CICS® Transaction Server for z/OS™

Release Guide

Version 2 Release 1

GC34-5701-00

CICS® Transaction Server for z/OS™

Release Guide

Version 2 Release 1

GC34-5701-00

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 209.

First edition (March 2001)

This edition applies to Version 2 Release 1 of CICS Transaction Server for z/OS, program number 5697-E93, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

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

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

IBM United Kingdom Laboratories,
User Technologies, Mail Point 095,
Hursley Park, Winchester, Hampshire, England,
SO21 2JN.

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

Contents

Preface

What this book is about

This book provides information about new and changed function in CICS® Transaction Server for z/OS™, Version 2 Release 1. It gives an overview of the changes to reference information, and points you to the manuals where more detailed reference information is given.

The programming interface information given in this book is intended to show only what is new and changed from the previous release of CICS TS, and to highlight the benefits of the new function. For programming interface information, read the primary sources of programming interface and associated information in the following publications:
- *CICS Application Programming Reference*
- *CICS System Programming Reference*
- *CICS Customization Guide*
- *CICS External Interfaces Guide*
- *CICSPlex System Manager Application Programming Guide*
- *CICSPlex System Manager Application Programming Reference*

Who this book is for

This book is for those responsible for the following user tasks:
- Evaluation and planning
- System administration
- Programming
- Customization

It describes what is new, changed, and obsolete in the CICS and CICSPlex® SM elements of CICS TS.

What you need to know to understand this book

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

How to use this book

This book is organized in six parts:
- **Part 1** provides a brief summary of new and changed function.
- **Part 2** describes Enterprise JavaBeans™ in CICS.
- **Part 3** describes system management and Parallel Sysplex® support
- **Part 4** describes some miscellaneous changes.
- **Part 5** covers hardware and software requirements for CICS TS Version 2, and describes the publications that are available, in both hardcopy and softcopy.
- **Part 6**, the Appendixes, contains the glossary and the CICS TS bibliography.

Notes on terminology

When the term "CICS"is used without any qualification in this book, it refers to the CICS element of IBM® CICS TS.

“CICSPlex® SM” is used for the CICSPlex System Manager element of IBM CICS TS.

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

Part 1. Summary of CICS TS

This Part provides a summary only of what is new and changed in CICS TS.

Part 1 contains only the summary chapter:

Chapter 1. Summary of CICS TS

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

- “Enterprise JavaBeans™ in CICS”
- “System management and Parallel Sysplex® support” on page 5
- “Miscellaneous enhancements and changes” on page 6
- “Hardware and software requirements” on page 6

Enterprise JavaBeans™ in CICS

CICS support for Enterprise JavaBeans™ is summarized by the following sections:

- “Enterprise JavaBeans”
- “Enhancements to CORBA support”
- “The CICS Connector for CICS TS”
- “CICS support for the persistent reusable JVM” on page 4
- “Enhancements to CICS support of TCP/IP” on page 4
- “Java Naming and Directory Interface™ (JNDI)” on page 4
- “Deploying enterprise beans” on page 4
- “CICSPlex® SM management of enterprise beans” on page 5
- “CICSPlex SM workload management of enterprise beans” on page 5

Enterprise JavaBeans

CICS TS provides partial support for Sun Microsystems Enterprise JavaBeans (EJB™) architecture, Version 1.1, enabling you to configure CICS as an EJB server.

For details, see “Chapter 2. Introduction to Enterprise JavaBeans™” on page 9.

Enhancements to CORBA support

CICS support for CORBA is enhanced to provide the necessary infrastructure for Enterprise JavaBeans by the addition of the following new function. This new function benefits all CICS IIOP applications, including stateless CORBA objects.

- Support for outbound IIOP requests from Java™ applications
- Support for the full CORBA 2.1 API
- Support for distributed transactions

GenericFactory IOR files are now created during the installation of CICS resources. The GenericFactory IOR (genfac.ior) file is created during a PUBLISH CORBASERVER command, and removed from JNDI during RETRACT CORBASERVER depending on settings in the system properties file. This removes the need for the GenFacIOR utility.

Note: IIOP applications are supported in JVM mode only. Java programs created using VisualAge™ for Java, Enterprise ToolKit for OS/390® with the `hpj` command cannot be used with IIOP applications.

For details, see “Chapter 3. Enhancements to CORBA support” on page 59.

The CICS Connector for CICS TS

The CICS Connector for CICS TS enables a Java program or EJB component running in a CICS region to link to a CICS server application program. This allows you to (for example) create powerful EJB server components that can exploit existing CICS application programs written in the traditional application languages supported by CICS.

For details, see “Chapter 4. The CICS Connector for CICS TS” on page 65.

CICS support for the persistent reusable JVM

CICS support for the Java Virtual Machine (JVM) exploits the enhancements in the persistent reusable JVM available on OS/390 Release 8 and later releases. The CICS implementation of the persistent reusable JVM provides performance optimizations designed for the execution of high-volume transactions in a CICS environment.

The CICS support of the persistent reusable JVM enables serial reuse of existing JVMs, which, together with new garbage collection mechanisms, provides significantly improved performance compared with the previous implementation.

For details, see “Chapter 5. CICS support for the IBM persistent reusable JVM” on page 83.

Enhancements to CICS support of TCP/IP

CICS support for TCP/IP is enhanced by the addition of new function:

- Socket management, which enables you to specify the maximum number of sockets that the CICS sockets domain should have active at one time.
- Functions that are used by the enhanced CORBA support:
  - Outbound socket support
  - Asynchronous receive
  - Lifetime management of sockets

For details, see “Chapter 6. Enhancements to CICS support of TCP/IP” on page 103.

Java Naming and Directory Interface™ (JNDI)

With CICS support for Enterprise JavaBeans, the JNDI application programming interface (API) enables enterprise beans, and other Java programs running under CICS, to look up a name or to locate an external enterprise bean, which can then be invoked

For details, see “Chapter 7. Java Naming and Directory Interface™ (JNDI)” on page 109.

Deploying enterprise beans

A desktop Java bean is developed, installed, and run on a workstation. An enterprise bean, however, which will run on a server, requires an additional stage, to prepare the bean for the runtime environment and install it into the EJB server.

For details, see “Chapter 8. Deploying enterprise beans” on page 111

CICSPlex® SM management of enterprise beans

CICSPlex SM is enhanced to support the implementation of enterprise beans in a CICS EJB server. There are new and modified operator views, new and modified BAS views, and enhancements to the CICSPlex SM API and Web User Interface to support for CORBA servers and jars.

For details, see “Chapter 9. CICSPlex SM management of enterprise beans” on page 129 .

CICSPlex SM workload management of enterprise beans

CICSPlex SM provides dynamic workload management of enterprise beans executing in CICS-provided CORBA servers, and extends the distributed routing program (DSRTPRG) routing model

For details, see “CICSPlex SM workload management of enterprise beans” on page 141.

System management and Parallel Sysplex® support

CICS system management and Parallel Sysplex support is enhanced by the following new function:

- “CICS support for VTAM® dynamic alias”
- “Domain name system (DNS) connection optimization”
- “Automatic restart of CICS data sharing servers”
- “Monitoring and statistics changes”

CICS support for VTAM® dynamic alias

Support for VTAM LU aliases enables CICS to use a VTAM-supplied LU alias for autoinstalled terminals and work stations. CICS supports both forms of the dynamic LU alias function—predefined and dynamic.

For details, see “Chapter 10. CICS support for VTAM alias facility” on page 149.

Domain name system (DNS) connection optimization

CICS supports the domain name system (DNS) connection optimization to balance IP connections in a sysplex domain.

For details, see “Chapter 11. Domain name system (DNS) connection optimization” on page 159.

Automatic restart of CICS data sharing servers

Support is added to all three types of CICS data-sharing server—temporary storage, coupling facility data tables, and named counters—to enable them to restart automatically using the services of the MVS™ automatic restart manager (ARM).

For details, see “Chapter 12. Automatic restart of CICS data-sharing servers” on page 165.

Monitoring and statistics changes

There are various changes to CICS monitoring and statistics data in support of the new and changed function in CICS.

For details, see “Chapter 13. Monitoring and statistics changes” on page 169.

Miscellaneous enhancements and changes

These include:
- Integration of the CICS translator into the latest releases of the COBOL and PL/I compilers
- Provision of a sample program for the XLGSTRM global user exit
- Removal of run-time support for the destination control table (DCT), making the DCT system initialization parameter obsolete
- Enhancements and changes to CICS file control
- Major changes to the sample statistics program, DFH0STAT
- Changes to other sample programs
- Enhancements to the CEOT transaction
- Changes to CICSPlex SM support
- Changes to the CICSPlex SM Web user interface
- Suuport for additional code pages.

See “Chapter 14. Other changes and enhancements” on page 177 for details.

Hardware and software requirements

Hardware and software requirements are described in “Chapter 15. Prerequisite hardware and software for CICS Transaction Server for z/OS” on page 189.

Part 2. Enterprise JavaBeans in CICS

This Part describes all the new function included in CICS to provide support for Enterprise JavaBeans. It covers the following topics:

Chapter 2. Introduction to Enterprise JavaBeans™

This chapter describes CICS support for the Enterprise JavaBeans (EJB) architecture.

About Enterprise JavaBeans

This chapter is intended as an introduction to CICS support for Enterprise JavaBeans. It does not attempt to describe the Enterprise JavaBeans architecture in depth. If you need a full description of the EJB architecture, see Sun Microsystem's *Enterprise JavaBeans Specification, Version 1.1*, which is available at `http://www.javasoft.com/products/ejb`.

The chapter covers the following topics:

- "Overview"
- "Benefits" on page 31
- "Requirements" on page 32
- "Changes to CICS externals" on page 32

Overview

This section begins by showing you the "big picture"—what CICS support for Enterprise JavaBeans means in general terms. The sub-sections that follow fill in the details.

The big picture

Sun Microsystem's *Enterprise JavaBeans Specification, Version 1.1*, defines a model for the development of reusable Java server components (known as **enterprise beans**) that can be used in any application server that provides the services and interfaces defined by the specification.

CICS Transaction Server for z/OS Version 2 Release 1 offers partial support for Version 1.1 of the Enterprise JavaBeans specification. Future releases of CICS will support the specification more fully, particularly in regard to security.

You can configure CICS as an EJB server. CICS provides a run-time environment where requests for EJB services are mapped to existing or enhanced CICS services.

You can write enterprise beans that give Java clients access to your past investment in CICS applications and data. For example, you can write enterprise beans that:

- Use the JCICS classes[1] to access CICS resources.
- Use JCICS or the new CICS Connector for CICS TS[1] to link to existing CICS programs written in procedural languages such as COBOL. (For information about the CICS Connector for CICS TS, see page 65.)

Figure 1 on page 10 shows, in simplified form, a CICS EJB application server interacting with its environment. It shows enterprise beans that have been developed on a workstation being installed into the EJB server by a process known

1. Enterprise beans that use the JCICS classes are not portable to a non-CICS environment.

as **deployment**. Once installed in the server, the enterprise beans are executed in a Java Virtual Machine (JVM) at the request of a client program.

Note: The details of Figure 1 are explained in the sections that follow.

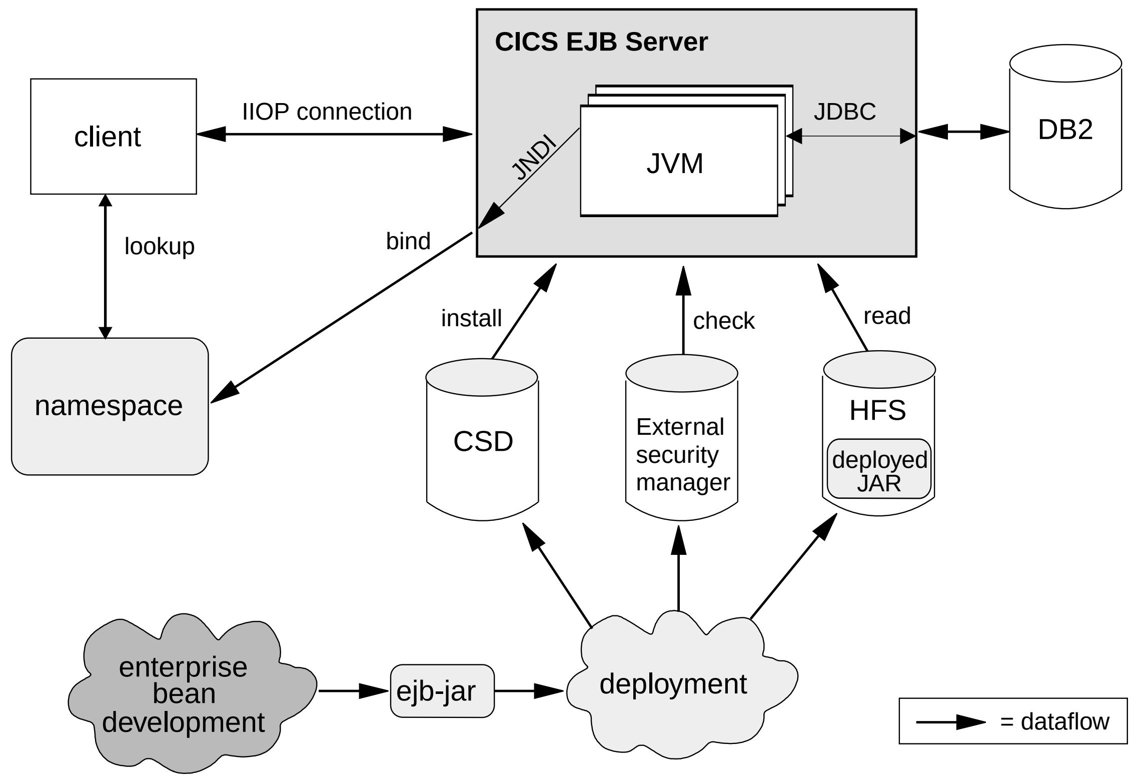


Figure 1. A CICS EJB application server. Enterprise beans developed on a workstation are installed into the EJB server by a process known as deployment. They are executed in a JVM at the request of a client program. The details of this picture are explained in the sections that follow.

Note: The picture shows an external security manager. EJB resource security—the checking of access to enterprise beans, based on EJB security roles—is not supported in CICS TS for z/OS Version 2.1.

JavaBeans and Enterprise JavaBeans

JavaBeans and Enterprise JavaBeans are component architectures for the Java language.

Components

A **component** is a reusable software building block; a pre-built piece of encapsulated application code that can be combined with other components and with handwritten code to produce a custom-built application rapidly.

An application developer can make use of a component without requiring access to its source code. Components can be customized to suit the specific requirements of an application through a set of external property values. For example, a button component has a property that specifies the caption that should appear on the button. An account management component has a property that specifies the location of the account database.

Components execute within a construct called a **container**, which (among other things) provides an operating system process in which to execute the component.

The **component model** defines the interfaces by which the component interacts with its container and with other components. The developer of a component can code it using a variety of internal methods and properties but, to ensure that it can be used with other components, he or she must implement the interfaces defined in the component model. These interfaces also allow components to be loaded into rapid application development (RAD) tools, such as IBM®'s VisualAge® for Java or Symantec's Visual Café.

JavaBeans

A **JavaBean** is a self-contained, reusable software component, written in Java, usually intended for use in a desktop or client application. Typically, desktop JavaBeans have a visual element, and execute within some type of visual container, such as a form, panel, or Web page. Examples might range from a simple button to a fully-featured software CD player.

Bean developers can use a visual tool, such as VisualAge for Java, to create JavaBeans. Application developers can use such tools to "wire" JavaBeans together into a larger application, and to set the properties of individual beans.

Enterprise JavaBeans

The **Enterprise JavaBeans architecture** supports *server components*. Server components are application components that run in an application server such as CICS. Unlike desktop components, they do not have a visual element and the container they run in is not visual.

Server components written to the Enterprise JavaBeans specification are known as **enterprise beans**. They are portable across any EJB-compliant application server.

To be useful, server components require access to the application server's infrastructure services, such as its distributed communication service, naming and directory services, transaction management service, data access and persistence services, and resource-sharing services. Different application servers implement these infrastructure services using different technologies. However, an EJB-compliant application server provides an enterprise bean with access to these services through standard interfaces, and manages many of them on behalf of the bean.

Bean developers can use a visual tool, such as VisualAge for Java, to create enterprise beans. Application developers can combine method calls to enterprise beans with desktop JavaBeans, Web servlets, and handwritten code to form client/server applications.

Note: Release 3.5 of VisualAge for Java supports Version 1.0 of the Enterprise JavaBeans specification, but does not support Version 1.1.

If you develop enterprise beans with VisualAge for Java 3.5, you will need to use the CICS JAR development tool for EJB technology and the CICS code generation utility for EJB technology before using one of the CICS deployment tools for EJB Technology. For more information about this, see the *Java Applications in CICS* manual.

The EJB server

An EJB-compliant application server is known as an **EJB server**. An EJB server could be a transaction processing monitor such as CICS, a Web server, a database, or some other type of server. Note that a CICS EJB server may comprise multiple CICS regions, as described in “Logical servers — enterprise beans in a sysplex” on page 25.

An EJB server provides a standard set of services to support enterprise bean components. These services include:

- Support of the Java Remote Method Invocation (RMI) interface that is used by enterprise beans for communication. RMI has two transport protocol options—JRMP for Java-to-Java interoperation and IIOP for interlanguage interoperation, mediated using a CORBA Object Request Broker (ORB).

  CICS Transaction Server for z/OS, Version 2 Release 1 supports RMI over IIOP (RMI-IIOP), but not JRMP. (JRMP is a proprietary interface which does not support a transaction service context.)
- A container, called an **EJB container**, which provides management services for enterprise beans.
- A distributed transaction management service that implements the javax.transaction.UserTransaction interface of the Java Transaction API (JTA).[2]
- Security services.
- Support for the Java Naming and Directory Interface (JNDI). The JNDI API provides the directory and naming function for Java applications. It enables a client to locate an enterprise bean.
- Support for the Java Data Base Connectivity (JDBC) interface.

