is configured with two DB2 instances in data sharing mode, one per z/OS LPAR.

- They use WebSphere MQ for z/OS V6 as the classical access layer for core banking and for publishing. It is configured with:
  - Two queue managers, one per z/OS LPAR
  - Two channel initiators, one per z/OS LPAR
- CICS Transaction Gateway (CICS TG) V6.0.1 is used for JEE connectivity. It is configured with two CICS TG daemons, one per z/OS LPAR.

As well as operating the existing organization, the bank is constantly acquiring smaller banks and other financial institutions.

1.12.2 Catalog sales store

The catalog sales store is a bricks-and-mortar store that is moving to be a purely online catalog sales store. They have an existing, simple CICS environment, but are moving to use MRO, CICSPlex SM, and CICS Explorer.

The store also wants to web-enable their business so that customers can purchase items world wide.

Part 2

CICS fundamentals

In this part of the book, we cover the following areas:

- CICS Facilities
- CICS Data
- CICS access technologies

2

CICS facilities

In this chapter, we describe the facilities and services that CICS provide. Unlike the application programming interface that is used by the application programmer, we discuss the CICS facilities and services that the CICS system programmer must understand.

This chapter explains how the CICS system programming staff can use CICS facilities and services to customize the system, design CICS for recovery, and manage performance.

2.1 System Initialization

In this section, we review all areas of CICS systems initialization

2.1.1 CICS Initialization Process

Figure 2-1 illustrates an overview of CICS startup.

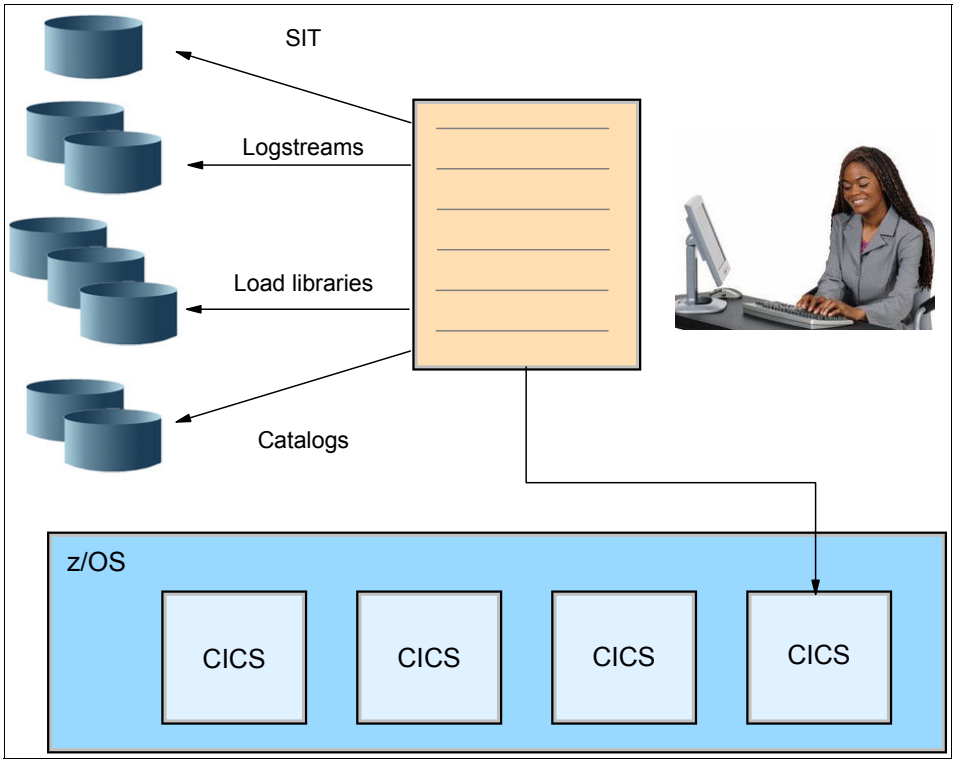


Figure 2-1 Overview CICS startup

2.1.2 CICS address space initialization

In this section, we discuss fundamental functions of the CICS initialization process.

Note: We use the term region throughout the section. An address space that runs CICS is actually referred to as a CICS region.

Our bank runs a fail-safe CICS high-availability environment. All regions are mirrored between 2 Logical partitions of z hardware (LPARs), hence there is no single point-of-failure. The bank's CICS applications can run on multiple application owning regions AOR. This is a major strength for critical applications such as our bank. The bank provides round-the-clock service; therefore, CICS is usually running continuously as long as the service is required for the end users. CICS actually can run 24 hours a day all year long. Therefore when we talk about CICS initialization, we must consider how to schedule the time frame to restart CICS in production. There are various situations that lead to CICS startup or restart:

- You are starting a new region for the first time. In this case you must start CICS initially. Recovery information is ignored because of course there is not any. You can use the System Initialization Table (SIT) parameter START=INITIAL to request an initial start of the CICS region.
- You want to start CICS after applying maintenance. Many organizations run their CICS regions continuously, but they have scheduled maintenance windows. CICS regions eventually must then be restarted after maintenance is applied. The required type of CICS system restart depends on the maintenance that is applied to CICS. Necessary actions, including the type of the CICS system restart, are documented in accompanying maintenance material. In this situation, maintenance is applied to CICS test regions by CICS system programmers. During the maintenance process CICS system programmers start the regions manually, probably several times, until the applied updates work well. When the test phase completes the updated CICS load libraries can be started up in production during the maintenance window. CICS startup and shutdown in production is a standard procedure that is done by automated processes or manually by operation staff.
- You need to restart a CICS region after getting a CICS system failure. Problems of catastrophic nature can bring down CICS abnormally; therefore, you must restart CICS in a way that all business transactions will be set to a consistent state. A business transaction is a self-contained business deal. Some transactions involve a short conversation (for example, requesting availability and cost, or providing payment). Others involve multiple actions that take place over an extended period.The properties of transactions in an online system are described with the acronym ACID: Atomicity, Consistency, Isolation, and Durability. In case of a CICS system failure CICS must recover business transactions according to properties atomicity and durability. Property atomicity says that a transaction must execute completely or not at all. If any part of the execution fails, for example, an update to a file, all traces of the transaction must be removed. Property durability says that if a transaction completes successfully, any of the changes it made will survive a failure.

- You want to startup CICS after a normal shutdown. In this case, you terminated CICS using the provided controlled shutdown process. During the normal shutdown process CICS waits for the business transactions to come to a normal end. After that CICS creates a flattened copy of CICS resources tables that are loaded in storage. The flattened copy of the resources is then stored to the CICS catalog. CICS also notifies the catalog that the next startup can be a so-called warm startup. During the process of a warm CICS startup, the flattened copy of the CICS resources can be moved from the CICS catalog straight into storage rather than building the necessary resources from scratch.

In the next section, we describe several startup options that you can specify in the System initialization table (SIT). The next chapter 2.1.6, “CICS System Initialization Parameters” on page 35 provides further information about system initialization parameters.

2.1.3 Start Auto

If you specify the Start Auto value for the START system initialization parameter, CICS determines whether to perform an initial, cold, warm, or emergency start by inspecting information about the CICS global catalog: During the process of a normal CICS startup the CICS catalog is updated with information about the next CICS restart. For example, in the case of an operator initiated CICS cold or warm start, the next start must be an emergency startup. If after that CICS shuts down normally then the information about the catalog gets marked saying that the next startup can be a normal startup. During a CICS startup the “emergency restart needed” information is turned on because in the case of a serious failure, CICS cannot update the catalog anymore, thus the next restart is emergency. The restart information about the catalog is stored in the following control records:

- The recovery manager control record
- The recovery manager autostart override record.

START=AUTO is the normal mode of operation, with the choice of start being made by CICS automatically.

2.1.4 Start Initial

The new or first run of CICS has no reference to any previous run. The CICS catalog and system log are initialized, and all information in them is lost.

2.1.5 START Cold

The new run of CICS has limited reference to the previous run and uses the same global catalog and system log. In particular, re synchronization information needed by remote systems to re synchronize their units of work is preserved.

2.1.6 CICS System Initialization Parameters

Figure 2-2 shows an overview of system initialization parameters.

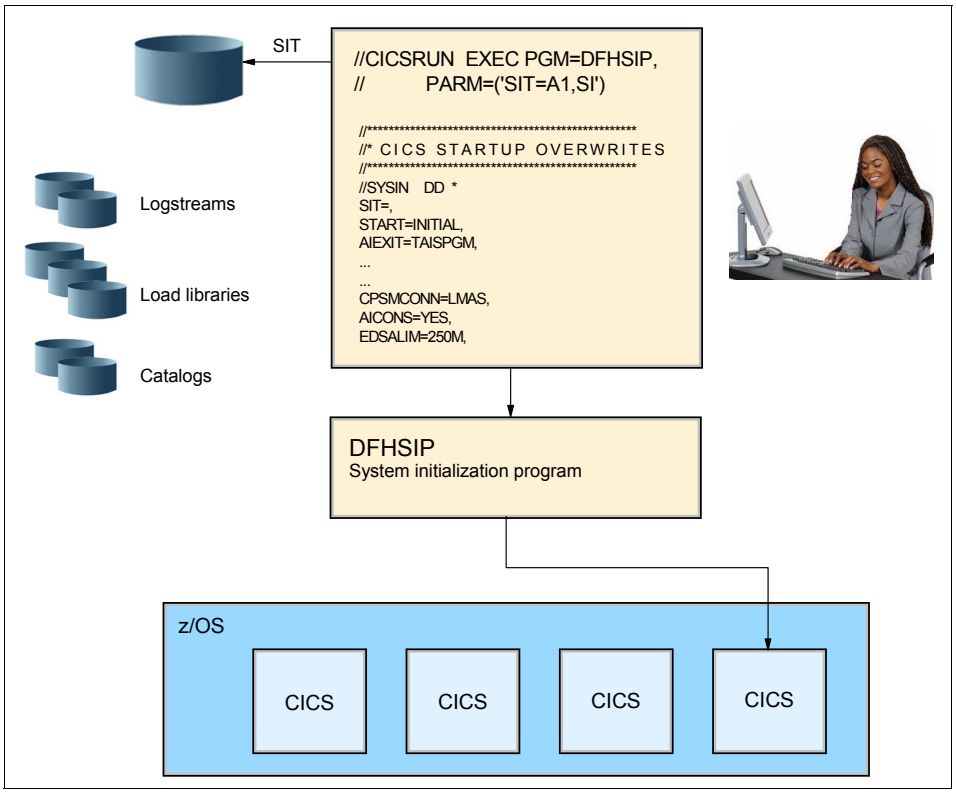


Figure 2-2 Overview system initialization parameters

In this section, we describe how our bank manages CICS system initialization parameters for their CICS startup procedures. The bank uses CICS regions with distinct assignments. We describe how the bank tailors the system initialization parameters to fit the needs of the different regions. The bank uses CICS regions that need to access DB2 or WebSphere MQ or alternately they use CICS regions like terminal owning regions that do not run applications at all.

2.1.7 Why we need system initialization parameters

You can control most of the CICS Facilities using the CICS system initialization parameters. System initialization parameters can be defined within the system initialization table, SYSIN data sets, and on the EXEC DFHSIP parm statement.

Note: A SYSIN job control statement points to a data set that contains system initialization parameters. Program DFHSIP is the CICS initialization program. It can take system initialization parameters on the EXEC PROGRAM DFHSIP job control statement.

Using the SIT parameters you can control CICS facilities and components, for example:

- The maximum tasks that can run concurrently in the CICS region.
- The size of Storage subpools that CICS and user applications can use.
- The virtual telecommunication access method (VTAM) Application ID which is unique for each CICS region. The VTAM APPLID identifies the CICS Region as VTAM Application in the network.
- You can activate/deactivate and manage CICS Tracing activity.
- You can manage CICS monitoring and statistics.
- You can specify the type of CICS startup.
- you can specify the directory for Java Virtual Machine (JVM) properties.

The basic idea of specifying system initialization parameters is to create a SIT macro table that contains all of the common parameters that can be used by all CICS regions that you use in your environment. CICS provides procedures to compile and link the SIT so that the CICS region can load it during the initialization process. After that you can use override parameters for each individual CICS region. Using the override parameters you can activate different CICS facilities, for example for test regions you might want to activate the CICS trace facility. For production regions you probably do not want the overhead of tracing, but you might need to use monitoring and interval statistics. You can use the SIT overrides to set up your CICS regions based on the components you want to use.

If you think of an organization that is using many CICS regions it might be the best practice to set up variables in your startup procedure that point to an individual SYSIN partitioned data set (PDS) member for each CICS region. The individual SIT overrides can then be easily managed by editing the SYSIN member that belongs to an individual CICS region. See Example 2-1 on page 37. The SYSIN DD Statement is using a variable &INDEX1 and another one &SIP, as

shown in. During execution the SYSIN Statement is translated to //SYSIN DD DISP=SHR,DSN=CICSUSER.SYSIN(XX01). Member XX01 will contain the SIT override parameters for region TAXPXX01.

Example 2-1 SYSIN SIT Overrides in CICS start procedure

```
//*********************************************************
//C66START PROC START='INITIAL',
// INDEX1='CICSUSER',
// INDEX2='CICSTS41.CICS',
// INDEX3='CICSUSER.CNTL',
// REG='OM',
// SIP=XX01,
// DB2LOAD='DB2.DBX1.SDSNLOAD',
// SCSQLOAD='MQSV7.SCSQLOAD',
// SCSQANLE='MQSV7.SCSQANLE',

//CICS    EXEC PGM=DFHSIP,REGION=&REG,TIME=1440,
// PARM='START=&START,SYSIN'
//*
//SYSIN    DD DISP=SHR,
// DSN=&INDEX1..SYSIN(&SIP)
      ...
      ...
```

2.1.8 System initialization Table

To tailor the bank's CICS production regions you can create a SIT table. The SIT provides input to the CICS initialization program DFHSIP. During the CICS initialization process CICS facilities are started and set up based on the information that you specified in the SIT.

The information defined by system initialization parameters can be grouped into three categories:

1. Information used to initialize and control CICS system functions (for example, information, such as the dynamic storage area limits and the region exit time interval)
2. Module suffixes used to load your own versions of CICS control tables and individual user replaced modules (URMs)
3. Special information used to control the initialization process

The primary method of providing system initialization parameters is with a system initialization table. The parameters of the SIT, which you assemble as a

load table, supply the system initialization program with most of the information necessary to initialize the system to suit your unique environment. You can generate more than one SIT, and at the time of system initialization select the one that is appropriate to your needs.

You can also specify other system initialization parameters, which cannot be coded in the SIT. You can specify which SIT you want, and other system initialization parameters (with a few exceptions), in any of three ways:

- In the PARM parameter of the EXEC PGM=DFHSIP statement
- In the SYSIN data set defined in the startup job stream
- Through the system operator's console

You can also use these methods of input to the system initialization process to override most of the system initialization parameters assembled in the SIT.

There are some system initialization parameters that cannot be coded in the SIT; therefore, other parameters cannot be specified in any other way. Finally, some system initialization parameters cannot be specified through the system operator's console. For more information, see the CICS System definition Guide.

2.1.9 Program Load Table

For our bank's CICS production environment, it is actually a requirement to establish specific facilities before CICS users can get access to the system.

You can write programs to run during the initialization and shutdown phases of CICS processing. Any program that is to run at these times must be defined to CICS in a program list table (PLT).

2.1.10 Why you use initialization and termination programs

During the CICS initialization process CICS passes control to user written programs at two distinct phases. While in the first phase CICS calls user programs that can issue EXEC CICS ENABLE commands to provide user exits services before users sign in. We discuss user exits later in the chapter.

During the final phase of CICS initialization, most CICS services are available to user-written PLT programs. During this stage, you can use your own or CICS-provided programs to start your online monitor, WebSphere MQ bridge support, or the DB2 attachment for example. When a CICS terminal user signs in to CICS all facilities they need must be established during CICS initialization.

Any program that is to run during CICS initialization must be specified in a program list table. Example 2-2 on page 39 shows a sample of how you can

define your programs to a PLT. Programs that are defined before entry DFHDELIM run during the first stage of CICS initialization. Programs that are defined after entry DFHDELIM run during the final stage of initialization.

Example 2-2 PLTPI table

```
*
 * LIST OF PROGRAMS TO BE EXECUTED SEQUENTIALLY DURING SYSTEM
 * INITIALIZATION.
 * REQUIRED SYSTEM INITIALIZATION PARAMETER: PLTPI=I1
 *
    DFHPLT TYPE=INITIAL,SUFFIX=I1
 *
 *  The following programs are run in the first pass of PLTPI
 *
    DFHPLT TYPE=ENTRY,PROGRAM=TRAQA  EXECUTED DURING 2ND INIT. PHASE
    DFHPLT TYPE=ENTRY,PROGRAM=TRAQB  (PROGRAMS SHOULD ALSO BE DEFINED
    DFHPLT TYPE=ENTRY,PROGRAM=TRAQC  BY RDO)
 *
    DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
 *
 *
 *  The following programs are run in the second pass of PLTPI
 *
    DFHPLT TYPE=ENTRY,PROGRAM=TRASA  EXECUTED DURING 3RD INIT. PHASE
    DFHPLT TYPE=ENTRY,PROGRAM=TRASB  (PROGRAMS MUST ALSO BE DEFINED
    DFHPLT TYPE=ENTRY,PROGRAM=TRASC  BY RDO)
    DFHPLT TYPE=FINAL
 *
    END
```

Shutdown PLT programs are mostly used to clean up the task control blocks (TCBs) used by non-CICS components, such as online monitor products. A TCB is a dispatchable unit in an address space. Another example is the CICS supplied shutdown assist process. It ensures that as many tasks as possible commit or back out cleanly, enabling CICS to shut down in a controlled manner. The shut down assist process runs during the first phase of CICS shut down. Example 2-3 shows the PLTSD table.

Example 2-3 PLTSD table

```
DFHPLT TYPE=INITIAL,SUFFIX=T1
 *
 *  The following programs are run in the 1st pass of PLTSD
 *
```

```
*
   DFHPLT TYPE=ENTRY,PROGRAM=TRARA  EXECUTED DURING 1st QUIESCE PHASE
   DFHPLT TYPE=ENTRY,PROGRAM=TRARB  (PROGRAMS MUST ALSO BE DEFINED
   DFHPLT TYPE=ENTRY,PROGRAM=TRARC  BY RDO)
*
   DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
*
*
*  The following programs are run in the 2nd pass of PLTSD
*
   DFHPLT TYPE=ENTRY,PROGRAM=TRAFA  EXECUTED DURING 2nd QUIESCE PHASE
   DFHPLT TYPE=ENTRY,PROGRAM=TRAFB  (PROGRAMS MUST ALSO BE DEFINED
*                                   BY RDO)
   DFHPLT TYPE=FINAL
*
   END
```

2.2 Defining my resources to CICS

Before you can run a program in CICS you must define at least a transaction to CICS. A *transaction* is a basic resource in CICS. CICS is sometimes called a transaction manager. Additionally you must define other resources, such as programs, files, libraries, and so on. However, some definitions can be avoided using auto install functions. But in this case, you must provide a program to supply the auto install capability. There are sample programs available for program auto install, terminal auto install, and so on. At a starting point, to run a HelloWorld program, we need a transaction definition and a program definition.

The transaction name is up to you. The restriction is not to use transaction codes starting with the character C. All transactions starting with C are reserved to CICS itself. Examples are CEDA, CEMT, CECI and a lot more. Refer to the IBM book *CICS Supplied Transactions*, SC34-7004 for further information about supplied transactions.

CICS Explorer is a graphical tool widely used and well described in Chapter 6, “CICS Explorer” on page 205 in this book, and for a complete description of CICS Explorer, refer to the IBM Redbooks publication *IBM CICS Explorer*, SG24-7778-01. With this tool you can easily define resources to your CICS system in the CICS Systems Definition (CSD) file or to CICSPlex SM data repositories.

CEDA is a transaction that you can enter on a 3270 panel (character-based green panel) to define all the resources you require to a CSD.

The CICS system definition file is a VSAM data set containing a resource definition record for every resource defined to CICS by means of CICS Explorer, CEDA, or DFHCSDUP. DFHCSDUP is a batch utility, providing the same functions as CICS Explorer and CEDA.

After defining the resources for the system, they must be installed in a running system using the CICS Explorer or CEDA install function. Resources can also be installed automatically at startup.

Single resources must be in a *group*, which are combined into lists. At CICS startup you can specify the list names to install all groups in the list.

The Group A collection of related resources is on the CSD. Each resource that you define must belong to a group. You cannot define a resource without naming the group.

List the names of groups that CICS installs at an initial or cold start. You can add groups to lists if you want them installed at an initial or cold start, or if it helps you to manage your groups better. Groups do not have to belong to lists and can be defined independently.

2.2.1 Resource definition options

You can define most resources to CICS using several methods.

CICS Explorer

You can use the CICS Explorer to define, install, and manage resources. If CICS Explorer is connected to a CICS system, definitions are stored in the CSD file, and are installed into an active CICS system from the CSD file. If CICS Explorer is connected to CICSPlex SM, definitions are stored in the CICSPlex SM data repository and can be installed either automatically, during CICS initialization, or dynamically, into a running CICS system.

CICSPlex SM Business Application Services

You can use CICSPlex SM Business Application Services (BAS) to define and manage resources. Definitions are stored in the CICSPlex SM data repository and can be installed either automatically, during CICS initialization, or dynamically, into a running CICS system. For information about CICSPlex SM BAS, see CICSPlex System Manager Managing Business Applications.

Resource definition online

The resource definition online (RDO) method uses the CICS-supplied online transactions CEDA, CEDB, and CEDC. Definitions are stored in the CSD file, and are installed into an active CICS system from the CSD file.

Application bundles

You can deploy an application into CICS as a bundle and use the BUNDLEresource to create and manage the associated CICS resources. When you install a BUNDLE resource, CICS creates the required application resources dynamically. CICS also maintains a relationship with each of the resources so that you can enable or disable the application using the BUNDLE resource, rather than managing each application resource individually. The BUNDLE resource that represents the application is stored in the CSD file. The resources that are created dynamically when you install a BUNDLE resource are not stored in the CSD file.

DFHCSDUP offline utility

The DFHCSDUP offline utility method allows you to make changes to definitions in the CSD file by means of a batch job submitted offline. The definitions are stored in the CSD file.

Automatic installation (auto install)

Auto install minimizes the need for a large number of definitions by dynamically creating new definitions based on a “model” definition provided by you.

System programming using EXEC CICS CREATE commands

You can use the EXEC CICS CREATE commands to create resources independently of the CSD file. For further information, see the CICS System Programming Reference.

System programming, using the EXEC CICS CSD commands

You can use the EXEC CICS CSD commands to manage resource definitions in the CSD file from a user-written program. EXEC CICS CSD commands can perform all of the functions of CEDA except CEDA CHECK.

Macro definition

You can use the assembler macro source to define resources that cannot be stored on the CSD. The definitions are stored in assembled control tables in a program library from which they are installed during CICS initialization.

You must use macro instructions to define non-VTAM networks and terminals, non-VSAM files, databases, and resources for monitoring and system recovery.

Table 2-1 shows a comparison of resource definition, highlighting advantages and disadvantages of each of the options. This is a useful table to use when deciding how to define your resources.

Table 2-1 resource definition comparison table

Method	Description	Advantages	Disadvantages
CICS Explorer	Using the CICS Explorer you can define, alter, and install resources in a running CICS system.	► Intuitive and easy to use interface ► Integration point for other CICS tools ► Centralized resource definition ► Logical scoping ► Distributed rresource installation	FEPI resources cannot be defined with CICS Explorer.
CICSPlex SM BAS	Using BAS, you can create, maintain, and install CICS resources in a running CICS system. For full information, see the CICSPlex System Manager Managing Business Applications.	► Centralized resource definition ► Logical scoping ► Distributed resource installation	None
RDO	This method uses the CEDA transaction, which allows you to define, alter, and install resources in a running CICS system.	RDO is used while CICS is running, so allows fast access to resource definitions.	Because CEDA operates on an active CICS system, take care if it is used in a production system. Use some form of auditing as a control mechanism.
EXEC CICS CREATE system commands	This method allows you to add CICS resources to a CICS region without reference to the CSD file.	It enables configuration and installation of CICS resources for large numbers of CICS regions from a single management focal point. It also allows you to write applications for administering the running CICS system.	CREATE commands neither refer to nor record in the CSD file. The resulting definitions are lost on a cold start, and you cannot refer to them in a CEDA transaction.

Method	Description	Advantages	Disadvantages
EXEC CICS CSD system commands	This method updates resources on the CSD file, which means you can define, alter, and install resources in a running CICS system.	You can write applications customized to your environment that can manage the CSD and installed resources. Resources updated by this method can be referred to by CEDA.	Requires more work to implement than some other methods.
DFHCSDUP	DFHCSDUP is an offline utility that allows you to define, list, and modify resources using a batch job. DFHCSDUP can be invoked as a batch program or from a user-written program running either in batch mode or under TSO. Using the second method, you can specify up to five user exit routines within DFHCSDUP.	► You can modify or define a large number of resources in one job. ► You can run DFHCSDUP against a non-recoverable CSD file while it is being shared between CICS regions using RLS access mode	► You cannot install resources into an active CICS system. ► You cannot make updates using DFHCSDUP against a recoverable CSD file that is being accessed in RLS mode.
Application bundles	This method applies to application resources only. It uses the bundle deployment support in CICS to create the application resources and maintains a relationship to enable and disable all resources together.	► You do not have to create all of the required resources for an application manually. ► You can change the availability of applications available by updating one resource.	► You cannot browse the contents of a bundle using CEMT. ► A limited set of application resources are supported.

Method	Description	Advantages	Disadvantages
Auto install	This applies to VTAM terminals, LU6.2 sessions, IPIC connections, journals, programs, mapsets, and partition sets. You set up “model” definitions using either RDO or DFHCSDUP. CICS can then create and install new definitions for these resources dynamically, based on the models.	If you have large numbers of resources, much time is needed to define them, and if they are not all subsequently used, storage is also wasted for their definitions. Using auto install reduces this wasted time and storage.	You must spend some time initially setting up auto install in order to benefit from it.
Macro	Using this method, you code and assemble macro instructions to define resources in the form of tables.	Where possible, use the other methods.	► You can change the definitions contained in the tables while CICS is running, but you must stop and restart CICS if you want it to use the changed tables. ► You must do time-consuming assemblies to generate macro tables.

2.2.2 Where are my resource definitions are held

Every resource defined to CICS by means of CEDA or DFHCSDUP is held on the CICS system definition file, which is a VSAM data set.

The CSD file can be defined as recoverable, so that changes made by CEDA or CEDB that were incomplete when an abend occurred are backed out. CICS allows a CSD file and its resource definitions to be shared between different CICS systems. For more information about defining the CSD, see in the CICS System Definition Guide.

CICS control tables contain resource definition records for resources that cannot be defined in the CSD. The tables and their resource definitions are created by using the CICS table assembly macro instructions. You must code assembler-language macro statements for each resource to appear in the table,

assemble the complete set of macro statements, link-edit the output to produce a load module, and specify the module suffix in DFHSIT.

2.2.3 Organization of resource definitions

Resource definitions that held on the CSD are organized into groups and lists:

Group A collection of related resources on the CSD. Each resource that you define must belong to a group. You cannot define a resource without naming the group.

List The names of groups that CICS installs at an initial or cold start. You can add groups to lists if you want them installed at an initial or cold start, or if it helps you to manage your groups better. Groups do not have to belong to lists, and can be defined independently.

2.2.4 Getting started with CEDA

To access CEDA from a CICS terminal:

Enter CEDA at a CICS terminal. The cursor is indicated by the symbol '_'.

A list of all the CEDA commands is displayed. The CEDB transaction does not support the INSTALL command. The CEDC transaction supports only the DISPLAY, EXPAND, and VIEW commands.

Creating a resource definition

You can use the CEDA transaction to create a new resource definition.

To create a map set definition:

1. Type `CEDA` to start the CEDA function.
2. To create a new resource definition, type `DEFINE`. Figure 2-3 on page 47 shows a CEDA Define panel.

```
 DEF
 ENTER ONE OF THE FOLLOWING

Atomservice  MQconn       Webservice
Bundle       PARTItionset
CONnection   PARTNer
CORbaserver  PIpeline
DB2Conn      PROCesstype
DB2Entry     PROFile
DB2Tran      PROGram
DJar         Requestmodel
DOctemplate  Sessions
Enqmodel     TCpipservice
File         TDqueue
Ipconn       TErminal
JOurnalmodel TRANClass
JVmserver    TRANSaction
LIbrary      TSmodel
LSrpool      TYpeterm
MApset       Urimap

                                                SYSID=RED1 APPLID=EPRED1

PF 1 HELP       3 END            6 CRSR            9 MSG            12 CNCL
```

Figure 2-3 Ceda define panel

This panel lists all the resource types that you can define for CICS using CEDA.

To create a transaction definition, as shown in Figure 2-4 on page 48:

1. Clear the panel, and start the CEDA transaction.
2. Create a transaction definition. You can type the whole command on the CEDA initial panel:

   `DEFINE TRANSACTION(TEST) PROGRAM(MENU) GROUP(EP)`

```
 OVERTYPE TO MODIFY                                          CICS RELEASE = 8810
  CEDA  DEFine TRANSaction( TEST )
   TRANSaction    : TEST
   Group          : EP
   DEScription  ==> MENU
   PROGram      ==> MENU
   TWasize      ==> 00000              0-32767
   PROFile      ==> DFHCICST
   PArtitionset ==>
   STAtus       ==> Enabled            Enabled | Disabled
   PRIMedsize     : 00000              0-65520
   TASKDATALoc  ==> Below              Below | Any
   TASKDATAKey  ==> User               User | Cics
   STOrageclear ==> No                 No | Yes
   RUnaway      ==> System             System | 0 | 500-2700000
   SHutdown     ==> Disabled           Disabled | Enabled
   ISolate      ==> Yes                Yes | No
   Brexit       ==>
 + REMOTE ATTRIBUTES

                                                  SYSID=RED1 APPLID=EPRED1
  DEFINE SUCCESSFUL                             TIME: 15.07.32  DATE: 06/21/11
PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 2-4 Define transaction to CICS using CEDA

You must specify a PROGRAM whenever you define a new TRANSACTION. If you do not, CEDA displays a severe error message saying that you must specify either a PROGRAM or REMOTESYSTEM.

You now have a new resource to work with: a TRANSACTION in group EP.

For more details about using CEDA, see the *CICS manual Resource Definition Guide*, SC34-7181-00, which walks you through all of the available options.

2.3 CICS DB2 Attachment Facility

In this section, we discuss the CICS DB2 Attachment Facility.

2.3.1 Why we use the CICS DB2 Attachment Facility

Figure 2-5 illustrates the CICS DB2 attachment.

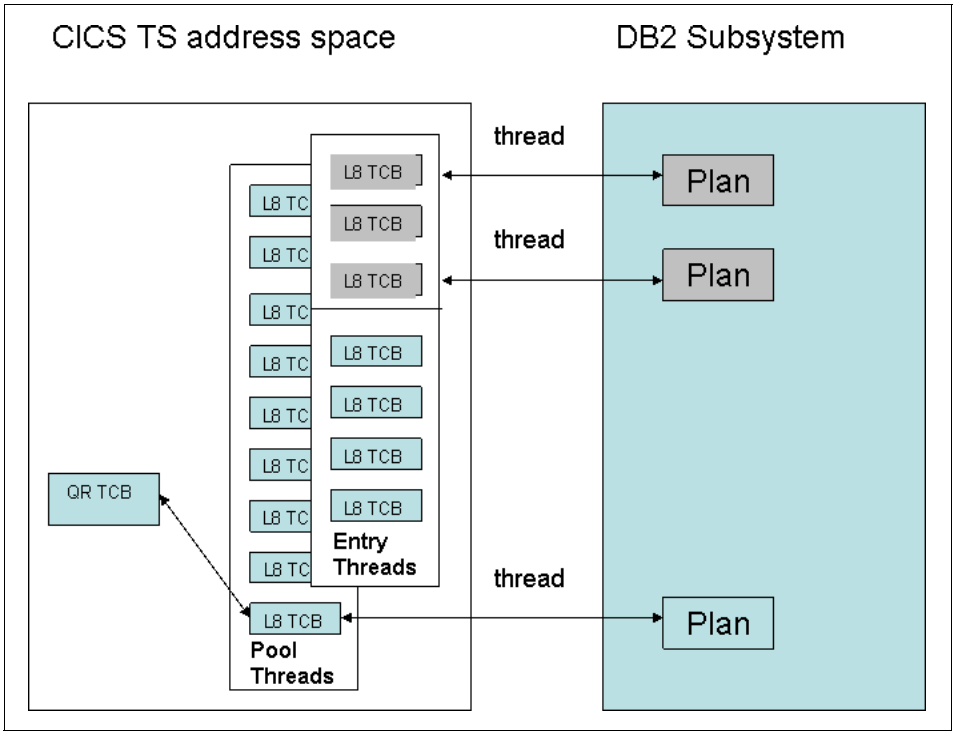


Figure 2-5 The CICS DB2 attachment

Our bank takes advantage of the DB2 data sharing technology in a z Parallel Sysplex to provide CICS applications with full concurrent read and write access to shared DB2 data.

CICS allows application programs to have access to relational SQL databases, such as DB2. Using SQL requests in CICS application programs allows you to fetch, update, define, or to authorize requests against database tables or databases. To run CICS DB2 programs you must connect CICS and DB2.

CICS and DB2 run in different address spaces; therefore, a mechanism to connect to DB2 is needed. Furthermore CICS runs many tasks that issue SQL DB2 requests concurrently in unpredictable rates; therefore, a multi threaded connection to DB2 is required. The CICS-supplied CICS DB2 attachment Facility connects CICS to DB2 and allows you to manage and tune the required resources.

Thread: A thread is a mechanism that a CICS task can use to connect to DB2.

The CICS DB2 attachment is well integrated into CICS and DB2 functions. You can use the CICS DB2 attachment to manage thread resources carefully. For example you can dedicate threads to high volume transactions or you can limit the number of threads that can access DB2 concurrently.

2.3.2 Design of the CICS DB2 attachment Facility

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

2.3.3 Threads

A CICS SQL DB2 application always gets started on a so-called Quasi Reentrant TCB (QR TCB). Within a z/OS address space, such as CICS, a TCB is a dispatchable unit that can run a Task. Quasi reentrant says that control is given to the application program by the CICS dispatcher only. Only one program can run in quasi reentrant mode at a time. From the CICS users perspective QR TCB tasks appear to run in parallel, but they get dispatched by the CICS dispatcher sequentially. Tasks run until they give up control thus the next task has a chance to get dispatched. Quasi reentrant CICS tasks run sequentially on the QR TCB.

Now when a CICS SQL DB2 program gains control on the QR TCB for the first time, the application logic runs until the first SQL call is about to be executed. At this point we need to switch to a so-called open L8 TCB. Unlike the quasi reentrant TCB that uses ‘QR’ in the mode name, the meaning of the ‘L’ in the TCB mode name is **CICS LE** TCB.

We must switch to an L8 TCB to process the SQL call issued by the application. When the SQL call completes, we switch back to the QR TCB to continue the quasi reentrant application program logic. While the SQL call is processed the CICS dispatcher gives control to the next task in the ready to run chain. From the DB2 perspective the L8 TCB is the application TCB. The L8 TCB is also called the Thread TCB. In a typical production system, you run many tasks that issue SQL requests therefore CICS attaches many L8 TCBs. We discussed that quasi reentrant programs run sequentially on the QR TCB. These tasks attach L8 TCBs to process SQL requests. Therefore we have a number of L8 TCBs but just one QR TCB. when SQL processing completes on the L8 TCB, the result has to be returned to the task that issued the SQL call. In order to return the result to the quasi reentrant program we have to switch back to the QR TCB.

A CICS systems programer has to take care that the QR TCB does not get constraint when many L8 TCBs switch back to the QR TCB. Tasks might have to wait for the QR TCB to get dispatched. It is the CICS system programmers responsibility to define a proper thread TCB limit to avoid delays for the QR TCB.

It is interesting enough that the QR TCB per default uses a lower z/OS dispatching priority than L8 TCBs. This shows that the CICS DB2 attachment is well integrated and designed for high throughput rates. As you can imagine things can get challenging if you run many tasks that issue high-volume SQL requests. In this case, we switch back and forth between QR and L8 TCBs many times. It is the systems programmers responsibility to detect CICS DB2 application programs that issue needless SQL requests due to unlimited generic searches in DB2 tables.

Threadsafe enabled mode: You can avoid QR-L8-QR switches if you run the application in threadsafe enabled mode. What you get is higher throughput rates and less CPU consumption.

A program or part of it is threadsafe if it can run in a multi threaded environment.

2.3.4 Why we use pool and entry threads

You can use the CICS DB2 attachment to specify the number of threads that can run concurrently between DB2 and CICS. In CICS, we can use two types of threads. Actually the threads are physically the same but they get handled differently. For low volume transactions you can share threads from a pool of threads. For high volume CICS DB2 transactions you can use dedicated threads. Dedicated threads can be defined for individual transactions or a set of transactions.

For dedicated threads or so-called entry threads you limit the number of threads they can use. You also specify a subset of the entry threads as protected threads.

Unlike pool threads or entry threads CICS does not terminate protected threads immediately after a task completes. There is a considerable cost of CPU time if threads cannot be reused for high volume transactions. It is the CICS system programmers responsibility to make sure that thread reuse for protected threads takes place.

If the thread limit for entry threads runs out of threads CICS can borrow a thread from the pool to satisfy the incoming request. This process is known as a thread wait overflow.event. If you have frequent thread wait overflows, the CICS system programmer can increase the thread limit for this specific entry thread.

Pool threads

Pool threads are used for all transactions and commands that are not dedicated to a specific user transaction ID or a DB2 command thread. Pool threads are intended for low volume transactions, and for overflow transactions that cannot obtain an entry thread or a DB2 command thread. A pool thread is terminated immediately if no CICS transaction is waiting to use it. Pool threads are defined in the pool threads section of the DB2CONN definition.

Entry threads

Entry threads are specially defined threads intended for transactions with special requirements, such as transactions that require a fast response time or transactions with special accounting needs. You can instruct the CICS DB2 attachment facility to give entry threads to particular CICS transactions. You define the types of entry threads that are needed for different transactions, and you can set a limit on the number of each of these types of entry thread. If a transaction is permitted to use an entry thread, but no suitable entry thread is available, the transaction can overflow to the pool and use a pool thread, wait for a suitable entry thread, or abend, as you chose in the definition for the entry thread.

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

2.3.5 Managing the CICS DB2 attachment facility

CICS provides a set of commands that help to manage the CICS DB2 connection. You can issue the commands by utilizing the supplied DSNC transaction. You can execute the following CICS DB2 attachment functions using the DSNC transaction:

- Enter DB2 commands from a CICS terminal.
- Cause threads to be terminated when they are released (DSNC DISCONNECT).
- Display information about transactions using the CICS DB2 interface and display statistics (DSNC DISPLAY).
- Modify the unsolicited message destinations, and modify the number of active threads used by a DB2ENTRY, the pool, or for commands (DSNC MODIFY).
- Shut down the CICS DB2 interface (DSNC STOP).
- Start the CICS DB2 interface (DSNC STRT).

You can also use the CICS master terminal transaction CEMT to manage the CICS DB2 attachment, for example, you can use transaction CEMT to manage the following CICS DB2 attachment resources:

- CEMT INQUIRE DB2CONN. Allows you to define the number of L8 TCBs that can be used to connect threads. You can also define the number of threads to be used by the pool. The DB2CONN resource definition is the place where we define the name of the DB2 member that CICS is going to connect to.
- CEMT INQUIRE DB2Entry. You an use the DB2Entry definition to manage threads for a specific transaction or a set of transactions. Within the DB2Entry resource definition you can specify a thread limit used for DB2ENTRY and furthermore you can define a subset of threads that can be used as protected threads. Protected threads can be reused and they are intended for high volume transactions.
- CEMT INQUIRE DB2TRAN. Use this command to inquire on the transactions that belong to DB2Entry definitions.

2.3.6 Why you manage the CICS DB2 Attachment

It is the CICS system programmers responsibility to keep the CICS DB2 attachment in good shape. It is essential to define the appropriate thread limit for pool and entry threads. CICS system programmers must always keep an eye on the peak values for used threads. If, for example, peak values for entry threads exceed the limit frequently, the number of threads must be adjusted to avoid thread wait overflows.

The CICS system programmer must also frequently check that entry threads have sufficient values for thread reuse.

The CICS system programmer must make sure that consistent thread limits are defined. The rule of thumb can be:

- The number of pool threads + the sum of all entry threads + the number of command threads = the number of thread TCBs. The number of thread TCBs is specified on the TCBLIMIT parameter in DB2CONN resource parameter.
- The number of thread TCBs must be equal to the limit of open L8 TCBs that CICS uses. You can use the MAXOPENTCB system initialization parameter to specify the limit of open L8 TCBs. You can require additional open TCBs for WebSphere MQ or TCP/IP socket application programs.

2.4 CICS and WebSphere MQ

Figure 2-6 illustrates the CICS WebSphere MQ connectivity.

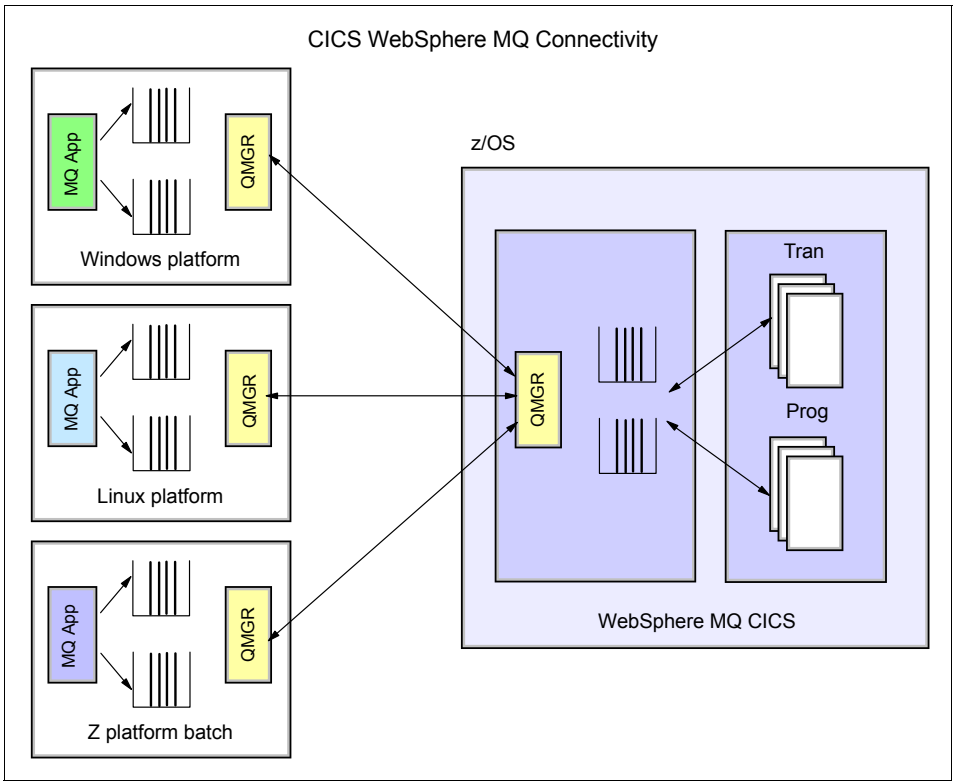


Figure 2-6 CICS WebSphere MQ connectivity

Our bank's international organization must integrate data between applications that run on distinct platforms in several locations. The bank uses the CICS WebSphere MQ connectivity to integrate CICS applications to the services that run in several geographic locations.

The bank takes advantage of the WebSphere MQ components, which include the CICS MQ adapter, the CICS MQ trigger monitor, and the CICS MQ bridge.

In this section, we describe how you can use the messaging services of WebSphere MQ to utilize the services of CICS applications.

WebSphere MQ applications are supported to run on many platforms. IBM WebSphere MQ provides a mechanism for application integration by passing messages between applications running on different platforms perhaps in several organizations.

2.4.1 Interfacing WebSphere MQ with CICS

Many organizations developed valuable CICS application programs a long time ago. These CICS applications run reliably every day. You can imagine that it might be useful to expose the services that CICS applications provide to programs that run outside of CICS, perhaps on different platforms.

To provide facilities that make use of CICS application services driven by WebSphere MQ messages, CICS came up with two types of interfaces that allow you to bring workload into CICS using WebSphere MQ messages:

- CICS WebSphere MQ Adapter
- CICS WebSphere MQ Bridge

The CICS WebSphere MQ Adapter or the so-called trigger monitor allows you to initiate user transactions in CICS using the WebSphere MQ message trigger mechanism. You can put a message to an application queue that is enabled for triggering. When the message releases the trigger WebSphere MQ sends a trigger message to CICS to initiate a defined-user transaction. We describe briefly how the MQ trigger monitor works later in the section.

Note: For a description of MQ triggering see 4.7, "WebSphere MQ" on page 159.

IBM 3270: IBM 3270 is a display device that IBM made. It started in 1972.

You can use the WebSphere MQ Bridge to run existing 3270 style transactions. The CICS WebSphere MQ Bridge also allows you to call CICS application programs directly by sending a message that contains the name of the target program to a queue. We describe the capabilities of the WebSphere MQ Bridge later in the section.

The WebSphere MQ Bridge protects the investment in CICS application programs. For 3270 terminal programs, it is recommended to split the logic that runs the terminal communication from the terminal that performs the real business logic. Although it is the best practice to code the presentation logic and the business logic separately, many programmers included their logic in one program module. If the presentation logic is coded separately, it can call one or more modules that contain business logic. During the process of calling the business logic a communication area is passed to the target module.

Using this mechanism you can of course replace the front end of the application with a more modern component, for example, you can replace the 3270 front end with a web service client, or you can use WebSphere MQ message queues to link to a business module in CICS. The communication data that must be passed to the target program is put to a so-called request queue.

If the presentation logic is not separated, use the CICS WebSphere MQ Bridge to emulate the 3270 Terminal requests. The 3270 CICS application can then be re-used without an existing 3270 device.

2.4.2 CICS WebSphere MQ Adapter

The CICS-WebSphere MQ adapter is supplied with CICS and communicates with WebSphere MQ as an external resource manager.

The CICS-WebSphere MQ adapter provides two main facilities:

- A set of control functions for use by system programmers and administrators to manage the adapter.
- WebSphere MQ Message Queue Interface (MQI) support for CICS applications.

The CICS WebSsphere MQ adapter provides a mechanism to initiate user transactions in the CICS address space that runs the so-called MQ trigger monitor. The MQ trigger monitor can be used to generate CICS workload in a single CICS region or in more than one CICS region in a high-availability environment. In a high availability environment, you have more than one CICS region with active trigger monitors available, perhaps one CICS region with an active trigger monitor per LPAR.

The initiation of CICS user transactions is triggered by the queue manager when an application message is put to an MQ queue that is configured for triggering. The WebSphere MQ queue manager sends a trigger message to the trigger monitor that is listening in a CICS address space to start a user transaction. If you have more than one trigger monitor active, either all MQ trigger monitors get a chance to receive the trigger message at the same time or trigger messages are sent to each trigger monitor in turn.

When user transactions are initiated in the CICS region that runs the trigger monitor, CICSPlex SM WLM can be used to route transactions to suitable target regions.

The system programmer's responsibility

The MQ trigger monitor listens for trigger messages. When a trigger message arrives the transaction ID named in the trigger message is initiated. For the application logic, the following design options are possible:

- The triggered application runs in the CICS region where the CICS trigger monitor resides. This region acts as a listener for MQ trigger messages but can also be the server for triggered applications.
- The triggered application uses synchronous EXEC CICS LINK commands to pass control to a program that will run in an application owning region (AOR). On return control is passed back to the initiating program.
- The triggered application issues asynchronous EXEC CICS START commands to route a new transaction to run on one of the AORs.

From the design perspective, the CICS system programmer must consider the following thoughts:

- The user application must retrieve the information from the trigger message to obtain the name of the reply-to queue and the name of the application message queue. The CICS application must issue MQ API requests to get the message from the application queue. If the transactions are routed to any AOR region, all AORs must have a WebSphere MQ adapter installed. For example, in this case the triggered user application can read the application queue and pass the data to the target transaction using communication areas or channels and containers. The remote target transactions can be started asynchronously using an EXEC CICS START TRANSID command. After the started transactions run on one of the AORs they do not return to the CICS MQ listener region. Any results on return are put directly into the reply-to queue.
- If the triggered CICS user application is designed to process remote transactions synchronously you can use the triggered application to retrieve the information from the trigger message and after that read the messages from the application message queue. To start the applications synchronously

on one of the AORs you can use a distributed program link request to link to the target application program. The information read from the application queue can be passed using communication areas or channels and container. On return the communication area is passed back to the CICS MQ listener region. The application that issued the link request sends back the results to the reply-to queue. Figure 2-7 shows the CICS MQ triggering using shared queues.

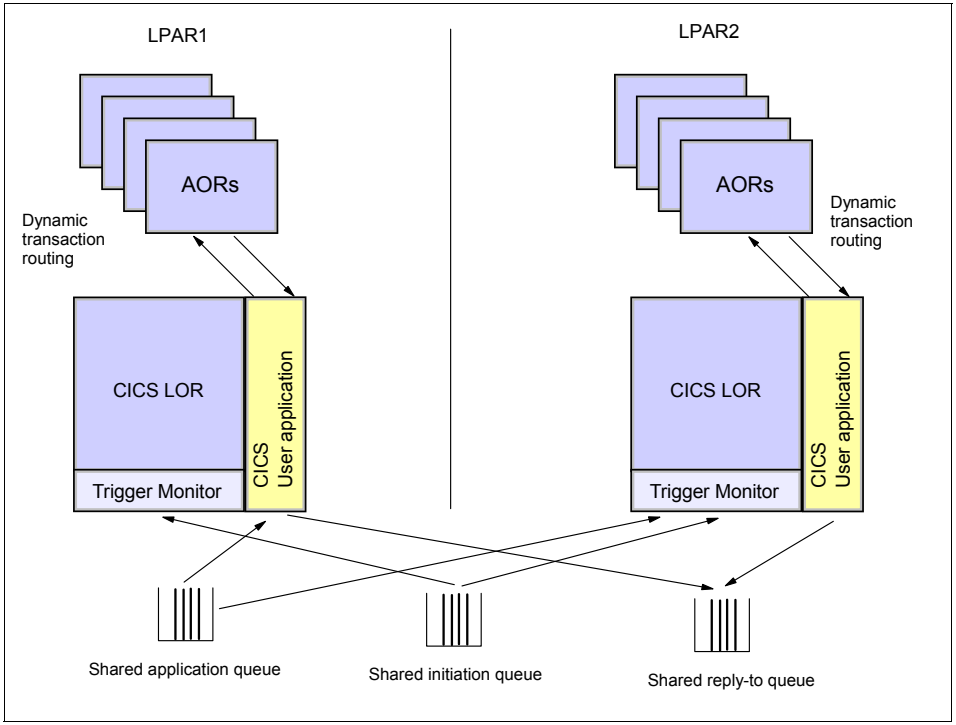


Figure 2-7 CICS MQ triggering using shared queues

In a CICS multiregion operation or intersystem communication (ISC) environment, each CICS address space can have its own attachment to the queue manager subsystem. A single CICS address space can connect to only one queue manager at a time. However, each address space can connect to a queue manager subsystem.

2.4.3 The trigger monitor CKTI

Figure 2-8 on page 59shows the CICS MQ trigger monitor.

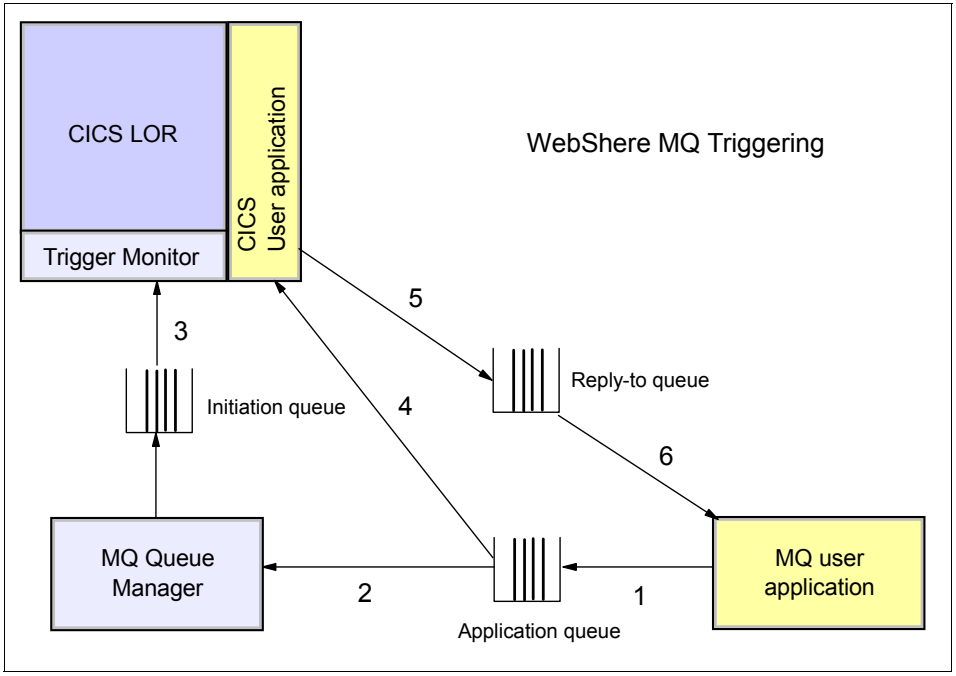


Figure 2-8 CICS MQ trigger monitor

The CICS MQ Adapter task initiator service CKTI starts a CICS transaction when a WebSphere MQ trigger message is read, for example, when a message is put onto a specific application queue.

When a message is put onto an application message queue:

1. A trigger is generated if the trigger conditions are met.
2. The queue manager then writes a message containing user-defined data, known as a trigger message to the initiation queue that was specified for that message queue. In a CICS environment, you can set up an instance of CKTI to monitor an initiation queue and to retrieve the trigger messages from it as they arrive. CKTI starts another CICS transaction, specified using the DEFINE PROCESS command, which typically reads the message from the application message queue and then processes it.
3. The process must be named on the application queue definition, not the initiation queue.
4. On return the CICS application puts information into the reply-to queue.
5. The MQ application finally can get the reply message from the reply-to queue.

Each copy of CKTI services a single-initiation queue. To start or stop a copy of CKTI, you must supply the name of the queue that this CKTI is to serve or is now serving. You cannot start more than one instance of CKTI against the same initiation queue from a single CICS subsystem.

If you issue start and stop commands to manage CKTI instances without specifying an initiation queue, these commands are automatically interpreted as referring to the default initiation queue for the CICS region. You can specify the name of the default initiation queue in the MQCONN resource definition for the CICS region. When you install the MQCONN resource definition, CICS creates and installs an implicit MQINI resource definition to represent the default initiation queue. You can change the name of the default initiation queue by changing and reinstalling the MQCONN resource definition to create a new MQINI resource definition. You can also name an alternative default initiation queue if you start the CICS-WebSphere MQ connection manually.

2.4.4 CICS WebShere MQ Bridge

Figure 2-9 illustrates the CICS WebSphere MQ Bridge.

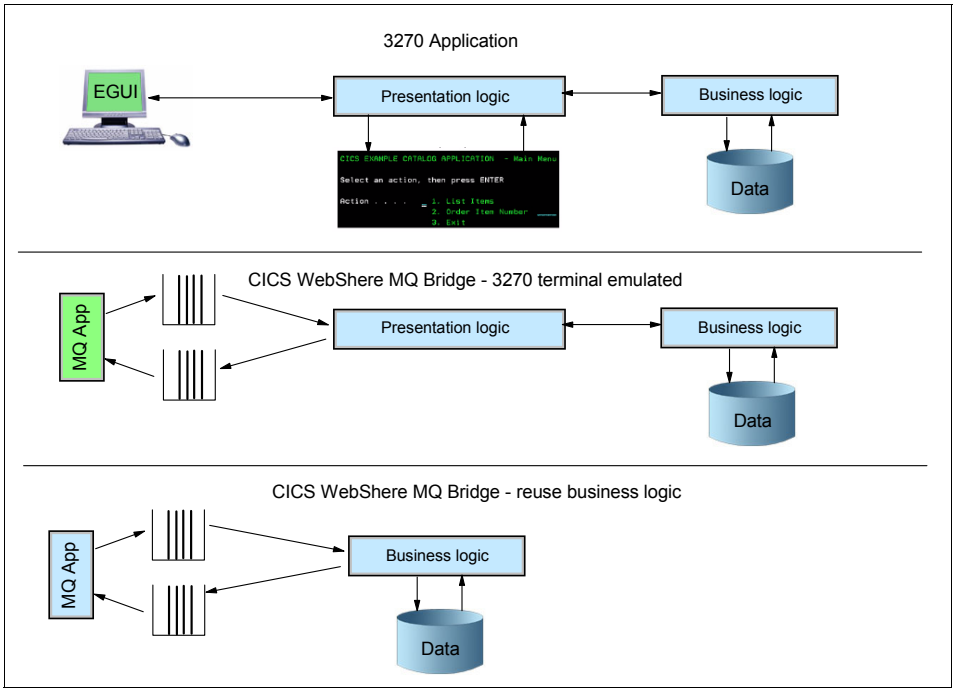


Figure 2-9 CICS WebSphere MQ Bridge

The CICS WebSphere MQ bridge is a component of CICS that allows direct access from WebSphere MQ applications to applications on your CICS system. Unlike MQ triggering, the CICS WebSphere MQ bridge provides direct access to MQ unaware CICS application programs.

CICS-MQ bridge: The term CICS-MQ bridge is used here to refer to the CICS shipped CICS WebSphere MQ bridge, which is used when connected to WebSphere MQ Version 6.0 and above. If CICS is connected to a release of WebSphere MQ earlier than Version 6.0, the CICS shipped CICS-WebSphere MQ bridge transfers control to the WebSphere MQ shipped bridge. Refer to the documentation for the relevant release of WebSphere MQ.

In bridge applications, there are no WebSphere MQ calls within the CICS application because the bridge enables implicit Message Queue Interface (MQI) support. This means that you can re-engineer traditional applications that were controlled by 3270-connected terminals to be controlled by WebSphere MQ messages, without having to rewrite, recompile, or re-link them.

Figure 2-10 on page 62 shows an example of a traditional 3270 application. In the traditional environment, the transaction starts when you enter its name at the CICS 3270 terminal and press Enter. Logic in the transaction causes it to issue EXEC CICS SEND MAP the first time that it is invoked in a pseudoconversation, and then to terminate by issuing EXEC CICS RETURN TRANSID(EGUI).

You enter values into fields in the map that is displayed at the terminal, and then press an AID key. Logic in the transaction the second time that it is invoked causes it to issue EXEC CICS RECEIVE MAP to receive the map. It updates certain fields in the map by changing values in its own application data structure and then issues EXEC CICS SEND MAP to redisplay the map at the user's terminal.

The user can then update fields in the redisplayed map and start the RECEIVE MAP - SEND MAP cycle again.

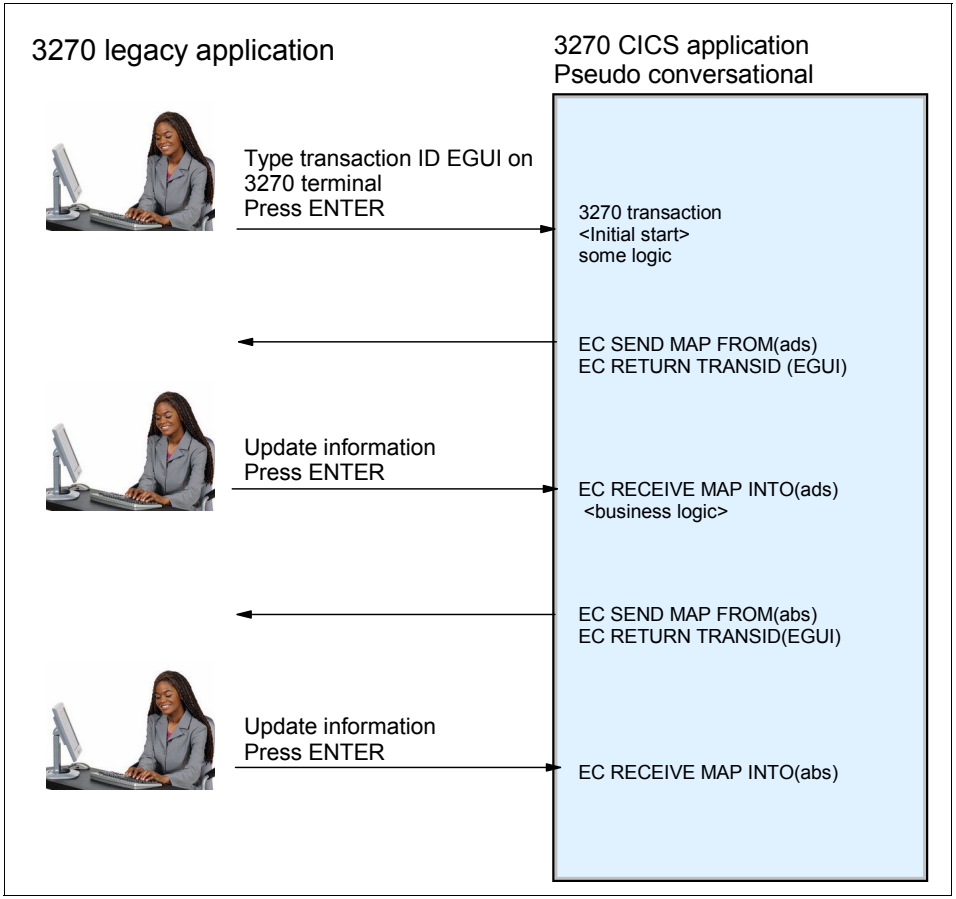


Figure 2-10 3270 traditional application

Figure 2-11 on page 63 shows the WebSphere MQ bridge application.

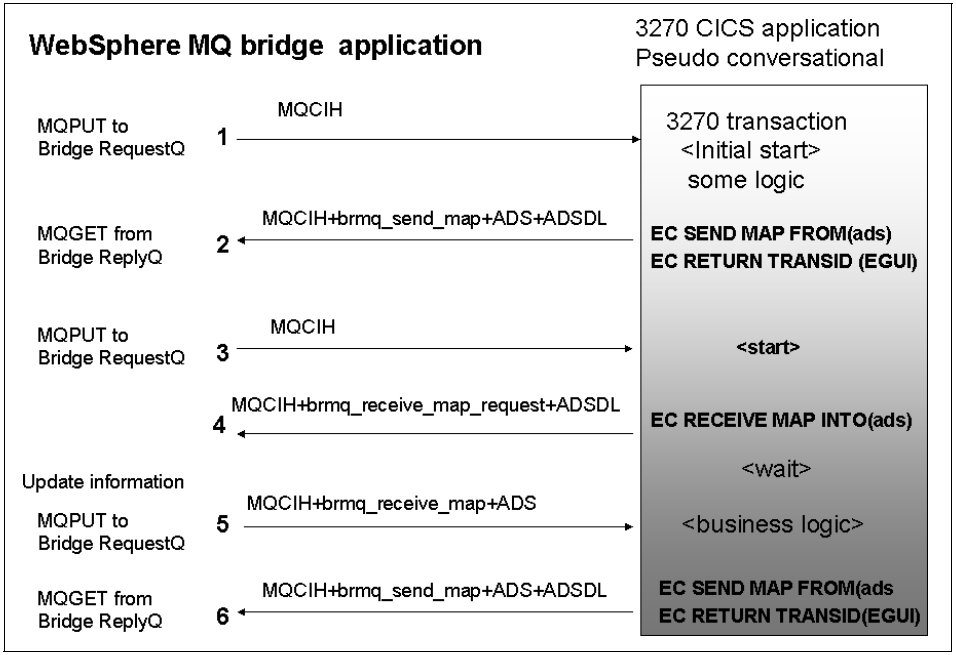


Figure 2-11 WebSphere MQ bridge application

In Figure 2-11:

1. The initial flow from the application contains just an MQCIH. The MQCIH includes control information specifying which transaction is to be started.
2. The return flow from the 3270 transaction contains an MQCIH, which includes a facility token to be used for all subsequent flows, and diagnostic information if an error occurred. It also contains a SEND MAP vector structure containing control information relating to the map itself, and data that represents the map. If the initiating application requested it, an application data structure descriptor is also included.
3. The bridge application sends a message back containing only an MQCIH. This contains control information to start the transaction again.
4. The 3270 transaction issues EXEC CICS RECEIVE MAP because it does in the traditional environment. However, in the bridge environment the map data is not immediately available. The call is converted to a message containing an outbound RECEIVE MAP request vector. The application data structure descriptor is also included in the message. In this example, the transaction waits while the message is turned around by the bridge application. The model here is a little different from that in the traditional environment. However, the bridge architecture allows messages to contain more than one

vector, so a number of requests can be satisfied by a single inbound message.

5. Having updated fields in the application data structure, the bridge application sends an inbound RECEIVE MAP reply vector to satisfy the outbound request.
6. The 3270 transaction issues EXEC CICS SEND MAP, which converts to a SEND MAP vector, and the cycle repeats.

WebSphere MQ applications use the CICS header (the MQCIH structure) in the message data to ensure that the applications can execute as they did when driven by non programmable terminals.

The bridge enables an application that is not running in a CICS environment to run a program or transaction on CICS and get a response. This non-CICS application can be run from any environment that has access to a WebSphere MQ network that encompasses WebSphere MQ for z/OS.

The bridge can be used by a CICS program that is invoked using the EXEC CICS LINK command. It must conform to the DPL subset of the CICS API, that is, it must not use CICS terminal or sync point facilities.

The bridge can be used by a CICS transaction designed to run on a 3270 terminal. This transaction can use BMS or TC commands. It can be conversational or part of a pseudoconversation. It is permitted to issue syncpoints. For information about the transactions that can be run, see the CICS External Interfaces Guide.

2.5 CICS logging and journaling

Figure 2-12 on page 65 illustrates CICS logging.

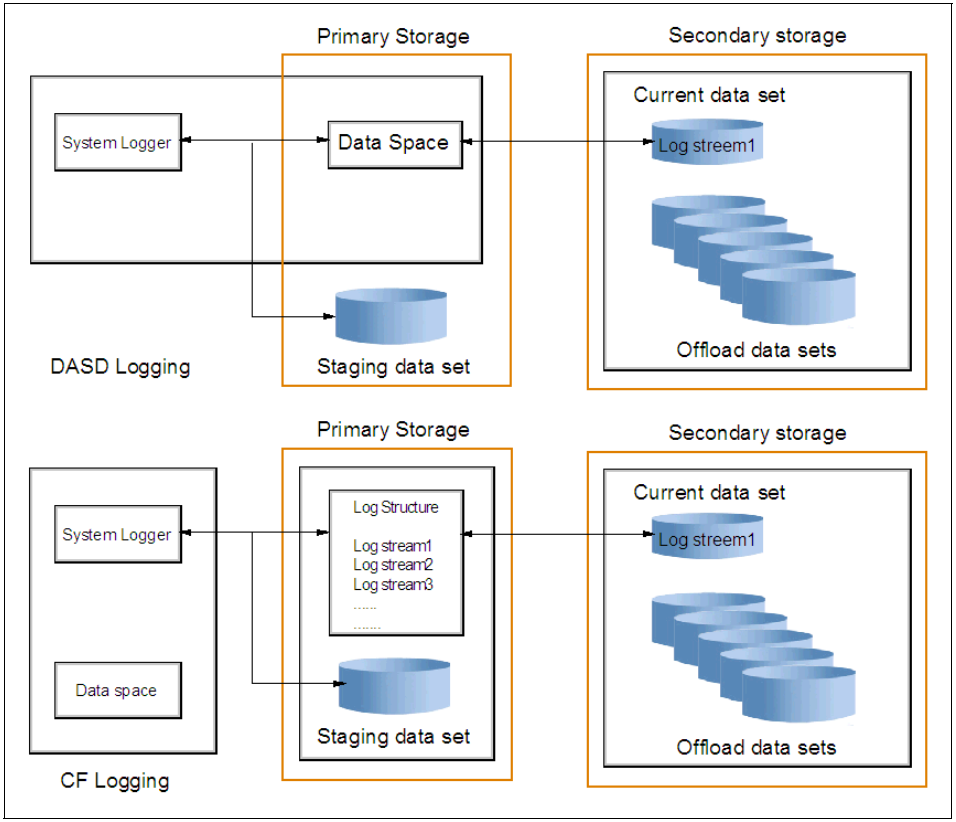


Figure 2-12 CICS logging

2.5.1 The need for logging and journaling for CICS

In a typical CICS production environment like the one that our bank uses, we run hundreds of transactions concurrently. Unlike batch programs, online transactions do not update resources sequentially; therefore, a single record in a file can be updated several times by more than one transaction. Data updates to resources occur at unpredictable rates, and they can fail in the middle of processing at any time because of hardware or software problems.

In the case of a system failure or transaction abend, a suitable and safe recovery mechanism is needed. In this section, we briefly describe why CICS uses the MVS logger to store recovery data and the basic concept of CICS the CICS recovery mechanism.

2.5.2 CICS MVS logging and journaling

In CICS versions earlier than CICS Transaction Server 1.1, another logging and journaling mechanism was used. Recovery information about application-initiated resource updates was written to a log buffer. Later on when the buffer exceeded a shift-up value, the data was written to a so-called log data set. CICS used two log data sets. When the first data set ran out of space, we switched to the second one and vice versa. When data set A switched to B and back to A the log data in data set A was overwritten.

The design problem using the old log mechanism

The following scenario is what the CICS systems programmer must contend with.

A CICS task starts and performs a number of updates. CICS is using log data set A. The task suspends for whatever reason. CICS continues until we switch to log data set B. While the task is still suspended log data set B fills up and we switch back to log data set A. CICS continues to run and after some time the recovery information for our task is overwritten. CICS now becomes vulnerable. In case of a system failure, CICS cannot roll back the changes since the last sync point of our long running task, which cannot be located anymore.

CICS Transaction Server 1.1 was the first version that used the MVS logger for recovery.

The CICS log manager provides facilities for the creation, control, and retrieval of journals during real-time CICS execution. Journals are intended to record, in chronological order, any information that you might later need to reconstruct data or events. For example, you can create journals to act as audit trails, to record database updates, additions, and deletions for backup purposes, or to track transaction activity in the system.

The CICS log manager uses the services provided by the MVS system logger. The MVS system logger provides a common logging mechanism for all subsystems, such as CICS or any other application. The CICS log manager stores information, such as, for example, recovery data to so-called logstreams.

The CICS log manager mainly provides:

- The system log: Information on the system log is used to recover CICS tasks to a consistent state. This happens, for example, if a task ends abnormally. Updates to recoverable resources for a task are rolled back to the last committed state. For example a typical CICS application issues the following EXEC CICS Commands in Example 2-4 on page 67. Assume CICS runs the application as task number 100. We added to new records to a VSAM file and after that the application issues a sync point. The sync point command

commits the Unit of work (UOW-100).Therefore when resources are being changed, there comes a point when the changes are complete and do not need backout if a failure occurs later. The period between the start of a particular set of changes and the point at which they are complete is called a unit of work.The *unit of work* is a fundamental concept of all CICS backout mechanisms. The CICS log manager uses the MVS logger services to write UOWs to the system log. The next two write requests for records 3 and 4 belong to the next UOW-100, which is not committed yet. Before the application can issue a sync point, we get an abnormal end for the task. The CICS recovery process handles the abend and reads the system log backwards until the last committed state. When the CICS recovery process completes, the file contains records 1 and 2. Records 3 and 4 are lost.

Example 2-4 UOWs for transaction backout

```
EXEC CICS WRITE FILE(FILEA) FROM(RECORD1) RIDFLD(KEY1)
EXEC CICS WRITE FILE(FILEA) FROM(RECORD2) RIDFLD(KEY2)
EXEC CICS SYNCPOINT
EXEC CICS WRITE FILE(FILEA) FROM(RECORD3) RIDFLD(KEY3))
EXEC CICS WRITE FILE(FILEA) FROM(RECORD4) RIDFLD(KEY4)
An abend occurs
```

- Forward recovery logs: If your CICS application comes to an abnormal end, you can use forward recovery to recover your changes to a VSAM file completely. CICS uses forward recovery logs for VSAM data sets. A *forward recovery log* is a general log stream managed by the MVS system logger. CICS writes the after images of changes made to a data set to a forward recovery log. Forward recovery logging is not automatic. You must specify that you want this facility for your files and also define the forward recovery log streams. See Example 2-5, where we change the contents of records 1 and 2. After the task comes to an abnormal end, you can do a forward recovery of the FILE. The contents of records 1 and 2 shows the updated data. The after images of the records can be recovered from the forward recovery log. You must use a tool, such as CICS VSAM Forward Recovery or you own solution to forward recover VSAM files.

Example 2-5 UOWs for forward recovery

```
EXEC CICS READ FILE(FILEA) INTO(RECORD1) RIDFLD(KEY1) UPDATE
Update record 1
EXEC CICS REWRITE FILE(FILEA) FROM(RECORD1) RIDFLD(KEY1)
EXEC CICS READ FILE(FILEA) INTO(RECORD2) RIDFLD(KEY2) UPDATE
Update record 2
EXEC CICS REWRITE FILE(FILEA) FROM(RECORD2) RIDFLD(KEY2)
An abend occurs
```

2.5.3 MVS logger design

CICS TS introduced the logger domain or the CICS log manager back in 1997. The CICS log manager replaces the journal control function of earlier releases. The CICS log manager improves the management of the CICS system log, forward-recovery logs, auto journals, and user journals.

The CICS log manager uses services provided by the z/OS system logger, which provides a programming interface to access records on a log stream. The CICS system log, forward recovery logs, autojournals, and user journals map into specific log streams. Log streams are defined in structures within the coupling facility or as DASD-only log streams.

DASD-only logging was introduced with CICS TS 1.2 and can be used for single-Image and non-Parallel-Sysplex customers. DASD-only log streams do not use coupling facility storage. For a DASD-only log stream, the log blocks span storage in local storage buffers and DASD log data sets. A DASD-only log stream has a single system scope. Only one system at a time can connect to DASD-only log stream. The MVS system logger programming interface can be used to merge data from multiple instances of an application, including from different systems across a Parallel Sysplex.

Note that DASD-only logging can be used in a Parallel Sysplex, but merging is supported within one MVS image only. For example, suppose you concurrently run multiple instances of an application in a Parallel Sysplex, and each application instance can update a common database. It is important for your installation to maintain a common log of all updates to the database from across the Parallel Sysplex, so that if the database must be damaged, it can be restored from the backup copy. You can merge the log data from applications across the Parallel Sysplex into a log stream, which is simply a collection of data in log blocks residing in a coupling facility list structure, on DASD, or both.

2.5.4 DASD only log streams

DASD-only log streams do not use the coupling facility storage. For a DASD-only log stream, the log blocks span storage in local storage buffers and DASD log data sets. A DASD-only log stream has a single system scope. Only one system at a time can connect to a DASD-only log stream. Multiple applications from the same system can, however, simultaneously connect to a DASD-only log stream. When a system logger application writes a log block to a DASD-only log stream, the system logger writes it first to the local storage buffers for the system and then automatically duplexes it to a DASD staging data set associated with the log stream. A DASD-only log stream can be upgraded to a coupling facility log stream simply by updating the log stream definition in the LOGR policy couple

data set to associate a coupling facility structure with the log stream. If the staging data set fills up to its defined HIGHOFFLOAD value, the system logger begins the offload process. Both a DASD-only log stream and a coupling facility log stream can have data in multiple DASD log data sets. As a log stream fills log data sets on DASD, the system logger automatically allocates new log data sets for the log stream.

2.5.5 Coupling Facility log streams

A coupling facility log stream spans two levels of storage: the coupling facility for interim storage and DASD log data sets for more permanent storage. When the coupling facility space for the log stream fills, the data is offloaded to DASD log data sets. A coupling facility log stream can contain data from multiple systems, allowing a system logger application to merge data from systems across the Parallel Sysplex. The main difference between coupling facility log streams and DASD-only log streams is the storage medium that the system logger uses to hold interim log data: In a coupling facility log stream, interim storage for log data is in the coupling facility list structures.

In a DASD-only log stream, interim storage for log data is in the local storage buffers on the system. Local storage buffers are data space areas associated with the system logger address space.Your installation can use coupling facility log streams, DASD-only log streams, or a combination of both types of log streams.

2.5.6 CICS implementation of the z/OS logger

The CICS system log is implemented as two MVS system logger logstreams. One stream is the primary system log stream, DFHLOG, which holds data for most normal (short-lived) in-flight UOWs. The other stream is the secondary system logstream, DFHSHUNT, which holds information for UOWs that are not short-lived, typically UOWs that cannot complete because of back-out failures, or because they are designed as long-running tasks that issue infrequent sync points. The two logstreams can be defined as DASD-only log streams or as coupling facility logstreams. See Example 2-6 that shows how to use program IXCMIAPU to define DFHLOG and DFHSHUNT logstreams.

Example 2-6 Define logstreams for CICS

```
//*
//IXCMIAPU EXEC PGM=IXCMIAPU
//STEPLIB DD DSN=SYS1.MIGLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
```

```
DATA TYPE(LOGR) REPORT(YES)
DEFINE LOGSTREAM
NAME(SCSCPAA9.DFHLOG)
DASDONLY(YES)
MAXBUFSIZE(64000)
HIGHOFFLOAD(95)
LOWOFFLOAD(20)
DEFINE LOGSTREAM
NAME(SCSCPAA9.DFHSHUNT)
DASDONLY(YES)
MAXBUFSIZE(64000)
HIGHOFFLOAD(80)
LOWOFFLOAD(0)
/*
```

2.5.7 CICS Log tail trimming process

CICS manages the system log by deleting records, for completed units of work, during activity keypoint processing (log-tail deletion). With an appropriately sized log stream, the system log data remains in primary storage, avoiding data spilling to DASD.

In a production environment, it is important to have an *activity keypoint* frequency of approximately one activity keypoint every one-to-two minutes during peak time. An activity keypoint triggers the log tail trimming process; therefore, you can avoid writing too many log records between syncpoints. You can define the activity keypoint frequency using the system initialization parameter AKPFREQ. Actually, you specify the required number of write requests to the CICS system log stream output buffer before CICS writes an activity keypoint.

Activity keypoints in action

Figure 2-13 on page 71 takes you through process that CICS is using to keep the system log in shape. The goal is to avoid data spilling to DASD because it is expensive to retrieve UOWs from DASD to commit them. CICS logger performance can be analyzed using the MVS logger SMF records type 88.

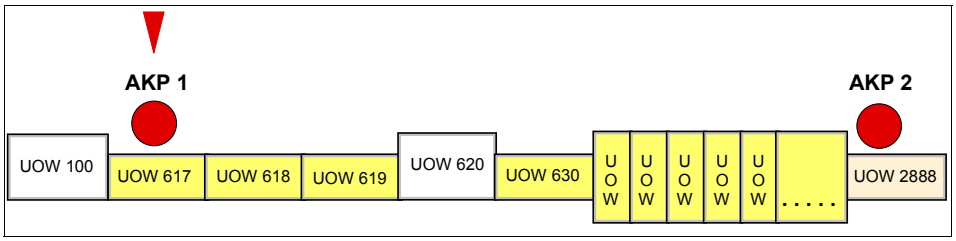


Figure 2-13 Log tail trim process - at AKP 1

At AKP 1 (activity keypoint) we have a long running UOW (UOW 100). This UOW touches recoverable resources and issued a sync point before AKP1. UOW 617 is currently running.

In Figure 2-14, CICS continues until activity keypoint 2.

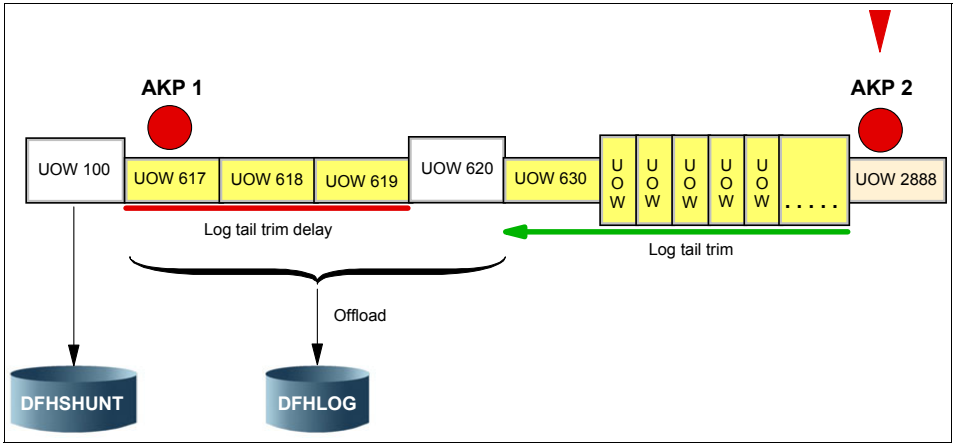


Figure 2-14 Log tail trim process at AKP 2

Now, we are at AKP 2. UOW 100 has not yet ended, but it did not touch any of the recoverable resources. This UOW is moved to DFHSHUNT.

Several UOWs (the ones with a yellowish background) ended and therefore committed their work. The UOWs that got started and committed after the begin of UOW 620 (a new long running UOW that has not yet ended) can be deleted from interim storage. Log tail trimming can occur for theses UOWs.

Other UOWs (the ones with a grey background) can also be deleted from interim storage. However, UOW 620 prevents this from happening.

Long running UOWs prevent log tail deletion. This means that if we have several long running UOWs it is quite possible, that offloading must occur several times.

Many offload datasets must be allocated. These allocations are called DASD shifts.

CICS continues to AKP 3, as shown in Figure 2-15.

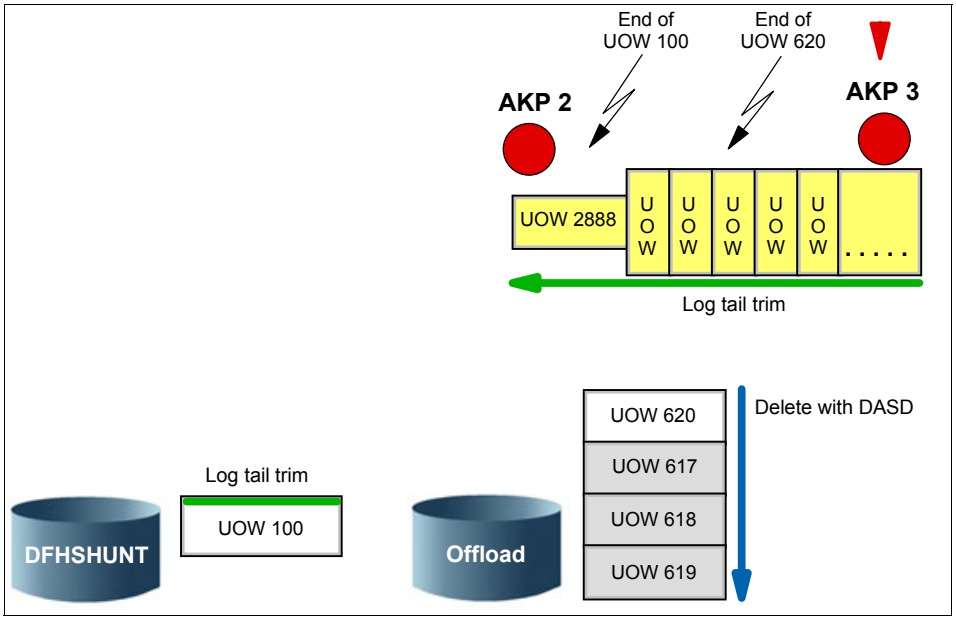


Figure 2-15 Log Tail trim process at AKP 3

When we reach AKP3, a log tail trim occurs for all the finished UOWs. Between AKP 2 and AKP3 (the long running) UOWs 100 and UOW 620 ended, and the logged information is no longer needed. UOW 100 got shunted; therefore, its information about DFHSHUNT gets deleted. UOW 617, 618, 619, and 620 had (at AKP 2) to be offloaded. Now that the UOW (UOW 620) that prevented log tail trim at AKP 2 ended, the information for all these UOWs on the OFFLOAD dataset can be deleted. This requires additional input/output (I/O) and processing time.

Considering these brief explanations, the objective must be to have:

- No OFFLOADs for CICS log streams
- Very few DASD shifts
- A low number of Deletes with DASD

2.6 CICS exits

In this section, we briefly describe what CICS exits are and how the bank's system programmers can use them to tailor the individual requirements for the organization.

CICS provides a wide range of customizing facilities that you can use to set up and run your environment. CICS is designed to suit the needs of all organizations, but there is still the requirement to adapt individual logic to the CICS internal logic flow.

CICS provides two extremely efficient facilities that allow additional customizing capabilities:

- CICS Global User Exits (GLUE)
- CICS Task Related User Exits (TRUE)

2.6.1 Global User exits

You can use the CICS global user exit points, in conjunction with programs of a special type that you write yourself (global user exit programs), to customize your CICS system.

A global user exit point, sometimes referred to as a global user exit, is a place in a CICS module or domain at which CICS can transfer control to a global user exit program that you wrote and at which CICS can resume control when your exit program finishes its work.

Each global user exit point has a unique identifier, and is located at a point in the module or domain at which it can be useful to do some extra processing. For example, at exit point XSTOUT in the statistics domain, an exit program can be given control before each statistics record is written to the SMF data set and can access the relevant statistics record. You might want to use an exit program at this exit point to examine the statistics record and suppress the writing of unwanted records.

Global user exit support is provided automatically by CICS. However, there are several conventions that govern how you write your exit program, which we describe in the CICS documentation. Because global user exit programs work as if they are part of the CICS module or domain, there are limits on the use you can make of CICS services. Most global user exit programs cannot use EXEC CICS commands, but can invoke some CICS services using the exit programming interface XPI. Example 2-7 on page 74 shows a code snippet from an XPI monitoring call. The monitoring XPI in the example call is part of a gloabal user

exit that gets control before and after each EXEC CICS call. The global user exit program is driven at exit points XEIIN and XEIOUT. You can specify the name of the global user exit program on an EXEC CICS ENABLE COMMAND to connect your exit program to the exit point.

Example 2-7 global user exit XPI monitoring call

```
*****************************************************************
* INQUIRE MONITORING DATA THAT HAS BEEN COLLECTED FOR THE TASK ..*
* .. SO FAR                                                     .*
*****************************************************************
         L     R8,=F'2200'         length of monitoring buffer
         USING DFHMNMN_ARG,R5      SET UP BASE REGISTER
         L     R13,UEPSTACK
         DFHMNMNX CALL,
               CLEAR,
               IN,
               FUNCTION(INQUIRE_MONITORING_DATA),
               DATA_BUFFER((R6),(R8),*),
               OUT,
               RESPONSE(*),
               REASON(*)
*
         DROP  R5
*
```

2.6.2 Task-related user exits

A *task-related user exit* allows you to write your own program to access a resource, such as a database, that is not otherwise available to your CICS system. CICS itself also uses a task-related user exits. For example task-related user exit DSNCSQL is used to access DB2 when a CICS application program issues a SQL call.

If your own program is using a task-related user exit to access a resource, such a resource is known as a *non-CICS resource*. The exit is said to be task related because it becomes part of the task that invoked it and because, unlike a global user exit, it is not associated with an exit point. You do not have to use any of the task related user exits, but you can use them to extend and customize the function of your CICS system according to your own requirements.

The most common use of a task related user exit is to communicate with a resource manager external to CICS, for example, a file or database manager. The CICS interface modules that handle the communication between the

task-related user exit and the resource manager are usually referred to as the resource manager interface (RMI) or the task related user exit interface.

CICS itself uses the RMI to access WebSpere MQ, DB2, DBCTL, and TCP/IP.

The task-related user exit mechanism is known as an adapter because it provides the connection between an application program that must access a non-CICS resource and the manager of that resource.

The adapter is made up of three or more locally written programs. These are a stub program, a task related user exit program, and one or more administration routines or programs. For example, Figure 2-16 shows how CICS uses a task-related user exit to access DB2. Your CICS DB2 user application requires the inclusion of a task-related user exit stub DSNCLI. This stub uses the CICS RMI to call the task related user exit DSNCSQL, which provides access to DB2.

DB2 Attachment – Thread TCBs

```
//LKED.SYSIN  DD  *
   MODE RMODE(ANY),AMODE(31)
   INCLUDE CICSLOAD(DSNCLI)
         NAME  SQLPRO1(R)
```

- You include stub DSNCLI during the link edit process of your CICS / DB2 application

```
DSNCLI      CSECT
DSNCLI      AMODE 31
DSNCLI      RMODE ANY
            PRINT GEN
DSNHLI      DFHRMCAL TO=DSNCSQL
            PRINT GEN
DSNWLI      DS  0D
            BALR 15,0
            B  8(15)
§§HX80B  DC XL4'80000000'
            O  1,4(15)
            DFHRMCAL TO=DSNCSQL
            END
```

- During the process of the first SQL control is passed to stub DSNCLI
- The stub calls DSNCSQL in order to access DB2

Figure 2-16 Sample of CICS task related user exit

2.7 CICS security

One of the top priorities of our bank is data security in all aspects. In this section, we describe the security concepts of CICS that the bank can use.

Today, an irresistible growing number of users are completely dependent on their systems and on the data managed by those systems. There is now access to CICS data from many users in several locations to your organization. At the same time, easy-to-use, high-level inquiry languages are available, and there is much greater familiarity with data processing methods. This means that more and more people can use their system to retrieve or modify data that is stored within an IT environment. The speed, flexibility, and size of modern systems make large quantities of data accessible to many terminal users.

As the systems become easier to use, there is also more scope for front-end users to gain access to confidential or valuable data. Without a corresponding growth in awareness of good data security practices, these advances can result in accidental (or deliberate) data exposure. This means that your data can be subject to:

- Unauthorized access
- Disclosure
- Modification
- Destruction

As an online transaction-processing system (often supporting many thousands of terminals and front-end users), CICS clearly needs the protection of a security system to ensure that the resources to which it manages access are protected and secure from unauthorized access.

In a CICS environment, the assets you normally want to protect are the application programs and the resources that are accessed by the application programs.

To prevent disclosure, destruction, or corruption of these assets, you need a security concept to control the access to the CICS region and to the many CICS components. You can limit the activities of a CICS user to only those functions that the user is authorized to use, by implementing one or more of the following CICS security mechanisms:

- Transaction security: Ensures that users who attempt to run a transaction are entitled to do so.
- Resource security: Ensures that users who use CICS resources, such as files and transient data queues, are entitled to do so.

- Command security: Ensures that users who use CICS system programming commands are entitled to do so.
- Surrogate security: Ensures that a surrogate user is authorized to act on behalf of another user.

When CICS security is active, requests to attach transactions and requests by transactions to access resources are associated with a user ID. When a user makes a request, CICS calls the external security manager, for example, RACF, to determine if the user ID has the authority to make the request.

If the user ID does not have the correct authority, CICS denies the request. In many cases, a user is a human operator interacting with CICS through a terminal or a workstation. However, the user can also be a web browser user or, in a Web services solution, a program executing on a client system.

User IDs

When a human operator signs into a CICS region at the start of a terminal session, the person is asked to provide a user ID and a password. The user ID remains associated with the terminal until the terminal operator signs out. Transactions executed from the terminal and requests made by those transactions are associated with that user ID.

For connections from Web users, there are other ways in which the user of a CICS transaction can be identified, to include situations where:

- A web browser user is challenged to provide Hypertext Transfer Protocol (HTTP) basic authentication information (a user ID and a password). The transaction that services the client's request and other requests made by that transaction are associated with that user ID.
- A client program that is communicating with CICS using the SSL supplies a client certificate to identify itself. The security manager maps the certificate to a user ID. The transaction that services the client's request and other requests made by that transaction are associated with that user ID.

In addition to these transport-level authentication mechanisms, Web service clients can also pass authentication data in the SOAP message.

CICS special user IDs

There are two particular user IDs that CICS uses in addition to those that identify individual users:

- Region user ID: The CICS region user ID is used for checking the authorization when the CICS system, rather than an individual user of the system, requests access to system resources, such as CICS data sets and other servers.

- Default user ID. When a user does not sign in, CICS assigns a default user ID to the user. It is specified in the system initialization table (SIT) parameter DFLTUSER. In the absence of more explicit identification, it identifies TCP/IP clients that connect to CICS, providing little authority to the default user ID.

2.7.1 CICS Security concepts using RACF

CICS uses an external security manger, such as RACF, to manage resources and users from the security perspective.

History

In CICS releases earlier than CICS ESA 3.2, there was an additional security option available, the so-called CICS internal security. User-related security information was defined in a macro called Sign On Table (DFHSNT). CICS internal security provided basic CICS security services tailored to fit the needs of most shops in the early 80s.

Today

CICS uses an external security manger, such as RACF. The basic concept of RACF security management for CICS is:

- CICS provides RACF *resource classes* for each resource where CICS is going to call the external security manager for. For example, you can have resource classes for transactions, programs, commands,
- To utilize resource classes to protect CICS resources, you can define *profiles* to the resource class, for example, you can define a profile to the transaction resource class. If you want to protect CICS system transactions, such as CEMT, CEDA, CEDB, and CRTE, from being used by unauthorized users, define a profile named SYSADM. You can use RACF commands to add members, or in this case transaction names, to the profile.
- Now we created a profile SYSADM that contains a number of CICS system transactions, shown in Example 2-8 on page 79. To allow CICS system programmers, or other users that require access to the profile resources, to start the transactions we use a supplied RACF command to permit users to the profile. You can permit NONE, READ, UPDATE, CONTROL, ALTER, or EXECUTE access to users.

You can define as many profiles as you want to a resource class. You can also define the same users or group of users to more than one profile of the same type.

Example 2-8 RLIST command to display profile SYSADM

```
CLASS      NAME
-----      ----
GCICSTRN   SYSADM

MEMBER CLASS NAME
------ ----- ----
TCICSTRN

RESOURCES IN GROUP
--------- -- -----
CDBC
CEDA
CEMT
CESD
CETR
CIND

LEVEL  OWNER      UNIVERSAL ACCESS  YOUR ACCESS  WARNING
-----  --------   ----------------  -----------  -------
 00    KLEIN4     NONE                    READ    NO

INSTALLATION DATA
-----------------
```

2.7.2 Passwords and pass phrases

Over the years, many customers have for improvements and modernization of user ID and password processing. Customer perception is that longer passwords improve system security.

In z/OS Release 8, RACF added the infrastructure for password phrases, as an alternative to passwords greater than eight characters. This addition enables applications that exploit this new infrastructure to specify an authentication value longer than eight characters to better fit in the heterogeneous world, while traditional applications continue to use the traditional password.

2.7.3 CICS and identity propagation: The end-to-end Security solution

Figure 2-17 on page 80 shows the current implementation of user identity and mapping.

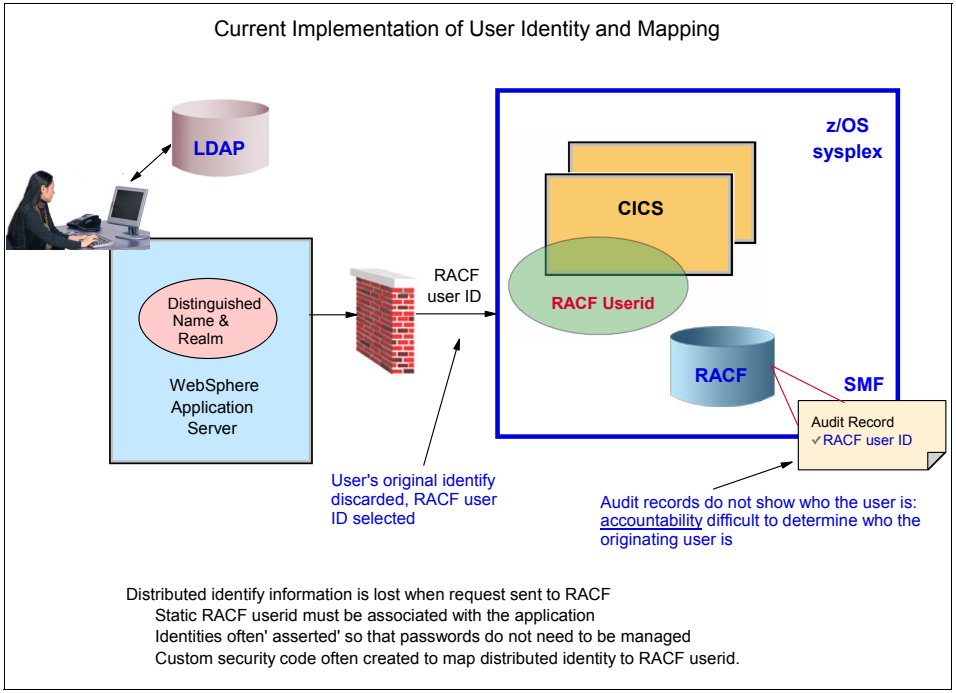


Figure 2-17 Current implementation of user identity and mapping

Identity propagation is a new identity assertion capability provided by z/OS V1R11 and CICS Transaction Server V4.1. Together with new functions in WebSphere DataPower® or the CICS Transaction Gateway, it supports a cross-platform, end-to-end security solution, providing for identity assertion, control, and auditing.

Modern enterprise information processing systems typically consist of multiple software components, each of which perform distinct services. Over time, these services tend to be reused in other topologies. This often introduces the challenge of how to provide secure access between middleware components that use disparate security technologies.

When considering IBM z/OS, any application that requires access to data usually must provide a valid credential for authentication and subsequent authorization through the IBM RACF security manager. The security strengths that z/OS provides can then become an issue if it is necessary to authorize requests that originate from a remote security registry that is unknown to RACF. The solution was often to provide a form of identity assertion, where a RACF identity can be asserted without a password check. This scenario typically requires a level of

predefined trust between the two systems. With IBM CICS TS, various techniques are possible, including:

- Use of a predefined link user ID
- Dynamic mapping of Secure Sockets Layer (SSL) client certificate identities to RACF user IDs
- Delegation through third-party identity management products, such as IBM Tivoli Federated Identity Manager.

No matter what solution is used for identity assertion, the process of mapping distributed user identities to a RACF identity has, until now, been a one-way function, resulting in the loss of the original distributed identity after the mapping occurs. Although effective, such mapping solutions have several issues, including lack of end-to-end accountability, inflexibility, and loss of control. Identity propagation, illustrated in Figure 2-18, addresses these issues by allowing the z/OS security administrator to create a set of flexible rules, stored in the RACF database, which ensure that the distributed identity persists after the mapping stage and remains visible for operational support and further auditing if required.

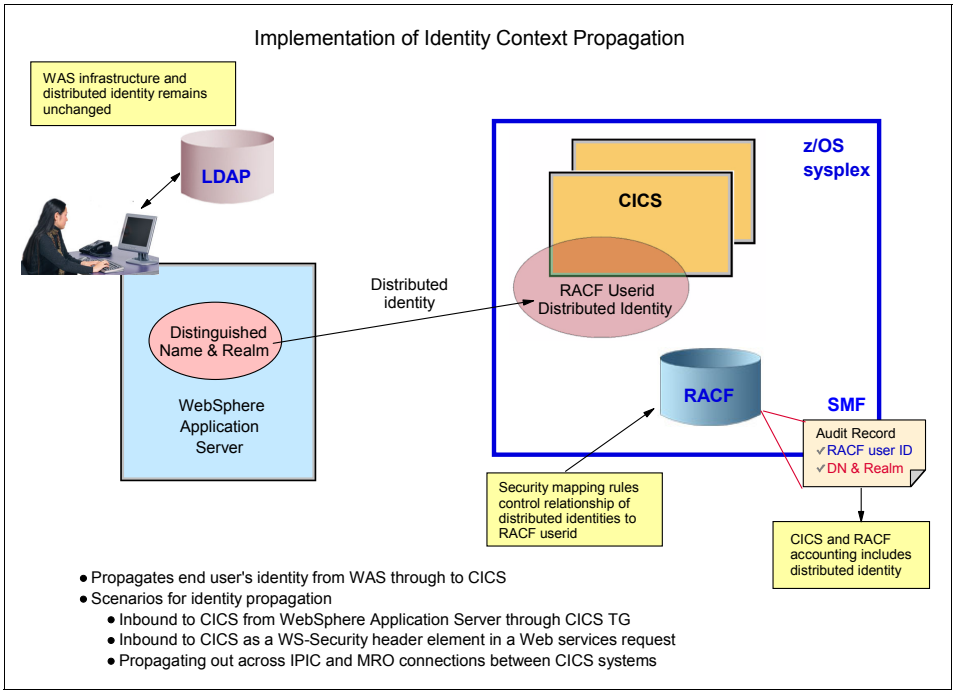


Figure 2-18 Implementation of identity propagation

The identity propagation function provided with RACF in z/OS V1R11 can be exploited when using CICS TS V4.1 with a set of enabling APARs: PK83741,

PK95579, and PM01622, along with an additional APAR PK98426 for IBM CICSPlex SM. If Web service requests are being sent to CICS, then an IBM WebSphere DataPower appliance can be used to propagate the distributed identity to CICS so it can be mapped to a RACF user ID. Alternatively, if Java EE (JEE) components are using the CICS JEE Connector Architecture (JCA) resource adapter to call CICS applications, CICS TG V8 can be used to propagate the distributed identity.

2.8 Monitoring the CICS System

Figure 2-19 illustrates CICS monitoring.

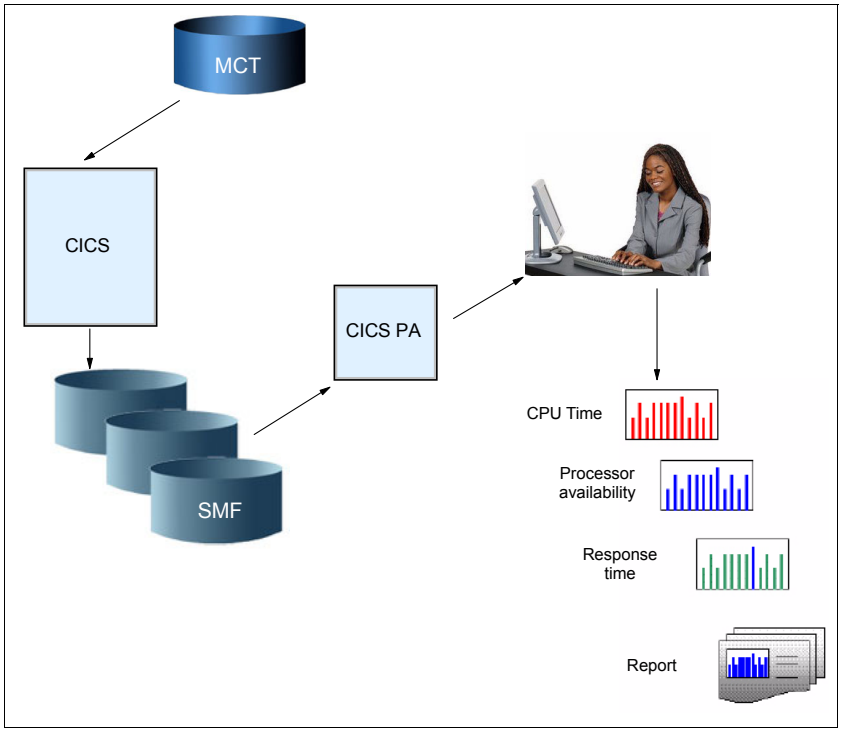


Figure 2-19 CICS monitoring

2.8.1 CICS monitoring

In these days, accounting and capacity planing is more and more important for our bank. CICS System programmers must observe performance factors, such as transaction throughput, response times, and CPU consumption every day.

Customers must run their CICS environment efficiently to assure given service level agreements and to keep cost within the budget.

Service providers are expected to increase the availability of CICS for internal and external customers. Planning to implement a performance and service level strategy is strongly recommended. Frequent observations of the CICS performance records is required to reduce system failures due to performance and capacity bottlenecks. You aim to achieve shorter response times, which means better service and thus increased customer satisfaction

2.8.2 Monitoring CICS transactions

The performance of CICS transactions can be monitored using real time monitoring or using SMF records that represent a specific day or time interval in the past.

Analysis of historical CICS performance data

During the process of CICS transactions, performance data is frequently accumulated in an internal monitoring area that belongs to the CICS task.When the transaction terminates CICS writes the contents of the monitoring area to the (SMF) System Facility Management record buffer. You can analyze the performance of CICS transaction by investigating the CICS SMF performance records.

You can utilize tools, such as the CICS Performance Analyzer (PA), to investigate SMF records. The basic idea to process CICS performance records is to run a tool that formats SMF records for a specific day or time interval. After that, you can download the resulting output file to your workstation to analyze the performance of transactions. You can use your preferred tool to manage spread sheets to create charts and tables to present the results of your analysis.

Why we use historical data

Investigating CICS performance data for an entire day makes it easier to investigate and document throughput and performance, for example, during peak time.

You can use your own established procedures to expose trends by looking at the data from the same perspective every day. If you see increased response time during the peak period, it is easier to investigate the cause using historical data. To come closer to what caused increased response time, you can create a comparison chart that shows the behavior of important performance indicators at the same time. Important performance indicators can be Processor availability or QR TCB constraints.

Figure 2-20 shows a comparison chart. The chart shows that response time peaks were caused by an insufficient CPU/Dispatch ratio. The number of tasks that CICS processed decreased considerably at the same time.

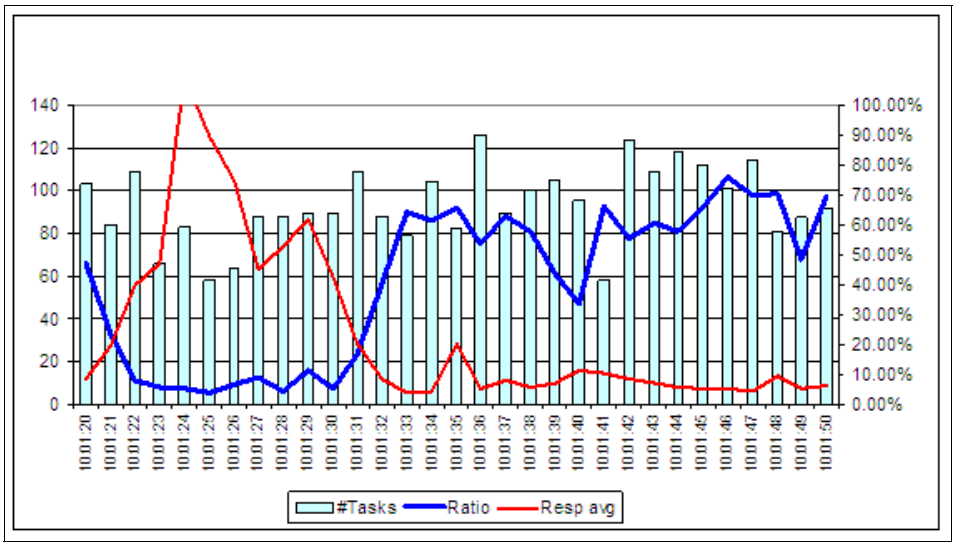


Figure 2-20 Comparison graphic

2.8.3 CICS SMF records 110

When CICS is running and the CICS Monitoring Facility is active, data is collected and written to the MVS System Management Facility (SMF) data set as type 110 records. SMF records can be written to SMF data sets or to MVS logstreams. SMF records 110 can be of several types:

1. Performance class records: Performance class data provides detailed resource-level data that can be used for accounting, performance analysis, and capacity planning. This data contains information relating to individual task resource usage and is completed for each task when the task terminates. This information can be used periodically to calculate the charges that are applicable to different tasks. If you want to set up algorithms for charging users for resources used by them, you can use this class of data collection to update the charging information in your site's accounting programs.
2. Exception class records: An exception record is created each time any of the resources covered by CMF exception class monitoring becomes constrained by system bottlenecks. Performance class data can be linked to the exception class data by the transaction ID and the UOW fields in each type of record. An SMF exception record is created for events, such as wait for storage, wait for

temporary storage, and wait for file resources, such as buffer and strings, or wait for Coupling facility data table requests.

3. Resource class records: Transaction resource class data provides additional transaction-level information about individual Files and Temporary Storage Queues used by a transaction

The SMF data is subsequently analyzed offline using tools, such as CICS PA.

In a typical CICS production environment, it is a good practice to collect CICS SMF records all the time. There is considerable cost to collect all sub types of SMF record 110 to the full extent. Therefore in a CICS production environment you probably just collect what is needed to carry out the daily accounting tasks.

Exception class and resource class records can be activated dynamically when they are needed, for example, in a CICS VSAM production environment you activate just performance class records. Performance class records tell you about response time or I/O wait time for your tasks, but you do not get information about the file name we wait for. Especially if you have tasks that process more than one file you need to activate SMF 110 resource class data to monitor specific resources.

2.8.4 Managing performance records

Figure 2-21 illustrates how to activate performance class monitoring.

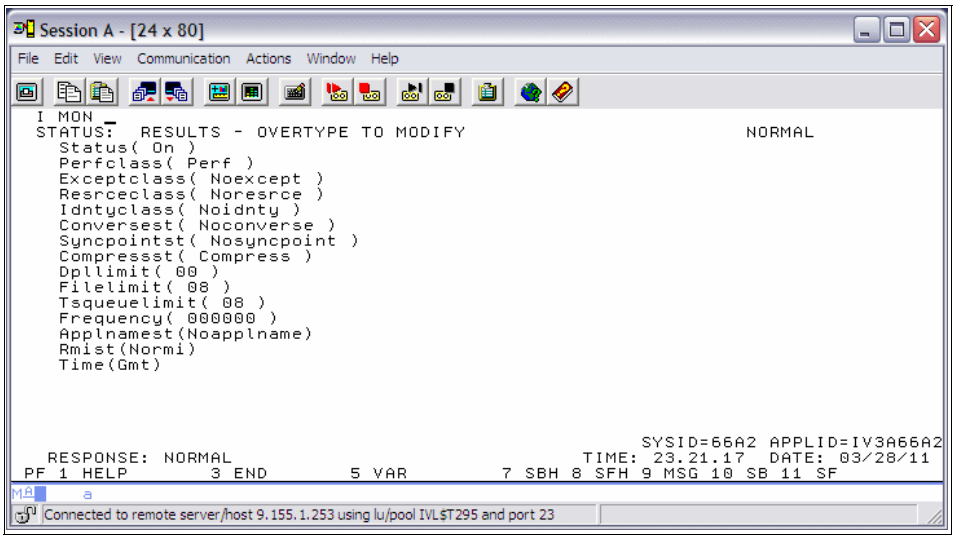


Figure 2-21 Activate performance class monitoring

As we discussed earlier, in a typical CICS production environment we usually just activate SMF 110 performance class records. CICS SMF performance class records are event driven. The CICS monitoring facility writes one SMF performance class record for each task when the task terminates.

The length of a SMF 110 record can be more than 2000 bytes. Therefore the number of transactions per day multiplied by the length of CICS SMF records results in considerable required dasd space and CPU time. It is therefore recommended to reduce the size of the SMF record.

You can use the CICS Monitoring Control Table (MCT) to EXCLUDE/INCLUDE parameters to reduce the size of the performance class record. You must ensure that the data fields that are required for your monitoring tools and accounting reports are not excluded.

If you are using the CICS Monitoring Control Table (MCT) EXCLUDE/INCLUDE parameters to reduce the size of the performance class record, you must ensure that the data fields that are required for some of the CICS PA reports and extracts are not excluded. SMF performance class records are organized in groups and performance data fields per group.

The size of the performance class record can be reduced by excluding an entire group and including required data fields. Example 2-9 shows a typical MCT that is often used to reduce the size of performance class records in production environments.

Another way to reduce the size of performance class records is data compression. CICS performs data compression, by default, on the SMF 110 monitoring records produced by the CICS monitoring facility. Data compression can provide a significant reduction in the volume of data written to SMF. The records are compressed and expanded using standard z/OS services.

Example 2-9 Optimized monitoring control table

```
//DFHSIT  EXEC C66AUPLE,NAME=SDFHAUTH                                  00010000
//ASSEM.SYSUT1 DD *                                                    00020000
//ASM.SYSUT1   DD UNIT=SYSALLDA,SPACE=(10000,(400,400))                00030000
         SPACE 3                                                      42000000
         DFHMCT TYPE=INITIAL,RMI=YES,                                *43000003
               SUFFIX=IT,                                            *44000001
               STARTER=YES                                            45000000
         EJECT                                                        46000000
*                                                                     46400000
         DFHMCT TYPE=RECORD,                                         *46800000
               CLASS=PERFORM,                                        *47200000
               EXCLUDE=(DFHCBTS,DFHDATA,DFHDOCH,DFHWEBB),            *47600000
               INCLUDE=(179,180,186,187,188,189)                      47800001
*                                                                     48000000
         DFHMCT TYPE=RECORD,                                         *49000000
               CLASS=PERFORM,                                        *50000000
               EXCLUDE=(DFHFEPI,DFHMAPP,DFHJOUR),                    *51000001
               INCLUDE=(10,156,151,152,153,154,155,157,158,159,58,172)  52000001
*                                                                     54000000
```

```
         DFHMCT TYPE=RECORD,                                          *54010001
               CLASS=PERFORM,                                         *54020001
               EXCLUDE=(DFHSYNC),                                     *54030001
               INCLUDE=(60,173,177,196)                                54040001
*                                                                      54050001
         DFHMCT TYPE=RECORD,                                          *56000000
               CLASS=PERFORM,                                         *58000000
               EXCLUDE=(DFHSOCK,DFHDEST),                             *60000001
               INCLUDE=(41,42,43,91,101,241,242,243)                   62000001
*                                                                      62001001
         DFHMCT TYPE=RECORD,                                          *62010001
               CLASS=PERFORM,                                         *62020001
               EXCLUDE=(DFHTERM,DFHTEMP),                             *62030001
               INCLUDE=(11,44,46,47,92,178,9,100,111,133,134,197,198)  62040001
*                                                                      64000000
         DFHMCT TYPE=RECORD,                                          *64020001
               CLASS=PERFORM,                                         *64030001
               EXCLUDE=(DFHFILE),                                     *64040001
               INCLUDE=(36,37,38,39,40,63,93,174,175,176)              64050001
*                                                                      64060001
         DFHMCT TYPE=RECORD,                                          *66000000
               CLASS=PERFORM,                                         *68000000
               EXCLUDE=(DFHPROG),                                     *70000000
               INCLUDE=(71,113,114,55,56,73)                     §L6C 72000001
*                                                                      74000000
         DFHMCT TYPE=RECORD,                                          *76000000
               CLASS=PERFORM,                                         *78000000
               EXCLUDE=(59,64,66,82,109,124)                     §L8C 78600001
*                                                                      79200000
         DFHMCT TYPE=RECORD,                                          *79800000
               CLASS=PERFORM,                                         *80400000
               EXCLUDE=(132,163,164,166,192)                     §L8C 81100001
*                                                                      81800000
         DFHMCT TYPE=RECORD,                                          *82500000
               CLASS=PERFORM,                                         *83200000
               EXCLUDE=(193,194,195,251,253)                     §P1A 83900001
*                                                                      84600000
         DFHMCT TYPE=RECORD,                                          *85300000
               CLASS=PERFORM,                                         *86000000
               EXCLUDE=(273,275,285)                             §L5C 88000001
*                                                                      90000000
         EJECT                                                         92000000
         DFHMCT TYPE=FINAL                                             94000000
         SPACE 2                                                       96000000
         END   DFHMCTBA                                                98000000
//LNKEDT.SYSLMOD DD DISP=SHR,DSN=CICS66.PROD.LOADLIB                   98010000
```

You specify the name of the MCT in the system initialization table. The MCT can not be dynamically changed. So if you want to include/exclude additional fields, you must recycle CICS, for example, if you want to analyze DB2 elapsed time, you must specify RMI=YES on the MCT TYPE initial statement. IF RMI=YES is not specified, you only have RMI elapsed time available in the SMF record. DB2 RMI elapsed time is a subset of RMI ELAPSED TIME. Other subsystems, such as IMS, TCP/IP, WebSphere MQ, and CICSPlex SM, also report their elapsed time to RMI ELAPSED TIME. You must recycle CICS to add additional performance fields to the MCT.

2.8.5 The importance of collecting CICS statistics

CICS statistics contain information about the CICS system as a whole, for example, its performance and usage of resources. Statistics data is therefore, useful both for performance tuning and for capacity planning.

Statistics are collected during CICS online processing for later offline analysis. The statistics domain writes statistics records to a SMF data set. The records are of SMF type 110, subtype 0002 or a so-called statistic record.

Let us assume that you want to analyze the performance of the CICS DB2 attachment during the peak time from 10:00 to 12:00 am. To do that, it is essential to have both performance class records and static records available. Using the performance class records you can analyze, for example:

- CPU time of the CICS DB2 transactions. That is QR CPU Time and L8 CPU Time
- Response time of the CICS DB2 transactions
- RMI DB2 elapsed time per task
- DB2 connection wait time

Using the CICS 110 statistic records you can analyze, for example:

- DB2 thread limit and peak for pool threads
- DB2 thread limit and peak for DB2 entry threads
- Limit and peak of open TCBs (L8) used for DB2
- The number of signon requests
- Number of thread wait overflows for DB2Entry definitions
- Number of thread reuse requests per DB2Entry definition

2.8.6 Interval statistics

CICS interval statistics are gathered by CICS during a specified interval. You can specify the interval value using:

- The system initialization table
- CEMT SET STATISTICS command
- An application issues a EXEC CICS SET STATISTICS command

CICS writes the interval statistics to the SMF data set automatically at the expiry of the interval. Choose proper intervals.

In the example described in the previous section we investigate the performance during peak time from 10:00 -12:00 am. If you specify a statistic interval of 15 minutes, You get four statistic snap shots per hour, which allows you to create

charts that show critical statistic peak values every 15 minutes. Figure 2-22 shows request interval statistics.

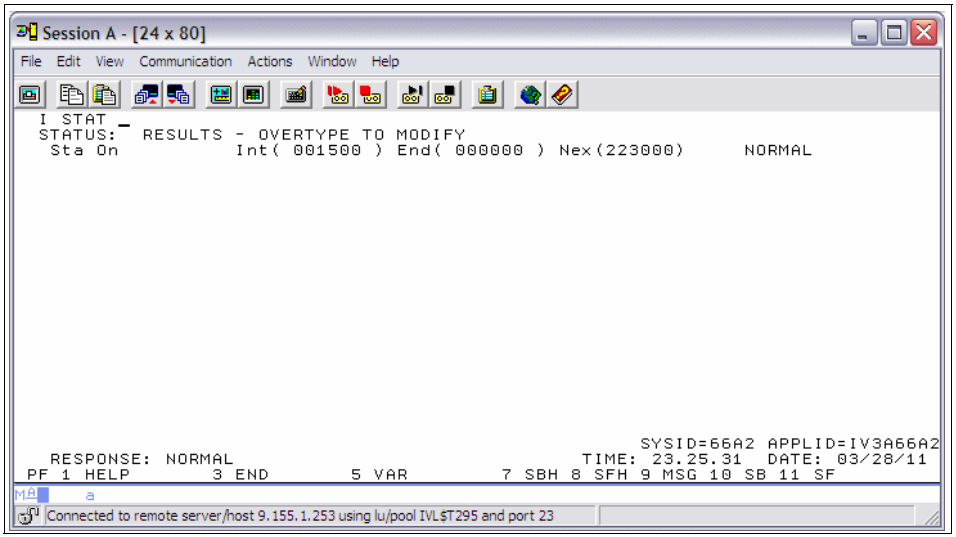


Figure 2-22 Request interval statistics

2.8.7 End of day statistics

End of day statistics are a special case of interval statistics where all statistics counters are collected and reset. Unlike interval statistics, you get a copy of end of day statistics when CICS terminates or by the time that you specify. End of day statics are used for the accounting and performance procedures that CICS system programmers periodically run. Interval statistics are used to investigate specific performance problems.

Figure 2-23 on page 90 shows a request for end of day statistics.

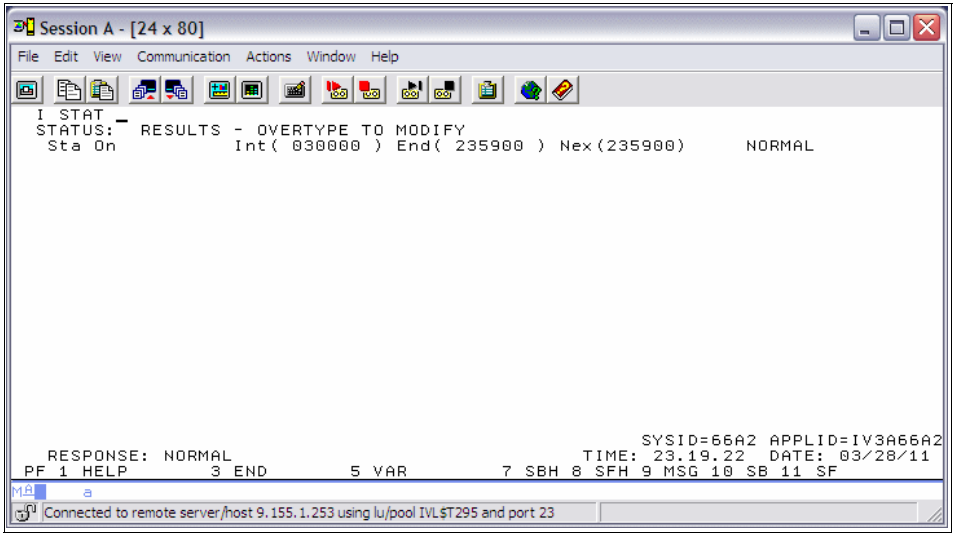


Figure 2-23 Request end of day statistics

2.9 CICS and the z/OS Workload Manager

Figure 2-24 on page 91 shows the basic concept of z/OS workload manager.

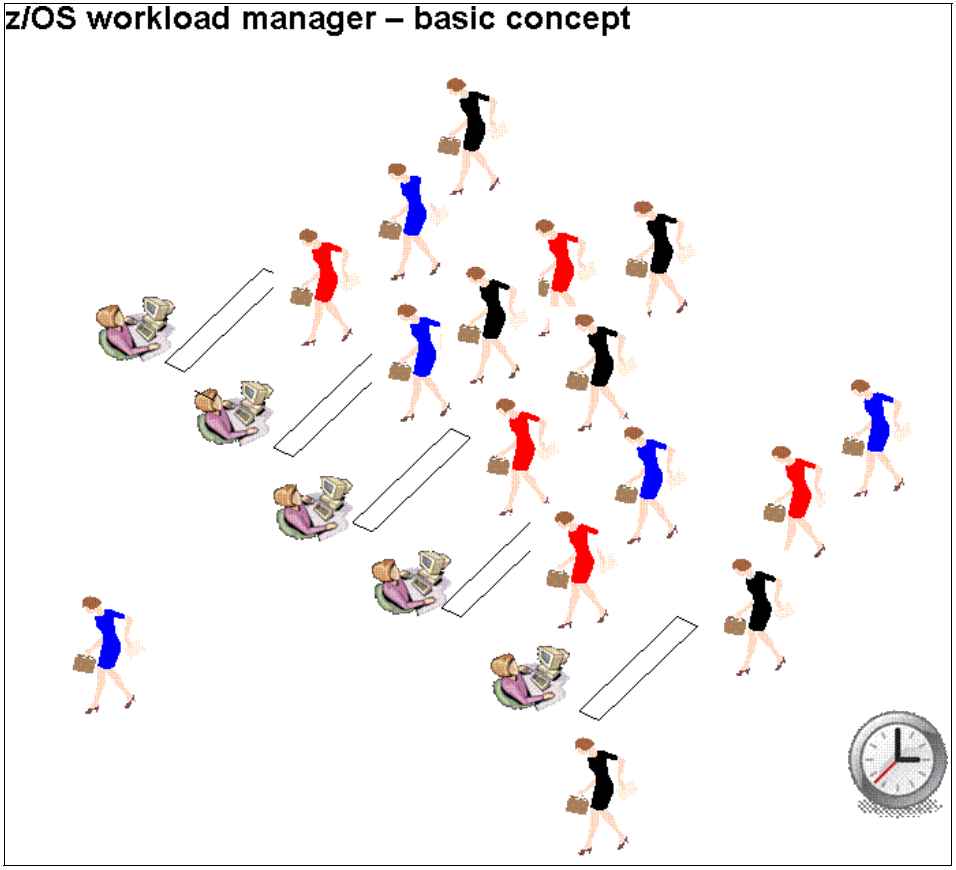


Figure 2-24 Basic concept of z/OS workload manager

The bank wants to benefit from the capabilities that the MVS workload manager provides. Our bank runs a 2 way parallel sysplex with 4 general processors as well as zIIP and zAAP processors. It is a requirement for the bank to use processor capacity and storage efficiently. The services provided by the MVS Workload manager can help the bank to run their CICSplex efficiently.

2.9.1 The need for z/OS workload management for CICS

In this section, we discuss the basic concept of the z/OS workload manager. Unlike CICSPlex SM WLM, which manages transactions, the z/OS workload manager cares about address spaces.

CICSPlex SM WLM manages transactions. CICSPlex SM WLM distributes transactions to the most suitable CICS region from the CICS perspective.

CICSPlex SM WLM indeed cannot manage the processor availability of CICS regions for example.

To understand the basic concept of the z/OS workload manager you can think of customers in a supermarket that rush to the cash desk. At times when just a few people pay for their groceries, you spend the expected time at the cash desk. Sure enough the market probably has less staff to operate cash desks. At times when many customers rush to the cash desk, you must spend more time to get served. It is obvious that the number of customers is the crucial factor of how much time we have to spend to get served.

As a resulting action the number of customers at the cash desks can be observed. To avoid waiting queues, more cash desks can be dynamically opened. There also can be fast path cash desks for customers with less than 12 items. You can achieve optimization when you identify where contention might occur.

2.9.2 Overview of the z/OS Workload Manager

Installations today process several types of work with different completion and resource requirements. Every installation wants to make the best use of its resources and maintain the highest possible throughput, achieving the best possible system responsiveness. Workload management makes this possible.

One of the strengths of the zSeries® platform and the z/OS operating system is the ability to run multiple workloads simultaneously within one z/OS image or across multiple images.

The function that makes this possible is dynamic workload management, which is implemented in the Workload Manager component of the z/OS operating system. The idea of z/OS Workload Manager is to make a contract between the installation (user) and the operating system.

The installation classifies the work running on the z/OS operating system in distinct service classes and defines goals for them that express the expectation of how the work must perform. WLM uses these goal definitions to manage the work across all systems of a sysplex environment.

With workload management, you define performance goals and assign a business importance to each goal. You define the goals for work in business terms, and the system decides how much resource, such as CPU and storage, is given to it to meet the goal. Workload Manager constantly monitors the system and adapts processing to meet the goals.

2.9.3 z/OS Workload Manager terms and definitions

The basic idea of the sample that we described in the previous section can be used to understand how CICS and z/OS workload manager work together:

- Identify the CICS workload
- Record the time to run transactions
- Find out about resource contention

To identify your CICS workload, you must understand the importance and volume of your CICS regions and transactions.

WLM provides a mechanism to separate the work into distinct classes. The name of this is *work classification*, and it allows an installation to deal with different types of work that runs on the system.

The work classification and WLM observation of how the work uses the resources provide the base to manage the system. This management occurs based on the goals that the installation defines for the work. After classifying the work into distinct classes, the installation associates a goal with each class. The goal determines how much service the work in the class can receive. These classes of work are named *service classes*.

From the CICS perspective, the meaning of workload classification is to categorize the workload into production and test regions, for example. You can think of a service class as a subset of a workload using the same goal.

You can use the following goal types:

- Response time goal:
  - Percentile response time
  - Average response time
- Velocity goal:
  - Percentage of when the work wants to run, it is able to
- Discretionary:
  - Run the work when there are resources left to do so

Besides the goal, it is also necessary to define which work is more important than other work. In the case where several service classes do not achieve their

target goals, the importance helps WLM decide which service class it must help get resources.

- CICS Subsystem – Service Class examples …

Service Class:	CICSPROD
Description:	Production CICS Transactions
Response Time:	90% .35 Second
Importance:	1
Service Class:	CICSTEST
Description:	Test CICS Transactions
Response Time:	1.5 second AVG
Importance:	4
Service Class:	CICSDFLT
Description:	Default CICS Transactions
Response Time:	5 second AVG
Importance:	5

Figure 2-25 CICS service class examples

Figure 2-26 on page 95 describes the WLM constructs that are used for performance management. In the service policy, we have workloads. For each workload, we have service classes that indicate the goal and the importance of the associated work. We also have the classification rules to link a workload to a service class.

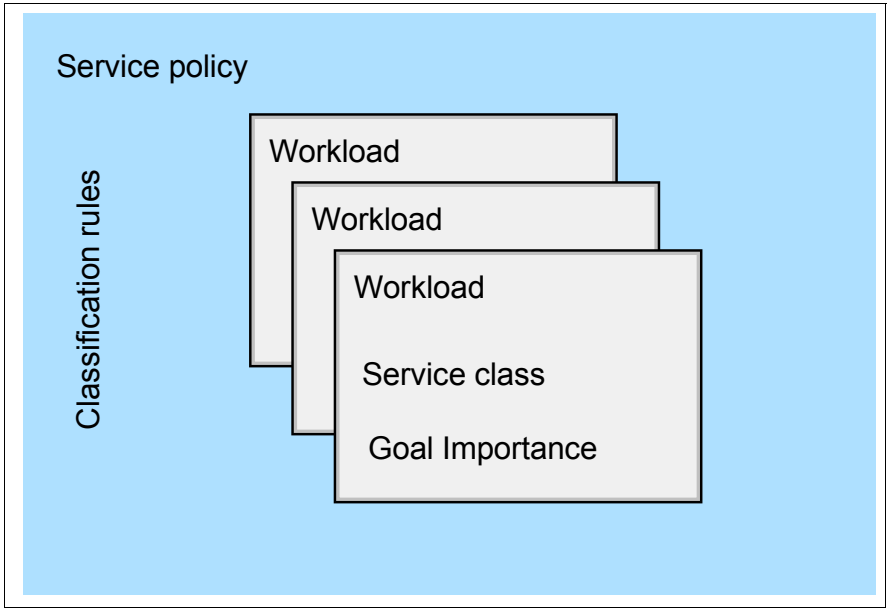


Figure 2-26 CICS WLM overview

2.9.4 Managing and monitoring the CICS workload

It is the CICS system programmer's responsibility to define realistic goals. If your goals are not aggressive enough, resources, such as central processor capacity, can fade away from the CICS address space. The CICS system programmer can periodically investigate the given service level agreements using RMF™ and CICS SMF 110 records. SMF records 110 can be used to observe CICS response time and CPU consumption. It is also a good practice to investigate the processor availability during peak time of your important CICS address spaces. Make sure CICS has enough processor capacity using the following formula:

```
QR CPU to Dispatch Ratio = QR CPU Time / QR Dispatch time
```

During peak time your CICS region must have a CPU to Dispatch Ratio of at least 80%. In 80%, when processor service is needed, CICS can get it.

2.10 CICS dumps and traces

Our bank needs to manage CICS problem documentation to provide it to IBM service staff or to the bank's application programmers.

In this section, we describe how to provide suitable problem determination documentation, such as dumps and traces, in various problem situations. We also discuss how to manage dumps and traces using the CICS provided facilities

2.10.1 CICS System programmers considerations

These days the CICS Transaction Server is still the workhorse of your high volume transaction workload. In a typical CICS production environment, transactions run reliably, and we do not expect them to come to an abnormal end (ABEND). Dumps and traces therefore are often suppressed in most organizations.

Furthermore it is costly and inadequate to run production environments with CICS dumps and traces active:

- Unexpected CICS system dumps are generated using operating system macros that issue supervisor calls (SVC). As a result CICS gets interrupted to take a snap shot of its memory. High volume CICS address spaces that run 100-200 tasks per second can suffer delays because of that.
- In production environments, transaction dumps are also mostly suppressed or limited to just a view occurrences. The process of taking many transaction dumps in a production environment can also influence the task throughput.
- The CICS internal trace increases the CPU consumption to a considerable extent; therefore, it is the best practice to reduce tracing to the minimum that is required.

In CICS test and development environments, it is recommended to enable system and transaction dumps and to run with CICS internal trace active.

CICS system dumps and transaction dumps are written unformatted to the appropriate dump data set. In other words, they are memory dumps of all or part of the CICS address space.

Unformatted dumps are not easy to interpret, and you are recommended not to use them for debugging. CICS provides utilities for formatting transaction dumps and CICS system dumps, and you must always use them before you attempt to read any dump. You can quickly locate areas of storage that interest you in a formatted dump, either by browsing it online or by printing it and looking at the hard copy.

CICS internal traces are formatted as part of the dump.

2.10.2 Transaction dumps

Figure 2-27 shows a transaction dump table.

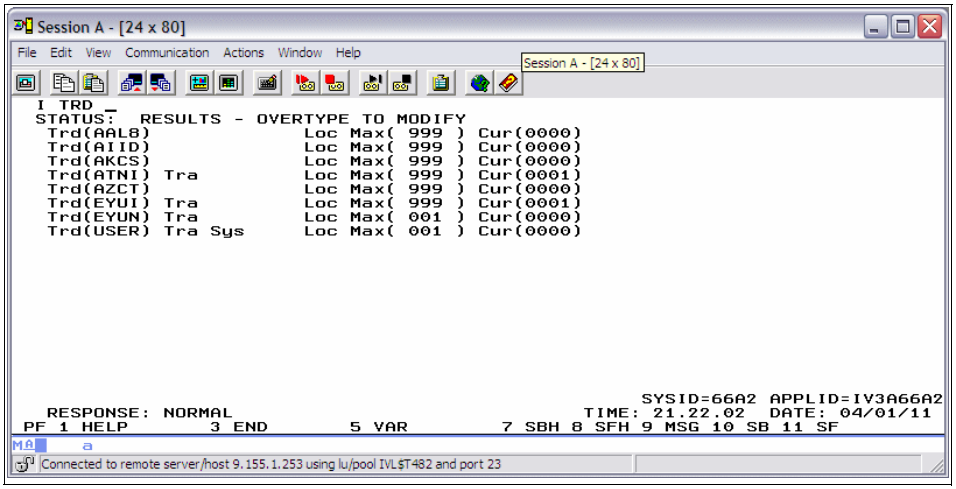


Figure 2-27 Transaction dump table

You can use the CEMT SET TRD command to manage CICS Transaction dumps. For some abend codes, such as time outs or other minor causes, you usually do not get transaction dumps printed by default. The Transaction dump table allows you to add abend codes to print transaction or even system dumps for them.

Transaction dumps are mostly used by developers to debug CICS user application programs. The contents of transaction dumps is tailored to fit the need of application programmers. Transaction dumps contain storage areas, program storage, and trace entries that you must use to fix the user application program. If the problem turns out to be some where else in the CICS logic, it is advisable to provide a CICS system dump.

You can format transaction dumps offline using the CICS dump utility program, DFHDU660. Individual transaction dumps must be formatted in their entirety, but you can control which dumps are formatted from the dump data set. You can also use the SCAN option with the dump utility program to get a list of the transaction dumps recorded on the specified dump data set.

You can select dumps to be formatted as follows:

- Those taken in a specific period of time
- Those associated with a specific transaction identifier

- Those taken for a specific transaction dump code
- Those having specific dump IDs--that is, those for specific CICS runs and dump count values.

2.10.3 CICS system dumps

Figure 2-28 shows a system dump table.

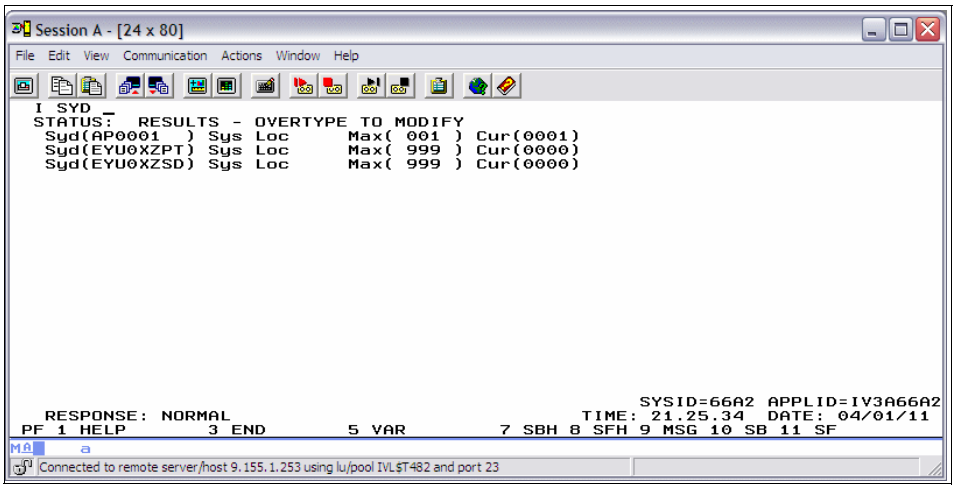


Figure 2-28 System dump table

As with transaction dumps you can manage CICS system dumps using a dump table. You can add abend codes or even CICS message numbers to produce system dumps. You can also limit the number of dumps you receive for an abend code.

A CICS system dump results from CICS issuing an operating system SDUMP macro. It includes almost the entire CICS address space, that is, the MVS nucleus and other common areas, as well as the CICS private storage areas. The SDUMP dump is written to an MVS dump data set, which you can process using the interactive problem control system (IPCS) or tools, such as the CICS Fault Analyzer.

System dumps for specific events

For the following events we get system dumps by default or CICS system programmer activity is required to produce suitable dumps:

- You get a message DFHAP0001 abend code AKEA, which indicates that a program check occurred in a CICS module. CICS provides a system dump for you. In this case, the system dump can be used to debug the problem. You

can use the trace to understand the history of the problem. All information you need to debug the problem is formatted by IPCS.

- CICS is eating a lot of CPU time and appears to loop. In this situation, no dump is provided by CICS. Suitable debugging information must be provided manually by the system programmer. The best practice is to activate a branch trace in z/OS, and take a dump using the MVS dump command. If you cancel CICS, you can get a system dump too but this action stops the loop too.
- You get a storage violation message DFHSM0102. CICS takes a system dump by default. The dump can be useful to fix the problem. Sometimes CICS storage violations are not immediately detected by CICS. If a task is suspended for some time and another active task wipes out storage check zones of the suspended task, it can take awhile until the violation gets detected. CICS provides an on board mechanism to detect violations earlier. In this case, you must activate the so-called storage checker and wait for the next occurrence of the problem.

2.10.4 CICS traces

The CICS tracing facility is extremely powerful and can be of outstanding help to understand the circumstances that led to the problem.

CICS trace records requests made by application programs to CICS for various services. Because this involves the recording of these requests each time they occur, the overhead depends on the frequency of the requests.

For a production system, background tracing might incur an unacceptable processing overhead. If you find this to be so, you are recommended to set up tracing so that exception traces only are recorded.

A trace table always exists and is used for recording exception conditions useful for any first failure data capture. Other levels of trace are under the control of the user.

2.10.5 CETR Transaction

Figure 2-29 on page 100 shows the CETR initial panel.

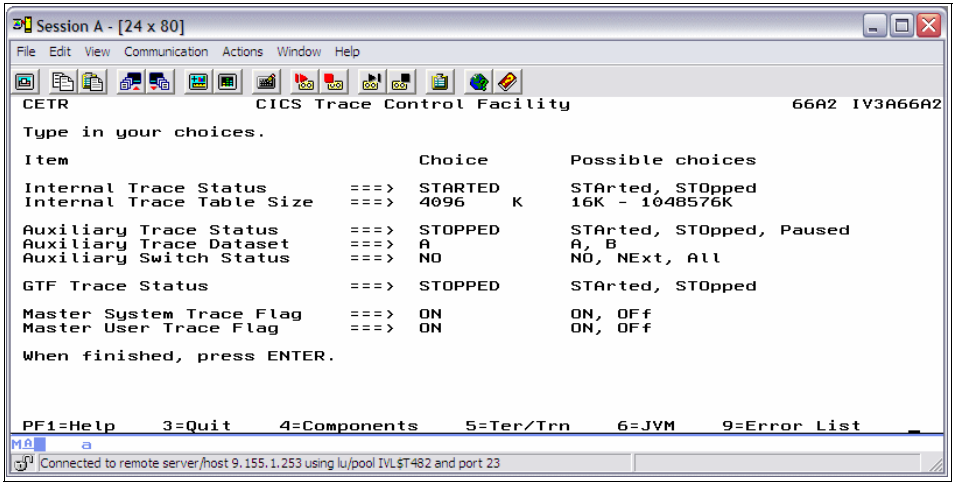


Figure 2-29 CETR initial panel

You can manage the CICS trace facility using the CETR transaction while CICS is running. Trace options can also be defined during the system initialization process. You can use the CETR transaction to define the size of the incore internal trace table. Trace entries are written to a wrap around in-core table. CICS trace entries can also go to DASD, called the auxiliary trace. The third option allows to offload trace entries to the MVS GTF trace data set, which is handy because it allows you to mix other component trace data with CICS trace entries. For example, you can merge TCP/IP trace data with CICS trace entries, which avoids looking at two or more different data sets.The initial panel of the CETR transaction also allows you to switch tracing off completely using the master system trace flag.

2.10.6 The CICS systems programmer challenge

In this section we describe how the CICS systems programmer uses the CETR trace facility to set up tracing more efficiently. Figure 2-30 on page 101 shows the CETR component trace options panel.

```
Session A - [24 x 80]
File  Edit  View  Communication  Actions  Window  Help
CETR                    Component Trace Options                    66A2  IV3A66A2

Over-type where required and press ENTER.                          PAGE 1 OF 4
Component Standard                            Special
--------  ------------------------------      ------------------------------
  AP      1                                   1-2
  BA      1                                   1-2
  BM      1                                   1
  BR      1                                   1-2
  CP      1                                   1-2
  DC      1                                   1
  DD      1                                   1-2
  DH      1                                   1-2
  DM      1                                   1-2
  DP      1                                   1-2
  DS      1                                   1-2
  DU      1                                   1-2
  EC      1                                   1-2
  EI      1                                   1-2
  EJ      1                                   1-2
  EM      1                                   1-2
  EP      1                                   1-2