The EJB container

Whereas desktop JavaBeans usually run within a visual container such as a form or a Web page, an enterprise bean runs within a container provided by the application server.

The EJB container creates and manages enterprise bean instances at run-time, and provides the services required by each enterprise bean running in it.

The EJB container supports a number of implicit services, including lifecycle, state management, security, transaction management, and persistence:

Lifecycle
Individual enterprise beans do not need to manage process allocation, thread management, object activation, or object passivation explicitly. The EJB container automatically manages the object lifecycle on behalf of the enterprise bean.

State management
Individual enterprise beans do not need to save or restore object state between method calls explicitly. The EJB container automatically manages object state on behalf of the enterprise bean.

Security
Individual enterprise beans do not need to authenticate users or check authorization levels explicitly. The EJB container can automatically perform all security checking on behalf of the enterprise bean.

Transaction management
Individual enterprise beans do not need to specify transaction demarcation code

2. The javax.transaction.UserTransaction interface is used by session beans that manage their own transactions, as described later in this chapter.

to participate in distributed transactions. The EJB container can automatically manage the start, enrollment, commitment, and rollback of transactions on behalf of the enterprise bean.

Persistence
Individual enterprise beans do not need to retrieve or store persistent data from a database explicitly. The EJB container can automatically manage persistent data on behalf of the enterprise bean.

The execution environment

Before enterprise beans can be deployed into an EJB server, their execution environment must be configured. In CICS, this is achieved by installing a CORBASERVER resource definition. A CORBASERVER defines an execution environment for enterprise beans and CORBA stateless objects. For convenience, we shall refer to the execution environment defined by a CORBASERVER definition as a **CorbaServer**.

Note that:

- A CICS EJB server may contain more than one CorbaServer.
- Any number of enterprise beans can be deployed into the same CorbaServer.
- A specific enterprise bean can be deployed multiple times into the same CICS EJB server, but not into the same CorbaServer. (In other words, to install a specific enterprise bean multiple times into the same CICS EJB server you must install it into different CorbaServer execution environments. One reason for doing this might be to make the bean available with different deployment properties—see “The deployment descriptor” on page 14.) Each deployment results in the creation of a distinct home object (see “The home and remote interfaces”).

The home and remote interfaces

Client applications do not interact with an enterprise bean directly. Instead, the client interacts with the enterprise bean through two intermediate objects that are created by the container from classes generated by a deployment tool—one of which classes implements the EJB **home interface** and the other the EJB **remote interface**. As the client invokes operations using these intermediate objects, the container intercepts each method call and inserts the management services.

The home and remote interfaces are implemented as Java RMI remote objects, which allows the ORB to support them as distributed objects.

The home interface
The home interface is the mechanism by which the client identifies the enterprise bean it wants. It allows a client to create, remove, and (for entity beans, not supported by CICS) find existing instances of, enterprise beans. *Note that the “client” might not be a program running on a network workstation; it might, for example, be a servlet running on a Web server; or an enterprise bean, program, or object on the local EJB server, or on another EJB server.*

When a bean is deployed in an EJB server, the container registers the home interface in a namespace that is accessible remotely. Using the Java Naming and Directory Interface (JNDI) API, any client with access to the namespace can locate the home interface by name.

The remote interface
The remote interface allows a client to access the business methods of the enterprise bean. When a client creates or finds an instance of an enterprise bean, the container returns an EJB remote interface object (one per instance).

The remote interface intercepts all business method calls from the client and inserts whatever transaction, state management, persistence, and security services were specified when the bean was deployed.

The deployment descriptor

The rules governing an enterprise bean’s lifecycle, transaction management, security, and persistence are defined in an associated XML document called a **deployment descriptor**. See “Deploying enterprise beans” on page 23.

Re-usable components may be customizable through a set of external property values, so that they can be modified to suit the requirements of a particular application without changing the source code. An enterprise bean developer can provide (within the deployment descriptor) a set of **environment properties** to allow the application developer to customize the bean. For example, a property might be used to specify the location of a database or to specify a default national language. At run time, an environment object is created which contains the customized property values set during the application assembly process or the bean deployment process.

Figure 2 on page 15 shows enterprise bean objects in a CICS EJB server.

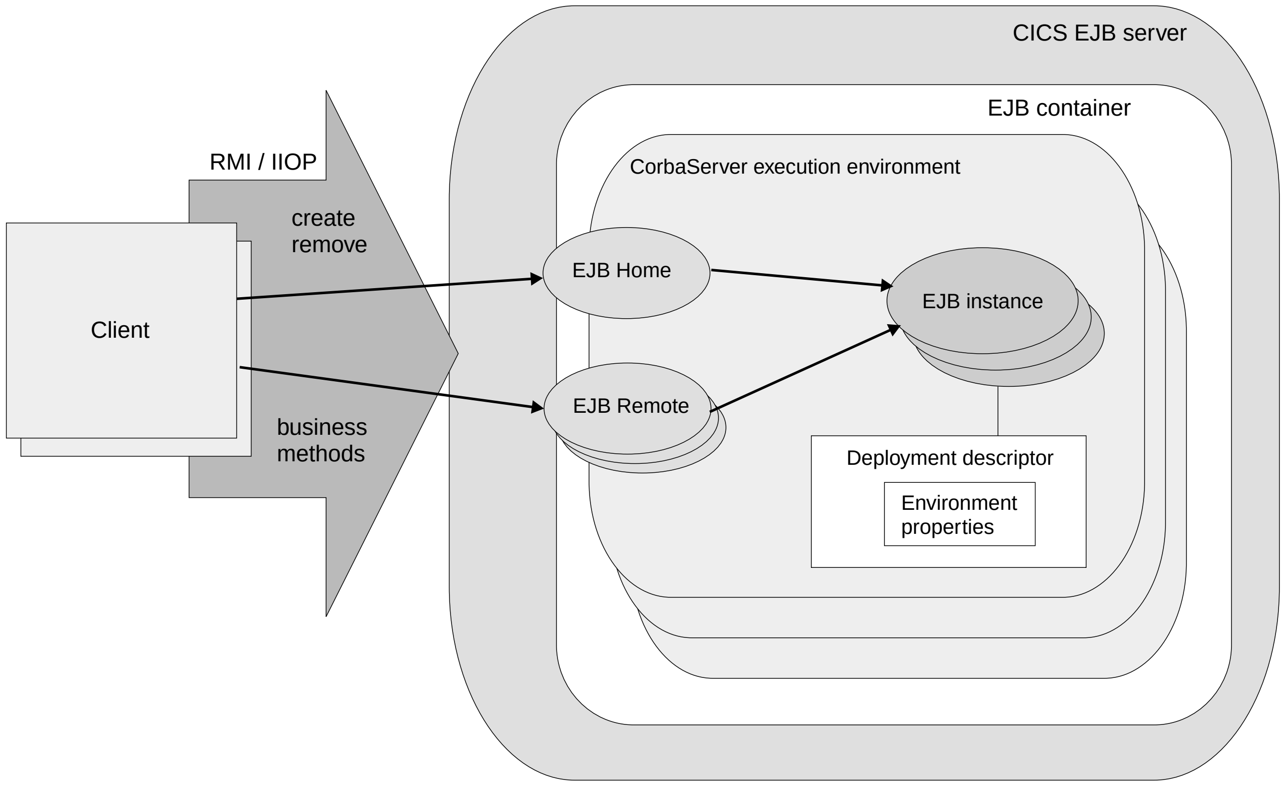


Figure 2. Enterprise bean objects in a CICS EJB server. The EJB container manages and provides services to the enterprise beans contained within it. When a bean is deployed, the deployment tool generates the EJB home and remote interface classes.

The home interface is accessible through JNDI and implements lifecycle services for the bean. The client uses it to create, remove, and (for entity beans, not directly supported by CICS) find instances of enterprise beans.

The container creates an EJB remote interface object for each instance of the bean. The remote interface provides access to the business methods within the bean. It intercepts all business method calls from the client and implements transaction, state management, persistence, and security services for the bean, based on the settings of the bean's deployment descriptor.

Types of enterprise bean

This section discusses the two types of enterprise bean—**session beans** and **entity beans**.

Session beans

A session bean:

- Is created by a client and represents a single conversation, or session, with that client.
- Typically, persists only for the life of the conversation with the client. In this sense, it can be likened to a pseudoconversational transaction.

  If the bean developer chooses to save information beyond the life of a session, he or she must implement persistence operations—for example, JDBC™ or SQL calls—directly in the bean class methods.
- Typically, performs operations on business data on behalf of the client, such as accessing a database or performing calculations.

- May or may not be transactional. If it’s transactional, it can manage its own Object Transaction Service (OTS) transactions, or use container-managed OTS transactions. For an explanation of the relationship between OTS transactions and CICS units of work, see “Managing transactions” on page 17.
- Is not recoverable—if the EJB server crashes, it may be destroyed.
- Has two flavors: **stateful** and **stateless**.

Stateful session beans: A stateful session bean has a client-specific conversational state, which it maintains across methods and transactions; for example, a shopping cart object would maintain a list of the items selected for purchase by the user.

A stateful session bean that manages its own transactions can begin an OTS transaction in one method and commit or roll it back in a subsequent method.

Stateless session beans: A stateless session bean has no client-specific (nor any other kind of) non-transient state; for example, a stock quotation object might simply return current share prices.

A stateless session bean that manages its own transactions and begins a transaction must commit (or roll back) the transaction in the same method in which it started it.

Entity beans

Important

CICS does not support entity beans directly. That is, entity beans cannot run in a CICS EJB server. However, a session bean or program running in a CICS EJB server can be a client of an entity bean running in a non-CICS EJB server.

An entity bean:
- Is typically an object representation of business data, such as a customer order. Typically, the data:
  - Are maintained in a permanent data store, such as a database.
  - Need to persist beyond the life of a client instance. Therefore, an entity bean is relatively long-lived, compared to a session bean.
- Object can be accessed by more than one client at the same time. This is possible because each instance of an entity bean is identified by a **primary key**, which can be used to find it via the home interface.
- Can manage its own persistence (**bean-managed persistence**), or delegate the task to its container (**container-managed persistence**).

  If the bean manages its own persistence, the bean developer must implement persistence operations—for example, JDBC or SQL calls—directly in the bean.

  If the entity bean delegates persistence to the container, the latter manages the persistent state transparently; the bean developer doesn’t need to code any persistence operations within the bean.
- May or may not be transactional. If it’s transactional, all transaction functions are performed implicitly by the EJB container and server. There are no transaction demarcation statements within the bean code. Unlike session beans, an entity bean is not permitted to manage its own OTS transactions. See “Managing transactions” on page 17.

• Is recoverable—it survives a server crash.

Session beans and entity beans compared

Table 1 is a summary of the differences between entity and session beans.

Table 1. Comparison of session and entity beans

Session bean	**Entity bean**
Represents a single conversation with a client. Typically, encapsulates an action or actions to be taken on business data.	Typically, encapsulates persistent business data—for example, a row in a database.
Is relatively short-lived.	Is relatively long-lived.
Is created and used by a single client.	May be shared by multiple clients.
Has no primary key.	Has a primary key, which enables an instance to be found and shared by more than one client.
Typically, persists only for the life of the conversation with the client. (However, may choose to save information.)	Persists beyond the life of a client instance. Persistence can be container-managed or bean-managed.
Is not recoverable—if the EJB server fails, it may be destroyed.	Is recoverable—it survives failures of the EJB server.
May be stateful (that is, have a client-specific state) or stateless (have no non-transient state).	Is typically stateful.
May or may not be transactional. If transactional, can manage its own OTS transactions, or use container-managed transactions. A stateful session bean that manages its own transactions can begin an OTS transaction in one method and commit or roll it back in a subsequent method. A stateless session bean that manages its own transactions and begins an OTS transaction must commit (or roll back) the transaction in the same method in which it was started. The state of a transactional, stateful session bean is not automatically rolled back on transaction rollback. In some cases, the bean can use session synchronization to react to syncpoint.	May or may not be transactional. Must use the container-managed transaction model. If transactional, its state is automatically rolled back on transaction rollback.
Is not re-entrant.	May be re-entrant.

Managing transactions

Clients can begin, commit, and roll back ACID transactions[3] using an implementation of the Java Transaction Service (JTS) or the CORBA Object Transaction Service (OTS). These transactions are analogous to CICS distributed

3. Transactions possessing atomicity, consistency, isolation, and durability. Jim Gray and Andreas Reuter, *Transaction Processing: Concepts and Techniques*, 1993.

units of work. We use the term **OTS transaction** to differentiate these transactions from CICS transaction definitions (the ones with 4-character transaction identifiers) and CICS transaction instances (which are sometimes loosely called tasks).

When a client calls an enterprise bean in the scope of an OTS transaction, information about the transaction flows to the EJB server in an IIOP **service context**, which is like an extra (hidden) parameter on the method request. The EJB server uses this information if it needs to participate in the transaction. Whether the method of an enterprise bean needs to run under a client's OTS transaction (if there is one) is determined by the setting of the **transaction attribute** specified in the bean's deployment descriptor. The method may run under the client's OTS transaction, under a separate OTS transaction which is created for the duration of the method, or under no OTS transaction.

Entity beans must use **container–managed OTS transactions**. All transaction functions are performed implicitly by the EJB container and server. There are no transaction demarcation statements within the bean code.

Session beans can use either container-managed OTS transactions or **bean–managed OTS transactions**. A session bean that uses bean–managed transactions uses methods of the javax.transaction.UserTransaction interface to demarcate transactions. A stateful session bean that manages its own transactions can begin an OTS transaction in one method and commit or roll it back in a subsequent method. A stateless session bean that manages its own transactions and begins an OTS transaction must commit (or roll back) the transaction in the same method.

At runtime, the EJB container implements transaction services according to the setting of the transaction attribute specified in the bean's deployment descriptor. The possible settings of the transaction attribute are:

Mandatory
Indicates that the bean must always execute within the context of the caller's OTS transaction. If the caller does not have a transaction when it calls the bean, the container throws a javax.transaction.TransactionRequiredException exception and the request fails.

Never
Indicates that the bean must not be invoked within the context of an OTS transaction. If a caller has an OTS transaction when it calls the bean, the container throws a java.rmi.RemoteException exception and the request fails.

NotSupported
Indicates that the bean cannot execute within the context of an OTS transaction. If a caller has an OTS transaction when it calls the bean, the container suspends the transaction for the duration of the method call. It resumes the suspended transaction when the method has completed. The suspended transaction context of the client is not passed to resource managers or enterprise bean objects that are invoked from the method.

Required
Indicates that the bean must execute within the context of an OTS transaction. If a caller has an OTS transaction when it calls the bean, the method participates in the caller's transaction. If the caller does not have an OTS transaction, the container starts a new OTS transaction for the method.

RequiresNew
Indicates that the bean must execute within the context of a new OTS

transaction. The container always starts a new OTS transaction for the method. If the caller has an OTS transaction when it calls the bean, the container suspends the caller’s transaction for the duration of the method call. The suspended transaction context of the client is not passed to resource managers or enterprise bean objects that are invoked from the method.

Supports
Indicates that the bean can run with or without a transaction context. If a caller has an OTS transaction when it calls the bean, the method participates in the caller’s transaction. If the caller does not have an OTS transaction, the method runs without one.

Note: Enterprise bean methods always execute in a CICS task, under a CICS unit of work. Even if an enterprise bean method executes under no OTS transaction, any updates that the method makes to recoverable resources are committed only at normal termination of the CICS task, and backed out if there is a need to roll back.

The setting of a method’s transaction attribute determines whether or not the CICS task under which the method executes makes its unit of work part of a wider, distributed OTS transaction.

Accessing data

CICS enterprise beans can use a variety of methods to access data. The methods available depend on the type of data to be accessed:

Relational data

To access relational data, an enterprise bean can use any of the following methods:

- Use a JCICS LINK command, or the **CICS Connector for CICS TS**, to link to a program that uses Structured Query Language (SQL) commands to access the data.
- Where a suitable driver is available, use Java Data Base Connectivity (JDBC) or Structured Query Language for Java (SQLJ) calls to access the data directly. A suitable JDBC driver is available for DB2®.
- Use **Data Access beans** developed using VisualAge for Java. Data Access beans give you a fast, easy, non-programming way of building SQL queries. They are described in “Using Data Access beans” on page 20.
- Use JavaBeans that use JDBC or SQLJ as the underlying access mechanism. You can use any suitable Java integrated development environment (IDE) to develop such JavaBeans.

  Visual Age for Java provides the Enterprise Access Builder for Data to facilitate building such JavaBeans. Use Data Access Builder if you are developing your application outside the Visual Composition Editor, or if you need specialized access to relational data.
- Use entity beans. CICS does not support entity beans running under CICS but does support access to entity beans running on other EJB servers. A CICS enterprise bean could, for example, use an entity bean running on WebSphere™ EE for OS/390 to access DB2 on OS/390.

DL/I data

To access DLI data, an enterprise bean can use a JCICS LINK command, or the CICS Connector for CICS TS, to link to a program that issues EXEC DLI commands to access the data.

VSAM data

To access VSAM data, an enterprise bean can use either of the following methods:

- Use a JCICS LINK command, or the CICS Connector for CICS TS, to link to a program that issues CICS File Control commands to access the data.
- Use the JCICS File Control classes to access VSAM directly.

Notes:

1. All the above techniques can be used by both CICS enterprise beans and CICS Java programs.
2. The same data can be accessed by CICS enterprise beans, CICS Java programs, and (excluding CICS VSAM data) by non-CICS entity beans.
3. For all the above techniques except the use of entity beans, data integrity is maintained by the CICS recovery manager. When entity beans are used, you can use CICS and, for example, WebSphere EE global transactional support to maintain data integrity.
4. You can encapsulate JCICS commands in a JavaBean. This makes it easier to program the enterprise beans that use JCICS to access data.
5. The CICS Connector for CICS TS is described in "Chapter 4. The CICS Connector for CICS TS" on page 65.

Using Data Access beans

To access relational databases, enterprise beans can use JDBC calls. However, the recommended method is to use Data Access beans, which package the native JDBC calls with extra function and make them more convenient to use. Data Access beans are JavaBeans, not enterprise beans. They are a feature of VisualAge for Java.