PF:  1=Help   3=Quit   7=Back  8=Forward  9=Messages   ENTER=Change
MA    a
Connected to remote server/host 9.155.1.253 using lu/pool IVL$T482 and port 23
```

Figure 2-30 CETR component trace options

CICS application programs occasionally abend in production environments only. These application programs probably passed the test cycles without problems. In this case, you must set up tracing in production. All things come with a price; therefore, it is the best practice to use CETR to select the minimum trace components only.

Figure 2-30 shows the CETR component selection panel that you can use to select the trace components you need to debug the problem. You can set tracing off or select different levels of tracing for each component.

There are two columns available to select components.

- Standard tracing: If the master system trace flag is switched on, standard tracing is active. Standard tracing is the actual CICS trace for all tasks. If a problem cannot be associated to a specific transaction, we need standard tracing.
- Special tracing: If the problem can be associated to a specific transaction or to a specific terminal, you can use special tracing. The master system trace flag can be set to off to run special tracing. In the case of special tracing, CICS just traces actions for specific transactions using the selected option from the special trace column.

Figure 2-31 on page 102 shows the special tracing panel of the CETR transaction.

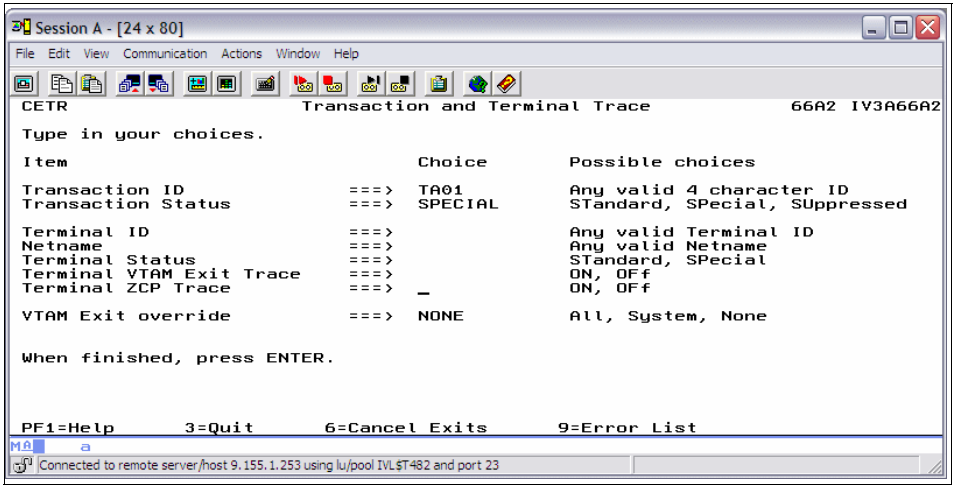


Figure 2-31 CETR special tracing

3

CICS data

There are certain capabilities and functions that you must understand to fully utilize CICS and its functionality. In this chapter, we focus on:

- VSAM Record Level Sharing (RLS)
- File Control
- Coupling Facility
- DB2
- Data tables
- Temporary storage
- Transient data

3.1 VSAM and RLS

Virtual Sequential Access Method (VSAM) was introduced as a collection of three data set organizations, sequential, indexed and direct-access by IBM in the early 1970s. Historically, access to the VSAM files was facilitated by CICS on behalf of the program being executed. This function shipping was managed by any sending CICS, but access to the data was managed by the FOR. As business requirements changed, the affinity created by having data directly accessed by one CICS region on one LPAR became apparent. Even though the shared data table function was available, this can only be facilitated on one LPAR as read operations only.

Record Level Sharing (RLS) allows multiple CICS regions to open the same dataset concurrently with full update capability. RLS have therefore removed the requirement for a dedicated FOR, and management of the dataset is now facilitated by DFSMS. However, for this capability to be exploited, here are some considerations:

- A parallel sysplex with a coupling facility is required (Coupling facility will be explained further in section 5.3 Coupling Facility)
- One SMSVSAM server started on each LPAR that CICS as a requestor is running on
- User data files must be SMS managed, and not be defined with the IMBED attribute
- All requestor CICS regions must execute on the same sysplex
- Batch jobs can concurrently read, but cannot update while in an unquiesced state in CICS

Further to RLS, is data access provided by Transactional VSAM (TVS). This allows the ability of both batch and online simultaneous update and read of recoverable datasets concurrently. Integrity is ensured by making use of the CF locking and cashing capabilities.

Figure 3-1 on page 105 shows how VSAM works prior to the development of RLS. When a task is updating a record in VSAM it has exclusive control of the entire control interval. All other tasks attempting to reference a record within the same control interval will be put into a File Control Exclusive Control Conflict (FCXCWAIT) until the first task completes the update. When a task is reading or browsing a record in VSAM, it has shared control of the entire Control Interval. If another task wants to update a record in the Control Interval, it is put into a FCXCWAIT until all tasks are through reading or browsing. You will learn more about the READ UPDATE command in section 3.2, “File control” on page 106.

Figure 3-2 on page 105 shows the relationship between CICS and VSAM prior to RLS.

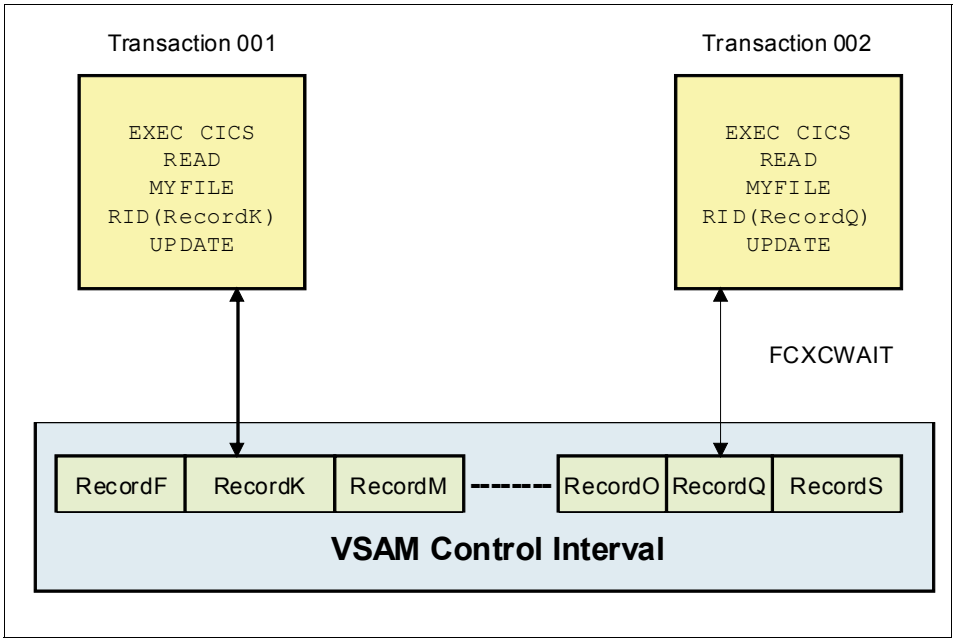


Figure 3-1 VSAM prior to RLS

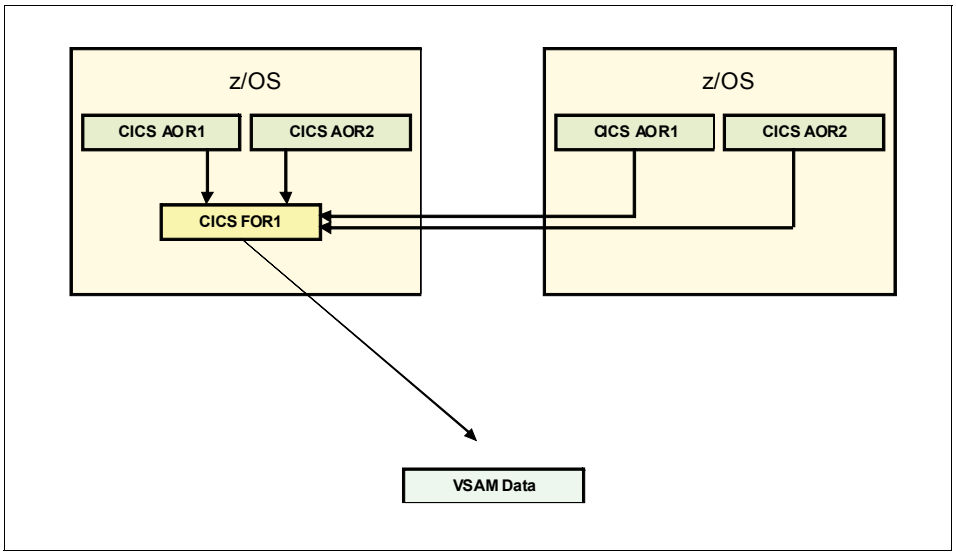


Figure 3-2 CICS and VSAM pre-RLS

Figure 3-3 shows how VSAM behaves with RLS. Multiple CICS regions and transactions can reference the same dataset concurrently with full update capability. There is no longer the problem of encountering the FCXCWAIT.

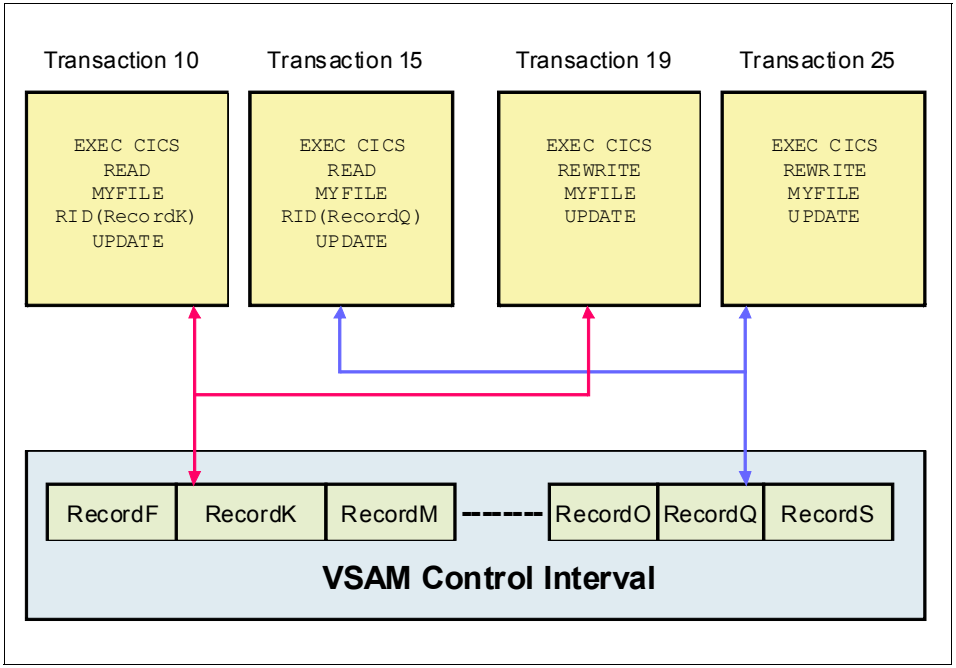


Figure 3-3 VSAM behaves with RLS

Refer to the CICS and DFSMS product manuals and the following IBM Redbooks publications:

- *CICS and VSAM Record Level Sharing: Planning Guide*, SG24-4765
- *CICS and VSAM Record Level Sharing: Implementation Guide*, SG24-4766

3.2 File control

In this section, we discuss file control.

3.2.1 What is CICS File control

File control provides a facility for accessing data sets, files, and data tables, using keyed or relative-byte-address (RBA) access through the virtual storage access method (VSAM), the basic direct access method (BDAM), shared data table services, and the coupling facility data tables server. VSAM data sets can be

accessed in either RLS or non-RLS mode. RLS mode allows sharing of data sets across a parallel sysplex. File control allows updates, additions, deletions, random retrieval, and sequential retrieval (browsing) of logical data in the data sets. If VSAM is used, access to logical data can be through a VSAM alternate index path and through the base data set.

File control reads from, and writes to, user-defined data sets and data tables, gathers statistics, and acquires dynamic storage for I/O operations. File control uses control information defined by the user in the file control table (FCT). This table describes the physical characteristics of all the data sets, and any logical relationships that can exist between them. To access a record, the application program must identify both the record and the data set that holds it. It must also specify the storage area into which the record is to be read or from which it is to be written.

3.2.2 Using file control

File Control allows you to do a number of things when accessing a file:

READ	Reads a record from a file on a local or a remote system.
READNEXT	Reads the next record during a browse of a file.
READPREV	Reads the previous record during a file browse—VSAM and data tables only.
READ UPDATE	Reads the update of a file.
REWRITE	Updates a record in a file on a local or a remote system. You must always precede this command with a read with the UPDATE option.
WRITE	Writes a new record to a file on a local or a remote system.
DELETE	Deletes a record from a file — VSAM KSDS, VSAM RRDS, and data tables only.
STARTBR	Specifies the record in a file, or in a data table, on a local or a remote system, where you want the browse to start. No records are read until a READNEXT command (or, for VSAM and tables, a READPREV command) is executed.
ENDBR	Ends a browse on a file or data table on a local or a remote CICS region.
UNLOCK	Releases the exclusive control established in response to a read command with the UPDATE option.

3.2.3 Files that CICS supports

CICS supports access to the following types of data set:

- Key-sequenced data set (KSDS)
- Entry-sequenced data set (ESDS)
- Relative record data set (RRDS) (both fixed and variable record lengths)

VSAM data sets are held on direct access storage devices (DASD) auxiliary storage. VSAM divides its data set storage into control areas (CA), which are further divided into control intervals (CI). Control intervals are the unit of data transmission between virtual and auxiliary storage. Each one is of fixed size and, in general, contains a number of records. A KSDS or ESDS can have records that extend over more than one control interval, which are called spanned records.

Key-sequenced data set

A key-sequenced data set has each of its records identified by a key. (The key of each record is a field in a predefined position within the record.) Each key must be unique in the data set.

When the data set is initially loaded with data, or when new records are added, the logical order of the records depends on the collating sequence of the key field. This also fixes the order in which you retrieve records when you browse through the data set.

To find the physical location of a record in a KSDS, VSAM creates and maintains an index. This relates the key of each record to the record's relative location in the data set. When you add or delete records, this index is updated accordingly.

Entry-sequenced data set

An entry-sequenced data set is one in which each record is identified by its relative byte address (RBA).

Records are held in an ESDS in the order in which they were first loaded into the data set. New records added to an ESDS always go after the last record in the data set. You cannot delete records or alter their lengths. After a record is stored in an ESDS, its RBA remains constant. When browsing, records are retrieved in the order in which they were added to the data set.

A standard RBA is an unsigned 32-bit number. The use of a 32-bit RBA means that a standard ESDS cannot contain more than 4 gigabytes (GB) of data. However, there is a different kind of ESDS that supports 64-bit extended relative byte addresses (XRBAs) and which is therefore not subject to the 4GB limit. This

type of ESDS is called an “extended format, extended addressing ESDS data set”.

Relative record data set

A relative record data set has records that are identified by their relative record number (RRN). The first record in the data set is RRN 1, the second is RRN 2, and so on.

Records in an RRDS can be fixed or variable length records, and the way in which VSAM handles the data depends on whether the data set is a fixed or variable RRDS. A fixed RRDS has fixed-length slots predefined to VSAM, into which records are stored. The length of a record on a fixed RRDS is always equal to the size of the slot. VSAM locates records in a fixed RRDS by multiplying the slot size by the RRN (which you supply on the file control request), to calculate the byte offset from the start of the data set.

A variable RRDS, on the other hand, can accept records of any length up to the maximum for the data set. In a variable RRDS VSAM locates the records by means of an index.

3.2.4 Transaction deadlocks

One of the more popular issues dealing with file control is a deadlock. A deadlock occurs if each of two transactions (for example, A and B) needs exclusive use of some resource (for example, a particular record in a data set) that the other already holds. Transaction A waits for the resource to become available. However, if transaction B is not in a position to release it because it, in turn, is waiting on some resource held by A, both are deadlocked and the only way of breaking the deadlock is to cancel one of the transactions, thus releasing its resources.

A transaction needs exclusive control of resources while executing file control commands. For VSAM datasets, any record that is being modified is held in exclusive control by CICS from the time when the modification begins (for example, when a READ UPDATE command is issued to obtain control of the record), to the time when it ends (for example, when a REWRITE command has finished making a change to the record).

With VSAM files accessed in RLS mode, only the individual record is ever locked during this process. With VSAM files accessed in non-RLS mode, when VSAM receives a command that requires control of the record, it locks the complete control interval containing the record. CICS then obtains an enqueue on the record that it requires, and releases the control interval, leaving only the record

locked. The control interval lock is released after each command, and only the individual record is locked for the whole of the modification process.

In releases before CICS Transaction Server for z/OS, Version 2 Release 2, the access method also held its lock on the complete control interval between the commands, from the time when the modification began to the time when it ended. This is no longer the case.

As well as CICS having exclusive control of a record during the modification process, there is an extra locking period when a transaction modifies a record in a recoverable file. In this situation, CICS (or VSAM if the file is accessed in RLS mode) locks that record to the transaction even after the request that performed the change is complete. The transaction can continue to access and modify the same record; furthermore, other transactions must wait until the transaction releases the lock.

Whether a deadlock occurs depends on the relative timing of the acquisition and release of the resources by several concurrent transactions. Application programs can continue to be used for some time before meeting circumstances that cause a deadlock. It is important to recognize and allow for the possibility of deadlock early in the application program design stages.

For VSAM files that are accessed in non-RLS mode, CICS detects deadlock situations, and a transaction about to enter a deadlock is abended with the abend code AFCF if it is deadlocked by another transaction. If the transaction has deadlocked itself, it is abended with code AFCG.

To avoid deadlocks, all applications that update (modify) multiple resources must do so in the same order. For instance, if a transaction updates more than one record in a data set, it can do so in ascending key order. A transaction that is accessing more than one file must always do so in the same predefined sequence of files.

3.3 Coupling facility

A coupling facility provides locking, caching, and list services between coupling-capable S/390® processors running MVS/ESA Version 5 or later. Coupling facility links connect the coupling facility to the coupling-capable processors. The coupling facility control code (CFCC) provides the coupling facility functions. You can use an Integrated Coupling Migration Facility (ICMF) for migration and for testing software that requires a coupling facility. ICMF provides the CFCC functions without coupling facility links and is limited to MVS/ESA systems running in an Logical Partition (LP) on a single physical system.

The coupling facility enables software on several systems in the Parallel Sysplex to share data with the assurance that the data is not corrupted and is consistent among all sharing users. To share data, systems must have connectivity to the coupling facility through coupling facility links.

To get a good grasp of the coupling facility and its capabilities, you must understand the types of coupling facilities that exist:

- Dedicated coupling facility
- Coupling Facility LP Defined with Other LPs on a Processor
- Coupling Facility LP and the ICMF
- Volatile Coupling Facility
- Nonvolatile Coupling Facility

Coupling facility data tables

Coupling facility data tables provide a method of file data sharing, using CICS® file control without the need for a file-owning region or VSAM RLS support. CICS coupling facility data table support is designed to provide rapid sharing of working data within a sysplex, with update integrity. The data is held in a coupling facility, in a table that is similar in many ways to a shared user-maintained data table. This topic describes how to define the resources required for coupling facility data tables in an MVS™ coupling facility resource management (CFRM) policy.

Because read access and write access have similar performance, this form of table is particularly useful for scratchpad data. 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 distinct requirements for scratchpad data, and most of these can be implemented using coupling facility data tables. Coupling facility data tables are particularly useful for grouping data into several tables, where the items can be identified and retrieved by their keys. For example, you can 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 can maintain a list of the numbers of lost credit cards in a coupling facility data table.

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.

Coupling facility log streams reside in the coupling facility and in either a data space in the IXGLOGR address space or in staging data sets. DASD-only log

streams reside in a data space in the IXGLOGR address space and in staging data sets.

All log streams that are needed by CICS® must be defined to the MVS™ system logger before CICS can use them. You can either define log streams explicitly, or you can let CICS create them dynamically when they are first used. To enable CICS to create log streams dynamically, you first define model log streams to the MVS system logger. To define explicit log streams and model log streams, use the MVS IXCMIAPU utility.

In CICS® TS for z/OS®, Version 4.1, workload throughput is improved through a more efficient workload management optimization function. This function is most effective for distributed workloads, for which the routing and target regions are managed by multiple CMASs. With new CPSM WUI views, you can monitor the distribution of dynamic workloads through your CICSplex.

Sysplex optimized workload routing is enabled at the z/OS coupling facility level by a region status (RS) server.

When a target region is running in optimized mode, the target region maintains the task count using the CICS transaction manager. The count includes all tasks in the CICS region, not just those that are dynamically routed. The load value for the CICS region, with its basic health status, is periodically broadcast to the coupling facility, where it is available for interrogation by other CICS regions and CMASs and by other CICS address spaces. If region status data is available, CICSPlex SM uses the data when it makes its dynamic routing decision.

For sysplex optimized workloads, routing regions review the same status data in the coupling facility for a potential target region regardless of which CMAS manages it. As a result, the routing region is using status data that might be updated many times a second to evaluate a target region, rather than status data that might be up to 15 seconds old. The refresh interval can vary from two seconds down to one millisecond. As the scale of this value is reduced, the usage effect on the coupling facility increases. Choose a value that provides a balance between workload throughput and the effect on the coupling facility. The default refresh value is 200 milliseconds. In an environment on which all routing targets are in a similar health and connectivity state, the spread of work across the workload target scope is more evenly balanced than in non-optimized mode.

If the coupling facility is not available, workload routing is managed by CICSPlex SM Workload Manager using z/OS data spaces that are owned by a CMAS to share cross-region load and status data.

Benefits of sysplex optimized workload routing

A sysplex optimized workload is a workload that is best suited to workloads that are contained in a single sysplex. For a workload that runs in a CICSplex that spans more than one sysplex, the benefits of optimized routing are reduced because region status data that is stored in the z/OS coupling facility is not shared across sysplexes.

Sysplex optimized workload routing is beneficial in these scenarios:

- When a workload consists of routers and targets managed by multiple CMASs and the bulk of the dynamic traffic flows through the DSRTPGM exit. For example, if you use MQ triggers to feed transactional data into CICSplexes, where the trigger regions tend to be managed by several separate CMASs to the processing regions. In these instances, the benefit of running workloads in optimized state is that no workload batching occurs, and the overall workload runs through faster. Fewer, if any, routed transactions are waiting in the queue of a CICS region already at its MAXTASKS limit.
- When the topology of a CICSplex is such that regions in a workload can be managed by the same CMAS, and nondynamic throughput is a high proportion of the workload.
- Impact of sysplex optimized workload routing on the coupling facility. Caching mechanisms are built into the coupling facility (CF) to reduce the number of I/O operations. CICS region status data is broadcast to the CF by target regions, and the data is subsequently read back by the routing regions when a route decision is being made. If CICS status data is broadcast at every change instance, and read back on every occasion that a route decision is made, the impact to the coupling facility might be considerable.

3.3.1 Dedicated coupling facility

The dedicated coupling facility is a central processor complex (CPC) that runs only the CFCC. The CFCC runs in a single LP with dedicated processor resources. There are two kinds of dedicated coupling facilities: the stand-alone IBM 9674, which can run only the CFCC and IBM 9672 CPCs, which give you the option of dedicating the CPC to run only the CFCC when you define the coupling facility LP.

3.3.2 Coupling facility LP defined with other LPs on a processor

You can define the coupling facility LP to run on a CPC in the S/390 microprocessor cluster, a 9021 711-based model, or a 9121 511-based model with other LPs defined to the CPC. The CPC in the S/390 microprocessor cluster and the 9021 711-based models enable you to use coupling facility links to

connect the coupling facility LP to MVS images on the same processor or on other processors that can connect to the coupling facility. On the 9121 511-based models, you must use ICMF on the processor to define an LP to run the coupling facility, and you do not install coupling facility links.

3.3.3 Coupling facility LP and the ICMF

For testing the coupling facility function, you can define coupling facility LPs with other LPs on a processor and use ICMF to simulate coupling facility links. With ICMF you do not use coupling facility links to connect the coupling facility, LPs, and MVS images, and you cannot connect the coupling facility LP to MVS images on other processors. ICMF is not generally recommended in a production Parallel Sysplex environment for CICS TS or VSAM RLS. The function is available on the following processors:

- 9021 711-based models
- 9121 511-based models
- IBM 9672 processors

3.3.4 Volatile coupling facility

A volatile coupling facility is one where the contents of memory are lost if the power supply is interrupted.

3.3.5 Nonvolatile coupling facility

Conversely, a nonvolatile coupling facility is one where the power supply is never interrupted. This is usually attained by adding an uninterrupted power supply (UPS) to the coupling facility. In the event of a power failure, the UP provides power to the coupling facility.

3.3.6 What this all means

Knowledge of what the coupling facility can do and what exactly it does is great. However the key is to use it in a way that allows for peak performance and throughput to meet business demands, which is inline with our goals as an IT service provider. To achieve this, we recommend that you use a nonvolatile dedicated IBM 9674 coupling facility. The 9674 allows for a coupling facility that is external to all the processors in the Parallel Sysplex; therefore, it does not incur problems that the Sysplex has. If one of the processors in the Sysplex were to fail, the coupling facility and the information it contains remains readily available to whomever needs it. Other processors in the coupling facility can access the information and the remaining parts of the Parallel Sysplex can run. You want to

attain the highest availability and avoid any situations where a failure of one processor can lead to the failure of the coupling facility as a whole.

For a more detailed explanation about how the coupling facility can be used in your business environment, see:

Coupling Facility Performance: A Real World Perspective, REDP-4014-00

Figure 3-4 illustrates the relationship between a coupling facility, VSAM, SMSVSAM, and CICS.

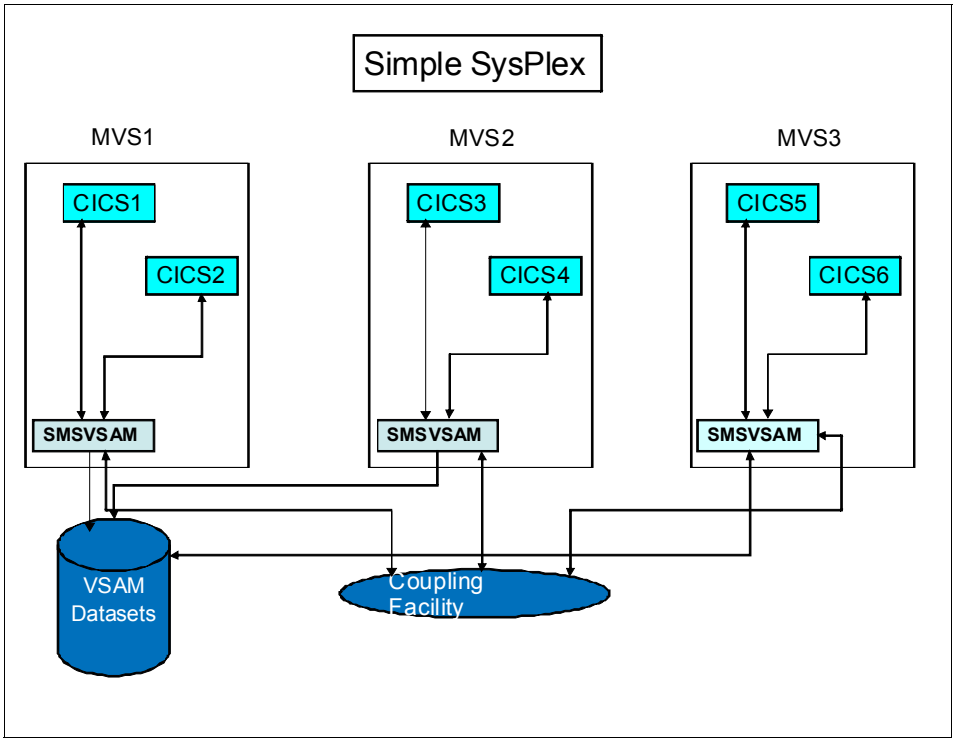


Figure 3-4 Intertwined CICS regions, VSAM/SMSVASM, and the coupling facility

3.4 CICS and DB2

One important aspect of CICS TS is its ability to communicate and exchange information with DB2. This is an important interaction because there are a large number of companies and customers who use DB2 to manage their databases and the information that they contain.

The key point is that a CICS program can contain EXEC SQL commands for accessing DB2 data.

CICS provides the CICS DB2 attachment facility so that you have the ability to access CICS data and DB2 data. The CICS DB2 attachment facility allows for your applications that use CICS to access DB2 data while they are still in a CICS environment. The connection between CICS and DB2 can be created or terminated at any time, and CICS and DB2 can be started and stopped independently. The connection between CICS and DB2 can be created or terminated at any time, and CICS and DB2 can be started and stopped independently.

There are a couple of aspects of the CICS DB2 attachment facility that must be understood:

- There are no DB2 release dependencies within the attachment facility. It can connect to a DB2 subsystem running any supported level of DB2
- A CICS address space can connect to one and only one DB2 system at a time.
- More than one CICS system can be connected to a DB2 system.
- CICS does not support SQL function shipping.

Resource Definition Online (RDO) defines the CICS DB2 connection characteristics. The three characteristics or definitions are:

- DB2 connection definition (DB2CONN): Defines the attributes of the connection between CICS and DB2.
- DB2 entry definition (DB2ENTRY): Defines the attributes of entry threads used by the CICS DB2 attachment facility.
- DB2 transaction definition (DB2TRAN): Defines a transaction, or a group of transactions, associated with a DB2ENTRY.

You must install a DB2CONN resource definition before you can start the CICS DB2 connection. You can also create DB2ENTRY and, if necessary, DB2TRAN definitions to ensure that your important transactions are prioritized.

The connection between CICS and DB2 is a multithread connection. Within the overall connection between CICS and DB2, there is a thread—an individual connection into DB2—for each active CICS transaction accessing DB2. Threads allow each CICS transaction to access DB2 resources, such as a command processor or an application plan (the information that tells DB2 what the application program's SQL requests are and the most efficient way to service

them). In this IBM Redbooks publication, we do not get in depth about the DB2 attachment facility and how it works; however, here is an example of how CICS and DB2 process an application program issuing its first SQL request:

1. A language interface, or stub, DSNCLI, that is link-edited with the application program calls the CICS resource manager interface (RMI).
2. The RMI processes the request, and passes control to the CICS DB2 attachment facility's task-related user exit (TRUE), the module that invokes DB2 for each task.
3. The CICS DB2 attachment facility schedules a thread for the transaction. At this stage, DB2 checks authorization, and locates the correct application plan.
4. DB2 takes control, and the CICS DB2 attachment facility waits while DB2 services the request.
5. When the SQL request completes, DB2 passes the requested data back to the CICS DB2 attachment facility.
6. CICS now regains control, and the CICS DB2 attachment facility passes the data and returns control to the CICS application program.

3.5 Data tables

In this section, we discuss data tables.

3.5.1 What are data tables

The concept of shared data tables is simple. It exploits the fact that it is more efficient to:

- Use MVS cross-memory services instead of CICS function shipping to share a file between two or more CICS regions in the same MVS image.
- Access data from memory instead of from DASD.
- Access a file of data from memory using services integrated within CICS file management instead of using VSAM services and a local shared resource (LSR) pool.

Shared Data Tables (SDT) completely replace and extend the basic data table services that were originally provided.

Under SDT, all files that are defined as data tables can potentially be shared using cross-memory services. No changes are required to the file definitions for existing data tables.

The use of cross-memory services is one of the major enhancements to data table services that is included in the SDT facility. This enhancement improves the performance of applications that currently use function shipping and makes file sharing feasible for applications that cannot accept the performance overhead of function shipping.

The other major enhancement is that nearly all read requests are supported for use with data tables. This enhancement extends the use of data tables to applications that include:

- Browse requests
- Read requests that use an imprecise key

3.5.2 Types of data tables

A CICS file is a representation of a data set on DASD. If you specify that the file is to use data table services, CICS copies the contents of the data set into an MVS data space when the file is opened and uses that copy whenever possible.

Because of the way that the data table services access the records, they can be used only with a VSAM key-sequenced data set (KSDS). The KSDS is called the source data set. The copy in memory is called the data table. The process of copying the records is called loading the data table.

When the file is read by a CICS application, the record is normally retrieved from the data table. When the file is updated by a CICS application, the effect depends on the type of data table that you defined for the file.

CICS data table services support two types of data tables:

- CICS-maintained data table (CMT)
- User-maintained data table (UMT)

CICS-maintained data table

A CICS-maintained data table is a data table whose records are automatically reflected in the source data set. When you update the file, CICS changes both the source data set and the data table.

A CICS-maintained data table is easy to implement—you must know little about the data table services, you do not need to change your existing application programs, and full recovery support of the file is retained. CICS-maintained data tables are discussed in more detail in CICS-maintained data tables. A data set being accessed in Record Level Sharing (RLS) mode cannot be used as the source for a CICS-maintained data table. The source data set must be accessed in non-RLS mode.

User-maintained data table

A user-maintained data table is a data table whose records are not automatically reflected in the source data set; instead, when you update the file, CICS changes only the data table.

A user-maintained data table lets you optimize the benefits of using a data table by allowing you to eliminate activity on the source data set, for update requests and read requests.

A small number of file operations are not supported for user-maintained data tables; therefore, you might need to make minor changes to existing application programs. Also, recovery of the file is supported after a transaction failure, but not a system failure.

A base VSAM KSDS accessed in either non-RLS or RLS mode can be used as the source data set for a user-maintained data table. You might want to make an RLS-mode data set the source of a user-maintained data table if you have other file definitions that access the data set and the data set is updated by other CICS regions.

3.6 Temporary storage

In this section, we discuss temporary storage.

3.6.1 What is temporary storage

CICS temporary storage is a scratchpad facility that is heavily used in many systems. Data in temporary storage tends to be short-lived, emphasis being placed on ease of storage and retrieval.

You can set up temporary storage for a CICS® region in three locations:

- Main storage is in memory, so faster but limited in size.
- Auxiliary storage on disk, so slower but less size constraints.
- Shared temporary storage pools in a z/OS® coupling facility.

Main storage

Main temporary storage is in the CICS region. Main temporary storage can be in 64-bit (above-the-bar) storage, rather than 31-bit (above-the-line) storage, depending on the version of the z/OS operating system and whether the CICS region operates with transaction isolation. You use the TSMAINLIMIT system initialization parameter to specify the amount of storage that is available to temporary storage queues.

You can use local main storage in the CICS region where the applications run, or you can function ship temporary storage requests to a remote queue-owning region (QOR).

Auxiliary storage

Auxiliary temporary storage is in a non indexed VSAM data set named DFHTEMP. You define the available space and any additional extents when you set up this data set. Some 31-bit (above-the-line) storage is used in the CICS region for VSAM buffers to make control intervals available from the VSAM data set. You use the TS system initialization parameter to set the number of buffers. Like main temporary storage, auxiliary temporary storage can be associated with the local CICS region or a remote queue-owning region.

Shared temporary storage pools in a z/OS coupling facility

Shared temporary storage pools (TS pools) are in a z/OS coupling facility that is managed by a temporary storage data sharing server (TS server). Each pool corresponds to a list structure in the coupling facility. You specify the size of each temporary storage pool using the coupling facility resource manager (CFRM) policy definition utility in z/OS. Shared temporary storage pools do not use any storage in the CICS region, and applications access them directly from the local CICS region.

When applications use the WRITEQ TS and READQ TS commands to access temporary storage queues, the requests are processed by the CICS temporary storage domain, which creates temporary storage queues in the appropriate storage location and places the data in them. Any task can retrieve the data using the symbolic name of the temporary storage queue. The CICS temporary storage domain can process multiple requests concurrently, but it serializes requests made for the same temporary storage queue, and the queue is locked for the duration of each request.

You use TSMODEL resource definitions to set up models that CICS uses to create temporary storage queues. Each model specifies the following attributes for temporary storage queues with names that match the model:

- The location of the temporary storage where the queue must be stored
- Whether the temporary storage is associated with the local CICS region or a remote CICS region, such as a queue-owning region
- Whether the queue is deleted automatically by CICS, if it remains unused for a period of time and is not deleted by an application
- Whether the queue is recoverable

Temporary storage items can be stored either in main storage or in auxiliary storage on DASD. Main-only support can be forced by specifying TS=(,0) (zero temporary storage buffers) in the SIT.

Effects of temporary storage

If main temporary storage is used, requests to a TS queue are serialized with the storage being allocated from the ECDSA. The performance of auxiliary temporary storage is affected by the characteristics of the data set where it resides. The VSAM control interval (CI) size affects transfer efficiency, with a smaller size being desirable if access to CIs is random, and a larger size if use of CIs is more sequential. In general, the larger the queues and write/read ratio, the more sequential the usage tends to be. Records that span control intervals are possible. Up to 32767 buffers and 255 strings can be specified, and overlap processing can be achieved, although a specific queue is still processed serially.

Main temporary storage: Key point is that MAIN is in memory and faster, but limited in size.

Limitations

Increasing the use of main temporary storage, using a larger CI size, or increasing the number of buffers, increases the virtual storage needs of the ECDSA and real storage needs. If you use auxiliary temporary storage, a smaller CI size can reduce the real storage requirements.

Allocating temporary storage

Temporary storage requests on a cold-started or initial-started system (that is with no existing auxiliary data) are allocated from the start of DFHTEMP (the temporary storage data set used for storing auxiliary data). They are processed by the CICS temporary storage domain. The first control interval within DFHTEMP is used until a WRITEQ is issued that is too long to fit into the remaining space. Temporary storage processing then switches to use control interval two, and so on. This process continues until all of the control intervals in DFHTEMP have data written to them.

WRITEQ requests after this point are directed back to the start of the data set. Temporary storage processing maintains a bytemap that represents the free space available within each control interval in the data set at any time. Temporary storage processing now starts interrogating the bytemap to find a control interval that can accommodate new data at, or near to, the start of DFHTEMP. The reasoning behind this is that by now, you can delete queues written earlier. Such deleted data remains in control intervals but is no longer required. If the bytemap shows that a control interval contains enough space, temporary storage processing reads it into a temporary storage buffer, compresses it to move all

valid records to the start of the control interval, and uses the remaining contiguous space to store the data from the new request.

Temporary storage recommendations

In this section, we provide recommendations for main temporary storage, auxiliary temporary storage, secondary extents, multiple buffers, concurrent input/output operations (multiple strings), and Control Interval (CI) sizes.

Main temporary storage

Temporary storage items are stored in the ECDSA above the 16MB line. No recovery is available. Queues are locked for the duration of the TS request.

The fact that temporary storage items are stored in main storage also means that there is no associated I/O, so we recommend main temporary storage for short-duration tasks with small amounts of data.

Auxiliary temporary storage

Auxiliary temporary storage occupies less address space than main temporary storage, and must be used for large amounts of temporary storage data or for data that is to be held for long periods.

Temporary storage I/O occurs only when a record is not in the buffer, when a new buffer is required, or if dictated by recovery requirements.

Secondary extents for temporary storage

On a cold start of temporary storage when the data set is empty, the data set is formatted to the end of the primary extent. Any secondary extents are not formatted. On a cold start of temporary storage when the data set is not empty or when temporary storage is not cold started, no formatting of the data set takes place.

The use of secondary extents allows more efficient use of DASD space. You can define a temporary storage data set with a primary extent large enough for normal activity, and with secondary extents for exceptional circumstances, such as unexpected peaks in activity.

It follows that you can reduce or eliminate the channel and arm contention that is likely to occur because of heavy use of temporary storage data.

Multiple buffers

Using multiple VSAM buffers allows multiple VSAM control intervals to be available in storage simultaneously. This makes it possible for the CICS temporary storage programs to service several requests concurrently, using different buffers.

Using multiple buffers also increases the likelihood that the control interval required by a particular request is already available in a buffer. This can lead to a significant reduction in the number of input/output requests (VSAM requests) that must be performed. (However, VSAM requests are always executed whenever their use is dictated by recovery requirements.) Although using a large number of buffers can greatly improve performance for non-recoverable TS queues, the associated buffers still must be flushed sequentially at CICS shutdown, and that might take a long time.

The number of buffers that CICS allocates for temporary storage is specified by the system initialization parameter, TS.

The benefits of multiple buffers depend on the way an installation's auxiliary temporary storage is used. In most cases, the default TS specification in the SIT (three buffers) must be sufficient. Where the use of temporary storage is high or where the lifetime of temporary storage data items is long, it might be worthwhile to experiment with larger numbers of buffers. The buffer statistics in the CICS temporary storage statistics give sufficient information to help you determine a suitable allocation.

In general, aim to minimize the number of times that a task must wait either because no space in buffers is available to hold the required data or because no string is available to accomplish the required I/O. The trade-off here is between improvement of temporary storage performance and increased storage requirements. Specifying a large number of buffers can decrease temporary storage I/O but lead to inefficient usage of real storage and increased paging.

Concurrent input/output operations (multiple strings)

Temporary storage programs issue VSAM requests whenever real input/output is required between the buffers and the VSAM temporary storage data sets. The use of multiple VSAM strings enables multiple VSAM requests to be executed concurrently, which, in turn, leads to faster servicing of the buffers.

VSAM requests are queued whenever the number of concurrent requests exceeds the number of available strings. Constraints caused by this can thus be relieved by increasing the number of available strings, up to a maximum equal to the number of buffers.

The number of VSAM strings that CICS allocates for temporary storage is specified by the system initialization parameter, TS.

Multiple strings allow more I/O operations to be performed concurrently. Several I/O requests can be outstanding at any time, up to the number of strings specified. Allowing the number of strings to default to the number of buffers ensures that no tasks are waiting for a string. Not all strings can be used in this

case, however, which causes inefficient use of storage. Adjust the number of strings using the peak number in use given in the statistics.

If the device containing the temporary storage data set is heavily used, the TS system initialization parameter can be used to regulate the activity, but this leads to an increase in internal CICS waits.

Control interval sizes

First consider whether the control interval (CI) size for the data set is suitable for your overall system requirements.

BMS paging might be on a large-screen device. Check whether it exceeds your temporary storage CI size.

Because temporary storage can use records larger than the control interval size, the size of the control intervals is not a major concern, but there is a performance overhead in using temporary storage records that are larger than the CI size.

The parameter, CONTROLINTERVALSIZE, of the VSAM CLUSTER definition is specified when you allocate your data sets.

The control interval size must be large enough to hold at least one (rounded up) temporary storage record, including 64 bytes of VSAM control information for control interval sizes less than, or equal to, 16 384, or 128 bytes of control information for larger control interval sizes.

3.7 Transient data

In this section, we discuss transient data

3.7.1 What is transient data

The transient data program provides a generalized queuing facility that enables data to be queued (stored) for subsequent internal or offline processing. Selected units of information can be routed to or from predefined symbolic queues. The queues are classified as either intrapartition or extrapartition. Intrapartition and extrapartition queues can be referenced through indirect destinations. This provides flexibility in program maintenance. Queue definitions can be changed, using the CEDA transaction, without having to recompile existing programs.

Indirect queues

Intrapartition and extrapartition queues can be used as indirect queues. Indirect queues provide some flexibility in program maintenance in that data can be routed to one of several queues with only the transient data definition, and not the program itself having to be changed.

When a transient data definition is changed, application programs continue to route data to the queue using the original symbolic name; however, this name is now an indirect queue that refers to the new symbolic name. Because indirect queues are established by using transient data resource definitions, the application programmer does not usually must be concerned with how this is done

The key difference between temporary storage and transient data:

Temporary storage items are written/read sequentially but can be updated individually and can be read multiple times.

Transient data reads are destructive, and writing can be used to trigger transactions at a specified level.

Intrapartition queues

Intrapartition queues are queues of data, held in a direct-access data set, for eventual input to one or more CICS transactions. Intrapartition queues are accessible only by CICS transactions within the CICS address space. Data directed to or from these internal queues is called intrapartition data. It can consist of variable-length records only.

An intrapartition queue is mapped onto one or more control intervals in the intrapartition data set. A control interval contains one or more temporary storage records. The control intervals are allocated to a queue as records are written and freed automatically as they are read or as the queue is deleted.

Examples of the data queued for intrapartition processing are:

- Transactions that require processes to be performed serially, not concurrently. An example of this type of process is one in which pending order numbers are to be assigned.
- Data to be used in a data set (file) update that can pass through the queue to allow the data to be applied in sequence.

Extrapartition queues

Extrapartition queues (data sets) reside on any sequential device (DASD, tape, printer, and so on) that is accessible by programs outside (or within) the CICS® region.

In general, sequential extrapartition queues store and retrieve data outside the CICS region. For example, one task might read data from a remote terminal, edit the data, and write the results to a data set for subsequent processing in another region. Logging data, statistics, and transaction error messages are examples of data that can be written to extrapartition queues. In general, extrapartition data created by CICS is intended for subsequent batched input to non-CICS programs. Data can also be routed to an output device such as a printer.

Data directed to or from an external destination is referred to as extrapartition data and consists of sequential records that are fixed-length or variable-length, blocked or unblocked. The record format for an extrapartition destination must be defined in a TDQUEUE resource definition by the system programmer. See Defining TDQUEUE resources for details about queue definitions.

If you create a data set definition for the extrapartition queue using JCL, the DD statement for the data set must not include the FREE=CLOSE operand. If you do specify FREE=CLOSE, attempts to read the queue after the queue is closed and then reopened can receive an IOERR condition.

3.7.2 Transient data services

The following services are performed by the transient data program in response to transient data commands issued in application programs:

- Intrapartition data disposition: Controls and queues data for serially reusable or re-enterable facilities (programs, terminals) related to this partition or region.
- Intrapartition data acquisition: Retrieves data that was placed in a queue for subsequent internal processing.
- Extrapartition data acquisition: Enters a sequentially organized data set into the system.
- Extrapartition data disposition: Writes fixed or variable-length data in a blocked or unblocked format on sequential devices, usually for subsequent offline processing.
- Automatic transaction initiation: Initiates a transaction to process previously queued transient data when a predefined trigger level is reached.
- Dynamic open/close: Logically opens or closes specified extrapartition data sets (queues) during the real-time execution of CICS.

- Dynamic allocation and deallocation of extrapartition queues: Extrapartition transient data queues do not have to be predefined in your JCL. They can be created dynamically.

3.7.3 Why implement transient data capabilities

There are a number of reasons to use transient data. Transient data is used in many circumstances within CICS, including:

- Servicing requests made by user tasks, for example, a request to build a queue of data for later processing.
- Servicing requests from CICS, primarily to write messages to system queues for printing. Transient data must, therefore, be set up at your installation to capture these CICS messages.
- Initiating tasks based on queue trigger level specification and on records written to an intrapartition destination.
- Requesting logging for recovery as specified in your CICS transient data definitions.
- Passing extrapartition requests (requests to external data sets) to the operating system access method for processing.

4

Access technologies

In this chapter, we introduce the CICS access technologies. We provide product information and an overview of the components and topologies.

The CICS access technologies we discuss here are:

- CICS web services
- CICS Transaction Gateway
- WOLA
- CICS web support
- WebSphere MQ
- CICS sockets

4.1 CICS web services

Application programs that run in CICS can participate in a heterogeneous web services environment as service requesters, service providers, or both. Figure 4-1 shows an outline of the web services support in CICS.

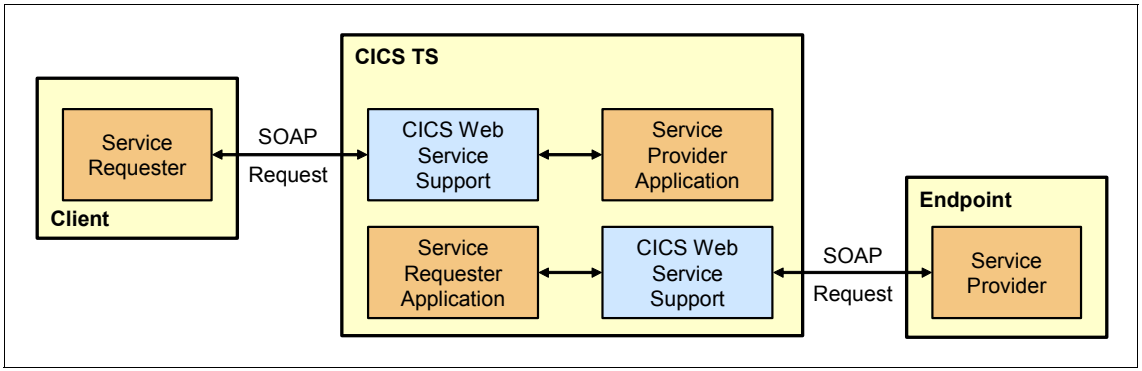


Figure 4-1 Web services in CICS

CICS support for web services conforms to open standards, including these standards:

- SOAP 1.1 and 1.2
- HTTP 1.1
- WSDL 1.1 and 2.0

CICS supports the most common type of communication between service requester and service provider: SOAP over HTTP.

CICS also receives and sends SOAP messages to WebSphere MQ (WMQ) using the WMQ transport, both in the role of service provider and service requester.

Figure 4-2 on page 131 shows an overview of how the component of the CICS web services support fits together.

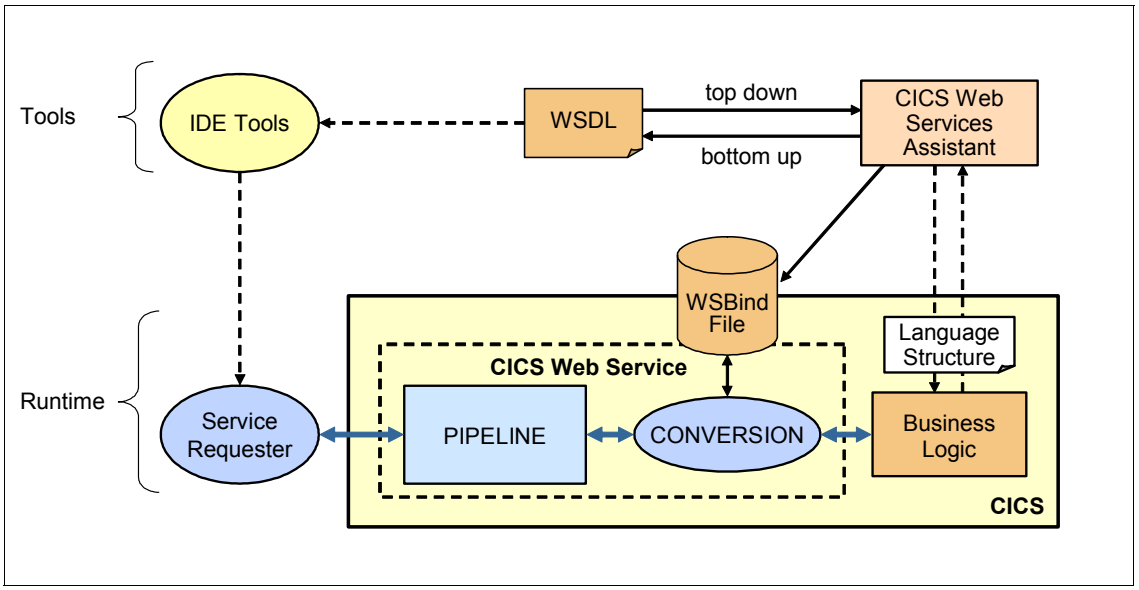


Figure 4-2 Overview of CICS web services support

Figure 4-2 makes a clear distinction between the tools and the runtime components of CICS web service support. The tooling includes the IDE called Rational Developer for System z (RDz) and the CICS web service assistant batch utilities. The runtime includes pipeline, message handlers and CICS resource definitions. The next section covers the tools and runtime.

4.2 Tools

The tools that are available to enable CICS programs as web services are:

- CICS Web Service Assistant

  The CICS Web services assistant is a set of batch utilities that can help you transform existing CICS applications into web services and enable CICS applications to use web services provided by external providers.

  The assistant can create a WSDL document from a simple language structure like a COBOL copybook, or a language structure (copybook) from an existing WSDL document, and supports COBOL, C/C++, and PL/I. It also generates information used to enable automatic run-time conversion of the SOAP messages to containers and COMMAREAs, and vice versa.

  The assistant generates WSBIND file. The WSBIND file is used by CICS to do message transformation (SOAP to application data and vice versa)

- Rational Developer for System z (RDz)

  RDz facilitates the development of both Java and z/OS-based applications. The XML Services for the Enterprise (XSE) capability of RDz provides tools that let you adapt COBOL-based applications so that they can consume and produce XML messages. The Web Services Enablement wizard is the XSE tool that supports the bottom-up approach for creating Web services based on existing CICS COBOL programs.

Both of these tools can be used to build web services in various development styles. You can use the tools to build web service in top down style, bottom up style, or meet in the middle.

4.3 Components for CICS web service

In this section, we discuss the components for CICS web services.

4.3.1 Base components

In addition to the TCP/IP Listener, Secure Sockets Layer (SSL) support, and HTTP handler as described in “Base components for CICS web support” on page 156, CICS also provides the following base components for CICS web service:

- MQ Listener
- SOAP protocol Handler
- XML data mapper

4.3.2 Resource components for web service requests

The following CICS resources are configured by a systems programmer to access CICS services over HTTP:

- TCPIPSERVICE

  A TCPIPSERVICE definition is required in a service provider that uses the HTTP or HTTPS as transport. It contains information about the port on which inbound requests are received, and whether any transport based security mechanisms will be applied by CICS.

- URIMAP

  A URI mapping or URIMAP resource definition matches the URIs of Web service requests. The URIMAP associates a URI for the request with a PIPELINE and WEBSERVICE resource that specifies the processing to be performed. You can use a URIMAP to specify:

  - The name of the transaction that CICS uses for running the pipeline alias transaction (default is CPIH).
  - The user ID under which the alias transaction runs.

- PIPELINE

  A PIPELINE resource definition provides information about the message handlers that will act on a service request and on the response. The information about the message handlers is supplied indirectly. The PIPELINE definition specifies the name of an HFS file, called the pipeline configuration file, which contains an XML description of the message handlers and their configuration. There are two kinds of pipeline configuration files: one describes the configuration of a service provider pipeline, and the other describes the configuration of a service requester pipeline.

- WEBSERVICE

  A WEBSERVICE resource defines the aspects of the run time environment for a CICS application program deployed in a Web services setting. The CICS web service assistant generates the mapping between application data structure and SOAP messages. Three objects define the execution environment that allows a CICS application program to operate as a Web service provider or a Web service requester, as shown in Figure 4-3 on page 134:

  - WEBSERVICE resource: The Web service description
  - WSbind file: The Web service binding file
  - PIPELINE resource definition

Web service binding file: A web service binding file contains abstract representations of the input and output messages used by the service. When a service provider or service requester application executes, CICS needs information about how the content of the messages maps to the data structures used by the application. This information is held in a web service binding file.

CICS resource relationships

The relationships between CICS Web services definitions are shown in Figure 4-3 on page 134.

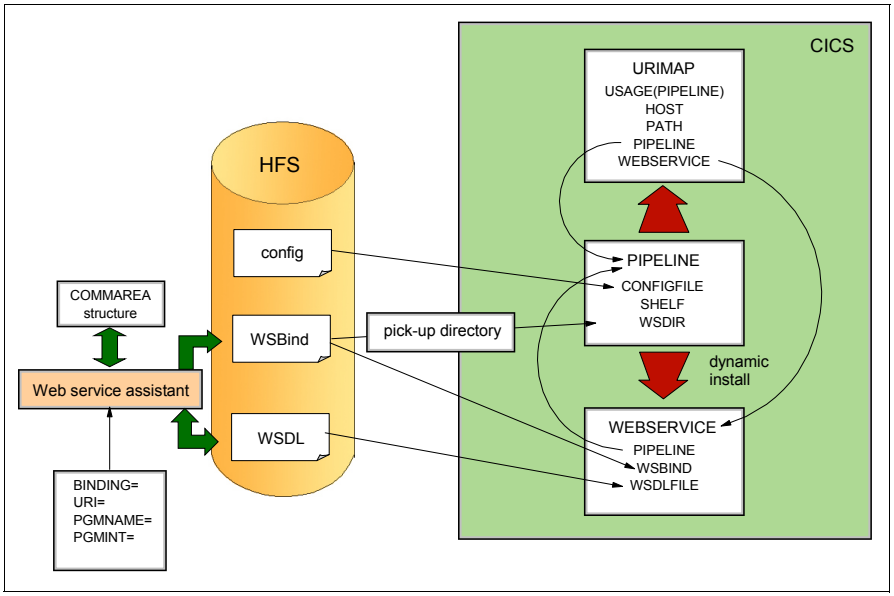


Figure 4-3 CICS Web services resource interrelationships

The following is sequence of steps are performed when CICS is a service provider:

1. When web service consumer sends a web service request, the URIMAP resource definitions are matched that has its USAGE attribute set to PIPELINE and its PATH attribute set to the URI found in the request.
2. If a matching URIMAP definition is found, the PIPELINE and WEBSER VICE definitions from the PIPELINE and WEBSERVICE attributes of the URIMAP definition are used.
3. The TRANSACTION attribute of the URIMAP definition determines the name of the transaction that must be attached to process the pipeline.

CICS support for Web service standards:

- It ensures maximum interoperability with other Web services implementations by conforming with the Web Services Interoperability Organization (WS-I) Basic Profile 1.1 and WS-I Simple SOAP Binding Profile1.0.
- It supports the WS-Atomic Transaction specification.
- It supports the WS-Security specification.
- It conditionally complies with WS-Trust, and support is subject to restrictions.

CICS as a service provider application

When CICS is a service provider, it receives a service request, which is passed through a pipeline to a target application program. The response from the application is returned to the service requester through the same pipeline. In this section, we first discuss how to prepare for running a CICS application as a service provider. We then discuss how CICS processes the incoming service request.

An existing COMMAREA-based application can be exposed as a service provider, normally without any application changes. When CICS is in the role of service provider, it must perform the following operations:

1. Receive the request from the service requester.
2. Examine the request, and extract the contents that are relevant to the target application program.
3. Invoke the application program, passing data extracted from the request.
4. Construct a response (when the application program returns control) using data returned by the application program.
5. Send a response to the service requester.

CICS as a service requester application

When CICS is a service requester, an application program sends a request, which is passed through a pipeline to a target service provider. The response from the service provider is returned to the application program through the same pipeline. In this section, we discuss how to prepare for running a CICS application as a service requester, and then we discuss how CICS processes the outbound service request.

When CICS is in the role of service requester, it must perform the following operations:

1. Build a request using data provided by the application program.
2. Send the request to the service provider.
3. Receive a response from the service provider.
4. Examine the response, and extract the contents that are relevant to the original application program.
5. Return control to the application program.

Web services using WebSphere MQ as transport

CICS can receive and send SOAP messages to WebSphere MQ using the WebSphere MQ transport, both in the role of service provider and service requester. As a service provider, CICS uses WebSphere MQ triggering to

process SOAP messages from an application queue. Triggering works by using an initiation queue and local queues. A local (application) queue definition includes the following information:

- The criteria for when a trigger message is generated. For example, when the first message arrives on the local queue, or for every message that arrives on the local queue. For CICS SOAP processing, specify that triggering occurs when the first message arrives on the local queue.
- The local queue definition can also specify that trigger data is passed to the target application, and in the case of CICS SOAP processing (transaction CPIL), this specifies the default target URL to be used if this is not passed with the inbound message.
- The process name that identifies the process definition. The process definition describes how the message is processed. In the case of CICS SOAP processing, specify the CPIL transaction.
- The name of the initiation queue that the trigger message must be sent to.

When a message arrives on the local queue, the Queue Manager generates and sends a trigger message to the specified initiation queue. The trigger message includes the information from the process definition. The trigger monitor retrieves the trigger message from the initiation queue and schedules the CPIL transaction to start processing the messages on the local queue.

As a service requester, on outbound requests you can specify that the responses for the target web service are returned on a particular reply queue. In both cases, CICS and WebSphere MQ require configuration to define the required resources and queues. Figure 4-4 shows how WebSphere MQ acts as transport for CICS service requester and service provider applications.

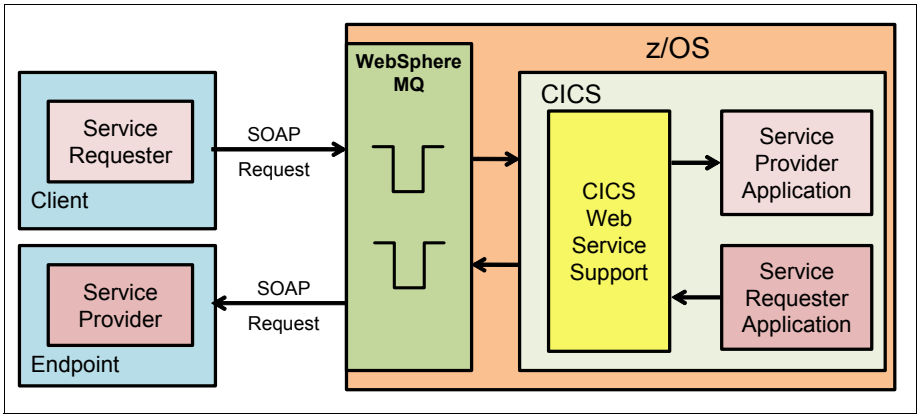


Figure 4-4 Example of WebSphere MQ transport flow

Support for MTOM

In standard SOAP messages, binary objects are base64-encoded and included in the message body, which increases their size by 33%. For large binary objects, the larger payload can significantly impact transmission time. CICS SOAP pipelines can support the Message Transmission Optimization Mechanism (MTOM) and XML-binary Optimized Packaging (XOP) specifications. These specifications define a mechanism for sending and receiving binary data using SOAP, without incurring the overhead of base64 encoding. CICS supports and controls the handling of MTOM messages in both web service provider and requester pipelines using an MTOM handler program and XOP processing.

Java web services using Axis2

Axis2 is a Java-based implementation of a web services SOAP engine that supports a number of the web services specifications. Axis2 is provided with CICS TS V4.2 to process web services in a Java environment.

You can optionally use the Axis2 Java-based SOAP engine to process web service requests in service provider and service requester pipelines. Because Axis2 uses Java, the SOAP processing is eligible for off loading to the IBM System z Application Assist Processor (zAAP).You can opt to use Axis2 by adding a Java SOAP handler to your pipeline configuration file and creating a JVM server to handle the Axis2 processing.

Enabling Axis2 does not require regenerating the binding files for any existing web services that use the pipeline. When CICS is a service provider, the Java-based handler uses Axis2 to parse the SOAP envelope for a request message. You can use header processing programs to process any SOAP headers associated with the SOAP message. Axis2also constructs the SOAP response message. This process is shown in Figure 4-5 on page 138.

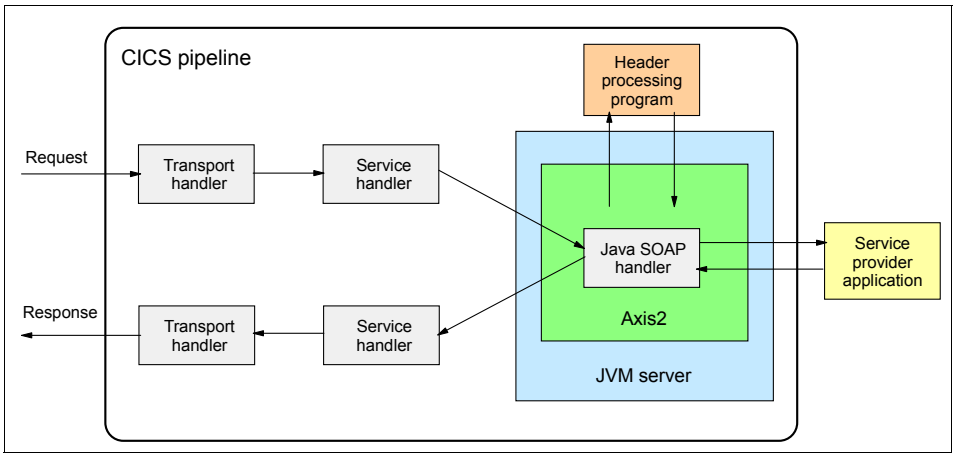


Figure 4-5 Axis2 processing when CICS is a service provider

Important: The response times might be slower when using Axis2, but you can off load the SOAP processing to zAAP.

Java web service topology using Axis2

In this section, we discuss the Java web service topologies using Axis2.

Topology 1

In this topology, a Java application is written using the Java Architecture for XML Binding (JAXB) and the Java API for XML Web Services (JAX-WS) libraries to generate and parse the XML. The Java application can now run in Axis2 in the same JVM server as the SOAP pipeline processing, as shown in Figure 4-6 on page 139, which is an example of a Java application that is a web service provider and is processed by the Axis2 SOAP engine in a JVM server. JCICS is a class library provided for API access to VSAM and TSQ.

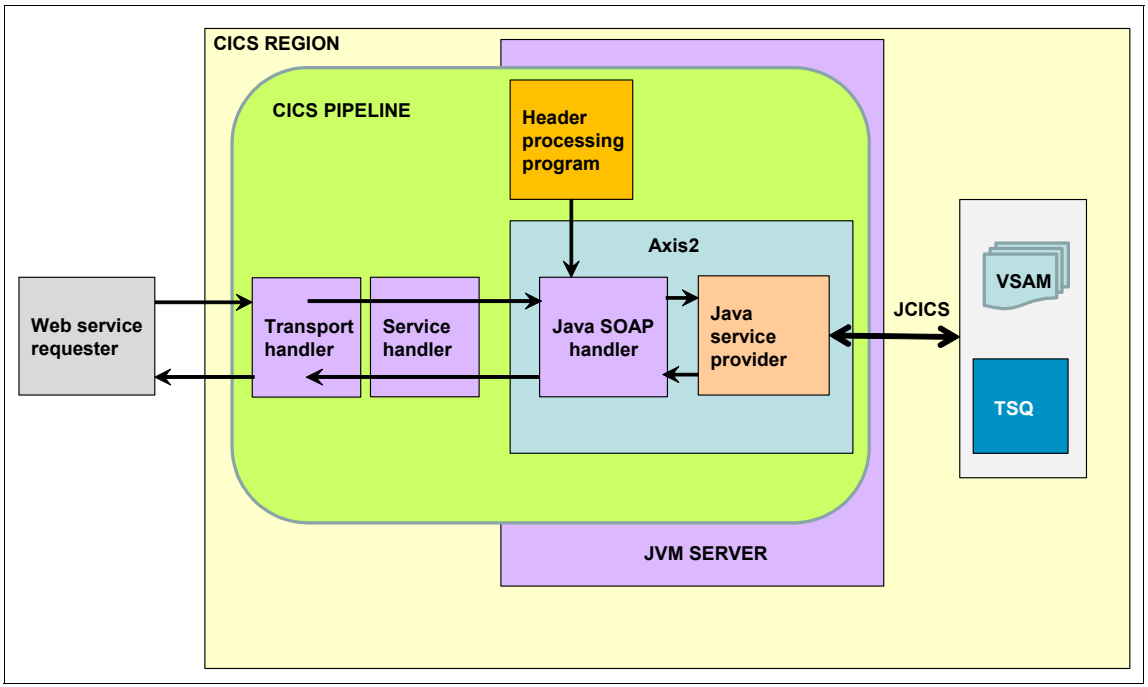


Figure 4-6 Web service with Java service provider and Axis2

Topology 2

The topology in Figure 4-7 on page 140 is an example of a COBOL application that is a web service provider. The request is processed in a pipeline that is configured to support Java. The SOAP handler is a Java program that are processed by Axis2 and run in a JVM server.

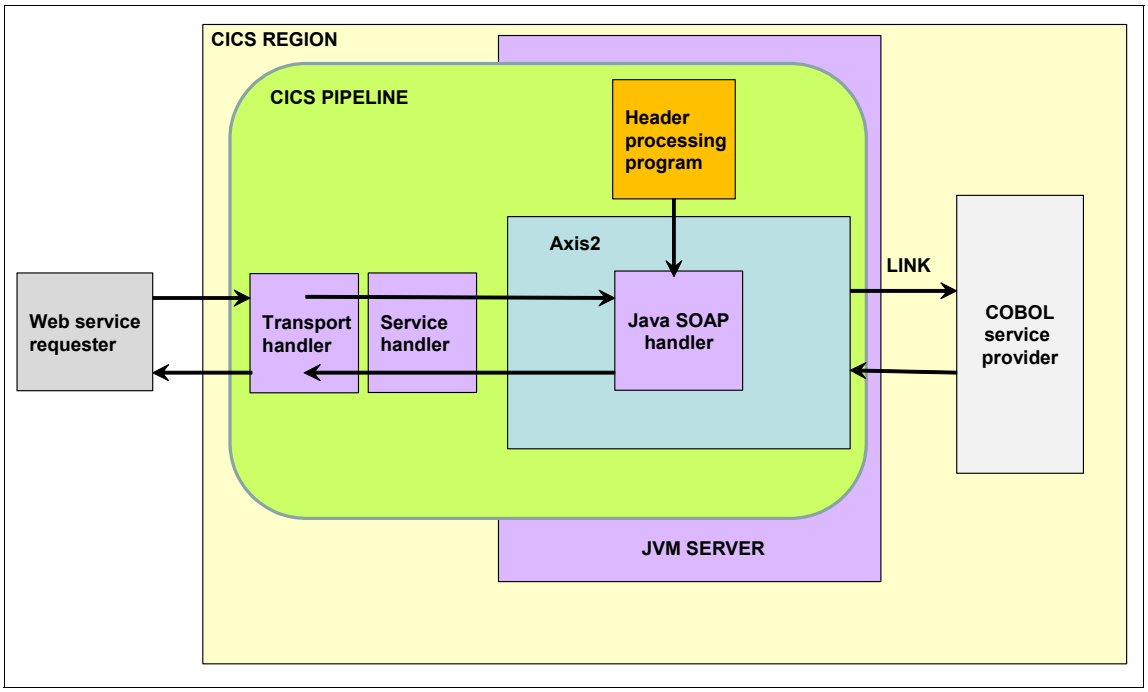


Figure 4-7 Web service with COBOL service provider and Axis2

Refer to Chapter 7, “CICS tools” on page 241 for further discussions about scalability on the two topologies.

4.4 CICS Transaction Gateway

The CICS Transaction Gateway (CICS TG) is a set of client and server software components that allow a remote client application to invoke services in a CICS region. The client application can be either a Java application or a non-Java application using either C, C++, COBOL or COM interfaces (depending on the platform used).

When a Java application is used, then the application can be any type of client (such as a servlet, or an enterprise bean). In the J2EE environment, the application is typically a servlet or enterprise bean that is deployed into a J2EE application server such as WebSphere Application Server. Figure 4-8 on page 141 shows how CICS assets can be accessed from multiple environments with CICS TG.

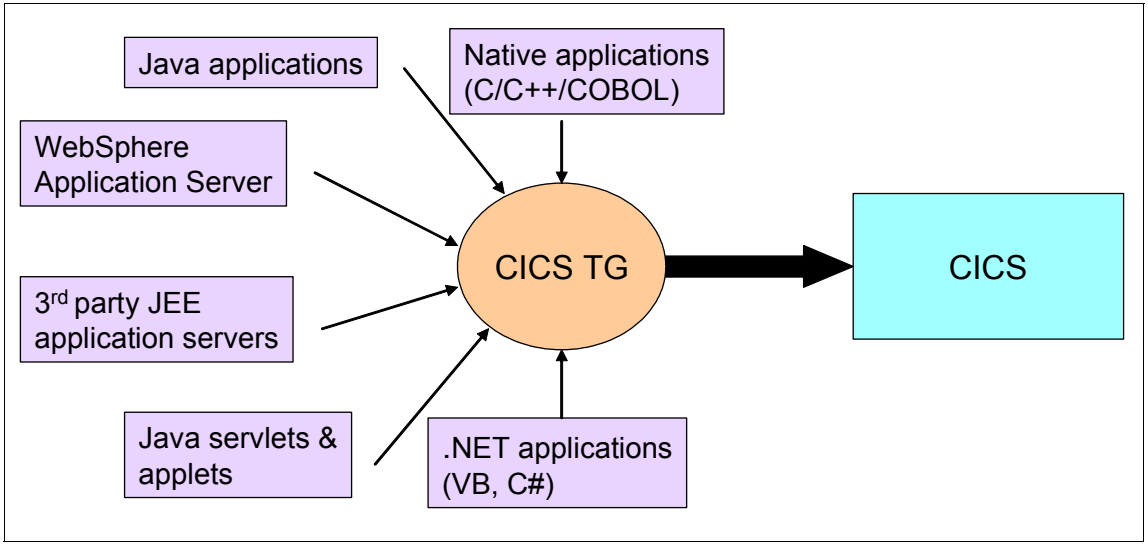


Figure 4-8 Extending CICS assets with CICS TG

4.4.1 CICS TG products

With CICS TG there are three distinct CICS TG products:

- CICS TG for Multiplatforms
- CICS TG Desktop Edition
- CICS TG for z/OS

4.4.2 CICS TG for Multiplatforms

CICS TG for Multiplatforms is supported on the following range of operating systems and platforms and is designed to support connectivity to all in-service CICS servers:

- Linux on System z
- Linux on Intel
- Linux on POWER®
- AIX
- HP-UX (on PA-RISC & Itanium)
- Solaris (on SPARC)
- Windows XP, Windows Vista, Windows 2003, Windows 2008 & Windows7

CICS TG for Multiplatforms consists of the following main runtime components:

- The Gateway daemon, which listens for incoming work and manages the threads and connections necessary to ensure good performance.It also provides connectivity to CICS.
- The Client daemon, which provides the communication to CICS servers and the non-Java APIs.
- A Java class library or JCA resource adapter, which is deployed into the client runtime environment. When used in a JCA environment the resource adapter is deployed into the J2EE application server.

A Java client program can connect to a remote Gateway daemon using the TCP or SSL protocols. The Client daemon then provides the transport drivers to connect to the CICS server.

CICS TG Desktop Edition

CICS TG Desktop Edition provides all the capabilities of CICS TG as described in the previous section. It is licensed and limited to single user access. It replaces the earlier product CICS Universal Client.

CICS TG for z/OS

CICS TG for z/OS is supported on z/OS and supports connectivity to CICS TS for z/OS. CICS TG for z/OS uses the external communication interface (EXCI) or IPIC provided by CICS TS to communicate with CICS (Figure 4-9). ECIv2 provides support for non-Java based applications.

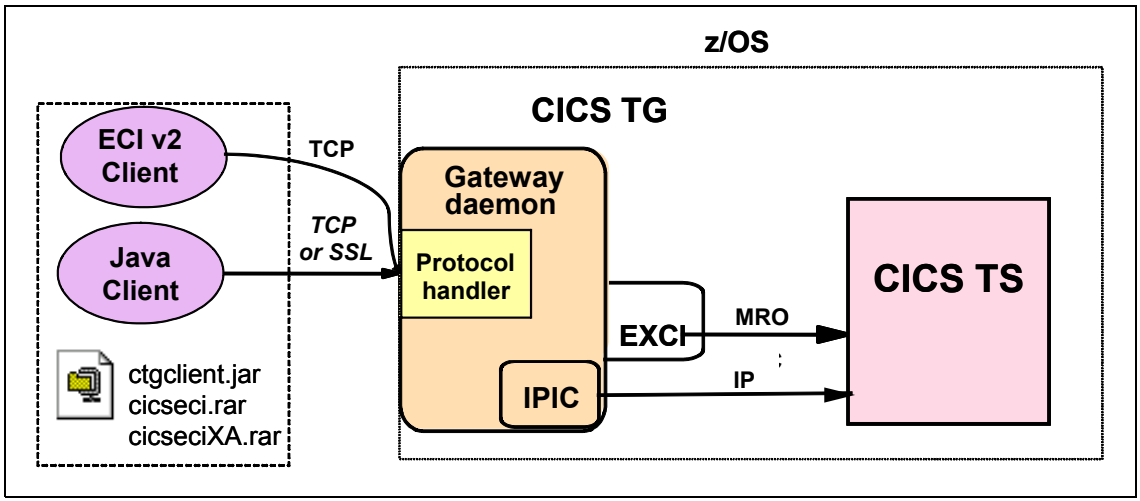


Figure 4-9 Components of CICS TG for z/OS

Figure 4-9 on page 142 depicts how java clients and ECIV2 clients can connect to CICS using the CICS TG on z/OS. CICS TG can be configured to use EXCI or IPIC to communicate with CICS.

CICS TG for z/OS modes of operation

There are two principle modes of operation for the CICS TG: remote and local.

Remote mode of operation

The *remote mode* of operation uses the Gateway daemon as a long running task which listens on specified ports for incoming ECI requests and then forwards them to the CICS server using the EXCI or the IPIC protocol. The Gateway daemon runs in its own address space and provides connection and thread management.

Local mode of operation

This is only available for Java clients. If the CICS TG is to be used on the same machine as WebSphere Application Server, it might be more efficient to use the CICS TG classes within WebSphere Application Server to provide the gateway functionality. This mode of operation allows WebSphere Application Server to manage the connections and threads and reduces the communications overhead. This configuration is known as the *local mode* of operation. In this mode, the gateway daemon runs inside WebSphere application server.

CICS TG local mode: To read more about CICS TG local mode of operation, refer to:

http://www.redbooks.ibm.com/abstracts/sg247161.html

CICS TG for z/OS has many advantages in the area of systems management, usability, and performance:

- XA transaction support enables CICS Transaction Server (CICS TS) for z/OS to participate in a global two-phase commit transaction that is initiated in a distributed JEE application server, such as WebSphere Application Server.
- Management of the Gateway daemon from SDSF provides better system administration capabilities.
- Better security with identity propagation, identity assertion and SSL directly into z/OS.

CICS TG application programming interfaces

CICS TG for Multiplatforms provides the following programming interfaces for accessing CICS applications:

- External Call Interface (ECI)

- External Presentation Interface (EPI)
- External Security Interface (ESI)

Note: The CICS TG on z/OS supports only the ECI call interface.

External Call Interface (ECI)

The ECI is used for calling COMMAREA or channel- based CICS programs. The COMMAREA and channel are the buffer that is used for passing the data between the client and the CICS server. CICS sees the client request as a distributed program link (DPL) request.

The ECI enables a user application to call a CICS program synchronously or asynchronously. It enables the design of new applications to be optimized for client-server operation, with the business logic on the server and the presentation logic on the client.

An ECI request can be invoked from a Java application using a variety of distinct interfaces:

- The ECIRequest class that is provided by the CICS TG base classes.

  This interface provides a simple procedural type interface to the ECI. It is supported in any Java environment (such as an stand-alone application) and provides similar capabilities to the JCA. However, it does not provide the same qualities of service (such as XA transaction support).

- The Common Client Interface (CCI) that is provided by the CICS ECI resource adapters (cicseci.rar or cicseciXA.rar).

  These classes define a standard architecture for connecting the Java 2 Platform Enterprise Edition (J2EE) platform to a heterogeneous EIS, such as CICS. Java applications interact with resource adapters using the Common Client Interface (CCI), which is a common framework of classes extended by each resource adapter to allow communication with a specific EIS. JCA provides high quality of service such as managed security, identity propagation and XA(2-phase commit).

The ECIv2 API enables a C client application to communicate with a CICS TG running on a remote machine in much the same manner as the Java APIs. An application developer using this API is able to call COMMAREA or CHANNEL based CICS applications. Applications can make use of the transaction support in CICS either by having each program run in its own transaction or by multiple calls within an extended logical unit of work (LUW). Applications written using this API can be developed for the Windows, Linux, or UNIX platforms that are supported by the CICS TG. This function is similar to that provided by the existing Client APIs in the CICS TG. However an application written using ECIv2 is not required to run on the same machine as a CICS TG installation.

An ECI request can use the following connections:

- EXCI
- IPIC

External Presentation Interface

The External Presentation Interface (EPI) invokes 3270-based transactions. A terminal is installed in CICS, and CICS sees the request as running on a remote terminal controlled by the CICS TG. This call interface is not supported on z/OS.

External Security Interface

The External Security Interface (ESI) verifies and changes the user ID and password information that is held in the CICS external security manager (ESM), such as RACF. It is based on the CICS Password Expiration Management (PEM) function.

ESI calls to CICS can only be made using the ESIRequest class that is provided by the CICS TG base classes. This is only available to Java APIs and when using APPC connections. This class provides a Java interface to the ESI, and provides two simple PEM requester functions:

Verify Password Allows a client application to verify that a password matches the password for a given user ID that is stored by the CICS ESM.

Change Password Allows a client application to change the password that is held by the CICS ESM for a given user ID.

There is no other interface available for the ESI, although both the EPI and ECI allow user IDs and passwords to be flowed within the actual requests. In this case, the user ID and password is authenticated either within CICS, if using a Multiplatform CICS TG, or within the CICS TG, if using the CICS TG for z/OS.

CICS TG and the JCA

The CICS TG is a J2EE connector for CICS TS, and in conjunction with IBM WebSphere Application Server provides, a high performing, secure, scalable, and tightly-integrated access method in CICS.

The JCA system-level contracts between a J2EE application server, such as WebSphere Application Server, and a resource adapter determine the scope of the JCA managed environment. The standard contracts include a connection-management contract, transaction-management contract and a security-management contract. These contracts provide the mechanisms by which the management of connections, security and transactions are performed:

- The connection-management contract enables the application server to pool and re-use connections into CICS, enabling a more scalable and efficient

environment that can support a large number of concurrent accesses to a CICS region.

- The transaction-management contract defines the scope of transactional integration between a J2EE application deployed in WebSphere Application Server and a CICS program.
- The security-management contract defines how security context information is passed between the application server and CICS.

Using the CICS ECI resource adapter with several topologies

The JCA system contracts are the keys to the qualities of service provided by WebSphere Application Server and the CICS ECI resource adapter. However, the qualities of service vary depending on the topology in use. The three most common topologies, shown in Figure 4-10 on page 147, are:

Topology 1 WebSphere Application Server and the CICS TG are both deployed on a distributed (non-System z) platform.

Topology 2 WebSphere Application Server is deployed on a distributed platform and the CICS TG is deployed on a z/OS system.

Topology 3 Both WebSphere Application Server and the CICS TG are deployed on System z.

We discuss topology 2 in more detail later in this book.

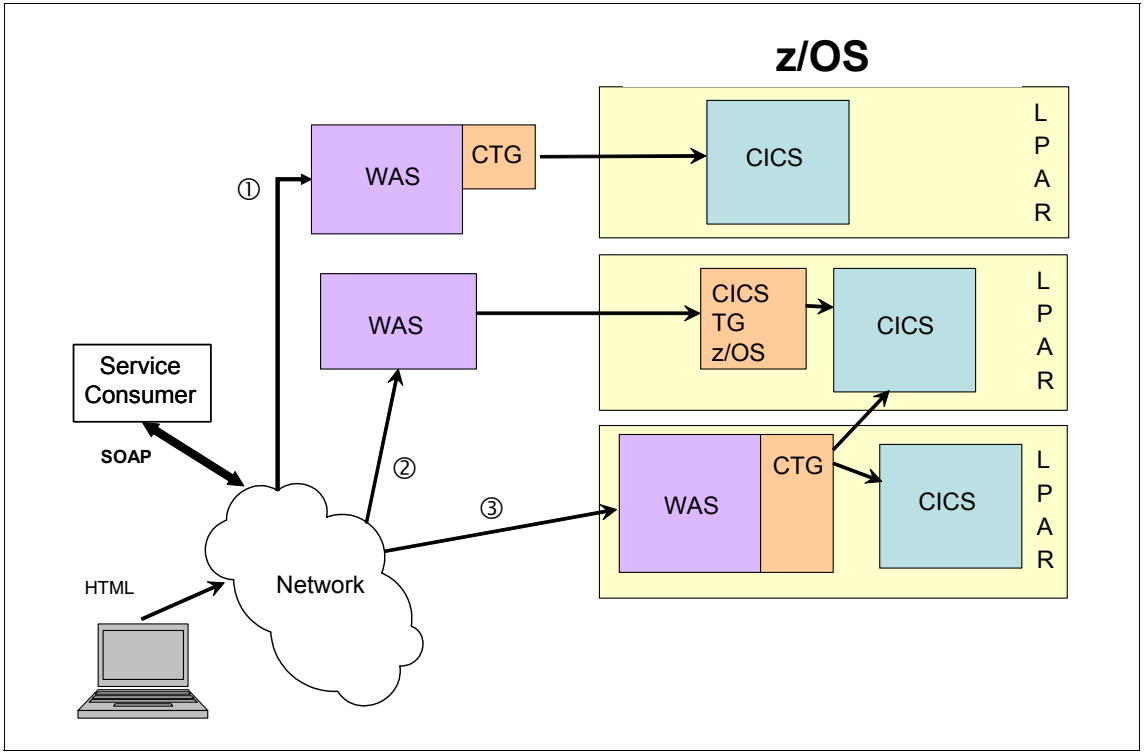


Figure 4-10 Common topologies for using CICS TG with WebSphere Application Server

Remote Gateway daemon on z/OS

In topology 2, where WebSphere Application Server is deployed on one of the distributed platforms, it is possible to access CICS through a Gateway daemon running on z/OS, as shown in Figure 4-11 on page 148.

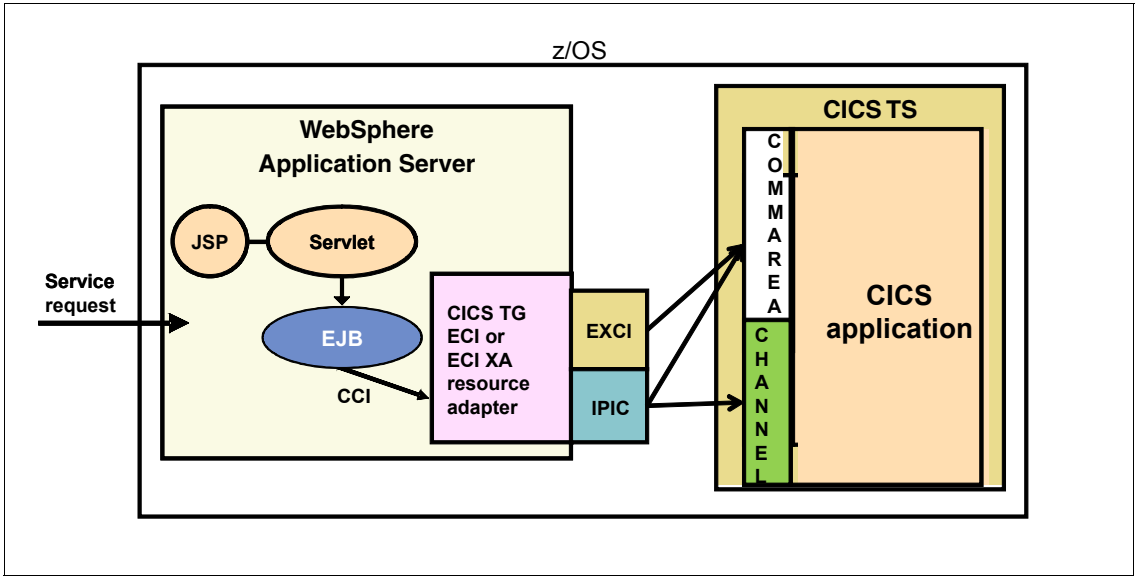


Figure 4-11 CICS TG Topology 2

In this configuration, the protocol used is one of the remote protocols (TCP, HTTP, SSL or HTTPS). The communication from the CICS Transaction Gateway on z/OS to the CICS server utilizes EXCI or IPIC.

This configuration is widely used today for the following reasons:

- Topology 1 only supports native TCP/IP connections into System z systems for CICS TS V2 onwards whereas topology 2 provides simple TCP/IP access to any release of CICS.
- Topology 2 provides advanced security with identity assertion and identity propagation.
- Topology 2 enables integration with z/OS IP workload-management functions, including Sysplex Distributor and TCP/IP port sharing for better HA.
- Deploying the CICS TG on z/OS can use the existing CICS systems-management skills within the enterprise for better system administration.
- Topology 2 provides XA transaction support, to enable resources to participate in two-phase commit transactions.

4.5 WebSphere Optimized Local Adapters

WebSphere Optimized Local Adapters (WOLA) is a functional component of WebSphere Application Server for z/OS that provides an efficient cross-memory mechanism for calls both inbound and outbound to and from WebSphere Application Server for z/OS. It can communicate with external address spaces, which includes CICS, batch, IMS, UNIX System Services and ALCS. Because it avoids the overhead of other communication mechanisms, it is capable of high volume exchange of messages. Figure 4-12 shows the topology of WOLA and CICS.

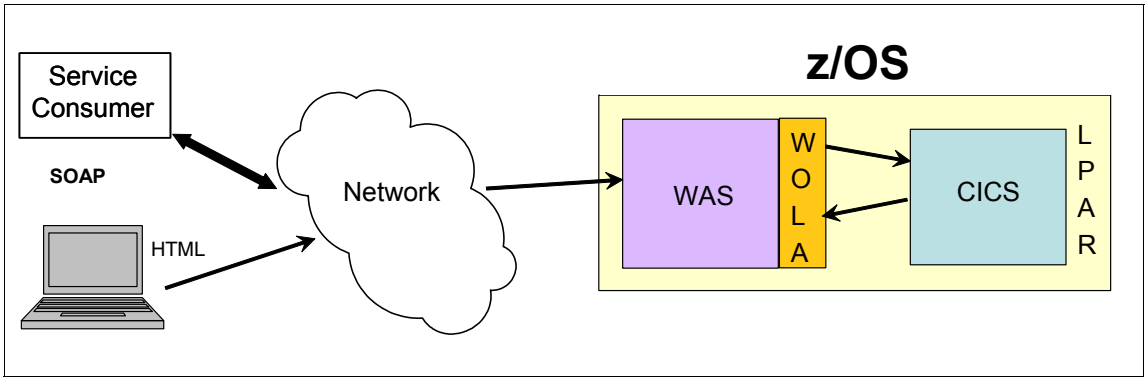


Figure 4-12 WOLA and CICS topology

What is WOLA

WOLA is:

- A low-level byte array pipe, and it does not care about the content of the data passed.
- An address space-to-address space communication link only. The address spaces must be on the same LPAR. WOLA is not a cross-LPAR or cross-platform technology.
- Any external address spaces that must use WOLA must register itself as a WOLA user.
- WOLA provides interfaces to connect in and out of WebSphere Application Server for z/OS (bidirectional connectivity).
- JCA resource adapter hides most of the WOLA implementation details which makes Java programming easy.

The benefit of WOLA

WOLA started out as a way to allow program access into WebSphere Application server for high transaction rate batch programs. There are various good technologies to connect to WebSphere Application server but there were some overhead and limitations from batch. Inside WebSphere Application Server for z/OS, there is a local communication function that sparked the idea of WOLA. The local communication function was externalized using interface modules so that it can be accessed from other address spaces. A standard JCA resource adapter was deployed into WebSphere Application Server for z/OS so that Java programs can have a standard Common Client Interface (CCI) to interact with.

CICS and WOLA

Figure 4-13 shows the high-level overview of the WOLA support in CICS.

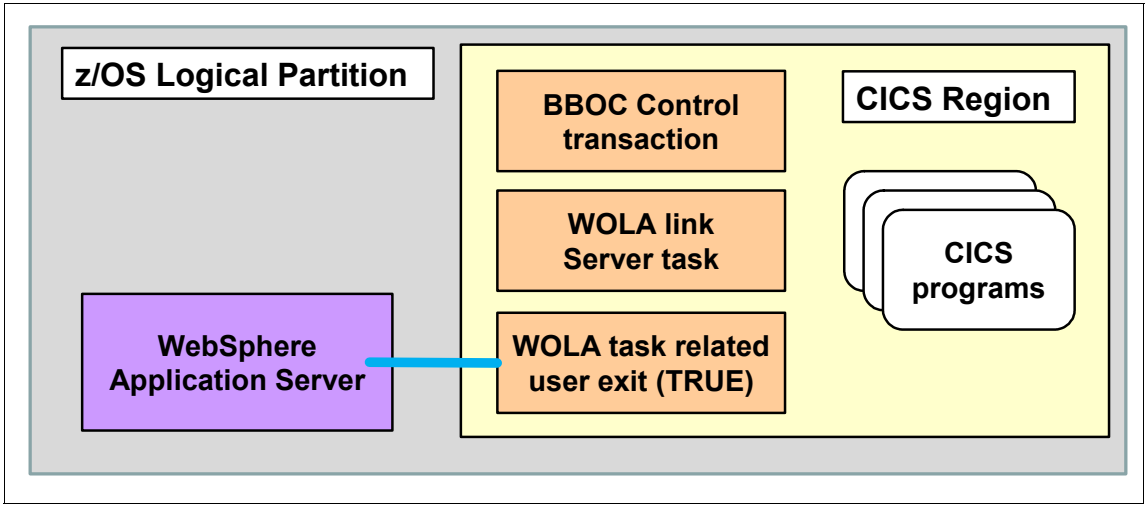


Figure 4-13 WOLA support in CICS

The three components inside CICS that provide WOLA support are:

- The Task Related User Exit (TRUE)

  This is the low-level heart of the WOLA support in CICS. It provides the low-level module support for access to the local communication function of WebSphere Application Server z/OS. CICS cannot use WOLA without it, and this must be installed in any CICS region that wants to use WOLA. The TRUE adapter is designed to run in a CICS region as a resource manager. In CICS, the Task Related User Exit (TRUE) is the primary vehicle used by resource providers. TRUE support provides the boundary between the CICS application threads and the external resource manager threads. Currently,

DB2, WebSphere MQ, and TCP/IP sockets execute in CICS using the TRUE support.

- The Link Server Task

  The link server task hides the CICS programs from the specifics of WOLA. It serves as a kind of "WOLA handler" from WebSphere Application Server to CICS. It accepts the request, and then turns and uses EXEC CICS LINK (DPL) of the named target CICS program passing either COMMAREA or channel/container. The Link Server Task does not come into play for calls from CICS to WebSphere Application Server. For CICS to WebSphere Application Server calls, some custom coding must occur with the WOLA API.

- BBOC 3270 control transaction

  This is a utility provided to make the management easier and is handy. The BBOC control transaction provides a convenient way to start and stop the

  TRUE and the Link Server Task and pass in parameters to modify the behavior of the environment.

Calls to CICS with WOLA

A Java program from WebSphere Application server on z/OS that wants to initiate an optimized local adapter (OLA) call outbound can be implemented as either a servlet or EJB. The Java program codes to the supplied JCA resource adapter (ola.rar) using the class files are supplied in the development tooling support.

In CICS, you can use the supplied link server task to act as the receiving agent on behalf of existing CICS program assets. The link server task (BBO$ by default) receives the call and issues an EXEC CICS LINK of the program named. No changes to the existing CICS program are necessary provided it supports either COMMAREA or CHANNEL. WOLA supports named containers. Figure 4-14 on page 152 shows how the call from WebSphere Application Server to CICS works.

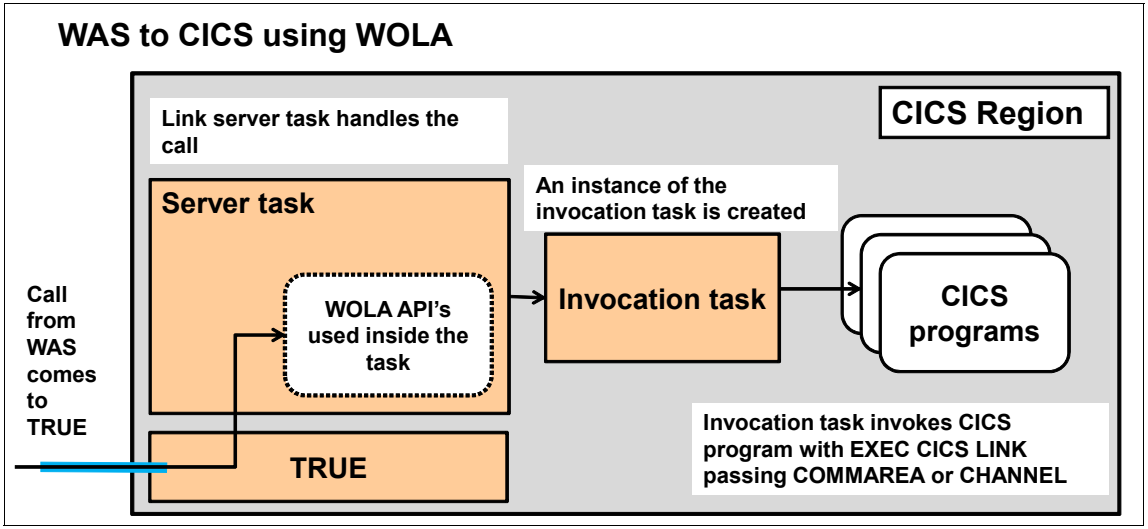


Figure 4-14 Inbound call to CICS

Calls from CICS with WOLA

Calls from CICS to WOLA involves some degree of coding to the APIs. The link server task does not assist in calls outbound from CICS to WebSphere Application Server. There are two approaches for coding the APIs, as shown in Figure 4-15 on page 153:

- Embed the WOLA API processing in the CICS program itself. This is easy enough to do if you are writing a new application or can easily modify an existing one. But it is a bit more problematic for existing programs where the mandate is to leave it untouched for modifications.
- The second option is to write a kind of custom WOLA bridge program that can be DPL'ed by your application programs. This becomes a kind of "WOLA Service" to existing CICS programs to utilize as your needs require.

Figure 4-15 on page 153 shows how outbound calls from CICS to WebSphere Application Server.

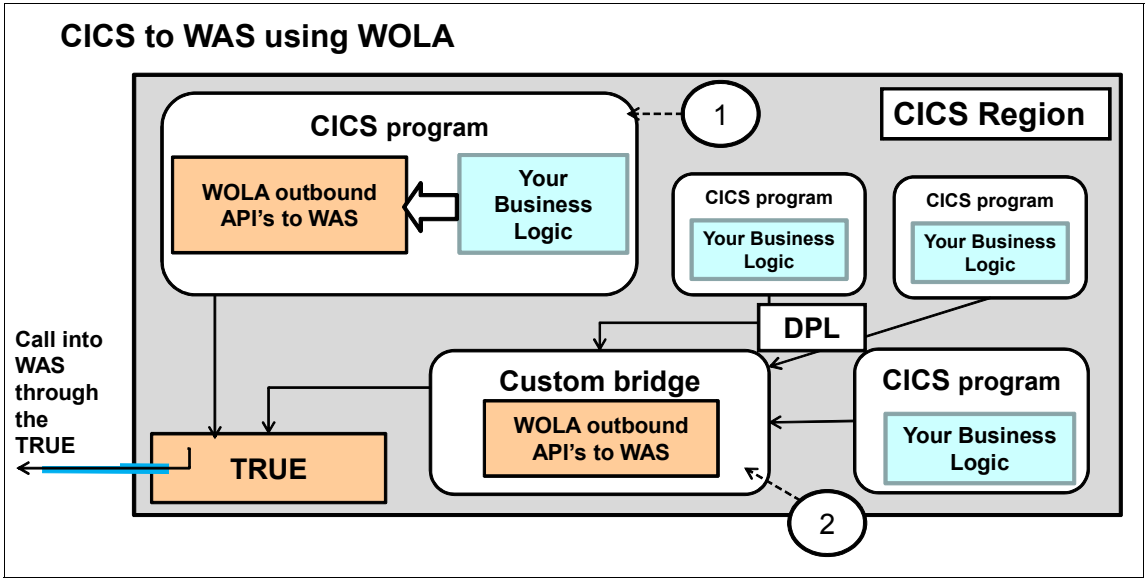


Figure 4-15 Outbound call from CICS

4.6 CICS web support

CICS web support is a set of resources supplied with CICS TS for z/OS that enable a CICS region to act as an HTTP server and as an HTTP client. This allows CICS applications to be invoked by and reply to HTTP requests. An overview of CICS Web support is illustrated in Figure 4-16.

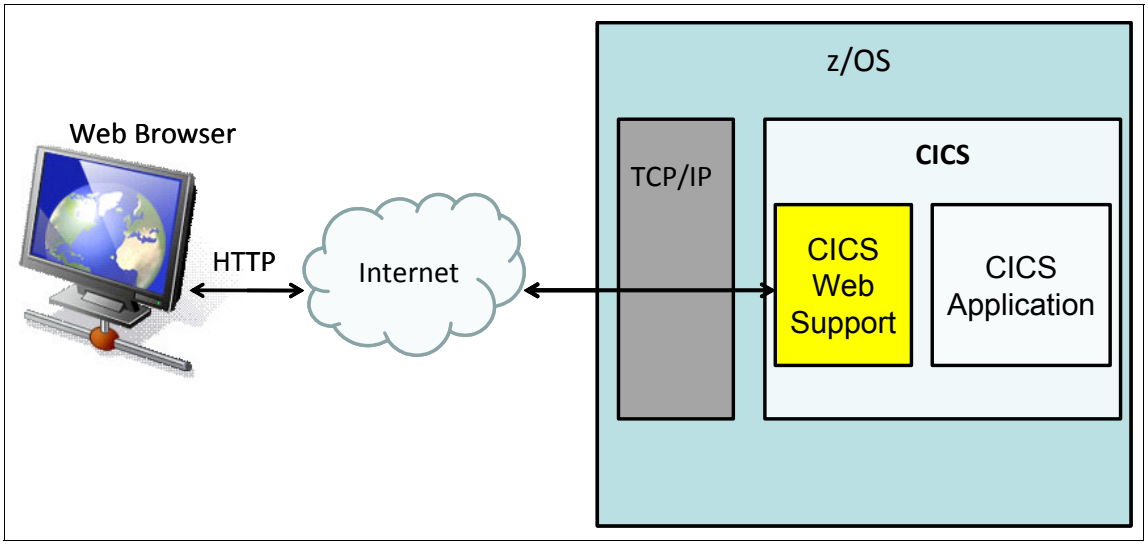


Figure 4-16 CICS web support overview

4.6.1 CICS as an HTTP server

When CICS is an HTTP server, a web client can send an HTTP request to CICS and receive a response. The response can be a static response created by CICS from a document template or static file or an application-generated response that is created dynamically by a user application program.

The actions of CICS as an HTTP server are controlled by TCPIPSERVICE definitions and URIMAP definitions, which are used to configure CICS web support and instruct CICS how to process requests and responses.

Processing flow for CICS as an HTTP server

Figure 4-17 on page 155 shows the process flow of CICS as an HTTP server:

1. User invokes a CICS using a URL.

2. The long running Sockets listener task detects inbound TCP/IP connection requests on all ports defined to CICS and invokes the CICS service associated with the port.
3. When the TCPIPSERVICE definition for a port has the protocol HTTP, the default transaction ID for the Web attach task is CWXN. When the protocol is USER, the default is CWXU. An alias can be used instead, but the transaction always executes program DFHWBXN.
4. Web attach task matches the URL to a URIMAP definition and looks at the PROGRAM attribute to run the application program for processing the request. It runs under the default alias transaction, CWBA.

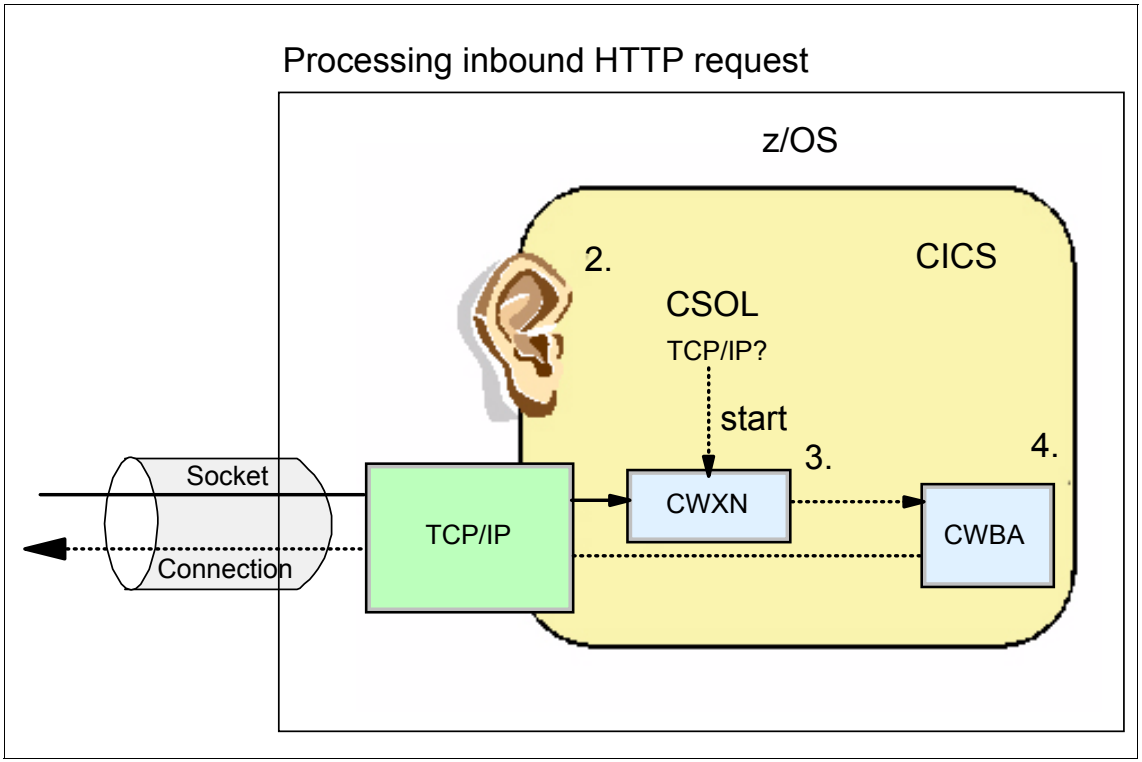


Figure 4-17 Processing inbound HTTP request

Note: The CICS TCP/IP listener is completely separate from, and not to be confused with, the TCP/IP Socket Interface for CICS, which provides an application level TCP/IP socket interface to CICS applications, and is described further in 4.8, “CICS sockets” on page 165.

CICS as an HTTP client

When CICS is an HTTP client, a user-application program in CICS can initiate a request to an HTTP server and receive a response from it. The actions of CICS as an HTTP client are controlled by user-written application programs. An application program that makes an HTTP request and receives a response must use the EXEC CICS WEB API commands.

Processing for CICS as an HTTP client

Processing for CICS as an HTTP client takes place as a sequence of EXEC CICS WEB APIs to send the request by establishing a socket connection, write HTTP header, write the body and transmit the request. Response processing is also done in a sequence of EXEC CICS WEB APIs to receive the data.

Connection pooling for outbound HTTP connections

By default CICS closes client HTTP connections after an application has finished using the connection. In CICS TS V4.2, you can set up connection pooling to reuse the connection to the same host and port. CICS places the connection in a pool in a dormant state. Connection pooling is implemented by specifying a non zero value to the SOCKETCLOSE time-out option on a URIMAP resource definition.

CICS also supports non-HTTP messages: Support for non-HTTP requests is primarily intended to provide support for requests from user-written clients that use non standard request formats. The processing that takes place for requests, and the response that is provided, are defined by the user. No specific support is provided for any formally defined protocols which are used for client-server communication

Base components for CICS web support

The base components for CICS web support are:

- Secure Sockets Layer (SSL) support provides security for the CICS Web support implementation.
- TCP/IP Listener: The Sockets listener task detects inbound TCP/IP connection requests, and invokes CICS Web support by attaching the Web attach task.
- HTTP Handler: Web attach task (CWXN) receives data from the Web client and deal with initial processing of the request.

Resource components for inbound HTTP requests

The resource components for inbound HTTP requests are:

- TCPIPSERVICE resource definitions are used to define each port that you use for CICS as an HTTP server.
- URIMAP resource definitions match the URLs of requests from Web clients, or requests to an HTTP server, and provide CICS with information about how to process the requests.
- TRANSACTION resource definitions are used to define alias transactions for HTTP request processing.This performs the CICS business logic. This is for inbound request processing.

Resource component for outbound HTTP requests

The resource component for outbound HTTP requests is the URIMAP, which is the only resource definition required for outbound HTTP requests from CICS.

Programming components

CICS provides EXEC CICS WEB application programming interface for handling HTTP requests and responses. Refer to chapter 6 for more details.

ATOM feeds

ATOM is both a protocol and an XML format for content providers to provide XML-based web feeds of updated content. An ATOM feed is a web feed that is provided using the ATOM protocol and format. This provision of updated content is known as syndicating a web feed. Web users can subscribe to a feed that allows them to see new content as soon as it is made available.

A Web feed, sometimes just called a *feed*, is a series of related items that a content provider publishes on the Internet. An ATOM feed is a web feed that uses the Atom Syndication Format and the Atom Publishing Protocol. ATOM comprises an XML-based format that describes an ATOM feed and the items of information in it, and a protocol for publishing and editing ATOM feeds.

This format and protocol are described in two Internet Society and Internet Engineering Task Force (IETF) Request for Comments documents (known as RFCs):

- RFC 4287, The Atom Syndication Format, available from: `http://www.ietf.org/rfc/rfc4287.txt`
- RFC 5023, The Atom Publishing Protocol, available from: `http://www.ietf.org/rfc/rfc5023.txt`

Content providers often deliver Web feeds in an earlier format called RSS. CICS supports ATOM, but does not support RSS.

The items of information that make up an ATOM feed are known as *Atom entries*. A content provider publishes, or *syndicates*, an ATOM feed by making it available through a URL on the Internet and updating it with new items. Web pages can display the items in the ATOM feed, and Web users can obtain the items from the feed using a feed reader or Web browser.

An ATOM feed might be used as part of a mashup, which is a web application that merges content from a number of data sources so that users can experience and understand the data in a new way. In a mashup, the data from the ATOM feed can be handled by a widget, which is a script application that runs in a web page.

ATOM feeds in CICS

CICS can serve ATOM feeds to web clients. The ATOM feeds consist of data that is supplied by CICS resources or application programs. When you expose a CICS resource or application program as an ATOM feed or collection, users can read and update the data by making HTTP requests from external client applications, such as feed readers or Web mashup applications.

CICS's ATOM support allows you to quickly expose a VSAM file, a TS queue, or an application program as an ATOM feed, as shown in Figure 4-18.

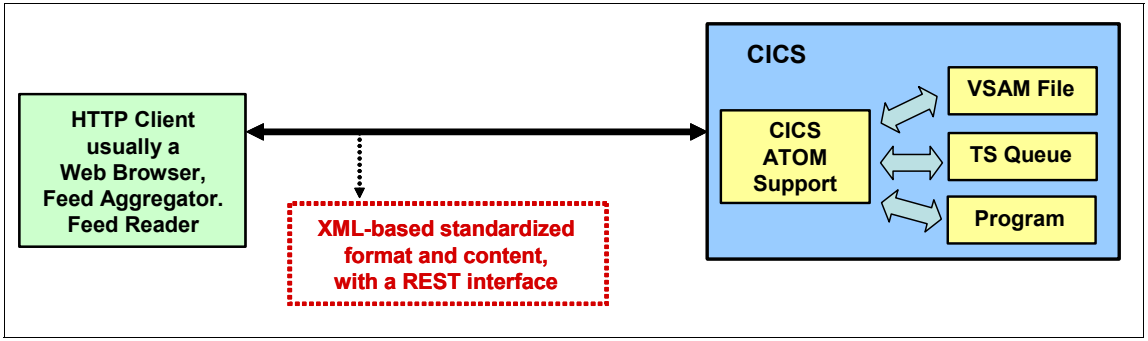


Figure 4-18 CICS ATOM support

CICS ATOM support

The CICS ATOMSERVICE resource definition defines an ATOM service, feed, collection, or category document. It identifies the ATOM configuration file, CICS resource or application program, and ATOM binding file that supply the data for the feed. URIMAP resource definitions handle the incoming requests and point to the appropriate ATOMSERVICE resource definition:

- The ATOM configuration file contains XML that specifies the metadata and field names for the ATOM document that is returned for this resource definition.

- The XML binding file specifies the data structures used by the resource named in RESOURCENAME, which supplies the data for the ATOM document that is returned for this resource definition.

Figure 4-19 shows how CICS implements ATOM feeds.

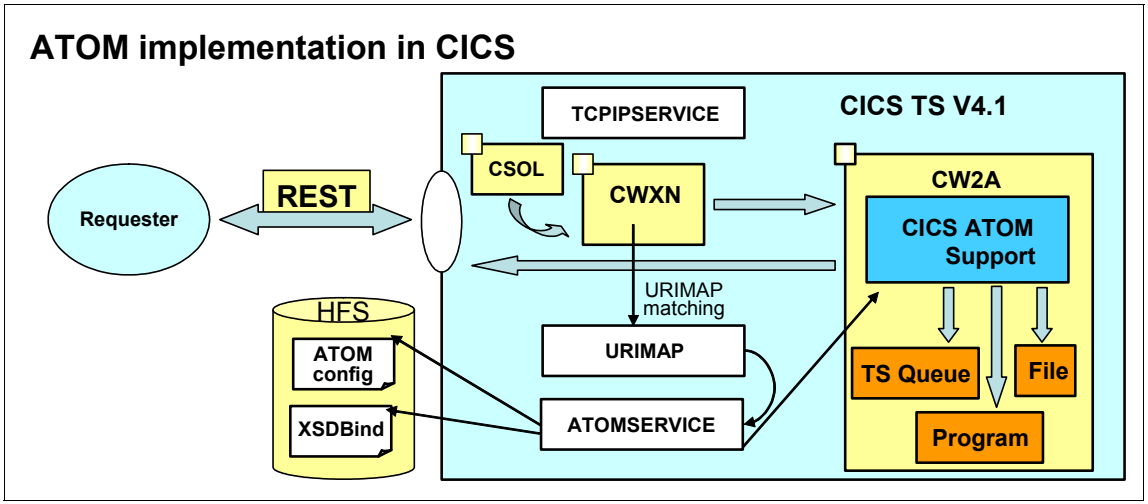


Figure 4-19 CICS ATOM implementation

4.7 WebSphere MQ

WebSphere MQ is a family of products that are available on all major operating system platforms. It provides an open, scalable, industrial-strength messaging and information infrastructure enabling enterprises to integrate business processes.

WebSphere MQ provides Java Message Service (JMS) APIs and native WebSphere MQ APIs for use by clients on a wide variety of platforms. This makes it suitable for communication between a range of types of application and CICS applications.

Figure 4-20 on page 160 shows a typical scenario for access of a CICS application using WebSphere MQ.

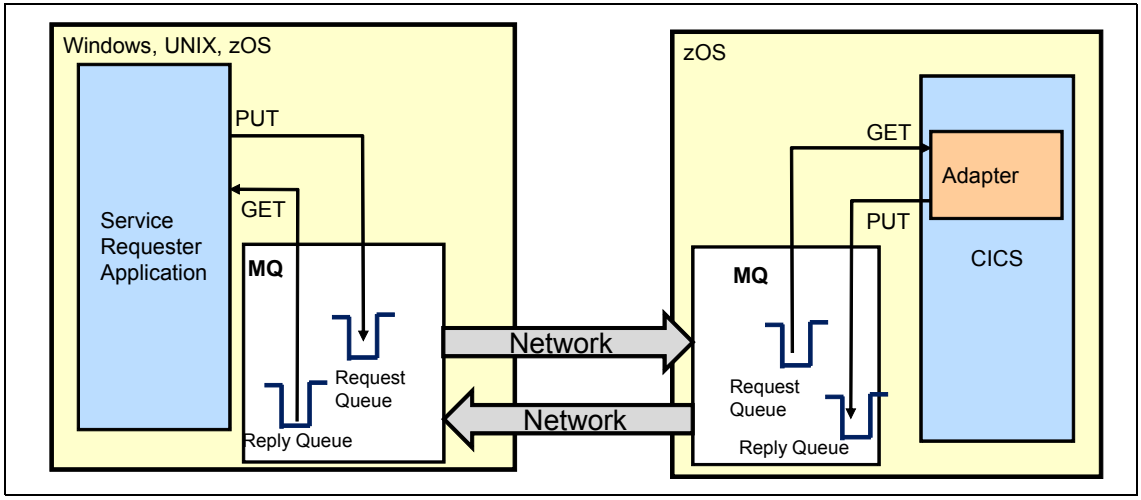


Figure 4-20 Using WebSphere MQ as a CICS access component

Figure 4-20 shows a request reply integration scenario. It uses WebSphere MQ as the transport to achieve a pseudo synchronous call. The service requester application puts a request message on a request queue and waits for the response on the reply queue. The response message's correlation ID is the same as the request message's message ID. WebSphere MQ is configured such that the request message arrives on the queue on the target platform where CICS runs. CICS provides the connection through the CICS-WebSphere MQ adapter, which allows the CICS application to process the request message and send a response to the reply queue passing the same correlation ID that came with the request.

CICS supplies the CICS-WebSphere MQ adapter to integrate CICS applications with WebSphere MQ.

4.7.1 CICS-WebSphere MQ adapter

The CICS-WebSphere MQ adapter runs in the same address space as CICS and provides three major functions:

- Managing the connections between CICS and WebSphere MQ.
- Supporting the use of MQ API calls from CICS applications.
- Support for CICS triggering with the CICS trigger monitor transaction (CKTI).

4.7.2 CICS integration with MQ

In this section, we discuss CICS integration with MQ:

- Topology 1: MQ API enabled CICS applications:
  - MQ Triggering

  Figure 4-21 shows how an arrival of a message on queue triggers a CICS application to process the message from a MQAPI aware application.

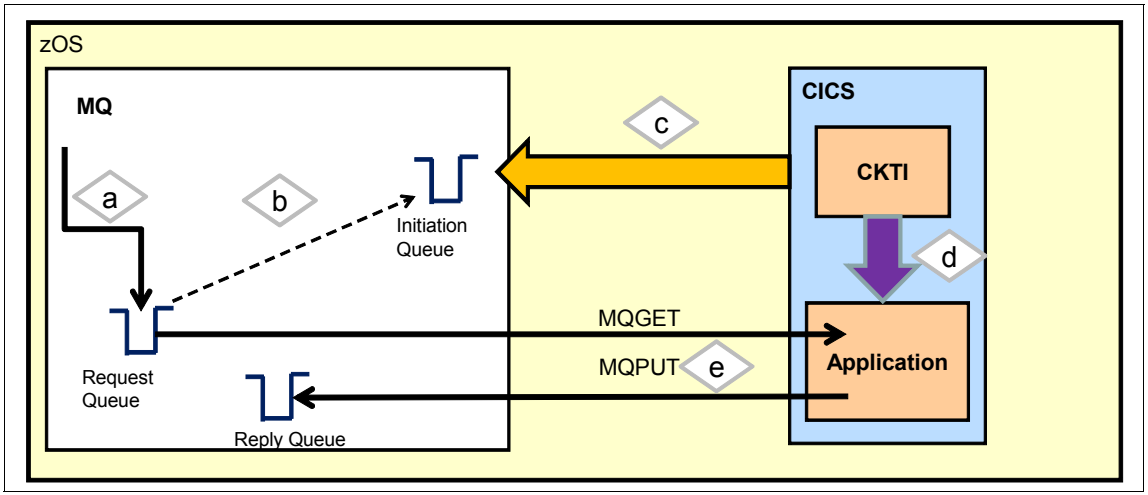


Figure 4-21 CICS MQ triggering

In Figure 4-21:

1. A message arrives on a request queue to be processed.
2. If the triggering conditions are met, MQ puts a trigger message to the initiation queue that was specified for the request queue.
3. CKTI is a CICS supplied transaction that monitors the initiation queue. CKTI has been set up in CICS to monitor the initiation queue. CKTI is a long running daemon process. It gets the trigger message from the initiation queue.
4. CKTI starts another application transaction passing the MQ trigger message. The application transaction that needs to be started is defined in the request queue in the process name.
5. The CICS application retrieves the trigger message and gets the name of the queue which started the trigger. The started transaction processes the message using MQGETs and MQPUTs.

Attention: The initiation queue and the process are set by WMQ administrator rather than by CICS administrator.

– MQ API based application with no triggering

This is a topology where the CICS application does not want to be triggered and might have an application defined situation to process the messages. In this topology, no triggering is defined in the request queue. It is entirely up to the application when to start processing the messages using the MQ APIs. These applications can be both inbound or outbound to CICS.

- Topology 2: CICS-WebSphere MQ bridge for non MQ API aware application

In this topology, you use the CICS-WebSphere MQ bridge to run a transaction on CICS containing no WebSphere MQ calls. The bridge makes the MQ API call. You can therefore re engineer existing CICS applications to be controlled from other platforms by WebSphere MQ messages, without having to rewrite, recompile, or relink them. The request messages include the WebSphere MQ CICS information header (the MQCIH structure) to supply control options for the CICS applications. The following types of CICS application are suitable for use with the CICS-WebSphere MQ bridge:

– Program interface (DPL bridge)

CICS programs that are called using the EXEC CICS LINK command, known as distributed program link (DPL) programs. You can use the CICS-WebSphere MQ bridge to run a single CICS program, or a set of CICS programs that form a unit of work. Figure 4-22 on page 163 shows how the CICS-WebSphere MQ bridge works.

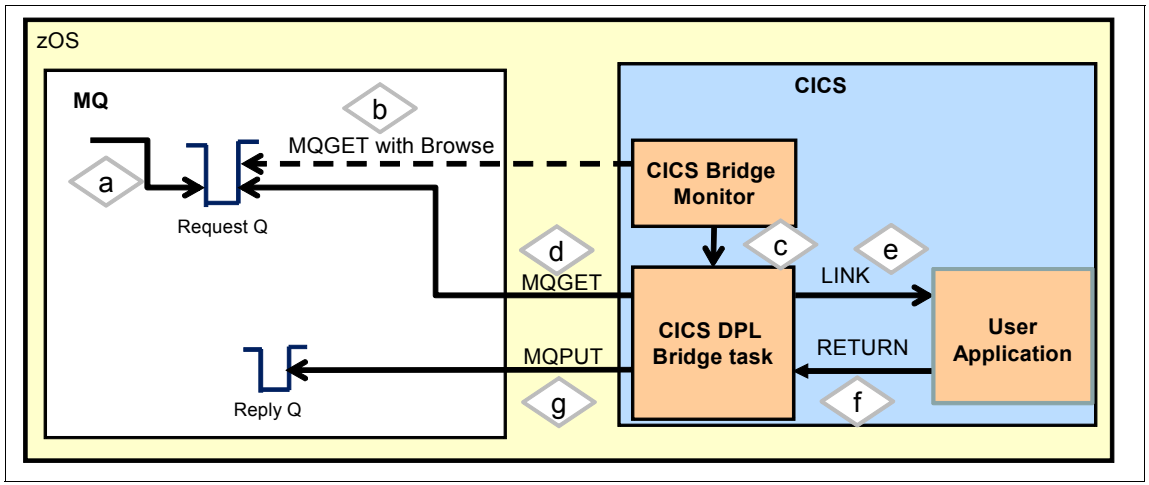


Figure 4-22 CICS-WebSphere MQ bridge workflow

In Figure 4-22:

1. A message arrives on a request queue to be processed.
2. The CICS bridge monitor task, which is constantly browsing the queue, recognizes that a *start unit of work* message is waiting.
3. The CICS bridge monitor starts a CICS DPL bridge task is started.
4. The CICS DPL bridge task makes an MQGET call to remove the message from the request queue
5. The CICS DPL bridge task builds a COMMAREA from the data in the message and issues an EXEC CICS LINK command for the program requested in the message.
6. The program returns the response in the COMMAREA used by the request.
7. The CICS DPL bridge task reads the COMMAREA, creates a message, and puts it on the reply-to queue specified in the request message.

   Terminal interface (3270 bridge): 3270 transactions are CICS transactions that were designed to be run from a 3270 terminal. The transactions can use Basic Mapping Support (BMS) or terminal control commands. They can be conversational or part of a pseudo conversation. You can use the CICS-WebSphere MQ 3270 bridge using the CICS Link3270 mechanism to access the CICS transaction. The request message provides the data needed to run a CICS 3270 transaction. The CICS transaction runs as though it has a real 3270 terminal, but instead uses one or more WebSphere MQ messages to communicate between the CICS transaction and the WebSphere MQ application. The workflow is similar, as shown in Figure 4-22.

- Topology 3: Asynchronous consume

  Asynchronous consume in WebSphere MQ is way of invoking a program by the arrival of a message instead of doing a MQGET waiting for the message to arrive. It is a message driven function directly invoked by the queue manager. The advantage of asynchronous consume is that application threads are not tied up.

  CICS message adapter supports the MQAPI MQCB (register message consumer) which is used to register a CICS load module as a callback routine. It supports the MQAPI MQCTL which starts the message consumer. Figure 4-23 shows an example of asynchronous consumption of message.

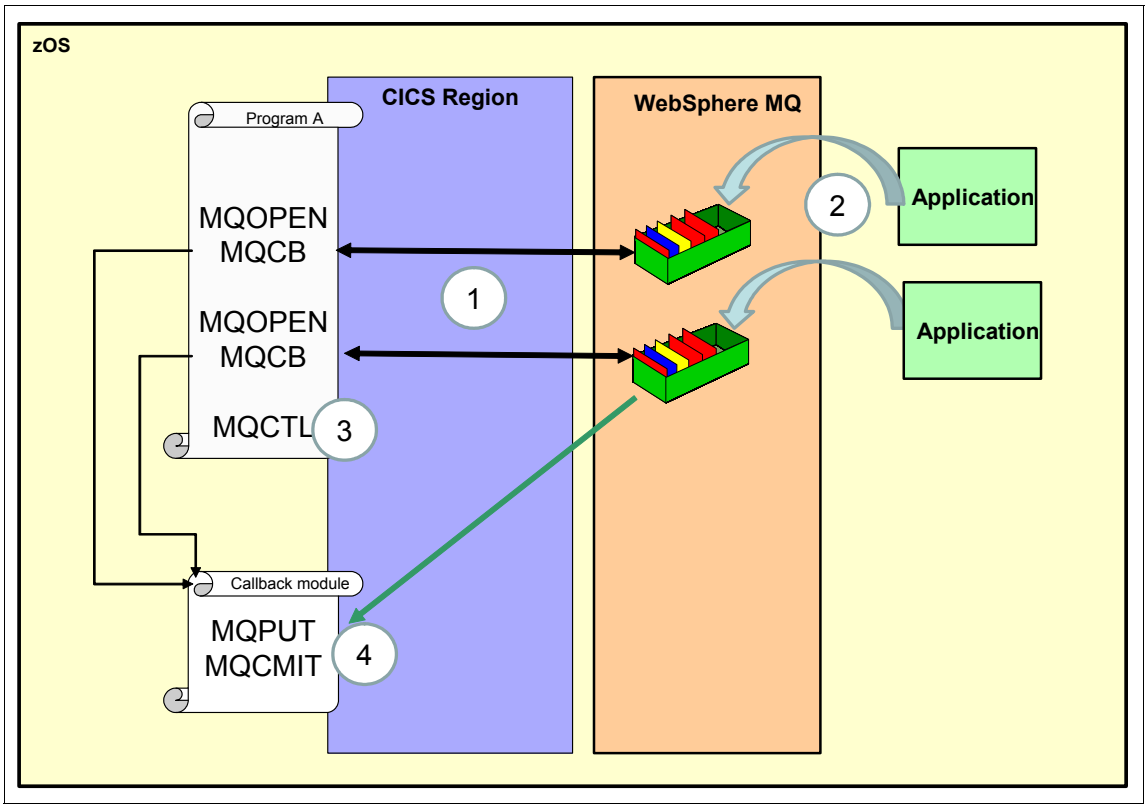


Figure 4-23 Asynchronous consume

In Figure 4-23:

1. Program A in CICS has opened Queues and registered a message consumer Program B using the MQCB API.

2. Remote applications puts messages on Queues that needs to be processed in CICS.
3. Program A uses the MQCTL API to start the message consumer.
4. The callback module is started and the message from the queues are passed to it in a CHANNEL. The callback module processes the message and can optionally send the response back to a reply queue.

4.8 CICS sockets

The TCP/IP Socket Interface for CICS (also known as CICS sockets) is a feature of z/OS Communications Server that brings the TCP/IP sockets API to your CICS applications. It is illustrated in Figure 4-24.

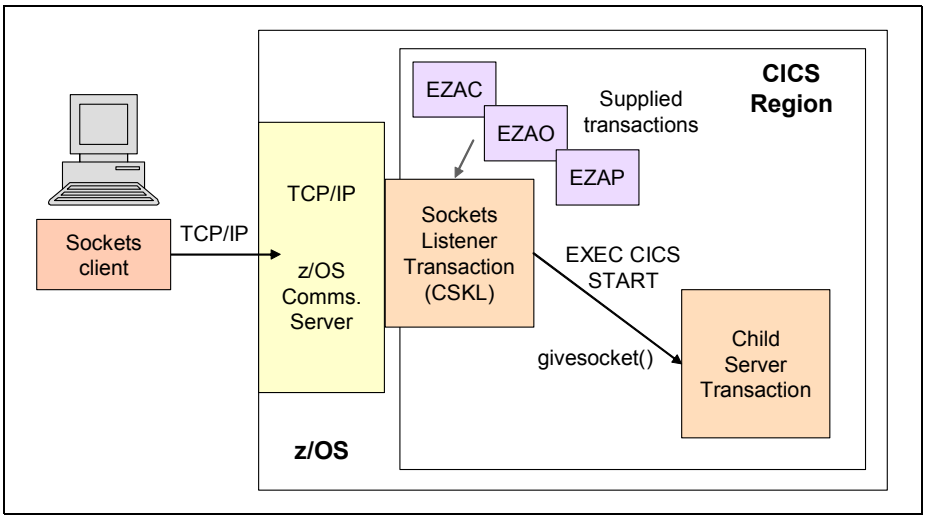


Figure 4-24 CICS to TCP/IP Sockets Interface

The main function of the CICS to TCP/IP Sockets Interface is provided by the Sockets Listener transaction (CSKL). This is a long running task that listens for incoming TCP connection requests on a specified port. The provided EZAC transaction can be used to configure the Sockets Listener, and in addition the EZAO and EZAP transactions can be used for operational requests. It uses a CICS Task Related User Exit (TRUE) to enable the use of native socket functions.

After CSKL receives a TCP connection request, it performs an **`EXEC CICS START`** command for a child server transaction and passes control of the socket conversation using the givesocket() call. The child server transaction is the user-written socket application and must retrieve the socket data using an EXEC CICS RETRIEVE and take control using a takesocket() call. This user application can be written in any language supported by CICS, including C, COBOL, Assembler or PL/I. The design of the client/server communication is entirely up to the user application, which is responsible for all design issues including authentication and data conversion issues.

Note: The child server transaction can also initiate outbound TCP connections and receive incoming requests.

For further details, refer to the Communications Server manual, *IP CICS Sockets Guide,* SC31-8518, and to the book *CICS/ESA and TCP/IP for MVS Sockets Interface,* GG24-4026.

Part 3

CICS Systems management and the CICS Tools

In this part of the book, we review the following management facilities, and CICS Tools.

- CICSPlex SM
- CICS Explorer
- CICS Tools
  - CICS Performance Analyzer for z/OS
  - CICS Interdependency Analyzer for z/OS
  - CICS Configuration Manager for z/OS
  - CICS Deployment Assistant for z/OS
  - CICS VSAM Recovery
  - CICS VSAM Transparency
  - CICS Batch Application Control
  - IBM Session Manager for z/OS

  - CICS Online Transmission Time Optimizer for z/OS
- IBM Problem Determination Tools

5

CICSPlex SM overview

In this chapter, we provide an overview of CICSPlex SM and describe the features that CICSPlex SM provides to manage complex transaction processing environments.

5.1 CICSPlex SM introduction

The CICSPlex System Manager element of CICS Transaction Server for z/OS Version 3 Release 2 (CICSPlex SM) is a system management tool that enables you to manage multiple CICS systems across multiple images from a single control point. Enterprises in which CICSPlex SM might be needed range from those running only a few CICS systems, to those running several hundred (or more) CICS systems. In the latest MVS sysplex environment, having such large numbers of CICS systems to support a transaction-processing workload is becoming an increasing requirement.

Each CICS region to be managed by CICSPlex SM is called a managed application system (MAS). The MASs are defined and managed as part of a CICSplex. Each MAS in a CICSplex is managed by a CICSPlex SM address space (CMAS).

To run the CICSPlex SM component of CICS Transaction Server for z/OS Version 3 Release 2, a MAS can be executing CICS Transaction Server for z/OS Version 2 Release 2, Version 2 Release 3, Version 3 Release 1, or Version 3 Release 2 on a system running an MVS image. However, the CMAS and the WUI server must execute the same release of CICS TS and CICSPlex SM.

The MASes in a CICSplex can be managed by several CMASs, but only one CMAS is defined as the maintenance point (MP) CMAS. This MP CMAS is responsible for keeping the data used by each CMAS synchronized.

CMASs communicate across defined CMAS-to-CMAS links, which are typically used for routing management commands and data between CMASs. The CICSPlex SM Web User Interface (WUI) is now, since CICS Transaction Server for z/OS Version 3 Release 2, the only user interface provided (the MVS/TSO ISPF user interface is no longer provided). The WUI is a server that runs on a dedicated CICSPlex SM local MAS at the same CICS Transaction Server release level as the connected CMAS.

Resource definitions are managed through Business Application Services (BAS). Workload management (WLM), Real Time Analysis (RTA), and monitoring services are used to manage the CICSPlex SM configuration and CICSplex environment, and gather statistical information.

All CICSPlex SM components, resources, system management requirements, and the relationships between them are held as objects in a data repository. These objects can be manipulated using the WUI user interface views. The batched repository-update facility is provided for the batched creation of CICSPlex SM resource definitions. Figure 5-1 on page 171 shows an overview of

the components in the CICSPlex SM. These are all discussed in 5.2, “Basic CICSPlex SM components” on page 172.

For CICSPlex SM’s purposes, a CICSplex is any grouping of CICS systems that you want to manage and manipulate as though they were a single entity. That is, a CICSplex is a management domain, made up of those CICS systems for which you want to establish a single system image (SSI). A CICSplex managed by CICSPlex SM can include every CICS system in your enterprise, or alternatively, you can define multiple CICSplexes. Each of these include a logical grouping of CICS systems. For example, a CICSplex can comprise all CICS systems on a particular MVS image or all CICS systems accessible by a subset of your users. It can perhaps even be all CICS systems serving a particular geographical area. Furthermore, the composition of a CICSplex can be altered without affecting the functions of the underlying CICS systems. The CICS systems in a single CICSplex managed by CICSPlex SM do not have to be explicitly connected to each other for management purposes.

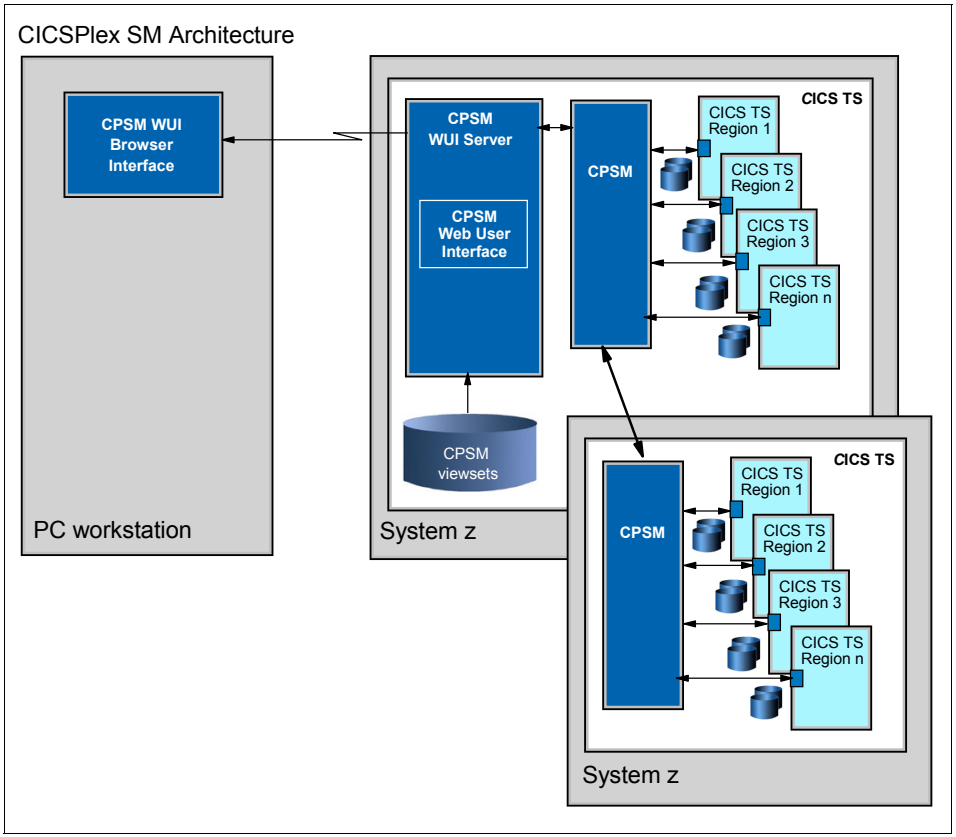


Figure 5-1 CICSPlex SM components

The most significant facts about a CICSplex managed by CICSPlex SM are:

- A CICSplex can be just on a single MVS image or on a sysplex or multiple sysplexes (for example, spanning multiple countries).
- The CICSplex is the largest unit that can be managed from a single point-of-control. That is, you cannot group CICSplexes and manipulate such a group as a single entity.
- You cannot copy CICSPlex SM data from one CICSplex to another. For system management purposes, the CICSplex is “sealed” against other CICSplexes.
- A CICSPlex SM managed CICS system can only be active in one CICSplex at a time.

CICSPlex SM enables you to define subsets of a CICSplex, which are known as CICS system groups. CICS system groups are not mutually exclusive, and can reference the same CICSPlex SM definitions. Thus, if you decide to include every CICS system in your enterprise in a single CICSplex, there are mechanisms for managing groups of CICS systems within the CICSplex as though each group were a single system. You can assign an unlimited number of CICS systems and CICS system groups to an existing CICSplex. Although you can define a CICS system to only one CICSplex, you can assign a CICS system to multiple CICS system groups within the CICSplex. You can also assign the CICS system group to any number of other CICS system groups.

Important: CICS TS V3.2 requires z/OS (5694-A01) V1.7 or later.

5.2 Basic CICSPlex SM components

Running CICSPlex SM for CICS Transaction Server for z/OS Version 3 Release 2 requires, at minimum, three dedicated address spaces:

- CICSPlex SM Address Space (CMAS)
- CICSPlex SM Web User Interface (WUI)
- Environment Services System Services (ESSS)

The CMAS and the WUI server are both dedicated CICS regions. They do not run or contain application code, and must be used solely for CICSPlex SM.

A CICS region that becomes part of a CICSplex is known as a Managed Application System.

Note: We recommend that in a real production environment the WUI server be placed in a CICS region as a dedicated MAS for CICSPlex SM WUI services.

CMAS overview

The CMAS is a CICS region dedicated solely to the CICSPlex SM function. It is responsible for managing and reporting all CICS regions and resources within the defined CICSplexes.The CMAS interacts with CICSPlex SM agent code running on each managed CICS region (MAS) to define events, conditions of interest, collect information gathered or to report on as a result of such definitions.

A maintenance point (MP) CMAS is the owner of a CICSplex. When more than one CMAS is involved in managing a CICSplex, then the CMAS that was used to create and define the CICSplex becomes the maintenance point. This is basically the master repository for the CICSplex. For the permanent modification to the environment or repository, the maintenance point CMAS must be available.

Important: The maintenance point for a CICSplex cannot be changed without deleting and redefining the whole plex. Make sure you know that the CMAS within which you are defining the plex is on the correct one.

The CICSplex topology (BAS, MON, WLM, and RTA definitions) and configurational definitions are all stored in the CMAS repository data set, EYUDREP.

In more complex environments, multiple CMAS regions can communicate between MVS regions to make up CICSplexes. These can consist of CICS regions across a sysplex or located at geographically separate sites.

For simplicity, we only consider a single CMAS for the Web User Interface server and all MAS regions here.

Note: A CMAS region is not part of a managed CICSplex although it manages one or more CICSplexes.

MAS overview

To be registered as a MAS to CICSPlex SM, a CICS region requires the following to be defined in the startup procedure:

- CICSPlex SM agent load libraries (SEUYLOAD and SEYUAUTH)
- CICSPlex SM parameter defining to which CMAS the CICS will connect to (a DD statement called EYUPARM in the CICS startup procedure)
- DFHSIT parameter denoting the type of CICS connection to be made (CPSMCONN=LMAS)

Note: DFHPLT program EYU9XLCS is now no longer required in the MAS.

The CICSPlex SM agent code is executed during CICS initialization to register the MAS with its respective managing CMAS.

An agent code that is running on the MAS communicates pertinent statistical and monitoring data back to the CMAS to which it connects.

Note: MASes do not have to run the latest version of CICS code.

The CICSPlex SM Web User Interface server

Traditionally, CICSPlex SM user interaction took place through the MVS/TSO ISPF user interface (EUI). However, the Web browser-based interface is now the *only* user interface to CICSPlex SM. This requires the use of a second dedicated CICS region known as a Web User Interface (WUI) region. The Web User Interface makes use of CICS Web Services to allow interaction with a Web Browser over TCP/IP. For more information about this topic, see 5.3, “CICSPlex SM Web User Interface” on page 174.

A page of information from the web user interface region presented on the browser is known as a *WUI view*; a collection of web user interface views is known as a *view set*.

The Environment Services System Services

There is one further address space that is automatically created upon startup of the CMAS, called the Environment Services System Services (ESSS). It is a limited function system address space that provides MVS system services to the CICSPlex SM components.

Note: There is one ESSS per CICSPlex SM release per MVS image, that is, you will have multiple ESSSs when migrating to a new release.

5.3 CICSPlex SM Web User Interface

The CICSPlex SM Web User Interface (WUI) offers an easy-to-use interface that you can use to carry out operational and administrative tasks necessary to monitor and control CICS resources. You can link to the Web User Interface from any location that provides IP network connectivity and firewall security access to a host from a workstation to the WUI server.

5.3.1 CICSPlex SM Web User Interface overview

The WUI is supplied with a set of linked menus and views to facilitate all your system management tasks. The WUI can also be customized to reflect your business procedures or to suit the needs of individual users.

The CICSPlex SM Web User Interface allows you to:

- Create clear, uncluttered menus and displays (called views) that present only the information that you want the user to see.
- Structure your data in a task-oriented way. You can:
  - Organize the user interface by resource category, by user task, or by application.
  - Define the links between views.
  - Define the buttons that will appear on a display and what they will do.
- Customize the layout of data. You can:
  - Have as many views of the same object as you like, each one showing a different selection of data depending on the user task.
  - If you have a Java-enabled browser, you can use graphical presentations of your data: You can have either a bar gauge that shows, for example, the number of tasks active in a CICS region, or a warning light that can be configured to change color or flash, depending on the threshold values you define for the field.
- Customize the panels to your business needs. You can:
  - Use terminology appropriate to your business.
  - Limit the data that is displayed using filters, so that users see only the data relevant to their task.
  - Include information for the user's guidance, for example, contact names and telephone numbers.
  - Define text that is written on action buttons.
  - For each menu choice, add explanatory text to help the user in the task.
  - For each view, provide buttons that accomplish a specific task, for example, a shutdown button on a CICS regions view.

Assign views to a set of favorites for quick and easy access. This allows you to reach frequently used views with just one click. Administrators have the additional authority to update and maintain the favorites of other users.

5.4 Customer environments

In this section, we discuss customer environments

5.4.1 The Bank

The Bank used CICS and CICSPlex SM for many years. Originally CICS was used to drive teller's terminals in branches. To better manage resources and provide consistent response times as increasing numbers of automated teller machines were added, they installed CICSPlex SM. With the increased popularity of internet banking, they are observing a greater transaction volume entering through TCP/IP.

The Bank's core banking configuration consists of:

- Sysplex Distributor for workload management of TCP/IP connections across two LPARs (TB01 and TB02)
- A terminal owning region (TOR) on each LPAR which listens on a shared TCP/IP port:
  - CICSTOR1 on TB01
  - CICSTOR2 on TB02
- Transaction requests that are dynamically routed to cloned application owning regions (AOR) using CICSPlex SM:
  - CICSAR10, CICSAR11, and CICSAR14 on TB01
  - CICSAR12, CICSAR13, and CICSAR15 on TB02
- CICS core banking programs that run in the AORs share access to the customer and accounts operational database using DB2 data sharing.
- CICSPlex SM provides a single point-of-control and dynamic workload transaction balancing. CICSPlex SM is configured with two CICSPlex SM address spaces (CMASes):
  - CPSMTB01 on TB01
  - CPSMTB02 on TB02
- A WUI server, CICSWUI1, on LPAR TB01
- CICS Explorer for CICSplex systems management

A map of The Bank's topology is shown in Figure 5-2 on page 177.

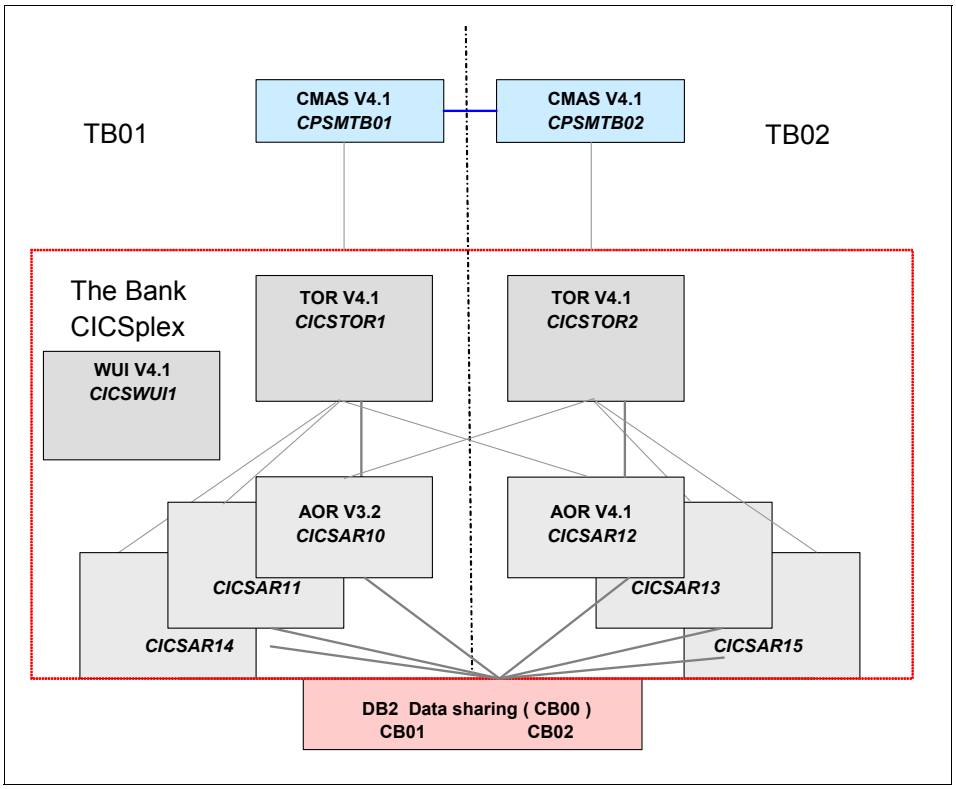


Figure 5-2 The Bank's CICSPlex SM topology

The Bank is in the late stages of a conversion from CICS Transaction Server V3.2 to V4.1 at this time. They implemented V4.1 of CICSPlex SM across their CICSplex. Both their TORs and the AORs on LPAR TB02 were upgraded to CICS TS V4.1, and the AORs on LPAR TB01 are scheduled to be upgraded in the near future.

5.4.2 The Store

The Store recently implemented CICSPlex SM. They invested heavily in technology to manage their online catalog and ordering systems and plan to use BAS to allow them to manage deployment of resources across their CICSplex.

The Store's CICSplex currently runs in a single LPAR. Their configuration consists of:

- Two TORs support point-of-sale and management terminals and online catalog and ordering systems.

- Three cloned AORs access databases managed by a DB2 subsystem.
- CICSPlex SM supports dynamic workload balancing and resource management and deployment.
- A single CMAS manages their CICSplex.
- Several workstations with CICS Explorer connecting through a WUI server provide access for management of their CICSplex.

The Store's CICSplex is shown in Figure 5-3.

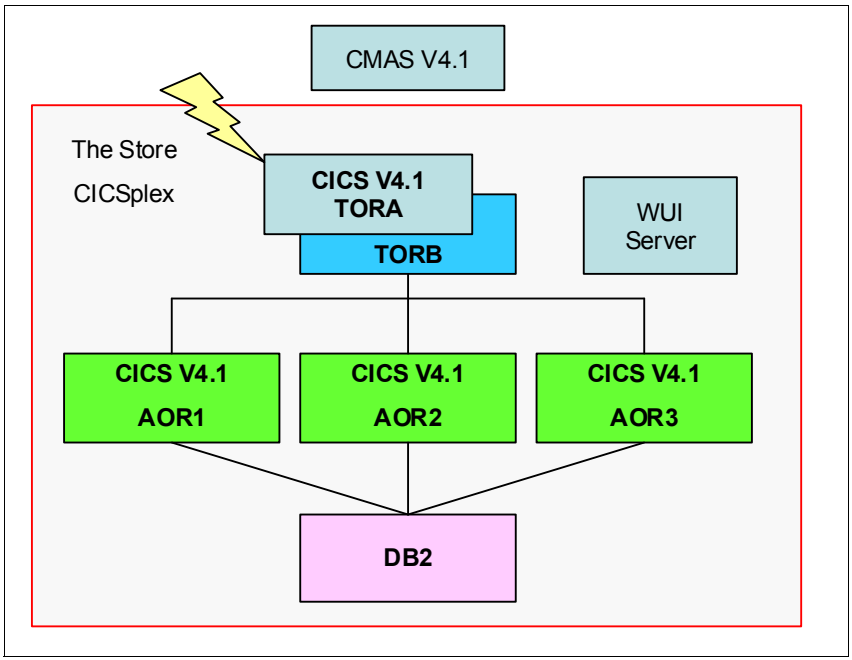


Figure 5-3 The Store's CICSPlex SM topology

5.5 Single system image

CICSPlex SM provides a real-time, single-system image (SSI) of all CICS regions and resources that make up each CICSplex in the transaction processing environment. CICSPlex SM creates an inventory of CICS systems and resources, and maps the relationship between them. This mapping is know as the CICSplex topology. CICSPlex SM maintains three levels of topology data:

- Network topology maps the relationship between CICSplex managing address spaces (CMAS) based on active CMAS-to-CMAS connections.

- Region topology maps the relationship between MASes and the CMASes that manage them and between CICS systems and CICS system groups. A CICS system group is a user defined collection of CICS systems that have a common property. A CICS system can be a member of many CICS system groups for distinct purposes.
- Resource topology maps the relation between a CICS system and the resources that are installed in that system.

Central to CICSPlex SM's SSI model are the concepts of context and scope. Every request for data and every action performed on a resource in a CICSplex has these two attributes:

- Context identifies the CICSplex in which the resource exists.
- Scope identifies the CICS system or group of systems within that CICSplex to which the request must be directed.

Figure 5-4 shows how a CICS system can be part of more than one scope.

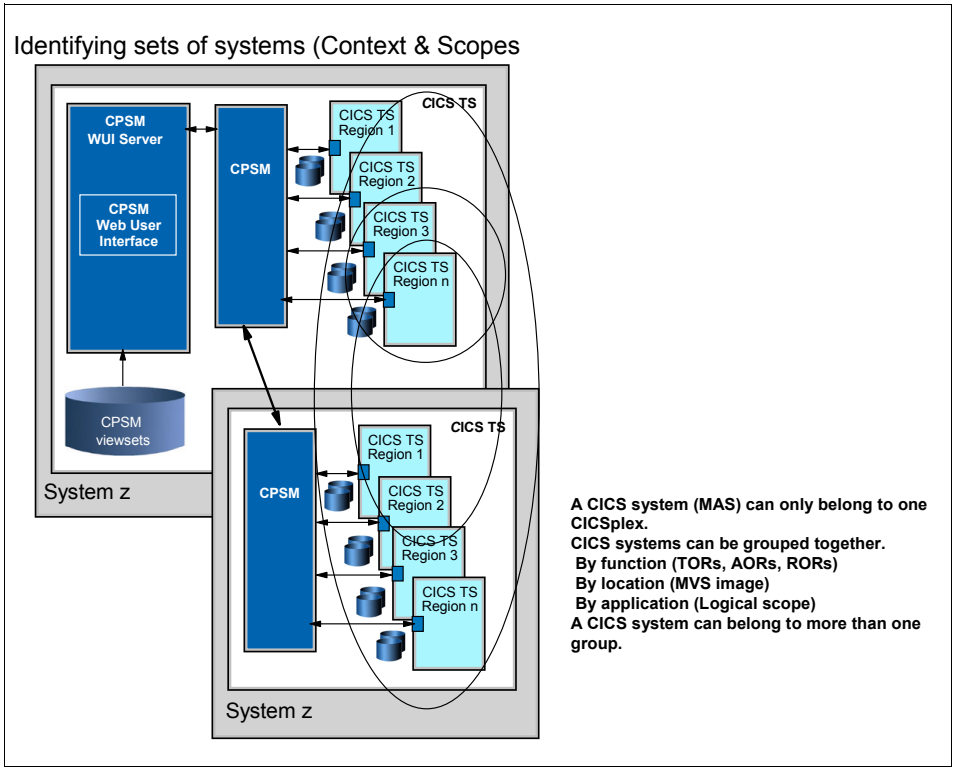


Figure 5-4 Identifying sets of systems

Valid scopes include the entire CICSplex, a user defined CICS system group containing a set of related CICS systems, and an individual CICS system. Logical scopes that group resources by application rather than by their physical location are discussed later, in 5.7.2, “CICS resources” on page 189.

The context and scope of a request identifies a specific set of CICS systems to which the request must be sent. This is further limited by checking resource topology to identify only those systems in which the targeted set of resources exist. Thus a request to display all files defined as local files (LOCFILE resource table) is shipped to every MAS in the target scope, but a request for LOCFILE resource tables whose file name matches the mask “PAY*” is only sent to that subset of systems in the scope in which a file matching the mask is installed.

Key attributes for installed resources: Resource topology maps contain the key attributes for each installed resource. If a query criteria expression contains a term defining an equal comparison against a key attribute, the scope list is restricted to systems in which the desired resources are defined. If the criteria expression does not contain terms comparing key attributes, or if one or more terms comparing key attributes use a comparison operator other than equal (=) then it will be necessary to send the request to all MASes in the requested scope. Therefore queries which request resources filtered by key attributes will be more efficient than those that do not. The key attributes for each resource table are identified in *CICS Transaction Server for z/OS Version 4 Release 1 CICSPlex SM Resource Tables Reference,* SC34-7032.

After region topology and resource topology generate a list of target regions, the network topology map is queried to identify the shortest network path to the CMAS which manages each MAS in the request.

5.5.1 CICSPlex SM resource tables

Information about resources managed by CICSPlex SM is externalized in the form of resources tables. Each resource table contains a number of attributes. In its most basic form, a resource table is a record map with each attribute represented by a field in the record. The resource table definition includes information of the versions of CICSPlex SM for which specific attributes are available, allowed values or ranges for specific attributes, help text which is displayed by the WUI, and supported actions and action parameters for the resource.

There are four types of resources that are managed by CICSPlex SM:

- Operations resources contain state information for installed CICS resources and for internal objects like DSAs and REQIDs.

- Monitoring resources contain performance related data for CICS resources.
- Administrative resources are definitional objects stored in a CMAS' data repository. Many administrative resources define the objects, like CICS systems, that make up a CICSplex. Others represent objects which define the operation of a CICSPlex SM component like real-time analysis (RTA) or workload management (WLM). These must be installed to be effective. Another set defines CICS resources and the way that Business Application Services (BAS) deploy CICS resources in CICS systems managed by a CICSplex.
- CMAS resources can be considered operations resources containing state information about a CMAS or CICSplex. This includes active CMASes, the relationship between CMASes and CICSplexes, and the state of active components like RTA or WLM.

CICS Transaction Server for z/OS Version 4 Release 1 CICSPlex SM Resource Tables Reference, SC34-7032 contains descriptions of all exposed resource tables.

5.5.2 Operations

CICSPlex SM currently recognizes 190 operational resource types. Many of these correspond directly to installed CICS resources. Others, for example LOCFILE and REMFILE (local and remote files) or EXTRATDQ, INTRATDQ, INDTDQ, and REMTDQ (extrapartition, intrapartition, indirect, and remote transient data queues), are split because each of the individual CICSPlex SM resource tables require a separate set of attributes. Still others, like LSRPOOL and LSRPBUF (LSR pools and buffers) or LIBRARY and LIBDSN (library and data set-in-library) reflect a one-to-many relationship between the primary and secondary resource tables. Many, such as CICSRGN (CICS region) or CICSDSA (CICS DSA), contain information describing the internal state of the CICS system.

The CICSPlex SM WUI starter set contains views to display all of the operational resource tables. Each view set, named EYUSTART`tablename` where `tablename` is the name of the resource table that the view set displays, contains at least one tabular view, and one or more detailed views. A tabular view contains one line for each resource. Each line contains a selection of attributes that identify the instance of the resource and provide high-level information about its state. Figure 5-5 on page 182 shows the WUI EYUSTARTCICSRGN.TABULAR view.

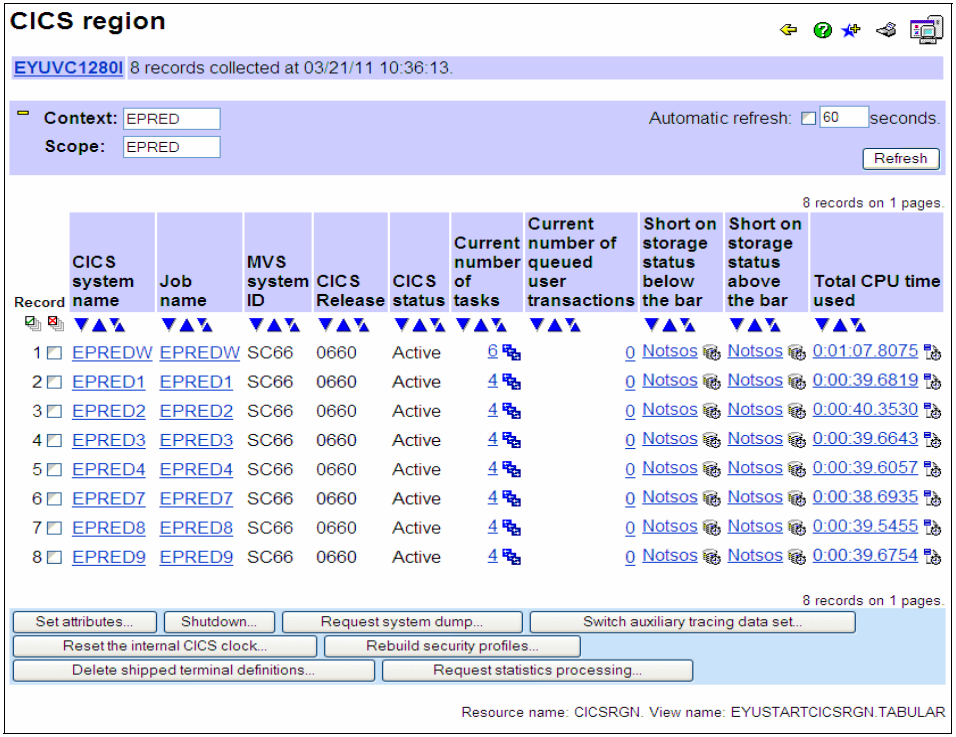


Figure 5-5 CICSPlex SM WUI tabular view

Clicking one of the highlighted fields in a WUI view executes a hyperlink to a detailed view that provides more information about a resource. A detailed view contains more detailed information about a single resource instance.

Operators at both The Bank and The Store use CICS Explorer to manage resources in the CICSplex. Their decision to use Explorer instead of the WUI was based on the fact that the tabbed interface of CICS Explorer allows an operator to manage a number of critical resources without having to navigate between views. Figure 5-6 on page 183 shows the CICS Explorer CICS SM perspective displaying the Regions tabular view and detailed information about CICS region EPRED1.

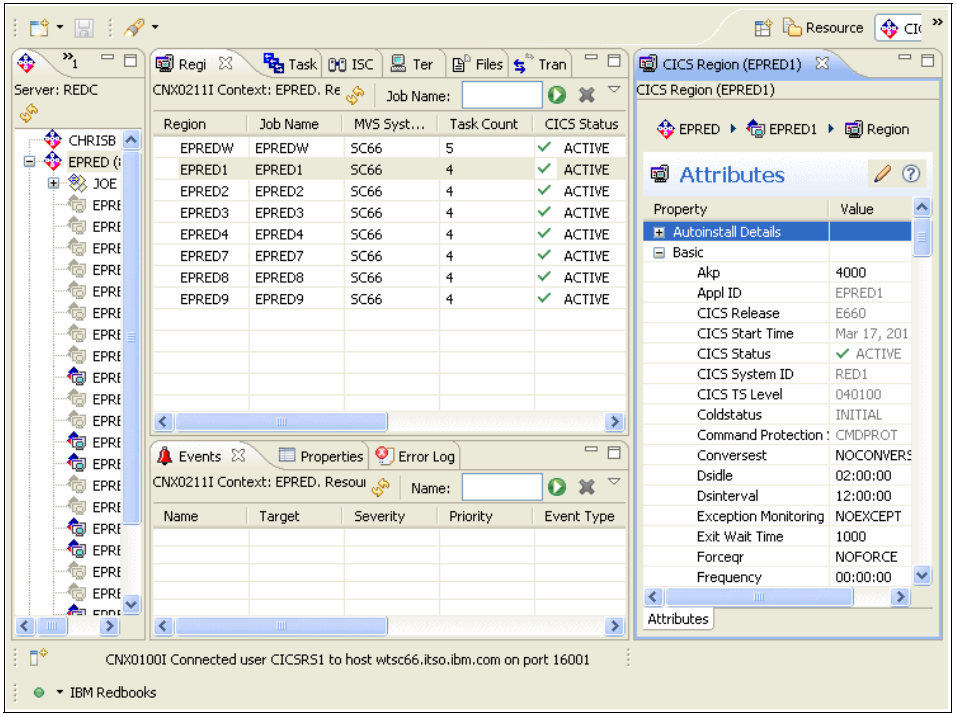


Figure 5-6 CICS Explorer CICS SM perspective

Although CICS Explorer does not provide views for all CICSPlex SM resource tables, using the CICS Explorer SDK users can extend Explorer to display resources not included in the default set. See the IBM Redbooks publication *Extend the CICS Explorer*, SG24-7819.

5.5.3 Monitoring

CICSPlex SM provides facilities to monitor resource usage and performance in a CICSplex. CICSPlex SM monitoring uses data collected by CICS monitoring and statistics, and retrieved by CICS inquire commands to track the performance of individual resources. Although a full set of views are provided in the WUI starter set to display CICSPlex SM monitoring data, the primary purpose of monitoring allows performance exceptions to be detected by real-time analysis (RTA) exception monitoring. For example The Bank uses RTA to monitor the average response time during the current sample interval (CURAVGRESP) for the transactions which support their ATMs. An operator is notified if transaction response time exceeds a predefined threshold. If CICSPlex SM monitoring support is required by RTA, the necessary monitoring definitions are created and installed dynamically.

For control purposes, these resources are grouped into several monitor classes:

- CICS systems
- Connections
- DB2 threads
- Files and data tables
- CICS journals
- Global resources
- Programs
- Terminals
- Transactions
- Transient data queues

Monitor data is accumulated for two time intervals:

- The sample interval, set for each class of monitor data in the monitoring specification (MONSPEC) definition, is the time in seconds between the collection of successive samples of monitoring data for each class. A sample interval of zero prevents collection of data for a monitor class.
- The monitor interval is the time in minutes during which monitoring data is collected and accumulated. At the end of each monitor interval all accumulators are reset to zero. The monitor interval is set in the definition for a CICSplex, and is the same for all regions and monitor classes.

CICSPlex SM enforces the requirement that an integral number of monitor intervals occur in a day. That is, the monitor interval must divide evenly into 1440. Thus a monitor interval of 750 minutes is not allowed. When monitor data collection starts for a MAS, the first monitor interval is shortened if necessary to ensure that the last interval of the day ends at midnight. After that all monitor intervals are the same length.

The collection of task execution history (HTASK) data is also controlled through CICSPlex SM monitoring. When monitoring is activated for the transaction class, the user can request that a task history record be captured every time a CICS task ends.

5.6 Single point-of-control

Along with being able to collect resource state and performance data for all CICS systems in a CICSplex, CICSPlex SM allows resources throughout the CICSplex to be managed from a single point-of-control. In general two types of operations are supported for both CSPM and CICS resources. Set allows the values of selected attributes to be modified in real-time. Actions allow specific functions (for example, discarding a CICSPlex SM RTA definition or activating a new copy of an

installed program) to be performed on one or more resources. For many CICS resources, an action (for example, ENABLE) might also be available through a set operation (for example ENABLESTATUS=ENABLED). These operations can be requested by a CICSPlex SM API program or through the WUI or CICS Explorer.

5.6.1 CICSPlex SM resources

If you change an installed CICSPlex SM definition which controls monitoring, RTA, or workload management (WLM), the change is not effective until the active resource is discarded and the changed definition is reinstalled. In addition, installed monitor and RTA definitions that are associated with a time period definition (PERIODEF) that causes them to be active only during the specified time period can be deactivated. After being deactivated, a definition again becomes active at the normal activation time.

The MAS resource table support actions to turn RTA and CICSPlex SM monitoring on and off for a managed CICS system or activate WLM routing support, if the necessary supporting definitions exist. The MAS agent can be stopped without bringing down the CICS system. However because stopping the MAS agent implicitly discards the connection between the MAS and the controlling CMAS, the MAS agent cannot be restarted from a CICSPlex SM API program, or WUI or CICS Explorer session. A CICS operator must enter the COLM transaction to restart the MAS agent.

Tip: CICS Deployment Assistant provides the ability to restart CICS systems or restart the MAS agent from the CICS DA perspective in a CICS Explorer session.

Both the MAS and the CMAS resource tables support actions to manage CICSPlex SM tracing and to take a system dump containing all data necessary to diagnose a problem in a managed CICSplex.

5.6.2 CICS resources

All of the functions that are available to a terminal operator using the CEMT transaction can be performed by a CICSPlex SM API program or from a WUI or CICS Explorer session. From the context and scope of the request, CICSPlex SM determines to which CICS systems the request must be sent, and whether each system in the scope can execute the requested operation. If the operation fails in one or more CICS systems, detailed feedback is returned to the invoking program. The WUI and CICS Explorer analyze this feedback data to provide meaningful exception messages to the operator. Figure 5-7 on page 186 and

Figure 5-8 show how an operator requests a New Copy action in CICS Explorer, and the action confirmation for the New Copy action against an active program with error feedback displayed.

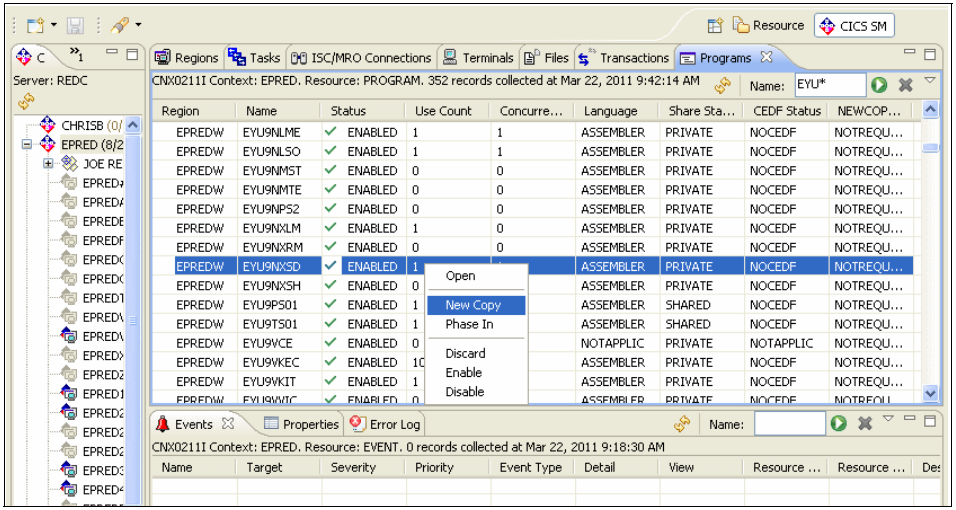


Figure 5-7 Requesting New Copy action for a program

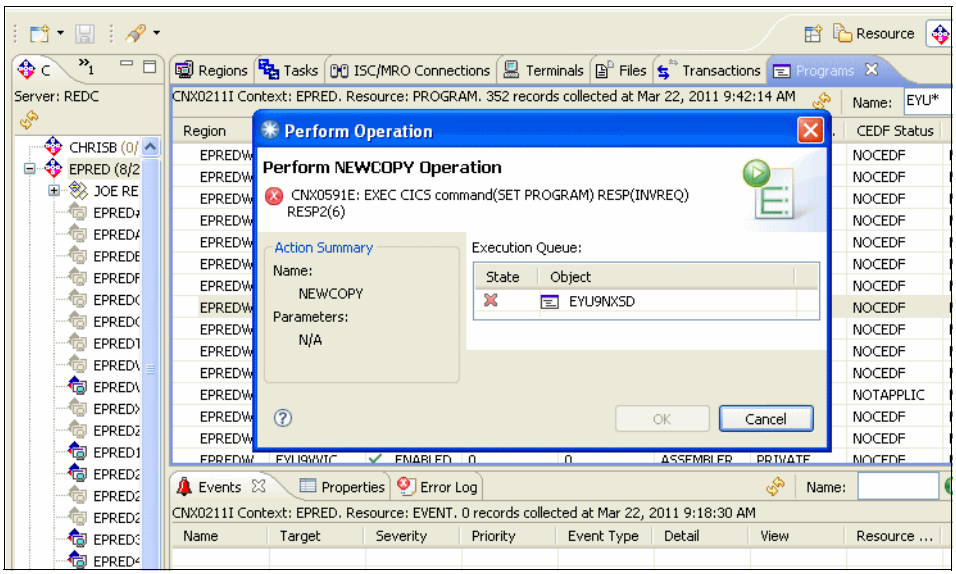


Figure 5-8 Feedback for a failed New Copy action

User written programs that invoke CICSPlex SM API commands can run as a CICS task or in a batch, TSO, or NetView environment. For example, a CICSPlex

SM API program might be executed under the control of an automated operations tool to close and disable files prior to the start of a batch window, and to enable and open the files when batch processing has completed.

5.7 Single point-of-definition

CICSPlex SM also provides a single system image for the definition and deployment of resources in a CICSplex. Definitions for CICSPlex SM resources are stored in the CICSPlex SM data repository. CICSPlex SM Business Application Services (BAS) can manage CICS resources that are stored in the CICS CSD. However the full power of BAS to deploy resources is realized when resources required by business applications are migrated to the data repository.

Every CICS and CICSPlex SM resource contains a set of attributes called the definition signature, which contain auditing information that is unrelated to the specific resource. Figure 5-9 shows the definition signature for a CICSPlex SM resource definition.

Definition signature	
Time created	03/22/11 16:17:44
Last modification	03/22/11 16:17:44
Last modification user ID	CICSRS1
Last modification agent	Drepapi
Last modification agent release	0660

Figure 5-9 Definition signature for a CICSPlex SM resource

Definition signature: The definition signature is not shown in figures in this chapter.

5.7.1 CICSPlex SM resources

CICSPlex SM uses resources stored in each CMAS' data repository to define the topology of a CICSplex, and to control the functioning of CICSPlex SM's monitoring, RTA, and WLM components.

Topology resources

The following resources define the topology of the CICSplex environment:

CPLEXDEF The primary definition for a CICSplex.

CMTCMDEF CMAS-to-CMAS connection definition resources define the network topology of the CICSplex. Each CMAS's data repository must contain a CMTCMDEF resource for every CMAS to which it connects.

CSYSDEF CICS system definitions define the characteristics of CICS systems which will be managed by a CICSplex.

CSYSGRP CICS system group definitions allow a subset of CICS systems in a CICSplex to be managed as a single entity.

Functional resources

Control of the functional components of CICSPlex SM uses three levels of resources to manage the availability of services. Additional resource types exist, which are specific to one component. These are mentioned here and described in more detail in following sections.

The definition (MONDEF, RTADEF, WLMDEF) defines the characteristics of a single functional object. A MONDEF defines the resources to be included (or excluded) from monitoring for a single monitor class. An RTADEF associates an evaluation (EVALDEF), action (ACTION), and a reporting interval to enable detection of a user-defined exception. A WLMDEF associates a target scope (a CICS system group containing available target regions) with a group of dynamic transactions (TRANGRP).

One or more definitions with common characteristics can be linked to a group (MONGROUP, RTAGROUP, WLMGROUP). For example, RTADEFs might be associated with one of three RTAGROUPs. One group is installed in and reports exceptions particular to WLM routing regions, the second is installed in and reports exceptions particular to WLM target regions, and the third is installed in and reports exceptions which are relevant to both types of regions.

Specifications (MONSPEC, RTASPEC, WLMSPEC) cause a component to be activated in a managed CICS system during initialization. One or more groups can be linked to a specification to cause the associated definitions to be installed automatically, or the specification can be empty, allowing groups or definitions to be installed manually as required. In addition, a specification can provide defaults for the component. For example, the WLMSPEC provides the default target scope and routing characteristics which are used to route dynamic transactions, which are not mapped by an active TRANGRP.

Time period definitions

Time period definitions (PERIODEFs) are used with CICSPlex SM to define periods of time. A PERIODEF might be associated with one or more CICS systems to allow CICSPlex SM to monitor system availability. An operator can be notified if a system is down during an active period. They can also be used to turn RTA and monitoring services on and off at specific times of day.

5.7.2 CICS resources

CICSPlex SM Business Application Services (BAS) manages resources required by business applications. BAS managed resource definitions can be stored in CMAS' data repositories or (CICS Transaction Server V4.1) in the CICS CSD.

Resources stored in the CICSPlex SM data repository

Native BAS manages resource definitions that are stored in the data repository. Two modes of operation are supported, basic mode and full-function mode.

Basic mode

BAS emulates the behavior of CICS resource definition online (RDO) with resource descriptions (RESDESCs) corresponding to RDO group lists, resource groups (RESGROUPs) corresponding to RDO groups, and parallel resource definitions. However, even in basic mode, BAS has several advantages over RDO:

- RDO resource definitions can only exist as part of a group. BAS resource definitions and RESGROUPs are independent objects that are linked by xxxINGRP resources (where xxx is a three character representation of the resource type). This has several advantages:
  - Reuse of BAS resource definitions is easier because adding a resource to a new group does not create a second instance of the resource definition.
  - If a definition must be modified, the changes need only be made in one place.
  - Finding a resource definition does not require knowledge of the group or groups in which it is defined.
- BAS validates resources when they are added to an install set to ensure that the new definition does not conflict with definitions already in the set. If resource definitions with conflicting attributes are installed from the CSD, resources installed later in the list replace earlier ones. If definitions have conflicting attributes, unexpected results might be observed.
- BAS SYSLINK resources allow a single connection and session definition (CONNDEF and SESSDEF) to be used as prototypes to define connections between any pair of managed CICS systems. Similarly an IP connection

definition and a TCP/IP service definition (IPCONDEF and TCPDEF) can be used as prototypes for IPIC connections between many pairs of systems. Attributes required to customize the resources being installed in each target system are contained in CSYSDEF resources.

- Up to 15 versions of a BAS resource definition can coexist in a CICSplex. If a application modification requires a change to a resource definition, and new version of the definition can be created. The new version can be deployed in the development environment and installed instead of the old version as the change is migrated through successive levels. When the change is deployed in the production environment, the old resource version can be retained for back out or can be discarded freeing the old version number for reuse.
- Logical scopes allow installed resources to be viewed and managed in terms of their participation in an application rather than their physical location in the CICSplex. A logical scope can be created when resources for a business application are deployed through a RESDESC. Figure 5-10 shows how a logical scope is created containing all resources deployed under a resource description.

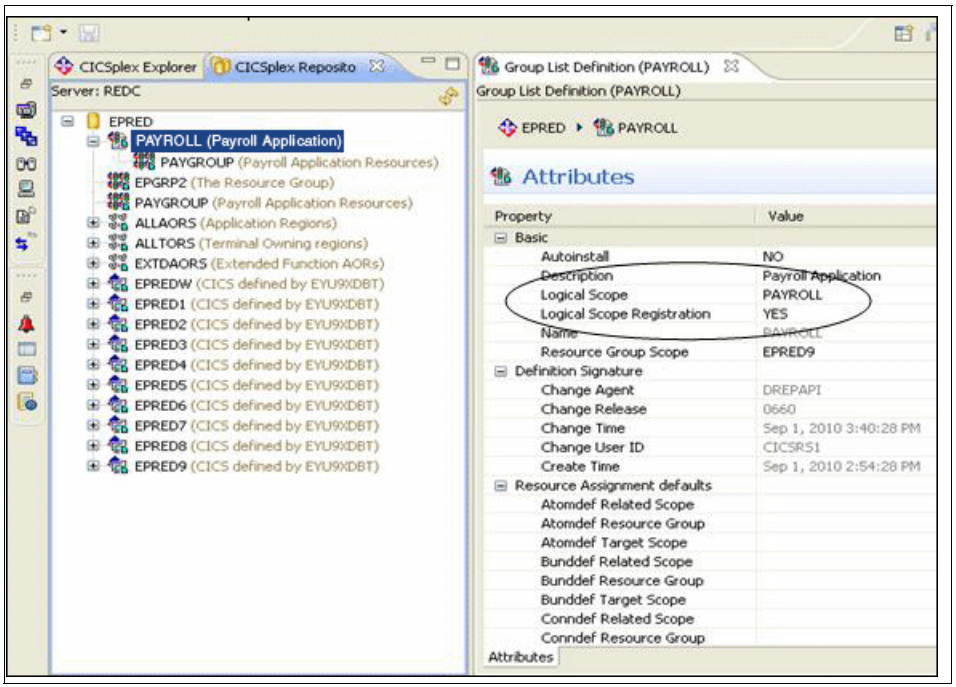


Figure 5-10 Use of logical scope

Full function mode

The full power of BAS is realized by adding a new object type, the resource assignment definition (RASGNDEF). Figure 5-11 shows how resource assignment definitions can expand the ability to deploy resources throughout a CICSplex.

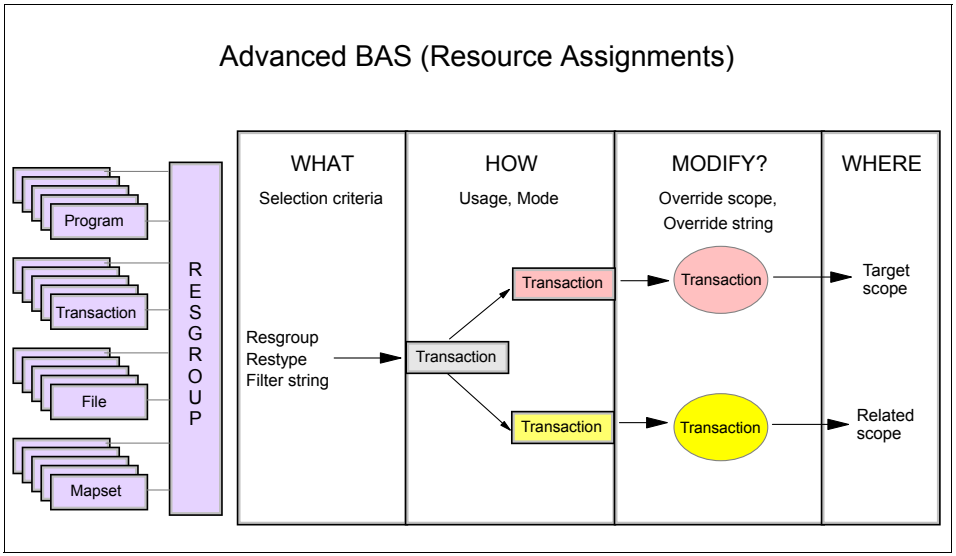


Figure 5-11 Full function BAS

With RASGNDEFs, you can deploy a single resource definition in multiple targets with differing attributes. For resource types that support remote and local installation (for example, transaction, programs, or files), the same resource definition can be installed as a remote resource in a group of systems (the target scope) and as a local resource in a single system or another group of systems (the related scope). RASGNDEFs can also modify selected resource attributes at install time. Thus a single file definition can be deployed in development, quality assurance, and production environments under the control of resource assignment definitions that alter the data set name attribute appropriately for each environment.

Resources stored in the CICS CSD

In CICS, systems that run CICS TS V4.1 and higher, CSD resources can also be defined and managed using the CICSPlex SM API, CMCI, and CICS Explorer. The same resource tables manage BAS and RDO instances of CICS resources. A new attribute, CSDGROUP, distinguishes resources that reside in the data repository and in the CSD. For RDO resources, the CSDGROUP attribute contains the name of the group in which the resources is defined. For resources

in the data repository, the CSDGROUP attribute must be blank. A new CSD parameter is also required for certain actions to identify the source of the resource against which the action is to be performed. There are several restrictions when managing RDO resources using BAS:

- Resource definitions cannot be migrated between the CSD and the data repository with CICS Explorer.
- RDO resources can only be accessed for a scope which is a single CICS system.

Migrating resource definitions

CICSPlex SM provides a utility program, EYU9BCSD, which runs as a user program driven by the DFHCSDUP EXTRACT command. Output of EYU9BCSD is a batched repository update facility (BATCHREP) input file that contains selected definitions from the source CSD. This file can be passed as input to BATCHREP to create the definitions in the CICSPlex SM data repository.

5.7.3 Customer scenarios

In this section, we discuss scenarios of The Bank and The Store.

The Bank

The Bank used BAS to manage resource definition and installation since they began using CICSPlex SM. They found that SYSLINKs allowed them to install new connections for cloned CICS regions by adding a single SYSLINK resource rather than by creating new connection and session definitions for each region. They also used resource assignments to manage deployment of dynamic transactions in routing and target regions and to allow single file definitions to be installed in their development, assurance, and production environments with appropriate data set name overrides for each environment.

The Store

The application programs supporting The Store's existing point-of-sale terminal network are written and maintained by the terminal vendor, and resource definitions are provided as CSD upgrades. However The Store is using BAS to manage resources that support the web services provided by their catalog and ordering applications.

5.8 Exception management

The RTA component of CICSPlex SM monitors managed CICS systems for exceptional conditions which impact transaction throughput. The types of notification provided when an exception is detected are defined in ACTION resources that are associated with each exception type. Options available in an ACTION definition include issuing messages to the system console which can be trapped and handled by automated operations products, creating an EVENT resource which can be displayed in the WUI or CICS Explorer or listened for by an API program, sending a generic alert to NetView, and causing the system in which the exception occurred to be restarted by z/OS automatic restart manager (ARM).

5.8.1 System availability monitoring

RTA and system availability monitoring (SAM) monitors the overall health of managed CICS systems. A SAM notification is triggered if a CICS system is associated with a time period (PERIODEF) and the system is down during the active time period (SAMOPS). Other conditions monitored by SAM are MAXTASK (SAMMAX), short on storage (SAMSOS), taking a system dump (SAMSDM), taking a transaction dump (SAMTDM), transaction stall (SAMSTL), and in V4.2 non-responsive MAS (SAMNRM). CICSPlex SM is notified through global user exits when a system is short on storage or taking a system or transaction dump. However CICS provides no notification when a system is at MAXTASK, and stall detection is a statistical process, so SAMMAX and SAMSTL conditions are detected by the MAS heartbeat task, which runs every 15 seconds. Thus a system might be at MAXTASK for most of the 15 second heartbeat interval, but if it is not a MAXTASK when the heartbeat task is posted to run, SAM does not detect or report the condition. The non-responsive MAS condition is actually detected in the managing CMAS. A MAS is considered to be non-responsive if it fails to return requests sent from the managing CMAS in a timely manner. Care must be taken to understand that SAMNRM does not actually report a condition detected in the MAS, but rather the fact that the actual state of the MAS is uncertain because it is not responding to requests sent from the CMAS.

5.8.2 MAS resource monitoring and analysis point monitoring

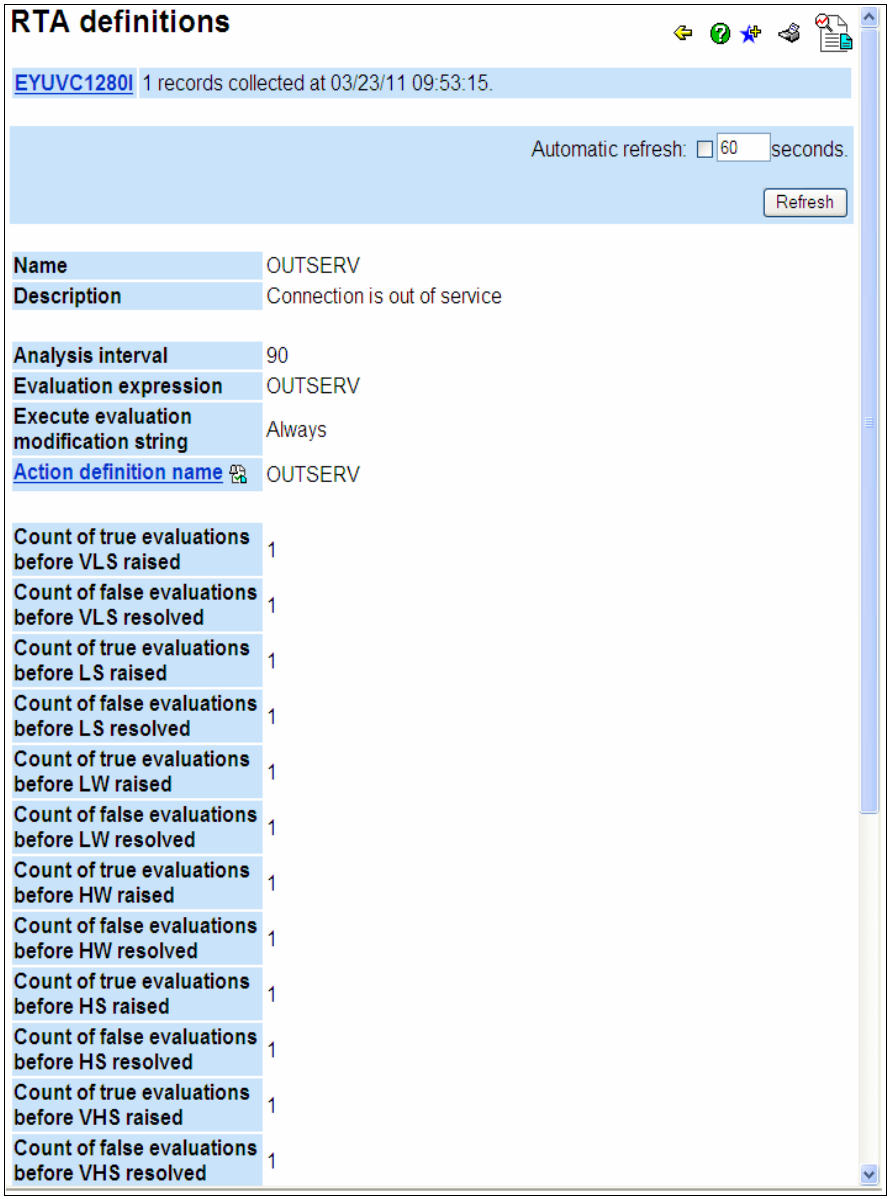
RTA/MRM and RTA/APM both detect and report user-defined exceptional conditions. MRM and analysis point management (APM) use the same functional definitions to identify the conditions to be monitored, the evaluation interval (how often data is collected for evaluation), and the analysis interval (how often the results of the evaluation are analyzed).

The main difference between MRM and APM is the scope of the notifications issued when an exception is detected. MRM issues a notification for every MAS in which an exception is detected. APM issues a single notification when an exception is detected anywhere in the CICSplex. An analysis point specification (APSPEC) is associated with a primary and secondary CMAS. The primary CMAS is responsible for analysis and notification for definitions installed under the APSPEC. If the primary CMAS is not active, the analysis function is switched to the secondary CMAS. APM is useful for exceptions that affect more than one CICS system. For example, reporting RLS data sets, which are quiesced outside the batch maintenance window, might be done under an APSPEC so a single notification is issued instead of a notification for every CICS system that shares access to the data set.

Conditions that can be detected by examination of attributes in a single resource table can be detected by user written RTA definitions (RTADEFs). An RTADEF relates an evaluation definition (EVALDEF) that describes a condition to be monitored and an action definition (ACTION) that describe how CICSPlex SM responds when the condition becomes true.

The RTADEF controls how often the results of the evaluation are analyzed, whether the modification string on the EVALDEF is to be executed when the RTADEF state becomes true and how many consecutive true or false evaluations must be processed before the RTADEF changes state. These counters, called entry and exit clocks, can prevent spurious notifications for transient conditions.

Figure 5-12 on page 195 shows the RTADEF used by The Bank to control the detection of MRO connections that are out of service and identify the actions to be taken when the RTADEF state becomes true.



RTA definitions

EYUVC1280I 1 records collected at 03/23/11 09:53:15.

Automatic refresh: 60 seconds.

Refresh

Name	OUTSERV
Description	Connection is out of service
Analysis interval	90
Evaluation expression	OUTSERV
Execute evaluation modification string	Always
Action definition name	OUTSERV
Count of true evaluations before VLS raised	1
Count of false evaluations before VLS resolved	1
Count of true evaluations before LS raised	1
Count of false evaluations before LS resolved	1
Count of true evaluations before LW raised	1
Count of false evaluations before LW resolved	1
Count of true evaluations before HW raised	1
Count of false evaluations before HW resolved	1
Count of true evaluations before HS raised	1
Count of false evaluations before HS resolved	1
Count of true evaluations before VHS raised	1
Count of false evaluations before VHS resolved	1

Figure 5-12 Sample RTA definition (RTADEF)

The EVALDEF, shown in Figure 5-13 on page 196, tests for MRO connections that are not in service and attempts to set the SERVSTATUS attribute to INSERVICE. All CONNECT resources in MASes in which this EVALDEF is installed are sampled every 30 seconds.

Evaluation definitions

EYUVC1280I 1 records collected at 03/30/11 09:27:27.

Automatic refresh: 60 seconds.

Refresh

Name	OUTSERV
Description	Connection out of service
Sample interval	30
Resource table	CONNECT
Instance identifier of evaluated resource	*
Method of evaluating results in result set	Any
Separate task indicator	No
Field being evaluated	SERVSTATUS
Evaluation type	Value
Value evaluation type parameters	
Evaluation logical operator	Ne
Evaluation data value	INSERVICE
Severity assigned when result meets criteria	Hs
Threshold evaluation type parameters	
Upper bound of range for VLS	
Upper bound of range for LS	
Upper bound of range for LW	
Lower bound of range for HW	
Lower bound of range for HS	
Lower bound of range for VHS	
View that may provide extra information	CONNECT
Filter string	TYPE='MRO'.
Modification string	SERVSTATUS='INSERVICE'.
Get parameters	

Figure 5-13 Sample evaluation definition (EVALDEF)

Figure 5-14 on page 197 shows the ACTION, which identifies the external notifications to be generated when the RTADEF becomes true.

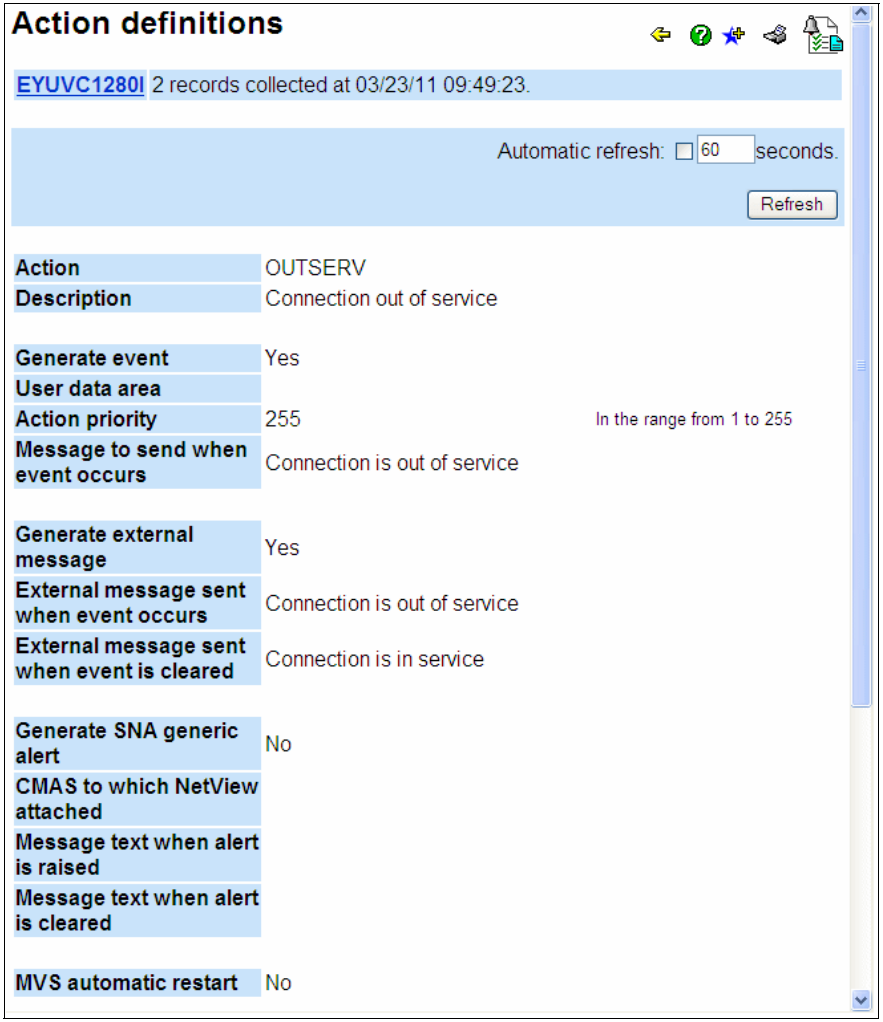


Figure 5-14 Sample action definition (ACTION)

The Store defined several transactions in a transaction class to limit the number of instances that can be active at any given time. They use RTA to monitor the active count for the transaction class, and issue a notification if the active count exceeds the limit for the transaction class. The have set the number of true evaluations before a notification occurs to two, to prevent a single transient spike from triggering a notification.

To detect exceptions, which are more complex to evaluate, CICSPlex SM can schedule execution of a status probe. Status probes are special programs that are passed a COMMAREA in which they return flags that RTA uses to set the

state of a status probe definition (STATDEF). Figure 5-15 shows a STATDEF, which causes program SPECIAL to be called as a status probe. The probe normally executes under transaction COIE, with the authority of the default user ID. To change the priority or to restrict access to resources, another user ID and transaction can be specified in the STATDEF. The entry and exit clock values have the same purpose and effect as in the RTADEF.

Status probe definitions

EYUVC1280I 1 records collected at 03/23/11 11:04:04.

Automatic refresh: 60 seconds.

Refresh

Name	SPECIAL
Description	Special status probe definition
Last modification	03/23/11 11:03:38
Status program name	STATPROB
Interval between calls to status program (seconds)	60
Action definition name	SPECIAL
User ID for task	CICSRS1
Transaction ID for task	SPEC
Evaluation count with VLS true before event	1
Evaluation count with VLS false before resolution	1
Evaluation count with LS true before event	1
Evaluation count with LS false before resolution	1
Evaluation count with LW true before event	1
Evaluation count with LW false before resolution	1
Evaluation count with HW true before event	1
Evaluation count with HW false before resolution	1
Evaluation count with HS true before event	1
Evaluation count with HS false before resolution	1
Evaluation count with VHS true before event	1

Figure 5-15 Sample status probe definition

5.8.3 Customer scenarios

Both The Bank and The Store found RTA to be a convenient way to monitor the state of the CICS regions in their respective CICSplexes. Their operators use CICS Explorer to display events raised by the SAM, MRM, and APM components of RTA. Figure 5-16 shows the Events view in the CICS SM perspective of CICS Explorer. The operator can see RTA events in the Events view and examine Operations resources to identify and repair exceptions.

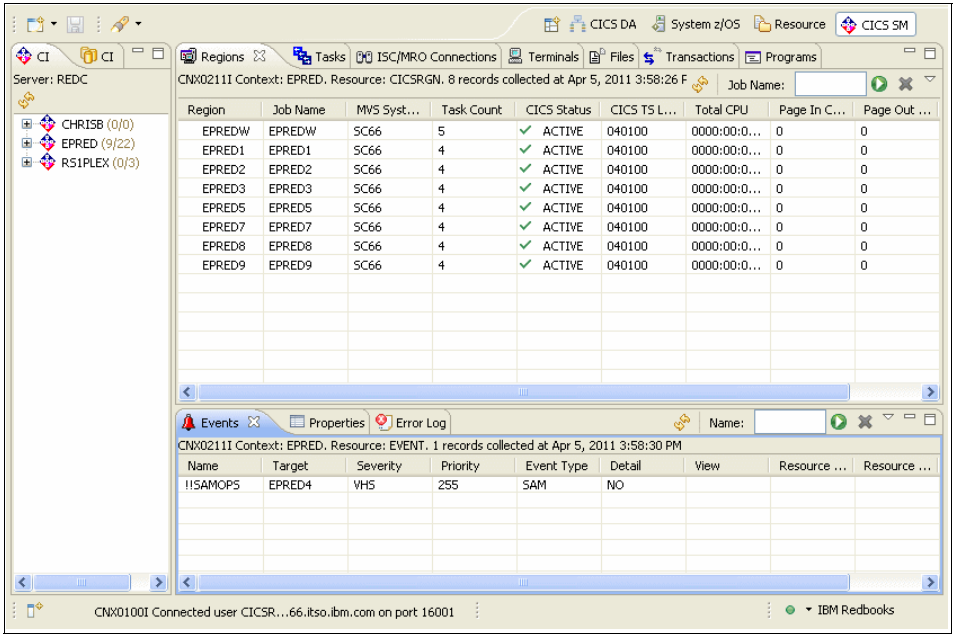


Figure 5-16 Events view in CICS Explorer

In this case, a CICS region is inactive. With CICS Deployment Assistant, the operator can access the CICS DA perspective to examine the system log and the job log from the last execution of the region to determine why it shut down and to restart the region from within CICS Explorer.

5.9 Workload management

CICSPlex SM provides a program, EYU9XLOP, which can be called by CICS as a dynamic routing program (DTRPGM) or a distributed routing program (DSRTPGM). EYU9XLOP can also be called by an application program. In each case, information is provided in a COMMAREA that identifies the source of the

request and the type of service that is being requested. CICSPlex SM compares the information provided by the caller with installed workload definitions to identify a group of one or more target regions to which the work might be routed (the target scope). A weight is calculated for each of the CICS systems in the target scope, and the SYSID of the target region considered best able to process the incoming work is returned to the caller in the COMMAREA.

5.9.1 Workload manager services: What we do for you

CICSPlex SM WLM provides three services to an enterprise:

- Workload balancing: For a given unit of work, CICSPlex SM selects the target region expected to provide the best response time.
- Workload affinity: A transaction affinity results when state information is stored in a target region between successive transactions. CICSPlex SM will honor requirements to route successive transactions to the same target until a defined event causes the affinity to end. (See *CICS Transaction Server for z/OS Version 4 Release 1 CICSPlex SM Managing Workloads,* SC34-7008 for information about supported affinity types and lifetimes.)

  CICS Interdependency Analyzer: You can use CICS Interdependency Analyzer to identify specific conditions that might result in transaction affinities.

- Workload separation: CICSPlex SM routes transactions within target scopes selected based on transaction ID, operator's user ID, terminal luname, or BTS process ID.

The most basic WLM configuration provides workload balancing for all dynamic transactions and program links across CICS systems in the target scope, and optionally honor a default affinity type and lifetime. This simplest form of workload management requires three things:

- A WLMSPEC must be associated with each CICS system in which EYU9XLOP will be called as dynamic routing program or distributed routing program. The WLMSPEC identifies the default target scope for the workload and can also establish a default affinity type and lifetime for the workload. These defaults are used to route all dynamic transactions and program links not otherwise associated with an installed TRANGRP.
- The CSYSDEF for each system must specify that Workload manager status for the CICS system is YES.

- The system connects to a CMAS to allow workload management to initialize. However, after WLM initializes, transaction routing continues even if the CMAS is brought down for a time.

To make use of CICSPlex SM's ability to perform workload separation, several more resources must be defined. Starting at the lowest level:

TRANGRP The transaction group identifies the characteristics of a set of dynamic transactions. This includes an affinity type and lifetime, which are honored for transactions associated with the TRANGRP, parameters to control whether CICSPlex SM attempts to avoid systems where transactions in the TRANGRP are observed to abend, and in version 4.2, the type of algorithm that is used to select a target region. The TRANGRP resource table also contains the name of an RTADEF whose state is included in the weight calculation for AORs in the target scope.

WLMDEF The WLM definition identifies a target scope, the TRANGRP whose transactions are routed to this scope, and selection parameters to identify the source for which this scope applies, if transactions are being routed to a particular target scope based on the operator's user ID, the terminal luname or the BTS process for the transaction.

WLMGROUP An envelop object that is associated with one or more WLMDEFs that are to be installed together. A WLMGROUP can be associated with a WLMSPEC, in which case all child WLMDEFs are installed at workload initialization. Otherwise, the WLMGROUP can be installed manually at any time after a workload initializes in a CICSplex.

5.9.2 Workload algorithms: How we find the best

Prior to version 4.2, CICSPlex SM provided two algorithms for calculating a target weight:

- The queue algorithm considers existing transaction load (expressed as a fraction of MAXTASK for the system), system health, and the observed probability that the incoming transaction abends in the target region. If an RTADEF is identified in the EVENTNAME attribute of the TRANGRP to which the transaction belongs, the current severity of the RTADEF will also be considered. Finally the derived weight is adjusted for the type of connection between the routing and target region. This provides a bias in favor of sending

work over faster connections, expecting better response times as a result. The connection types, from most to least preferred. are:

  - Local (the work executes in the routing region)
  - Cross memory MRO
  - Cross system (XCF) MRO
  - VTAM
  - TCP/IP
  - Indirect

- The goal algorithm calculates a weight based on the ability of target regions to meet average or percentile response-time goals defined to the z/OS workload manager for the transaction's service class. If two or more regions are found to be equally capable of meeting the response time goals, the queue algorithm is applied to the set of target regions that are projected to meet service goals to select a final target.

One algorithm was used to calculate target weights for all work managed by a workload. The desired algorithm is identified by the MODE attribute of the WLM specification.

Three changes were made to WLM in version 4.2:

- A routing algorithm can be specified for each TRANGRP in a workload. This allows routing to be optimized for groups of transactions with specific characteristics.
- Link weight factors were adjusted to favor target regions connected to the routing regions by IP connections over target regions connected by VTAM LU6.2 connections.
- Two new link-neutral algorithms, LNQUEUE and LNGOAL are supported. These function in exactly the same manner as the link-aware algorithms except that the type of connection is not considered in calculating a target's weight.

Link neutrality: Why link-neutrality? Standard routing algorithms favor target regions which execute in the same LPAR as the routing region. If the routing regions feeding a workload are not distributed evenly across all the LPARs that support target regions, incoming work will not be routed evenly to all available target regions. This does not normally cause performance issues, as the importance of the link weight factors decreases as the load on a target regions increases. However there is a perceived problem that all target regions are not being fully utilized.

5.9.3 Sysplex optimized workload: How the best got even better

In a pre-version 4.1 workload, a serious problem existed if more than one routing region fed work to targets that were not connected to the same CMAS as the routers. The state of each target region active in a workload is maintained in structures in the WLM data space of each CMAS that takes part in a workload. If all target regions are connected to the same CMAS as the routing regions that feed them, and if no work enters a target region except through a CICSPlex SM managed workload, state information, and especially load level, is updated in real time as transactions are routed and terminate.

The problem

If a target region receives work from a routing region that connects to another CMAS, or if work enters the target from other sources, the load statistics saved in the WLM data space can become increasingly inaccurate. Every 15 seconds, though, a heartbeat task running in every MAS sends a status update to its managing CMAS, which is then broadcast to all CMASes in the CICSplex. This status update includes the real current load for the region. When the load stored in the WLM data space is updated, WLM can suddenly discover that the region was at or beyond MAXTASK because all of the work entering was not accounted for. This causes WLM to stop routing work to the overloaded system and start routing to another system which had previously been less favorable. Fifteen seconds later, another status update is broadcast, and the same switch occurs as the new target is found to be overloaded. The result, called *surging*, is that work is routed to a target region, driving it to MAXTASK, and shifting from target to target at 15 second intervals.

The solution

The solution, introduced in CICS Transaction Server V4.1, is Sysplex Optimized Workload Management. Mediated by the new CICS region status (RS) domain, the status of each target region is posted to a Region Status Server, where it is placed in a Coupling Facility Data Table. Each routing region has access to near real-time transaction load and region health information for all target regions in the MVS sysplex.

The grouping of target regions into two load ranges occurs because CICSPlex SM takes into account expected transmission times when determining a target region for a transaction. At low to moderate transaction loads (based on percent of MAXTASK) CICSPlex SM prefers target regions on the same LPAR as the routing region because the expected transmission time is lower. As transaction load is increased, the effect of this bias decreases. If the workload is driven to the point that all targets are near MAXTASK, the bias is eliminated altogether, and all target regions show similar transaction loads. The link-neutral weighting

algorithms described in section 5.9.2, “Workload algorithms: How we find the best” on page 201 were introduced in V4.2 as a solution to this problem.

5.9.4 Customer scenario summaries

The Bank

The Bank is anxious to complete the conversion to CICS TS V4.1 because they observed problems balancing their workload during periods of peak load. Workload surging, as CICSPlex SM adjusts the transaction load in target regions in response to status updates received from the MAS heartbeat task, results in degraded response when their incoming load is highest. As soon they complete the upgrade of all their CICS systems to a release that supports the region status domain, they intend to implement sysplex optimized workload balancing, which eliminates the surging problem.

Although a plan to upgrade to CICS TS V4.2 is still in the future, they are already considering whether they can receive benefit from the new link-neutral routing algorithms. They hope to improve the ability for incoming work in either TOR to be routed to the AOR best able to handle the load, regardless of the type of connection between the TOR and AOR.

The Store

The Store began using WLM to balance the distribution of transactions across their target AORs. Most of their business transactions require an active DB2 connection. To ensure that transactions are not routed to an AOR if the DB2 connection is inactive, their system programmer wrote an RTADEF, EVALDEF, and ACTION to monitor the DB2 connection status in their AORs. By setting the EVENTNAME attribute in the TRANGRP, which identifies their DB2 workload to this RTADEF name, WLM will consider the state of the DB2 connection in calculating the weight for each AOR in the target scope

6

CICS Explorer

In this chapter, we explain what you must know about the CICS Explorer and how the CICS Explorer can be used to benefit organizations.

The CICS Explorer is a product download that you can obtain from:
http://www-01.ibm.com/software/htp/cics/explorer/

6.1 CICS Explorer

The CICS Explorer is the smart new modern interface to CICS. It has a common windows look and feel that makes it easy for anyone to learn and use. It provides a framework of navigators and views backed by a rich CICS data model that is fed by CICSPlex System Manager (CICS SM).

The CICS Explorer also provides an integration point for CICS tooling (plug-ins) each adding their own special value. The CICS Explorer has a common, intuitive, Eclipse-based environment for architects, developers, administrators, system programmers, and operators to efficiently develop and manage new and existing applications. Using integrated help, tutorials, and a direct connection to the CICS Information Center both experienced and new users will find getting started easy. All the information about the CICS resources and the logic to manage that information being passed to the user is executed in CICSPlex SM.

The Explorer task-oriented views provide integrated access to a broad range of data and control capabilities, and it also has powerful, context-sensitive resource editors. Integration point for CICS TS, CICS Tools, CICS TG, PD Tools, and Rational Tools are extensible by independent software vendors (ISV), system integrators (SI), and customers who use our CICS Explorer Software Development Kit (SDK).

Throughout this chapter the capabilities of the Explorer working with CICS TS are discussed along with the latest versions of the CICS Tooling plug-ins. Some of the plug-ins that we cover are CICS Interdependency Analyzer, CICS Configuration Manager, CICS Performance Analyzer, CICS Deployment Assistant and more. We also discuss several CICS Explorer perspectives, such as the z/OS perspective, which contains a set of views to help you manage the System z artifacts, such as z/OS UNIX Files, data sets, jobs, console, and system messages. The Resource perspective is explained along with how it provides CICS Event Processing, which allows management of real time business and system events.

The CICS Explorer is a great way to accelerate the transfer of knowledge, skills, and best practices to the next generation of technical staff and experts. No matter how big or small your organization, CICS Explorer and System Z can lead the way to platform simplification. Figure 6-1 on page 207 provides a quick look at the new face to CICS, The CICS Explorer.

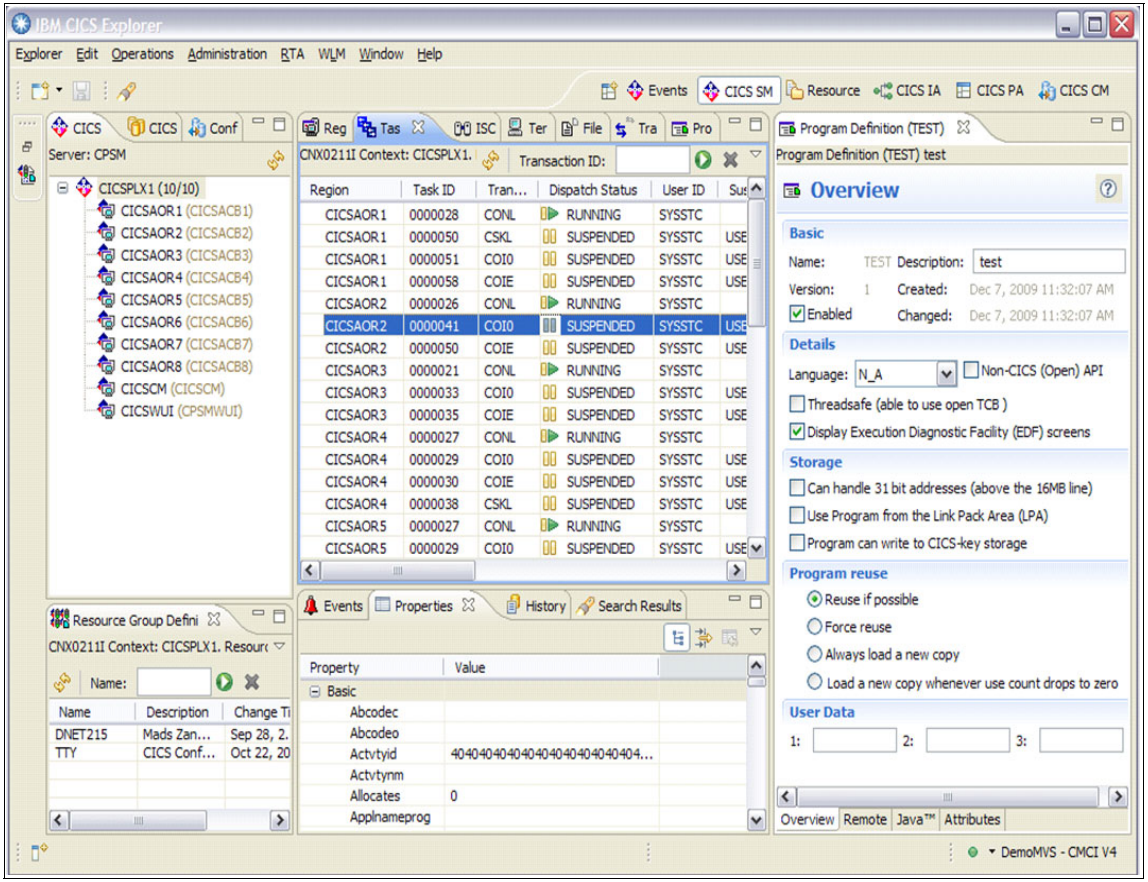


Figure 6-1 Sample CICS Explorer window

6.1.1 The Eclipsed-based environment

Eclipse is a platform for building and deploying client applications in which the majority of data manipulation is done by the client application rather than the server. This platform is known as a Rich Client Platform (RCP). The platform provides the ability to deploy native graphical user interface (GUI) applications to a variety of desktop operating systems, and present the user with a simple, consistent view.

One of the features of Eclipse that makes it a good choice of technology for the CICS Explorer is its plug-in architecture, which allows different components to be added that can extend and complement each other. A number of plug-ins exist that can be integrated into Eclipse environments, such as workflow, messaging clients, or development tools. One of these products is IBM Rational Developer

for Z (RDz), which is an integrated development environment for the System z platform that allows the creation of programs that access the CICS API to perform transaction processing. By both products sharing Eclipse as their common base technology, the CICS Explorer integrates within IBM Rational Developer for System z version (version 7.5 or higher) to provide system administrative capabilities to CICS application developers.

For more information about Eclipse, refer to:

- http://en.wikipedia.org/wiki/Eclipse_%28software%29
- http://eclipse.org/

6.2 Why CICS Explorer was developed

As a newcomer into a CICS environment or as an experienced user moving roles, CICS System Management was a daunting challenge. Prior to the CICS Explorer, to perform each task required the uses of several tools all with separate disconnected interfaces. These applications were often CICS 3270 transactions, web browsers, TSO ISPF-based panels, and batch jobs. When you consider that there is also a whole host of third-party vendor or customer home-grown CICS systems management tools, each with their own user interface and implementation, you can see why CICS system administrators and programmers tend to regard with suspicion yet another tool with yet another user interface to learn.

If the completion of a specific task crossed application boundaries, you had to switch back and forth between the applications, each having its own commands and user interface. This all changed with the development of the CICS Explorer. Now you can perform most all CICS System Management tasks from one common integration point, the CICS Explorer.

While CICS is modernizing the operational and management experience, Rational continues to simplify and modernize System z development. Rational Developer for System z (or RDz) now includes an integrated copy of the CICS Explorer, which allows you to query, define, and install needed CICS resources for programs developed using RDz. The combination of CICS Explorer and RDz simplifies the CICS development process by removing or automating manual steps. The developer no longer must return to the green panel to define many of the CICS application resources they need for testing.

As well as basic CICS System Management, the definition and operation of regions and their installed resources, a typical CICS shop also attracted an extensive third-party ecosystem of application providers, independent software vendors, systems integrators, and independent consultants. They kept pace with

the changing face of CICS, and give CICS customers the applications, tools, and support they need, to benefit quickly from the latest releases of CICS.

The CICS Explorer is not just for IBM use. By integrating third-party applications into this strategic CICS interface, the CICS ecosystem can provide even better support for, and exploitation of CICS. So, to support this belief, we are today making available a CICS Explorer Software Development Kit (SDK). This SDK allows you to develop and deploy your own plug-ins to use and extend the perspectives, views, rich data, and methods provided by the CICS Explorer.

In summary, providing an integration platform so that all of the tools and applications that you need to perform a business task work together coherently in an integrated environment was one of the driving forces behind the creation of the CICS Explorer. The benefit of CICS Explorer is to help new users become familiar with the platform, to present a single and re-usable way of accessing and manipulating CICS to promote consistency, and to allow tools that are built by IBM, vendors, and customers alike to be able to extend and enhance in a unified way. Each tool must introduce its own value to CICS management without having to reinvent or redefine the interface to the base platform itself. The CICS Explorer achieved this by embracing the Eclipse open source community's extensible development platform, runtimes, and application frameworks for building, deploying, and managing software.

6.2.1 The CICS Explorer API Software Development Kit

The Software Development Kit (SDK) provides a collection of Java objects that give a programmatic interface for creating, updating, and performing actions on CICS resources and definitions.

The CICS Explorer can be extended by plug-in providers using the Explorer API / SDK. Using the Explorer SDK customers can develop their own dedicated plug-ins to satisfy requirements particular to them. Using this framework maintains the consistency of the interface and integrates with the rest of the functionality that is provided in the Explorer and is the mechanism by which IBM and vendors provide plug-ins for their value-add tools. In summary, the SDK:

- Provides separately packaged plug-ins for any Eclipse-based environment including RDz
- API allows third parties to extend or integration existing capability
- Layered architecture allows Eclipse-independent components to be used in any environment

Note: For a more detailed understanding of the CICS Explorer SDK, see IBM Redbooks publication *Extend the CICS Explorer*, SG24-7819.

6.2.2 CICS Explorer is a companion for CPSM web user interface

The simplest way to appreciate the advantages of the CICS Explorer is to compare it with a more traditional web application the CICSPlex SM WUI. They do much the same thing, but most users agree that the CICS Explorer is a far superior implementation.

While the WUI is a web browser (an HTTP client), it is a thin client with almost no client-side programming logic other than Java script to help with the formatting of the HTML data being passed from the CICS WUI region to the browser.

The data sent from CICS to the web browser is in HTML format and is often large, resulting in large amounts of data flowing across the network and slower response times. This is because all of the data relevant to a request from the user is returned in a single response. If you repeat the request, all of the information is resent from CICS to the web browser. When you use the WUI, all communication between the web browser is synchronous. The user must wait for the HTTP response from the last input be returned from CICS before continuing with work.

With the CICS Explorer architecture, there is a lot more happening on the client side than in the WUI model. The client (Eclipse RCP) incorporates logic and data to handle the interface with the user, the state of the user session, and to hold information about the CICS configuration being used. This information extends to a detailed model of the CICSPlex being managed (based on the CICSPlex SM Resource Tables Reference model).

There is logic in CICS Explorer to cache data and manage the session, so CICS Explorer is not obliged to send an HTTP request to the WUI CICS region for each user interaction. Because CICS Explorer is managing the user interface, the data that flows from the WUI CICS region need not be HTML. In fact, CICS Explorer exchanges XML messages with the target WUI CICS region.

Responsibility for holding information about CICS resources is shared between CICS Explorer and CICSPlex SM. In several cases, when you make a request for what might be a large amount of data, the WUI CICS region caches the requested data and passes subsets of the cached data to the CICS Explorer only when they are requested. So, for example, as you scroll through a list of CICS regions, CICS Explorer sends requests for the next subset of data at the moment you need it.

6.3 CICS Explorer perspectives

A perspective is a layout of one or more views in the CICS Explorer workbench. Perspectives define visible action sets, which can be changed to customize a perspective. You can decide how you want to lay out the views in the CICS Explorer and save the layout as a new perspective.

CICS Explorer currently provides two default perspectives: The CICS SM perspective and the System z perspective. Eclipse also provides another perspective called the Resource perspective. In this section, an overview of the three perspectives is given.

Note: For a more details about configuration of the CICS Explorer perspectives, see Explorer Help or the IBM Redbooks publication *Extend the CICS Explorer*, SG24-777819-01.

6.3.1 CICS Explorer System Manager perspective

The CICS Explorer System Manager (CICS SM) perspective is the default perspective in the CICS Explorer and is concerned with viewing and manipulating CICS resources. It allows you to monitor and manage your CICS regions, tasks, files, terminals, and so on. You can enable and disable resources, open and close resources, acquire and release resources, place resources in and out of service, purge tasks, and discard resource definitions.

Whether you are connected to a single CICS region or using a CICSPlex SM configuration, the Explorer CICS SM perspective allows you access to the CICS resources.

For the small-scale user, CICS Explorer can be exploited directly with a single CICS region, providing access to basic operational capabilities and resources definition using the CSD. Small enterprises can therefore benefit from a modern management interface and integration into the CICS tools plug-ins. Exploitation is a matter of minutes, requiring minimal set up.

For those exploiting the extensive capabilities provided by CICSPlex SM, CICS Explorer can access a CICSPlex SM environment. This interface provides the ability to take actions on resources. You can see Active Workloads view, views of router and target regions, plus the ability to activate or quiesce target regions. The Task Associations view and Task Association search displays information about the tasks that make up a distributed transaction and helps to identify tasks that are not responding. Establishing access is again a simple set of actions, even for a first time CICSPlex SM user.

When you open the CICS Explorer for the first time, the CICS SM perspective contains a basic set of views to help you get started. You can add other views to the perspective workbench to display additional information about your CICSplex. There are four sets of views that come with the CICS SM perspective, as shown in Figure 6-2:

- General views
- CICS SM Administration views (like CEDA command)
- CICS SM Operation views (like CEMT command)
- Help views

You can tailor the perspective by adding new views, removing existing views, moving views around or changing the size of views. When you close CICS Explorer, any changes you made to the perspective are saved, and the next time that CICS Explorer starts, your tailored perspective is displayed.

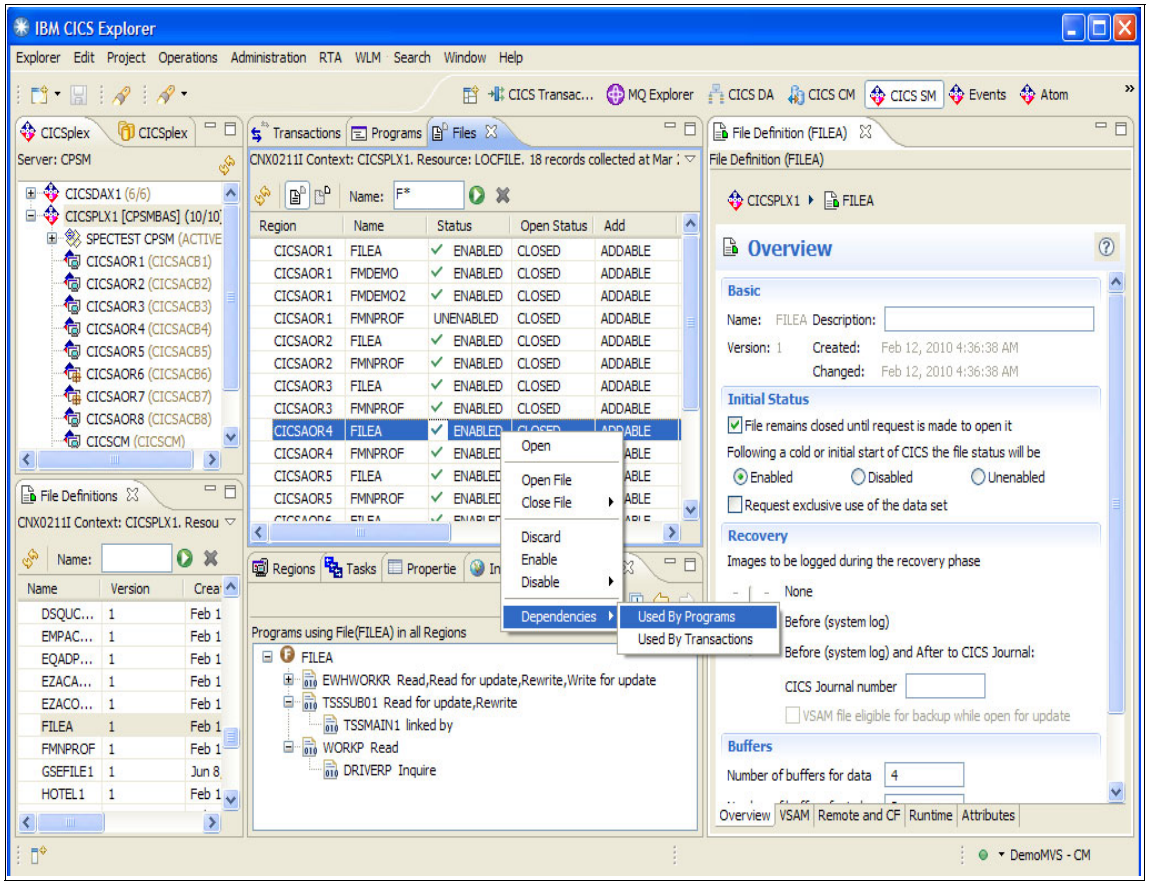


Figure 6-2 View of CICS SM perspective

6.3.2 CICS Explorer System z/OS perspective

The System z/OS perspective lets you manage z/OS data sets, edit and submit jobs, view and edit z/OS UNIX Files, and view a history of the actions performed.

The default System z/OS perspective, shown in Figure 6-3 on page 214, contains a set of views to help you manage your System z artifacts. The default perspective contains the following views:

- The Data Sets view: Lists all the data sets that you are authorized to view and that match the data set name qualifier you specify. You can open a data set to view the members. You can edit Partitioned Data Set (PDS) members that contain text but you cannot edit members that contain binary code.
- The Jobs view: Lists all the completed and running jobs that you are authorized to view and that match the job name prefix and owner ID that you specify. You can expand the job to show the ddnames associated with the spool files, and open the spool files to show the content.
- The z/OS UNIX Files view: Shows the HFS and zFS file system structure and contents when connected to a z/OS system. The files are shown in a tree structure, and you can expand the tree to show individual files. When you connect to a system the z/OS UNIX Files view shows the shows the structure tree for the root directory of the z/OS system to which you are connected. You can expand the branches of the tree to find files, or you can type the path in the text field at the top of the view, for example, /compete/cics/.
- The Console view: Provides a history of the actions you have performed, such as submitting a job, or changing and saving a file. Every time you submit a job or change a file, an entry is recorded in the Console as a history of your actions, and in shown in the screen capture.

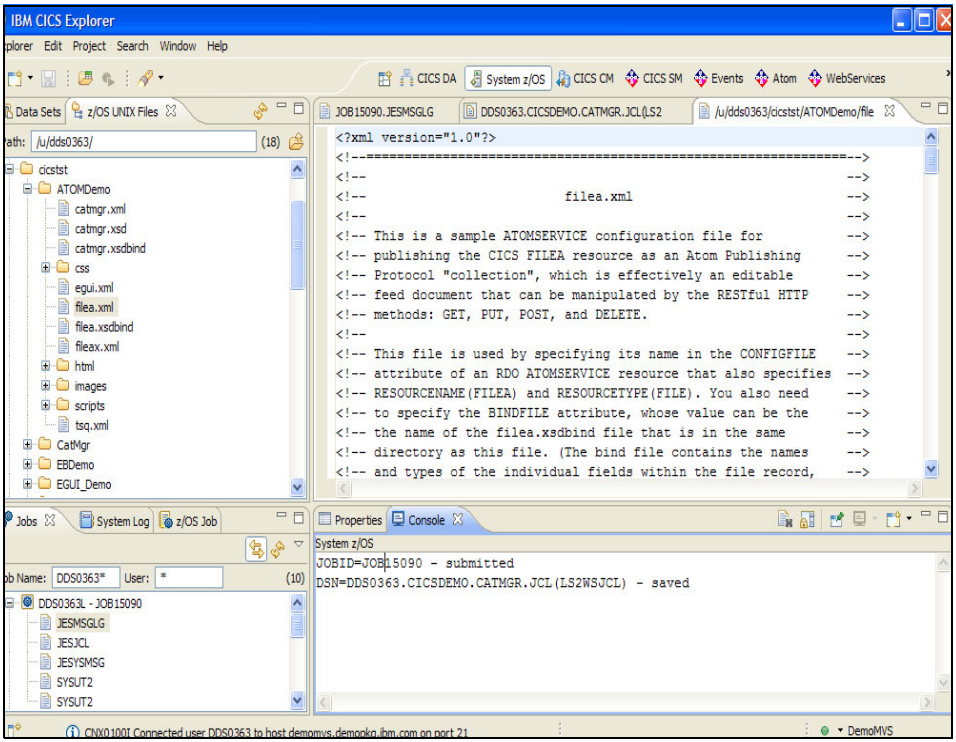


Figure 6-3 View of the System z/OS perspective

The area in the center of the perspective is the editor area. When you edit a file the file contents are displayed in this area.

If you install the CICS Deployment Assistant plug-in (discussed later), the System Log view and the z/OS Jobs view are added to the System z/OS perspective.

You can tailor the perspective by adding new views, removing existing views, moving views around or changing the size of views. When you close CICS Explore, any changes you made to the perspective are saved, and next time CICS Explorer starts, your tailored perspective is displayed. You can restore the perspective to the last saved configuration at any time. You can create and save multiple perspectives. For example, you might want to set up one perspective that only shows the administration views, and one perspective that only shows the operation views.

6.3.3 CICS Resource perspective

The Resource perspective, shown in Figure 6-4, is the standard Eclipse resource perspective. It contains a basic set of views to help you get started with the management of projects and the resources they contain. CICS uses this perspective for CICS Event Processing and System Events.

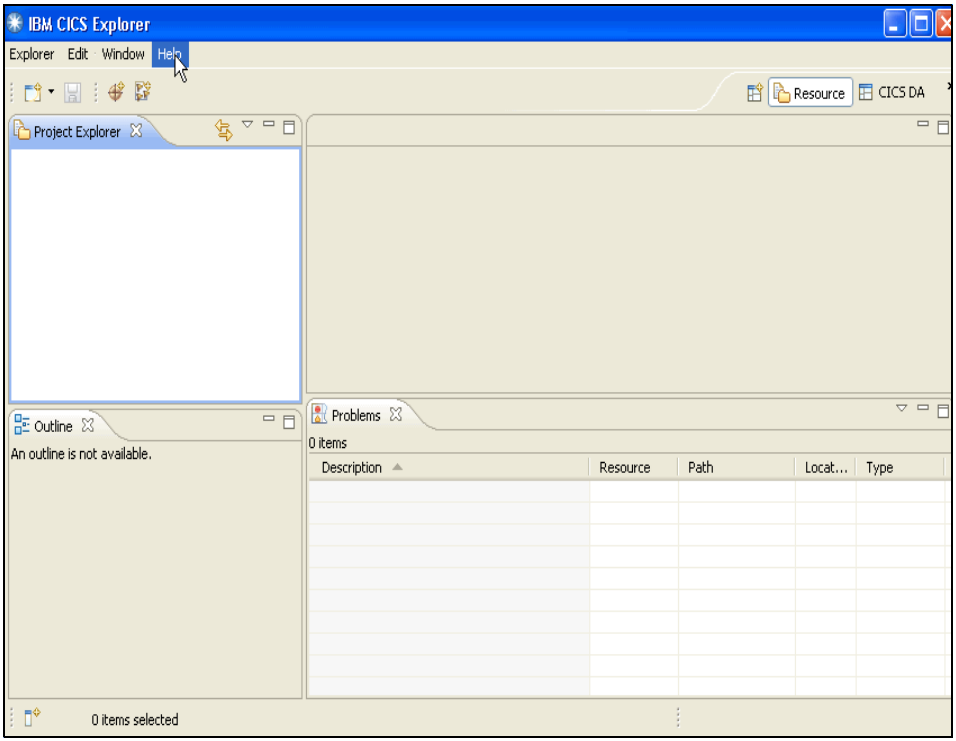


Figure 6-4 The default Resource perspective is shown

6.4 CICS Explorer and Event Processing

In this section, we discuss the following topics about CICS Explorer and Event Processing:

- Introduction to Event Processing
- Why CICS Event Processing and How
- Event Binding Editor

6.4.1 Introduction to Event Processing

What is an *Event*? An Event is anything that happens or does not happen. An event has a name and usually some data (its payload). There are several types of Events:

- Simple Events are single business events, the kind of events that happen during CICS transaction processing, such as order placement, bank account update, stock trade, and shipment arrives. These can be consumed by a *complex event processing* engine where they can be combined with events from other sources in addition to CICS. They can be sent to a Business Monitor to provide insight into processing within CICS. IBM WebSphere Monitor and Business Events (WBE) are consumers of CICS events.
- System Events are a type of business event that results from system activity and contains system data. System events can include resource state changes, thresholds being crossed, or unusual system states or actions. System events help you understand changes in the state of your system resources or system health. You can be alerted to certain CICS system conditions by capturing events for those conditions. Receiving a notification for any changes to the state of system resources avoids the need to poll for changes after they happen. It also means that you can quickly respond to these system events.

  CICS Event Processing supports the following system events:

  - FILE enabled or disable status
  - FILE open or close status
  - DB2CONN connection status
  - TASK threshold
  - TRANCLASS TASK threshold
  - Unhandled transaction abend
- Complex Event Processing detects and responds to patterns of events, such as three orders from a customer in two days. A bank withdrawal after a PIN change update. An interesting pattern of stock trades. A credit card is used at two locations simultaneously. WebSphere Business Events (WBE) is used for detecting and responding to a pattern of events.

- Business Event Processing detects and responds to events that indicate business impacting situations across the enterprise. This extends event processing capabilities to the business users. IBM WebSphere Business Events (WBE) provides complex event processing for business users.

6.4.2 Why CICS Event Processing and How

CICS Event Processing addresses the need for agility in today business world. Modern businesses must react quickly to circumstances. Decision makers need reliable, timely information. CICS systems run an enormous amount of existing business logic, and using an Event-based approach, there is potential to gain insight into the processing in CICS and to introduce additional extensions to applications without the need to change the applications. That is the key, CICS Transaction Server for z/OS allows you to emit business events from existing applications without having to modify the existing applications and send the events to your choice of destinations for analysis, such as WebSphere Business Monitor, WebSphere Business Events and more. A few ways CICS Event Processing can help your business:

- Getting the right information at the correct level of detail to the correct person at the right time.
- Facilitating quick observation of exceptional business behavior and notifying the appropriate people.
- Diagnosing problems based on symptoms and resolving them.
- Providing data for dashboard display of real-time business service availability.
- Start of monitoring the health of your system and changes to it.

Event Processing involves three main aspects, as shown in Figure 6-5 on page 218:

1. *Event producers* emit events into the event processing system, which can be simple RFID sensors and actuators through to business flows or CICS applications. The event processor can carry out a variety of actions, ranging from simple enriching of the event in some way (for example, date/time stamp, add information about the source of the event, producer, and so on), through to comparing multiple events (potentially from multiple sources) against event patterns and producing a new derived event.
2. The event resulting from *processing* is available for consumption.
3. *Event consumers* react to the event. The event consumer can be simple and just update a database or a visual dashboard with the data carried with the event, or it might carry out new business processing as a result of the event. The event consumer can also carry out event processing itself.

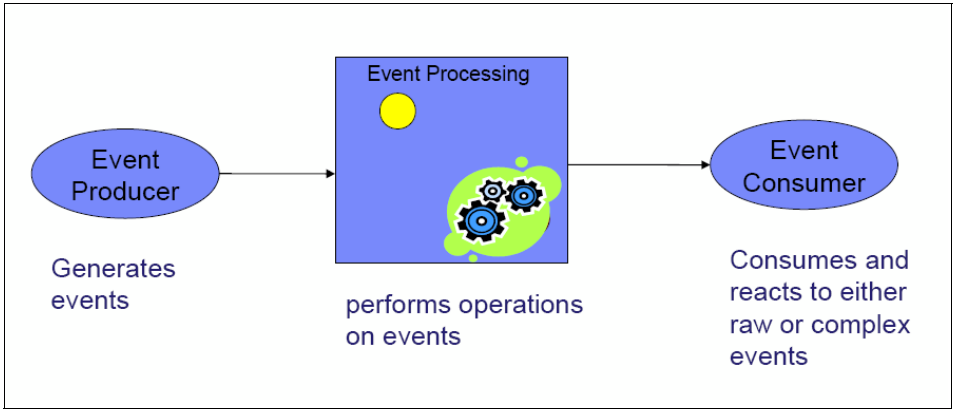


Figure 6-5 Event Processing in a nutshell

Events are valuable to Enterprise Systems, providing the ability to respond in real-time or near real-time. Given the considerable amount of business processing that is carried out in CICS systems across the world (over 30 billion transactions a day), CICS is a significant source of business events. This can provide enhanced business flexibility and the ability to meet governance and compliance regulations.