Three Data Access beans provide core function for accessing databases:

- Select bean
- Modify bean
- ProcedureCall bean

Additional beans provide user interfaces to invoke methods on the core beans and to help display output from the database:

- CellSellector bean
- RowSelector bean
- ColumnSelector bean
- CellRangeSellector bean

All the beans mentioned are non-visual.

The Select, Modify, and ProcedureCall beans have properties that contain connection aliases and SQL specifications. These properties allow you to connect to relational databases and access data. You can also use parameterized SQL statements with the Select, Modify, and ProcedureCall beans.

For detailed programming information about Data Access beans, see the softcopy document *Data Access*, supplied with VisualAge for Java Enterprise Edition, Version 3.

Important

There are special considerations for using Data Access beans in CICS. For the latest information, ensure that you read the CICS-supplied “How to” file, `doc/HOWTO/Data-Access-Beans-HOWTO`. This supplements and takes precedence over the documentation of Data Access beans supplied with VisualAge for Java Enterprise Edition.

Security

EJB security is concerned with authentication and access control.

Authentication

Authentication of EJB clients uses the TCP/IP secure sockets layer (SSL) protocol. How to configure CICS to use SSL is described in the *CICS Internet Guide*.

Access control and EJB security roles

Access to enterprise beans is based on the concept of security **roles**. An EJB role represents a type of user that should have a particular level of access to an application. It maps to a group of users defined to your external security manager (ESM).

The roles that are permitted to execute a particular enterprise bean or particular methods of a bean are specified in the bean’s deployment descriptor.

Important

EJB security roles are not yet supported by CICS.

CICS transaction and resource security: You can use CICS transaction security and resource security with EJB resources.

CICS transaction security applies to the CICS transactions associated with enterprise bean methods—that is, the transactions named on EJB REQUESTMODEL definitions.

CICS resource security applies to the CICS resources accessed by enterprise beans (by means of, for example, JCICS).

User tasks

Typically, several people are involved in the development and deployment of applications that use enterprise beans:
- The bean provider
- The application assembler
- The deployer
- The system administrator

Note: In smaller organizations, one person may be responsible for more than one of these tasks.

The bean provider

The bean provider develops reusable enterprise beans that typically implement business tasks or business entities.

The bean provider's output is an **ejb-jar** file that contains one or more enterprise beans. The bean provider is responsible for:

- The Java classes that implement an enterprise bean's business methods.
- The definition of the bean's remote and home interfaces.
- The bean's deployment descriptor.

  The deployment descriptor includes the structural information—for example, the name of the enterprise bean class—of the enterprise bean and declares all the bean's external dependencies—for example, the names and types of the resource managers that the enterprise bean uses.

The application assembler

The application assembler creates applications that use enterprise beans. He combines enterprise beans and hand-written client code into a client/server application. Although he must be familiar with the function provided by the enterprise beans' remote and home interfaces, he does not need to have any knowledge of the enterprise beans' implementation.

The input to the application assembler is one or more ejb-jar files produced by the bean provider. His output is one or more ejb-jar files that contain the enterprise beans, along with their application assembly instructions and customized environment settings. He has inserted the application assembly instructions, security roles, and environment values into the deployment descriptors.

The application assembler may also combine enterprise beans with other types of application components—for example, JavaBeans—when assembling an application.

Typically, the application assembly step occurs before the deployment of the enterprise beans. However, sometimes assembly may be performed after the deployment of all or some of the enterprise beans.

The deployer

The deployer takes one or more ejb-jar files produced by the application assembler and deploys the enterprise beans contained in the ejb-jar files into a specific CorbaServer in an EJB server.

The deployer must:

- Resolve all the external dependencies declared by the bean provider. For example, he must ensure that all resource manager connection factories used by the enterprise beans are present in the operational environment, and bind them to the resource manager connection factory references declared in the deployment descriptor.
- Follow the application assembly instructions defined by the application assembler. For example, the deployer is responsible for mapping the security roles defined by the application assembler to CICS user groups and external security manager profiles. (EJB security roles are not supported in CICS TS for z/OS Version 2.1.)

The deployment process is semi-automated. To perform his role, the deployer uses a **deployment tool**. Deployment tools are provided by VisualAge for Java and by CICS.

The deployer's output are enterprise beans that have been customized for the target operational environment, and deployed in one or more CorbaServers.

The system administrator

The system administrator is responsible for configuring and administering the CICS regions that comprise the logical EJB server, together with their network connections. He or she is also responsible for overseeing the well-being of the deployed EJB applications at runtime.

Deploying enterprise beans

A desktop Java bean is developed, installed, and run on a workstation. An enterprise bean, however, which will run on a server, requires an additional stage, **deployment**, to prepare the bean for the runtime environment and install it into the EJB server.

Enterprise beans are produced by the bean provider and customized by the application assembler. They are supplied to the deployer in an ejb-jar file. This file contains:

- The Java classes for one or more enterprise beans.
- A single deployment descriptor, written in XML, which describes the characteristics of each of the enterprise beans, such as:
  - Transaction attributes
  - Environment properties
  - Security levels
  - Application assembly information.

Also required is CICS-specific information, such as resource definition requirements, in either resource definition online (RDO) format (for DFHCSDUP) or CICSPlex SM Business Application Services (BAS) format (for BATCHREP).

Here's an outline of the deployment process:[4]

1. A **deployment tool** (such as the CICS JAR development tool, described in "The CICS JAR development tool for EJB technology" on page 115) is used to transform the ejb-jar file into a form suitable for deployment. The transformed file contains the XML deployment descriptor and enterprise bean classes from the ejb-jar file, plus additional classes generated in support of the EJB container. The transformed file is stored as a **deployed JAR file** on the hierarchical file system (HFS) used by z/OS.
2. CICS resource definitions are created on the CSD. Definitions are required for:
   - The CorbaServer execution environment (CORBASERVER). (The same CORBASERVER definition will be installed on each CICS AOR in the logical EJB server.)
   - Deployed JAR files (DJARs), each of which includes the HFS filename of a deployed JAR file.
   - TCP/IP services (for IIOP).
   - Request models (to enable client IIOP requests to be processed correctly).

   Note: "Setting up a logical EJB server" on page 27 contains more information about these RDO definitions.
3. Security definitions are added to the external security manager (CICS TS for z/OS Version 2.2 only). These specify which roles can execute particular beans and methods, and which CICS user IDs are associated with each role.
4. The resource definitions on the CSD are installed in CICS. Installing a DJAR definition causes CICS to:

4. This simplified description of the deployment process assumes that you're using RDO rather than BAS.

- Copy the deployed JAR file (and the classes it contains) to a shelf directory on HFS
- Read the deployed JAR from the shelf, parse its XML deployment descriptor, and store the information it contains

5. Using SPI commands, a reference to the home interface class of each deployed bean is published in an external namespace. The namespace is accessible to clients through JNDI.

Figure 3 shows the deployment process.

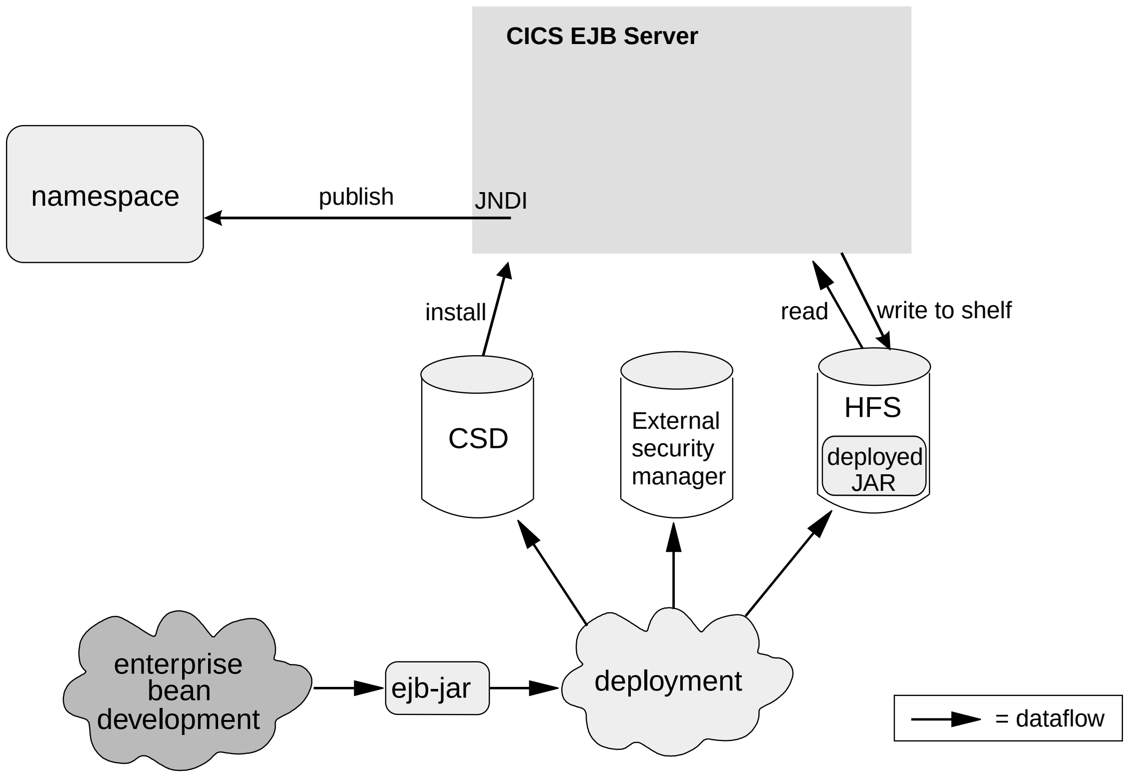


Figure 3. Deploying enterprise beans into a CICS EJB server. A deployment tool is used to perform code generation on the ejb-jar file containing the bean classes. The transformed file is stored as a deployed JAR file on HFS. An RDO definition of the deployed JAR file is created on the CSD and installed in CICS, together with other definitions for TCP/IP services, request models, and the CorbaServer execution environment. Security definitions are created on the external security manager.

Note: The picture shows an external security manager. EJB resource security—the checking of access to enterprise beans and the CICS resources they use, based on EJB security roles—is not supported in CICS TS for z/OS Version 2.1.

Configuring CICS as an EJB server

A CICS EJB server contains the following basic components:

The listener

The job of the listener is to listen for (and respond to) incoming TCP/IP connection requests. An IIOP listener is configured by a **TCPIPSERVICE** resource definition to listen on a specific TCP/IP port and to attach an IIOP **request receiver** to handle each connection.

Once an IIOP connection has been established between a client program and a particular request receiver, all subsequent requests from the client program over that connection flow to the same request receiver.

The request receiver

The request receiver analyzes the structured IIOP data. It passes the incoming request to a **request processor** by means of a **request stream**, which is an internal CICS routing mechanism. The object key in the request determines whether the request must be sent to a new or an existing request processor.

If the request must be sent to a new request processor, a CICS TRANSID is determined by comparing the request data with templates defined in **REQUESTMODEL** resource definitions. (If no matching REQUESTMODEL definition can be found, the default TRANSID, CIRP, is used.) The TRANSID defines execution parameters that are used by the request processor.

The request processor

The request processor is a transaction instance that manages the execution of the IIOP request. It:
- Locates the object identified by the request
- For an enterprise bean request, calls the container to process the bean method
- For a request for a stateless CORBA object, the ORB typically processes the request itself (although the transaction service may also be involved).

Logical servers — enterprise beans in a sysplex

You can implement a CICS EJB server in a single CICS region. However, in a sysplex it's likely that you'll want to create a server consisting of multiple regions. Using multiple regions makes failure of a single region less critical and enables you to use workload balancing. A **CICS logical EJB server** consists of one or more CICS regions configured to behave like a single EJB server.

Typically, a CICS logical EJB server consists of:
- A set of cloned **listener regions** defined by identical TCPIPSERVICE definitions to listen for incoming IIOP requests.
- A set of cloned application-owning regions (AORs), each of which supports an identical set of enterprise bean classes in an identically-defined CorbaServer.

Note: The listener regions and AORs can be separate or combined into listener/AORs.

Workload balancing in a sysplex: Workload balancing is implemented at two levels:
1. To balance client connections across the listener regions, you can use any of the following methods:
   - Connection optimization by means of dynamic Domain Name System (DNS) registration. (Connection optimization is described in "Chapter 11. Domain name system (DNS) connection optimization" on page 159.)
   - IP routing.
   - A combination of connection optimization and IP routing.

   With connection optimization by means of dynamic DNS registration, for example, multiple CICS regions are started to listen for IIOP requests on the same port (using virtual IP addresses). Each client IIOP connection request contains a generic host name and port number. The generic host name in each connection request is resolved to a real IP address by MVS DNS and Workload Management (WLM) services.
2. To balance OTS transactions across the AORs, you can use either of the following:
   - CICSPlex SM

- A customized version of the CICS distributed routing program, DFHDSRP.

Important

It is convenient to talk of balancing (or dynamically routing) OTS transactions across AORs. Strictly speaking, however, what are dynamically routed are *method requests* for enterprise beans and CORBA stateless objects. There is a correlation between routing method requests dynamically and routing OTS transactions dynamically: CICS invokes the routing program for requests for methods that will run under a *new* OTS transaction, but not for requests for methods that will run under an *existing* OTS transaction—these it directs automatically to the AOR in which the existing OTS transaction runs. However, because requests for methods that will run under *no OTS transaction* can also be dynamically routed, the correlation is not exact.

We must be clear about what we mean by new and existing OTS transactions. For the purposes of this chapter:

a. By a *new OTS transaction* we mean an OTS transaction *in which the target logical server is not already participating*, prior to the current method call; *not* necessarily an OTS transaction that was started immediately before the method call.
b. By an *existing OTS transaction* we mean an OTS transaction *in which the target logical server is already participating*, prior to the current method call; *not* simply an OTS transaction that was started some time ago.

For example, if a client starts an OTS transaction, does some work, and then calls a method on an enterprise bean with the **Supports** transaction attribute, so far as the CICS EJB server is concerned this is a new OTS transaction, because the server has not been called within this transaction's scope before. If the client then makes a second and third method call to the same target object, before committing its OTS transaction, these second and third calls occur within the scope of the existing OTS transaction.

Figure 4 on page 27 shows a CICS logical EJB server. In this example, the listener regions and AORs are in separate groups, connection optimization is used to balance client connections across the listener regions, and distributed routing is used to balance OTS transactions across the AORs.

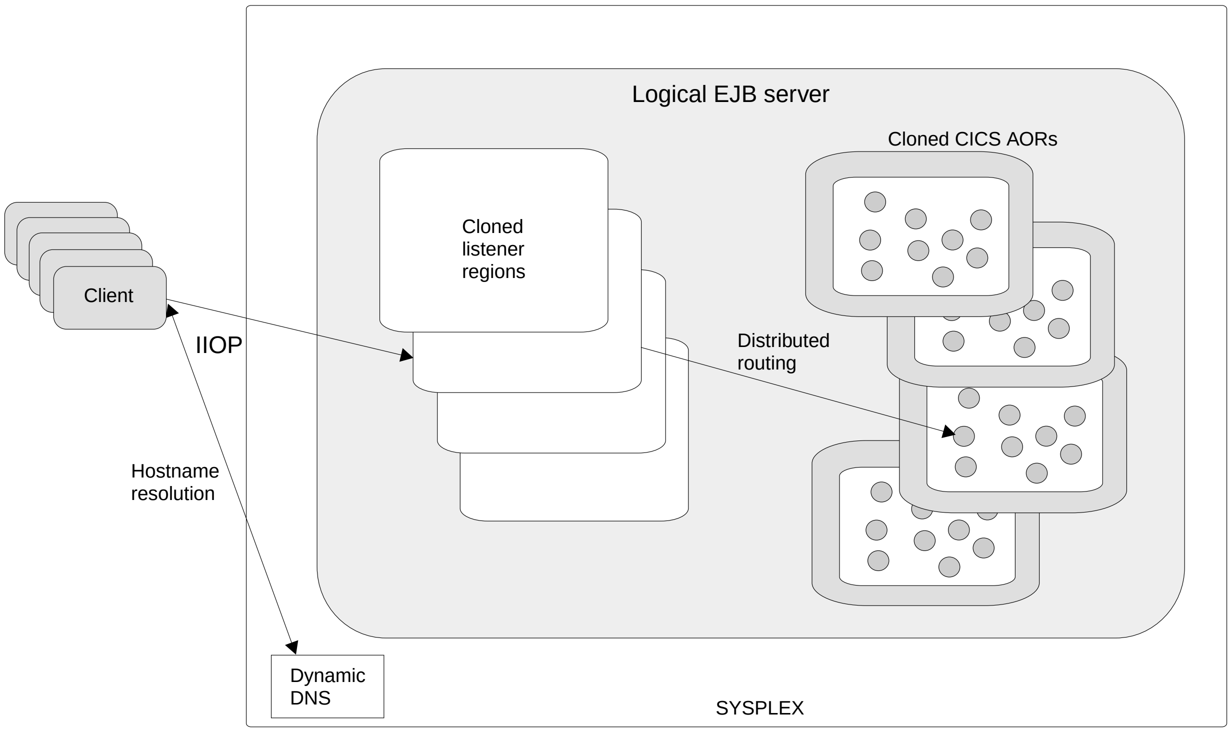


Figure 4. A CICS logical EJB server. The logical server consists of a set of cloned listener regions and a set of cloned AORs. In this example, connection optimization by means of dynamic DNS registration is used to balance client connections across the listener regions. Distributed routing is used to balance OTS transactions across the AORs.

Setting up a logical EJB server

In simplified form, the steps involved in setting up a CICS logical EJB server to provide access to a specific enterprise bean are:

1. Create a set of cloned listener regions.
2. Create a set of cloned AORs. Each of the AORs must:
   - Be set up to use JNDI
   - Use the same JNDI initial context as the other AORs
   - Be connected to all of the listener regions by MRO (not ISC).
3. Take the ejb-jar file and perform code generation on it to produce a deployed JAR file on HFS.
4. Create the following resource definitions. You can create them on a CSD that is shared by all the regions in the logical server, copy them to all the CSDs used by the regions, or add them to a CICSPlex SM Resource Description that applies to all the regions. Optionally, you can use the CICS-supplied deployment tool to create some of these definitions.
   - A TCPIPSERVICE.
   - Some REQUESTMODEL definitions. *These are only required if the default TRANSID, CIRP, cannot be used*.

     The BEANNAME attribute of each REQUESTMODEL definition must “match” (in a pattern-matching sense) the name of an enterprise bean in the deployment descriptor in the deployed JAR file on HFS. The value of the CORBASERVER attribute must be identical with the name of the CorbaServer on the CORBASERVER definition.
   - A CORBASERVER definition.

The 'server ORB' attributes of the CORBASERVER definition (HOST, SSL, and PORT or SSLPORT) must match the corresponding attributes of the TCPIPSERVICE definition (IPADDRESS or DNSGROUP, SSL, and PORTNUMBER respectively). To clarify:

a. The value of the HOST option of the CORBASERVER definition must match that of the IPADDRESS option of the TCPIPSERVICE definition. However, if the TCPIPSERVICE specifies a value for DNSGROUP, the HOST option of the CORBASERVER definition must specify a matching generic host name.
b. If the CORBASERVER definition does not support the secure sockets layer—SSL(NO)—the CorbaServer has only one, non-SSL, TCP/IP port. Its PORT number must match the value of PORTNUMBER on the TCPIPSERVICE definition.
c. If the CORBASERVER definition supports the secure sockets layer—SSL(YES) or SSL(CLIENTCERT)—the CorbaServer has two TCP/IP ports—one which supports the secure sockets layer and one which does not. Either its SSLPORT number or its PORT number must match the value of PORTNUMBER on the TCPIPSERVICE definition. Alternatively, you can install a TCP/IP service on both ports (using two TCPIPSERVICE definitions).

- A DJAR definition.

  The HFSFILE attribute of the DJAR definition points to the deployed JAR file on HFS. The CORBASERVER attribute matches the name of the CorbaServer on the CORBASERVER definition.
- FILE definitions for the following files required by CICS:

  The EJB directory, DFHEJDIR
  is a file containing a request streams directory which must be shared by all the regions (listeners and AORs) in the logical EJB server. (Request streams are used in the distributed routing of method requests for enterprise beans and CORBA stateless objects.) You must define DFHEJDIR as recoverable.

  The EJB object Store, DFHEJOS
  is a file of stateful session beans that have been passivated. It must be shared by all the AORs in the logical EJB server. You must define it as non-recoverable.

  To share DFHEJDIR and DFHEJOS across multiple regions, you could, for instance, use any of the following methods:
  – Define them as remote files in a file-owning region (FOR)
  – Define them as coupling facility data tables
  – Use VSAM RLS.

  There are sample FILE definitions for DFHEJDIR and DFHEJOS in the CICS-supplied RDO group, DFHEJVS. There are sample coupling facility FILE definitions for DFHEJDIR and DFHEJOS in the CICS-supplied RDO group, DFHEJCF. There are sample VSAM RLS FILE definitions for DFHEJDIR and DFHEJOS in the CICS-supplied RDO group, DFHEJVR. (DFHEJVS, DFHEJCF, and DFHEJVR are not included in the default CICS startup group list, DFHLIST.)

Note: For clarity's sake, we're assuming that there's only one CorbaServer in the logical server, and that all the enterprise beans you want to deploy are in a single ejb-jar file. To create another CorbaServer, you'll need a second CORBASERVER definition and another TCPIPSERVICE definition. To deploy beans in other ejb-jar files, you'll need further DJAR and REQUESTMODEL definitions.

5. Define the underlying VSAM data sets for DFHEJDIR and DFHEJOS. CICS supplies sample JCL to help you do this, in the DFHDEFDS member of the SDFHINST library.
6. On each of the listener regions, install:
   - The TCPIPSERVICE definition
   - The REQUESTMODEL definitions
   - The file definition for DFHEJDIR

   On each of the AORs, install:
   - The REQUESTMODEL definitions
   - The CORBASERVER definition
   - The DJAR definition
   - The file definitions for DFHEJDIR and DFHEJOS
7. Issue a PERFORM CORBASERVER(CorbaServer_name) PUBLISH command on at least one of the AORs. This binds the homes of the enterprise beans into the JNDI namespace. The command can be issued using EXEC CICS, the CEMT master terminal transaction, or via a CICSPlex SM EUI or WUI View.

Figure 5 shows the RDO definitions required to define a CICS logical EJB server. It shows which definitions are required in the listener regions, which in the AORs, and which in both.

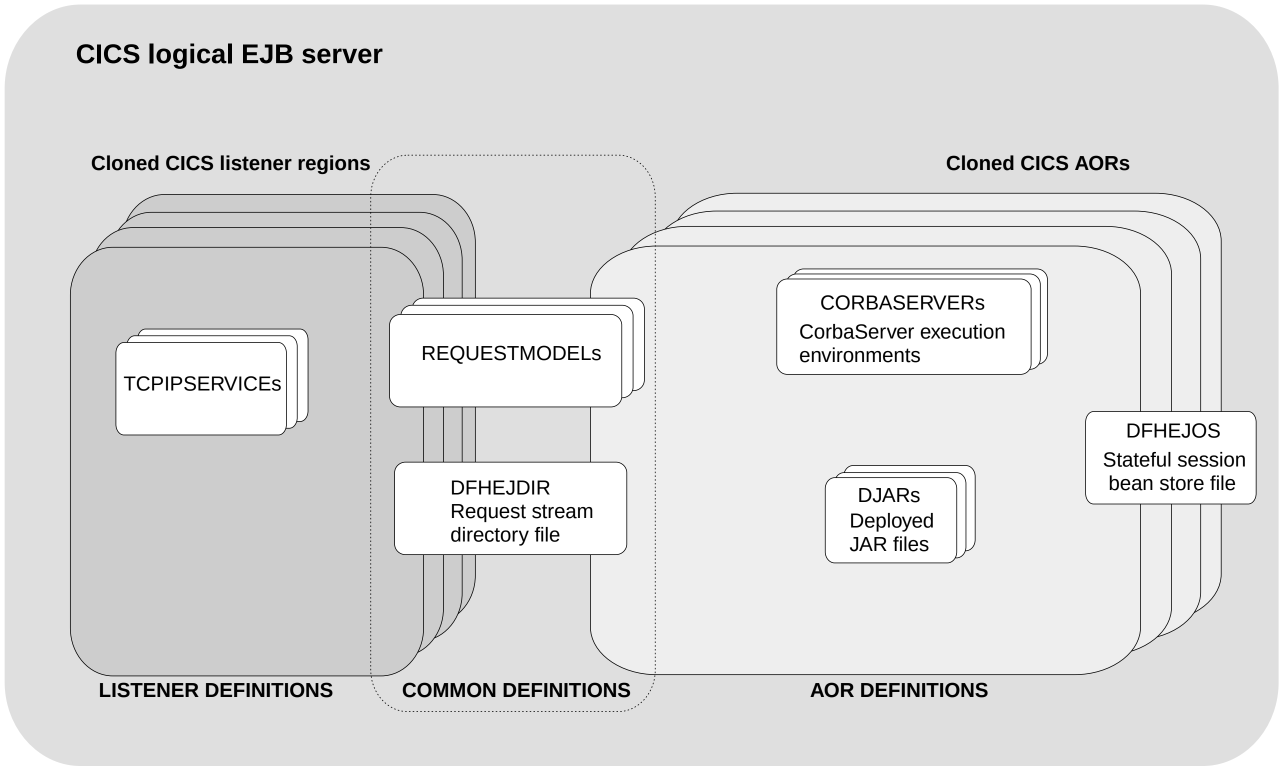

CICS logical EJB server
Cloned CICS listener regions
Cloned CICS AORs
TCPIPSERVICEs
REQUESTMODELs
DFHEJDIR Request stream directory file
CORBASERVERs CorbaServer execution environments
DJARs Deployed JAR files
DFHEJOS Stateful session bean store file
LISTENER DEFINITIONS
COMMON DEFINITIONS
AOR DEFINITIONS

Figure 5. Resource definitions in a CICS logical EJB server. The picture shows which definitions are required in the listener regions, which in the AORs, and which in both.

What can a client do with a bean?

This section contains example code fragments that illustrate how a client program can use an enterprise bean.

Get a reference to the bean’s home

In order to do anything with the bean, the client must obtain a reference to the bean’s home interface. To do this, it looks up a well-known name via JNDI:

```
// Obtain a JNDI initial context
Context initContext = new InitialContext();

// Look up the home interface of the bean
Object accountBeanHome = initContext.lookup("JNDI_prefix/AccountBean");
// where:
// 'JNDI_prefix/' is the JNDI prefix on the CORBASERVER definition
// 'AccountBean' is the name of the bean in the XML deployment descriptor

// Convert to the correct type
AccountHome accountHome = (AccountHome)
            PortableRemoteObject.narrow(accountBeanHome,AccountHome.class);
```

Use the home interface

The client can use the bean’s home interface to:
- Create a new instance of the bean
- Delete an instance of the bean

For example:

```
// Create two bean instances
Account anAccount = accountHome.create();
Account anotherAccount = accountHome.create("12345");

// Remove a bean instance
accountHome.remove("12345");
```

Use the remote interface

The client can use the bean’s remote interface to:
- Invoke the bean’s methods
- Delete the bean

For example:

```
// Use the bean
anAccount.deposit(1000000);
// Remove it
anAccount.remove();
```

What can a bean do?

An enterprise bean benefits from many services—such as lifecycle management and security—that are provided implicitly by the EJB container, based on settings in the deployment descriptor. This leaves the bean provider free to concentrate on the bean’s business logic. This section looks at some of the things a bean can do.

Look up JNDI entries
A bean can use JNDI calls to retrieve:
- References to resources
- Environment variables
- References to other beans.

Access resource managers
A bean can:
- Obtain a connection to a resource manager
- Use the resources of the resource manager

- Close the connection.

Link to CICS programs
A bean can use JCICS or the CICS Connector for CICS TS to link to a CICS program, that may be written in any of the CICS-supported languages and be either local or remote. The bean provider can use the CICS Connector for CICS TS to build beans that make use of the power of existing (non-Java) CICS programs.

The CICS Connector for CICS TS is described in “Chapter 4. The CICS Connector for CICS TS” on page 65.

Access files
A bean can use JCICS to read and write to files.

Call other beans
A bean can:
- Obtain references to the home and remote interfaces of other bean objects
- Invoke the methods of another bean object.

A bean can act as the client of another bean object, as the server of another bean object, or as both.

Manage transactions
Optionally, a session bean can manage its own OTS transactions, rather than using container-managed transactions.

Benefits

Some of the benefits of using enterprise beans are:

Component portability
The EJB architecture provides a simple, elegant component container model. Java server components can be developed once and deployed in any EJB-compliant server.

Architecture independence
The EJB architecture is independent of any specific platform, proprietary protocol, or middleware infrastructure. Applications developed for one platform can be redeployed on other platforms.

Developer productivity
The EJB architecture improves the productivity of application developers by standardizing and automating the use of complex infrastructure services such as transaction management and security checking. Developers can create complex applications by focusing on business logic rather than environmental and transactional issues.

Customization
Enterprise bean applications can be customized without access to the source code. Application behavior and runtime settings are defined through attributes that can be changed when the enterprise bean is deployed.

Multitier technology
The EJB architecture overlays existing infrastructure services.

Versatility and scalability
The EJB architecture can be used for small-scale or large-scale business transactions. As processing requirements grow, the enterprise beans can be migrated to more powerful operating environments.

In addition to these general benefits of using EJB technology, there are specific benefits of using enterprise beans with CICS. For example:

Superior workload management

You can balance client connections across a set of cloned listener regions.

You can use CICSPlex SM or the CICS distributed routing program to balance OTS transactions across a set of cloned AORs.

Superior transaction management

Enterprise beans in a CICS EJB server benefit from CICS transaction management services—for example:
- Shunting
- System log management
- Performance optimizations
- Runaway detection
- Deadlock detection
- TCLASS management
- Monitoring and statistics

Access to CICS resources

You can, for example, use JCICS or the CICS Connector for CICS TS to build enterprise beans that make use of the power of existing (non-Java) CICS programs. The developer of a Java client application can use your server components to access CICS—without needing to know anything about CICS programming. See "Chapter 4. The CICS Connector for CICS TS" on page 65.

Requirements

Hardware

There are no specific hardware requirements for enterprise beans, over and above those for CICS Transaction Server for z/OS, Version 2 Release 1 itself.

Software

The software requirements for enterprise beans are:
- IBM Developer Kit for OS/390, Java 2 Technology Edition
- A Corba Object Services (COS) Naming Directory Server that supports the Java Naming and Directory Interface (JNDI) Version 1.2. WebSphere Application Server Advanced Edition is shipped with CICS for this purpose.

  Note: The JNDI API provides directory and naming function for Java applications. It enables a client to locate an enterprise bean. The JNDI is mapped to an external Naming Directory Server. The latter must be a COS Naming Directory Server, which is most conveniently obtained by use of WebSphere Application Server Advanced Edition running on an external Windows NT® machine.
- DB2 with Java Data Base Connectivity (JDBC) Version 1.2 extensions.

Changes to CICS externals

There are changes to a number of CICS external interfaces in support of enterprise beans. These are:
- "Changes to system initialization" on page 33
- "Changes to system definition" on page 33
- "Changes to resource definition" on page 34

- “Changes to the application programming interface” on page 48
- “Changes to the system programming interface” on page 48
- “Changes to CICS supplied transactions” on page 52
- “Changes to global user exits” on page 53
- “Changes to the exit programming interface (XPI)” on page 53
- “Changes to user-replaceable programs” on page 54
- “Changes to monitoring” on page 54
- “Changes to statistics” on page 54
- “Changes to problem determination” on page 54
- “Changes to sample programs” on page 56

Changes to system initialization

New system initialization parameters

There is one new system initialization parameter:

KEYRING=*key_ring_name*
: Specifies the name of a key ring define in the external security manager’s database, and which contains the keys and X.509 certificates used by CICS support for the secure sockets layer.

Changes to existing system initialization parameters

Support for enterprise beans introduces several new CICS domains. Consequently, there are new codes for use with the STNTRxx and SPCTRxx system initialization parameters used to set the levels of standard and special tracing for selected CICS components. Table 2 shows the new component codes and the domains to which they relate.

Table 2. CICS component names and abbreviations

Code	Component name
EJ	Enterprise Java domain
II	IIOP domain
OT	Object Transaction Service domain
RZ	Request streams domain
SJ	CICS JVM domain

Obsolete system initialization parameters

The following system initialization parameter is obsolete:

KEYFILE=*key-ring-path-name*
: In CICS TS 1.3, KEYFILE specified the name of an HFS file containing keys and certificates managed by the **gskkyman** utility. These keys and certificates are now managed by the external security manager, and defined by a RACDCERT ADDRING command, and defined to CICS by the KEYRING system initialization parameter.

Changes to system definition

There are several changes to CICS system definition to enable EJB support.

Cloned CICS regions

CICS supports the cloning of the listener regions and application-owning regions that comprise the logical EJB server.

You clone regions by specifying, with some key exceptions, the same system initialization parameters and, through the system initialization parameters, specify the same resource definitions. The only attributes of cloned regions that are different are their identifiers, specified on the APPLID, SYSIDNT, and MNSUBSYS system initialization parameters.

VSAM data sets

There are two new system data sets required, with DDNAMES of DFHEJDIR and DFHEJOS. Unless these CICS files reside in a coupling facility (in which case it is not necessary to define them as VSAM data sets), define the underlying VSAM data sets that relate to these CICS files. (see “File definitions” on page 47).

The DFHDEFDS member of the SDFHINST library contains two sets of IDCAMS DEFINE statements to define the DFHEJDIR and DFHEJOS clusters. The data set names used in these definitions are tailored by the parameters you specify on the DFHISTAR installation job. You can edit these jobs further after installation (for example, to vary the size of the initial primary allocation).

Shelf directories

You must define one or more shelf directories on HFS. A **shelf** is primarily used to hold **JARs**—see the description of the SHELF option of the CEDA DEFINE CORBASERVER command.

CICS regions into which CORBASERVER definitions are installed must have full permissions to the shelf directory—read, write, and execute.

Changes to resource definition

Support for enterprise beans introduces some new RDO objects; and there are changes to some existing RDO objects.

New RDO objects

Two new RDO objects are introduced: CORBASERVER and DJAR.

CORBASERVER resource definition: Use CORBASERVER to define an execution environment for enterprise beans and stateless CORBA objects (a CorbaServer).

An identical CORBASERVER definition must be installed in each of the (one or more) application-owning regions (AORs) that form a *logical EJB/CORBA server* for workload balancing.

If you are not using load balancing, you should not define and install CORBASERVER definitions with the same name (but different attributes) in different CICS regions, as only the first PERFORM CORBASERVER PUBLISH command will register an entry with the nameserver for the CORBASERVER name.

The definition’s 'Server ORB' attributes (HOST, SSL, PORT, and SSLPORT) are included in Interoperable Object References (IORs) exported from this logical server. They must match the corresponding attributes (IPADDRESS or DNSGROUP, SSL, and PORTNUMBER) of an IIOP TCPIPSERVICE definition installed in each of the (one or more) listener regions of the logical EJB/CORBA server. To clarify:

1. The value of the HOST option of the CORBASERVER definition must match that of the IPADDRESS option of the TCPIPSERVICE definition. However, if the TCPIPSERVICE specifies a value for DNSGROUP, the HOST option of the CORBASERVER definition must specify a matching generic host name.

2. If the CORBASERVER definition does not support the secure sockets layer—SSL(NO)—the CorbaServer has only one, non-SSL, TCP/IP port. Its PORT number must match the value of PORTNUMBER on the TCPIPSERVICE definition.
3. If the CORBASERVER definition supports the secure sockets layer—SSL(YES) or SSL(CLIENTCERT)—the CorbaServer has two TCP/IP ports—one which supports the secure sockets layer and one which does not. Either its SSLPORT number or its PORT number must match the value of PORTNUMBER on the TCPIPSERVICE definition. Alternatively, you can install a TCP/IP service on both ports.