Event emission can be asynchronous or synchronous to the emitting application, and the consumption of the event is decoupled from its originator.

CICS TS emits simple, single events, which can be consumed by a complex event processing engine where they can be combined with events from other sources in addition to CICS. They can be sent to a Business Monitor to provide insight into processing within CICS.

Figure 6-6 on page 219 illustrates CICS Event Processing with CICS Explorer as part of the tooling.

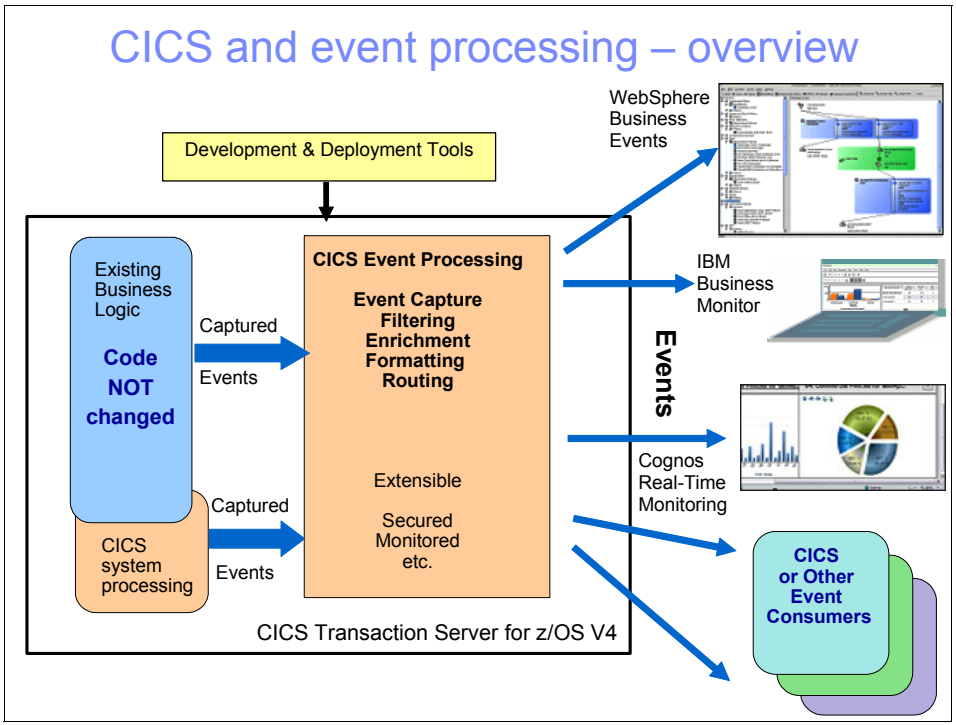


Figure 6-6 CICS Event Processing with CICS Explorer as part of the tooling

This gives a high-level summary of CICS as a source of business events. CICS event processing support allows existing business logic to be instrumented to emit events without change to the application code.

CICS Explorer tooling defines events and their data, to specify to the CICS runtime how to detect when the events occur, to indicate how they are to be formatted and routed, and to deploy the events to CICS.

The CICS runtime detects occurrences of events that are currently enabled, and captures the events without making application code changes. This feature enables rapid, easy deployment of event-based solutions.

CICS Event Processing is a core component of the CICS runtime, and provides all of the qualities of service you expect of CICS. When CICS captures events, it carries out specified filtering, enriches the event with information about the application context in which it occurred, formats the event, and routes it such that it can be consumed by the appropriate event consumer.

It is possible to emit events in formats suitable for consumption by WebSphere Business Events, WebSphere Business Monitor, and other consumers.

6.4.3 Example scenarios using our two customers

This section contains information about our example scenarios.

Catalog sales store

Our catalog sales store identifies key points in their order processing business logic to set event capture points, such as order requested, order placed, order confirmed, order dispatched, and order cancelled. As orders are placed, key order information is Event captured and the relevant contextual data associated with the event, including a way to correlate events for the same order, are sent to the WebSphere Business Monitor where order managers observe orders being received, processed, and cancelled. Order managers can study the number of orders received per week, time to process and dispatch orders, and so on. Managers can take action when thresholds are exceeded, when value of a customer's orders exceeds a certain amount, and so on. They can offer high-dollar amount orders, pricing discounts, manage inventory, and contact customers of canceled orders. At the same time, the original application continues processing independently. The Event instrumentation is non–invasive to the application.

The bank using events

This scenario is an example of "complex" event processing, as shown in Figure 6-7 on page 221, with events being potentially combined from multiple sources including CICS. A complex event processing engine (such as WebSphere Business Events) can collate events from multiple sources and carry out pattern matching to derive additional insight. Our bank firm collect events relating to credit and other bank card usage. These events are passed to WebSphere Business Events where the data and is checked for unusual patterns of behavior. Some patterns of behavior include:

- New card ordered within a week of an address change request
- Several online purchases where none had been made before
- Two or more cash withdrawals in quick succession when withdrawals rare on this card, or normally for smaller amounts
- Purchases in different geographical locations in short period of time

These events can have specific actions to take in WBE, for example: Confirm with cardholder the number of charges and amounts, Confirm with cardholder the address change and new card order, and Suspend Account and Investigate Activity.

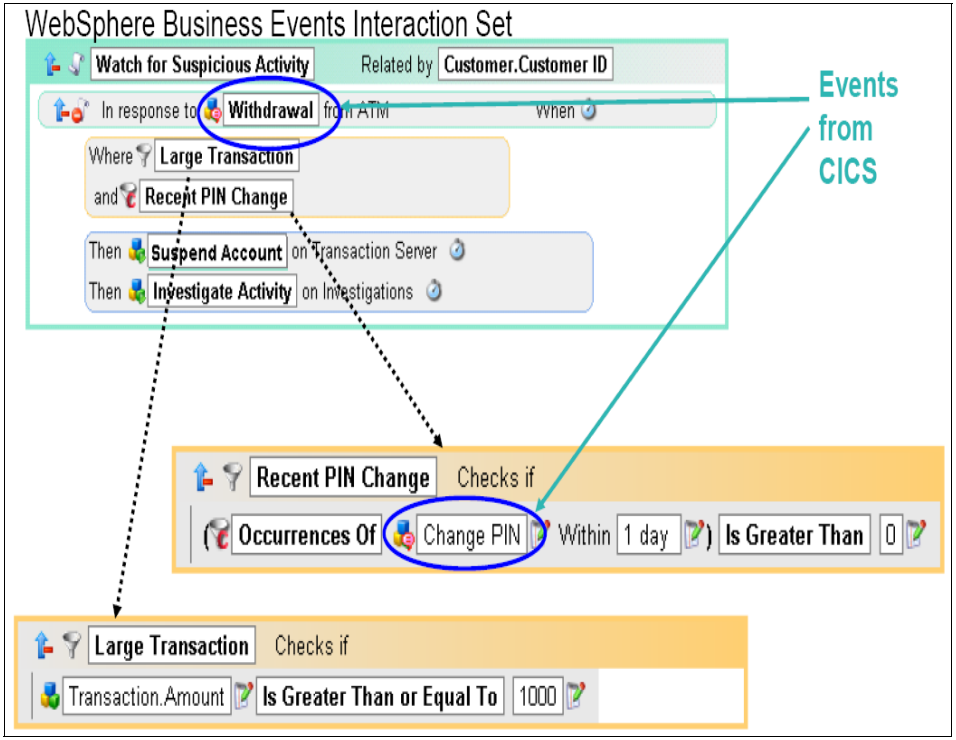


Figure 6-7 Detecting event patterns using CICS and WebSphere Business Events

Events from CICS can be used in interaction sets and conditions (filters) defined in WebSphere Business Events. Figure 6-7 illustrates Event Processing scenario by showing an interaction set and filters defined in the WebSphere Business Events Design tool to watch for a suspicious activity.

System events usage

This is a scenario showing how a company can use system events to monitor the activities of a mature application without having to invest in editing or updating the code. The company wants to be alerted of any unusual activity, for example, system abends, and also wants to monitor general use of the application. The application is used by company employees to view, add, delete, and update insurance policies for life, house, motor, and commercial property insurance.

Currently, if there is a problem with the application or the underlying CICS system, or the company wants to retrieve large amounts of data from the system, the system programmer must investigate and make application changes to retrieve the necessary information. The company wants to see any problems

alerted to them and easily view general statistics about use of the application to understand the sorts of policies their customers use.

The company decides to use a combination of functions provided in the CICS Explorer and CICS System Events to provide the information that they require. System Events are created to alert business management and application analysts of system abends within the application. Events are triggered every time an insurance policy is closed along with the canceling customers contact information allowing management to try and save a customer.

6.4.4 Event Binding Editor

Before we talk about the Event Binding Editor, let us look at Figure 6-8, which illustrates a CICS TS system and how events are processed at a high level.

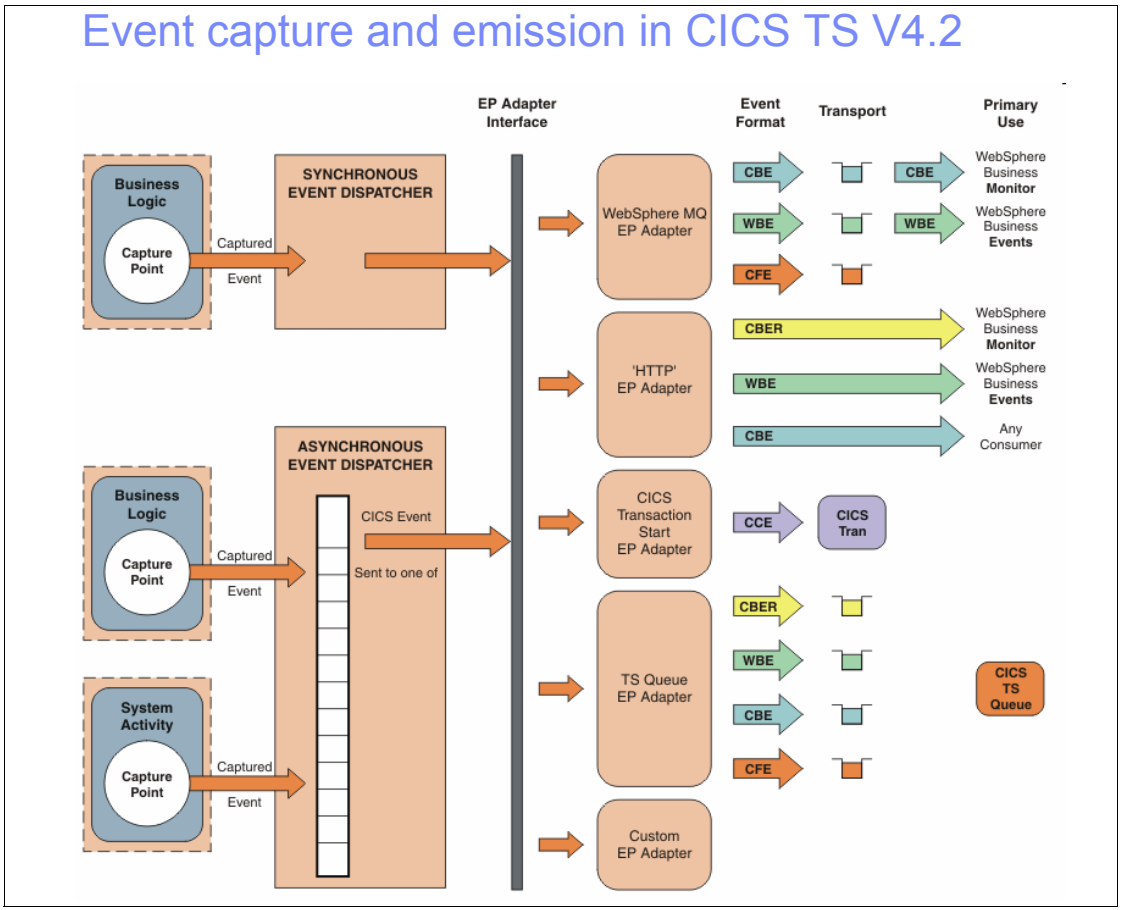


Figure 6-8 Ways in which events are emitted

Figure 6-8 on page 222 shows a CICS TS system with some existing business logic running in it, and it also shows the CICS Event Processing support, which is an integral part of the CICS runtime. The Event Binding Editor (located in CICS Explorer Resource perspective) is used to create and install event bindings, and when the application carries out some processing that matches with an installed capture specification, the event details are captured, enriched with information about the application context in which the event happened, and sent to the Event Processing adapter as specified in the event binding. All Event Processing adapters run within CICS and use a standard interface, and there are a number of adapters supplied with CICS, which can send the event in the required format to its intended consumer.

One option is to write the event to WebSphere MQ, and formats are supported to allow the events to be monitored using WebSphere Business Monitor, included in pattern-matching interactions with WebSphere Business Events, or read from the queue by some other program. An alternative to using the robust WMQ transport is to send events over HTTP, with equivalent formats to allow events to be consumed by WebSphere Business Events, WebSphere Business Events or other consumers of Common Base Event format events. It is also possible for the event to start a CICS transaction with the event data being passed in containers within a CICS channel or written to a CICS temporary storage queue, which is a convenient way to test that the events are captured when intended and with the required data. Because these adapters all use a standard interface, it is also possible to write custom adapters to emit events in any format and to any destination.

The CICS Event Binding Editor is an Eclipse editor used by business analysts to define event specifications and used by application analysts to define event specifications and dispatcher information. The application analyst can use the CICS Explorer to deploy complete event bindings to a CICS system.

The event binding file is an XML file that defines a set of event specifications to CICS. Each event binding file can be separately deployed. The event binding editor consists of several parts, each edited in a separate tab in the editor.

When an event binding file is open in the editor, you can edit the event binding, create, edit, and delete capture specifications, and edit adapter and dispatcher information. When an event binding is complete, you can use the CICS Explorer to deploy an event binding to a CICS system.

The Eclipse-based tooling, part of the new CICS Explorer tooling, which allows you to create event bindings containing event specifications. The screen at the top left shows an event specification. The items listed down the left hand side are Bundles, which are the resource used to deploy events into CICS. One of the bundle projects is shown expanded, and an event binding called OrderingPatterns is selected.

The main part of this screen shot is the Event Binding Editor, open at the page which shows an event specification. So, within the event binding editor, you can see three events which are part of the event binding: Order_Placed, About_to_place and Order_Dispatched. These events are grouped into a single event binding because they are to be sent to the same destination in the same format. The event called Order_Dispatched is selected, and its specification lists the items of business data, the Emitted Business Information, which are to be included in this event when it is emitted. You can just see that there are 3 items of business information for this event, the customer to which the order is dispatched, the item that was ordered, and the order number. This gives the business-level view of the event.

The figure to the bottom right of the slide is the Capture Specification for this event, which is where you indicate to CICS the circumstances in which the event occurs. The capture point, which is not shown on this figure but is under the Capture Point tab at the top, indicates one of the supported EXEC CICS commands, in this example it is actually an EXEC CICS LINK command. What we are looking at here is the Filtering tab, which provides details about which of the LINK commands which are processed in this CICS system relate to the Order_Dispatched event. In this case, the capture specification indicates that if the Transaction ID is EGUI, and the current program that is running is DFH0XVDS and the program that is being linked to is the program which handles dispatch of orders, DFH0XSOD, then we have the event that we are interested in. (Some of you might recognise these programs from the CICS catalog manager example application.)

Although not shown in this example, we can also filter based on data which is passed on the command, such as in the Commarea or Channel that is passed on a LINK command. The final part of the capture specification is the Information Sources, which indicates to the CICS event processing support where to capture the data that is to form the emitted business information; so, for example, the Order_ID can come from a container that is passed in the channel on the LINK from DFH0XVDS to DFH0XSOD.

So, to generate the events, the bundle must be deployed to CICS, which cause the event binding to be installed and enabled. Whenever the filtering conditions are true, CICS will capture the data required and generate an event.

6.5 CICS Explorer plug-ins

In this section, we describe the CICS Explorer Eclipse-based plug-in tools and how they can benefit you organization. The following plug-ins are discussed:

- CICS Performance Analyzer

- CICS Interdependency Analyzer
- CICS Configuration Manager
- CICS Deployment Assistant
- OMEGAMON® XE for CICS
- CICS Transaction Gateway
- CICS Explorer and WebSphere MQ Explorer integration

The ability to create a composite perspective is a powerful feature of the CICS Explorer. You can add the views provided by the CICS TG plug-in for CICS Explorer to other perspectives. You can customize the CICS TG perspective to add other CICS Explorer views.

If you want to install any of the plug-ins, go to the CICS Explorer Download Site at:

http://www.ibm.com/software/cics/explorer/download

To download the item:

1. From the Other Downloads section, select the plug-in link you are interested in, and select download to download the archive file.
2. Unzip the archive file. The contents include a documentation file with detailed installation instructions and another archive file.
3. Follow the installation instructions in the documentation file to install the plug-in.

The documentation also covers connection configuration and how to information.

6.5.1 CICS Explorer Performance Analyzer plug-in

The CICS Explorer Performance Analyzer (CICS PA) plug-in is used by CICS systems programmers, performance analysts, architects, application developers, and IT managers who need a comprehensive CICS system and application performance reporting and analysis solution, and who value its ability to help find new ways to improve CICS system performance, lower maintenance costs, and strategically plan IT investments.

The CICS PA plug-in can access statistics and statistics alert data and performance summary data. The navigators and views in the CICS PA perspective display over 80 statistics classes, grouped in more than 20 categories. In addition, statistics alerts can be viewed in the context of the related statistics and performance data, to highlight potential tuning opportunities or identify trends that can lead to poor CICS performance or unnecessary CICS system outages.

CICS PA plug-in, shown in Figure 6-9 on page 227, provides access to over 180 CICS performance metrics, enabling analysis of CPU, response time, TCB usage for threadsafe, memory, VSAM file, DB2, and WebSphere MQ usage. Contextual linkage is available from CICS Explorer and CICS Configuration Manager resource views to tabular and graphical performance views, and from performance views to relationship data provided by CICS Interdependency Analyzer. Performance data can be accessed using downloaded csv files or by direct access to DB2 data sources. When you open CICS PA for the first time, the following views are displayed:

- The Overview view: Shows all the Applids and Transaction IDs in the selected date range. You can change the date range, select one or more Applids, or one or more Transaction IDs to restrict the analysis. The view is accessible when the CICS PA perspective is active.
- The Tree view: Displays the structure of the database in a tree format, in date, time, applid, and transaction ID order. Using the Tree view you can quickly locate data for a specific date and time. The view is accessible when the CICS PA perspective is active.
- The Project Explorer view: Allows you to view your performance data from a csv file. The most direct method of accessing your performance data is by connecting directly to your PA database on the host, but you can also view the data in a csv file. The csv file must be loaded into a project in your workspace. Projects can contain both folders and files.
- The Sheets view: The resulting data is displayed in the Sheet view. The view tab displays the database name followed by the total number of records in parenthesis. The sheet view is a fixed view that cannot be closed and it always reflects the data selected in the Overview view, the Tree view, or the Project Explorer view.

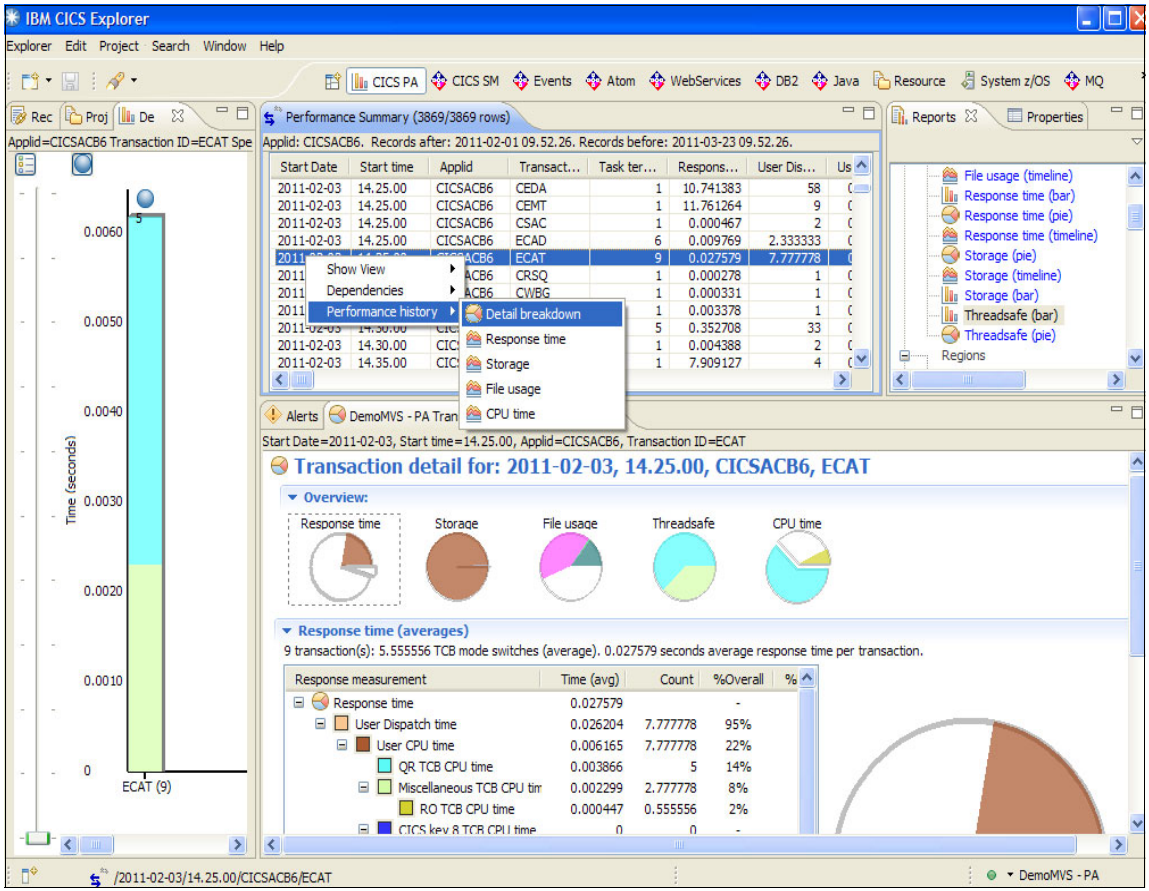


Figure 6-9 View of the CICS Explorer Performance Analyzer plug-in

6.5.2 CICS Explorer Interdependency Analyzer plug-in

The CICS Explorer Interdependency Analyzer (CICS IA) plug-in provides an Eclipse-based infrastructure to identify CICS application resources (IMS, DB2, MQ) and their interdependencies and to analyze transaction affinities. Some capabilities, such as threadsafe and affinity analysis, a new user interface, and support for Software AG Natural, make it possible to achieve better reuse, management, and control of your applications through improved understanding of an even wider range of CICS systems and resources. CICS IA facilitates projects, such as CICS version-to-version migration, affinity removal, and web service refactoring that depend on deep knowledge of application, system, and resource relationships for their success.

CICS IA allows you to understand the makeup of your application set, such as:

- Which transactions use which programs
- Which programs use which resources (files, maps, queues)
- Which resources are no longer used
- What applications does a CICS region contain

When you open CICS IA for the first time, the views in Figure 6-10 are displayed.

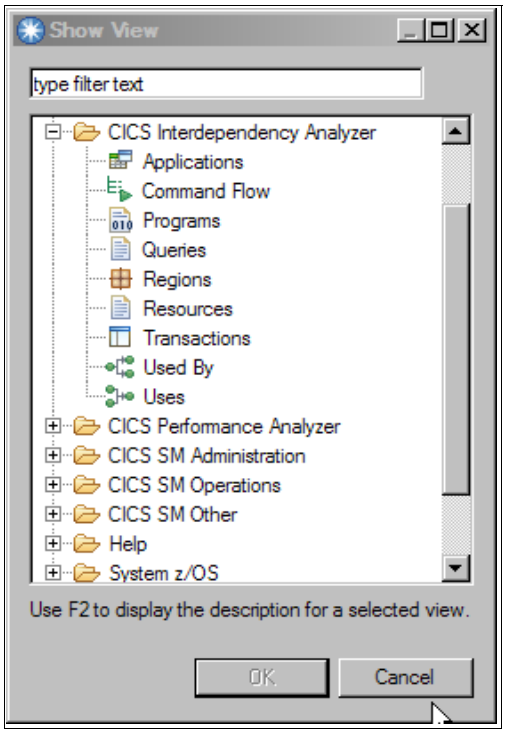


Figure 6-10 CICS IA main screen

The initial views are:

- The Queries view is used to view a list of queries and folders. It allows you to organize and group queries.
- The Programs view shows an alphabetic list of all programs known to CICS IA, such as: those that are the sources of interactions, make calls into CICS, CICS resources of type PROGRAM, and programs that are the results of interactions.
- The Resources view displays resources from a number of sources, including searches run from the toolbar, from the Queries window, and from the Regions window. Resources are organized into a tree structure.

- The Uses view is populated with information from a selected resource or region. From any view where a transaction or program resource is shown in the CICS IA plug-in, you can explore resources further to see which resources the transaction is using and in which regions.

Figure 6-11 provides a view of the CICS Explorer Interdependency Analyzer plug-in.

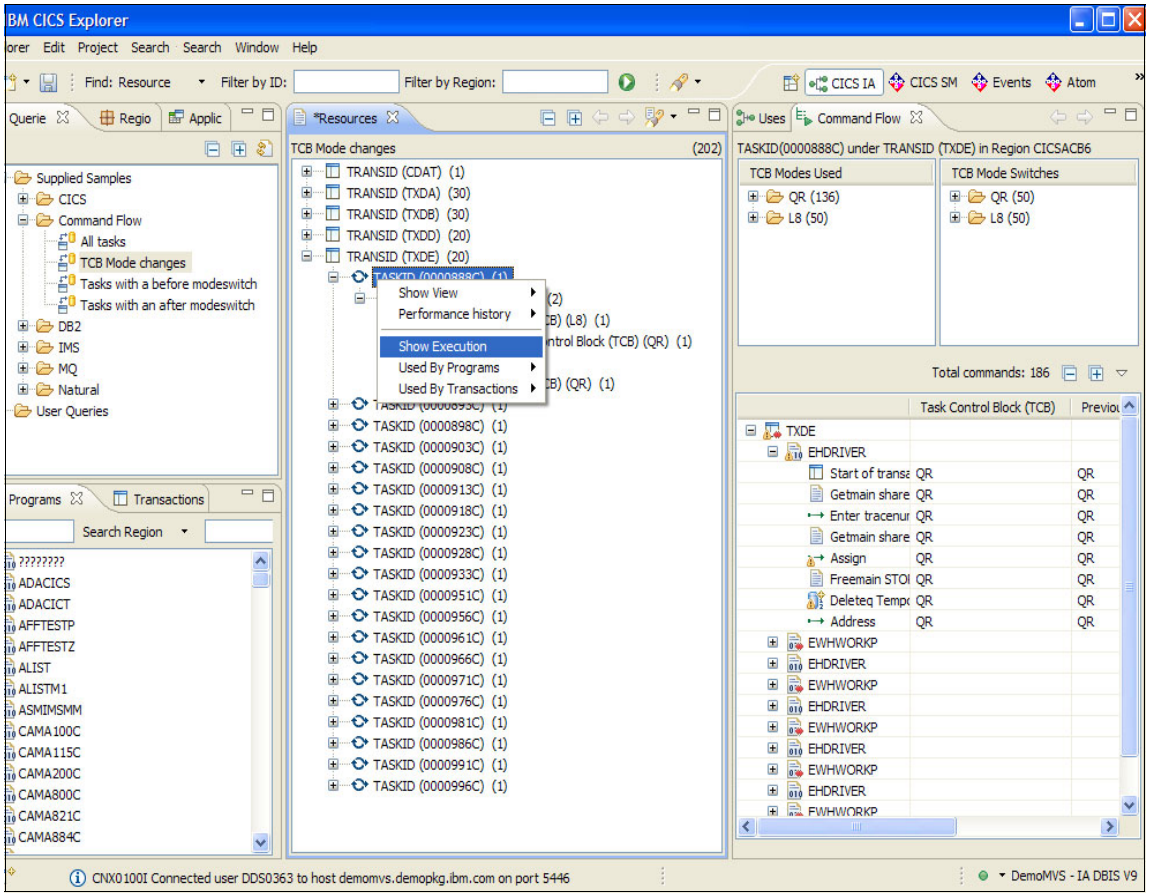


Figure 6-11 View of CICS Explorer Interdependency Analyzer plug-in

6.5.3 CICS Explorer Configuration Manager plug-in

The CICS Explorer Configuration Manager (CICS CM) plug-in, shown in Figure 6-12 on page 231, provides an Eclipse-based infrastructure to view and manage CICS CM resource definitions across an enterprise. It supports a subset of the function available in CICS CM.

CICS CM introduces the concept of a configuration that can be backed by either a CSD file or a CICSPlex SM context, which means that the question of how a resource is stored is removed from you because CICS CM provides a single experience regardless of whether you are dealing with a CSD or a CICSPlex SM context. CICS CM provides a single point of control for editing, reporting, and migrating CICS resource definitions across an enterprise. CICS CM is a CICS application that can read from and write to a CICS system definition (CSD) files and CICSPlex SM contexts.

You can use the CICS CM plug-in to perform the following tasks:

- View all the CICS CM configurations.
- View all lists in a configuration and groups in a list.
- View all groups in a configuration.
- View orphaned resources and groups in a CICSPlex SM configuration.
- View orphaned groups in a configuration.
- View history for, resource definitions, configurations and groups and restore changes made to a resource definition.
- View, edit, and delete resource definitions.
- Search across one or more configurations and search across one or more groups in a configuration.
- Create new resources.
- Install resources from a Configuration in one or more active CICS systems.
- Compare two lists in the same configuration, two groups in the same or different configurations, or two of the same type of definition.
- Search for groups.
- Search the history for a configuration or a group.
- Install a group if you have a CICSPlex SM connection.
- Add a group to a list for a CSD.

When you open the CICS CM plug-in for the first time, the following default views are displayed:

- The Configurations view displays all CICS CM Configurations. Both those representing CSDs and those representing CICSPlex SM Contexts. The configuration information is retrieved when you connect to CICS CM and contains details of the CSD or context that the configuration represents.
- The Lists view displays all lists in the currently selected configuration. For a configuration that represents a CSD, lists represent the Group Lists in the

configuration. For a configuration that represents a CICSPlex SM Context, lists represent the ResDescs in the configuration.

- The Groups view displays all Groups in either the currently selected configuration or the currently selected list. Again, for a configuration that represents a CSD, groups represent the csdgroups. And for a configuration that represents a CICSPlex SM Context, groups represent the ResGroups.
- The Search Results view displays search results on resource definitions.
- The History view allows you to view changes made to definitions.

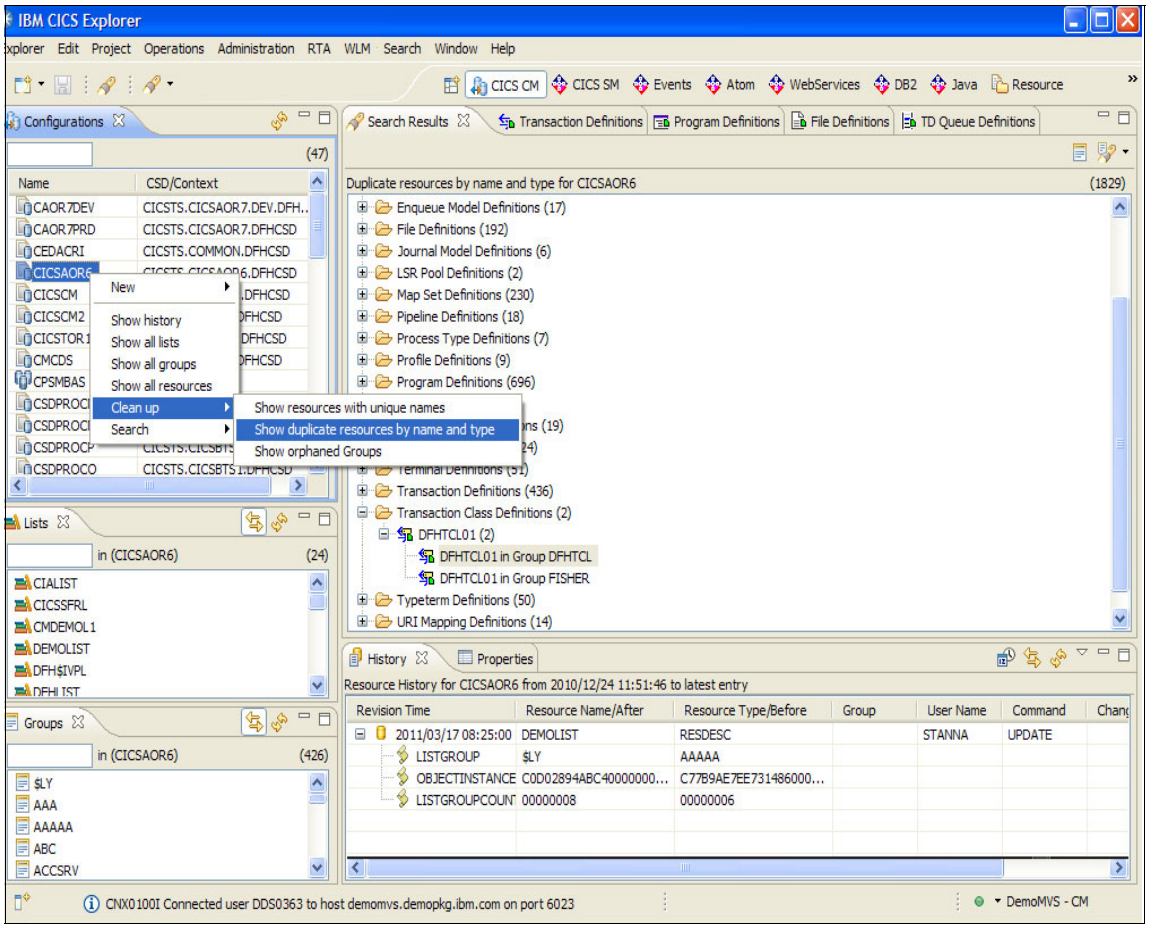


Figure 6-12 View of CICS Explorer Configuration Manager plug-in

6.5.4 CICS Explorer Deployment Assistant plug-in

The CICS Explorer Deployment Assistant (CICS DA) plug-in, shown in Figure 6-13 on page 233, works in conjunction with CICS Explorer to help to discover, model, visualize, and deploy CICS regions. It allows CICS systems programmers to create and maintain a CICS environment easily. One can deploy new and existing CICS regions using automation and policy-driven advice, improving productivity and reducing complexity, time to effectiveness, and risk.

Using CICS DA you can perform the following tasks:

- Discover and visualize CICSplexes and CICS regions in a sysplex.
- Create an offline model to share with others.
- Discover connectivity relationships and major subsystem dependencies.
- Start and stop CICS regions.
- View held and running z/OS jobs and the System Log.
- Clone existing CICS regions by creating and submitting JCL directly from CICS Explorer.
- Add an existing CICS region to a CICSplex.

When you start the CICS DA plug-in for the first time the following default views are displayed:

- DA Projects view provides a hierarchical view of the resources in the CICS Explorer Workbench. When you create a CICS DA project it is shown in the DA Projects view, and the system information discovered by CICS DA is imported into the project. The imported information is known as the model. The components of the model are the CICS regions in the system. From the DA Projects view you can view the model and its components, and perform actions against them.
- z/OS Command view provides a command line interface to the z/OS console. You can type an z/OS command in the text field and the response is shown in the view.
- System Log view shows the last 1000 messages in the system log for the current CICS DA connection.
- z/OS Job view shows all the output data sets for a specified job. The job name can be entered in the Job ID field or the Link to Selection button can be selected so that any job that is selected in the Jobs view is automatically displayed in the z/OS Job view. When you select an output data set in the z/OS view, the content of the data set is displayed in the right pane.

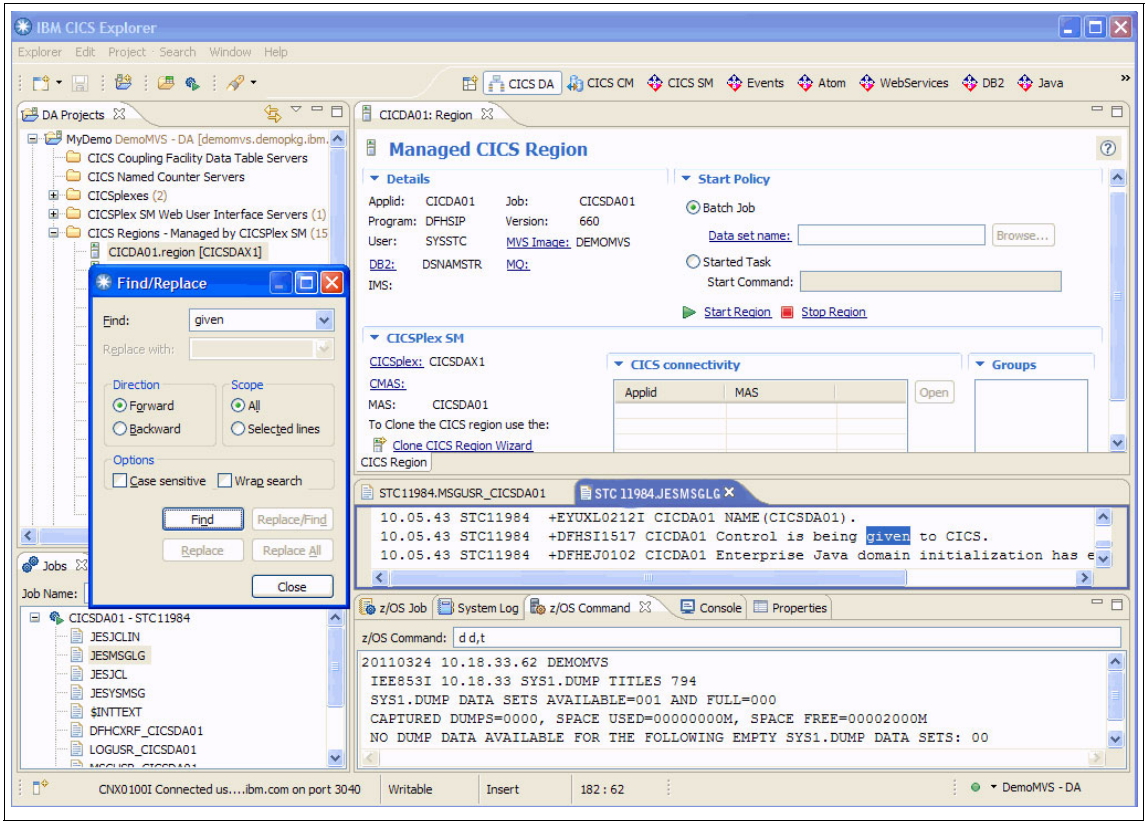


Figure 6-13 View of the CICS Explorer Deployment Assistant plug-in

6.5.5 OMEGAMON XE for CICS: A Tivoli OPAL plug-in

The OMEGAMON XE for CICS plug-in, shown in Figure 6-14 on page 234, provides the ability to view OMEGAMON data, in Eclipse-based applications, such as CICS Explorer. Most Tivoli monitoring systems, including their status information, can be displayed. Monitoring data from Tivoli OMEGAMON XE for CICS and Tivoli OMEGAMON for CICS TG, can be accessed. It provides the ability to proactively monitor basic and complex CICS systems. It provides performance and availability information that can be leveraged to pinpoint bottlenecks and avoid downtime. OMEGAMON XE provides insight into currently occurring issues, which allows the user to choose an informed course of action.

The IBM Tivoli Monitoring perspective is made update of the following default views when you first open the plug-in:

- Navigation Tree display view is similar to the TEP client topology display and performs a function that is similar to the CICS Explorer CICS SM perspective topology panel. Clicking an item in the tree controls the scope of the resources that are displayed in the query results view.
- Monitoring Data display view contains query results. This panel displays the results of a request for data from the SOAP server. The request is issued by selecting the report in the topology tree.
- Monitoring Situations display view displays the current TEMS situation events. The view is updated every 30 seconds.

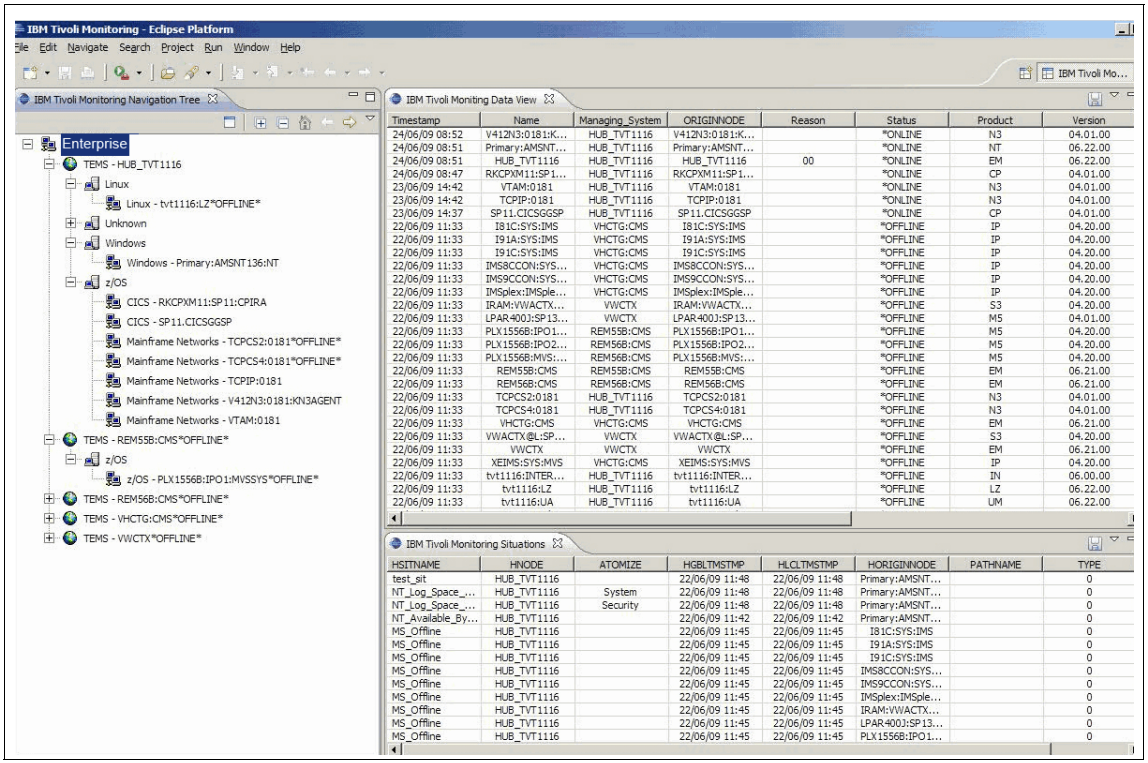


Figure 6-14 View of the CICS Explorer OMEGAMON plug-in

6.5.6 CICS Explorer Transaction Gateway plug-in

The CICS Explorer Transaction Gateway (CICS TG) plug-in, shown in Figure 6-15 on page 236, default view displays the CICS Transaction Gateways that are being used and the CICS server connections associated with the CICS Transaction Gateway. You can test Gateway and CICS connections and refresh the status of the CICS Transaction Gateways.

The following default CICS Explorer CICS TG plug-in views are provided:

- The Gateway daemons view displays status and activity information for Gateway daemons that are both defined and available. You can test a Gateway connection and customize the columns in this view.
- The CICS connections view displays status and activity information for the defined CICS server connections and active CICS server connections. You can test CICS connections and customize the columns in this view.
- The CICS TG Explorer view displays the CICS Transaction Gateways that are being used and the CICS server connections associated with the CICS Transaction Gateway. You can test Gateway and CICS connections and refresh the status of the CICS Transaction Gateways.

When you switch to the CICS TG perspective, you obtain fast access to the new views. Each view includes a number of attributes by default. You can add or remove attributes as required. You can see the full set of attributes in the Properties view. The attributes are grouped into predefined categories.

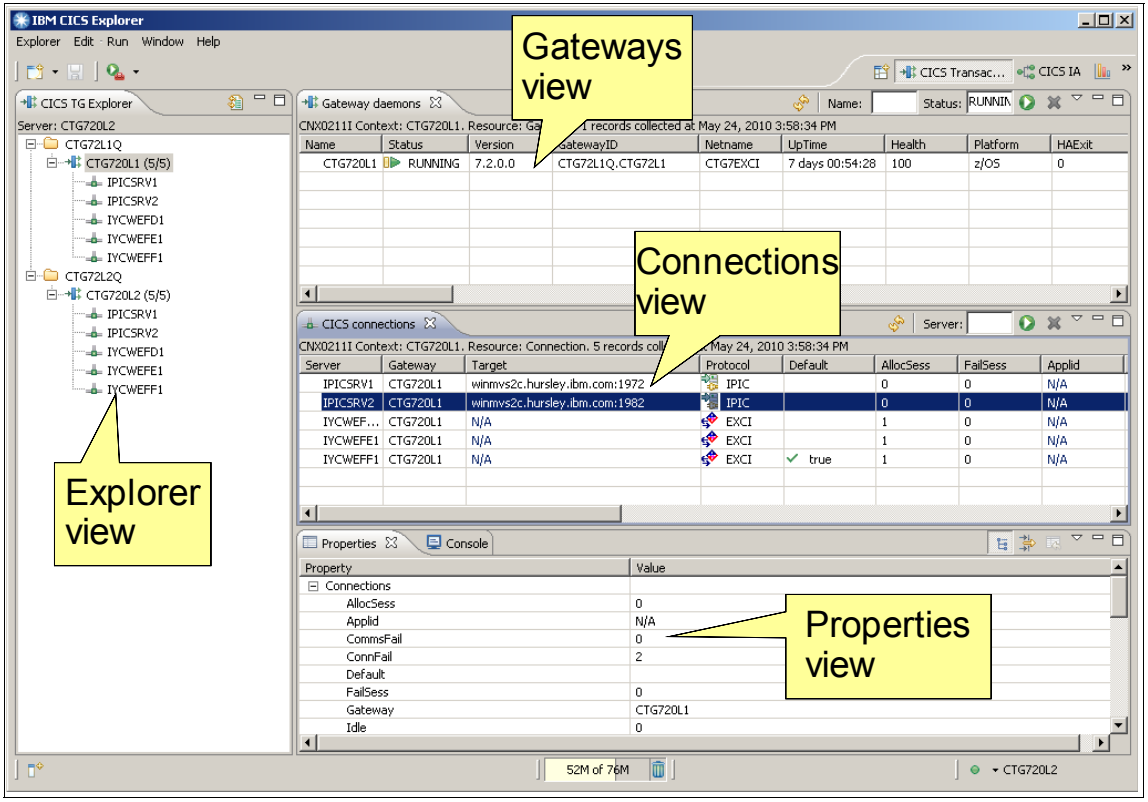


Figure 6-15 View of the CICS Explorer Transaction Gateway plug-in

6.5.7 CICS Explorer WebSphere MQ plug-in

The WebSphere MQ plug-in for CICS Explorer, shown in Figure 6-16 on page 237, makes it is possible to operate CICS systems and WebSphere MQ resources from the same desktop application, providing benefit if you carry out cross system administration, monitoring, and problem determination. With this single point of control, you can manage you CICS resources and you WebSphere MQ resources together.

The following default WebSphere MQ views are shown when you open the plug-in for the first time:

- MQ Explorer Navigator view displays with a tree representing the parts of MQ. Some parts include: Queue Managers, Queues, Channels, Client Connections, Listeners, Services, and Process Definitions.
- Content view shows information about the selected item in the Navigator view tree.

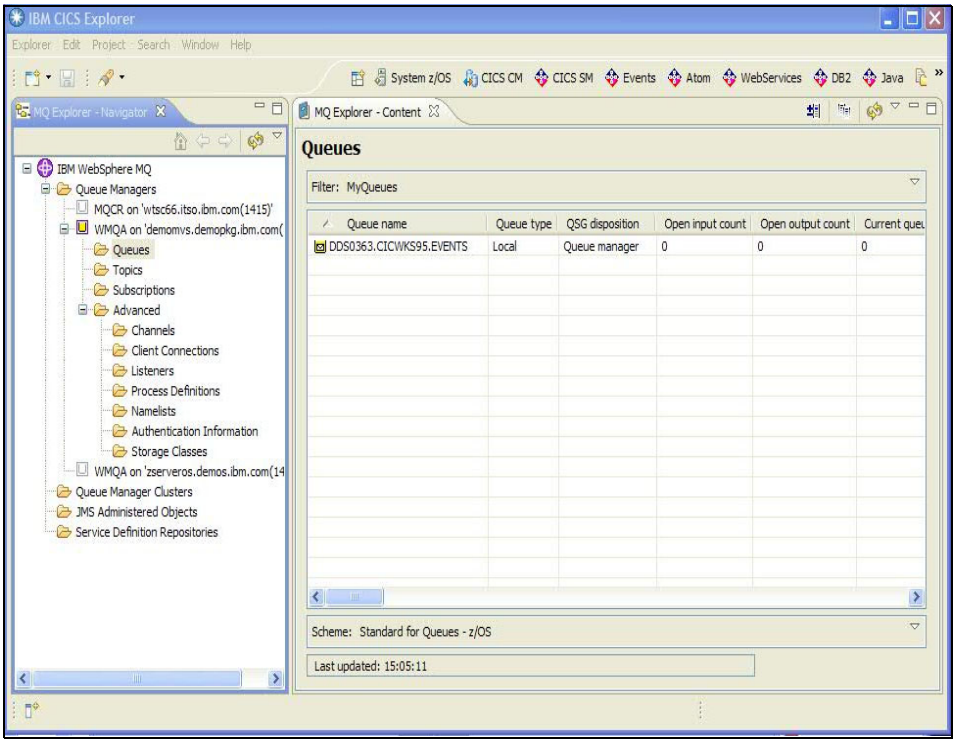


Figure 6-16 View of WebSphere MQ plug-in for CICS Explorer

6.6 IBM Rational Developer for System z (RDz)

The IBM Rational Developer for System z application development workbench, shown in Figure 6-17 on page 239, provides your development teams with the tools they need to speed the development of traditional mainframe, web, and composite applications.

RDz consists of a common workbench and an integrated set of tools that support end-to-end, model-based development, runtime testing, and rapid deployment of simple and complex applications. It offers an integrated development environment (IDE) with advanced, easy-to-use tools and features to help WebSphere, CICS, and IMS developers rapidly design, code, and deploy complex applications.

Some tooling that can be integrated with RDz with CICS Explorer are:

- Web Services
- Service Flow
- SCA
- CICS Event Bindings
- COBOL and PL/I
- Java
- EGL
- Debug and Fault analysis
- File Management
- Applications Deployment Manager
- CICS Explorer and CICS Tools

Developers can create or maintain existing application programs with RDz, switch to the CICS SM perspective (within RDz) and newcopy programs, add new program definitions, transaction definitions, file definitions, execute and test from one interface.

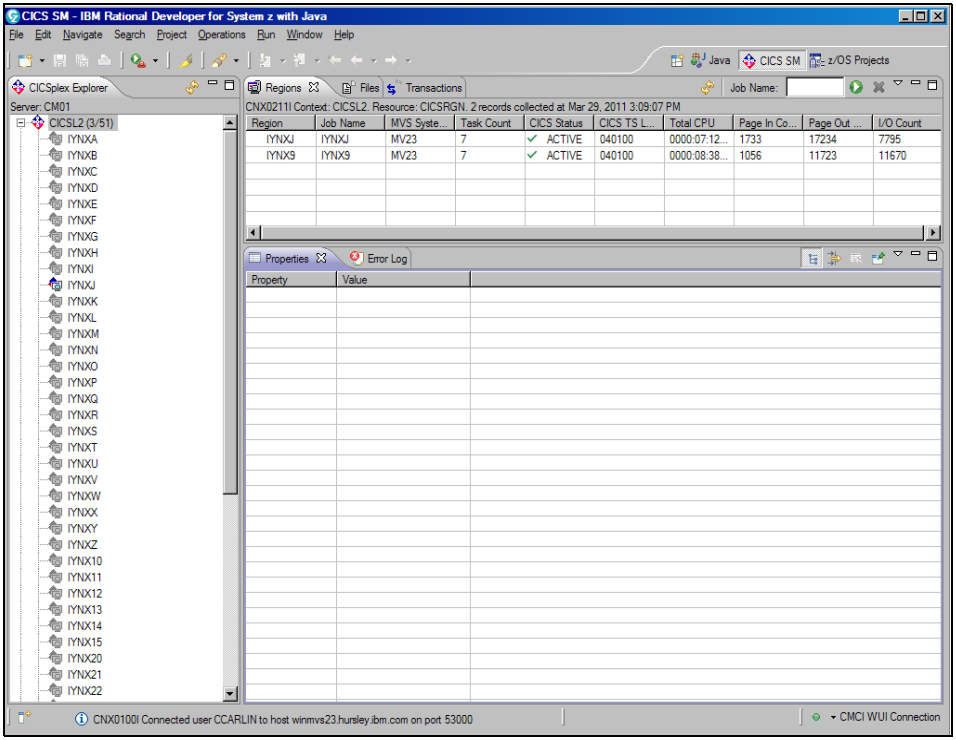


Figure 6-17 View of RDz with the CICS SM perspective

For more information about RDz, visit the Rational Developer for System z website:
http://www-01.ibm.com/software/rational/products/developer/systemz/

7

CICS tools

The CICS Transaction Server environment is an extremely flexible platform used to run applications ranging from simple applications consisting of just a few transactions and programs to highly-complex applications built up of thousands of CICS resources that span many CICS regions on multiple LPARs and platforms.

With such a wide range of applications, customers have varying levels of service requirements and processing volume demands placed on them. Each individual shop builds their own unique CICS environment to meet these processing and business requirements, which are usually built up over many years of growth.

To help manage the wide spectrum of application environments that can be built, IBM created a family of CICS-related tools to help the CICS Professional (whether they are developers, administrators, operators, or systems programmers) design and manage CICS applications and the systems environments they run in.

Figure 7-1 on page 242 lists the current suite of tools, each providing its own solution to common problems that a shop might experience as their applications grow in scope and complexity.

Figure 7-1 CICS tools family

In the following sections, we describe each tool and give you a simple example on how it can help improve your productivity and effectiveness as a CICS Professional.

7.1 CICS Performance Analyzer for z/OS

IBM CICS Performance Analyzer for z/OS is a performance reporting and analysis productivity tool that helps the CICS Professional explore historical performance and statistics information about their systems and applications using SMF data collected from the CICS Monitoring Facility, CICS Statistics and Server Statistics, CICS Transaction Gateway Statistics, IMS, DB2, WebSphere MQ, System Logger, and Omegamon XE. See Figure 7-2 on page 243 to see the inputs and components that make up the CICS PA environment.

Many of today's large CICS customers run mission critical applications that traditionally experience high transaction per second processing rates. CICS PA gives you access to reports and graphics that help you identify, analyze, and resolve performance bottlenecks, along with the ability to generate extracts that can be imported into spreadsheet tools for analysis on your workstations.

CICS PA provides you with two interfaces, the TSO/ISPF interface gives you access to the major reporting component with over 180 predefined reports along with the ability to create your own custom reports and extracts.

The CICS Explorer plug-in interface integrates the analysis process with your day-to-day interactions, detailed information and graphs are just a click away, for

example, while you are displaying CICS resources in the CICS Explorer, you can right-click an item and display historical performance information, such as Response Time, Storage, File, Threadsafe, or CPU usage graphs.

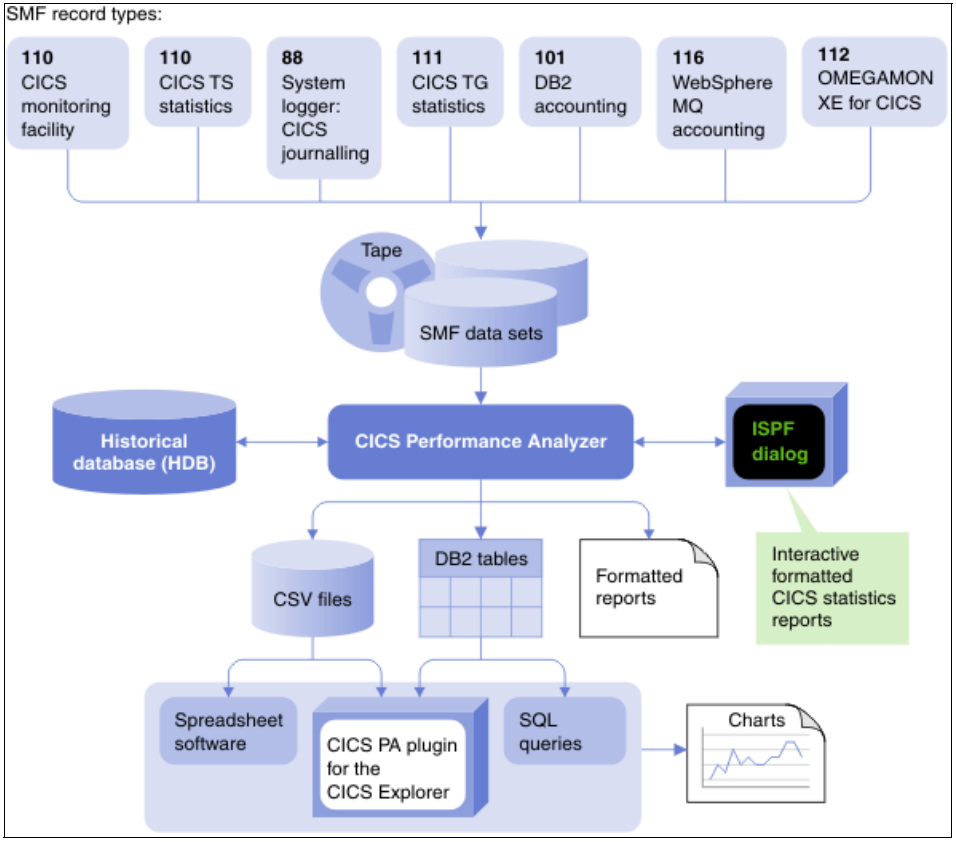


Figure 7-2 CICS Performance Analyzer Components

CICS PA provides you with many advanced features, such as Cross System reports that combine your performance information from multiple CICS regions to give you the complete picture of your multi region transactions. You can also group transactions, or using Transaction Profiling you can make before and after style comparisons to identify changes over time periods, such as before and after an upgrade.

7.1.1 Performance Analyzer usage scenario

In this section, we provide usage scenarios for CICS Performance Analyzer (CICS PA).

CICS PA usage scenario 1

A customer calls the bank's support team to report an application running slow. The support team is not finding any significant problems with the system, so they decide to check CICS PA to see how this transaction ran in the past few weeks. Quickly they notice that the application started to run slower three days ago. They check CICS Configuration Manager to see if there was any change activity on this application and notice that a change package was installed Monday morning.

Further analysis identified that a program module changed the way they searched for customer records and caused a significant slowdown compared to the way searches were performed before the change was installed.

CICS PA usage scenario 2

The store's support team were asked to look for ways to reduce costs. They decide to use the Threadsafe Analysis feature of CICS PA to look for heavy-hitter customer transactions that are not defined as Threadsafe. With the results of their analysis, they identified several high-volume transactions that were not configured as Threadsafe in CICS. By working with the development team that owned each program, the programs were re-configured as Threadsafe.

7.2 CICS Interdependency Analyzer for z/OS

Over the years, many companies built large complex CICS environments running potentially hundreds of applications, each accessing resources, such as files and databases, and interacting with other application components and possibly interacting with other applications running on distributed systems.

CICS Interdependency Analyzer for z/OS is a productivity tool used to discover and analyze CICS resources and identify relationships between them.

CICS Interdependency Analyzer for z/OS (CICS IA) uses a runtime scanner to gather actual usage information related to the execution of CICS applications, giving you a true picture of your application and its interactions. CICS IA accomplishes this by monitoring applications for API and SPI commands along with optional DB2, IMS, MQ and COBOL calls to give you a complete picture of your application and the interactions and resources that are referenced along with their inter-relationship. See Figure 7-3 on page 245 for a chart showing the collector and reporting structure of CICS IA.

CICS IA collects this dependency information, storing it in a DB2 database that allows you to run queries and reports using an offline reporting interface or using the CICS IA plug-in under the CICS Explorer, which along with other CICS Tools

plug-ins provides a seamless integration between the productivity tools. For example, while you display CICS resources in the CICS Explorer, you can right-click an item and display all resources related to it.

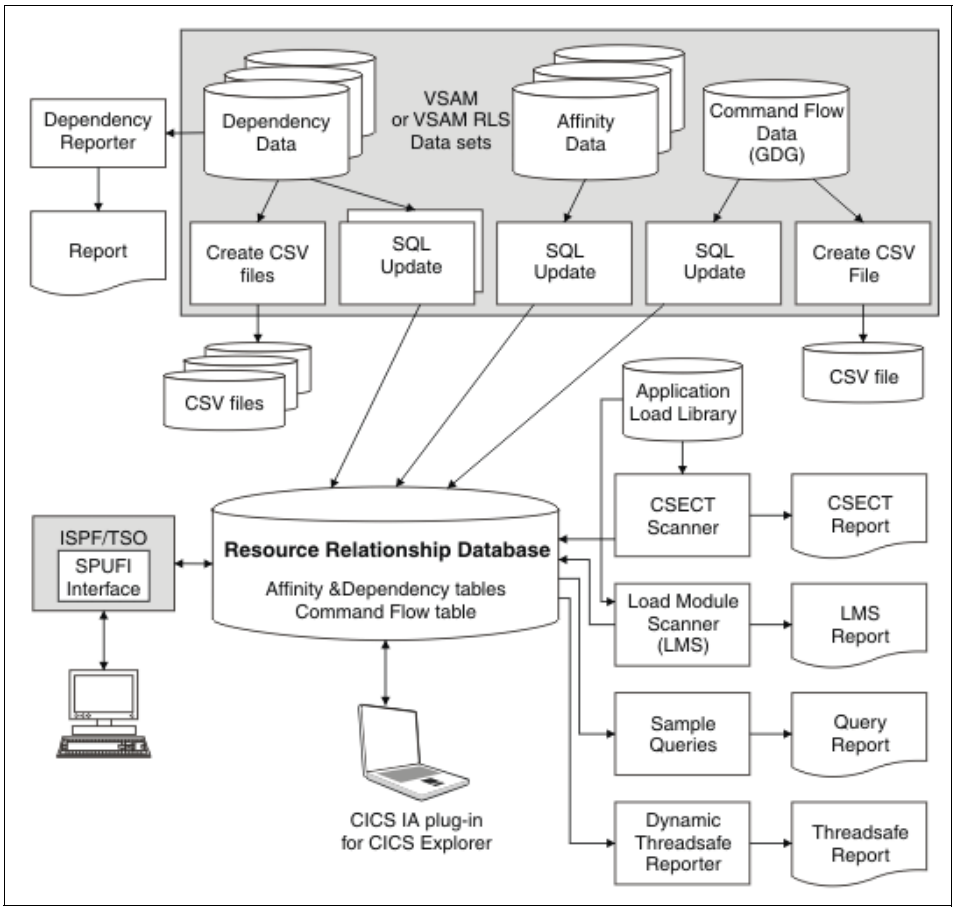


Figure 7-3 CICS Interdependency Analyzer Reporting Structure

As our work environments challenge us to do more with less, CICS IA can help a CICS professional analyze applications for Threadsafe considerations to improve the overall execution efficiency. You can run reports to analyze your applications to identify the impacts of a CICS version or release upgrade by identifying what API commands are affected to help with upgrade planning and testing.

CICS IA's Command Flow feature gives you the ability to capture and view all EXEC CICS, SQL, MQ, and IMS calls in chronological order. This feature can be helpful in diagnosing the flow of your application as it executes along with the

ability to identify TCB mode switches to help with tuning and Threadsafe analysis.

In a CICS environment where Work Load balancing is being used, CICS IA can provide affinities analysis and can also generate affinity-transaction-group definitions that can be directly input into CICSPlex SM.

7.2.1 CICS Interdependency Analyzer usage scenario

In this section, we provide usage scenarios for CICS Interdependency Analyzer.

CICS IA usage scenario 1

The bank's support team gets a call that a customer is having a problem. They use CICS Interdependency Analyzer to identify all of the components related to the customers transaction and then analyze the entire application to see where the potential problem is occurring.

CICS IA usage scenario 2

In the previous section, the store's support team used the CICS PA product to identify heavy-hitter transactions that were good candidates to be converted to Threadsafe. Using CICS IA's built in Threadsafe Queries and Command Flow feature, they were able to review the application to help them identify any possible issues that can prevent them from configuring the application as Threadsafe.

7.3 CICS Configuration Manager for z/OS

In a CICS environment, the CICS systems programmer or administrator has two choices on how to manage application and system resources: the CICS System Definition (CSD) or CICSPlex SM Business Application Services (BAS). The CSD is the default method used to manage CICS Resources and CICSPlex SM provides BAS, which takes resource definition to the next level by providing administration from an Application perspective.

IBM CICS Configuration Manager for z/OS (CICS CM) takes both methods to a higher level of management by providing Auditing and Reporting, Controlled Backout, Redundancy and Cold Start Analysis, Packaging and Approval support all in a Secure managed environment.

As the number of CICS regions increase, administrators can experience difficulties trying to migrate application changes as they are promoted through the various development, testing, and production environments. CICS CM

simplifies the process by allowing you to deploy changes automatically and dynamically from a single point-of-control.

CICS CM interacts seamlessly with both CSD-defined resources and CICSPlex SM BAS resources by automatically translating stored resources into their correct form and structure when passing from one repository type to another. Figure 7-4 shows how CICS CM integrates into the CICS environment.

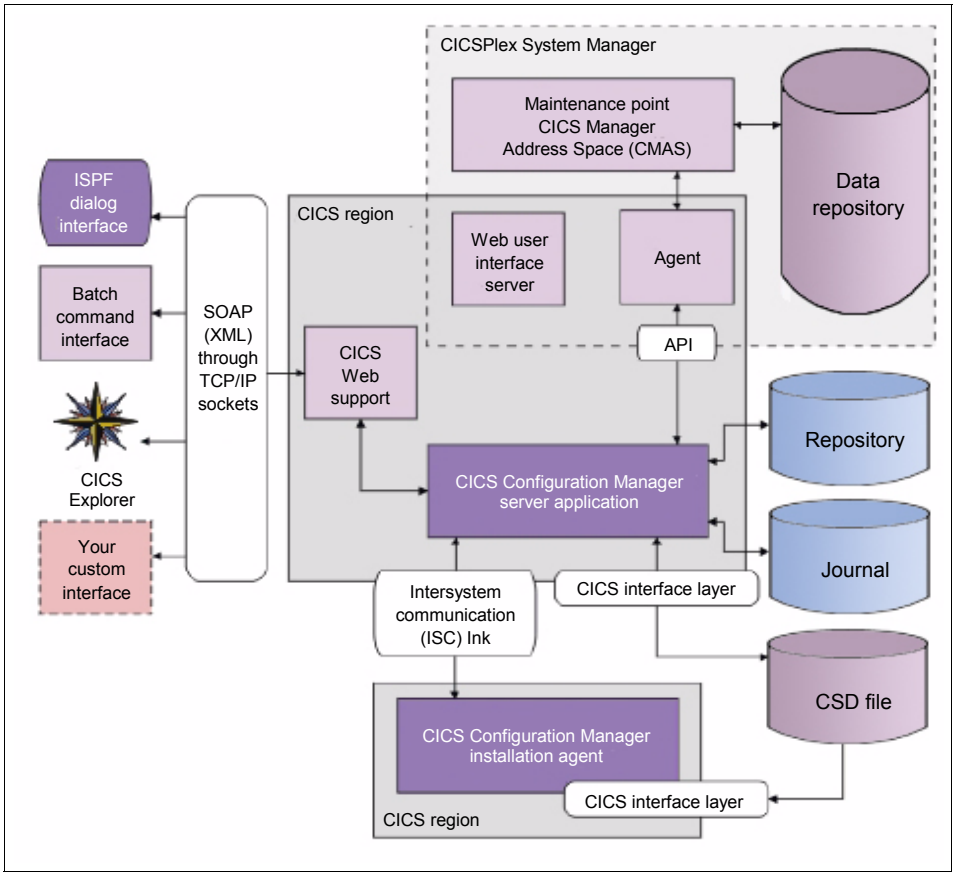


Figure 7-4 CICS Configuration Manager Architecture

CICS CM provides you with many usability features, such as security and standards enforcement, where you limit who can see or modify resources. You can also enforce how changes are made, such as implementing naming or parameter use standards. CICS CM also provides you with features, such as Change Packages, Migration Schemes, Transformational Rules and Approval Profiles, to give you full control over every phase of the definition, migration, and

install process to allow you to distribute the management of your CICS resources to include development staff into the process.

7.3.1 CICS CM usage scenario

The bank's change team saw a significant growth of work in their area and are experiencing trouble keeping up with all the work. They noticed that when they implement changes the change actions are documented in a change control record of which they use as input to her change process.

After reviewing the Change Package process of CICS CM, they worked with their development teams to take advantage of CICS CM to allow the developers to build change packages of CICS Resources that they can approve and install without having to manually process the commands for each region they update.

7.4 CICS Deployment Assistant for z/OS

IBM CICS Deployment Assistant for z/OS (CICS DA) is the newest member to the CICS Tools Family of products bringing CICS Region and System management to the table. CICS systems programmers or administrators can use CICS DA from within the CICS Explorer to discover, model, visualize, create, and deploy CICS regions using the CICS DA plug-in interface.

Using the CICS Explorer, you can view resources in a CICS region or in a CICSPlex. Adding the CICS Deployment Assistant extends the scope outside the CICS environment into the z/OS arena, giving you a complete picture of every aspect of your CICS environment and providing you with a complete topology of your CICS Environment.

CICS DA provides a Discovery feature that uses a two-phase process to discover information about running CICS regions, and then drills down into each CICS region to discover detailed information, such as startup options, connectivity relationships, and major subsystem dependencies. After the discovery process is complete, you can save an offline model in an XML format that can be shared with other team members.

Figure 7-5 on page 249 shows the CICS DA plug-in in the CICS Explorer viewing a CICS region, with the Plex selection information available in the left and the regions MSGUSR log in the bottom.

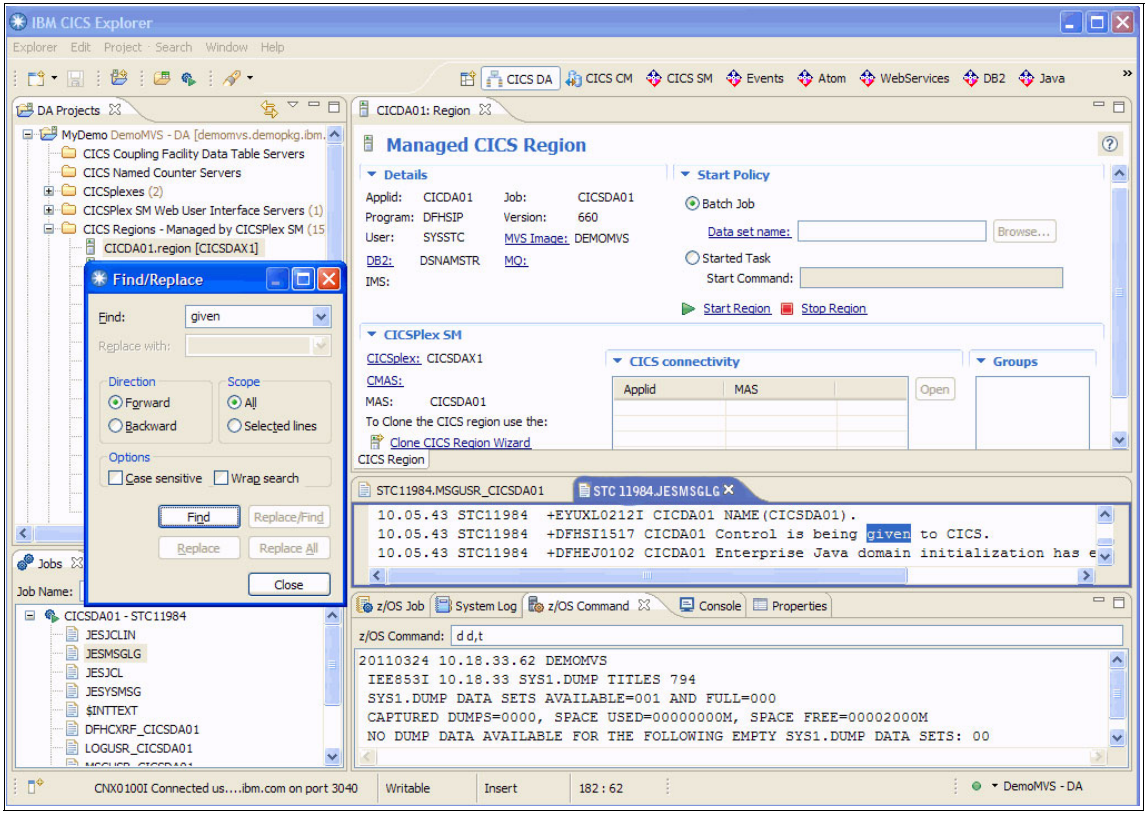


Figure 7-5 CICS Deployment assistant

The offline models can provide you with a graphical visualization of your CICS topology that integrates with the CICS Explorer to allow you to visualize and navigate through your topology using the CICS SM Operations and Administration views.

CICS DA also provides you with operational functionality to allow you to start and stop CICS regions along with viewing their associated job logs and system logs.

Using the automations capabilities of CICS DA, you now have the functionality at your disposal to clone existing CICS regions and add CICS regions to your CICSPlex.

CICS DA gives you the ability to create guided help documents called “Cheat Sheets”, viewable from the CICS Explorer that you can use to create custom operational process and procedure documentation.

7.4.1 CICS DA usage scenario

The bank's support team is working on a new project to bring onboard some new business applications that the bank just acquired. After they have access to the new LPAR they set up, CICS DA can then run a discovery against the system to generate a list of all CICS components that are related to the new systems. They can then use this information to update the Operations Team's process and procedures along with building a CICSPlex to allow them to control the new environment in a similar fashion that they manage their existing systems. Using the CICS DA offline model, they provided other team members with documentation of their new business environment.

7.5 CICS VSAM Recovery

Almost every CICS shop has VSAM files that house critical customer application data. CICS VSAM Recovery provides the CICS professional with the ability to manage and automate the recovery of damaged, corrupted, or lost VSAM files for both the CICS and Batch environments.

When VSAM data is vital to the business, CICS VR can greatly improve the recovery process by automating the construction of the required jobs to restore and forward recovery VSAM datasets in the event of a data loss.

Figure 7-6 on page 251 shows the components that make up CICS VR, illustrating how they integrate with the MVS System Logger. Updates to your VSAM datasets are recorded by the MVS System Logger. The logs created by the MVS System Logger can be used along with backup copies of the VSAM datasets to construct the required jobs to forward recover or backout our CICS VSAM datasets.

CICS and batch use the MVS System Logger to save before-images and after-images, which are used along with backups created by DFSMShsm, DFSMSdss, or any other backup product, to perform forward recovery or batch backout. CICS VR completely manages the recovery process using System Managed Storage (SMS) services and the Recovery Control Data Set (RCDS) to gather the information that is required to build your recovery jobs.

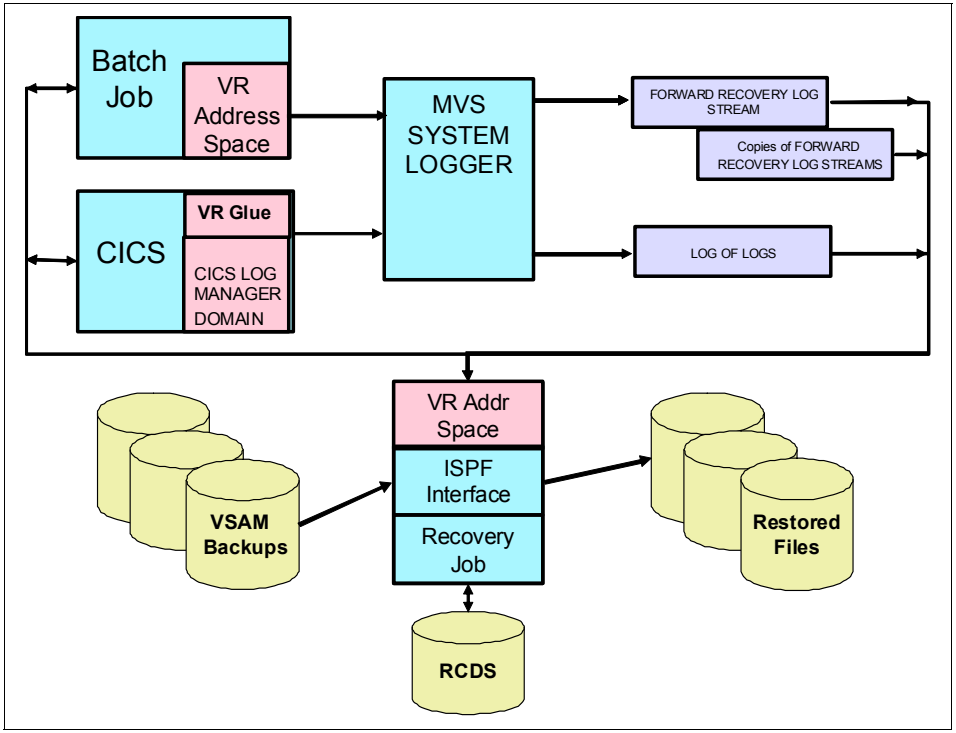


Figure 7-6 CICS VSAM Recovery

7.5.1 CICS VSAM Recovery usage scenario

One of the bank's major credit card applications has been running for years with no problems, but just last week they experienced a major outage and after recovery they were missing all customer updates that were made after the last backup image. They were able to use CICS VR to successfully perform a forward recovery starting from the last backup, restore all customer changes to the file, and successfully bring it back online. Because they do not have failures of this type often, they were glad that they had CICS VR to generate the recovery jobs for them. CICS VR allowed them to the get the application up and running as quickly as possible.

7.6 CICS VSAM Transparency

As applications expand or look for ways to modernize, the back-end data repository comes into play, and many shops look at standardizing around a single source of production data with the ability to provide 24 x 7 access.

Migrating VSAM-based applications to DB2 can provide applications with the ability to integrate with newer applications and to provide ad-hoc queries to their existing data. Additionally they can take advantage of newer products, such as analytical engines and can capitalize on the abundance of database management tools and DBA services.

CICS VSAM Transparency (CICS VT) was introduced to help applications migrate their application's data from VSAM files to DB2 without incurring the costs of application re-designs or costly and complex rewrites. CICS VT provides a solution where the actual data is moved from VSAM into DB2, while the application continues to operate unchanged performing VSAM programming operations, which are intercepted by CICS VT, to make the actual data retrieval calls from DB2.

Figure 7-7 on page 253 shows how CICS VT intercepts all VSAM calls and decides if the data is retrieved from VSAM or DB2. For each VSAM call, a check is made to see if the VSAM data moved to DB2. If not, processing continues normally. If the data was moved to DB2, CICS VT uses generated drivers to access the data through SQL calls and return it to the calling program in the original file format that was expected before the data was migrated.

The design and functionality of CICS VT allows applications to implement a staggered approach to migrating VSAM files to DB2 as opposed to a mass and complex all or nothing approach. Each file can be migrated and tested on a one-by-one bases without change to the original applications. After the data is in DB2, future changes to the application can include DB2 access to eventually replace the actual VSAM access calls.

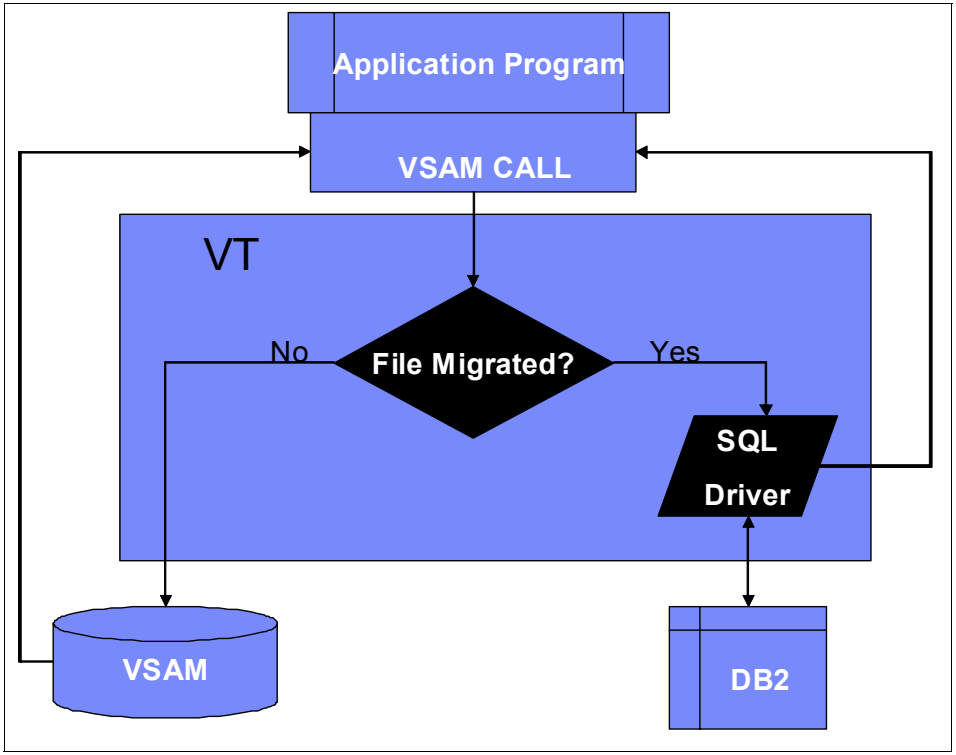


Figure 7-7 CICS Application VSAM Call Using CICS VSAM Transparency

CICS VT provides several tools and components to aid with the conversion process. The mapping component identifies the relationship between the original VSAM file layout and its proposed DB2 format. For VSAM files that map to a single DB2 table, CICS VR provides an automated mapping facility and for more complex mappings, a manual feature provides you advanced options to allow a mapping to multiple DB2 tables.

After the mapping phase is complete, the data can be migrated to DB2 using a combination of CICS VR and DB2 utilities along with a verification process. CICS VT provides a dual-mode facility to allow testing against the original VSAM file and the new DB2 tables during the cutover process. Both modes of access can be tested and verified to ensure the accuracy of the migration before a final cutover.

The mapping process generates CICS VT run-time drivers, containing static SQL calls, which are used by CICS VT to access the DB2 data and return the results to the calling application in the original VSAM file format, thus earning the name VSAM Transparency.

7.6.1 CICS VSAM Transparency usage scenario

The store is struggling to make their applications' data available to other teams because they are already having difficulties meeting their SLAs with the current processing load. They saw a presentation on CICS VT and decided that moving their applications' VSAM data over to a DB2 environment can allow them to continue running their current applications as is without interruption but can also allow concurrent access to applications' data by other teams through DB2 Queries.

After everything was up and running, they also benefited from the existing DB2 DBA teams skills using their suite of DB2 tools to improve the recovery and management capabilities of his applications data.

7.7 CICS Batch Application Control

CICS is traditionally an online transactional environment, but in some situations, applications require the ability to take some or all of their application resources offline for batch processing. As applications grow, the online requirements for an application can become more critical. CICS administrators frequently require a solution to minimize the outages to applications while providing the capabilities of a batch processing window.

IBM CICS Batch Application Control allows batch programs to dynamically change the state of CICS system-owned resources, such as VSAM files, Transient Data Queues, programs and transactions.

CICS Batch Application Control (BAC), shown in Figure 7-8 on page 255, provides you with a batch request facility that allows your batch job to control CICS resources at the job-step level. CICS BAC uses a control file to coordinate all interactions, and each request must pass security checks before being allowed to modify a resource. Each action is also logged in an audit log.

Figure 7-8 on page 255 shows a CICS Region and a Batch Stream sharing access to a VSAM File using the CICS BAC servers and utilities.

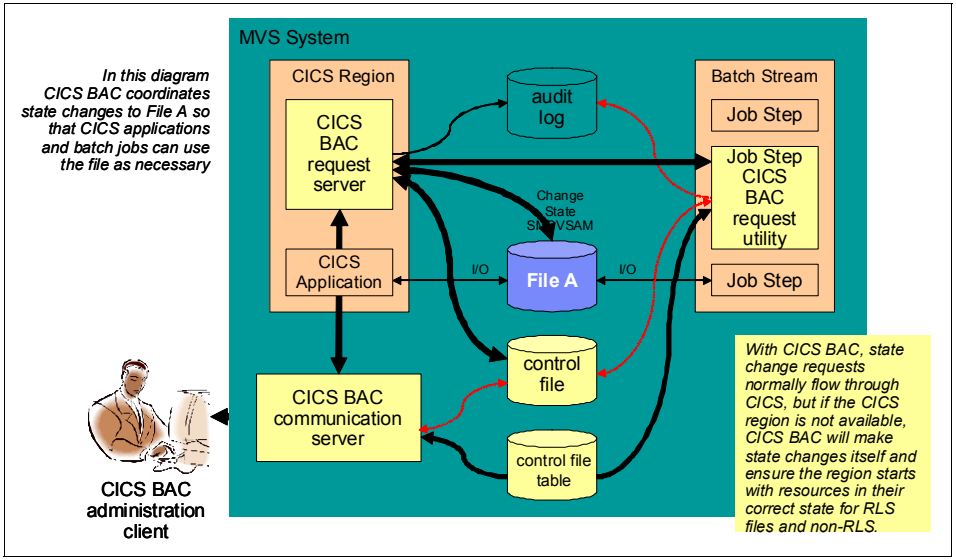


Figure 7-8 CICS Batch Application Control

7.7.1 CICS BAC usage scenario

Jim is the application analyst for the bank's application, which runs in CICS Region CICRED01 along with several other applications. For years region CICRED01 started at 7:30am and shutdown daily at 10:00pm to allow a nightly batch update window. Recently, some new sites were added to the bank's application that caused a conflict with the CICRED01 region's SLA. The bank's application team asked to have the region up longer to allow the new customers, whom are in Asia, extended time to use the systems.

Jim pulled together a team of people to research the problem and came up with a solution to use CICS Batch Application Control for z/OS to allow them to keep region CICRED01 up until 3:00 am. This solution also meets the other applications' batch cycle by allowing them to selectively take some of the application files off line for batch processing while the IBM Redbooks applications files continue in an online fashion.

CBAC gave Jim's team the flexibility to provide varying levels of service to different applications all running in the same CICS Environment without having to perform major restructuring of their applications and CICS environment. Each application can now add control steps to their own batch flows, giving them direct control over their own resources while not impacting peer applications running in the same CICS Environment.

7.8 IBM Session Manager for z/OS

IBM Session Manager for z/OS, shown in Figure 7-9, is a terminal session manager for VTAM and TCP/IP that allows access to multiple 3270 applications from a single 3270 terminal or a single workstation running a 3270 emulator.

In todays complex environments, you might have the requirement of connecting to many applications or systems to perform your job. IBM Session Manager is a middleware component that sits between users and the applications giving them a single secure sign-on and allowing access to all their applications from a single menu using multiple concurrent virtual sessions.

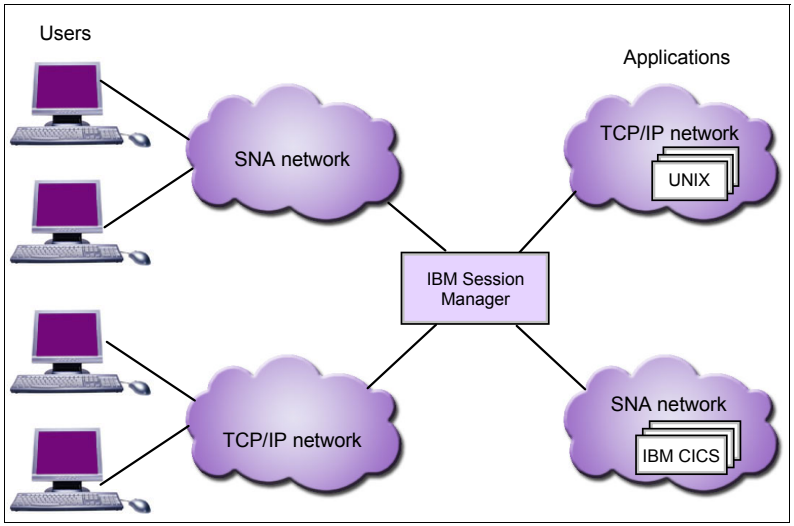


Figure 7-9 IBM Session Manager Configuration

IBM Session Manager provides you with usability features, such as single key stroke application switching, messaging between users, copy and paste between sessions, Extended Logon Facility to allow automated signon from a workstation, along with many advanced features, such as a scripting facility, an inbound/outbound compression facility called MISER, a help-desk Demonstration Facility, plus many more features not mentioned here.

7.8.1 IBM Session Manager usage scenario

The bank uses IBM Session Manager (ISM) to allow their system professionals to work in their TSO/ISPF environment while also accessing their CICS applications, this allows them to be much more productive since they only have to

log on one time and then have all their applications available using one menu. Switching between sessions is as simple as clicking a hot key.

7.9 CICS Online Transmission Time Optimizer for z/OS

For customers with large 3270 terminal networks, CICS Online Transmission Time Optimizer for z/OS can be used to optimize the 3270 data streams that flow over the network, which helps a customer reduce network utilization and improve response time. CICS Online Transmission Time Optimizer for z/OS (OTTO) works with the following network resources:

- 3270 terminals and printers
- SCS printers
- 3600/4700 banking terminals

Techniques, such as dynamically compressing data streams and eliminating repetitive characters, are an example of how OTTO can improve network utilization.

8

Introduction to the IBM Problem Determination tools

In this chapter we introduce the IBM tools that support application development on System z:

- Application Performance Analyzer for z/OS
- Debug Tool for z/OS and Debug Tool
- Fault Analyzer for z/OS
- File Manager for z/OS
- Workload Simulator
- CICS Explorer
- Rational Developer for System z

These tools provide you with a comprehensive suite of products that can help improve application delivery and enhance production availability and performance.

8.1 Introduction

IBM Application Performance Analyzer for z/OS, IBM Debug Tool for z/OS, IBM Fault Analyzer for z/OS, and IBM File Manager for z/OS, along with CICS Explorer and Rational Developer for System z, provide a robust suite of problem determination tools that can help improve application delivery throughout the application life cycle.

You can use these tools to help increase productivity and IT effectiveness across source code generation and debugging, application abend analysis, data management, and application-performance analysis. IBM problem determination tools do much more than support traditional applications. They include capabilities that enable you to build service-oriented architecture (SOA) applications. They are also tightly integrated with other tools in the IBM problem determination tools portfolio. All the tools also continue to support and make the most of the latest subsystem levels.

The IBM Problem Determination Tools provide support for all of the major IBM subsystems. Day one support is available for any new subsystem release.

As businesses and the critical IT systems that support them grow increasingly complex, application developers are constantly struggling to meet the demands placed upon them. Service oriented architecture has become widely accepted in IT today because it fulfills the promise of helping to address these demands. The ability to reuse existing assets is the cornerstone of SOA. This possibility is significant because reusing assets can be significantly less expensive than rewriting them. With the vast amount of existing applications running on the IBM System z platform, it only makes sense that System z applications can be a big part of SOA.

8.2 IBM Application Performance Analyzer for z/OS

Application Performance Analyzer for z/OS helps programmers and systems personnel to identify constraints and improve the performance of their applications. It is a non-intrusive performance analyzer that helps you during the design, development, production, and maintenance cycles.

The product's key function is to measure and report how system resources are used by applications running in a z/OS address space, such as TSO and batch and online subsystems, such as IMS, Customer Information Control System (CICS), and WebSphere Application Server, as well as DB2 stored procedures.

You can monitor applications in both test and production and in multiple source languages, including Assembler, C/C++, COBOL, PL/I, and Java. Optimized code support for COBOL and PL/I is provided to enable you to monitor production applications.

8.2.1 Highlights

The main features of Application Performance Analyzer are that it:

- Provides easy-to-use function that helps isolate application performance problems.
- Helps pinpoint performance bottlenecks affecting online transaction response times.
- Assists in reducing batch-application turnaround time.
- Supports Assembler, C/C++, CICS, COBOL, DB2, IMS, Java, PL/I, WebSphere MQ and WebSphere Application Server, including all the latest versions.
- Increases application understanding during stress and regression testing.
- Provides support for UNIX System Services.
- Provides support for IBM Parallel Sysplex Coupling Facility.
- Can be invoked from other programs such as IBM Tivoli OMEGAMON and IBM Tivoli Enterprise Portal.
- Provides the ability to compare two observation reports to see the relevant differences.
- Supports threshold monitoring.
- Supports Java source mapping, along with support for a wide range of Java applications.
- Provides general hints and tips along with guidance that can help you to improve performance.
- Utilization of zIIP or zAPP processors, if installed, is reported
- Includes an Eclipse-based GUI that provides a workstation interface to create sample requests and view reports such as those available through the host ISPF interface.
- Provides support for measuring multiple IMS MPP regions.
- Provides support for multiple CICS address spaces.
- Provides integration with CICS Explorer.
- Provides integration with Rational Developer for System z (RDz).

Using Application Performance Analyzer helps you maximize the performance of your existing hardware resources and improves the performance of your applications and subsystems. It lets you evaluate applications in the development and system test phases, review the impact of increased data volume or changes in business requirements on performance, and generate historical data and reports to analyze performance trends and evaluate program changes.

Running in a separate address space, Application Performance Analyzer non-intrusively collects resource utilization, wait time, and statistical samples from the address space being monitored. This data is then analyzed and documented. The reports that are generated help you to identify the key performance bottlenecks that require examination and resolution. This data is available online and in printed reports that you can choose to create as PDF files for viewing on a workstation.

8.2.2 Subsystem support

In this section, we discuss topics in subsystem support.

CICS

Support for IBM CICS Transaction Server enables you to monitor specific CICS transactions using wildcard transaction prefixes or by termID selection. With this support, you can trace transactions during critical situations rather than waiting to review data that is collected periodically. Java application code running in the Java 2 Platform and Standard Edition (J2SE) environment under CICS Transaction Server are also supported. Multiple CICS regions can be measured simultaneously and transaction data from these regions merged to produce a set of CICS reports showing multi-region activity.

DB2

Support for DB2 delivers relevant information for performance analysis and tuning, including SQL statements and processor usage by SQL statement as well as for IBM DB2 stored procedures written in a traditional language or in Java.

IMS

Support for IMS applications means that you can have IMS application performance data-on-call time and service-call time for DL/I. You can also choose to trace all IMS calls. A specific IMS transaction in a single IMS subsystem can be measured when it is eligible to execute in multiple MPP regions.

WebSphere MQ

WebSphere MQ support provides information about CPU usage by queue, by request, and by transaction. Application Performance Analyzer also provides service time by queue, request, and transaction, and wait time, for the same categories.

ADABAS/Natural

Support is provided for sampling and reporting on ADABAS calls, provided that ADABAS=YES is set in the CONFIG SAMPLE statement. When creating a NEW sampling request, an ADABAS extractor is presented if this configuration change has been done.

8.2.3 Java support

The Java data extractor, when turned on, collects Java call stack information for each Java application thread. The call stack information identifies the methods in the call chain. Information about each method includes the package (if any), class, method, signature (parameter types and return type), and the source line number being executed (if available).

The generated reports include a summary, CPU usage, service times, and wait times.

HFS data is also collected and reported, provided that the Java data extractor is selected.

8.3 IBM Debug Tool for z/OS

IBM Debug Tool for z/OS provides debugging capability for applications running in a variety of environments, such as IBM CICS, IBM IMS, IBM DB2 stored procedures, and IBM UNIX System Services. To meet the requirements of IBM DB2 Universal Database™ for z/OS, Versions 8 and 9, and IBM CICS Transaction Server for z/OS, Versions 3 and 4, Debug Tool also includes features to help you identify old OS/VS and VS COBOL II source code and to upgrade the code to IBM Enterprise COBOL.

To effectively build and service applications, you require robust, easy-to-use tools to compile, test, and debug them. IBM Debug Tool for z/OS software provides a complete solution that can help you to reduce application development cycle times.

8.3.1 Highlights

The main features of IBM Debug Tool for z/OS are that it:

- Provides a single debugging tool for batch, TSO, CICS, DB2, DB2 stored procedures, and IMS applications written in COBOL, PL/I, C/C++, Assembler and Java.
- Offers more productivity enhancements when integrated with CICS Explorer or Rational Developer for System z.
- Allows you to use RDz to provide a fully-integrated development, test, and debugging environment for all applications running on z/OS.
- Includes tools to identify OS/VS COBOL code and convert it to ANSI 85 standard.
- Supplies a code coverage to help you determine how thoroughly your code has been tested.
- Provides a CICS utility transaction (DTCN) that enables you to control debugging in the CICS environment.
- Provides an interactive interface with multiple windows that enable single-step debugging, dynamic patching, and breakpoint setting.
- Helps you debug an application while it runs in a host environment, such as a batch application, TSO, Interactive System Productivity Facility (ISPF), CICS, IMS, or DB2 Universal Database (including IBM DB2 stored procedures) environments.

Debug Tool can help you debug almost any application in almost any host environment, including COBOL, PL/I, C/C++ and Assembler applications running on z/OS systems. With Debug Tool, you can compile and link your COBOL, PL/I, C and C++, and Java programs, and assemble and link Assembler programs, as well as preprocessing and compiling your CICS and DB2 programs.

With Debug Tool, you can debug Enterprise COBOL and Enterprise PL/I applications that have been compiled with standard or full-optimization compiler options. You can also analyze your load modules to help you identify candidate OS/VS COBOL programs for conversion and then convert these OS/VS COBOL applications to Enterprise COBOL. These applications then can be compiled and debugged to extend the life of your existing code. Debug Tool software also provides coverage tools that enable you to conduct analysis on your test cases to determine how thoroughly they exercise your program code.

8.3.2 Support for IBM subsystems

Debug Tool works with individual load modules that are independent of the subsystem that they execute in.

8.4 IBM Fault Analyzer for z/OS

Fault Analyzer for z/OS provides the information that you require to determine the cause and assist with the resolution of application and subsystem failures. You can use this tool to assist in composite-application abend analysis. It helps you repair failures quickly by gathering information about an application and its environment at the time of failure.

8.4.1 Highlights

The main features of Fault Analyzer for z/OS are that it:

- Provides a detailed report about program failures to help you resolve them quickly.
- Includes a fault-history file that enables you to track and manage application failures and fault reports.
- Offers a view of storage contents, trace tables, and panel images at the time of failure to help speed corrective action.
- Provides the ability to customize message descriptions to be used in application-failure reports.
- Supports most z/OS subsystems and compilers.
- Provides integration with CICS Explorer.
- Provides integration with Rational Developer for System z.

When an application abend occurs, Fault Analyzer captures and analyzes real-time information about the application and its environment then generates an analysis report detailing the cause of the failure. The report describes the failure in terms of the application code, so you no longer lose time reviewing cumbersome, low-level system error messages. Fault Analyzer allows you to choose a report format to locate the information more easily.

Each application abend is recorded by Fault Analyzer in a fault-history file by job name, failure code, and other details, along with the analysis report and storage pages referenced during the analysis. This information can later be retrieved to reanalyze the failure.

Through the inclusion of information taken from selected manuals, Fault Analyzer can extract message and failure-code descriptions and insert them into the analysis report where applicable. You can also provide your own descriptions for messages.

You can write your own user exits. For example, you can write a user exit to access compiler listings that are compressed or available only through a proprietary access method.

Integration through a GUI interface allows application developers to work with fault entries directly from their development environment.

8.4.2 Support for IBM subsystems

In this section, we discuss topics in support of IBM subsystems.

CICS

Fault Analyzer functions are available from within the CICS transaction-based environment in a manner that is consistent with the ISPF usage, permitting you to review and analyze fault entries in history files without the necessity for a TSO logon.

Fault Analyzer provides general information about a CICS region along with detail of control blocks, transaction storage, the last panel buffer, the trace table, and an explanation of the CICS abend code.

The ability of Fault Analyzer to detect and analyze dump formatting and storage violations is particularly helpful for system programmers who have to debug CICS system problems. Options are provided to speed processing of duplicate dumps and to skip analysis of repeated abends.

DB2

Details of the last SQL activity and plan and package information are provided.

IMS

Fault Analyzer provides general information about an IMS region along with detail of the last DL/I call parameter list, information for all PCBs in the order of their relative PCB number, and, if available, JCB call trace information. IMS accounting information is also provided.

You can eliminate the overhead of duplicate IMS faults occurring across all MVS images in a sysplex.

WebSphere MQ

API information and return-code diagnostics are provided to assist in problem determination of WebSphere MQ applications.

8.4.3 Java support

When Fault Analyzer is invoked from within a Java application, abending traditional programs (COBOL, PL/I) can be analyzed.

A feature unique to the interactive component of Fault Analyzer is the ability to analyze information related to Java. The Java execution might be under WebSphere, CICS, or UNIX System Services on MVS. Typically, the environment is Java calling heritage programs. The User's Guide and Reference manual has a chapter that tells you how to perform these tasks:

- Set options for Java analysis
- Select a Java dump data set for analysis
- Display the resulting Java information in the interactive report
- Create a history file entry for the analyzed dump data set

8.4.4 Analysis options

Fault Analyzer provides three modes to help you better track and analyze application and subsystem failure information. Reports generated can be viewed through the SDSF or through the Fault Analyzer ISPF, CICS Explorer, or RDz interface.

Real-time analysis

When an application failure occurs, the Fault Analyzer exit starts real-time analysis. After failure processing, you can view the analysis report in your TSO session or in printed output.

Batch reanalysis

Batch reanalysis generates a new analysis report. This report is based on the dump and information gathered in real time, but with potentially different options specified or with compiler listings or side files made available. You can submit a Fault Analyzer batch-mode job using either the Fault Analyzer ISPF or your own job control language.

Interactive reanalysis

Interactive reanalysis runs under ISPF and CICS and enables you to navigate through a formatted, structured view of a fully-detailed reanalysis. This Fault Analyzer mode enables you to view working storage and control blocks at the time the dump was written. The interface has many point-and-click fields for easy navigation through the interactive reports.

8.5 IBM File Manager for z/OS

File Manager for z/OS offers comprehensive, user-friendly tools for working with z/OS data sets, CICS resources, DB2 data, and IMS data.

Extending the standard browse, edit, copy, and print utilities of ISPF, File Manager not only contains tools that support personnel will find useful, but also delivers enhancements that are designed to address the requirements of application developers working with files containing structured data. Also, because the basic features of the File Manager editor and the ISPF/PDF editor are deliberately almost identical, you can take advantage of extra features without having to relearn fundamental skills.

File Manager includes four components:

- A base component to manage z/OS data sets, such as queued sequential access method (QSAM), VSAM, partitioned data sets (PDS) and IBM z/OS UNIX System Services hierarchical file system (HFS) files
- A DB2 component to manage DB2 data
- An IMS component to manage IMS data
- A CICS component to manage data in CICS VSAM file resources, transient data queues, and temporary data queues

You only need to install those components that are relevant to your environment.

8.5.1 Highlights

The main features of File Manager for z/OS are that it:

- Includes a File Manager editor that maintains the same general look and feel across both view (read-only) and edit functionality in all of the key areas, whether they are traditional z/OS data sets, CICS resources, or data stored in a DB2 or IMS database.

- Provides the ability to define record and field properties in the same manner as your application programs, that is, by using your existing COBOL, PL/I, and HLASM copybooks stored in partitioned data sets or library-management systems.
- Enhances usability by enabling you to customize which fields to display, copy, or print.
- Enhances usability by providing the ability to define fields describing your data even when you do not have an existing copybook using dynamic templates.
- Supports development and production through extensive editing, browsing, and batch and interactive utility capabilities.
- Provides data scrambling support to provide for certain forms of data protection.
- Allows you to access WebSphere MQ queues on the local z/OS system where File manager is running; names of the List Managers and queues are displayed, and queues can be viewed and edited.
- Provides integration with Rational Developer for System z.
- Provides the ability to find, create, browse, copy, edit, print, format, or reformat data files in the most-popular z/OS file formats.

Not only can the same data mapping be applied as your applications use, but by the usage of File Managers templates, filtering can easily be applied to these operations based on the content of fields in your data. The template concepts of identification and selection criteria provide a simple to use, yet flexible interface for providing the simplest to the more complex of criteria.

8.5.2 File Manager base component

The File Manager base component helps speed the application-development process by identifying the structure of your records and displaying each field in a human readable format, according to its data type.

This component supports VSAM, including Innovation Access Method (IAM) files, QSAM, PDS, and WebSphere MQ queues. It also supports z/OS UNIX System Services hierarchical file system (HFS) data sets, including support for double byte character set (DBCS) data in these data sets.

You can edit entire files (regardless of size) and use a template or copybook for formatting and record selection during an edit or browse session.

8.5.3 Support for IBM subsystems

In this section, we discuss topics in support of IBM subsystems.

CICS

If you want to manage or query data in CICS VSAM file resources, transient data queues, and temporary data queues, this is provided for with the File Manager for CICS feature that access to CICS resources under a CICS transaction. The CICS resources supported for view, edit, and certain File Manager utilities (such as data create, copy and compare) are VSAM files, temporary storage queues, and transient data queues.

You can authorize users to view and change the status of supported local and remote CICS resources and perform File Manager base and File Manager IMS tasks from the File Manager CICS transaction.

The same display and record selection capabilities are present in the CICS environment as are in the BASE product, providing for quick easy access to CICS data. Also useful is the ability to create full copies of open CICS files (or even TS or TD queues) to another QSAM, VSAM, HFS, or even PDS file.

Another nice benefit of File Manager for CICS is that the majority of File Manager related processing happens offline to the CICS task, so it runs little risk of adversely affecting other CICS users, even when (for example) copying an entire file somewhere else.

File Manager for CICS even maintains the same look and set of editor commands that you might be used to in an ISPF environment. It is not even necessary to log off of your CICS environment to log on to your ISPF, if all you want to do is use the File Manager BASE or IMS components.

DB2

Whether you are a DB2 DBA, application programmer, or you just want to retrieve information stored in a DB2 database, the File Manager DB2 component provides something for you.

Database management abilities (such as create and drop objects, copying data within DB2, handling privileges, and import/export) and an editor for DB2 tables and views (in either read only or update mode), encompassing all the usual Insert, Delete, and Update functionality that is typically required in a database application.

Full tables, views, or the results of a limited SQL query that you customized yourself are provided, with the ability to save the query you made, in a File Manager template.

If you are writing SQL to be used in your applications, another handy File Manager usage is the ability to refine and test an SQL query by the prototyping and analysis feature, which includes an explanations feature.

Getting data out of, or back into, DB2 to or from QSAM or VSAM files is also provided for by a powerful utility that allows the export of DB2 data in a number of formats, including the external format used by the DB2 UNLOAD utility, the format used by the DB2 sample unload program DSNTIAUL, a tailorable File Manager export format with multiple options to control handling of NULL values and options for various column data types, and a delimited text format, such as comma-separated value output.

You can also generate batch JCL for the most commonly used DB2 utilities to save time.

IMS

With File Manager's IMS component, accessing data stored in an IMS database also becomes easy. Although the database storage concepts are different than DB2, File Manager still delivers a similar look when viewing or editing your data. Also, creating and saving customized queries is just as easy as using the other components.

Using record structures defined in COBOL or PL/I copybooks, just like the BASE component, the File Manager IMS component enables you to view or edit IMS segments displayed with formatting according to their individual field data types. You can find and change data across an entire database hierarchy or in specified segments and fields.

Data displays for IMS always include the segment name to help you identify where you are in the database (handy in the cases where you are viewing a large result set, or an entire hierarchy).

Database navigation commands are available with flexible parameters and scoping, to allow selective navigation or applicability, or a more generic approach, depending on your requirements.

You can use flexible criteria to select IMS database segments containing specific field values, and extract the segments into a sequential data set for later use. Or you can use the IMS component to load them into a different database.

WebSphere MQ

From within File Manager, you can access WebSphere MQ (WMQ) queues on the local z/OS system where File Manager is running. From the Primary Option Menu panel, select option 9, **WebSphere MQ**. File Manager then displays the WebSphere MQ Functions panel, from which you can do these tasks:

- List Managers and queues
- View a WMQ queue
- Edit a WMQ queue

8.5.4 Java support

File Manager can access an HFS file as a simulated QSAM/BSAM file. This means that at the I/O level, the file is seen as a single-volume, physical, sequential data set that resides on DASD.

8.5.5 SOA support

File Manager enables you to generate XML data from files. A File Manager template that describes the data-record layouts is required. The XML tags are generated based on the field names from the template, and the XML content comes from the data. A number of options are provided for the handling of invalid and unprintable data.

File Manager provides a plug-in for integration with Rational Developer for System z, allowing all aspects of Web-service (and traditional application) development to be undertaken from the same developer tool.

File Manager provides a scrambling algorithm that modifies data in a field while maintaining its system data type. Scrambling is intended to disguise personal information in several ways for various data types. The goal of repeatable scrambling is that application relationships based on equality tests can be maintained, if desired, even after the data is scrambled. A number of standard algorithms are provided to give the user complete control over the type of scrambling performed.

8.6 Workload Simulator

Workload Simulator provides the ability to simulate terminals and the associated messages. It also lets you alter message loads during a run. It can be used to generate a large volume of messages to evaluate the reliability and approximate performance characteristics of a network under expected operating conditions. Simply stated, anything that a real user can do at terminal, Workload Simulator can do faster, more reliably, and typically at less cost.

The features and benefits of IBM Workload Simulator are that it:

- Helps prepare your networks for peak transaction volumes.
- Enables testers to conduct reliable tests on stress, performance, and capacity as well as regression and function tests.
- Simulates different terminals, terminal features, and terminal operator actions.
- Provides support for enhanced TCP/IP support, SNA, and CPI-C (LU 6.2).
- Helps manage the test process.
- Offers several options for creating scripts to use in simulations.
- Provides panel images, data, and reports during simulation.
- Helps testers compute and analyze test results.

You can run Workload Simulator on MVS in batch mode, as a procedure, using the more user friendly Workload Simulator/ISPF Interface, or under TSO.

8.7 CICS Explorer

CICS Explorer, first introduced in 2009 as the new face of CICS, provides an integration point for CICS tooling with rich CICS views, data, and methods. Additionally, it provides:

- A common, intuitive, Eclipse-based environment for architects, developers, administrators, system programmers, and operators
- Task-oriented views provide integrated access to broad range of data and control capabilities.
- Powerful, context-sensitive resource editors
- An integration point for CICS TS, CICS Tools, CICS Transaction Gateway, the IBM Problem Determination Tools, and Rational Tools.
- Extension by Independent Software Vendors, System Integrators, and customers using the Java Software Development Kit.

The code is downloadable, at no cost, from the CICS Explorer website:

http://www-01.ibm.com/software/htp/cics/explorer/

Be aware that this code is refreshed approximately every three months.

8.7.1 Support for the IBM Problem Determination tools

The IBM Problem Determination Tools plug-ins, when used with CICS Explorer, provide easy access through a graphical user interface (GUI) on the workstation to the power of the host IBM Problem Determination Tools: Application Performance Analyzer for z/OS, Debug Tool for z/OS, and Fault Analyzer for z/OS. Plug-ins for each of these are available for download, from the following web site:

http://www-01.ibm.com/software/awdtools/deployment/pdtplugins/

Application Performance Analyzer

This plug-in encompasses both the Observation Request and Reporting functions, including the R02 screens list, detail views, edit functions and reports for the Observation. The plug-in GUI can be used for submitting new observation requests and for navigating the performance analysis reports generated from observation requests. It can also display and provide functions to multiple components of Application Performance Analyzer at the same time.

Here are the major views:

- STC View, which lists all active started tasks
- Observations List View, which lists all observations
- Observation Detail View, which provides details of an observation
- Reports List View, which lists all reports for an observation
- Report View, which displays an individual report

Debug tool

The plug-in GUI interface allows you to access debugging capability similar to that accessed by the ISPF interface, allowing you to:

- Set and clear breakpoints at a specific line
- Set and clear breakpoints for an error or warning-level error that is based on Language Environment severities
- Run to a breakpoint
- Step into a procedure
- Step over a procedure
- View variable values and change them as you step through the code

- View variable values in the context of a larger area of storage
- View the call stack

Fault Analyzer

The plug-in GUI interface provides access to problem reports for diagnosing mainframe application errors and ABENDs. Key features include these:

- An interface to manage views and multiple fault history files
- The ability to browse fault entries that were created during real-time analysis of abending programs
- A browser for browsing the dump storage associated with a fault entry
- A source listing of abending programs using side files

8.7.2 Support for CICS Tools

The following CICS Tools are supported through plug-ins to CICS Explorer:

- Configuration Manager
- Interdependency Analyzer
- Performance Analyzer

Details about these plug-ins is at the CICS Tools web site:

http://www-01.ibm.com/software/htp/cics/tools/

To download the plug-ins, go to:

http://www-01.ibm.com/software/htp/cics/explorer/

8.8 Rational Developer for System z

Rational Developer for System z (RDz) speeds the development of traditional mainframe, Web, and composite applications. Built on an Eclipse platform, it lends itself readily to integration with other products, both IBM and non-IBM.

8.8.1 Support for the IBM Problem Determination tools

In this section, we discuss the IBM Problem Determination (PD) tools.

Debug Tool

COBOL and PL/I applications, EGL applications, and Java applications can be debugged through a common interface using the local debugger. The remote debugger supports debugging of code that runs in the following z/OS environments:

- Batch
- TSO
- CICS
- IMS, both IMS Database Manager and IMS Transaction Manager, with or without Batch Terminal Simulator (BTS)
- DB2 (including stored procedures)
- WebSphere Application Server

The debugging sessions are cooperative, meaning that the remote distributed debugger resides on the workstation and interfaces with the IBM Debug Tool, running on the host with your application. The workstation interface communicates with the host z/OS products through TCP/IP.

Fault Analyzer

Fault Analyzer (FA) integration allows you to:

- Browse FA abend reports
- View dump selections relating to abends
- Annotate reports for future reference or to share comments with other users who browse the same reports

File Manager

File Manager (FM) integration enables access to:

- VSAM KSDS files for browsing and updating
- Template-driven display of VSAM, PDS members, and sequential file data

8.8.2 Remote compile generation

With remote editing, compiling, and debugging, you can develop or enhance many types of applications, including CICS, batch, TSO, or IMS Transaction Manager applications. These applications can access many forms of data such as DB2, VSAM, DL/I, and QSAM data. You can save time by editing, compiling, and debugging host applications remotely. When editing, compiling, and debugging host files from the workstation, you work in a cooperative Windows

and TCP/IP-based development environment, avoiding lengthy downloads and uploads unless explicitly desired.

8.9 IBM Problem Determination Tools in summary

IBM Problem Determination Tools include Application Performance Analyzer, Debug Tool, Fault Analyzer, and File Manager. These tools, along with CICS Explorer and Rational Developer for System z, are designed to help ease the burden of developing, testing, and supporting service-oriented and composite applications across complex IBM System z environments.

By helping to improve application delivery throughout the application life cycle, these tools provide increased user productivity and IT effectiveness across source code debugging, application-abend analysis, data management, and application-performance management.

System z tools, including the Problem Determination Tools, CICS tools, and application development tools, support the entire application life cycle to help you build, integrate, test, and manage enterprise solutions. As a result, you can make the most of your System z platform investments and expedite your move to SOA. With these tools, you can transform your applications and optimize your IT operations to achieve greater business flexibility, without impacting governance and compliance.

Part 4

CICS Application Development

In this part of the book, we cover all of the APIs for CICS. We provide easy-to-understand sample programs written in COBOL and PL/1 languages that are used in CICS application programming. We also show examples in Assembler language, which might be required for CICS systems programming. With the advent of Java technology, CICS opened itself to a new world. We provide easy-to-understand examples of how to use Java programs in CICS. The most current technology in programming languages is PHP and Groovy for dynamic scripting.

9

Application development

In this chapter, we describe which programming languages can be used in CICS. We provide easy-to-understand sample programs written in COBOL and PL/I languages, which are used in CICS application programming. We also show examples in Assembler language, which might be required for CICS systems programming. With the advent of Java technology, CICS opened itself to a new world. We provide easy-to-understand examples of how to use Java programs in CICS. The most current technology in programming languages is PHP and Groovy for dynamic scripting. We also provide an example for dynamic scripting.

9.1 Languages supported in CICS

In this section, we list all of the programming languages that are supported in CICS.

CICS is supporting the programming languages COBOL, PL/I, Assembler, C and C++ as well as Java, PHP and Groovy. Most application programmes still running in CICS are written in COBOL and PL/I. The development of new applications is still popular in these programming languages, while Java is also popular in CICS today.

Language products, such as COBOL and PL/I has been traditionally used as CICS programming languages. They have the advantage of being widely taught, ensuring that we have access to a pool of skilled programmers. After the turn of the century new COBOL programmers for instance were scarcely available.

Why use Java? Programmes written in Java tend to be platform independent. So this makes Java applications portable. Java also has built-in support for mulithreading, and with Java remote method invocation you have the ability to access objects on other machines as though they were local. In the beginning of the 21st century Java programmers were highly available in the market. That helped in the evolution of CICS collaboration with Java.

PHP and Groovy gives the world wide web access to CICS.

9.2 HelloWorld program

In this section, we look at the HelloWorld program running in a non-CICS environment and in a CICS environment.

9.2.1 A COBOL HelloWorld program in a non-CICS environment

Before we introduce COBOL in CICS, we provide a basic example in a non-CICS environment. The reason we use COBOL, PL/I, and Assembler, as the programming language for these examples is to show how we can access these programming languages from the web.

We show a basic COBOL HelloWorld program in a non-z/OS environment. We assume that you have experience on client platforms, such as Windows, and will get experience on z/OS. Therefore, we start with an example in an environment you are familiar with. Later on we show how this program works in z/OS batch and what is necessary to run this program in CICS.

Figure 9-1 shows the COBOL HelloWorld program.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLOWORLD.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
    DISPLAY '*** Hello world ***'.
    STOP RUN.
END PROGRAM HELLOWORLD.
```