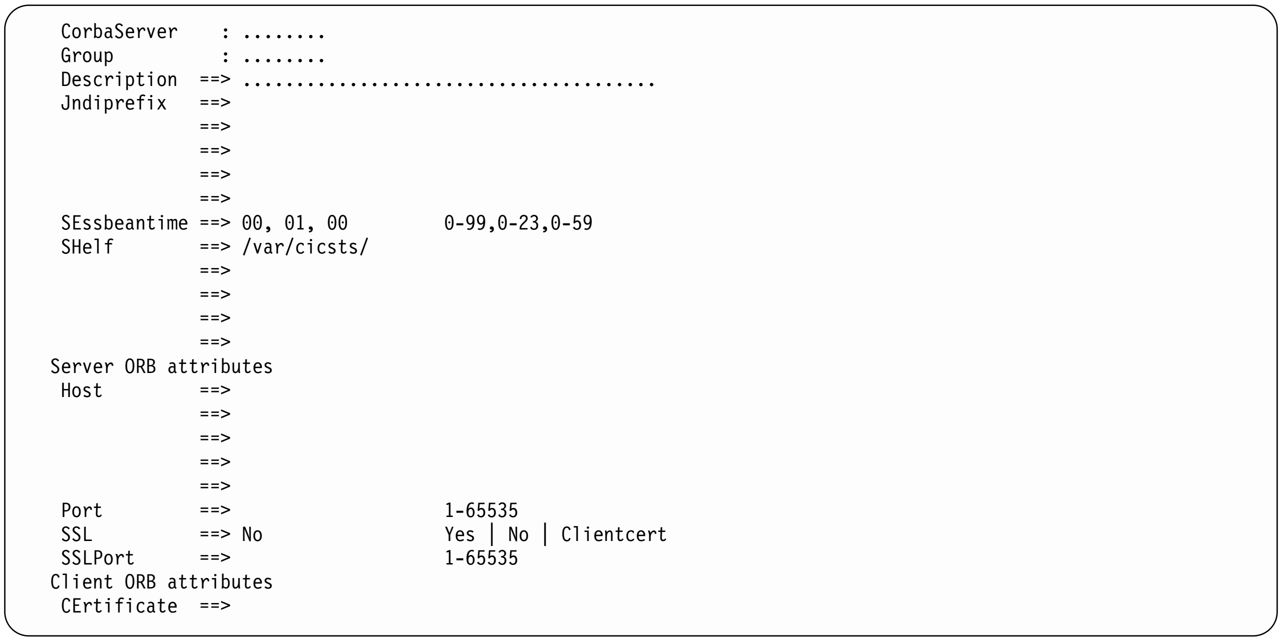
See also the descriptions of the PORT and SSLPORT options.

The definition's Client ORB attributes are used when making outbound method requests on objects in remote EJB or CORBA servers.

Notes:

1. Typically, for enterprise beans, CORBASERVER definitions are created by a deployment tool, as part of the bean deployment process. Usually, you do not have to code them by hand.
2. You can install more than one CORBASERVER definition in the same CICS region.
3. For performance reasons, CICS installs CORBASERVER definitions in two stages. On receipt of an install request, CICS puts the resource into a pending state. Subsequently, possibly after CICS initialization has ended, CICS completes the installation of any pending resources. If this secondary installation stage (which involves clearing the HFS shelf directory used to store copies of JAR files used by the CorbaServer) fails, the resource becomes unusable. The STATE option of the INQUIRE CORBASERVER command returns the current state of the CorbaServer.
4. To update a CORBASERVER definition—that is, to replace an existing definition by installing another of the same name—you must first discard the existing definition.

Figure 6 on page 36 shows the CEDA panel for DEFINE CORBASERVER.



```
 CorbaServer    : ........
 Group          : ........
 Description  ==> ....................................
 Jndiprefix   ==>
              ==>
              ==>
              ==>
              ==>
 SEssbeantime ==> 00, 01, 00        0-99,0-23,0-59
 SHelf        ==> /var/cicsts/
              ==>
              ==>
              ==>
              ==>
Server ORB attributes
 Host         ==>
              ==>
              ==>
              ==>
              ==>
 Port         ==>                   1-65535
 SSL          ==> No                Yes | No | Clientcert
 SSLPort      ==>                   1-65535
Client ORB attributes
 CErtificate  ==>
```

Figure 6. THE CEDA DEFINE CORBASERVER panel

Options:

CERTIFICATE(*label*)
specifies the 1–32 character label of the certificate in the key ring that is to be used (as a client certificate) in the SSL handshake for outbound IIOP connections. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one.

If this option is not specified, the default certificate for the key ring is used.

Notes:

1. If the specified label does not match that of a certificate in the key ring, the CORBASERVER definition cannot be installed.
2. The distinguished name within the specified certificate provides inputs to the distinguished name user-replaceable program, DFHEJDNX.

CORBASERVER(*name*)
specifies the 1-4 character name of the CorbaServer. The acceptable characters are A-Z a-z 0-9. Do not use names beginning with DFH, because these characters are reserved for use by CICS.

DESCRIPTION(*text*)
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and $. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

HOST(*value***)**
specifies the TCP/IP host name, or a string containing the dotted-decimal TCP/IP address, of this logical EJB/CORBA server.

The host name is included in Interoperable Object References (IORs) exported for objects in this logical server. Clients use this host name to access the CICS listener regions.

If you are using connection optimization by means of dynamic DNS registration, to balance client connections across the listener regions of your logical EJB server, specify the generic host name quoted by client connection requests. (The generic host name is the value of the DNSGROUP attribute of the TCPIPSERVICE definition—see the description of the TCPIPSERVICE resource definition.)

JNDIPREFIX(*prefix***)**
specifies a prefix of up to 255 characters to be used at runtime when publishing enterprise beans to the Java Naming and Directory Interface (JNDI). The acceptable characters are A-Z a-z 0-9 . _ /.

If this option is not specified, no prefix is added when publishing beans to JNDI.

PORT(*number***)**
specifies the TCP/IP port number to be used for non-SSL communication to this logical EJB/CORBA server. The port number must be in the range 1–65535. The default is 00683.

You must not specify the same port number for PORT and SSLPORT.

If you install a TCP/IP service on this port, the TCPIPSERVICE definition must specify SSL(NO).

SESSBEANTIME({00,00,00|00,00,10|*dd,hh,mm***})**
specifies, in days, hours, and minutes, the period of inactivity after which a session bean may be discarded by CICS.

00,00,00
Session beans will not be timed out.

00,00,10
Session beans may be discarded after ten minutes of inactivity. This is the default value.

dd,hh,mm
Session beans may be discarded after the specified period of inactivity. The maximum value you can specify is 99,23,59—99 days, 23 hours, and 59 minutes.

SHELF(*directoryname***)**
specifies the 1–255 character fully-qualified name of a directory (a shelf, primarily for JARs) on HFS. The acceptable characters are A-Z a-z 0-9 . _ /. The name is case-sensitive.

CICS regions into which the CORBASERVER definition is installed must have full permissions to the shelf directory—read, write, and execute.

A single shelf can be shared by multiple CICS regions and by multiple CORBASERVER definitions. Each CICS region uses a separate subdirectory of the shelf directory to keep its files separate from those of other CICS regions. The subdirectories for CORBASERVER definitions are contained within the subdirectories of the CICS regions into which they are installed. After a CICS region performs a cold or initial start, it deletes its subdirectories from the shelf before trying to use the shelf.

You should not modify the contents of a shelf that is referred to by an installed CORBASERVER definition. If you do, the effects are unpredictable.

SSL({CLIENTCERT|NO|YES})
specifies the secure sockets layer (SSL) type for this logical EJB/CORBA server:

CLIENTCERT
SSL is used and authentication must be performed using a client certificate. You must specify a value for SSLPORT.

If you install a TCP/IP service on the SSL port, the TCPIPSERVICE definition must specify SSL(CLIENTAUTH) and AUTHENTICATE(CERTIFICATE). (This means that the client is required to send an SSL certificate which maps to an external security manager userid.)

NO
SSL is not used. This CorbaServer does not have an SSL port. This is the default.

YES
SSL is used. You must specify a value for SSLPORT.

If you install a TCP/IP service on the SSL port, the TCPIPSERVICE definition must be specified in one of the following ways:

1. SSL(CLIENTAUTH) and AUTHENTICATE(NO). The client is asked for an SSL certificate and, if it sends one, CICS uses any userid configured for it.
2. SSL(YES) and AUTHENTICATE(NO). SSL is used, but the client is not asked for an SSL certificate.

SSLPORT(*number*)
specifies the TCP/IP port number to be used for SSL communication to this logical EJB/CORBA server. The port number must be in the range 1–65535.

If SSL is NO (the default), the value of this option is ignored.

You must not specify the same port number for PORT and SSLPORT.

DJAR resource definition: Use DJAR to define a deployed JAR file. A deployed JAR file is an ejb-jar file, containing enterprise beans, on which code generation has been performed and which has been stored on the hierarchical file system (HFS) used by z/OS. When you install the DJAR definition, CICS copies the deployed JAR file (specified by HFSFILE) into a subdirectory of the HFS shelf directory of the specified CORBASERVER.

An identical DJAR definition must be installed in each of the (one or more) application-owning regions (AORs) of the logical EJB server.

Notes:

1. Typically, DJAR definitions are created by a deployment tool, as part of the bean deployment process. Usually, you do not have to code them by hand.

2. Installation of the DJAR fails if the deployed JAR file contains a bean with the same name as a bean which is already installed in the specified CORBASERVER.
3. For performance reasons, CICS installs DJAR definitions in two stages. On receipt of an install request, CICS puts the resource into a pending state. Subsequently, possibly after CICS initialization has ended, CICS completes the installation of any pending resources. If this secondary installation stage (which involves copying the deployed JAR file to the HFS shelf directory and parsing the information it contains) fails, the resource becomes unusable. The STATE option of the INQUIRE DJAR command returns the current state of the deployed JAR file.
4. To update a DJAR definition—that is, to replace an existing definition by installing another of the same name—you must first discard the existing definition.

Figure 7 shows the CEDA panel for DEFINE DJAR.

```
DJAR           : ........
Group          : ........
Description  ==> ......................................
CorbaServer  ==>
Hfsfile      ==>
             ==>
             ==>
             ==>
```

Figure 7. The CEDA DEFINE DJAR panel

Options:

CORBASERVER(*name*)
specifies the 1-4 character name of the CorbaServer in which this DJAR is to be installed. The acceptable characters are A-Z a-z 0-9.

DESCRIPTION(*text*)
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

DJAR(*name*)
specifies the 1-8 character name of this DJAR. The acceptable characters are A-Z a-z 0-9 $ @ #.

Do not use DJAR names beginning with DFH, because these characters are reserved for use by CICS.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and $. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

HFSFILE(*filename*)
specifies the 1-255 character fully-qualified file name of the deployed JAR file on HFS. The acceptable characters are A-Z a-z 0-9 . _ /. The name is case-sensitive.

Changes to existing RDO objects

The following existing RDO objects have been changed to support enterprise beans:

- PROFILE
- REQUESTMODEL
- TCPIPSERVICE
- TRANSACTION

PROFILE: The meaning of the RTIMOUT option has been modified:

RTIMOUT({NO|*value*})
specifies the time-out value for:

1. The terminal read time-out feature. If no terminal input is received within the specified interval, the task is terminated and receives an AKCT or AZCT abend.
2. IIOP request processor tasks that are waiting for method requests. If no method request is received within the specified interval, the task is terminated and receives an AIIT abend.

(Note that if a value is specified and you wish to let it default to NO, you must completely delete the value previously specified.)

RTIMOUT has no effect for MRO or basic (unmapped) APPC connections.

NO
Neither terminal reads nor IIOP request processor tasks are timed out.

value
This is an interval (MMSS for minutes and seconds) after which the task is terminated if no input has been received. The maximum value that can be specified is 70 minutes. The value specified in this option is rounded up to units of 16.78 seconds. Thus, the minimum value (after rounding-up) is 16.78 seconds.

REQUESTMODEL definition: A REQUESTMODEL resource definition provides the relationship between an Internet Inter-ORB Protocol (IIOP) inbound request and the CICS transaction that is to be initiated.

IIOP inbound requests may be:
- Method requests for enterprise beans
- Method requests for CORBA stateless objects

The inbound IIOP request is formatted according to CORBA standards; it does not specify a CICS transaction name explicitly. REQUESTMODEL definitions define templates that are compared with the inbound IIOP message to identify the type of request. CICS uses the pattern-matching options (CORBASERVER, TYPE, EJB PARAMETERS, CORBA PARAMETERS, and COMMON PARAMETERS) of the REQUESTMODELs to select the REQUESTMODEL that most closely matches the request.

The TRANSID attribute of the selected REQUESTMODEL specifies the name of a CICS transaction, which associates the IIOP request with a set of execution characteristics such as security, priority, and monitoring data.

For an enterprise bean, a matching RequestModel may be specified for:
- Each method in the bean’s remote interface (including methods inherited from the EJBObject interface)
- Each method in the bean’s home interface (including methods inherited from the EJBHome interface)

For a CORBA stateless object, a matching RequestModel may be specified for each of the object’s public methods.

Note: You can use generic values to define RequestModels that match multiple objects, multiple methods, or both.

Figure 8 shows the CEDA panel for DEFINE REQUESTMODEL.

```
 Requestmodel   : ........
 Group          : ........
 Description  ==> ....................................
 Corbaserver  ==> *
 TYpe         ==> Generic           Corba | Ejb | Generic
EJB PARAMETERS
 Beanname     ==> *
              ==>
              ==>
              ==>
 INTFacetype  ==> Both              Both | Home | Remote
CORBA PARAMETERS
 Module       ==> *
              ==>
              ==>
              ==>
 INTErface    ==> *
              ==>
              ==>
              ==>
COMMON PARAMETERS
 OPeration    ==> *
              ==>
              ==>
              ==>
TRANSACTION ATTRIBUTES
 TRansid      ==> CIRP

CICS TS V1R3 ATTRIBUTES
 OMGModule    :
 OMGInterface :
 OMGOperation :
```

Figure 8. The CEDA DEFINE REQUESTMODEL panel

The following fields may contain generic values consisting of zero or more characters followed by an asterisk (*):
- CORBASERVER
- BEANNAME
- INTERFACE
- MODULE
- OPERATION

The value of the TYPE option determines which, if any, of the EJB PARAMETERS, CORBA PARAMETERS, and COMMON PARAMETERS are valid:

- If TYPE is EJB, only the EJB and COMMON PARAMETERS are valid and the CORBA PARAMETERS must be blank.
- If TYPE is CORBA, only the CORBA and COMMON PARAMETERS are valid and the EJB PARAMETERS must be blank.
- If TYPE is BOTH, the EJB, CORBA, and COMMON PARAMETERS are all valid.

If one of the valid fields is generic (or BOTH in the case of INTFACETYPE), all the valid fields further down the panel (except TRANSID) must be '*' (or BOTH in the case of INTFACETYPE).

If a request is received that matches several REQUESTMODEL definitions, CICS uses the definition most specific to the request. (Where generic values are used, the longest generic pattern results in the most specific match.)

Notes:

1. You cannot install a REQUESTMODEL definition which has identical pattern-matching options (CORBASERVER, TYPE, EJB PARAMETERS, CORBA PARAMETERS, and COMMON PARAMETERS) as an installed REQUESTMODEL with a different name. If you try, the install is rejected and a message is written to CSMT indicating the name of the conflicting REQUESTMODEL.
2. The CORBA PARAMETERS must equal the most derived interface implemented by the objects they are required to match. Matching according to the inheritance hierarchy of an object is not supported.

Options:

BEANNAME(*name***)**
specifies a (possibly generic) bean name, of up to 240 characters, matching the name of the enterprise bean in the XML deployment descriptor. The acceptable characters are A-Z a-z 0-9 _ and accented alphabetic characters. The name is case-sensitive.

If a generic BEANNAME is specified, INTFACETYPE must be BOTH and OPERATION must be '*'.

For CORBA REQUESTMODELs—that is, if TYPE is CORBA—this field should be blank.

CORBASERVER(*name***)**
specifies the (possibly generic) name of the destination CORBASERVER for this REQUESTMODEL. The name can be up to 4 characters in length. The acceptable characters are A-Z a-z 0-9 and (but only at the end) *.

If a generic CORBASERVER is specified, BEANNAME, the CORBA parameters, and the common parameters must all be asterisk (*) or blank; INTFACETYPE must be BOTH or blank.

DESCRIPTION(*text***)**
You can provide a description of the resource you are defining in this field. The DESCRIPTION text can be up to 58 characters in length. There are no restrictions on the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. For each single apostrophe in the text, code two apostrophes.

GROUP(*groupname*)
Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed.

The GROUP name can be up to eight characters in length. The characters allowed are A-Z 0-9 @ # and $. Lowercase characters are treated as uppercase characters. Do not use group names beginning with DFH, because these characters are reserved for use by CICS.

INTERFACE(*text*)
specifies a (possibly generic) name, of up to 255 characters, matching the IDL interface name. The acceptable characters are A-Z a-z 0-9 _ and accented alphabetic characters. Case is significant and should match the original Java or IDL source. However, to comply with CORBA, installation of REQUESTMODELS that specify INTERFACE with values differing only in case from previously installed definitions, will be rejected.

If a generic INTERFACE is specified, OPERATION must be '*'.

For EJB REQUESTMODELs—that is, if TYPE is EJB—this field should be blank.

INTFACETYPE({<u>BOTH</u>|HOME|REMOTE})
specifies the Java interface type for this REQUESTMODEL:

<u>BOTH</u>
matches either the home or remote interface for the bean. OPERATION must be '*'.

HOME
specifies that this is the home interface for the bean.

REMOTE
specifies that this is the remote interface for the bean.

For CORBA REQUESTMODELs—that is, if TYPE is CORBA—this field should be blank.