Figure 9-1 HelloWorld program

We compiled this program on a Windows system and run it in a Windows command prompt box. Figure 9-2 shows the output.

```
COBOL/PLI

!cob2 HelloWorld.cbl
PP 5724-T07 IBM COBOL for Windows  7.6.0 in progress ...
End of compilation 1,  program HELLOWORLD,  no statements flagged.

!HelloWorld
*** Hello world ***

!
```

Figure 9-2 Run the HelloWorld program in a Windows command prompt box

9.2.2 The Cobol HelloWorld Program moved to z/OS batch

Figure 9-3 on page 284 shows the HelloWorld code embedded in Job Control Language (JCL). The nature of a batch job is that there is no screen involved. The message issued with the DISPLAY command goes to a SYSOUT DD statement.

Job Control Language (JCL): Job Control Language (JCL) is a scripting language used on IBM mainframe operating systems to tell the system how to run a batch job or start a subsystem.

```
//ITSO01A JOB (1159406),'FB31',MSGLEVEL=(1,1),
// NOTIFY=&SYSUID,MSGCLASS=A,CLASS=A
//PROCLIB JCLLIB ORDER=PP.COBOL390.V410.SIGYPROC
//STEP010 EXEC IGYWCLG,REGION=50M
//COBOL.STEPLIB DD DSN=PP.COBOL390.V410.SIGYCOMP
//COBOL.SYSIN DD *
       IDENTIFICATION DIVISION.
       PROGRAM-ID. HELLOWORLD.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       PROCEDURE DIVISION.
           DISPLAY '*** HELLO WORLD ***'.
           STOP RUN.
       END PROGRAM HELLOWORLD.
//GO.SYSOUT DD SYSOUT=*
```

Figure 9-3 COBOL HelloWorld program in a z/OS batch job

While the name HELLOWORLD works perfect on a non-z/OS system, on z/OS we have a restriction that the program names must not exceed 8 bytes in length. In the example shown in Figure 9-3, this rule really does not matter because it is a compile-link-and-go procedure. The compiler issues a warning that the program name has cut to HELLOWOR. The load module is stored in a temporary data set and loaded from there. There are no other modules in this temporary data set, so no name conflicts can occur. Later on in CICS, we do not ignore this warning and use only program names that are no longer than 8 bytes.

```
 Display  Filter  View  Print  Options  Help
 ------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY ITSO01A  JOB27175  DSID   107 LINE 0       COLUMNS 02- 81
 COMMAND INPUT ===>                                          SCROLL ===> CSR
********************************* TOP OF DATA *********************************
*** HELLO WORLD ***
******************************** BOTTOM OF DATA ********************************
```

Figure 9-4 SDSF Output written to DD-statement SYSOUT

9.2.3 What makes the HelloWorld Program a CICS program

What makes a program a CICS program? A CICS program is a program that uses the special CICS commands that start with EXEC CICS, and the object code created is link-edited differently than a batch program. The program must be defined to the CICS system, automatically by program autoinstall or manually by defining it to the system using a CICS supplied transaction, called CEDA. Also, a main program must be connected to a transaction code. To invoke a main program you must enter a transaction ID and not the program name.

Figure 9-5 shows what we changed to make the HelloWorld example a CICS program. In the first line, we added the compiler option CICS, which tells the compiler to involve the CICS translator for every EXEC CICS command to translate it into a language specific CALL statement. Next, the program name is shortened to not exceed 8 bytes in length. Then, the ENVIRONMENT DIVISION is completely removed. In this division, you define your input-output requirements. But as we said before, I/O is completely done by CICS. Therefore, this division is unnecessary in a CICS program. In the procedure division, you see that the COBOL DISPLAY statement is replaced by the EXEC CICS SEND command and the COBOL STOP RUN is changed to EXEC CICS RETURN.

With STOP RUN, we tell COBOL to return to the operating system. In CICS, we return to CICS using EXEC CICS RETURN. Good practice is to add a GOBACK after the EXEC CICS RETURN to meet the compiler's needs.

```
CBL CICS
IDENTIFICATION DIVISION.
PROGRAM-ID. HIWORLD.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  HELLOWORLD PIC X(19) VALUE IS '*** HELLO WORLD ***'.
PROCEDURE DIVISION.
    EXEC CICS SEND TEXT FROM(HELLOWORLD)
    END-EXEC
    EXEC CICS RETURN
    END-EXEC
    GOBACK.
END PROGRAM HIWORLD.
```

Figure 9-5 CICS HelloWorld program

9.3 What is a CICS application

An *application* is a collection of related programs that together perform a business operation, such as processing a product order or preparing a company payroll. CICS applications execute under CICS control using CICS services and interfaces to access programs and files.

CICS is a transaction processing system. It provides services for you to run applications online, by request, at the same time as many other users are submitting requests to run the same applications, using the same files and programs. CICS manages the sharing of resources, integrity of data, and prioritization of execution with fast response.

CICS applications are traditionally run by submitting a transaction request from a panel. Execution of the transaction consists of running one or more application programs that implement the required function. In CICS documentation, you can find CICS application programs sometimes called programs, and sometimes the term transaction is used to imply the processing done by the application programs.

9.3.1 The CICS transaction flow

To begin an online session with CICS, you usually begin by “signing on,” which is the process that identifies you to CICS. Signing on to CICS gives you the authority to invoke certain transactions. When signed on, you invoke the particular transaction that you intend to use. A CICS transaction is usually identified by a one to four-character transaction identifier or TRANSID, which is defined in a table that names the initial program to be used for processing the transaction.

Application programs are stored in a library on a direct access storage device (DASD) that is attached to the processor. They can be loaded when the system is started or simply loaded as required. If a program is in storage and is not being used, CICS can release the space for other purposes. When the program is next needed, CICS loads a fresh copy of it from the library.

In the time it takes to process one transaction, the system might receive messages from several terminals. For each message, CICS loads the application program (if it is not already loaded) and starts a task to execute it. Thus, multiple CICS tasks can run concurrently.

CICS maintains a separate thread of control for each task. When, for example, one task is waiting to read a disk file, or to get a response from a terminal, CICS can give control to another task. Tasks are managed by the CICS *task control* program.

CICS manages both multitasking and requests from the tasks themselves for services (of the operating system or of CICS itself). This process allows CICS processing to continue while a task is waiting for the operating system to complete a request on its behalf. Each transaction that is being managed by CICS is given control of the processor when that transaction has the highest priority of those that are ready to run.

While it runs, your application program requests various CICS facilities to handle message transmissions between it and the terminal and to handle any necessary file or database accesses. When the application is complete, CICS returns the terminal to a standby state. Figure 9-6, Figure 9-7 on page 288, and Figure 9-8 on page 288 help you understand occurs.

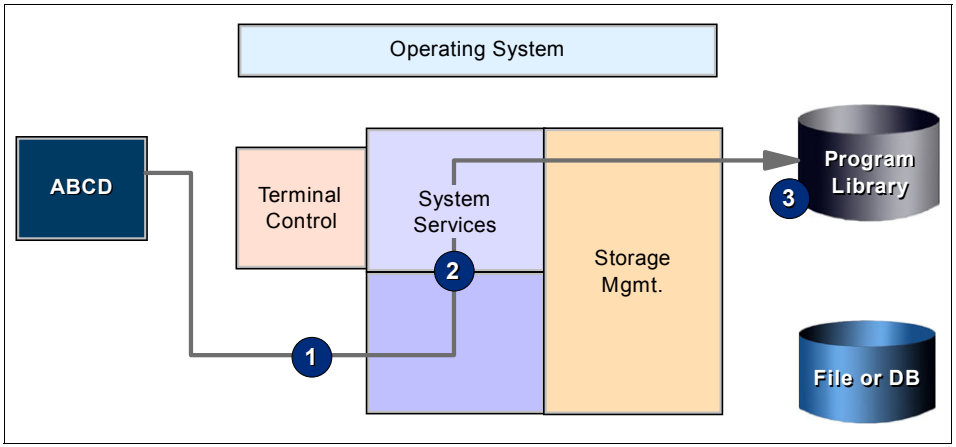


Figure 9-6 CICS transaction flow (Part 1)

The flow of control during a transaction (code `ABCD`) is shown by the sequence of numbers one to eight. (We only use this transaction to show some of the stages than can be involved.) The meanings of the eight stages are:

1. *Terminal control* accepts characters `ABCD`, typed at the terminal, and puts them in working storage.
2. *System services* interpret the transaction code `ABCD` as a call for an application program called `ABCD00`. If the terminal operator has authority to invoke this program, it is either found already in storage or loaded into storage.
3. Modules are brought from the *program library* into working storage.

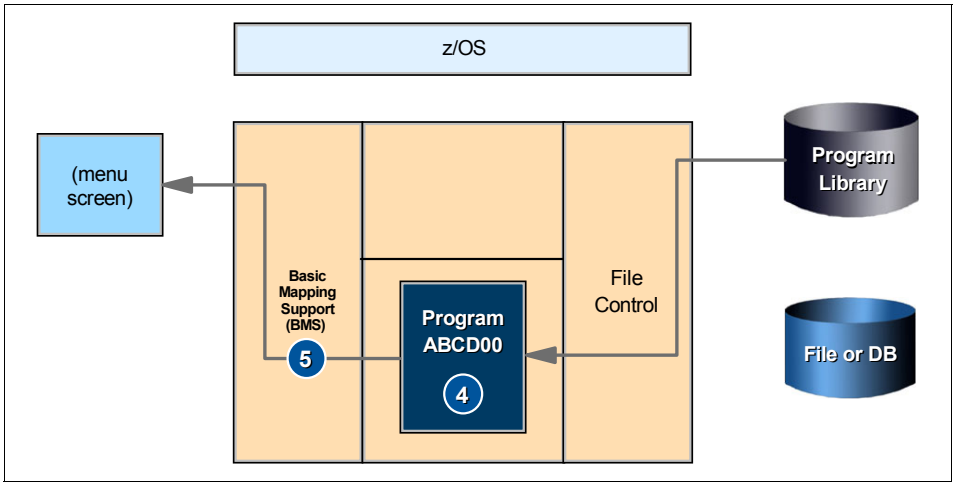


Figure 9-7 CICS transaction flow (Part 2)

4. A *task* is created. Program `ABCD00` is given control on its behalf.
5. `ABCD00` invokes *Basic mapping support* (BMS) and terminal control to send a menu to the terminal, allowing the user to specify precisely what information is needed, as shown in Figure 9-7.

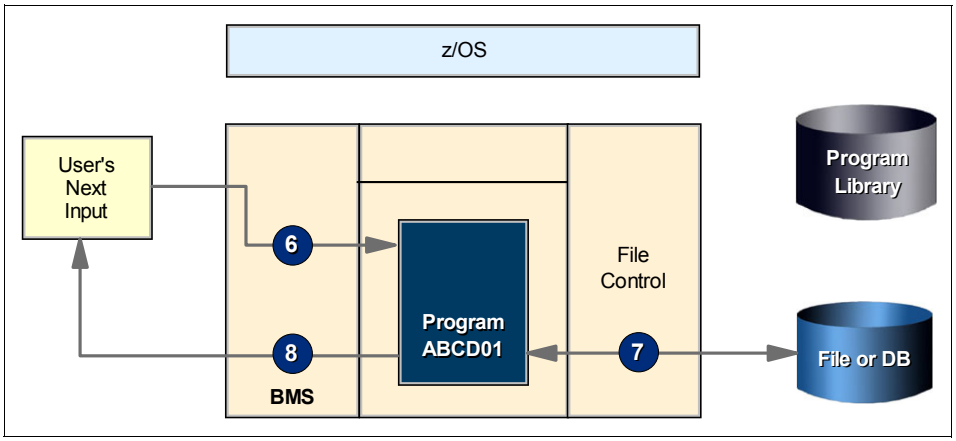


Figure 9-8 CICS transaction flow (Part 3)

6. BMS and terminal control also handle the user's next input, returning it to `ABCD01` (the program designated by `ABDC00` to handle the next response from the terminal), which then invokes file control.
7. *File control* reads the appropriate file for the invocation that the terminal user requested.

8. Finally, `ABCD01` invokes BMS and terminal control to format the retrieved data and present it on the terminal, as shown in Figure 9-8 on page 288.

9.3.2 Program control

A transaction (task) can execute several programs in the course of completing its work.

The program definition contains one entry for every program used by any application in the CICS system. Each entry holds, among other things, the language in which the program is written. The transaction definition has an entry for every transaction identifier in the system, and the important information kept about each transaction is the identifier and the name of the first program to be executed on behalf of the transaction.

In Figure 9-9 on page 290, you can see how these two sets of definitions, transaction, and program work in concert:

1. The user types in a transaction identifier at the terminal (or the previous transaction determined it).
2. CICS looks up this identifier in the list of installed transaction definitions, which tells CICS which program to invoke first.
3. CICS looks up this program in the list of installed transaction definitions, finds out where it is, and loads it (if it is not already in the main storage).
4. CICS builds the control blocks necessary for this particular combination of transaction and terminal using information from both sets of definitions. For programs in command-level COBOL, this includes making a private copy of working storage for this particular execution of the program.
5. CICS passes control to the program, which begins running using the control blocks for this terminal. This program can pass control to any other program in the list of installed program definitions, if necessary, in the course of completing the transaction.

There are two CICS commands for passing control from one program to another. One is the LINK command, which is similar to a CALL statement in COBOL. The other is the XCTL (transfer control) command, which has no COBOL counterpart. When one program links another, the first program stays in main storage. When the second (linked-to) program finishes and gives up control, the first program resumes at the point after the LINK. The linked-to program is considered to be operating at one logical level lower than the program that does the linking.

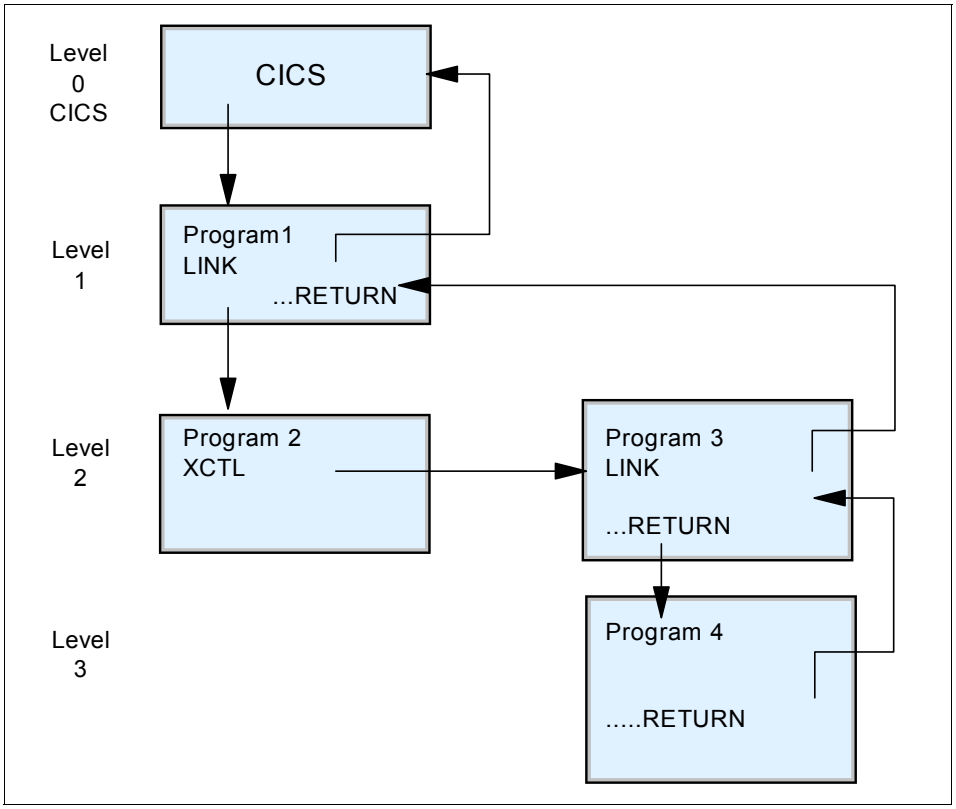


Figure 9-9 Transferring control between programs (normal returns)

In contrast, when one program transfers control to another, the first program is considered terminated and the second operates at the same level as the first. When the second program finishes, control is returned not to the first program, but to whatever program last issued a LINK command.

Some people like to think of CICS itself as the highest program level in this process, with the first program in the transaction as the next level down, and so on. Figure 9-9 illustrates this concept.

The LINK command looks like this:

```
EXEC CICS LINK PROGRAM(pgmname)
      COMMAREA(commarea) LENGTH(length) END-EXEC.
```

In the example, `pgmname` is the name of the program to which you want to link. `Commarea` is the name of the area containing the data to be passed and the area to which results are to be returned. The COMMAREA interface is also an option to invoke CICS programs.

A sound principle of CICS application design is to separate the presentation logic from the business logic. Communication between the programs is achieved by using the LINK command, and data is passed between such programs in the COMMAREA. Such a modular design provides not only a separation of functions, but also much greater flexibility for the web enablement of existing applications using new presentation methods.

9.4 Introduction to the CICS Application Programming Interface

All CICS commands start with EXEC CICS. They provide special functions not available in native languages, or they replace existing native functions. Why?

CICS is a system between your application and the actual operating system. Sometimes CICS is also called middleware. So instead of talking directly to the operating system, as you do in a batch program, you talk to CICS. The reason is that CICS manages not only a single user only, but a lot of users all with different requests in parallel. A batch job reading a file, for instance, issues a READ command and is waiting for the data. In CICS, we do not have time to wait for a data request for each single task, which delays all other tasks running in parallel. In a CICS environment, you ask CICS to get the data with an EXEC CICS READ. But CICS now does not wait until the data is available from the file system. Instead, it puts the requesting task in a wait and serves other tasks in the meantime. So CICS can serve many requests simultaneously.

The CICS API is rich. All commands with their options are described in the CICS Application Programming Reference manual. The functions they replace are listed in the CICS Application Programming Guide manual.

For each programming language, there is a chapter with restrictions and requirements. The functions and statements for input and output processing that must not be used are: OPEN, CLOSE, READ, WRITE, REWRITE, and DELETE. For these commands, you must use the corresponding CICS equivalents, for example, EXEC CICS WRITE.

Another example is the interaction with the operator. The DISPLAY command is replaced by various EXEC CICS SEND commands. In the example programs, we introduce two types of EXEC CICS SEND: send plain text and send formatted text using Basic Mapping Support (BMS), which we explain in chapter 9.10, “Basic Mapping Support” on page 300.

9.5 Compiling and link-editing a CICS program

In this section, we describe how to compile and link-edit a CICS program.

9.5.1 Compiling a CICS program

A CICS program contains the EXEC CICS command. The language compilers are unaware of that command. Therefore a kind of pre-processor is necessary to translate each EXEC CICS commands into a language-specific CALL statement. A language translator reads your source program and creates a new one. Language statements remain unchanged, but CICS commands are translated into CALL statements of the form required by the language in which you code. The CALL invokes a CICS-provided "EXEC" interface module or stub, which is a function-dependent piece of code that the CICS high-level programming interface uses. Figure 9-10 shows all of the steps that are necessary to translate source code into a executable load module.

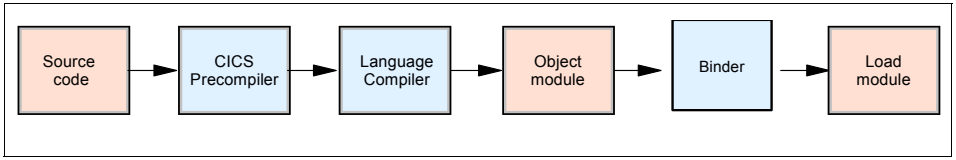


Figure 9-10 From source code to a load module

Rather than using a translator (translator = precompiler) in a separate step to translate all EXEC CICS commands before compiling the source, you can invoke the translator dynamically during the compile process. This feature is called the *integrated translator*. When the compiler detects an EXEC CICS command, it invokes and presents the translator the EXEC CICS command and gets back the corresponding language CALL statement. The EXEC CICS command is changed into a comment. Then the compiler can continue compiling the source. The integrated translator is activated with a compiler option, for example, CBL CICS in COBOL or *PROCESS SYSTEM(CICS) in PL/I. The CICS SDFHLOAD library must be concatenated under STEPLIB DD-statement that the translator module can be found by the language compiler.