MODULE(*text*)
specifies a (possibly generic) name, of up to 255 characters, matching the IDL module name (which defines the name scope of the OMG interface and operation). The acceptable characters are A-Z a-z 0-9 _ : and accented alphabetic characters. Case is significant and should match the original Java or IDL source. However, to comply with CORBA, installation of REQUESTMODELS that specify MODULE with values differing only in case from previously installed definitions, will be rejected.

If a generic MODULE is specified, INTERFACE and OPERATION must be '*'.

To indicate the default package, leave this field blank and specify a non-blank (but possibly generic) INTERFACE.

For EJB REQUESTMODELs—that is, if TYPE is EJB—this field should be blank.

OMGINTERFACE(*text*)
This attribute is obsolete, but is supported to provide CSD file compatibility with CICS TS 1.3.

OMGMODULE(*text*)
This attribute is obsolete, but is supported to provide CSD file compatibility with CICS TS 1.3.

OMGOPERATION(*text*)
This attribute is obsolete, but is supported to provide CSD file compatibility with CICS TS 1.3.

OPERATION(*text*)
specifies a (possibly generic) name, of up to 255 characters, matching the IDL operation or bean method name. The acceptable characters are A-Z a-z 0-9 _ and accented alphabetic characters. Case is significant and should match the original Java or IDL source. However, to comply with CORBA, installation of REQUESTMODELS that specify OPERATION with values differing only in case from previously installed definitions, will be rejected.

For information about how to specify operations of overloaded methods, see the *Java to IDL Language Mapping Specification*, published by the Object Management Group (OMG), and available from www.omg.org.

REQUESTMODEL(*name*)
specifies the name of this REQUESTMODEL definition. The name can be up to 8 characters in length. The acceptable characters are A-Z a-z 0-9 $ @ #.

TRANSID(*name*)
defines the 4-character name of the CICS transaction to be used when a new request processor transaction instance is required to process a method request matching the specification of the REQUESTMODEL.

The transaction definition must have as its initial program a JVM program whose JVMClass is com.ibm.cics.iiop.RequestProcessor. It must be installed in all the AORs of the EJB/CORBA server; it need not be installed in listener regions that are not also AORs.

TYPE({BOTH|CORBA|EJB})
specifies the type of REQUESTMODEL:

BOTH
matches both enterprise bean and CORBA requests. BEANNAME, the CORBA parameters, and the common parameters must all be asterisk (*). INTFACETYPE must be BOTH.

CORBA
matches CORBA requests as specified by the CORBA parameters. The EJB parameters must be blank.

EJB
matches enterprise bean requests as specified by the EJB parameters. The CORBA parameters must be blank.

Examples: The following screen shows an example of an enterprise bean-specific REQUESTMODEL definition.

Note: The transaction definition for EJHE should be copied from that of CIRP. Any attributes of the transaction definition can be changed except the program name, which must be that of a JVM program whose JVMClass is com.ibm.cics.iiop.RequestProcessor.

```
 Requestmodel   : DFH$EJB
 Group          : DFH$EJB
 Description  ==> EJB HelloWorld sample
 CorbaServer  ==> EJC1
 TYpe         ==> Ejb              Ejb | Corba | Both
EJB PARAMETERS
 Beanname     ==> HelloWorld
              ==>
              ==>
              ==>
 INTFacetype  ==> Both             Home | Remote | Both
CORBA PARAMETERS
 OMGModule    :
 Module       ==>
              ==>
              ==>
              ==>
 OMGInterface :
 INTErface    ==>
              ==>
              ==>
              ==>
 OMGOperation :
COMMON PARAMETERS
 OPeration    ==> *
              ==>
              ==>
              ==>
TRANSACTION ATTRIBUTES
 Transid      ==> EJHE
```

The next screen shows an example of a stateless CORBA REQUESTMODEL.

Note: The transaction definition for IIHE should be copied from that of CIRP. Any attributes of the transaction definition can be changed except the program name, which must be that of a JVM program whose JVMClass is com.ibm.cics.iiop.RequestProcessor.

```
 Requestmodel   : DFH$IIRH
 Group          : DFH$IIOP
 Description  ==> Hello world CORBA java server sample
 CorbaServer  ==> IIOP
 TYpe         ==> Corba            Ejb | Corba | Both
EJB PARAMETERS
 Beanname     ==>
              ==>
              ==>
              ==>
 INTFacetype  ==>                  Home | Remote | Both
CORBA PARAMETERS
 OMGModule    :
 Module       ==> hello
              ==>
              ==>
              ==>
 OMGInterface :
 INTErface    ==> HelloWorld
              ==>
              ==>
              ==>
 OMGOperation :
COMMON PARAMETERS
 OPeration    ==> *
              ==>
              ==>
              ==>
TRANSACTION ATTRIBUTES
 Transid      ==> IIHE
```

The following screen shows an example of a generic definition that matches any enterprise bean or stateless CORBA object request. This example changes the default request processor transaction from CIRP to EJB1.

Note: The transaction definition for EJB1 should be copied from that of CIRP. Any attributes of the transaction definition can be changed except the program name, which must be that of a JVM program whose JVMClass is com.ibm.cics.iiop.RequestProcessor.

```
 Requestmodel   : GENERIC
 Group          : TEST
 Description  ==> Generic default definition
 CorbaServer  ==> *
 TYpe         ==> Both               Ejb | Corba | Both
EJB PARAMETERS
 Beanname     ==> *
              ==>
              ==>
              ==>
 INTFacetype  ==> Both               Home | Remote | Both
CORBA PARAMETERS
 OMGModule    :
 Module       ==> *
              ==>
              ==>
              ==>
 OMGInterface :
 INTErface    ==> *
              ==>
              ==>
              ==>
 OMGOperation :
COMMON PARAMETERS
 OPeration    ==> *
              ==>
              ==>
              ==>
TRANSACTION ATTRIBUTES
 Transid      ==> EJB1
```

TCPIPSERVICE: A new option, PROTOCOL, is added:

PROTOCOL({HTTP|IIOP})
specifies the protocol to be used by this TCPIP service.

HTTP
Hypertext transfer protocol.

IIOP
Internet inter-ORB protocol.

The description of the PORTNUMBER option is changed as follows:

PORTNUMBER(*value*)
specifies the decimal number of the port on which CICS is to listen for incoming client requests.

Notes:

1. The well-known ports are those from 0 through 1023. Do not use the well-known port range for IIOP TCPIPSERVICES unless you use the well known ports allocated to IIOP—that is, 683 (non-SSL) or 684 (SSL).
2. Take care to resolve any conflicts with other IIOP servers on the same MVS image that also use the well-known ports.

3. Port sharing must be enabled for any port that you want to share across the CICS systems within an MVS image.

TRANSACTION: A new option, OTSTIMEOUT, has been added:

OTSTIMEOUT({NO|*0–240000*})
specifies, in hours, minutes, and seconds, the default period an Object Transaction Service (OTS) transaction, created in an Enterprise JavaBeans environment and executing as a task under this CICS transaction, is allowed to execute without the initiator of the OTS transaction taking a syncpoint (or rolling back the OTS transaction). If the specified period elapses, CICS purges the task.

The initiator of the OTS transaction may be:
- The client of the enterprise bean.
- The EJB container. (The container issues a syncpoint at the end of the bean method.)
- A session bean that manages its own OTS transactions.

Figure 9 shows an OTS transaction. If the period specified by OTSTIMEOUT expires before the initiator of the OTS transaction commits or rolls back the transaction, CICS purges the task.

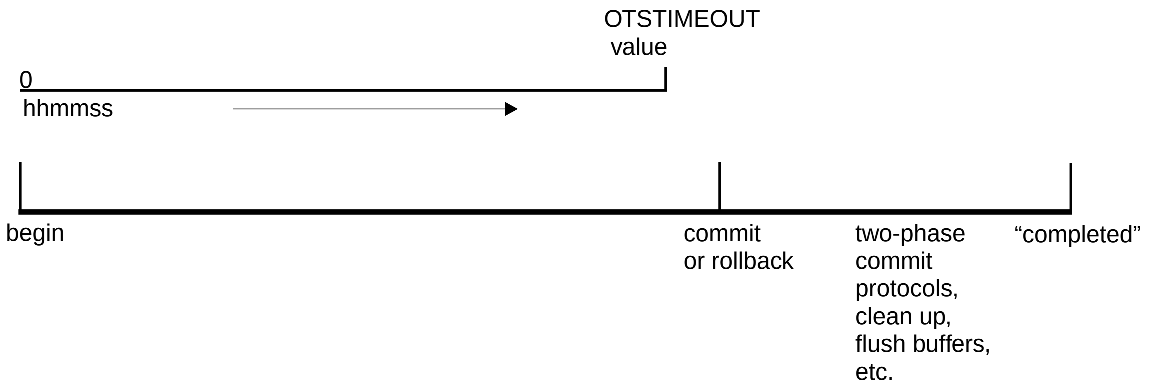


Figure 9. An OTS transaction. If the period specified by OTSTIMEOUT expires before the initiator of the OTS transaction commits or rolls back the OTS transaction, CICS purges the task.

Methods of session beans that manage their own OTS transactions can override the default timeout value by using the set_timeout method of the javax.Transaction.UserTransaction class.

NO
OTS transactions will not time out. This is the default.

0–240000
The period of time (in HHMMSS format) before the task is purged. The maximum period is 24 hours (240000).

Other changes to resource definition

File definitions: The following files must be defined to CICS:

DFHEJDIR, the EJB directory
is a file containing a request streams directory which must be shared by all the listener regions and AORs in the CICS logical EJB/CORBA server. You must define it as recoverable.

DFHEJOS, the EJB object store
is a file of stateful session beans that have been passivated. It must be shared by all the AORs in the CICS logical EJB/CORBA server. You must define it as non-recoverable.

To share DFHEJDIR and DFHEJOS across multiple regions, you could, for instance, use any of the following methods:
- Define them as remote files in a file-owning region (FOR)
- Define them as coupling facility data tables
- Use VSAM RLS.

There are sample FILE definitions for DFHEJDIR and DFHEJOS in the CICS-supplied RDO group, DFHEJVS. There are sample coupling facility FILE definitions for DFHEJDIR and DFHEJOS in the CICS-supplied RDO group, DFHEJCF. There are sample VSAM RLS FILE definitions for DFHEJDIR and DFHEJOS in the CICS-supplied RDO group, DFHEJVR. (DFHEJVS, DFHEJCF, and DFHEJVR are not included in the default CICS startup group list, DFHLIST.)

Changes to DFHFCT resource definition macros: There are changes to the following DFHFCT resource definition macros:

DFHFCT TYPE=INITIAL
The MIGRATE operand is removed.

DFHFCT TYPE=GROUP
This macro is removed.

DFHFCT TYPE=REMOTE
This macro is removed.

The resource management global user exit: The resource management global user exit, XRSINDI, is invoked at new points:
- When CORBASERVER resource definitions are installed or discarded.
- When DJAR resource definitions are installed or discarded.
- When the beans in a DJAR are installed or discarded (even though beans are not directly installable through RDO). The exit is called once for each bean in the JAR. This happens after a DJAR definition has been installed and before it is discarded.

Changes to the application programming interface

CICS programs that execute as part of an enterprise bean method (by means of a JCICS LINK command, for example) are restricted to using the DPL subset of the EXEC CICS API. In particular, they must not issue EXEC CICS SYNCPOINT commands.

New Java API commands

CICS supports the Enterprise JavaBeans API Version 1.1, including the javax.transaction.UserTransaction interface of the Java Transaction API (JTA), Version 1.0. These APIs are for use by developers of enterprise beans. The javax.transaction.UserTransaction interface can be used only by session beans that manage their own OTS transactions.

Changes to the system programming interface

New SPI commands

The following new system programming commands are introduced:

EXEC CICS CREATE CORBASERVER
Builds a CORBASERVER definition in the local CICS region, without reference to data in the CICS system definition (CSD) file. If the named CORBASERVER already exists, it is replaced by the new definition. Any DJARs and beans installed in the old CORBASERVER are discarded.

EXEC CICS CREATE DJAR
Builds a DJAR definition in the local CICS region, without reference to data in the CICS system definition (CSD) file. If the named DJAR already exists, it is replaced by the new definition. Any beans from the old DJAR are discarded; any beans in the new DJAR are installed.

EXEC CICS DISCARD CORBASERVER
Removes a CORBASERVER definition from the local CICS region, together with any associated DJAR definitions and beans.

EXEC CICS DISCARD DJAR
Removes a DJAR definition from the local CICS region, together with any associated beans.

EXEC CICS INQUIRE BEAN
Determines whether a specified bean is installed in a specified CORBASERVER and, if so, which DJAR was installed to install the bean.

The browse form of the command is also supported.

EXEC CICS INQUIRE CORBASERVER
Retrieves the attributes of a specified CORBASERVER definition. A CORBASERVER is an execution environment for enterprise Java beans or for stateless CORBA objects.

The browse form of the command is also supported.

EXEC CICS INQUIRE DJAR
Retrieves the attributes of a deployed JAR file (DJAR).

The browse form of the command is also supported.

EXEC CICS PERFORM CORBASERVER
Performs a specified action (PUBLISH or RETRACT) on the beans in a CORBASERVER.

EXEC CICS PERFORM DJAR
Performs a specified action (PUBLISH or RETRACT) on a deployed JAR file.

EXEC CICS SET CORBASERVER
Sets the timeout value for the session beans in a specified CORBASERVER.

Restriction: There is a restriction on the use of some of the new SPI commands. The SPI commands affected are:
- CREATE CORBASERVER
- DISCARD CORBASERVER
- PERFORM CORBASERVER (PUBLISH and RETRACT)
- CREATE DJAR
- DISCARD DJAR
- PERFORM DJAR (PUBLISH and RETRACT)

Because only one program-link-level of a CICS transaction instance can contain a JVM program, SPI commands that invoke JVM programs cannot be invoked from a transaction instance which already has a JVM program on the program-link stack. For example, it is not valid for an enterprise bean to link to a COBOL program which then invokes one of the above SPI commands.

If this restriction is infringed, CICS makes an exception trace entry, issues message DFHII0002E, and takes a system dump. The trace entry and message indicate that the link to the request processor JVM program DFJIIRP failed. Thereafter the failing SPI command behaves unpredictably.

Modified SPI commands

The following existing system programming commands have been modified:

EXEC CICS COLLECT STATISTICS

You can specify the following new resource type on this command:
- CORBASERVER

EXEC CICS CREATE REQUESTMODEL

The following new options have been added:
- BEANNAME
- CORBASERVER
- INTERFACE
- INTFACETYPE
- MODULE
- OPERATION
- TYPE

The OMGINTERFACE , OMGMODULE, and OMGOPERATION options are obsolete.

For the meanings of these options, see the description of the REQUESTMODEL resource definition on page 40.

EXEC CICS CREATE TRANSACTION

The following new option has been added:
- OTSTIMEOUT

For the meanings of this option, see the description of the TRANSACTION resource definition on page 47.

EXEC CICS INQUIRE REQUESTMODEL

The following new options have been added:
- BEANNAME
- CORBASERVER
- INTERFACE
- INTFACETYPE
- MODULE
- OPERATION
- TYPE

The OMGINTERFACE , OMGMODULE, and OMGOPERATION options are obsolete.

EXEC CICS INQUIRE TRACETYPE

You can use the following new CICS component identifiers with this command:

Table 3. CICS component names and IDs

Component ID	Component name
EJ	Enterprise Java domain
II	IIOP domain
OT	Object Transaction Service domain
RZ	Request streams domain

Table 3. CICS component names and IDs (continued)

Component ID	Component name
SJ	CICS JVM domain

EXEC CICS INQUIRE TRANSACTION

The following new option has been added:

OTSTIMEOUT(*data-area*)

returns a fullword binary field containing the default period in seconds that an Object Transaction Service (OTS) transaction created in an Enterprise JavaBeans environment and executing under this CICS transaction is allowed to execute without the initiator of the OTS transaction taking a syncpoint (or rolling back the OTS transaction).

A value of zero indicates that OTS transactions will not time out.

EXEC CICS INQUIRE UOW

The following new option has been added. It returns information about the OTS transaction associated with the unit of work.

OTSTID(*data-area*)

returns the first 128 bytes of the transaction identifier (TID) of the OTS transaction of which the UOW is part.

If the TID is less than 128 bytes, it is padded on the right with binary zeros.

EXEC CICS INQUIRE UOWLINK

The following new option has been added. It returns information about the partner in the OTS transaction associated with a distributed unit of work.

HOST(*data-area*)

returns, for a TYPE of IIOP, the TCP/IP hostname, or a string containing the dotted decimal TCP/IP address, used to refer to the participant in the OTS transaction. This is useful for identifying the participant, especially when problems occur.

This is a 255–character data area. Strings of fewer than 255 characters are padded with blanks.

Note that the UOW and the participant may belong to the same CORBASERVER.

For TYPE values other than IIOP, HOST returns blanks.

EXEC CICS PERFORM STATISTICS RECORD

You can specify the following new resource type on the command:
- CORBASERVER

EXEC CICS SET TRACETYPE

You can use the following new CICS component identifiers with this command:

Table 4. CICS component names and IDs

Component ID	Component name
EJ	Enterprise Java domain
II	IIOP domain
OT	Object Transaction Service domain
RZ	Request streams domain
SJ	CICS JVM domain

Changes to CICS supplied transactions

New commands

The following new versions of the CEMT transaction are introduced:

CEMT DISCARD CORBASERVER
Removes a CORBASERVER definition from the local CICS region, together with any associated DJAR definitions and beans.

CEMT DISCARD DJAR
Removes a DJAR definition from the local CICS region, together with any associated beans.

CEMT INQUIRE BEAN
Determines whether a specified bean is installed in a specified CORBASERVER and, if so, which DJAR was installed to install the bean.

CEMT INQUIRE CORBASERVER
Retrieves the attributes of a specified CORBASERVER definition. A CORBASERVER is an execution environment for enterprise Java beans or for stateless CORBA objects.

CEMT INQUIRE DJAR
Retrieves the attributes of a deployed JAR file (DJAR).

CEMT PERFORM CORBASERVER
Performs a specified action (PUBLISH or RETRACT) on the beans in a CORBASERVER.

CEMT PERFORM DJAR
Performs a specified action (PUBLISH or RETRACT) on a deployed JAR file.

CEMT SET CORBASERVER
Sets the timeout value for the session beans in a specified CORBASERVER.

Modified commands

The following existing CEMT commands have been modified:

CEMT INQUIRE REQUESTMODEL
The following new options have been added:
- BEANNAME
- CORBASERVER
- INTERFACE
- INTFACETYPE
- MODULE
- OPERATION
- TYPE

The OMGINTERFACE , OMGMODULE, and OMGOPERATION options are obsolete.

CEMT INQUIRE TRANSACTION
The following new option has been added:

OTSTIMEOUT(*value*)
displays the default period in seconds that an Object Transaction Service (OTS) transaction created in an Enterprise JavaBeans environment and executing under this CICS transaction is allowed to execute without the initiator of the OTS transaction taking a syncpoint (or rolling back the OTS transaction).

A value of zero indicates that OTS transactions will not time out.

CEMT INQUIRE UOW

The following new option has been added. It returns information about the OTS transaction associated with the unit of work.

OTSTID(*value*)

displays the first 128 bytes of the transaction identifier (TID) of the OTS transaction of which the UOW is part.

CEMT INQUIRE UOWLINK

The following new option has been added. It returns information about the OTS transaction associated with the partner in the distributed unit of work.

HOST(*value*)

displays, for a TYPE of IIOP, the TCP/IP hostname, or a string containing the dotted decimal TCP/IP address, used to refer to the participant in the OTS transaction. This is useful for identifying the participant, especially when problems occur.

Note that the UOW and the participant may belong to the same CORBASERVER.

For TYPE values other than IIOP, HOST displays blanks.

CEMT PERFORM STATISTICS

You can specify the following new resource type on this command:

- CORBASERVER

CETR

You can use the following new CICS component identifiers with this command:

Table 5. CICS component names and IDs

Component ID	Component name
EJ	Enterprise Java domain
II	IIOP domain
OT	Object Transaction Service domain
RZ	Request streams domain
SJ	CICS JVM domain

Changes to global user exits

The resource management global user exit, XRSINDI , is invoked at new points:

- When CORBASERVER resource definitions are installed or discarded.
- When DJAR resource definitions are installed or discarded.
- When the beans in a DJAR are installed or discarded (even though beans are not directly installable through RDO). The exit is called once for each bean in the JAR. This happens after a DJAR definition has been installed and before it is discarded.

Changes to the exit programming interface (XPI)

A new parameter is added to the INQUIRE_TRANDEF function of the DFHXMXDX macro:

OTSTIMEOUT(*name4*)

returns the default period in seconds that an Object Transaction Service (OTS) transaction created in an Enterprise JavaBeans environment and executing under this CICS transaction is allowed to execute without the initiator of the OTS transaction taking a syncpoint (or rolling back the OTS transaction).

name4
The name of a 4-byte location to receive the timeout setting, expressed as a binary value.

Rn
A register to receive the timeout setting, expressed as a binary value.

A value of zero means that the transaction resource definition specifies OTSTIMEOUT(NO).

Changes to user-replaceable programs

The distinguished name program, DFHEJDNX

There is a new user-replaceable program, the distinguished name program, DFHEJDNX. DFHEJDNX is used to obtain a string representation of the distinguished name of an EJB client, when the client has not presented an X.509 certificate containing a name. A sample program is provided.

The security program for IIOP

The communications area passed to the security program for IIOP has been extended. You can now use your security program to assign user IDs to incoming requests for EJB objects, as well as to requests for CORBA objects.

On a successful return from the security program, a syncpoint is now taken, if the return code indicates permission to continue.

The dynamic and distributed routing programs

The communications area passed to the dynamic and distributed routing programs, DFHDYP and DFHDSRP, has been changed. Some of the fields may now contain new, IIOP-related, values.

You can now use a customized version of the distributed routing program, DFHDSRP, to dynamically route EJB work (OTS transactions) and requests for CORBA stateless objects across the AORs in a logical EJB/CORBA server.

Changes to monitoring

New fields are introduced into performance-class monitoring records in groups DFHSOCK and DFHTASK. The new fields are described in the *CICS Performance Guide*.

Changes to statistics

CORBASERVER statistics

These statistics are available online, and are mapped by the DFHEJRDS DSECT. They are described in the *CICS Performance Guide*.

Changes to problem determination

Messages

There are some new CICS messages in the following ranges:

- DFHEJ0001—DFHEJ1299
- DFHII0001—DFHII1020
- DFHOTxxxx—DFHOTxxxx
- DFHRZxxxx—DFHRZxxxx

Messages in the following ranges are removed:

- DFHAP1400—DFHAP1409
- DFHCA5171—DFHCA5173
- DFHCA5267—DFHCA5269
- DFHCZ0359
- DFHFC105
- DFHFC206—DFHFC207

All messages are described in the *CICS Messages and Codes* manual.

Abend codes

There are some new abend codes:

- AII1—AII5

Trace points

There are new CICS trace entries in the following ranges:

- AP 1800—AP 180F
- EJ 0600—EJ 060F
- II 0000—II 100F
- OT 1000—OT 100F

These trace points are generated in a Java environment and then mapped to the listed CICS trace points. This approach means that any number of Java generated trace points can use each of the CICS trace points. For each of the four domains (AP, EJ, II, and OT), the interface defines 15 different trace types. It is also possible to determine whether the trace point is an entry or an exit, which gives a total of 16 different trace point IDs (RASITraceEvent types). Setting CICS trace at a particular level can generate several of these RASITraceEvent types. As is generally the case in CICS, these trace levels are exclusive, for example trace level 2 does not include trace level 1. These trace types and the corresponding CICS trace levels are explained in table Table 6.

Table 6. Types of Java generated trace points (RASITraceEvent types) and the associated CICS trace levels

RASITraceEvent type	CICS trace level	Meaning
TYPE_API	1	Defines an application programming interface (API) trace point.
TYPE_CALLBACK	2	Defines a callback method trace point
TYPE_ENTRY_EXIT	1	Defines method entry and exit trace points
TYPE_ERROR_EXC	off	Defines an error or exception condition trace point
TYPE_MISC_DATA	2	Defines a miscellaneous data trace point
TYPE_OBJ_CREATE	2	Defines an object creation (constructor) trace point
TYPE_OBJ_DELETE	2	Defines an object deletion trace point
TYPE_PRIVATE	2	Defines a private method trace point
TYPE_PUBLIC	2	Defines a public method trace point. (This typically includes package and protected scope, as all of these methods may be used by other classes.)
TYPE_STATIC	2	Defines a static method trace point

Table 6. Types of Java generated trace points (RASITraceEvent types) and the associated CICS trace levels (continued)

RASITraceEvent type	CICS trace level	Meaning
TYPE_SVC	2	Defines a service code trace point. Service code is generally ″low-level″ code which provides commonly used services to other classes
TYPE_LEVEL1	1	Defines a ″low-detail″ trace point
TYPE_LEVEL2	2	Defines a ″medium-detail″ trace point
TYPE_LEVEL3	3	Defines a ″high-detail″ trace point
TYPE_PERF	2	Defines a performance-monitoring trace point

Trace points in the following ranges are removed:

- DFHAP21E1—DFHAP21E4
- DFHAP0400—DFHAP0404
- DFHAP0680—DFHAP0697

Trace points are listed in the *CICS Trace Entries* manual.

Dump

A new II section, relating to the IIOP domain, is added to CICS system dumps.

Resources on which tasks can wait

Table 7 shows the new resources on which tasks in a CICS system can wait. The resource names and resource types that are shown are the ones that you can see in formatted trace entries or by online inquiry.

Table 7. Resources on which a suspended task might be waiting

Resource type	Resource name	Suspending module	DSSR call WLM wait type	Task
IIRR	SOCBNOTI	DFHIIRR	SUSPEND IO	User
IIRP	NOTI	DFHIIRP	SUSPEND IO	User

A request receiver DFHIIRR task suspends with resource type IIRR and resource name SOCBNOTI when it has no work to do and the TCPIP connection is still open. These are resumed by a NOTIFY gate when IIRR is told there is another request from the sockets domain or a reply has come in from the request streams domain.

A request processor DFHIIRP task suspends with resource type IIRP and resource name NOTI when it is waiting for requests or replies. These are resumed by a NOTIFY gate when IIRP is told there is another request or reply from the request streams domain.

Changes to sample programs

Sample user-replaceable programs

There is a new sample program for the distinguished name user-replaceable program, DFHEJDNX.

The sample security program for IIOP, DFHXOPUS, has been updated to use the new communications area passed to it.

Sample programs for RDO

The sample programs for use with the CICS system definition utility, DFHCSDUP, have been updated. The samples, which can be invoked by DFHCSDUP during EXTRACT processing, have been changed to reflect the new and changed RDO objects.

Chapter 3. Enhancements to CORBA support

This chapter describes enhancements to the CORBA support first introduced into CICS in CICS TS 1.3.

The chapter covers the following topics:

Overview

CICS support for the Internet Inter-ORB protocol (IIOP) allows inbound requests from CORBA clients to CICS Java applications. This function, and the object request broker (ORB) implementation, is enhanced to support the following levels of the Common Object Request Broker Architecture (CORBA):

- CORBA 2.1
- IIOP 1.1

You can find information about IBM WebSphere at:

```
http://www.ibm.com/websphere/
```

You can find information about the CORBA specification at the OMG Web address:

```
http://www.omg.org/library
```

The enhanced function provides the necessary support for enterprise beans, using the remote method invocation (RMI) interface of IIOP. See “Chapter 2. Introduction to Enterprise JavaBeans™” on page 9 for more information about the Enterprise JavaBeans specification and its implementation in CICS.

These CORBA enhancements add new function and make changes that also affect existing IIOP applications, sometimes called **stateless CORBA objects**. The following new function has been added:

- Support for outbound IIOP. CORBA applications can now act as both client and server.
- Support for the CORBA 2.1 API, excluding dynamic invocation interface (DII), dynamic skeleton interface (DSI), and GIOP 1.1 fragments.
- Method invocations can participate in object transaction service (OTS) distributed transactions. If a client calls an IIOP application in the scope of an OTS transaction, information about the transaction flows on the IIOP call. If a target stateless CORBA object implements CosTransactions::TransactionalObject, then the object is treated as transactional.

  Note: An OTS transaction is analogous to a distributed unit of work, not a CICS transaction or resource definition.
- REQUESTMODEL attributes MODULE, INTERFACE, and OPERATION (which replace attributes OMGMODULE, OMGINTERFACE, and OMGOPERATION supported in CICS TS 1.3) can have names up to 255 characters long.

  Note: The changes to the REQUESTMODEL resource definition are such that it is incompatible with CICS TS 1.3. Although you can update a 1.3 definition from a CICS TS 2.1 region, you cannot share a

REQUESTMODEL resource definition for use on both releases. See the *CICS Transaction Server for z/OS Migration Guide* for migration information about REQUESTMODEL resource definitions.

Changes affecting existing IIOP applications

The following changes affect existing IIOP applications:

- CICS programs that execute as part of an IIOP application (by means of a JCICS LINK command, for example) are restricted to using the DPL subset of the EXEC CICS API. In particular, they must not issue EXEC CICS SYNCPOINT commands. This affects IIOP applications that are not subject to workload balancing (which was implemented in terms of DPL) in CICS TS OS/390 Version 1 Release 3. The DPL subset is defined in Appendix G in the *CICS Application Programming Reference*.
- IIOP applications are supported in JVM mode only. The VisualAge for Java, Enterprise Edition for OS/390 bytecode binder cannot be used.
- A CORBASERVER resource definition is needed to define the execution environment of the IIOP application.
- The REQUESTMODEL resource definition is changed. The OMGMODULE, OMGINTERFACE, and OMGOPERATION attributes are renamed to MODULE, INTERFACE, and OPERATION. New fields are added to identify the related CORBASERVER and to support Enterprise JavaBeans.

  Generic pattern matching has been changed to allow only zero or more characters followed by an asterisk (*). In cases where several different generic patterns match a given string, there is now a simple rule for choosing the most specific match. The longest generic pattern results in the most specific match.
- The new PROTOCOL parameter of the TCPIPSERVICE resource definition for the IIOP port must be set to IIOP.
- The offline GenFacIOR utility is no longer needed to build an Interoperable Object Reference (IOR) to be used by the client to access the object. An IOR (genfac.ior file) is created by CICS when the CORBASERVER resource definition is installed. This can be published to JNDI using the PERFORM CORBASERVER PUBLISH command, or can be downloaded to the client system. All existing stringified IOR files need to be recreated.
- Any user-replaceable module (URM) implementations for IIOP security must be changed to support the updated COMMAREA structure. The URM is now called only if it is specified in the TCPIPSERVICE definition for the IIOP port. It is no longer possible to update the transaction identifier from the URM.

Note: Listener regions and AORs that form part of a CICS IIOP server must both be at the CICS TS 2.1 level.

Supported client and server platforms

At execution time the client (for inbound) or target server (for outbound) must satisfy the following requirements :

- Provide an ORB compliant with the CORBA 2.1 specification, and support IIOP 1.1.
- Be able to resolve homes from the naming server or store homes in the naming server, such as a Windows NT CosNaming Server.
- Provide a compatible transaction service if a transaction is to be shared between a client and CICS, or CICS and an outbound target server.
- Support SSL if secure communications are required.

IIOP request processing

The following diagram shows the IIOP request flow in CICS TS OS/390 Version 1 Release 3:

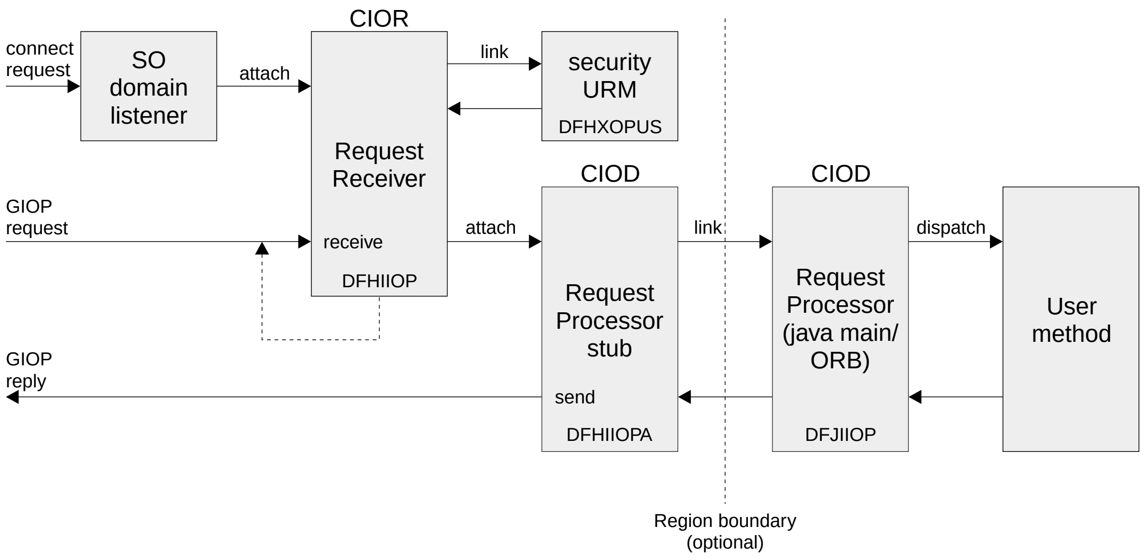


Figure 10. The CICS TS Release 1.3 IIOP execution flow

The enhanced IIOP request processing is similar in concept, with the following main differences:

- The CIRR request receiver does not attach a transaction directly. It creates or joins a *request stream* to allow appropriate OTS transaction and security context to be associated with the *request processor* task. The request stream logic uses a directory to relate IIOP requests and OTS transactions, so that requests that are part of the same transaction are routed to the same request processor. You need to provide a resource definition for DFHEJDIR for this file, and provide an appropriate VSAM dataset.
- The CIRR request receiver terminates when it has no further work to do, instead of waiting for further work. It is attached by the socket domain when the connection is first created and when there are further client requests to process for the connection.

The following diagram shows the new IIOP request flow:

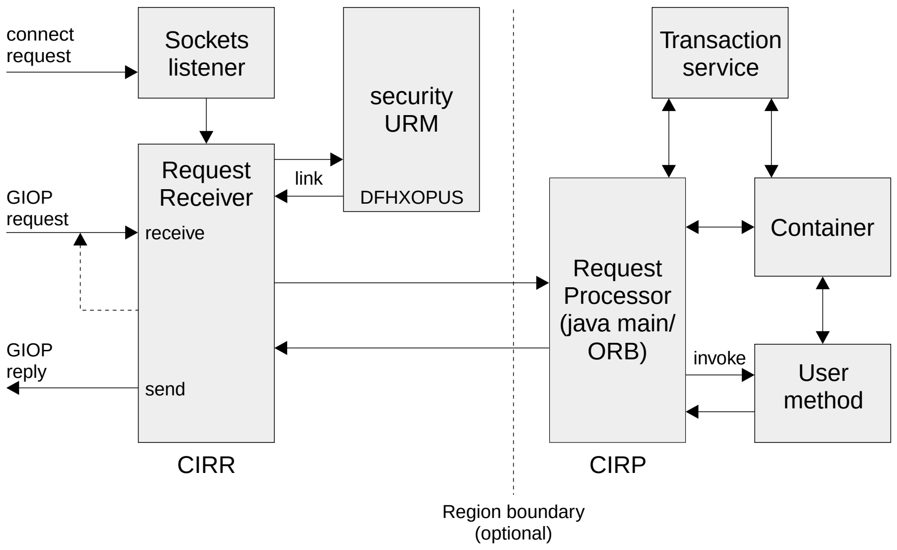


Figure 11. The CICS TS Version 2 IIOP execution flow

Workload balancing

Workload balancing of requests is implemented at three levels:

TCP/IP port sharing
TCP/IP port sharing is provided by Communications Server for OS/390.

Domain name server (DNS) connection optimization for TCP/IP
This facility balances IP connections and workload in a sysplex domain. The initial interoperable object reference (IOR) to the CICSplex contains a generic host name and port number. With DNS connection optimization, multiple CICS regions are started to listen for IIOP requests on the same port (using virtual IP addresses), and the host name in the initial IOR is resolved to an IP address by MVS DNS and workload management (WLM) services.

Connection optimization in a sysplex domain is described in the *OS/390 V2R8.0 SecureWay CS IP Configuration, SC31-8513-03*.

CICS dynamic routing
This facility routes the flow from request receiver to request processor across CICS regions. Dynamic selection of the target is provided by CICS or CICSPlex SM dynamic routing services, which select the least loaded or most efficient application region.

Benefits

Some of the benefits arising from the enhancements to the CORBA support in CICS are:

- Support for the Enterprise JavaBeans specification, providing component portability and improved application development productivity allied with the established strengths of CICS in security, reliability and scalability
- Support for outbound CORBA requests from both Enterprise JavaBeans and stateless CORBA objects
- Support for the CORBA object transaction service for both Enterprise JavaBeans and stateless CORBA objects

Requirements

The main software requirement for the execution of enterprise beans is IBM Developer Kit for OS/390, Java 2 Technology Edition, Version 1.3.

Changes to CICS externals

There are changes to a number of CICS external interfaces that support the CORBA server and related Enterprise JavaBeans implementations. Most of these are described in the Enterprise JavaBeans chapter in “Changes to CICS externals” on page 32. The changes to the TCPIPSERVICE resource definition are described here:

The following new parameters are added to TCPIPSERVICE:

AUTHENTICATE(NO|BASIC|CERTIFICATE|AUTOREGISTER|AUTOMATIC)
specifies the level of authentication required on connections associated with this TCPIPSERVICE definition.

NO
No authentication of the client is required. However, if a registered certificate is provided by the client, it is used. This is the default.

BASIC
HTTP Basic Authentication of the client is attempted. If the client has sent an HTTP Authorization header, its contents are decoded as a user id and password. If these are valid, the user id is passed to the user-replaceable module for this TCPIPSERVICE definition. Otherwise, an HTTP 401 response is returned, together with a WWW-authenticate header, which causes the browser program to prompt the user for a new user id and password. These are returned in the required Authorization header. This process continues until the client either supplies a valid user id and password, or cancels the connection.

CERTIFICATE
A valid X.509 client certificate is required from the client, and it must map to a valid trusted user id in the external security manager’s database. If such a certificate is not received, the connection is rejected with an HTTP 403 response. Otherwise, the derived user id is passed to the user-replaceable module for this TCPIPSERVICE definition.

This attribute cannot be specified unless SSL(CLIENTAUTH) is also specified.

AUTOREGISTER
This allows the client to register a certificate automatically. If the client presents a certificate that is not registered, an HTTP Basic Authentication dialogue is entered, in which the client must enter the user id for the certificate to be registered, together with its corresponding password. If this dialogue is completed successfully, the certificate is registered to the specified user id.

This attribute cannot be specified unless SSL(CLIENTAUTH) is also specified.

AUTOMATIC
This combines the AUTOREGISTER and BASIC functions. It attempts to authenticate the client as best it can. If a registered certificate is available, it is used. Otherwise, Basic Authentication is used to prompt

the client for a user id and password. If an unregistered certificate is used, and Basic Authentication is successful, the certificate is registered.

DNSGROUP
specifies the location parameter passed on the IWMSRSRG register call to Workload Manager. The value may be up to 18 characters, and any trailing blanks are ignored. This parameter is referred to as group_name by the TCP/IP DNS documentation and is the name of a cluster of equivalent server applications in a sysplex. It is also the name within the sysplex domain that clients use to access the CICS TCPIPSERVICE.

More than one TCPIPSERVICE may specify the same group name. The register call is made to WLM when the first service with a specified group name is opened. Subsequent services with the same group name do not cause more register calls to be made. The deregister action is dictated by the GRPCRITICAL attribute. It is also possible to explicitly deregister CICS from a group by issuing a master terminal or SPI command.

GRPCRITICAL(NO|YES)
marks the service as a critical member of the DNS group, meaning that this service closing or failing causes a deregister call to be made to WLM for this group name. The default is NO, allowing two or more services in the same group to fail independently and CICS still remains registered to the group. Only when the last service in a group is closed is the deregister call made to WLM, if it has not already been done so explicitly. Multiple services with the same group name can have different GRPCRITICAL settings. The services specifying GRPCRITICAL(NO) can be closed or fail without causing a deregister. If a service with GRPCRITICAL(YES) is closed or fails, the group is deregistered from WLM.

PROTOCOL
identifies to CICS the type of service to be provided on the TCP/IP port. Values are:

HTTP connections are handled by CICS Web support.

IIOP connections are handled by CICS IIOP support. IIOP is required for TCPIPSERVICEs that are to accept inbound requests for enterprise beans.

Changes to the IIOP sample resource definitions

There are changes to the IIOP sample resource definitions group, DFH$IIOP. See the *CICS Transaction Server for z/OS Migration Guide* for details.

Chapter 4. The CICS Connector for CICS TS

This chapter describes the CICS Connector for CICS TS. It covers the following topics:

- "Overview"
- "Using the CICS Connector for CICS TS's CCF interface" on page 69
- "Using the CTG API" on page 76
- "Benefits" on page 79
- "Requirements" on page 79
- "Restrictions and recommendations for the CICS Connector for CICS TS" on page 79
- "Installation" on page 80
- "Changes to CICS externals" on page 81

Overview

The CICS Connector for CICS TS helps you to build Enterprise JavaBean (EJB) server components that make use of existing CICS programs.

Previous releases of CICS support **CICS connectors** that enable a Java client program, *running outside CICS* (on Windows NT, OS/2®, UNIX™, or native OS/390), to connect to a specified program on a CICS server. CICS Transaction Server for z/OS introduces a new CICS connector—the CICS Connector for CICS TS—that enables a Java program *running on CICS Transaction Server for z/OS* to connect to a specified program on a CICS server.

What are CICS connectors?

A CICS connector is a software component that allows a Java client application to invoke a CICS application. Typically, the Java client programs that use a CICS connector are applets or servlets.

A CICS connector is supported on each of the following platforms: AIX®, OS/2, Windows NT, Solaris, and OS/390. The Java client application may run on the same platform as the connector, or it may run on any Java-enabled platform and drive the connector by means of a gateway process running on one of the listed platforms.

The CICS connectors that run on AIX, OS/2, Windows NT, and Solaris are shipped as part of the CICS Transaction Gateway (CTG) product and connect to CICS on OS/390 using an SNA or TCP62 connection. The CICS connector that runs on native OS/390 is shipped as part of the CICS Transaction Gateway for OS/390 product and connects to CICS on OS/390 using an external CICS interface (EXCI) connection.

In every case, the Java client application using the connector is coded using one of two application programming interfaces (APIs):

1. A lower-level, CICS-specific, API known as the CTG API. The CTG API consists of the external call interface (ECI) and the external presentation interface (EPI).
2. A higher-level API known as the Common Connector Framework (CCF) Client Interface.

   The Common Connector Framework is an IBM architecture that defines a standard way for a Java program to interact with an application server such as CICS, IMS™, or SAP. One of the advantages of this architecture is that it provides a client API with the same style, regardless of the application server that it drives. This enables application development tools, such as VisualAge for Java (VAJ) Enterprise Access Builder, to provide generic tooling independent of the type of application server being accessed.

All the CICS connectors support the CCF Client Interface.

The CICS Connector for CICS TS runs on CICS TS z/OS. Like the other CICS connectors, it provides both the CTG API and the CCF Client Interface.

The recommended way to create a Java application or bean that uses a CICS connector is to use the VAJ Enterprise Access Builder, or a product that offers similar function, to program to the connector's CCF Client Interface. If you do not have such a product, you must use the CTG API.

The Java applications, or beans, that you create are portable across the set of CICS connectors. Thus, for example, a bean that invokes a CICS program on a CICS OS/390 server could be used from:

- An applet in a browser, connecting by means of a gateway process running on any of the platforms supported by the CTG
- A servlet in a Web server running on any of the platforms supported by the CICS connectors
- An enterprise bean in an EJB server running on any of the platforms supported by the CICS connectors
- A CICS Java application or enterprise bean running in CICS TS z/OS.

The CICS Transaction Gateway for OS/390 is described in the *CICS Transaction Gateway for OS/390 Administration* manual, SC34–5528–01. The CICS Transaction Gateway API is described in the *CICS Transaction Gateway Programming Guide*, SC34–5594–00.

The CICS Connector for CICS TS

The CICS Connector for CICS TS allows a Java program or enterprise bean running on CICS TS z/OS to link to a CICS server program. It allows you, for example, to create powerful EJB components that make use of existing CICS programs.

The CICS server program:

- May be written in any of the CICS-supported languages
- Must use a suitable communications area (COMMAREA)
- Must not do any terminal input/output
- Typically, runs on a separate back-end CICS OS/390 region, but optionally may be on the same CICS region as the Java program or bean.

The background—accessing CICS programs from Java

Frequently, new Java applications can be developed more quickly and reliably by harnessing the power of existing (non-Java) CICS programs. Typically, the Java application is network-based, perhaps started from a browser, and the CICS program is written in a language such as COBOL. This section reviews the several ways in which existing CICS OS/390 programs can be accessed from Java code, and shows how the CICS Connector for CICS TS fits into this pattern.

From Java programs outside CICS

From the network, a Java client application or applet can use the CICS connector interface—that is, either the CCF Client Interface or the CTG API—to link to a CICS OS/390 program.

This method is shown in Figure 12. In this example, because the client applet is not running on the same host as the CICS connector, the CICS Transaction Gateway for OS/390 is used to communicate with the connector. The connector uses EXCI to pass requests to CICS.

The picture also shows a Java servlet. It too uses the CICS connector interface to connect to a CICS OS/390 server program. Because the servlet is running on the same host as the connector, it uses the local protocol to communicate with the latter. The CICS Transaction Gateway for OS/390 is bypassed.

The CICS connector for native OS/390 supports the external call interface but not the external presentation interface. Thus, ECI but not EPI calls are supported.

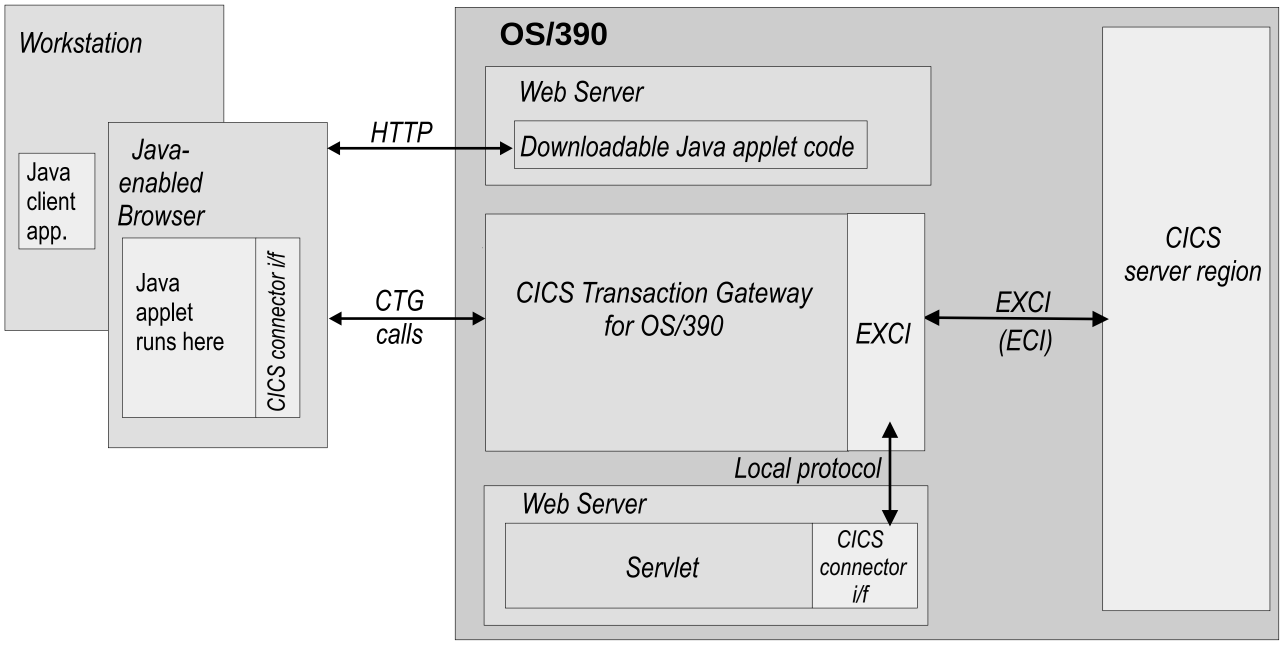


Figure 12. Java clients connect to a CICS server program from outside CICS. A Java applet, running on a browser, uses the CICS connector interface to link to a CICS OS/390 server program. Because the client applet is not running on the same host as the CICS connector, the CICS Transaction Gateway for OS/390 is used to communicate with the connector. The connector uses EXCI to pass requests to CICS. (To CICS, these appear to be ECI calls. Because EXCI is used, the CICS server region must be on the same OS/390 operating system, or Parallel Sysplex, as the connector.)

The picture also shows a Java servlet. It too uses the CICS connector interface to connect to a CICS OS/390 server program. Because the servlet is running on the same host as the connector, it uses the local protocol to communicate with the latter. The CICS Transaction Gateway for OS/390 is bypassed.

A variation is shown in Figure 13 on page 68. In this scenario, the CICS Transaction Gateway runs on an intermediate Windows NT, OS/2, or Solaris machine. On these platforms, the CICS connector uses a CICS Universal Client to pass requests to a back-end CICS OS/390 region. In this setup, the full CTG API (including both ECI and EPI functions) is supported. In other words, the Java client can access

3270-based CICS programs, as well as CICS programs that use a suitable communications area.

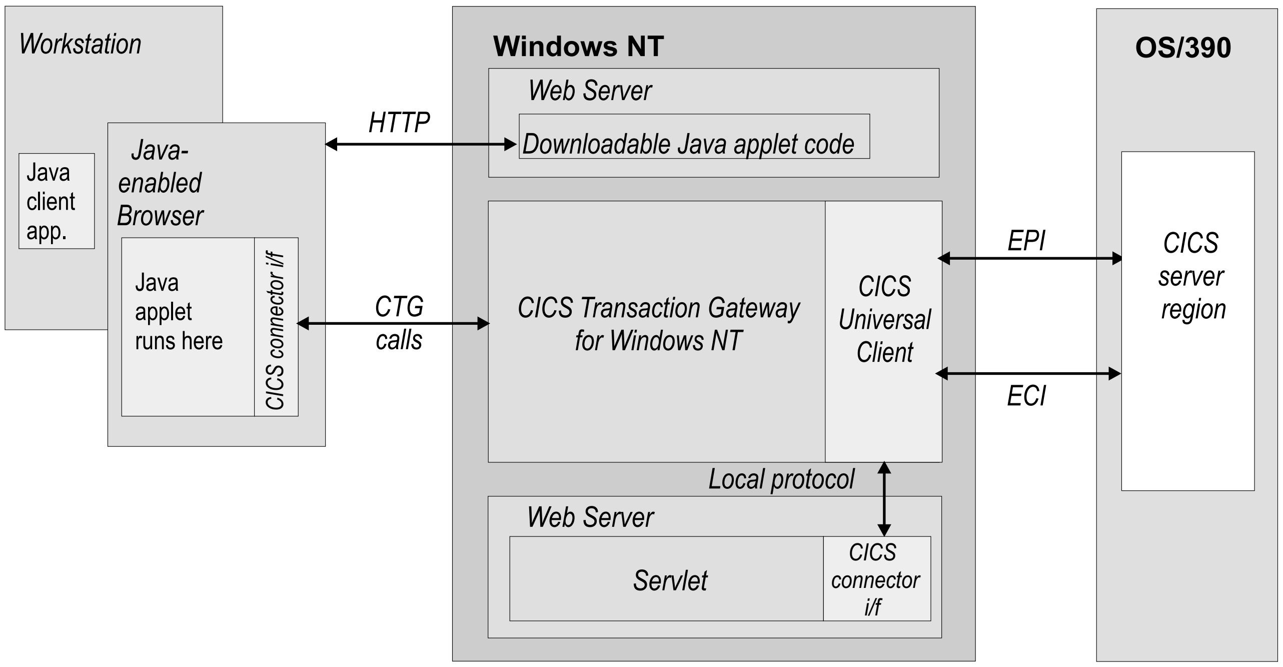


Figure 13. Java clients connect to a CICS OS/390 server program from outside CICS. In this scenario, the CICS Transaction Gateway runs on an intermediate Windows NT, OS/2, or Solaris machine. The CICS connector uses a CICS Universal Client to pass requests to a back-end CICS OS/390 region. Both ECI and EPI calls are supported.

To use CICS programs as servers in this way, the Java programmer requires some knowledge of CICS programming.

From Java programs inside CICS

There are two methods by which Java programs running within CICS can access non-Java CICS programs:

Using JCICS: A CICS Java program or CICS enterprise bean can use the JCICS classes to link to a CICS server program. The server program can be written in any of the CICS-supported languages and be either local or remote. It must use a suitable communications area and must not do any 3270–based terminal input/output.

The Java programmer requires a detailed knowledge of CICS.

Using the CICS Connector for CICS TS: *In CICS Transaction Server for z/OS only*, a Java program or enterprise bean running on CICS can use the CICS Connector for CICS TS to link to a suitable CICS server program. The connector uses a CICS LINK call, rather than EXCI, to access the back-end server program. Link and distributed program link (DPL) calls are supported. This scenario is shown in Figure 14 on page 69.

The CICS Connector for CICS TS uses the connector classes provided with the CICS Transaction Gateway for OS/390; however, because the client program or bean runs on the same host as the connector, it is not necessary, to use the connector, to run the CICS Transaction Gateway for OS/390 as a server application in its own address space.