Example 9-1 on page 293 shows the output by the translator when processing a simple EXEC CICS RETURN command.

Example 9-1 Translator output

```
*EXEC CICS RETURN END-EXEC
Call 'DFHEI1' using by content x'0e0800000600001000'
end-call.
```

The reference to DFHEI1 is resolved by the inclusion of the DFHELII sub routine in the linkage editor step. See 9.5.2, “Link-editing a CICS program” on page 293.

Integrated translators are available for COBOL, PL/I, and C (C++). The advantage of using the integrated CICS translator is that you can translate and compile your source code in a single step.

CICS supplies procedures, such as DFHYITVL or DFHZITCL, to compile CICS programs that are written in COBOL. You can find this procedures in the SDFHPROC library.

Note: CICS supplies material separated into different libraries. JCL programs to compile your programs are in SDFHPROC. Sample programs are in SDFHSAMP. Language-related sample programs are in SDFHPL1, SDFHCOB, or SDFHC370. Assembler macros are available in SDFHMAC. The load modules to run CICS are in SDFHLOAD and SDFHAUTH. Try to obtain the naming conventions on your system, and browse the SDFHPROC to find the compile procedures for the language of choice.

9.5.2 Link-editing a CICS program

All High-Level language (HLL) compilers produce a program listing all the information that it generated during the compilation. Additionally, another file is created that contains the object code. This newly created file is input for another program, the linkage-editor or binder.

During the bind process all external references (for example, calls to sub-routines and service modules) are resolved. The binder uses information in the object deck to combine them into a load module. A load module is an executable program stored in a Partitioned Data Set (PDS) program library. At program fetch time, the load module produced by the binder is loaded into virtual storage. When the program is loaded, it then can be run.

The binder can also add additional external object code and so-called stubs to the load module. A *stub* is a small piece of code to find the right CICS service module or enable your application program to access non-CICS resource managers. The CICS-related stubs are provided in the SDFHLOAD library. A stub must be link-edited with your application program to provide communication

between your code and the CICS EXEC interface program, DFHEIP. These stubs are invoked during execution of EXEC CICS commands. CICS provides stubs for all supported programming languages.

The CICS-supplied stub routines work with an internal programming interface. This is the CICS command-level interface, which is never changed in an incompatible way. Consequently, these stub modules are upward and downward compatible, and CICS application modules never need to be re-linked to include a later level of any of these stubs.

There are stubs for each programming language. COBOL, PL/I, and C use the stub DFHELII. This stub has several entry points. COBOL uses entry point DFHEI1. PL/I uses entry point DFHEI0, and C uses DFHEXEC. Assemblers use the stub DFHEAI.

Figure 9-11 shows the parameters we used to link-edit the sample program. There are two parameters not mentioned yet: AMODE and RMODE. With AMODE, we specify that the program is running in 31-bit mode. It can address all virtual storage addresses up to 2**31. RMODE specifies that it can be loaded at any place in a 31-bit address space. To explain the difference: If you specify, for instance, RMODE(24) in combination with AMODE(31) that means that the program is loaded in an area up to address 2 ** 24. We call this “below the line”. But, it can access all data even above the line because of AMODE(31).

```
MODE AMODE(31)
MODE RMODE(ANY)
INCLUDE SYSLIB(DFHELII)
INCLUDE OBJECTS(HIWORLD)
ORDER DFHELII
ENTRY HIWORLD
NAME HIWORLD(R)
```

Figure 9-11 Linking the CICS HelloWorld program

Future prospects: At the time of writing this book, CICS is already 64-bit aware, which means that the CICS domain architecture is partially enabled to run in and exploit the 64-bit addressing mode. CICS applications are not. In longer terms, the aim is to support AMODE 64 applications too.

All high-level languages supported by CICS, namely COBOL, PL/I, C and C++, depend on Language Environment. When modifying existing application programs, or writing new programs, you must use a compiler supported by Language Environment. This requires that your application programs must be link-edited using the Language Environment SCEELKED library, and this in turn

means that the resulting application load module can execute only under Language Environment.

9.6 Virtual storage

The virtual storage concept presents to the user a logical space for data storage and handles the process of mapping it to the actual physical location. The mapping between virtual storage and real storage is called Dynamic Address Translation (DAT).

We use the following real-life example to explain what virtual storage is. Imagine an exclusive clothes shop in the busiest street in town. The rental fee is high, but the shop window and the sales room is small and cannot house all the clothes. In the sales room, there is a counter with a book on it. Behind the counter there is a door to a huge back store where all clothes that cannot be in the shop window or shop room are stored. In the book on the counter, the retail seller writes down at which place in the back office the clothes are. And sometimes the seller moves clothes from the shop window to the back store and from the back store to the shop window, always updating the book to keep track of where the clothes are.

The shop window and the sales room are small and expensive and represents the real storage in a computer. The back store is huge and cheap. This is the auxiliary storage in a computer system. The virtual storage is actually the book where everything is listed.

All HLL computer programs only work with virtual addresses. The address translation is done by the computer automatically. So, if the retailer has a customer in the sales room who wants to buy a piece of clothing, which is currently not in the sales room, the retailer must bring the clothing from the back store into the sales room. If a computer program accesses data at a specific virtual address, and the data at this address is not in real storage, the computer must access the auxiliary storage and move the data into the real storage before it can be processed by the program. This process is called *page-in*. The process of moving data from real storage to auxiliary storage is called *page-out*.

9.7 Load module storage

The linkage-editor, today also called the binder, stores a load module onto disk. The data set format for containing load modules is a partitioned data set (PDS) or a partitioned data set extended (PDS/E) on direct accessible storage (DASD). However, both data set types can be used to hold load modules. Library is a synonym for PDS and PDS/E. PDSs and PDSEs are divided into partitions called

members. A member is simply a sequential data set stored in a library. The names and addresses of all sequential data set (members) that are stored in a library are listed in a directory at the beginning of the data set. A load library contains programs that are ready to be executed.

CICS searches for load modules in libraries that are concatenated under a DD-statement DFHRPL by default. You can find this DD-statement in the CICS start-up Job Control Language (JCL). In addition, you can add libraries dynamically using CEDA, a CICS-supplied transaction. Dynamic load libraries are not visible in JCL.

9.8 Defining resources to CICS

Before you can run a program in CICS you must define at least a transaction to CICS. A transaction is a basic resource in CICS. CICS is sometimes called a transaction manager. Additionally, you must define other resources, such as programs, files, libraries, and so on. However, some definitions can be avoided using autoinstall functions. But in this case, you must provide a program to supply the autoinstall capability. There are sample programs available for program autoinstall, terminal autoinstall, and so on. At a starting point, to run a HelloWorld program, we need a transaction definition and a program definition.

The transaction name is up to you. The restriction is not to use transaction codes starting with the character C. All transactions starting with C are reserved to CICS itself. Examples are CEDA, CEMT, CECI, and a lot more. Refer to the IBM book *CICS Supplied Transactions*, SC34-7004 for further information about supplied transactions.

CICS Explorer is a graphical tool widely used and well described in Chapter 6, “CICS Explorer” on page 205 in this book, and for a complete description of CICS Explorer and how to use it, refer to the IBM Redbooks publication *IBM CICS Explorer*, SG24-7778-01. With this tool you can easily define resources to your CICS system in the CSD file or to CICSPlex SM data repositories.

CEDA is a transaction that you can enter on a 3270 panel (character-based green screen) to define all the resources you require to a CICS System Definition File (CSD).

The CICS system definition file is a VSAM data set containing a resource definition record for every resource defined to CICS by means of CICS Explorer, CEDA, or DFHCSDUP. DFHCSDUP is a batch utility, providing the same functions as CICS Explorer and CEDA.

After defining the resources the system, they must be installed in a running system using the CICS Explorer or CEDA install function. Resources can also be installed automatically at startup.

Single resources must be in a group. Groups are combined into lists. At CICS startup, you can specify the list names to install all groups in the list.

Group A is a collection of related resources on the CSD. Each resource that you define must belong to a group. You cannot define a resource without naming the group.

List the names of groups that CICS installs at an initial or cold start. You can add groups to lists if you want them installed at an initial or cold start, or if it helps you to manage your groups better. Groups do not have to belong to lists and can be defined independently.

You can define most resources to CICS using several methods, such as CICS Explorer, CICSPlex SM Business Application Services and Resource definition online.

CICS Explorer

You can use the CICS Explorer to define, install, and manage resources. If CICS Explorer is connected to a CICS system, definitions are stored in the CICS system definition file and are installed into an active CICS system from the CSD file. If CICS Explorer is connected to CICSPlex SM, definitions are stored in the CICSPlex SM data repository and can be installed either automatically, during CICS initialization, or dynamically, into a running CICS system.

CICSPlex SM Business Application Services

You can use CICSPlex SM Business Application Services (BAS) to define and manage resources. Definitions are stored in the CICSPlex SM data repository and can be installed either automatically, during CICS initialization, or dynamically into a running CICS system.

Resource definition online (RDO)

This method uses the CICS-supplied online transactions CEDA, CEDB, and CEDC. Definitions are stored in the CSD file and are installed into an active CICS system from the CSD file.

9.8.1 Getting started with CEDA

You can access CEDA from a CICS terminal.

To access CEDA:

1. Enter CEDA at a CICS terminal. The cursor is indicated by the symbol '_'.

   A list of all the CEDA commands is displayed. The CEDB transaction does not support the INSTALL command. The CEDC transaction supports only the DISPLAY, EXPAND, and VIEW commands.

Creating a resource definition

You can use the CEDA transaction to create a new resource definition.

To create a map set definition:

1. Type `CEDA` to start the CEDA function.
2. To create a new resource definition, type `DEFINE`.

Figure 9-12 shows a CEDA DEFINE panel. This panel lists all the resource types that you can define for CICS using CEDA.

```
 DEF
 ENTER ONE OF THE FOLLOWING

Atomservice  MQconn       Webservice
Bundle       PARTItionset
CONnection   PARTNer
CORbaserver  PIpeline
DB2Conn      PROCesstype
DB2Entry     PROFile
DB2Tran      PROGram
DJar         Requestmodel
DOctemplate  Sessions
Enqmodel     TCpipservice
File         TDqueue
Ipconn       TErminal
JOurnalmodel TRANClass
JVmserver    TRANSaction
LIbrary      TSmodel
LSrpool      TYpeterm
MApset       Urimap

                                                SYSID=RED1 APPLID=EPRED1

PF 1 HELP       3 END             6 CRSR             9 MSG             12 CNCL
```

Figure 9-12 Ceda DEFINE panel

To create a transaction definition, as shown in Figure 9-13 on page 299:

1. Clear the panel, and start the CEDA transaction.

2. Create a transaction definition. You can type the whole command on the CEDA initial panel. Type:

```
DEFINE TRANSACTION(TEST) PROGRAM(MENU) GROUP(EP)
```

```
 OVERTYPE TO MODIFY                                        CICS RELEASE = 8810
  CEDA  DEFine TRANSaction( TEST )
   TRANSaction    : TEST
   Group          : EP
   DEScription  ==> MENU
   PROGram      ==> MENU
   TWasize      ==> 00000              0-32767
   PROFile      ==> DFHCICST
   PArtitionset ==>
   STAtus       ==> Enabled            Enabled | Disabled
   PRIMedsize     : 00000              0-65520
   TASKDATALoc  ==> Below              Below | Any
   TASKDATAKey  ==> User               User | Cics
   STOrageclear ==> No                 No | Yes
   RUnaway      ==> System             System | 0 | 500-2700000
   SHutdown     ==> Disabled           Disabled | Enabled
   ISolate      ==> Yes                Yes | No
   Brexit       ==>
+ REMOTE ATTRIBUTES

                                                 SYSID=RED1 APPLID=EPRED1
  DEFINE SUCCESSFUL                            TIME: 15.07.32  DATE: 06/21/11
PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 9-13 Define transaction to CICS using CEDA

You must specify a PROGRAM whenever you define a new TRANSACTION. If you do not, CEDA displays a severe error message saying that you must specify either a PROGRAM or REMOTESYSTEM.

You now have a new resource to work with: a TRANSACTION in group EP.

For more details about using CEDA, see *CICS manual Resource Definition Guide*, SC34-7181-00, which walks you through all of the available options.

9.9 Running the HelloWorld program in CICS

Let us summarize what we have so far:

- We wrote a CICS program using CICS API to display Hello World on a CICS panel.

- We described how to compile the program and link-edit it with the right CICS stub onto a library.
- We defined a transaction ID and the program name to CICS using CICS Explorer, CEDA, or DFHCSDUP.

What we intend to do is run the transaction on a CICS panel. Before you can run a transaction in CICS, you must logon to CICS, clear the screen with the CLEAR key (default clear key is Pause), and enter the 1- to 4-byte long transaction code into the left upper corner area of the panel, and press Enter.

9.10 Basic Mapping Support

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

The EXEC CICS SEND TEXT command used in the CICS HelloWorld example sends unmapped plain data to the panel. Later on, we introduce a more sophisticated but still easy-to-understand sample program. Therefore we want to introduce the concept of mapping the data on a CICS screen. Basic Mapping Support (BMS) is the CICS solution of how to manage the contents that a terminal screen displays.

A screen on a text-based terminal consists on a matrix of character. The standard CICS screen layout is 24 rows each and 80 columns. The entity used to describe this matrix is called a *map*. The map makes it possible for a program to associate meaningful names with fields of data on the screen so that program commands can refer to pieces of data by name. Maps send to and receive data from a screen.

A BMS map is a collection of assembler macros. It sounds complicated because it involves assembler language concepts, but it is not. We give a small example, sending the string Enter your name: on a specific position on the screen. And we receive at another position of the screen the answer from the operator. The target of the program is to send back Hello concatenated with the previously entered name.

Figure 9-14 on page 301 shows a small mapset definition. A *mapset* can be a collection of several maps. There is one map defined in this mapset. The map name is SCREEN1. Note the 4 X on the right hand side. They are in column 72 and are the continuation signs, which indicates that this line is continued on the next line.

```
MYMAPS  DFHMSD TYPE=MAP,MODE=INOUT,LANG=COBOL,                          X
               MAPATTS=(COLOR,HILIGHT,OUTLINE),                         X
               DSATTS=(COLOR,HILIGHT),STORAGE=AUTO
SCREEN1 DFHMDI SIZE=(24,80)
        DFHMDF POS=(10,20),LENGTH=16,INITIAL=
NAMEIN  DFHMDF POS=(10,37),LENGTH=20,COLOR=GREEN,OUTLINE=BOX,           X
               ATTRB=(UNPROT,BRT,IC),HILIGHT=REVERSE,CASE=MIXED,        X
               INITIAL=
        DFHMDF POS=(10,58),LENGTH=1,ATTRB=PROT
SAYHI   DFHMDF POS=(15,30),LENGTH=40,COLOR=BLUE
        DFHMSD TYPE=FINAL
        END
```

Figure 9-14 Map definition

This small mapset definition uses three macros: DFHMSD, DFHMDI and DFHMDF, as shown in Figure 9-14.

DFHMSD is the mapset definition macro. It configures basic allocations, for example, what is the programming language using this map (to generate the appropriate copybook)? Is the map used for input, output, both, or in out?

DFHMDI tells BMS the name of the map and the screen size in rows and columns. We named the map SCREEN1. This is the identifier we use in the SEND MAP commands. It is 24 lines long and 80 columns wide.

DFHMDF defines each field on the screen. In it, you indicate, for instance, the position of the field on the screen, the length of the field, the color and other attributes of the field (highlighting, used as a box). Furthermore you can use the initial option to assign an initial value to the field.

SCREEN1 is the map sending out a screen, prompting the operator to enter a name.

Find in Figure 9-15 on page 302 the CICS screen send for map SCREEN1.

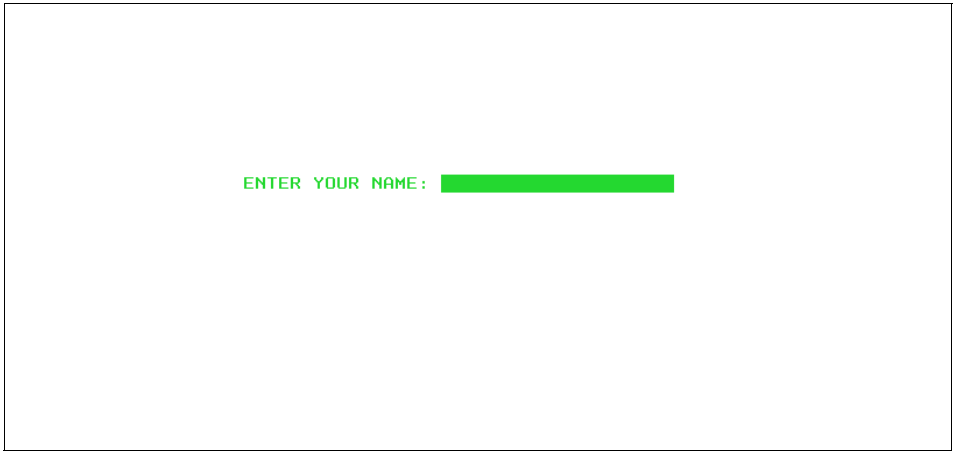


Figure 9-15 CICS screen SCREEN1

BMS has been used for over 40 years and is still used today. We currently encourage the use of modern interfaces, such as web and graphical user interfaces.

9.11 CECI for interactive testing

Transaction CECI invokes the CICS command-level interpreter. It allows programmers to try out individual CICS programming commands from a display panel. You can select any command keyword from a menu and add operands and data values to it, building a complete command before executing. Start CECI followed by the name of the command you want to test. When developing maps, CECI is helpful. After you assemble your mapset and define it to CICS using CEDA, you can immediately test the map to see the actual results on the panel without having a program ready to send the map. To test SCREEN1 in mapset MYMAPS, enter the string provided in Example 9-2. Figure 9-15 shows the output of this command.

Example 9-2 Interactively test a map using CECI

```
ceci send map(screen1) mapset(mymaps) freek erase maponly
```

Figure 9-16 shows you the menu screen, which is what you get when you simply enter `CECI`. If you do not have the entire command ready to enter in one string, enter `CECI`, select **SEND**, and you will get the next panel. Select MAP, and so on. In this way, you can build the whole command string step-by-step.

```
 STATUS:

ABend        DEFine       FREE         POP          RETUrn       SUspend
ACquire      DELAy        FREEMain     POSt         REWInd       SYncpoint
ADD          DELETE       GDs          PURge        REWRite      TEst
ADDRess      DELETEQ      GET          PUSh         ROute        TRACe
ALlocate     DEQ          GETMain      PUT          RUn          TRANsform
ASKtime      DISAble      GETNext      Query        SENd         UNlock
ASSign       DISCard      Handle       READ         SET          UPdate
BIf          DOcument     IGnore       READNext     SIGNAl       Verify
BUild        DUmp         INQuire      READPrev     SIGNOFf      WAIT
CAncel       ENAble       INVoke       READQ        SIGNON       WAITCics
CHAnge       ENDBR        ISsue        RECeive      SOapfault    WEb
CHEck        ENDBROwse    Journal      RELease      SPOOLClose   WRITE
COLlect      ENQ          LInk         REMove       SPOOLOpen    WRITEQ
CONNect      ENTer        LOad         RESET        SPOOLRead    WSAContext
CONVERSe     EXtract      MONitor      RESETBr      SPOOLWrite   WSAEpr
CONVERTtime  FEpi         MOVe         RESUme       START        Xctl
CReate       FORCe        PErform      RESYnc       STARTBR
CSd          FORMattime   POInt        RETRieve     STARTBROwse


PF 1 HELP 2 HEX 3 END 4 EIB 5 VAR 6 USER              9 MSG
```

Figure 9-16 All the command which can be tested with CECI

9.12 CICS programming styles

In this section, we look at both pseudo conversation and conversational programming styles in CICS.

9.12.1 CICS conversational programming style

CICS programs often go into a conversation with the operator. To demonstrate this we introduce an example program that asks you, the operator, to enter your name. CICS replies with Hello yourname. The map is already developed and tested in a previous chapter. In this chapter, we provide the COBOL sample program using the maps. The mapset assembly does not only create a load module in a library, but also a language-specific copybook that must be included in the sample COBOL program.

Figure 9-17 on page 304 shows an example of a conversational transaction.

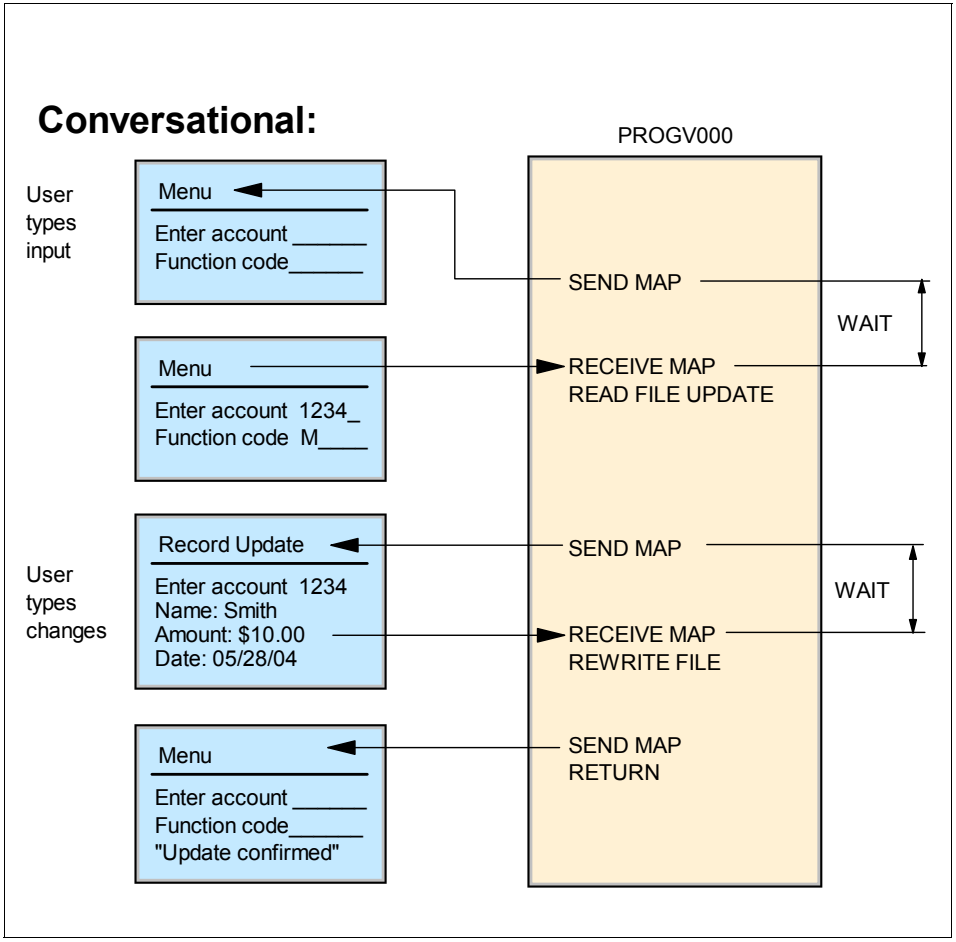


Figure 9-17 Example of a conversational transaction

In a conversational transaction, programs hold resources while waiting to receive data. In a pseudo-conversational model, no resources are held during these waits.

In Figure 9-18 on page 305, you can review the generated *copybook*. A copybook is the language structure created when you assemble the map. Notice that the variables now have suffixes, for example, NAMEINI for input or SAYHIO for output.

```
01   SCREEN1I.
     02   FILLER PIC X(12).
     02   NAMEINL     COMP   PIC   S9(4).
     02   NAMEINF     PICTURE X.
     02   FILLER REDEFINES NAMEINF.
       03 NAMEINA     PICTURE X.
     02   FILLER    PICTURE X(2).
     02   NAMEINI   PIC X(20).
     02   FILLER PIC X.
     02   SAYHIL     COMP   PIC   S9(4).
     02   SAYHIF     PICTURE X.
     02   FILLER REDEFINES SAYHIF.
       03 SAYHIA     PICTURE X.
     02   FILLER    PICTURE X(2).
     02   SAYHII   PIC X(40).
01   SCREEN1O REDEFINES SCREEN1I.
     02   FILLER PIC X(12).
     02   FILLER PICTURE X(3).
     02   NAMEINC     PICTURE X.
     02   NAMEINH     PICTURE X.
     02   NAMEINO   PIC X(20).
     02   FILLER PIC X.
     02   FILLER PICTURE X(3).
     02   SAYHIC     PICTURE X.
     02   SAYHIH     PICTURE X.
     02   SAYHIO   PIC X(40).
```

Figure 9-18 COBOL symbolic map generated during mapset assembly

Figure 9-19 on page 306 is a COBOL sample program that includes a copybook with a COBOL COPY statement (COPY MYCOBMAP) and uses the appropriate CICS API command to send the maps to the CICS panel. A copybook is the language structure created when you assemble the map.

The first part of the structure, under the label SCREEN1I, is new. This is the symbolic input map—the structure required for reading data from a screen formatted with map SCREEN1. For each named field in the map, it contains three subfields. As in the symbolic output map, each subfield has the same name as the map field, suffixed by a letter that indicates its purpose. The suffixes and subfields related to input are:

- L = The length of the input in the map field.
- F = The flag byte, which indicates whether the operator erased the field and whether the cursor was left there.
- I = The input data itself.

The input and output structures are defined so that they overlay one another, field-by-field. That is, the input (I) subfield for a given map field always occupies the same storage as the corresponding output (O) subfield. Similarly, the input

flag (F) subfield overlays the output attributes (A) subfield. For implementation reasons, the order of the subfield definitions vary somewhat among languages. In COBOL, the definition of the A subfield moves to the input structure in an INOUT map, but it still applies to output, just as it does in an output-only map. In assembler, the input and output subfield definitions are interleaved for each map field.

BMS uses dummy fields to leave space in one part of the structure for subfields that do not occur in the other part. For example, there is always a 2-byte filler in the output map to correspond to the length (L) subfield in the input map, even in output-only maps. If there are output subfields for extended attributes, such as color or highlighting, BMS generates dummy fields in the input map to match them.

The correspondence of fields in the input and output map structures is convenient for processes where you use a map for input and then write back in the same format, as you do in data entry transactions or when you get erroneous input and must request a correction from the operator.

```
CBL CICS
IDENTIFICATION DIVISION.
PROGRAM-ID. HIWORLD1.
DATA DIVISION.
WORKING-STORAGE SECTION.
    COPY MYCOBMAP.
PROCEDURE DIVISION.
    EXEC CICS SEND MAP('SCREEN1') MAPSET('MYMAPS') MAPONLY
              FREEKB ERASE END-EXEC
    EXEC CICS RECEIVE MAP('SCREEN1') MAPSET('MYMAPS')
              END-EXEC
    STRING '*** HELLO ' DELIMITED SIZE
            NAMEINI     DELIMITED SPACE
            ' ***'      DELIMITED SIZE
            INTO SAYHIO END-STRING
    EXEC CICS SEND MAP('SCREEN1') MAPSET('MYMAPS') DATAONLY
              FREEKB END-EXEC
    EXEC CICS RETURN END-EXEC
    GOBACK.
END PROGRAM HIWORLD1.
```

Figure 9-19 Conversational COBOL program sending and receiving a map

Figure 9-20 on page 307 shows the CICS panel after entering Paul.

Figure 9-20 Program replies the name entered

Avoid the conversational transaction programming style because the time needed for a response from a terminal user is longer than the time needed for the computer to process the input. Conversational transactions are much longer than non-conversational transactions. Conversational transactions allocate storage, maintain locks, and hold other resources for perhaps a long period of time. When one of these resources are critical, you have a strong reason for using the pseudo-conversational transaction style, which we introduce in section 9.13, “CICS pseudo conversation programming style” on page 308.

9.12.2 CEMT: The master terminal transaction

After the SCREEN1 is sent out, the program waits on input. You can use the CEMT transaction to view which programs are currently running in your system.

In Figure 9-21 on page 308, we used CEMT INQ TASK to show all currently running tasks. Notice that the task at the top is HIWO. This task invokes the CICS HelloWorld program. The task is waiting on operator input, that is ZCIOWAIT. This is a classic wait for a conversational task. Conversation programming is not the recommended programming style in CICS because all resources used by a conversational program are not available for other tasks (for example, storage, files, and so on.

In the next paragraph, we show how to migrate this sample program from a conversational style into a pseudo-conversational style.

```
I TAS
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Tas(0000157) Tra(HIWO) Fac($AAA) Sus Ter Pri( 001 )
    Sta(TO) Use(CICSUSER) Uow(C77D3AD2D7D6D5C4) Hty(ZCIOWAIT)
 Tas(0000158) Tra(CEMT) Fac($AAB) Run Ter Pri( 255 )
    Sta(TO) Use(CICSUSER) Uow(C77D3AD8D13C935C)

                                                SYSID=CICS APPLID=IYCKZCHG
  RESPONSE: NORMAL                            TIME: 15.25.54  DATE: 03/18/11
PF 1 HELP       3 END       5 VAR       7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

Figure 9-21 CEMT INQ TASK show conversational task HIWO

9.13 CICS pseudo conversation programming style

The ideal CICS application is short running and concise. Figure 9-21 is the opposite of that. When it enters the system it stays there for a long time, or until the operator presses Enter. The CICS transaction terminates and disappears from the system.

CICS offers additional facilities that allow programmers to perform what is known as pseudo-conversational programming. With this technique, a terminal user is thinking to be in a permanent conversation with CICS, but he is not. It is not quite a true conversation, hence the name pseudo-conversation.

In a pseudo-conversational programming style, CICS remembers which program to run next to handle the input. If we have a single pseudo-conversational program, shown in Figure 9-22 on page 309, the program must be able to check if it is the first invocation or a subsequent invocation.

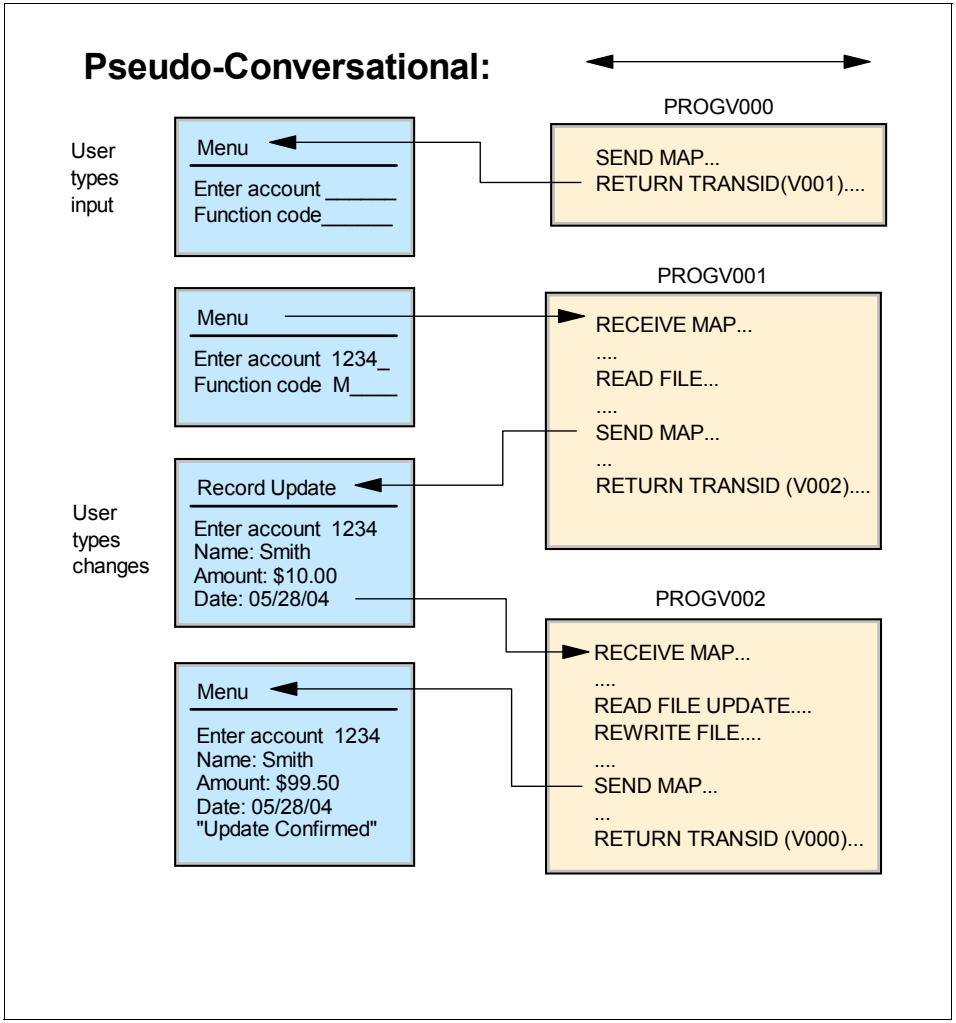


Figure 9-22 Example of a pseudo-conversational transaction

In Figure 9-23 on page 310, you can see what is necessary to change the sample conversational HelloWorld program to a pseudo-conversational.

```
CBL CICS
IDENTIFICATION DIVISION.
PROGRAM-ID. HIWORLD2.
DATA DIVISION.
WORKING-STORAGE SECTION.
    COPY MYCOBMAP.
01  MY-CA PIC X.
PROCEDURE DIVISION.
    IF EIBCALEN = ZERO
       THEN
         EXEC CICS SEND MAP('SCREEN1') MAPSET('MYMAPS')
                   MAPONLY FREEKB ERASE END-EXEC
         EXEC CICS RETURN TRANSID(EIBTRNID) COMMAREA(MY-CA)
                   END-EXEC
    END-IF
    EXEC CICS RECEIVE MAP('SCREEN1') MAPSET('MYMAPS')
              END-EXEC
    STRING '*** HELLO '  DELIMITED SIZE
           NAMEINI       DELIMITED SPACE
           ' ***'        DELIMITED SIZE
           INTO SAYHIO END-STRING
    EXEC CICS SEND MAP('SCREEN1') MAPSET('MYMAPS') DATAONLY
              FREEKB END-EXEC
    EXEC CICS RETURN END-EXEC
    GOBACK.
END PROGRAM HIWORLD2.
```

Figure 9-23 Pseudo-Conversational COBOL program sending and receiving a map

Two things are new in this program. You see two variables starting with Executive Interface Book (EIB): EIBCALEN and EIBTRNID. EIB is the abbreviation for Execute Interface Block. The EIB provides some information to every task. We describe the EIB in detail in the next chapter. The second new thing is the introduction of a commarea. Whether a task is invoked with a commarea is reflected in the EIBCALEN. A value of zero in EIBCALEN indicates that no commarea is available. A task started on a terminal with a transaction ID always has no commarea. We check this first, and when we detect zero in EIBCALEN, we send the map out with the instruction to the operator to enter their name. Then we leave the system, telling CICS to provide a small commarea at the next invocation. At the moment, we do not provide real information in this commarea. The information we get is that we have a commarea; therefore, it cannot be the first invocation. The next invocation must be the same transaction ID, which is meant with EIBTRNID. EIB fields are inserted into your program automatically by the CICS translator. You do not have to code or copy any EIB fields in your source. But you can see all of the fields in the program listing created by the language compiler.

The map is now standing on the panel, and the program has completely left the CICS system. If you use CEMT INQ TASK, as we did in the conversational example, you cannot find any hint to transaction HIWO. However, if you press Enter after you enter your name, CICS remembers the small commarea you gave it on trust. CICS also invokes the transaction with the one-byte commarea. The same program then runs a different path.

The IF statement, IF EIBCALEN=ZERO, is now NOT true, and the SEND MAP and RETURN TRANSID is not executed. Instead, the program receives the input (your name) from the screen, builds the output string, sends it out, and terminates definitively.

9.14 The Execute interface block

In addition to the usual CICS control blocks, each task in a command-level environment has a control block known as the *EXEC interface block* (EIB) associated with it.

An application program can access all of the fields in the EIB by name. The EIB contains information that is useful during the execution of an application program, such as the transaction identifier, the time and date (initially when the task is started, and subsequently, if updated by the application program using ASKTIME), and the cursor position on a display device. The EIB also contains information that is helpful when a dump is used to debug a program.

We show the entire control block in Figure 9-24 on page 312.

```
1 DFHEIBLK. . . . . DS 0CL85          Group
 2  EIBTIME . . . . . DS 4P           Packed-Dec
 2  EIBDATE . . . . . DS 4P           Packed-Dec
 2  EIBTRNID. . . . . DS 4C           Display
 2  EIBTASKN. . . . . DS 4P           Packed-Dec
 2  EIBTRMID. . . . . DS 4C           Display
 2  DFHEIGDI. . . . . DS 2C           Binary
 2  EIBCPOSN. . . . . DS 2C           Binary
 2  EIBCALEN. . . . . DS 2C           Binary
 2  EIBAID. . . . . . DS 1C           Display
 2  EIBFN . . . . . . DS 2C           Display
 2  EIBRCODE. . . . . DS 6C           Display
 2  EIBDS . . . . . . DS 8C           Display
 2  EIBREQID. . . . . DS 8C           Display
 2  EIBRSRCE. . . . . DS 8C           Display
 2  EIBSYNC . . . . . DS 1C           Display
 2  EIBFREE . . . . . DS 1C           Display
 2  EIBRECV . . . . . DS 1C           Display
 2  EIBFIL01. . . . . DS 1C           Display
 2  EIBATT. . . . . . DS 1C           Display
 2  EIBEOC. . . . . . DS 1C           Display
 2  EIBFMH. . . . . . DS 1C           Display
 2  EIBCOMPL. . . . . DS 1C           Display
 2  EIBSIG. . . . . . DS 1C           Display
 2  EIBCONF . . . . . DS 1C           Display
 2  EIBERR. . . . . . DS 1C           Display
 2  EIBERRCD. . . . . DS 4C           Display
 2  EIBSYNRB. . . . . DS 1C           Display
 2  EIBNODAT. . . . . DS 1C           Display
 2  EIBRESP . . . . . DS 4C           Binary
 2  EIBRESP2. . . . . DS 4C           Binary
 2  EIBRLDBK. . . . . DS 1C           Display
```

Figure 9-24 Execute Interface Block (EIB)

9.15 CICS debugging at tracing

In this section, we discuss CICS debugging at tracing.

9.15.1 Debugging the application using CEDF and CEDX

Now while the pseudo-conversational program is running, let us try to debug it. CEDF allows you to intercept your application programs at the program's initiation, at each CICS command, and at the program termination. EDF stands for Execution Diagnostic Facilities.

If you type `CEDF` on a cleared CICS screen, CICS replies with the message THIS TERMINAL: EDF MODE ON. The message, `THIS TERMINAL`, implies that you can also debug programs on other terminals. Imagine you have a huge application that sends many maps onto a screen. You know a problem occurs on the last screen, which is boring to step through the application until you reach the point from where you really want to debug. In this case, run the application to the

screen from where you want to start. Then issue a second session and enter CEDF followed by the terminal ID where the map was issued and now waiting for input. Lets say our pseudo-conversation transaction HIWO is started on terminal $AAA and the map is displayed asking the operator to enter his/her name. You want to debug the application starting from this point. It is not possible to enter CEDF on terminal $AAA because this terminal expects to continue with operators name. So you have to open a second session to CICS. Perhaps you get a terminal ID of $AAB. Then you can enter on terminal $AAB the debug transaction followed by the terminal you want to debug: CEDF $AAA. CICS reply the message TERMINAL $AAA: EDF MODE ON.

Switch back to $AAA and enter your name. You see that on the bottom line a X followed by a analog clock is displayed and it appears that the session is hanging. Go back to terminal $AAB and you will see CEDF stopped at program initiation. The first screen shows important information for your debug session. You can see at the top line what the transaction ID is, the program name and the task number is displayed and on the body of the screen you see the EIB with all its values.

Press enter and EDF stops at the first command. The command is not executed at that time. You have the chance to change values of the command options before you press enter to continue and execute the command.

If the command is executed you can see the results, hopefully RESPONSE: NORMAL on the left lower corner. If something fails, that is perhaps the reason why you debug the program, you can see the reason why the command failed. In our example we see a problem when the operator press enter key without enter a name before. In the CEDF session we see the response RESPONSE: MAPFAIL.

That you get an impression how the debug screens look we provide the program initiation screen in Figure 9-25 on page 314 and the response mapfail on Figure 9-26 on page 314.

```
TRANSACTION: HIWO PROGRAM: HIWORLD2 TASK: 0000146 APPLID: IYCKZCHG DISPLAY:  00
 STATUS:  PROGRAM INITIATION
    COMMAREA      = '.'
    EIBTIME       = 150022
    EIBDATE       = 0111077
    EIBTRNID      = 'HIWO'
    EIBTASKN      = 146
    EIBTRMID      = '$AAA'

    EIBCPOSN      = 757
    EIBCALEN      = 1
    EIBAID        = X'7D'                                          AT X'216300EA'
    EIBFN         = X'0000'                                        AT X'216300EB'
    EIBRCODE      = X'000000000000'                                AT X'216300ED'
    EIBDS         = '........'
+   EIBREQID      = '........'


ENTER:  CONTINUE
PF1 : UNDEFINED            PF2 : SWITCH HEX/CHAR      PF3 : END EDF SESSION
PF4 : SUPPRESS DISPLAYS    PF5 : WORKING STORAGE      PF6 : USER DISPLAY
PF7 : SCROLL BACK          PF8 : SCROLL FORWARD       PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY     PF11: EIB DISPLAY          PF12: UNDEFINED
```

Figure 9-25 CEDF screen at program initiation

```
TRANSACTION: HIWO PROGRAM: HIWORLD2 TASK: 0000146 APPLID: IYCKZCHG DISPLAY:  00
 STATUS:  COMMAND EXECUTION COMPLETE
 EXEC CICS RECEIVE MAP
  MAP ('SCREEN1')
  INTO ('.......D..FD............................X        ................'...)
  MAPSET ('MYMAPS ')
  TERMINAL





 OFFSET:X'00055C'    LINE:000053         EIBFN=X'1802'
 RESPONSE: MAPFAIL                       EIBRESP=36

ENTER:  CONTINUE
PF1 : UNDEFINED            PF2 : SWITCH HEX/CHAR      PF3 : END EDF SESSION
PF4 : SUPPRESS DISPLAYS    PF5 : WORKING STORAGE      PF6 : USER DISPLAY
PF7 : SCROLL BACK          PF8 : SCROLL FORWARD       PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY     PF11: EIB DISPLAY          PF12: ABEND USER TASK
```

Figure 9-26 CEDF screen at MAPFAIL

To summarize the chapter, you can use CEDF to debug terminal oriented CICS transaction. The terminal-ID is the principal facility for a transaction started on a 3270 panel. CEDX is another CICS supplied transaction useful to debug non-terminal transactions. There are methods to start a transaction without a terminal as its principal facility. Then you can enter CEDX transaction ID, for example, CEDX HIWO to debug this transactions. The panels are the same as in CEDF.

9.15.2 Tracing the application

The CICS trace program records significant events in the life of your application. You can switch tracing to on with the CETR transaction. With this transaction you can specify how big your trace data is, if you want internal tracing (in-storage) or external or auxiliary tracing (on-disk), and which components you want to see in the trace and at which level.

We run one instance of our sample application with standard tracing on for all domains. We formatted the auxiliary trace data set with a CICS batch utility DFHTU660. 660 is the CICS version we used. Keep in mind that this utility is version dependent, and you must always use the correct utility for your CICS version.

Figure 9-27 shows a snippet of the trace of our sample application. It shows that the operator pressed Enter without entering his name. This is an unconsidered condition that was not handled by the application program.

As we said before, a trace can be helpful; however, it is CPU intensive. Hence during development and for debugging purposes, trace must be on. In production where the applications run error free, it is a good practice to run without tracing to send the available CPU to the applications instead of for tracing purposes. However, if you see problems in production, and you must take a dump to analyze the problem or you decide to involve the IBM software support for help, we strongly recommend that you activate standard tracing before you take dumps. With standard tracing on the IBM software support, they can see what is going on in your system and can perhaps make the decision if special tracing is necessary for specific domains, depending on what the error is.

```
 Display  Filter  View  Print  Options  Help
 ------------------------------------------------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY VOER4A   JOB22953  DSID   102 LINE 66      COLUMNS 02- 133
 COMMAND INPUT ===>                                            SCROLL ===> CSR
00129 QR    AP 00E1 EIP    ENTRY RECEIVE-MAP                          0004,21639380 ..l.,00001002 ....          =000267=
00129 QR    AP 00FA BMS    ENTRY MAP-FROM              IN MAP MAPSET  0003,00020505 ....,00000020 ....          =000266=
00129 QR    AP 00FA BMS    ENTRY MAP-FROM              IN MAP MAPSET  0003,00020505 ....,00000020 ....          =000265=
00129 QR    PG 0501 PGLD   ENTRY LOAD                  MYMAPSM,TASK_LIFE,MAPSET,NO                              =000270=
00129 QR    DO 0301 DOLO   ENTRY LOCATE                20645E60,214C2200,PPT,MYMAPSM                            =000271=
00129 QR    DO 0302 DOLO   EXIT  LOCATE/OK              D7D7E3C5 , 21923030                                      =000272=
00129 QR    LD 0001 LDLD   ENTRY ACQUIRE_PROGRAM        21920E00,YES                                             =000273=
00129 QR    LD 0002 LDLD   EXIT  ACQUIRE_PROGRAM/OK     00049000,00049000,150,REUSABLE,CDSA,OLD_COPY,            =000274=
00129 QR    PG 0502 PGLD   EXIT  LOAD/OK                00049000,00049000,150                                    =000275=
00129 QR    SM 0C01 SMMG   ENTRY GETMAIN                58,219285T0,YES,00,TERMINAL                              =000276=
00129 QR    SM 0C02 SMMG   EXIT  GETMAIN/OK             219BT630                                                 =000277=
00129 QR    SM 0D01 SMMF   ENTRY FREEMAIN               219BT630,219285T0                                        =000278=
00129 QR    SM 0D02 SMMF   EXIT  FREEMAIN/OK            TERMINAL storage at 219BT630                             =000279=
00129 QR    AP 00FA BMS    EXIT  CANNOT-MAP-I/O-AREA                  0405,04000000 ....,00000000 ....          =000280=
00129 QR    AP 00FA BMS    EXIT  CANNOT-MAP-I/O-AREA                  0405,04000000 ....,00000000 ....          =000281=
00129 QR    SM 0D01 SMMF   ENTRY FREEMAIN               219BT510,219285T0                                        =000282=
00129 QR    SM 0D02 SMMF   EXIT  FREEMAIN/OK            TERMINAL storage at 219BT510                             =000283=
00129 QR    PG 0T00 PGHM   ENTRY INQ_CONDITION          24                                                       =000284=
00129 QR    PG 0T01 PGHM   EXIT  INQ_CONDITION/OK       AE19,00000000,00000000,,0,SYSTEM_DEFAULT,0               =000285=
00129 QR    AP 00E1 EIP    EXIT  RECEIVE-MAP            MAPFAIL                    08F4,00000000 ....,00241002 ....  =000286=
00129 QR    AP 2000 PCPG   ENTRY ABEND                                                                           =000287=
00129 QR    PG 0500 PGIS   ENTRY INQUIRE_CURRENT_PROGRAM                                                         =000288=
```

Figure 9-27 Trace output for transaction HIWO with MAPFAIL problem

The right side of Figure 9-27 shows every trace line as a specific number. Trace entry number =000267= shows the start of command EXEC CICS RECEIVE MAP, EIP ENTRY RECEIVE-MAP. EIP is the EXEC Interface Program, which

provides the support for application programs containing EXEC CICS commands. EIP EXIT RECEIVE-MAP at trace entry =000286= shows the and of this command. Obviously, something went wrong with this command. You can see the reason MAPFAIL. The immediate following trace entry shows that CICS starts the abend (abnormal ending) processing for this task.

9.16 Abend processing and dumps

When a program runs into an abnormal condition that is not handled in the program, or a situation where program execution cannot continue normally, CICS starts abend processing for the task. Depending on the settings in the transaction dumpcode table, CICS creates a transaction dump or a system dump.

With the sample application provided so far, you can easily simulate an abnormal condition when you press Enter before entering something. This leads to MAPFAIL condition, which is not handled at the moment. So CICS abends the task with a transaction dump. In the next section, we show how you can handle conditions in general.

9.17 Return-code handling RESP and RESP2

One important thing we ignored up to now in our samples is condition handling. We wanted to keep the samples small and concise. But it is really necessary to use condition handling in a CICS program. When a CICS command fails at execution time, you can catch that condition and decide how to proceed. At the moment, without any condition handling, CICS abnormally ends the transaction on any condition, which is not normal.

Let us give an example. If you start one of the previous examples, sending a map to a screen, and you press Enter before populating the required field, CICS abends the EXEC CICS RECEIVE MAP with a mapfail condition. This is definitely not what you want in a production program. So it is good practice to handle this condition. There are two ways to do this. You can use EXEC CICS HANDLE CONDITION to specify the label to which control is to be passed if a condition occurs. You must include the name of the condition and a label to which control is to be passed, if the condition occurs. In any case, a control break occurs and a branch takes place. Example 9-3 on page 317 shows how to code a EXEC CICS HANDLE CONDITION.

Example 9-3 Example of a handle condition

```
PROCEDURE DIVISION.
EXEC CICS HANDLE CONDITION MAPFAIL (A100-MAPFAIL) END-EXEC
...
EXEC CICS RECEIVE MAP('SCREEN1') MAPSET('MYMAPS')
          END-EXEC
...
A100-MAPFAIL.
*** DO SOME ERROR PROCESSING HERE ***
```

This technique is somewhat ill-reputed as being spaghetti programming because we branch around in the program. Therefore we recommend that you use the RESP option.

You can use the RESP option with any command to test whether a condition was raised during its execution. With some commands, when a condition can be raised for more than one reason, you can, if you have already specified RESP, use the RESP2 option to determine exactly why a condition occurred.

Using the RESP and optionally RESP2 option directly in the EXEC CICS RECEIVE MAP command avoids that you branch away if the condition is not normal. You can directly handle the condition after the command is executed. And you stay in-line in the code. You can treat RESP as a return-code and RESP2 as a reason-code.

Example 9-4 shows how to add a RESP condition to the **receive** command and how to check the condition right after it occurs.

Example 9-4 Use RESP option and check the condition

```
WORKING-STORAGE SECTION.
01 RC PIC 9(8) BINARY.
01 RSN PIC 9(8) BINARY.
...
PROCEDURE DIVISION.
.
EXEC CICS RECEIVE MAP('SCREEN1') MAPSET('MYMAPS') RESP(RC) RESP2(RSN)
END-EXEC

IF RC = DFHRESP(MAPFAIL)
   THEN
        *** DO SOME ERROR PROCESSING HERE ***
```

A solution to a mapfail condition can be to resend the map with the demand to enter a value because input is mandatory.

9.18 Sample pseudo-conversation program in PL/I

Figure 9-28 on page 319 shows our sample program in PL/I. Of course, the shape looks quite different from COBOL. But if you concentrate on the CICS API, you recognize that it is exactly the same as in our COBOL sample, except that the delimiter for the EXEC CICS commands in PL/I is simply a semicolon instead of an END-EXEC.

Additionally, notice the first lines *PROCESS statement. SYSTEM(CICS) signifies to the PL/I compiler that the source is a CICS program, and the integrated translator must be invoked for every EXEC CICS statement.

PL/I uses %include to move copybooks into the source code. While we defined the map only one time, as shown in Figure 9-14 on page 301, we can generate a copybook of this map for each programming language supporting BMS. To generate the copybook for PL/I we changed LANG=COBOL to LANG=PL1 (refer to Figure 9-14 on page 301) and renamed the output to MYPL1MAP.

If you have copybooks generated for each programming language, programs written in other languages can use the same physical map.

```
*PROCESS SYSTEM(CICS);
 HIWORLD: PROC(COMPTR) OPTIONS(MAIN);
 %INCLUDE MYPL1MAP;   /* DSECT GENERATED BY BMS */
 DCL MY_CA CHAR(1) BASED(COMPTR);
 DCL COMPTR PTR;
 DCL (LENGTH,VERIFY,ADDR,STG,LOW) BUILTIN;
 DCL STR CHAR(32767) BASED;
 DCL ALPHABETH CHAR(26) STATIC INIT('ABCDEFGHIJKLMNOPQRSTUVWXYZ');
 IF EIBCALEN = 0
    THEN DO;
            SUBSTR(ADDR(SCREEN1O)->STR,1,STG(SCREEN1O)) =
            LOW(STG(SCREEN1O));
            EXEC CICS SEND MAP('SCREEN1') MAPSET('MYMAPS')
                      MAPONLY FREEKB ERASE;
            ALLOC MY_CA SET(COMPTR);
            EXEC CICS RETURN TRANSID(EIBTRNID) COMMAREA(MY_CA);
         END;
 EXEC CICS RECEIVE MAP('SCREEN1') MAPSET('MYMAPS');
 SAYHIO = '*** HELLO ' !!
          SUBSTR(NAMEINI,1,(VERIFY(NAMEINI,ALPHABETH)-1)) !!
          ' ***';
 EXEC CICS SEND MAP('SCREEN1') MAPSET('MYMAPS') DATAONLY FREEKB;
 EXEC CICS RETURN;
 END HIWORLD;
```

Figure 9-28 Sample in PL/I

Notice that we used the VERIFY built-in function in PL/I to find the position in the input string, which is not a character from A-Z. This delimits the input string. The program must be enhanced to check also for lower case characters or other special language-specific characters, such as ö or è. To keep it simple, we check only for uppercase English characters.

9.19 Sample pseudo-conversation program in Assembler language

Assembler language does not support an integrated translator. Therefore you must put an additional step before the assembly step. In this step, the translator is invoked separately, changing all EXEC CICS commands into a native Assembler CALL statement. The result is saved in an intermediate file, mostly a temporary data set. The intermediate file is than input to the assembler. Figure 9-29 on page 320 shows the Assembler sample, and again, all EXEC CICS statements are the same as in COBOL and PL/I.

The delimiter for EXEC CICS commands in Assembler language is space. If the EXEC CICS command needs more that one line, the line which must be

continued needs a non-blank character in column 72. We use + as the continuation character.

```
         DFHEISTG
MY_CA    DS     CL1
HEXTAB   DS     CL256
         COPY   MYASMMAP
HIWORLD5 DFHEIENT
         CLC    EIBCALEN,ZERO    /* IF EIBCALEN = ZERO                 */
         BNE    RECEIVE_MAP      /* THEN CONTINUE, ELSE BRANCH         */
         MVI    SCREEN1O,X       /* INITIALIZE MAP ...                 */
         MVC    SCREEN1O+1(L'SCREEN1O-1),SCREEN1O /* WITH LOW-VALUES */
         EXEC   CICS SEND MAP('SCREEN1') MAPSET('MYMAPS')              +
                     MAPONLY FREEKB ERASE
         EXEC   CICS RETURN TRANSID(EIBTRNID) COMMAREA(MY_CA)
RECEIVE_MAP      EQU *
         EXEC   CICS RECEIVE MAP('SCREEN1') MAPSET('MYMAPS')
         MVC    SAYHIO(10),=C
         MVC    SAYHIO+10(L'NAMEINI),NAMEINI
         XC     HEXTAB,HEXTAB
         MVI    HEXTAB+64,X
         LA     R1,SAYHIO+10
         TRT    0(30,R1),HEXTAB
         MVC    0(4,R1),=C
         EXEC   CICS SEND MAP('SCREEN1') MAPSET('MYMAPS') DATAONLY      +
                FREEKB
         EXEC   CICS RETURN
         DFHREGS
ZERO     DC     H
         LTORG
         END    HIWORLD5
```

Figure 9-29 Sample in Assembler language

9.20 Separating business and presentation logic

In general, it is good practice to split applications into a part containing the business code that is reusable and a part that is responsible for presentation to the client. This technique enables you to improve performance by optimizing the parts separately and allows you to reuse your business logic with several forms of presentation.

The small CICS sample program provided in this book accepts data (the name) from and end user at a 3270 terminal, builds a message including the name entered previously, and sends a response back to the end user. The whole simple logic was implemented into one pseudo-conversational program.

In this chapter, we split this logic into two programs. The first program sends a BMS map to the end user's terminal, accepts the input from the end user, and calls the second program, which is the business logic. The second program builds the response message and send it back to the primary program, which sends the response out to the end user.

After the business logic of the transaction is isolated from the presentation logic and given a communication area (commarea) or channel and container interface, it is available for reuse with other presentation methods. For example, you can use a distributed program link (DPL) to implement a two-tier-model or a CICS web service with the CICS business logic interface, where the presentation is HTTP-based.

In Figure 9-30, notice that the sample program in COBOL is narrowed down to only include the presentation logic. The program link to a second program shown in Figure 9-31 on page 322 adheres to the business logic. The communication is done with and EXEC CICS LINK, and the data is sent with a COMMAREA. We explain in the next paragraph what a COMMAREA is and the other types of linkage between programs that are available.

Notice that LINK passes control from an application program at one logical level to an application program at the next lower logical level. In some circumstances, the linked-to program might reside on another CICS region. The linked-to program can be written in any supported programming language. We provide the business logic in PL/I, and the presentation logic is still COBOL.

```
CBL CICS
IDENTIFICATION DIVISION.
PROGRAM-ID. HIWORLD6.
DATA DIVISION.
WORKING-STORAGE SECTION.
    COPY MYCOBMAP.
01  MY-CA PIC X.
01  CA-TO-BUSINESS-LOGIC.
    02  NAME-IN       PIC X(20).
    02  RESPONSE-OUT PIC X(40).
PROCEDURE DIVISION.
    IF EIBCALEN = ZERO
       THEN
         EXEC CICS SEND MAP('SCREEN1') MAPSET('MYMAPS')
                   MAPONLY FREEKB ERASE END-EXEC
         EXEC CICS RETURN TRANSID(EIBTRNID) COMMAREA(MY-CA)
                   END-EXEC
    END-IF
    EXEC CICS RECEIVE MAP('SCREEN1') MAPSET('MYMAPS')
              END-EXEC
* ------------------------------------------------------------ *
    MOVE NAMEINI TO NAME-IN
    EXEC CICS LINK PROGRAM('BLOGIC1')
              COMMAREA(CA-TO-BUSINESS-LOGIC)
              END-EXEC
    MOVE RESPONSE-OUT TO SAYHIO
* ------------------------------------------------------------ *
    EXEC CICS SEND MAP('SCREEN1') MAPSET('MYMAPS') DATAONLY
              FREEKB END-EXEC
    EXEC CICS RETURN END-EXEC
    GOBACK.
END PROGRAM HIWORLD6.
```

Figure 9-30 Presentation logic in COBOL using a COMMAREA

```
*PROCESS SYSTEM(CICS);
 BLOGIC1: PROC(COMPTR) OPTIONS(MAIN);
 DCL 1 CA_TO_BUSINESS_LOGIC BASED(COMPTR),
       2 NAME_IN CHAR(20),
       2 RESPONSE_OUT CHAR(40);
 DCL COMPTR PTR;
 DCL (LENGTH,VERIFY,ADDR,STG,LOW) BUILTIN;
 DCL STR CHAR(32767) BASED;
 DCL ALPHABETH CHAR(26) STATIC INIT('ABCDEFGHIJKLMNOPQRSTUVWXYZ');
 RESPONSE_OUT = '*** HELLO ' !!
            SUBSTR(NAME_IN,1,(VERIFY(NAME_IN,ALPHABETH)-1)) !!
            ' ***';
 EXEC CICS RETURN;
 END BLOGIC1;
```

Figure 9-31 Business logic in PL/I using a COMMAREA

The small pieces of programs in Figure 9-30 on page 321 and Figure 9-31 do exactly the same as in one program. The advantage is that the business logic can now be called from other front-end programs without any change.

We conclude this subject with a snap-shot of the interface between both programs. Therefore we use CEDF and stop when the EXEC CICS LINK comes back from the business logic program. Figure 9-32 shows the system directly after the call. The communication area contains the name-in PAUL and the response-out message '*** HELLO PAUL ***'.

```
TRANSACTION: TEST PROGRAM: HIWORLD6 TASK: 0000329 APPLID: IYCKZCHG DISPLAY:  00
 STATUS:  COMMAND EXECUTION COMPLETE
 EXEC CICS LINK PROGRAM
  PROGRAM ('BLOGIC1 ')
  COMMAREA ('PAUL                *** HELLO PAUL ***                    ')
  LENGTH (60)









 OFFSET:X'00056A'    LINE:000060         EIBFN=X'0E08'
 RESPONSE: NORMAL                        EIBRESP=0

ENTER:  CONTINUE
PF1 : UNDEFINED          PF2 : SWITCH HEX/CHAR     PF3 : END EDF SESSION
PF4 : SUPPRESS DISPLAYS  PF5 : WORKING STORAGE     PF6 : USER DISPLAY
PF7 : SCROLL BACK        PF8 : SCROLL FORWARD      PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY   PF11: EIB DISPLAY         PF12: ABEND USER TASK
```

Figure 9-32 CEDF at EXEC CICS LINK return

One thing to mention here is return-code handling. On this CEDF debugging panel, you see that everything works smoothly, and the business program returns

the message with a response code of normal. However, up until now, we pay no attention to situations where something can go wrong. What happens, for example, when the map is displayed on the panel, and the operator presses Enter before entering a name. That is a situation that must be handled in the program. We discuss the technique to handle this issue later on.

9.21 Linkage in CICS

How do you call business logic or other sub-programs in CICS? An EXEC CICS LINK command passes control from an application program at one logical level to an application program at the next lower logical level. If the program receiving control is not already in main storage, it is loaded. When a RETURN command is processed in the linked program, control is returned to the program initiating the link at the next sequential process instruction.

When EXEC CICS XCTL is used, it transfers control from one application program to another at the same logical level. The program from which control is transferred is released.

You can use EXEC CICS START to start a new transaction. With EXEC CICS START CHANNEL, you can start a task by passing it a channel.

LINK and XCTL commands allow you to transfer data to the sub-program by either a COMMAREA or a CHANNEL (See 9.22, “COMMAREA versus channels and containers” on page 323). The START CHANNEL only provides a channel to the started task. The START command uses temporary storage to provide data to a new task. The started task must use a RETRIEVE command to get the data.

Last, you can pass a COMMAREA or a CHANNEL with the RETURN command, as shown in Figure Figure 9-30 on page 321.

9.22 COMMAREA versus channels and containers

In this section, we discuss COMMAREA versus CHANNELs and containers.

9.22.1 COMMAREA

A COMMAREA is a facility that transfers information between two programs within a transaction or transfers information between two transactions from the same terminal.

Information in a COMMAREA is available only to the two participating programs, unless those programs take explicit steps to make the data available to other programs, which can be invoked later in the transaction. When one program links to another, the COMMAREA can be any data area to which the linking program has access. It is often in the working storage or linkage section of this program. In this area, the linking program can both pass data to the program it is invoking and receive results from the specified program. When one program uses an **xctl** command to transfer control to another program, CICS can copy the specified COMMAREA into a new area of storage because the invoking program and its control blocks are no longer available after it transfers control. In either case, the address of the area is passed to the program that is receiving control, and the CICS command-level interface sets up addressability.

9.22.2 Why COMMAREA was created

In 1975, command-level programming was introduced in CICS using an in-memory process called the communication area (COMMAREA) as a method of passing data from one application to another. The COMMAREA was limited in size to 32 KB. However, this was not a concern because application data was 3270 based and so not resource hungry. The 32 KB limit was enough to manage data propagation between applications.

9.22.3 Limitations and latest developments

In this day and age, CICS must be flexible in its ability to handle data. There are many different languages and applications that access and use CICS, and they all do not utilize the same structure.

The COMMAREA is a large and contiguous block of data that contains all of the data to be passed to the called program, even if only part of this data is required. This might not be acceptable for the modern application design, where the flexibility of data structure is a basic component that offers the possibility of accommodating future business requirements.

If an XML document must be exchanged, the parameter data contained within it is different from the data format previously known to CICS programmers.

By design, XML is extensible, so you can add further data elements when application requirements change. XML structures can accommodate a varied mix of data types, and it is common for an XML document to contain large binary objects, such as images and character and numeric payload data. When you add the lengthy tag descriptions, XML-based parameter data areas can require large areas of memory when passed by value between program elements. Therefore,

the 32 KB limit together with the static structure imposed on the traditional CICS COMMAREA is insufficient for such applications.

While the COMMAREA remains the basic way to pass data between CICS application programs with the recognized qualities mentioned earlier, a new way to exchange data is required alongside the COMMAREA solution. This provides an alternative way to satisfy the requirements introduced by the use of new application programming styles.

For customers and companies that have used CICS and COMMAREA s since its early inception, this can prove to be problematic as they attempt to keep up with ever-changing technologies. CICS Transaction Server Version 3.1 introduces a new approach that provides an easy and more flexible mechanism for exchange of large volumes of structured parameter data between CICS programs. This new approach is known as the *channels and containers model*.

Refer to the following IBM Redbooks publication for more information about channels and containers:

- *CICS Transaction Server V3R1 Channels and Containers Revealed,* SG24-7227-00

9.22.4 Channels and containers

Containers are named blocks of data that are designed to pass information between programs (essentially, named COMMAREAs). It can hold any form of application data. A container can be any size and can hold data in any format that the application requires. An application can reference any number of containers. CICS provides EXEC API verbs to create, delete, reference, access, and manipulate a container and associate it with a channel. The size of containers is limited only by the amount of storage available.

Containers are grouped together in sets called channels. Programs can pass a single channel between them. Think of it as a parameter list. The same channel can be passed from one program to another. If a channel is not explicitly specified, the current channel is used as the default value for the CHANNEL (channel-name) parameter on the EXEC CICS command.

CICS provides EXEC API, which associates a named channel with a collection of one or more containers. This is an easy way of grouping parameter data structures, which can pass to a called application. CICS destroys a channel when it can no longer be referenced, which means, when a channel becomes out of scope. Channels and COMMAREAs are mutually exclusive in the sense that an EXEC CICS LINK, EXEC CICS XCTL, and EXEC CICS RETURN can only pass a channel or a COMMAREA.

9.22.5 Migrating from COMMAREAs to channels and containers

One of the more common situations that long time CICS users run into when dealing with COMMAREAs is the prospect of migrating from Commareas to channels and containers. Such is the situation that our Catalog sales store ran into as they are moving to becoming a purely online catalog store. To help them with this, we explained two key points:

- CICS application programs that use traditional COMMAREAS to exchange data continue to work as before.
- If you employ a user-written dynamic or distributed routing program for workload management, rather than CICSPlex System Manager, you must modify your program to handle the new values that might be passed in the DYRTYPE field of the DFHDYPDS communications area.

Prior to the actual migration from COMMAREAs to channels and containers, we went thorough the customer's application logic to determine how to best utilize channels and containers. Their applications must be changed to accommodate this new function, so we must be clear about what the application does, how it works, and whether this migration is necessary or feasible. After reviewing the code, we cannot replace one COMMAREA with one channel that has only one container. This is bad practice and does not benefit our clients. It might seem to be the easiest and most standard way to do this, but it is not.

There are four main ways to exchange COMMAREAs: using link command, **`xctl`** command, pseudo-conversational transactions, and using start data. In the figures that follow, you will see how the COMMAREAs were originally passed and how channels and containers are used after migration.

Migrating link commands that pass COMMAREAs

To migrate two programs that use a COMMAREA on a **`link`** command to exchange a structure, change the instructions as shown in Table 9-1 on page 327.

Table 9-1 Migrating link commands that pass COMMAREAs

Program	Before	After
PROG1	EXEC CICS LINK PROGRAM(PROG2) COMMAREA(structure)	EXEC CICS PUT CONTAINER(contr-name) CHANNEL(channel-name) FROM(structure) EXEC CICS LINK PROGRAM(PROG2) CHANNEL(channel-name) . . . EXEC CICS GET CONTAINER(contr-name) CHANNEL(channel-name) INTO(structure)
PROG2	EXEC CICS ADDRESS COMMAREA(structurePtr) ... RETURN	EXEC CICS GET CONTAINER(contr-name) INTO(structure) ... EXEC CICS PUT CONTAINER(contr-name) FROM(structure) EXEC CICS RETURN

Note: In the COMMAREA example, PROG2, having put data in the COMMAREA, must only issue a **`return`** command to return the data to PROG1. In the channel example, to return data, PROG2 must issue a **`put container`** command before the **`return`**.

Migrating xctl commands that pass COMMAREAs

To migrate two programs that use a COMMAREA on an **`xctl`** command to pass a structure, change the instructions, as shown in Table 9-2 on page 328.

Table 9-2 Migrating xctl commands that pass COMMAREAs

Program	Before	After
PROG1	EXEC CICS XCTL PROGRAM(PROG2) COMMAREA(structure)	EXEC CICS PUT CONTAINER(contr-name) CHANNEL(channel-name) FROM(structure) EXEC CICS XCTL PROGRAM(PROG2) CHANNEL(channel-name)
PROG2	EXEC CICS ADDRESS COMMAREA(structurePtr)	EXEC CICS GET CONTAINER(contr-name) INTO(structure)

Migrating pseudo-conversational transactions COMMAREAs

To migrate two programs that use COMMAREAs to exchange a structure as part of a pseudo-conversation, change the instructions, as shown in Table 9-3.

Table 9-3 Migrating pseudo-conversational COMMAREAs on return command

Program	Before	After
PROG1	EXEC CICS RETURN TRANSID(TRAN2) COMMAREA(structure)	EXEC CICS PUT CONTAINER(contr-name) CHANNEL(channel-name) FROM(structure) EXEC CICS RETURN TRANSID(TRAN2) CHANNEL(channel-name)
PROG2	EXEC CICS ADDRESS COMMAREA(structurePtr)	EXEC CICS GET CONTAINER(contr-name) INTO(structure)

Migrating start data

To migrate two programs that use start data to exchange a structure, change the instructions as shown in Table 9-4 on page 329.

Table 9-4 Migrating start data

Program	Before	After
PROG1	EXEC CICS START TRANSID(TRAN2) FROM(structure)	EXEC CICS PUT CONTAINER(contr-name) CHANNEL(channel-name) FROM(structure) EXEC CICS START TRANSID(TRAN2) CHANNEL(channel-name)
PROG2	EXEC CICS RETRIEVE INTO(structure)	EXEC CICS GET CONTAINER(contr-name) INTO(structure)

The COMMAREA option of LINK and XCTL commands specifies the name of a data area (known as a communication area) in which data is passed to the program being invoked.

In a similar manner, the COMMAREA option of a RETURN command specifies the name of a communication area in which data is passed to the transaction identified in the TRANSID option. The invoked program receives the data as a parameter. The program must contain a definition of a data area to allow access to the passed data.

In a PL/I program, the data area can have any name, but it must be declared as a based variable, based on the parameter passed to the program. The pointer to this based variable must be declared explicitly. Figure 9-31 on page 322 shows an example of how the commarea CA_TO_BUSINESS_LOGIC is passed using the pointer COMPTR.

The theoretical upper limit of a COMMAREA is 32 763 bytes. This limitation was one of the reasons we introduced a new method of transferring data between CICS programs. Alternatively to using a communication area, you can use a channel, which has several advantages over COMMAREAs. Unlike COMMAREAs, channels are not limited in size to 32 KB, and channels do not require the programs that use them to know the exact size of the data returned. Because a channel can comprise multiple containers, it can be used to pass data in a more structured way. In contrast, a COMMAREA is a monolithic block of data.

Figure 9-33 on page 330 and Figure 9-34 on page 330 show how we migrated the use of a COMMAREA as the communication method between two programs to a CHANNEL.

```
CBL CICS
IDENTIFICATION DIVISION.
PROGRAM-ID. HIWORLD7.
DATA DIVISION.
WORKING-STORAGE SECTION.
    COPY MYCOBMAP.
01  MY-CA PIC X.
PROCEDURE DIVISION.
    IF EIBCALEN = ZERO
       THEN
         EXEC CICS SEND MAP('SCREEN1') MAPSET('MYMAPS')
                   MAPONLY FREEKB ERASE END-EXEC
         EXEC CICS RETURN TRANSID(EIBTRNID) COMMAREA(MY-CA)
                   END-EXEC
    END-IF
    EXEC CICS RECEIVE MAP('SCREEN1') MAPSET('MYMAPS')
              END-EXEC
* ------------------------------------------------------------ *
    EXEC CICS PUT CONTAINER('NAME') FROM(NAMEINI)
              CHANNEL('MY-CHANNEL') END-EXEC
    EXEC CICS LINK PROGRAM('BLOGIC2') CHANNEL('MY-CHANNEL')
              END-EXEC
    EXEC CICS GET CONTAINER('RESPONSE') INTO(SAYHIO)
              CHANNEL('MY-CHANNEL') END-EXEC
* ------------------------------------------------------------ *
    EXEC CICS SEND MAP('SCREEN1') MAPSET('MYMAPS') DATAONLY
              FREEKB END-EXEC
    EXEC CICS RETURN END-EXEC
    GOBACK.
END PROGRAM HIWORLD7.
```

Figure 9-33 Presentation logic in COBOL using a CHANNEL

```
*PROCESS SYSTEM(CICS);
 BLOGIC2: PROC OPTIONS(MAIN);
 DCL NAME_IN CHAR(20);
 DCL RESPONSE_OUT CHAR(40);
 DCL (LENGTH,VERIFY,ADDR,STG,LOW) BUILTIN;
 DCL STR CHAR(32767) BASED;
 DCL ALPHABETH CHAR(26) STATIC INIT('ABCDEFGHIJKLMNOPQRSTUVWXYZ');
 EXEC CICS GET CONTAINER('NAME') INTO(NAME_IN)
           CHANNEL('MY-CHANNEL');
 RESPONSE_OUT = '*** HELLO ' !!
           SUBSTR(NAME_IN,1,(VERIFY(NAME_IN,ALPHABETH)-1)) !!
           ' ***';
 EXEC CICS PUT CONTAINER('RESPONSE') FROM(RESPONSE_OUT)
           CHANNEL('MY-CHANNEL');
 EXEC CICS RETURN;
 END BLOGIC2;
```

Figure 9-34 Business logic in PL/I using a CHANNEL

9.23 Writing the business logic in Java

Previously, we called the business logic that was written in PL/I out of a COBOL program. In this section, we provide the business logic written in Java.

You can write CICS application programs in Java starting with CICS/TS V1.3, which provides a library of Java classes to access important CICS services. These classes allow you, for instance, to read files, write to transient data, start transactions links, communicate to other programs using COMMAREA or channels and containers, and access many other resources in CICS.

Some other resources are not built up in Java classes. For instance Java programs cannot use Basic Mapping Support. The CICS translator is used by all the other CICS supported languages. It is not required for Java. You can develop and compile Java programs on a workstation, and then FTP the class file to a UNIX System Services directory in z/OS. That is completely different to the classic CICS-supported languages, where load modules are stored in Partitioned Data Sets (PDS or PDS/E).

Notice in the classpath of your environment, where you compile the Java CICS program, that you must specify a path to find the required dfhjcics.jar file when your program uses the JCICS classes. JCICS is supplied in the dfjcics.jar JAR file and can be downloaded to your workstation.

Figure 9-35 on page 332 shows the sample business logic in Java as previously provided in PL/I in Figure 9-34 on page 330.

```
package itso;
import com.ibm.cics.server.*;
import java.io.*;
public class HworldSample {
  public HworldSample() {
    // Get the current channel
    final Task    task           = Task.getTask();
    final Channel currentChannel = task.getCurrentChannel();
    String name_in = null;
    String errorString = null;
        byte b;
    // get the Byte[] array from NAME container
    if (currentChannel != null) {
      try {
        // Get the reference number
        Container container = currentChannel.getContainer("NAME");
        byte[]    bytes     = container.get();
        name_in = new String(bytes);
                                  name_in = name_in.replaceAll(" ", "");
      }
      catch (ContainerErrorException e) { errorString = "Container not found"; }
      catch (CCSIDErrorException     e) { errorString = "CCSID error "; }
      catch (ChannelErrorException   e) { errorString = "Channel error "; }
      catch (CodePageErrorException   e) { errorString = "code page error"; }
    }
    //*****************************************************************************************
    // return the data we received
    if (name_in != null) {
      try {
        final Container containerout = currentChannel.createContainer("RESPONSE");
        String outstring = "*** HELLO " +name_in +" ***";
         containerout.put(outstring);
      }
      catch (ContainerErrorException e) { errorString = "oops container exception"; }
      catch (ChannelErrorException   e) { errorString = "oops Channel error creating container"; }
      catch (InvalidRequestException e) { errorString = "oops Invalid request error "; }
      catch (CCSIDErrorException e) { errorString = "CCSID error"; }
      catch (CodePageErrorException   e) { errorString = "code page error"; }
    }
    // If there was an error message then attempt to put that in a container
    if (errorString != null) {
      try {
        final Container errContainer = currentChannel.createContainer("error-msg");
        errContainer.put(errorString);
      }
      catch (ContainerErrorException e) { System.err.println("Container error creating container"); }
      catch (ChannelErrorException   e) { System.err.println("Channel error creating container"); }
      catch (InvalidRequestException e) { System.err.println("Invalid request error"); }
      catch (CCSIDErrorException e) { errorString = "CCSID error"; }
      catch (CodePageErrorException   e) { errorString = "code page error"; }
    }
  }
  public static void main(String[] args) {
    new HworldSample();
  }
}
```

Figure 9-35 Business logic in Java using a CHANNEL

When you have this sample compiled and implemented, the class file in the UNIX Systems Services file system, you must set up the CICS system to find this Java class file. Therefore CICS provides the JVMPROFILEDIR system initialization parameter. Here you specify the name of the z/OS UNIX directory that contains the JVM profiles for CICS. CICS searches this directory for the profiles it needs to

configure JVMs. The default value of JVMPROFILEDIR is /usr/lpp/cicsts/cicsts41/JVMProfiles.

In the directory you specify as your JVMPROFILEDIR, you must have a profile with the name DFHJVMPR, which is the default profile name, when you define the Java program to CICS. Edit this profile to find the CLASSPATH_SUFFIX. There you can add the directory where your Java class file is located.

Use CEDA or CICS Explorer to define your Java program to CICS. You can still define program s to CICS that are 8-bytes in length. This program definition then maps to the real Java class, which does not have the length restriction of 8-bytes. The language option in the program definition is blank, the concurrency option must be threadsafe, and under the JVM attributes, you specify JVM=YES, and provide the JVM class name. Figure 9-36 shows how we defined our Java sample to CICS.

```
 OVERTYPE TO MODIFY                                          CICS RELEASE = 0660
  CEDA  ALter PROGram( BLOGIC4  )
+  REMOTESystem ==>
   REMOTEName   ==>
   Transid      ==>
   EXECUtionset ==> Fullapi            Fullapi | Dplsubset
  JVM ATTRIBUTES
   JVM          ==> Yes                No | Yes
   JVMClass     ==> itso.HworldSample
   (Mixed Case) ==>
                ==>
                ==>
                ==>
   JVMProfile   ==> DFHJVMPR                                      (Mixed Case)
  JAVA PROGRAM OBJECT ATTRIBUTES
   Hotpool         : No                No | Yes
  DEFINITION SIGNATURE
   DEFinetime      : 03/30/11 16:32:58
+  CHANGETime      : 03/30/11 17:19:43

                                                     SYSID=66A2 APPLID=IV3A66A2

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

Figure 9-36 Define the Java sample to CICS using CEDA

Notice that the sample Java program `HworldSample` is in a package with the name `itso`. So you must prefix the Java program name with the package name. While the class file resides in a directory, /u/itso/javaprogs/itso/HworldSample.class, you must specify /u/itso/javaprogs in the classpath in profile DFHJVMPR.

9.23.1 The OSGi-based JVMServer

With the arrival of CICS TS V4.2 (June, 2011), CICS now supports 64-bit Java 6.0.1. You can use the traditional pooled JVM approach (which will be removed in a future version of CICS), or you can use the JVMServer resource that was added in CICS TS V4.1 and extended to be used for customer- written Java code in CICS TS V4.2.

The JVMServer resource in CICS TS V4.2 requires that Java programs are packaged using *Open Services Gateway initiative* (OSGi) bundles. OSGi is a dynamic module system for Java and was adopted by all the major Java application servers. The modularity in OSGi provides a mechanism for dividing a system into independent modules, known as *bundles*, that are independently packaged and deployed with independent lifecycles. This modularity allows you to divide a complex application into independent pieces for easier development, testing, and maintenance.

Without OSGi, Java classes and JAR files are made available to a JVM by specifying a CLASSPATH environment variable. This CLASSPATH is static and does not change for the life of the JVM. With OSGi, Java classes can be added and removed dynamically from a JVM.

OSGi also provides an in-memory service registry where other modules can find and bind to registered services. Components made from modules can be published in the OSGi service registry, and from there, other modules can find and bind to those services. Components can expose all classes in a bundle or can expose specific interfaces. Likewise, a component can declare its dependencies. A module can contain lifecycle methods that are invoked as the module is installed, uninstalled, or upgraded. You can also provide extension points to let others extend your module's functionality.

One or more Java packages (containing classes) are grouped into an OSGi bundle. The CICS environment allows you to group one or more OSGi bundles into a CICS BUNDLE. As the CICS BUNDLE is installed, enabled, disabled, and discarded, the OSGi bundles in the CICS BUNDLE go through the appropriate OSGi lifecycle changes. This provides the capability to dynamically add and remove Java classes and packages into a running CICS JVMServer. The lifecycle state of an OSGi bundle and the CICS BUNDLE can be displayed using the CICS Explorer.

The OSGi-based JVMServer is considered the strategic direction for Java-based business logic in CICS. When CICS receives a request, CICS starts a program to provide a response. You therefore must define Java programs to CICS using a PROGRAM definition. For a CICS-based Java program that is to run in a JVMServer, you must define a PROGRAM definition specifying

CONCURRENCY(REQUIRED), a storage key of CICS, the name of the JVMServer in which this program is to run, and the name of the OSGi registry entry that references the Java program to run.

When packaging your Java programs in an OSGi bundle, for those classes where CICS is to begin execution, you include a CICS-Mainclass= keyword in the bundle's manifest, which specifies the class where execution begins. The CICS-MainClass= keyword causes the CICS OSGi environment to register the specified class in the OSGi registry. Like the pooled JVM, Java program execution begins in the specified class in a method with a signature of public static void main(CommAreaHolder ca) or public static void main(String[] args).

Java programs interact with CICS using a Java equivalent to the EXEC CICS commands referred to as the JCICS classes. In addition to interacting with CICS resources and invoking CICS programs written in other languages, Java programs can interact with DB2, MQ, and do other Java-specific work. Java programs must always use CICS facilities, if they exist (for example, CICS's DB2 connections), so that CICS can control units of work (one of the many areas in which CICS excels).

The memory for JVMServers comes from the CICS region address space. CICS TS V4.2 uses a 64-bit JVM, which can reside above the bar in 64-bit storage so that multiple JVMServers can exist in a single CICS region. Each JVMServer CICS can dispatch between one and 256 concurrent tasks in a single JVMServer with 15 being the default. The sum of all concurrent tasks in all JVMServers for a given region cannot exceed 1024.

9.23.2 The Axis2-based JVMServer

CICS offered the ability to expose CICS-based program as web services since CICS TS V3.1 (March, 2005), before that using a SupportPac (March, 2003), and through the SOAP Feature for CICS (September, 2003). The web service support initially provided with CICS TS V3.1 allows for exposing CICS programs that are invoked using the CICS COMMAREA or channels and containers (both discussed later).

As an additional option, to expose Java-based CICS programs as web services, CICS TS V4.2 (June 2011) includes the open-source Axis2 technology. Axis2 is an open source web services engine from the Apache foundation and is provided with CICS to process SOAP messages in a Java environment. Utilizing the Axis2 technology for Java-based web services in CICS can provide better utilization of the zAAP specialty processor (Java and XML parsing in some situations can be offloaded to a specialty processor like the System z Application Assist Processor (zAAP) instead of running the work on a general processor in an IBM mainframe).

The Axis2 support in CICS allows you to expose CICS-based Java programs and develop Handler programs in Java. Axis2 supports hot deployment of web services and handlers, asynchronous web services, contains MEP support (message exchange pattern (part of WSDL 2.0)), WSDL 1.1 and 2.0 support, and allows you to expose POJOs (Plain Old Java Objects) as web services using the annotations that are part of JAX-WS. Tooling to build Axis2 artifacts is part of the Java 6 JSE (Java Standard Edition).

9.24 CICS web services

We cleanly separated the business logic from the presentation logic in our sample program. We can now show how to expose the business logic as a web service. To expose an existing program as a web service, we use a CICS-supplied program to generate a Web Service Definition Language (WSDL) file. The program name is DFHLS2WS, which is a Java program using the COMMAREA structure as input and generates a WSDL file and a Web Service Binding File (WSBIND) file as output.

Figure Figure 9-37 is an example of a DFHLS2WS job.

```
//MYPROCS JCLLIB ORDER=(PTFB.CICS660T.CICS.SDFHINST)
//LS2WS     EXEC DFHLS2WS,
//              PATHPREF='',
//              TMPDIR='/u/itso/temp',
//              USSDIR='cics660',
//              JAVADIR='java6_31/J6.0/'
//INPUT.SYSUT1 DD *
MAPPING-LEVEL=3.0
MINIMUM-RUNTIME-LEVEL=3.0
LOGFILE=/u/itso/HelloWorld/provider/HelloWorld.log
PDSLIB=//ITSO.CPY
REQMEM=HIWORLD
RESPMEM=HIWORLD
LANG=PLI-ENTERPRISE
PGMNAME=BLOGIC3
PGMINT=CHANNEL
URI=http://winmvs20.hursley.ibm.com:4080/HelloWorld
WSBIND=/u/itso/HelloWorld/provider/HelloWorld.wsbind
WSDL=/u/itso/HelloWorld/provider/HelloWorld.wsdl
/*
//
```

Figure 9-37 DFHLS2WS to create the wsbind file and the wsdl file

In PDSLIB, we specified the copybook library where we saved the interface to our sample program in member HIWORLD, as shown in Figure 9-38.

```
DCL 1 CA_TO_BUSINESS_LOGIC,
      2 NAME_IN CHAR(20),
      2 RESPONSE_OUT CHAR(40);
```

Figure 9-38 Interface to business logic of sample program

DFHLS2WS stores the wsbind files into the UNIX Systems Services file system. The directory we used for our sample is in Figure 9-37 on page 336. WSBIND files are used by CICS at runtime to convert between XML and the language structure.

We changed the business logic program in a way that it can process every request. We provide program versions, which can be called with a COMMAREA, with a channel, or as a web service, as shown in Figure 9-39 on page 338.

```
*PROCESS SYSTEM(CICS);
 BLOGIC3: PROC(COMPTR) OPTIONS(MAIN);
 DCL 1 CA_TO_BUSINESS_LOGIC BASED(COMPTR),
       2 NAME_IN CHAR(20),
       2 RESPONSE_OUT CHAR(40);
 DCL COMPTR PTR;
 DCL (LENGTH,VERIFY,ADDR,STG,LOW,CSTG) BUILTIN;
 DCL STR CHAR(32767) BASED;
 DCL ALPHABETH CHAR(26) STATIC INIT('ABCDEFGHIJKLMNOPQRSTUVWXYZ');
 DCL CURRENTCHANNELNAME CHAR(16);
 EXEC CICS ASSIGN CHANNEL(CURRENTCHANNELNAME);
 IF EIBCALEN = 0                                          /* NO CA ->      */
    THEN DO;                                              /* THEN          */
         ALLOCATE CA_TO_BUSINESS_LOGIC SET(COMPTR);       /* ALLOCATE ONE  */
         IF CURRENTCHANNELNAME = 'DFHAHC-V1'              /* IS IT A WEB   */
            THEN EXEC CICS GET CONTAINER('DFHWS-DATA')    /* REQUEST?      */
                           CHANNEL('DFHAHC-V1')
                           INTO(CA_TO_BUSINESS_LOGIC);
            ELSE IF CURRENTCHANNELNAME = 'MY-CHANNEL'     /* IS IT A       */
                    THEN EXEC CICS GET CONTAINER('NAME')  /* CHANNEL       */
                                   CHANNEL('MY-CHANNEL')  /* REQUEST?      */
                                   INTO(NAME_IN);
                    ELSE EXEC CICS ABEND ABCODE('FRED');  /* STOP IT       */
         END;                                             /* ALLOCATE ONE  */
 RESPONSE_OUT = '*** HELLO ' !!
           SUBSTR(NAME_IN,1,(VERIFY(NAME_IN,ALPHABETH)-1)) !!
            ' ***';
 IF CURRENTCHANNELNAME = 'MY-CHANNEL'
    THEN EXEC CICS PUT CONTAINER('RESPONSE') CHANNEL('MY-CHANNEL')
                   FROM(RESPONSE_OUT);
    ELSE IF CURRENTCHANNELNAME = 'DFHAHC-V1'
            THEN EXEC CICS PUT CONTAINER('DFHWS-DATA')
                           CHANNEL('DFHAHC-V1') FROM(CA_TO_BUSINESS_LOGIC);
 EXEC CICS RETURN;
 END BLOGIC3;
```

Figure 9-39 Business logic in PL1 ready for web services

In Figure 9-39, we made the program ready to accept web services and commarea and channels. When invoked as a web service, we test on channel name DFHAHC-V1 and receive input and output into the CA_TO_BUSINESS_LOGIC structure. When a request arrives as an EXEC CICS LINK request with a channel, we only receive the input container and reply only the output container, which is application dependent. An application can also send input and output in one container. Last, this module can also be linked-to a traditional commarea.

You can change this program to get the container names dynamically by using the API command STARTBROWSE CONTAINER. There are good examples in *CICS Transaction Server V3R1 Channels and Containers Revealed*, SG24-7227.

Use the CICS Information Center for any questions you have about the CICS API. In the Application Programming Reference, you can see how to code a STARTBROWSE CONTAINER. Figure 9-40 on page 339 shows what you see in the CICS Information Center when searching for the STARTBROWSE command.

```
STARTBROWSE CONTAINER

>>-STARTBROWSE--CONTAINER---------------------------------------->

>--+-----------------------------------------------+------------->
   +-ACTIVITYID(data-value)-----------------------+
   +-PROCESS(data-value)--PROCESSTYPE(data-value)-+
   '-CHANNEL(data-value)-------------------------'

>--BROWSETOKEN(data-area)---------------------------------------><
```

Conditions: ACTIVITYERR, CHANNELERR, IOERR, NOTAUTH, PROCESSERR

Figure 9-40 CICS Information Center STARTBROWSE command

Visit the CICS Information Center for CICS/TS V4.1 at:

http://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp

9.24.1 Invoking the web service from a CICS

CICS can be used as a web service provider and as a web service requester. Figure 9-41 on page 340 shows CICS as a web service requester and how we invoked the business logic as a web service using the CICS API. Notice that the EXEC CICS LINK was replaced with an EXEC CICS INVOKE SERVICE command.

```
CBL CICS
IDENTIFICATION DIVISION.
PROGRAM-ID. HIWORLD0.
DATA DIVISION.
WORKING-STORAGE SECTION.
    COPY MYCOBMAP.
01  MY-CA PIC X.
01  CA-TO-BUSINESS-LOGIC.
    02  NAME-IN      PIC X(20).
    02  RESPONSE-OUT PIC X(40).
PROCEDURE DIVISION.
    IF EIBCALEN = ZERO
       THEN
         EXEC CICS SEND MAP('SCREEN1') MAPSET('MYMAPS')
                   MAPONLY FREEKB ERASE END-EXEC
         EXEC CICS RETURN TRANSID(EIBTRNID) COMMAREA(MY-CA)
                   END-EXEC
    END-IF
    EXEC CICS RECEIVE MAP('SCREEN1') MAPSET('MYMAPS') END-EXEC
* ------------------------------------------------------------ *
    MOVE NAMEINI TO NAME-IN
    EXEC CICS PUT CONTAINER('DFHWS-DATA') CHANNEL('NC-CHANNEL')
              FROM(CA-TO-BUSINESS-LOGIC) END-EXEC
    EXEC CICS INVOKE SERVICE('Helloworld') CHANNEL('NC-CHANNEL')
              OPERATION('BLOGIC3Operation') END-EXEC
    EXEC CICS get CONTAINER('DFHWS-DATA') CHANNEL('NC-CHANNEL')
              into(CA-TO-BUSINESS-LOGIC) END-EXEC
    MOVE RESPONSE-OUT TO SAYHIO
* ------------------------------------------------------------ *
    EXEC CICS SEND MAP('SCREEN1') MAPSET('MYMAPS') DATAONLY
              FREEKB END-EXEC
    EXEC CICS RETURN END-EXEC
    GOBACK.
END PROGRAM HIWORLD0.
```

Figure 9-41 Invoke business logic as a web service out of CICS

9.25 Invoking the business logic from a web server

Our example organization, the Bank, used CICS for 20 years and has multiple CICS systems. This organization worked mainly in the past with 3270 applications. With the advent of the Internet, the Bank introduced applications running on web servers and accessing business logic programs in CICS.

In this section, we provide an example for a Java program running in a web server and invoking the same business logic written in PL/I as invoked in previous sections from a COBOL program running in CICS.

We used a light-wight Apache Tomkat server to send the HTML page with the request to enter your name. Two java server pages were defined, one to save the name from input (SaveTheNamePage.jsp) and one to get the response from CICS back (GetTheResultPage.jsp). The real web service invocation is done by the Java program WebAppToCICS.java. To start the communication to CICS, we

started a browser and entered `http://localhost:8080/axis/EnterYourNamePage.htm`. The HTML page is displayed, as shown in Figure 9-42.

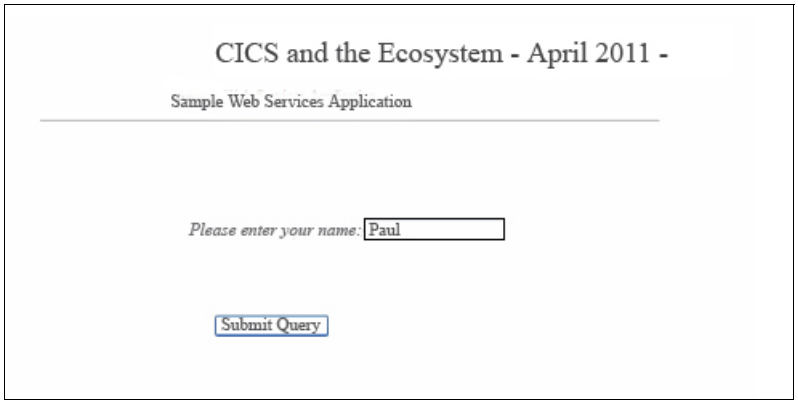


Figure 9-42 Invoke web service in CICS from a browser

We entered the name Paul, and got the output back from CICS, as shown in Figure 9-43.

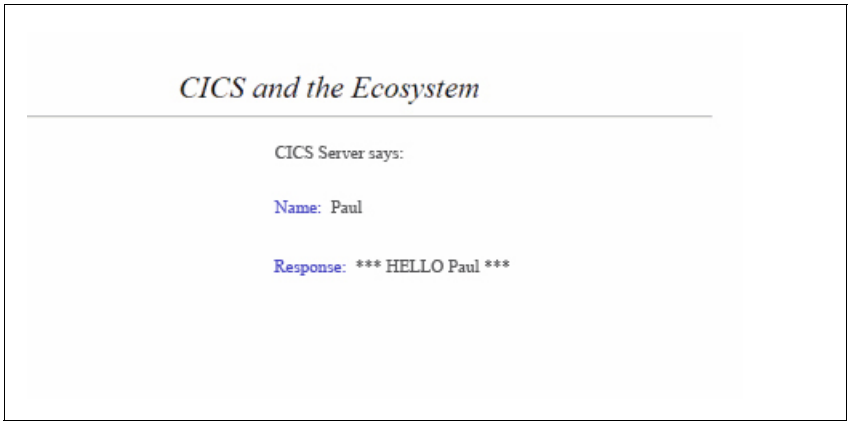


Figure 9-43 Response from CICS displayed in a browser.gif

Remember the 3270 panels? The browser looks modern, but the message you get is the same. Always keep in mind that you communicate to a CICS PL/I program using a web service and a browser interface.

We provide the HTML page, the two Java server pages, and the Java program in the additional material of this book, in case you want to recreate this on your system.

9.26 Dynamic scripting

The CICS Dynamic Scripting Feature Pack provides an agile web application platform for developing and running modern web applications. You can take advantage of web technologies and improve your productivity using CICS Dynamic Scripting to create, assemble, and run applications that meet your specific needs or the needs of your clients.

You can use the feature pack to take advantage of the following web technologies:

- A dynamic scripting CICS-based run time for Groovy and PHP applications
- Java as a system programming language
- Application programming interfaces that use Representational State Transfer (REST) services
- Rich Asynchronous JavaScript and XML (Ajax) web user interfaces
- Integration of mash-ups and feeds

The CICS Dynamic Scripting Feature Pack is developed using technology from WebSphere sMash. You can obtain more about developing dynamic scripting applications by visiting the community website for WebSphere sMash at:

http://www.projectzero.org

9.26.1 Downloading and installing the CICS Transaction Server Feature Pack for Dynamic Scripting

You can download the CICS Transaction Server Feature Pack for Dynamic Scripting at:

http://www-01.ibm.com/software/htp/cics/scripting/

You can find help about how to install the Feature Pack in the IBM Redbooks publication, *Introduction to CICS Dynamic Scripting*, SG24-7924-00.

9.26.2 Preparing the CICS application

In the CICS Information Center, in Chapter “Using JZOS with dynamic scripting applications”, there are instruction for how to prepare you CICS application. Read and follow the instructions in that chapter.

In a nutshell, we compiled our CICS sample business logic program with the option ADATA. We FTPed the file created by the compiler under DD-statement SYSADATA into the UNIX System Services file system. Then, we used JZOS to process this file to create helper classes in Java to access our CICS business logic program COMMAREA in a dynamic scripting application.

We previously provided the business logic in PL/I with several linkage methods: Commarea, Channels and Containers and web services. In Figure 9-44, we provide an additional simple Cobol sample using a COMMAREA interface. We used this Cobol example to create the ADATA file. Of course, there is a ADATA option in PL/I too.

```
CBL CICS
IDENTIFICATION DIVISION.
PROGRAM-ID. BLOGIC5.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01  DFHCOMMAREA.
    05  NAME PIC X(20).
    05  RESPONSE PIC X(40).
PROCEDURE DIVISION USING DFHCOMMAREA.
    STRING '*** HELLO '  DELIMITED SIZE
           NAME          DELIMITED SPACE
           ' ***'        DELIMITED SIZE
           INTO RESPONSE END-STRING.
    EXEC CICS RETURN END-EXEC
    GOBACK.
END PROGRAM BLOGIC5.
```

Figure 9-44 Business logic in Cobol used to create the ADATA file

9.26.3 Creating and starting the script to access the Cobol business logic program

In this section, we discuss how to create and start the script to access the Cobol business logic program.

Groovy example

In UNIX Systems Services, /usr/lpp/cicsts/dynapps/hiworld/public/, we created the file hiworld.gt. Because the script contains both HTML and groovy, the file extension is .gt and not .groovy, as shown in Figure 9-45.

```
<html>
<body>
<h1>CICS and the Ecosystem - using Groovy </h1>
<%
import sample.HiWorld
import com.ibm.cics.server.Program
// Prepare the commarea
def commArea = new HiWorld()
def namein = "${zget('/request/params/name')}."
commArea.setName(namein);
// Invoke the program
def program = new Program()
program.name = "BLOGIC5"
program.link(commArea.byteBuffer)
// Read the commarea
def responseout = commArea.getResponse();
%>
<br>
<h2 style="color:red">  <% println responseout %> </h2>
```

Figure 9-45 Groovy script to access business logic in CICS

We used a browser, as shown in Figure 9-46 on page 345, to invoke the Cobol program with this Groovy script.

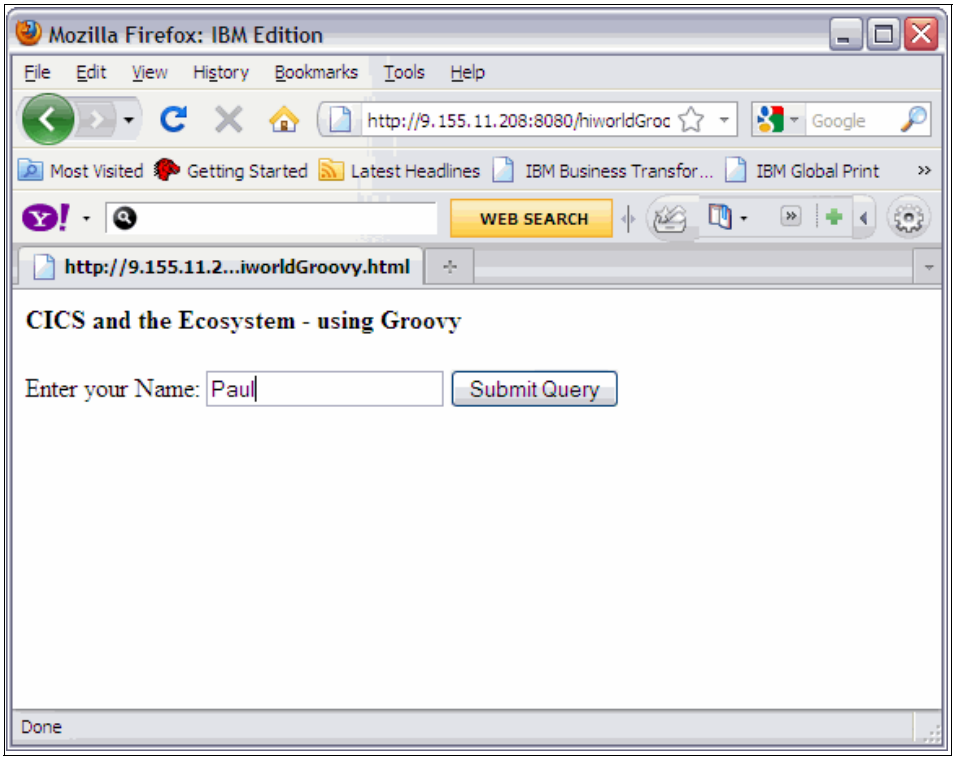


Figure 9-46 Run the Groovy script to access the Cobol Program in CICS

Figure 9-47 on page 346 shows the results.

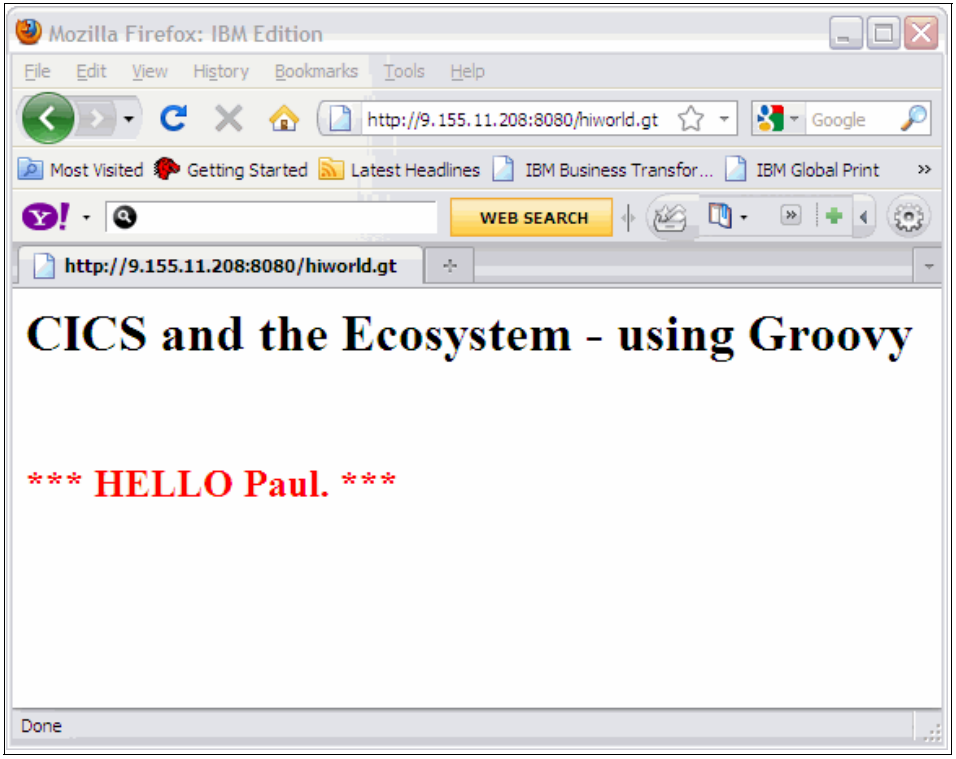


Figure 9-47 Message returned from CICS

Our PHP script returns pretty much the same output, so we simply provide the PHP script in Figure 9-48 on page 347 for those who are interested in PHP instead of Groovy.

```
<html><body>
<h1>CICS and the Ecosystem - using PHP </h1>
<?php
// Import the CICS classes
java_import('sample.HiWorld');
java_import('com.ibm.cics.server.Program');
// Prepare the commarea
$commArea = new HiWorld();
$commArea->NAME = $_POST['name'];
// Invoke the program
$program = new Program();
$program->name = 'BLOGIC5';
$program->link($commArea->byteBuffer);
// Read the commarea
$responseout = $commArea->RESPONSE;
?>
<br>
<h2 style="color:red"> <?php echo $responseout; ?> </h2>
</body></html>
```

Figure 9-48 PHP script to access business logic in CICS

CICS Dynamic Scripting integrates Project Zero technology into CICS for a robust environment for PHP and Groovy. If you want to learn more about Project Zero technology and CICS Dynamic Scripting, visit:

http://www.projectzero.org/

9.27 Open Transaction Environment and threadsafe

In this section, we discuss Open Transaction Environment (OTE) and threadsafe.

9.27.1 Explanation of Quasi-reentrant

Before jumping into OTE and threadsafe, you must understand the idea of being *quasi-reentrant*.

CICS runs user programs under a CICS-managed task control block (TCB). If a program is defined as quasi-reentrant, using the CONCURENCY attribute of the program resource definition, CICS always invokes the program under the CICS quasi-reentrant (QR) TCB.

CICS requires an application program to be reentrant to guarantee a consistent state. A program is considered reentrant if it is read only and does not modify storage within itself. In practice, an application program might not be truly reentrant. CICS expects quasi-reentrancy. This term means that the application program must be in a consistent state when control is passed to it, both on entry to the program and before and after each EXEC CICS command. Such quasi-reentrancy guarantees that each invocation of an application program is unaffected by previous runs.

CICS quasi-reentrant user programs (application programs, user-replaceable modules, global user exits, and task-related user exits) are given control by the CICS dispatcher under the QR TCB. When running under this TCB, a program can be sure that no other quasi-reentrant program can run until it relinquishes control during a CICS request. The user task is suspended at this point, leaving the program still in use. The same program can then be re-invoked by another task. This means the application program can be in use concurrently by more than one task although only one task at a time can actually be executing. To ensure that programs cannot interfere with each other's working storage, CICS obtains a separate copy of working storage for each execution of an application program. Therefore, 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 globally-shared resources, for example, the CICS common work area (CWA), without the need to protect those resources from concurrent access by other programs. Such resources are effectively locked by the running program until it relinquishes control. Therefore, an application can update a field in the CWA without using Compare and Swap (CS) instructions or locking (enqueuing on) the resource. Specifying Quasirent on the program definition COncurrency attribute is supported for all executable programs.

9.27.2 Overview of CICS Open Transaction Environment

CICS Open Transaction Environment (OTE) is an architecture that was introduced mainly for three purposes:

- To increase throughput using more concurrency
- To improve performance
- To introduce the possibility to use non-CICS APIs

Prior to OTE, most application code ran under the main CICS TCB (Task Control Block) called the quasi-reentrant TCB. The CICS dispatcher sub-dispatches the use of the QR TCB between the various CICS tasks. Each task voluntarily gives up control when it issues a CICS service, which then can cause a CICS

dispatcher wait. Only one CICS task can be active at any one time on the QR TCB. But the one and only QR TCB can only execute on one CPU, so CICS execution only used one physical CPU at a time. For that reason, the limit of a specific CICS system's execution capacity was set by the MIPS size of the single CPU of the related MVS system.

OTE introduces a new class of TCB, which can be used by applications, called an open TCB. An open TCB is characterized by the fact that it is assigned to a CICS Task for its sole use, and multiple OTE TCBs can run concurrently in CICS. There are several modes of open TCBs that are used to support various functions, such as Java in CICS, open API programs, and C and C++ programs, which are compiled with the XPLink option.

There is no sub-dispatching of other CICS tasks on an open TCB. The OTE introduces a lot of new engines to CICS program execution. Each new TCB can execute on one CPU in parallel (concurrently), which gives the potential of increased throughput for a single CICS system, as long as the necessary CPU power is present. When an application makes a request that involves the use of an open TCB, CICS first tries to find a suitable TCB that is available for reuse in the appropriate pool of open TCBs. CICS can match a request with an available TCB of the correct mode only if the TCB has matching attributes. CICS attaches a new TCB if it cannot find a suitable match with a free TCB, provided that the MAX number of TCBs for the pool was not reached.

If CICS cannot find a suitable match, and the MAX number of TCBs for the pool is reached, CICS might fulfil the request by destroying a free TCB that has the wrong attributes and replacing it with a TCB that has the correct attributes. This technique is called stealing. Stealing can be costly on performance, depending on the type of open TCB, so CICS avoids it where it makes sense to do so. For L8 mode TCBs, used by task-related user exits enabled with OPENAPI, the cost of stealing a TCB is low, so CICS always steals a TCB if it receives a request for which it cannot find a suitable match with a free TCB or attach a new TCB. However, for J8 and J9 mode (JVM) TCBs, the cost of TCB stealing is high because CICS must destroy and reinitialize the JVM and the TCB, so CICS has a selection mechanism to decide if stealing the TCB is worthwhile or if the request will be made to wait. CICS maintains statistics of excess TCB management and TCB stealing activities.

9.27.3 Details of being threadsafe

In the previous section, we noted that OTE essentially created a number of new TCBs that can be used to carry out work. Each new TCB represents a thread where a CICS program can execute in parallel. When the CICS program continues to execute on the open TCB, it is called a threadsafe execution of the program. The result is a reduced number of TCB switches between the open

TCB and the QR TCB. This, in turn, results in reduced CPU consumption, corresponding to the number of saved TCB switches. The more CICS commands that are made threadsafe, the more probability you will remain executing on the open TCB. The CICS QR TCB provides protection through exclusive control of global resources only if all user tasks accessing those resources run under the QR TCB. It does not provide automatic protection from other tasks that execute concurrently under another (open) TCB.

Programs are said to be quasi-reentrant programs because they take advantage of the behavior of the CICS dispatcher and the QR TCB—in particular there is only ever one CICS task active under the QR TCB. This means that although the same program can be being executed by multiple CICS tasks, only one of those CICS tasks is active at any given point. Compare this with a situation in which multiple instances of the same program are each executing under a separate TCB. In this scenario, multiple tasks are active in the same program at the same time, and the program must be fully MVS reentrant at the least. For a program to be threadsafe, it must go beyond being fully reentrant and use appropriate serialization techniques when accessing shared resources. Quasi-reentrant programs always run under the QR TCB and can access shared resources, such as the Common Work Area (CWA) or shared storage obtained using EXEC CICS GETMAIN SHARED safe in the knowledge they are the only CICS user task running at that point. This is because running under the QR TCB guarantees serialized access to those shared resources. An example is a program that updates a counter in the CWA. The program is sure to be alone to update this counter, and when it stops or gets suspended by the CICS dispatcher, it is sure to know that the counter still has the value that was assigned.

It is important to understand that a single program operating without the agreed-upon serialization technique can destroy the predictability and therefore the integrity of an entire system of otherwise threadsafe programs. Therefore, an application system cannot be considered threadsafe until all programs that share a common resource implement that application's threadsafe standards.

Essentially, when an application is threadsafe it:

- Uses appropriate serialization techniques, such as Compare and Swap or enqueue, when accessing any shared application resources. It must be capable of running concurrently on multiple TCBs and must not rely on quasi-reentrancy to serialize access to shared resources and storage.
- Uses no shared application resources whatsoever.
- Incorporates threadsafe application logic (which means that the native language code in between the EXEC CICS commands must be threadsafe).
- Is defined to CICS as threadsafe.

For more details about threadsafe, how to enable threadsafe in your environment, or just more details about how it works, see the IBM Redbooks Publication *Threadsafe considerations for CICS*, SG24-6351.

9.28 Introduction to the system programming interface

The CICS system programming interface (SPI) commands are for managing the CICS system and its resources, in contrast to the application programming interface (API) commands, with which you implement end-user applications. SPI commands either retrieve information about the system and its resources or modify them. System programming commands are supported in the same way that application programming commands are supported. They can be used in programs written in any CICS-supported language. However, there are some differences between SPI and API commands:

- You cannot function ship SPI commands
- Additional security checking is available for SPI commands
- Programs containing SPI commands must be translated with the SP translator option

Remember Figure 9-5 on page 285, where we introduced the compiler option CBL CICS to make the language compiler aware to invoke the CICS translator for every CICS command it finds? If your program contains CICS SPI commands, you must expand this option to CICS (SP); otherwise, the CICS translator issues an error message indicating that the required option is not specified.

Let us assume that we will only issue the HelloWorld message when the sample FILEA is closed. Figure 9-49 on page 352 shows the slightly-changed sample program with the appropriate SPI command to close the file and the corresponding compiler sub option SP.

```
CBL CICS('SP')
IDENTIFICATION DIVISION.
PROGRAM-ID. HIWORLD.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  HELLOWORLD PIC X(19) VALUE IS '*** HELLO WORLD ***'.
PROCEDURE DIVISION.
    EXEC CICS SET FILE('FILEA') CLOSED END-EXEC
    EXEC CICS SEND TEXT FROM(HELLOWORLD) END-EXEC
    EXEC CICS RETURN END-EXEC
    GOBACK.
END PROGRAM HIWORLD.
```

Figure 9-49 CICS program including a SPI command and compiler sub option SP

All SPI commands are documented in the CICS System Programming Reference.

There is a third program interface available in CICS, the *user exit programming interface* (XPI). The user exit programming interface provides global user exit programs with access to some CICS services. It consists of a set of macro function calls that you can use in your user exit programs. There are a number of exits where you can use both EXEC CICS commands and XPI calls. XPI calls are used by CICS system programmers only.

9.29 Language Environment

Language Environment provides a common set of runtime libraries. Before the introduction of Language Environment, each of the high-level languages (HLLs) had to provide a separate runtime environment. This was bad because intercommunication between programs written in different programming languages was problematical. With Language Environment, you use one runtime environment for your applications, regardless of the programming language or system resource needs because most system dependencies were removed.

You can mix all the languages supported by CICS in a single application. The same Language Environment callable services are available for all programs. System messages from programs written in several languages are all sent to the same output destination. Because of these advantages, high-level language support under CICS requires Language Environment.

The CICS programming guidance documentation requires that your CICS system is using the services of Language Environment, which provides a

common runtime environment for IBM implementations of assembler and those high-level languages (HLLs) supported by CICS, namely COBOL, PL/I, C, and C++.

Most of the compilers supported by CICS and Language Environment are Language Environment-conforming compilers, meaning that programs compiled by these compilers can take advantage of all Language Environment facilities that are available to a CICS region.

The native runtime libraries provided with pre-Language Environment compilers are not supported. Language libraries, other than the Language Environment libraries, must not be present in your CICS startup JCL.

When modifying existing application programs or writing new programs, you must use a compiler that is supported by Language Environment. This requires that your application programs are link-edited using the Language Environment SCEELKED library, which in turn means that the resulting application load module can execute only under Language Environment. See 9.5.2, “Link-editing a CICS program” on page 293.

9.30 Summary

This chapter describes the programming languages used in CICS in which programming techniques are used to make CICS applications. We summarize the chapter:

- Created sample HelloWorld program in COBOL running in CICS sending plain text to a 3270 panel.
- Changed the plain text to formatted text using Basic Mapping Support.
- Introduced the pseudo-conversational programming style and gave examples.
- Split the sample into presentation and business logic, and provided the business logic in PL/I while the presentation logic stays COBOL.
- Provided the business logic in PL/I, Assembler, and Java.
- Used a COMMAREA and a channel to provide communication between programs.
- Exposed the business logic program as a web service.
- Invoked the web service from a CICS application written in COBOL.
- Invoked the web service from a Java program running in a web server.
- Provided a sample application accessing CICS to get back a .gif file.

Part 5

CICS Topologies, and Architecture

In this part of the book, we look at typical CICS topologies and provide brief insight into CICS architecture.

10

CICS topology

Topology is an inventory of CICS and CICSplex SM resources and a map of their relationships. CICSplex SM supports the definition of resource and system topology.

It is important to learn about the topology of CICS to understand concepts of the internal configurations that are used in a CICS system.

Resource definitions are descriptions of the types of applications and procedures that are used for individual data actions. Every CICS region has its own set of these resource definitions stored in a data set called the *CICS system definition* (CSD).

Resource definitions provide CICS with the information to recognize and manipulate data appropriately and the information regarding the properties and interactions between resources. If a resource is not defined correctly in CICS, it is not recognized and can cause errors or transaction failure.

In CICSplex architecture, CICS is divided into several regions, and each region can play a distinct role in the system. Each region contains its own CSD that details the resources that are used exclusively (or owned) by that region. CICS regions are distributed across z/OS systems. A CICS region is a named instance of CICS Transaction Server that runs its own z/OS address space. A z/OS address space is a location that corresponds to physical or virtual memory. A CICS region can be started and stopped independently of other CICS regions.

Figure 10-1 shows a topology view that aims to show how a CICS system is constructed.

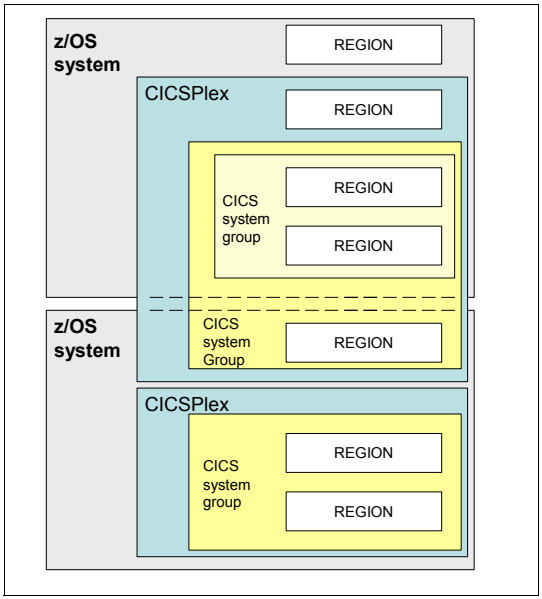


Figure 10-1 Basic CICS topology

CICS regions can be grouped into CICS system groups on a CICSplex. Although you can define a CICS system to only one CICSplex, you can assign a CICS system to multiple CICS system groups. You can also assign a system group to any number of other system groups, provided they are on the same sysplex. CICS system groups identify subsets of the CICS system and, if you are not using workload management facilities, you can combine CICS systems and CICS system groups in a number of different ways. For example, you might associate systems in the following ways:

- Function: CICS systems that are application-owning regions (AORs), file-owning regions (FORs), or terminal-owning regions (TORs).
- Applications: CICS systems serving as AORs, FORs, and TORs that are used by a specific application or group of applications.
- Time period: CICS systems that are normally active at the same times.

The term sysplex is used quite often when talking about mainframe systems. A sysplex is a set of z/OS systems.

10.1 Intercommunications

As customer environments grew, so did the need for additional CICS regions. As a result, resources, such as files, were spread out among multiple regions. This creates the need for CICS to communicate with other systems to access data or initiate requests. In this section, we discuss the various ways CICS communicates with other systems. This is referred to as CICS *intercommunication*.

10.1.1 Functions

First we want to touch on the various intercommunication functions provided by CICS:

- Transaction routing allows a transaction and an associated terminal to be owned by several CICS systems.

  `For example: A terminal user on region A issues transaction ABCD which is defined to run on region B. This transaction will be sent to region B to execute.`

- Function shipping in CICS allows an application program access a resource owned by, or accessible to, another CICS system.

  `For example: A transaction wants to read/write a file owned by another system`

- Distributed program link (DPL): CICS distributed program link enables CICS application programs to run programs residing in other CICS regions by shipping program-control LINK requests.

  `For example: A user can create a program that can link to another program on a remote region.`

- Asynchronous processing allows a CICS transaction to initiate a transaction in a remote system and to pass data to it. The remote transaction can then initiate a transaction in the local system to receive the reply.

  `For example: A transaction needs to signal a remote system that it should start a particular transaction.`

- Distributed transaction processing (DTP): The technique of distributing the functions of a transaction over several transaction programs within a network is called distributed transaction processing (DTP). DTP allows a CICS transaction to communicate with a transaction running in another system.

For further information regarding DTP, review the following link:

http://publib.boulder.ibm.com/infocenter/cicsts/v4r1/topic/com.ibm.cics.ts.intercommunication.doc/topics/dfht1ko.html

10.1.2 Methods

In the 1978-1980 time frame, CICS delivered its support for Multiple Region Operation (MRO) and Intersystem Communication (ISC), enabling you to easily distribute applications and data resources across multiple locations. This allowed customers to overcome the early storage limitations of virtual systems and enabled the customer to have multiple, concurrent units of CICS-related work running on multiple machine processors (multi-processors).

10.1.3 Multiregion operation

Multiregion operation (MRO) enables CICS systems that are running in the same MVS image, or in the same MVS sysplex, to communicate with each other. MRO does not support communication between a CICS system and a non-CICS system, such as Information Management Systems (IMS). The support within CICS that enables this region-to-region communication is called *interregion communication* (IRC). IRC can be implemented in three ways:

- Through support in CICS terminal control management modules and by use of a CICS-supplied interregion program (DFHIRP) loaded in the link pack area (LPA) of MVS. DFHIRP is invoked by a type 3 supervisor call (SVC).
- By MVS cross-memory services, which you can select as an alternative to the CICS type 3 SVC mechanism. In this case, DFHIRP is used only to open and close the interregion links.
- By the cross-system coupling facility (XCF) of IBM. XCF is required for MRO links between CICS regions in different MVS images of an MVS sysplex. It is selected dynamically by CICS for such links, if available.

  The Benefits of XCF/MRO are:

  – A low communication overhead between MVS images, providing much better performance than using ISC links to communicate between MVS systems. XCF/MRO thus improves the efficiency of transaction routing, function shipping, asynchronous processing, and distributed program link across a sysplex. (You can also use XCF/MRO for distributed transaction processing, provided that the LUTYPE6.1 protocol is adequate for your purpose.)

  – Easier connection resource definition than for ISC links, with no VTAM tables to update.

- Good availability, by having alternative processors and systems ready to continue the workload of a failed MVS or a failed CICS.
- Easy transfer of CICS systems between MVS images. The simpler connection resource definition of MRO, and having no VTAM tables to update, makes it much easier to move CICS regions from one MVS to another. You no longer need to change the connection definitions from CICS MRO to CICS ISC (which, in any event, can be done only if CICS startup on the new MVS is a warm or cold start).
- Improved price and performance.

Figure 10-2 shows a MRO connection between CICS regions in a single logical partition of z hardware (LPAR) and a MRO connection using MRO/XCF between CICS regions on separate LPARs that are in the same sysplex.

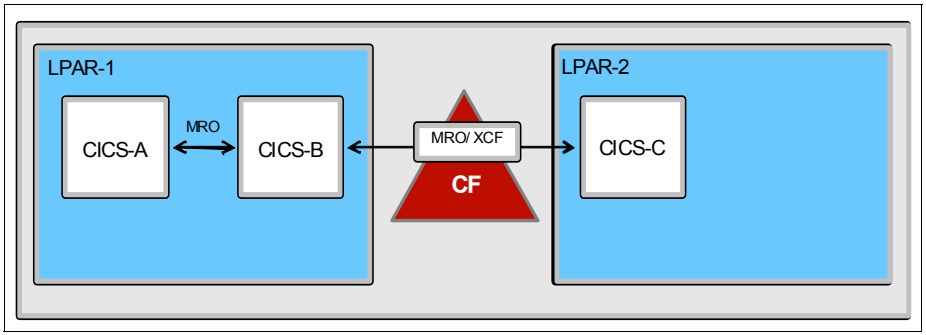


Figure 10-2 Example of MRO configuration

Note: The Store recently added CICS regions to manage their online ordering and catalog systems. They consist of:

- Two TORs to support point-of-sale and management terminals and online catalog and ordering systems
- Three cloned AORs to run their application code.

Because these are in the same LPAR, they use MRO connections between the TORs and the AORS, as shown in Figure 10-3.

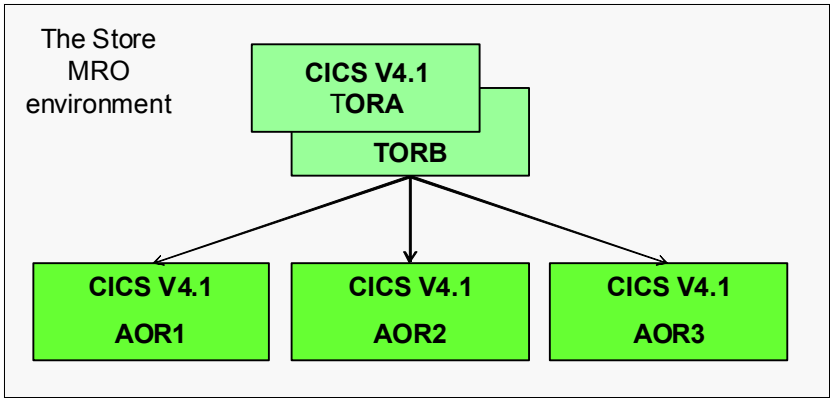


Figure 10-3 Example of the Store MRO environment

10.1.4 Intersystem communication

For communication between CICS and non-CICS systems or between CICS systems that are not in the same operating system or MVS sysplex, a SNA access method, such as VTAM, can be used to provide the necessary communication protocols. Communication between systems through SNA is called *intersystem communication* (ISC). This form of communication can also be used between CICS systems in the same operating system or MVS sysplex, but MRO provides a more efficient alternative.

The SNA protocols that CICS uses for intersystem communication are:

- Advanced Program-to-Program Communication (APPC), otherwise known as Logical Unit Type 6.2 (LUTYPE6.2), which is the preferred protocol.
- Logical Unit Type 6.1 (LUTYPE6.1), which is used mainly to communicate with IMS systems

Figure 10-4 on page 363 shows an example of a configuration using MRO and ISC connections:

- CICS regions A and B are connected through MRO.
- Since LPAR-1 and LPAR-2 are part of the same sysplex, CICS regions B and C are connected through MRO/XCF through the MVS coupling facility.
- LPAR-3 is not part of the sysplex, therefore CICS regions B and C are connected to D through VTAM/ISC.

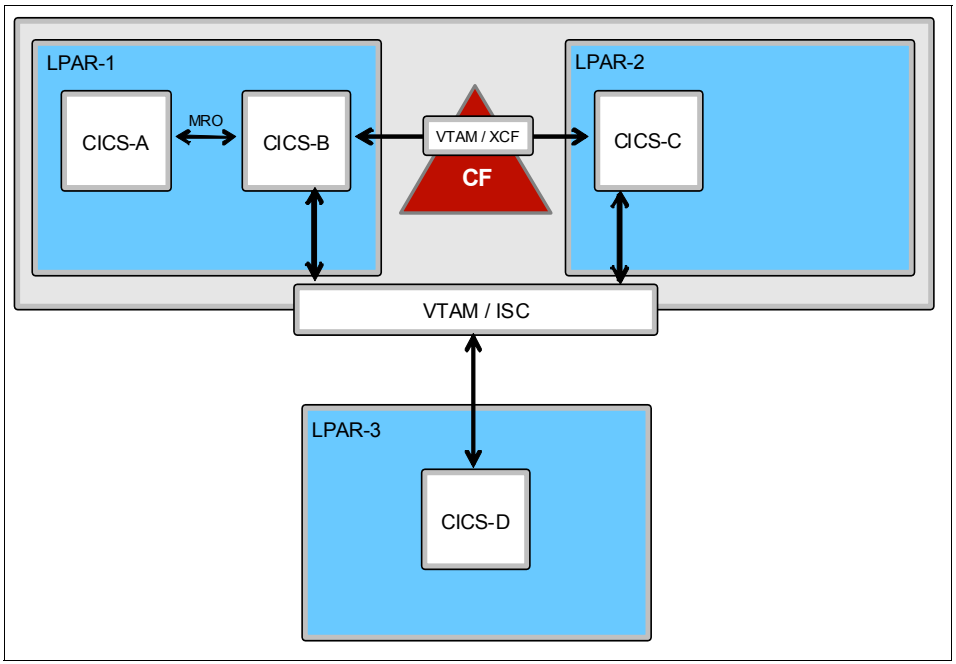


Figure 10-4 Example of MRO, ISC, and XCF connected regions

10.1.5 Terminal-owning region

A *terminal-owning region* (TOR) is a CICS region that looks after the end-user network facilities and needs. The CICS system definition or this region has definitions for all resources used in terminal sessions, including monitors, printers and communication devices. When you define transactions to CICS, you can define them as local or remote transactions. Local transactions always run in the TOR, that is, in the CICS region to which the terminal initiating the transaction is directly logged on. Remote transactions are routed to another CICS region connected to the TOR by multiregion operation links or routed to other CICS regions that are connected by CICS intersystem communication links.

Figure 10-5 on page 364 shows a simple CICS system with a terminal-owning region connected to a single application-owning region by MRO.

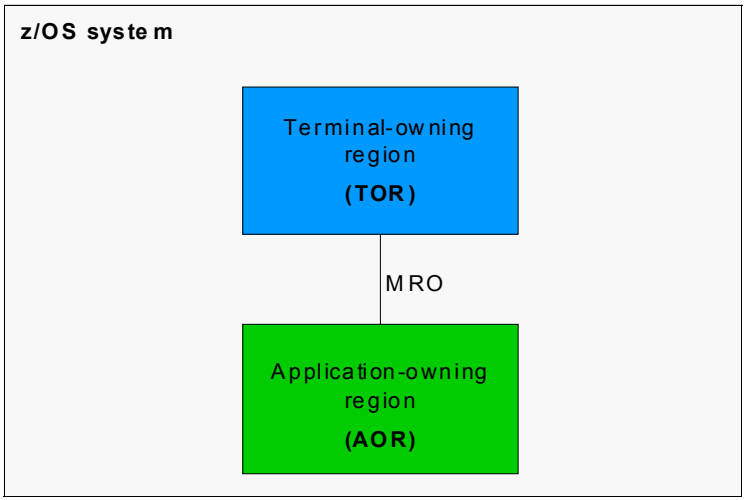


Figure 10-5 A simple CICS system with a terminal-owning region connected to a single application-owning region by MRO

When you define transactions to CICS, you can define them as local or remote transactions. Local transactions always run in the terminal-owning region, in the CICS region to which the terminal initiating the transaction is directly logged on. Remote transactions are routed to another CICS region connected to the terminal-owning region by multiregion operation links.

CICS supports two forms of transaction routing for remote transactions:

- Static transaction routing

  The transaction resource definition is predefined with the name of a remote CICS region to which it is to be routed for execution. Transactions defined in this way are always routed to the specified remote CICS region. With this method, you can change the remote destination of the specified transaction only by manually altering and reinstalling the transaction resource definition.

- Dynamic transaction routing

  You do not need a transaction resource definition in the terminal-owning region to specify transaction routing. When a transaction is invoked at a user's terminal, CICS searches the table of installed transaction definitions for the name of the transaction. If CICS does not find an entry in the table, CICS automatically calls the dynamic transaction routing program to determine whether it should be routed to a remote CICS region. This is the recommended method for using dynamic transaction routing.

Alternatively, you can use transaction resource definitions for each individual transaction. For transactions defined with DYNAMIC(YES), CICS invokes the dynamic transaction routing program to determine where to route the transaction.

You can specify both a remote system name and dynamic transaction routing. In this case, CICS still invokes the dynamic transaction routing program, allowing the defined remote system name to be dynamically changed.

Using dynamic transaction routing across a CICSplex, compared with static routing, can improve performance and improve availability for terminal end users

If you are operating with combined terminal-owning/application-owning regions (TOR/AORs) you are recommended to split them into separate terminal- and application-owning regions. For example, if your combined CICS regions are organized to support different applications as shown in Figure 12, you should reconfigure them as illustrated.

10.1.6 Application-owning region

The application-owning region owns user transactions and application programs. The CICS system definition for this region includes definitions for the resources relating to transactions and application programs. Transaction requests received by a terminal-owning region are passed to an AOR, also known as a target region, for processing. Assuming that the terminal-owning region performs only transaction routing functions, which are only a fraction of the work done by an application-owning regions, one terminal-owning region should be able to serve between 10 and 20 application-owning regions.

Figure 10-6 on page 366 shows a single AOR with multiple AORs.

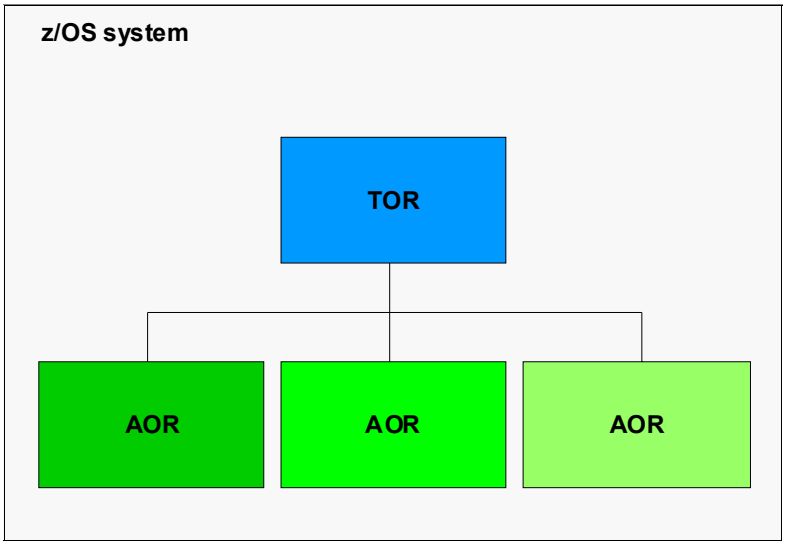


Figure 10-6 A single AOR with multiple AORs

An AOR, or a target region, is a CICS region that manages application programs. Requests for data are passed to a *data-owning region* (DOR). A DOR is a CICS address space that manages access to files and databases.

10.1.7 Web-owning region

Requests sent over the internet are supported by web servers. WebSphere is an example of a web server. A web services provider performs a specific function. It does not provide generalized functionality. All requests that a web server provider performs can be executed in a web server, but not all functionality of a web server is available in a web services provider. CICS can act as a web services provider because it has some of the functionality of a web server.

A web service request, such as a car insurance quote web form being submitted, arrives into a *web-owning region* (WOR) through HTTP. The z/OS Communications Server supports many CICS regions that have the same TCPIPSERVICE definition, which defines the port. Web-owning regions have similar functions to TORs excepts WORs contain the following:

- TCPIPSERVICE (listener socket): Receives the web service request
- URIMAP: Checks that the request is a known request
- PIPELINE: Used in the alias transaction

The alias transaction runs in another CICS region in a user security context. It deciphers the XML using XMLSS parser, producing a set or CONTAINERs

containing the XML data and a ciphered version of the contents. Figure 10-7 shows how a web service request is processed.

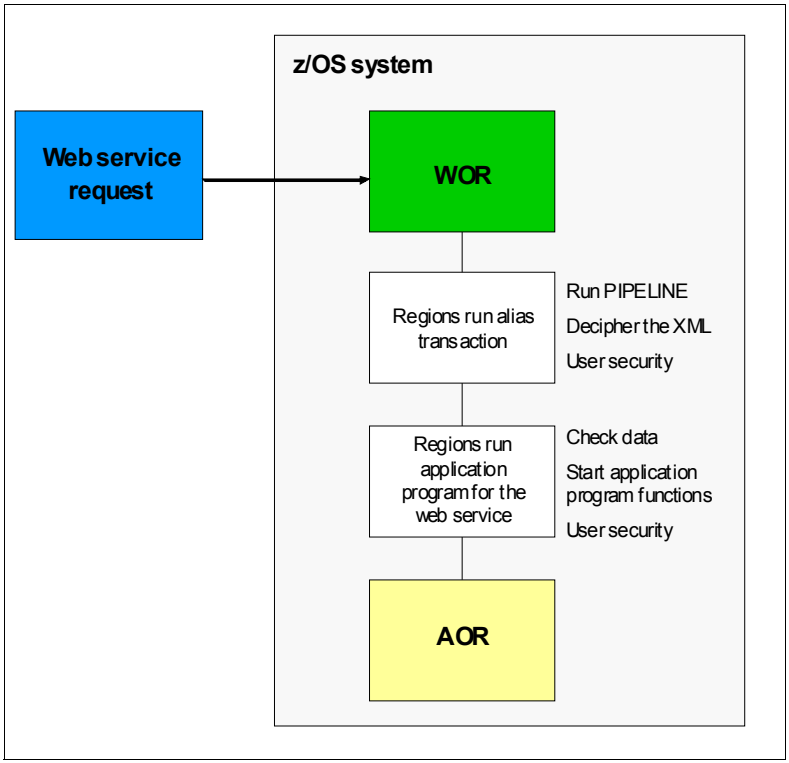


Figure 10-7 Processing a web service request

Request processing by the alias transaction occurs through a routed application program (defined in the PIPELINE_ that accesses the unravelled XML request and uses this information to access existing CICS application programs through the AOR.

10.1.8 Resource-owning region

CICS resource-owning regions (RORs) own data resources, such as files, temporary storage queues, and transient data queues. The RORs can also have access to database managers, such as DB2 and DBCTL.

There are two main types of data-owning region:

- File-owning region (FOR)
- Data-owning region (DOR)

A data-owning region is a CICS address space that manages files and databases. A file-owning region is a specific type of data-owning region that manages files.

FIle-owning regions are needed to provide shared access to those data sets for which RLS is not suitable or not supported, for example, BDAM data sets.

With files defined as remote CICS files, the application-owning region's function ship CICS file control requests to the remote region that is defined as owning the files. This enables CICS to operate a form of data sharing, using the CICS file control facility.

CICS file control manages its own record locks to ensure a record can be accessed by only one file request at a time, using its own record locking mechanism. A CICS file owning region runs tasks under the quasi-reentrant TCB with some limited subtasking under the concurrent TCB. It also manages data integrity in the event of failure, ensuring that uncommitted updates are backed out.

Function shipping allows you to define VSAM and BDAM data files as remote resources. This allows application programs to request data set services from a connected CICS region where the data sets are physically defined.

The principal reason for function shipping CICS file control requests is to enable more than one CICS region to have access to VSAM data that would otherwise be available to only a single CICS region. This is because, for data integrity, only one CICS region can have a data set open at any one time. in a dynamic transaction routing environment you need the ability to access VSAM data from a number of application-owning regions.

Figure 10-8 on page 369 shows one FOR in a CICSplex possibly causing single point failure.

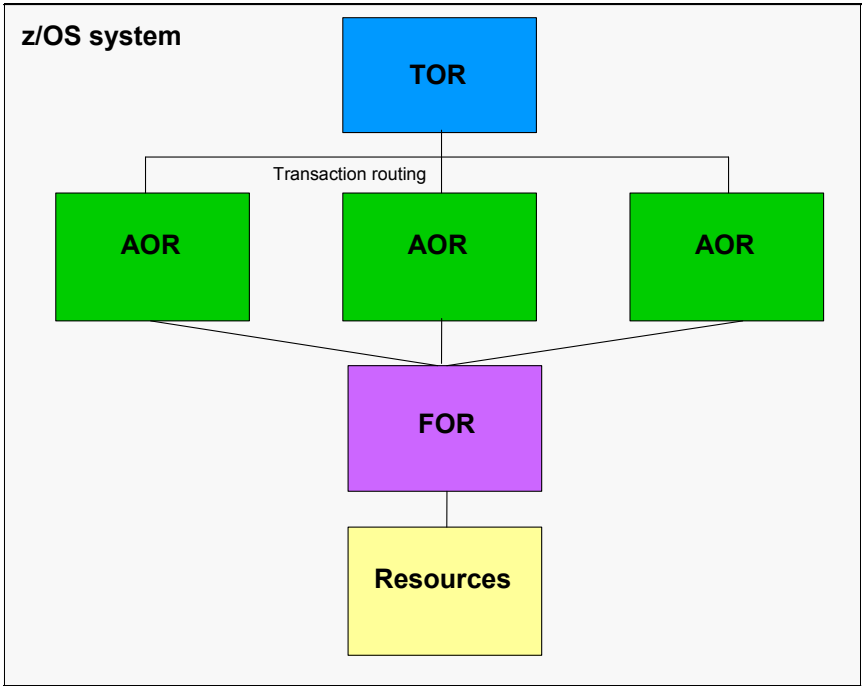


Figure 10-8 One FOR in a CICSplex possibly causing single point failure

If a CICSplex is configured with only one file-owning region, that region becomes a potential single point-of-failure. For example, if all the data sets required by three application-owning regions were owned by a single file-owning region, a failure of that file-owning region causes the failure of all the applications. Separating the ownership of the files, where possible, into different file-owning regions ensures that the loss of one file-owning region does not cause the loss of all applications.

10.1.9 Queue-owning region

The queues that are needed by CICS to pass data and internal state information are added when a region is created. These queues include the temporary storage queues (TSQ), transient data (TD) queues, and internal queues that CICS uses for auxiliary temporary storage queues and intrapartition data queues.

Temporary storage is a CICS facility that allows application programs to store data in a temporary storage queue for later retrieval.

Transient data is a CICS facility that provides the ability to read and write data in sequential queues. Transient data is created within an application session. At the end of the session, it is discarded or reset back to its default and not stored in a database.

Queues are data structures for processing work in which the first element added to the queue is the first element processed.

If there are recoverable temporary storage queues (which are not supported by CICS temporary storage data sharing servers), that must be accessible through all AORs, the solution is to include a queue-owning region in the CICSplex. Defining queues to the application-owning region, ensures that they are accessible by any application-owning region using function-shipping requests.

Another form of data queue is the transient data (TD) queue. The dynamic transaction routing considerations for TD queues have much in common with those for temporary storage, and, like temporary storage, not all transient data queues need to be shared. Some queues can be locally owned (when there is no question of inter-transaction affinity), or queues can be replicated.

To enable transactions sharing a TD queue to be dynamically routed to a set of application-owning regions, you must ensure that the TD queues are globally accessible to those application owning regions.

There are two common types of database; relational and hierarchical. CICS supports both types of database through DB2 and IMS.

The DB2 family, for example, DB2 for ESA and DB2 for VSE, includes the most common relational database products found in mainframe environments with CICS TS. CICS provides both attachment interfaces for DB2 and monitoring and control services. The DB2 data manager supplies utilities that enable programmers to write data access commands in Structured Query Language (SQL) with the structure EXEC SQL. CICS TS has an attachment facility that allows you to operate DB2 with CICS. Through this facility, CICS applications can access DB2 data while operating within the CICS TS environment. The DB2 system can be shared by several CICS systems through the DB2 attachment facility, but an individual CICS system can be connected to only a single DB2 system at a time. CICS application programs make requests to DB2 in the form of SQL statements.

DB2 data sharing, a DB2 subsystem in CICS needs a CICS application-owning region with each application-owning region connected to the DB2 in its z/OS system.

The language used for data manipulation in IMS is Data Language 1 (DL/1). DL/1 enables the definition of data structures and the relation of structures to

applications, and the loading and reorganizing of these structures. DL/1 can enable application programs to retrieve, replace, delete, and add segments to databases. In many CICS systems, CICS applications interface with the IMS Database Control (DBCTL) facility, as shown in Figure 10-10 on page 372.

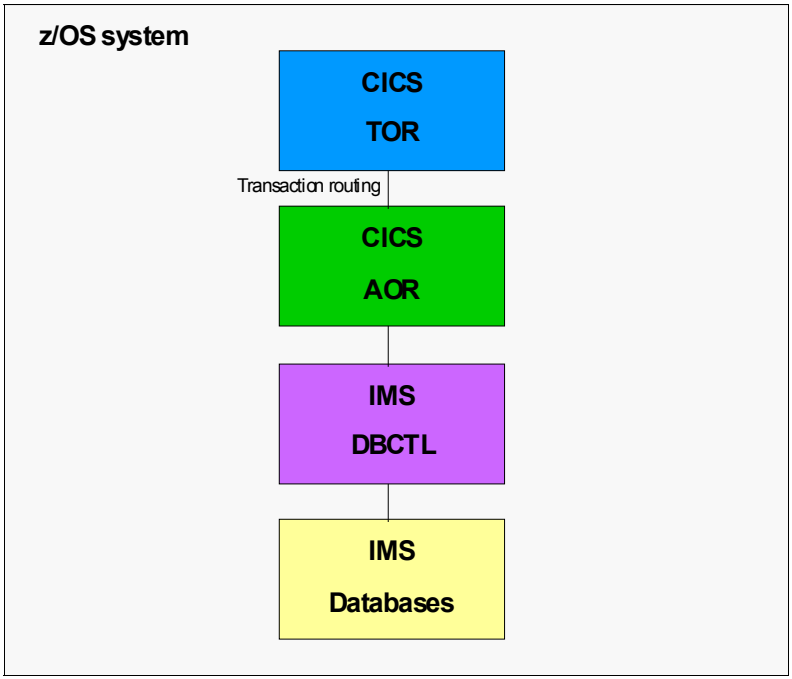


Figure 10-9 CICS access to IMS databases through the IMS DBCTL interface

For IMS data sharing, an IMS DBCTL environment on each z/OS system that has CICS application-owning regions, with each application owning region connected to the DBCTL in its z/OS system.

10.1.10 Example company: CICS system topologies

This chapter has so far described the components of a CICS system topology. The topology of the example companies are described to demonstrate real-world setups of CICS.

The Bank used CICS for many years. Originally CICS was used to drive teller's terminals in branches. With the increased popularity of internet banking, they are observing a greater transaction volume entering through TCP/IP.

The Bank's core banking configuration consists of:

- A TOR on each LPAR (TB01 and TB02) which listens on a shared TCP/IP port:
  - CICS TOR1 on TB01
  - CICS TOR2 on TB02
- Transaction requests that are dynamically routed to cloned AORs:
  - CICS AR10, CICS AR11, and CICS AR14 on TB01
  - CICS AR12, CICS AR13, and CICS AR15 on TB02
- CICS core banking programs that run in the AORs share access to the customer and accounts operational database using DB2 data sharing.
- A WUI server, CICSWUI1, on LPAR TB01
- CICS Explorer for CICSplex systems management

A map of the bank topology is shown in Figure 10-10.

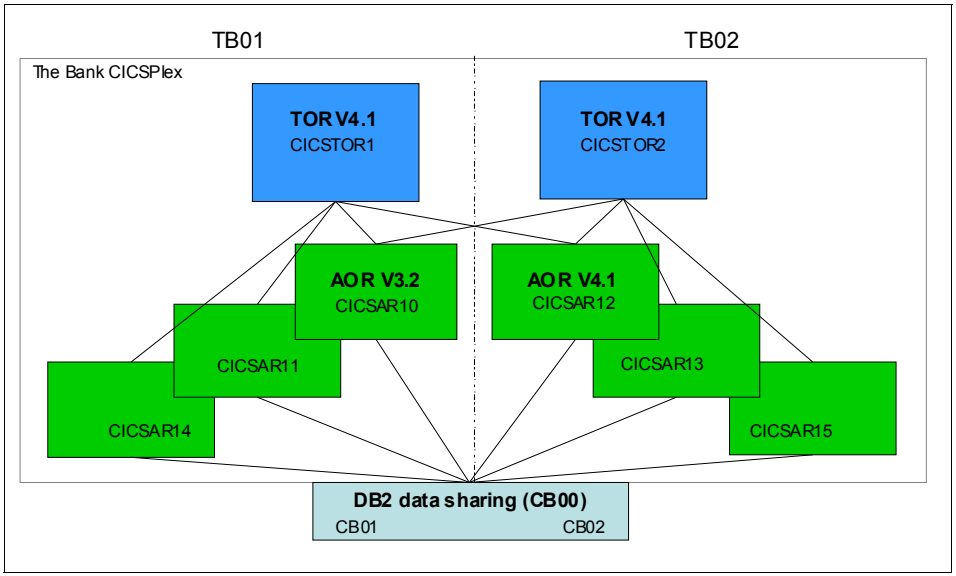


Figure 10-10 Topology of the example bank

The Store invested heavily in technology to manage their online catalog and ordering systems.

The Store's CICSplex currently runs in a single LPAR. Their configuration consists of:

- Two TORs

- Three cloned AORs access databases managed by a DB2 subsystem.

The Store's CICSplex is shown in Figure 10-11.

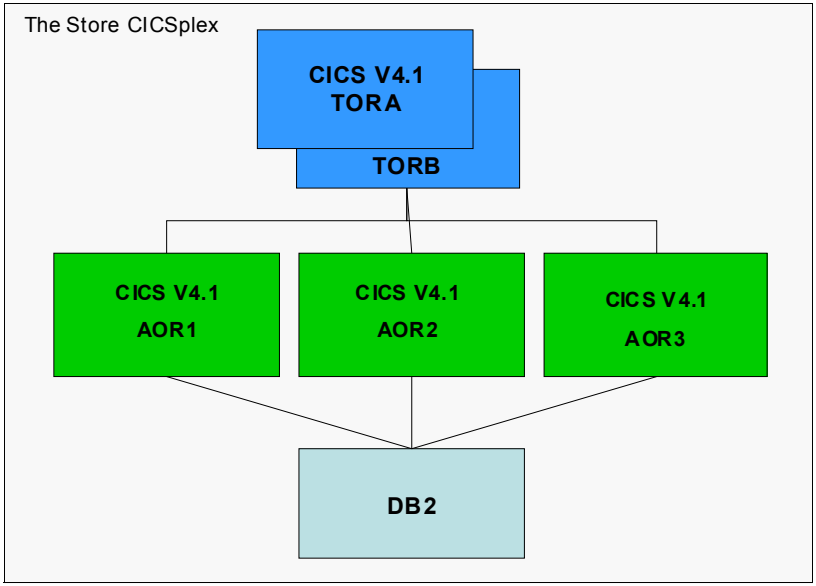


Figure 10-11 Topology of the example store

The benefit of having two TORs is to support point-of-sale and management terminals and online catalog and ordering systems. If one TOR fails, the business system can still function. The AORs access databases managed by a DB2 subsystem. The databases contain data, such as customer details, order histories, and products.

11

CICS architecture

In this chapter, we describe the CICS system architecture and the topology of CICS using the example companies to demonstrate particular CICS configurations. This chapter also looks at the personnel needed to develop and maintain a functioning CICS system.

11.1 CICS architecture

When building a house, the first step is to create a blueprint, grouping each room by function and placing structural items within each room according to their best utility. This is a rational approach called architecture. In the same way as designing a house, a system must be designed to group resources and functions to achieve maximum efficiency.

To explain the architecture of CICS, it is important to first understand CICS components. User application programs communicate with CICS management modules to handle terminal I/O, file I/O, program loading and control, and access to other system resources.

CICS system resources are defined in CICS system tables using resource definition online (RDO). Before you can use CICS, information about the resources it should use, the properties of the resources, and how resources are to interact with each other must be defined. Examples of resources are:

- Connections
- Files
- Journals
- Programs
- Terminals
- Databases

RDO uses the CICS-supplied online transactions CEDA, CEDB, and CEDC. Definitions are stored on the CICS system definition (CSD) file and are installed into an active CICS system. Because RDO is used while CICS is running, it allows fast access to resource definitions, and it can also provide an offline facility to inspect or change resource definitions.

The management modules use CICS system tables to obtain information about the terminals, files, and application programs that are part of the CICS system. The tables are loaded into storage when the CICS system is started, and remain active until the system is shut down.

Management modules create *control blocks* to keep track of the status of all transactions that are currently being processed. The control blocks are released when they are no longer needed, for instance when the task exits the system.

Management modules and application programs use the *CICS system data sets* for transaction logging, system recovery, and storing processing results that are needed by other transactions or application programs.

At the top level, CICS is organized into *domains*.

11.1.1 Domains

A domain is a functionally isolated area of a CICS system that owns resources to which it has sole access. Each domain is a single major component of a CICS system, except for the application domain which contains several components. CICS regions are divided into a set of domains, each containing a set of objects grouped together to perform a common set of functions. Domains contain management modules, tables, and control blocks. A small set of domains handles a large part of the transaction processing cycle. Three domains supervise and control transaction processing:

- Transaction manager (XM): Responsible for receiving transactions requests, and creating and organizing tasks to process transaction requests. Organizes all tasks currently in the system.
- Program manager (PG): Responsible for locating and invoking application programs required for processing transactions.
- Storage manager (SM): Responsible for allocating memory resources required for transaction processing, such as the storage used for programs, I/O areas and workspace (storage remaining after CICS has been loaded).

Figure 11-1 shows a typical domain structure.

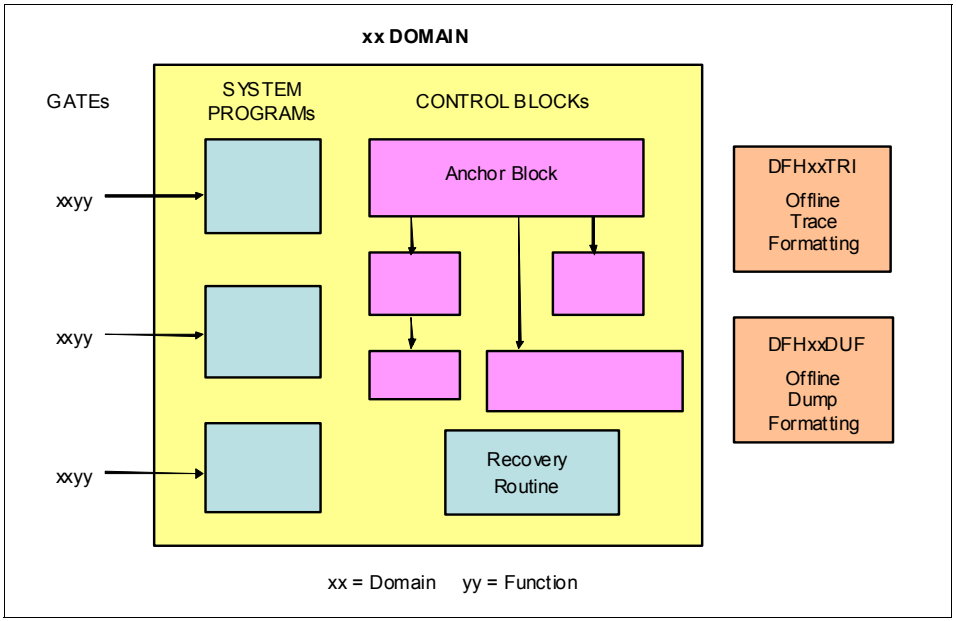


Figure 11-1 Typical domain architecture

A domain is the packaging of functional areas in CICS, encapsulating code, control blocks, and resources. Management of resources is performed by the code and control blocks in a domain; however, a domain and its control blocks do not have to be in the same area. Domains manage storage, programs, catalogs, tasks, and transactions depending on the particular function of the domain.

Each domain in CICS manages its own data. Domains cannot communicate directly with each other, as shown in Figure 11-2. If a domain needs data that belongs to another domain, it must call that domain, and that domain passes data back to the caller's parameter area. Calls can be made only to official interfaces to the domains, and they must use the correct protocols. The request is passed through a *kernel linkage routine* to a particular entry gate to perform the requested service.

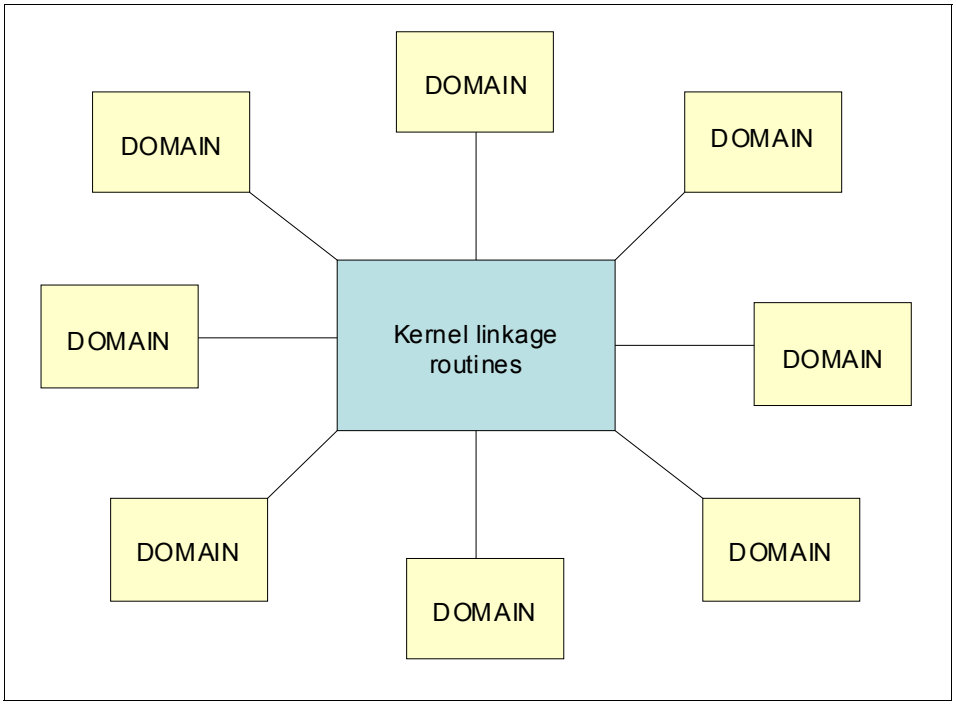


Figure 11-2 Domain communication

There are a number of domains in CICS, each having an important role in a CICS system. The major domains in CICS are:

- Kernel Domain (KE)

  The kernel domain is the central component, depicted in Figure 11-2. KE is involved in every call from one domain to another, providing constant linkage

and recovery in CICS environments. It also manages the pre-initialization of CICS and tracks the existence of domains. System programmers have no external interface to kernel linkage.

- Application Domain (AP)

  Application programs run on the applications domain, and it contains several key components such as application and system services, multiregion operation (MRO) and intersystem communication (ISC). The application domain is responsible for terminal management, file access control, and interface between CICS and application programs.

- Domain Manager (DM)

  The domain manager maintains permanent information about all individual domains through the use of catalog services. It manages initialization and termination tasks for other domains.

- Global Catalog Domain (GC)

  The global catalog is a system data set in which CICS records CICS system information. The global catalog domain, together with the local catalog domain, is a repository used by other domains to hold information to allow an orderly restart. The two catalog domains enable CICS code to read, write, and purge records on the global and local catalog data sets so that a record of the CICS state can be monitored.

  The CICS domains use the global catalog to save their domains status between runs, but the status can be modified during a restart by system initialization parameters. For example, CICS monitoring uses the cataloged status at restart, but the information can be modified by any system initialization parameter. Where no system initialization parameter exists, in the statistics domain for example, the interval time that is stored in the global catalog is always used.

- Local Catalog Domain (LC)

  The local catalog is a system data set that CICS uses to record data used by the internal workings of CICS. The local catalog domain is a repository used by other CICS domains to hold information to allow the domains to re-initialize. The kernel domain, domain manager, and other domains need an individual domain parameter record and these are all stored in the local catalog. The local catalog domain enables CICS code to read, write, and purge records on the local catalog data set. In the case of the dump domain, the local catalog contains the status record to show which transaction dump data set was in use during the previous run. This information is used in a warm or emergency restart by CICS. The domains that write to the local catalog are:

  - Dispatcher domain
  - Dump domain

- Loader domain
- Message domain
- Parameter manager domain
- Storage manager domain

Many other domains exist to provide distinct CICS functions. LMLock manager domain provides both locking and associated queuing facilities for CICS resources.

11.1.2 Gates

A *gate* is an entry point into a domain through that all calls from other domains are made. A gate provides a single service or multiple services. There are three types of gates within CICS, these are specific, generic, and call back gates, as shown in Figure 11-3.

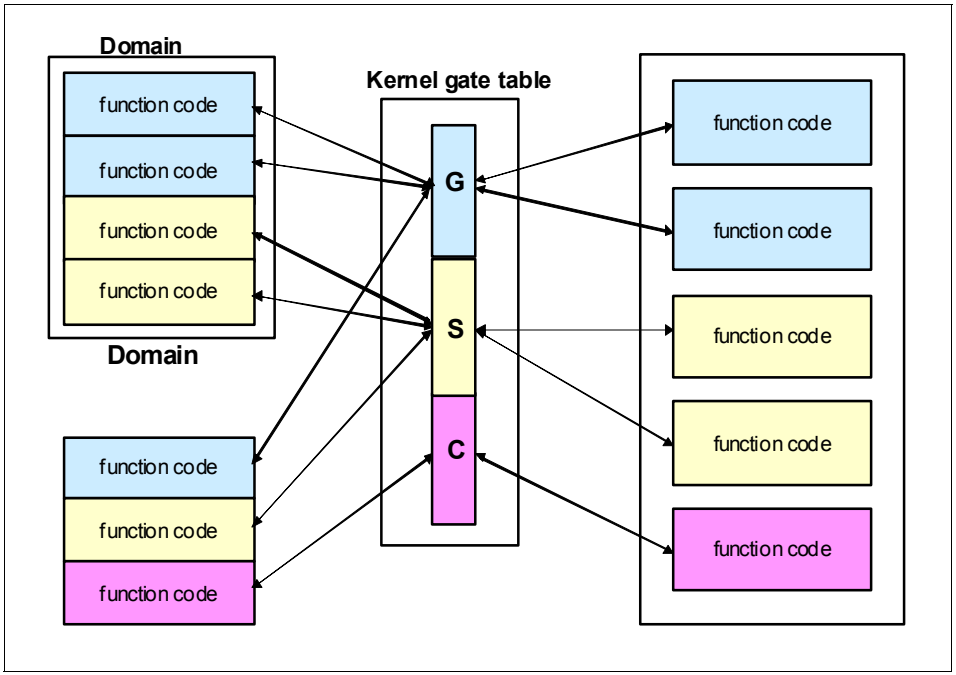


Figure 11-3 Generic, specific, and callback gates

A specific gate gives access to a set of functions that is provided only by a particular domain; however, requests can be made to a single domain by a number of callers. The Dispatcher Domain (DS), for example, has a specific gate (DSAT) that provides CHANGE_MODE and CHANGE_PRIORITY functions and

others that are specific to the dispatcher domain. Only the DS domain provides those functions, but they can be requested by many different callers.

When several domains provide the same set of functions, access is provided by generic gates. Common generic gates include the DMDM gate and the STST gate. DMDM gate performs initialization and termination, and STST performs statistic gathering functions. Most domains contain these two generic gates. Instead of each domain defining call parameters for the generic gates, the Domain Manager (DM) dictates the call format for all DMDM gates in each domain. The Statistics Manager Domain (ST) and Domain Manager (DM) dictate the call formats for the gates, which are always the same for a particular generic gate, but the functions are provided by the other domains.

In the same way as a generic gate, a callback gate gives access to a set of functions that can be provided by several dissimilar domains. Calls to generic gates are broadcast to all domains that have that particular gate; whereas, callback gates use the format owned by the calling domain. For example, a call is made to specific gates only if they registered an interest in sync point processing using the PERFORM_PREPARE function that the Recovery Manager Domain (RM) owns is called by the RM domain. Other domains that have the gate but have not registered an interest are not called.

11.1.3 Task control blocks

A *task control block* (TCB) represents a task on a z/OS address space. Each task runs on a TCB. A TCB is a dispatchable unit of work. It is the CICS dispatcher that dispatches CICS tasks to be run on TCBs.

Before the open transaction environment (OTE) was implemented in CICS (see Chapter 3, “CICS data” on page 103 for more details about OTE), a single TCB was used to run all user applications. A quasi-reentrant program is a program that is in a consistent state when control is passed to it, both on entry, and before and after each EXEC CICS command. Quasi-re entrance guarantees that at each start of an application program is unaffected by previous runs, or by concurrent multi-threading through the program by multiple CICS tasks.

Programs can be defined as quasi-reentrant on the CONCURRENCY attribute of the program resource definition. QR programs are given control by the CICS dispatcher under the QR TCB.

When running under this TCB, no other QR 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 can then be re-invoked for another task, which means the application program can be used concurrently by more than one task, although only one task at a time can be running. QR allows

programs to access globally-shared resources, for example the CICS common work area (CWA), without the need to protect those resources from concurrent access by other programs. Such resources are effectively locked exclusively to the running program until it issues its next CICS request.

Figure 11-4 shows the CICS TCB structure.

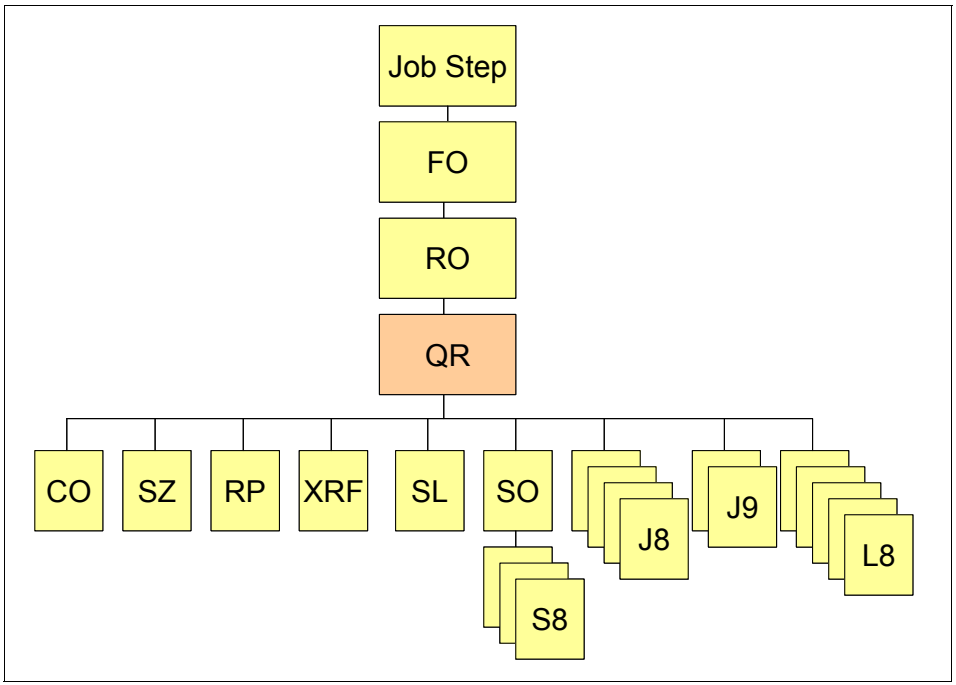


Figure 11-4 CICS TCB structure

Other modes of TCBs that do not run on the open transaction environment areas are:

FO TCB There is always one file-owning TCB, used in CICS file processing, for opening and closing user data sets.

RO TCB There is always one resource-owning TCB that is used for opening and closing CICS data sets, loading programs, and issuing resource access control facility (RACF) requests.

CO TCB The concurrent mode TCB is an optional TCB. It exists only when the system initialization parameter SUBTSKS=1. The CO TCB is used for processes that can

safely run in parallel with other CICS activity, such as VSAM write requests.

With the open transaction environment, CICS application code can use non-CICS services inside the CICS address space without interference from other transactions. Applications that use the OTE can run on open TCBs rather than the QR TCB. The open TCBs that are available are:

J8 and J9 TCBs The J8 and J9 TCBs run Java programs under the Java Virtual Machine (JVM) runtime environment. The JVM is created on the TCB. Each mode has a two-character identifier to indicate its specific purpose and is handled by CICS in a different way.

L8 and L9 TCBs The L8 and L9 TCBs are used for THREADSAFE tasks that are running in the OTE. A task has an L8 mode TCB for its sole use when it calls a program that was enabled with the OPENAPI option and is defined with EXECKEY=CICS or when it calls a task-related user exit program that was enabled with the OPENAPI option. A task has an L9 mode TCB for its sole use when it calls a program that was enabled with the OPENAPI option and is defined with EXECKEY=USER.

SP and S8 TCBs The SP and S8 mode TCBs are used by CICS to manage Secure Sockets Layer (SSL) connections, and if a task needs to use LDAP over the DFHDDAPX XPI interface.

TP and T8 TCBs TP and T8 mode TCBs are used by a JVM server to perform system processing. The JVM server provides a mechanism for CICS to use the same JVM for multiple tasks concurrently.

X8 and X9 TCBs Tasks have exclusive use of an X8 or X9 mode TCB when it calls a C or C++ program that has been compiled with XPLINK and defined with EXECKEY=CICS or EXECKEY=USER.

Tasks have the use of many other TCBs, and TCBs can also be established when IMS DBCTL or WebSphere MQ subsystems are connected.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *CICS Transaction Server V3R1 Channels and Containers Revealed*, SG24-7227-00
- *Introduction to CICS Dynamic Scripting*, SG24-7924-00
- *Threadsafe Considerations for CICS*, SG24-6351-03
- *IBM CICS Explorer*, SG24-7778-01
- *IBM Problem Determination Tools for z/OS*, SG24-7918-00
- *Smarter Banking with CICS Transaction Server*, SG24-7815-00
- *Application Development for CICS Web Services*, SG24-7126-01
- *Implementing Event Processing with CICS*, SG24-7792-00
- *Java Application Development for CICS*, SG24-5275-03
- *Implementing CICS Web Services*, SG24-7657-00
- *CICS and VSAM Record Level Sharing: Planning Guide*, SG24-4765
- *CICS Web Services Workload Management and Availability*, SG24-7144-01
- *Coupling Facility Performance: A Real World Perspective*, REDP-4014-00
- *Extend the CICS Explorer*, SG24-7819

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this Web site:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- *CICS Supplied Transactions*, SC34-7004
- *CICS Transaction Server for z/OS Version 4 Release 1 CICSPlex SM Resource Tables Reference*, SC34-7032

Online resources

These Web sites are also relevant as further information sources:

- Project Zero technology and/or CICS Dynamic Scripting

  http://www.projectzero.org/

- CICS TCP/IP

  http://publib.boulder.ibm.com/infocenter/zos/v1r12/topic/com.ibm.zos.r12.halc001/toc.htm

- XML

  http://www.w3.org/XML/

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

CICS Transaction Server from Start to Finish

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

CICS Transaction Server from Start to Finish

CICS Transaction Server introduction and history

CICS Transaction server facilities, interfaces, and data

CICS Explorer and the CICS Tools

In this IBM Redbooks publication, we discuss CICS, which stands for Customer Information Control System. It is a general-purpose transaction processing subsystem for the z/OS operating system. CICS provides services for running an application online where, users submit requests to run applications simultaneously.

CICS manages sharing resources, the integrity of data, and prioritizes execution with fast response. CICS authorizes users, allocates resources (real storage and cycles), and passes on database requests by the application to the appropriate database manager, such as DB2.

We review the history of CICS and why it was created. We review the CICS architecture and discuss how to create an application in CICS. CICS provides a secure, transactional environment for applications that are written in several languages. We discuss the CICS-supported languages and each language's advantages in this Redbooks publication.

We analyze situations from a system programmer's viewpoint, including how the systems programmer can use CICS facilities and services to customize the system, design CICS for recovery, and manage performance. CICS Data access and where the data is stored, including Temporary storage queues, VSAM RLS, DB2, IMS, and many others are also discussed.

SG24-7952-00 ISBN 0738436178

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks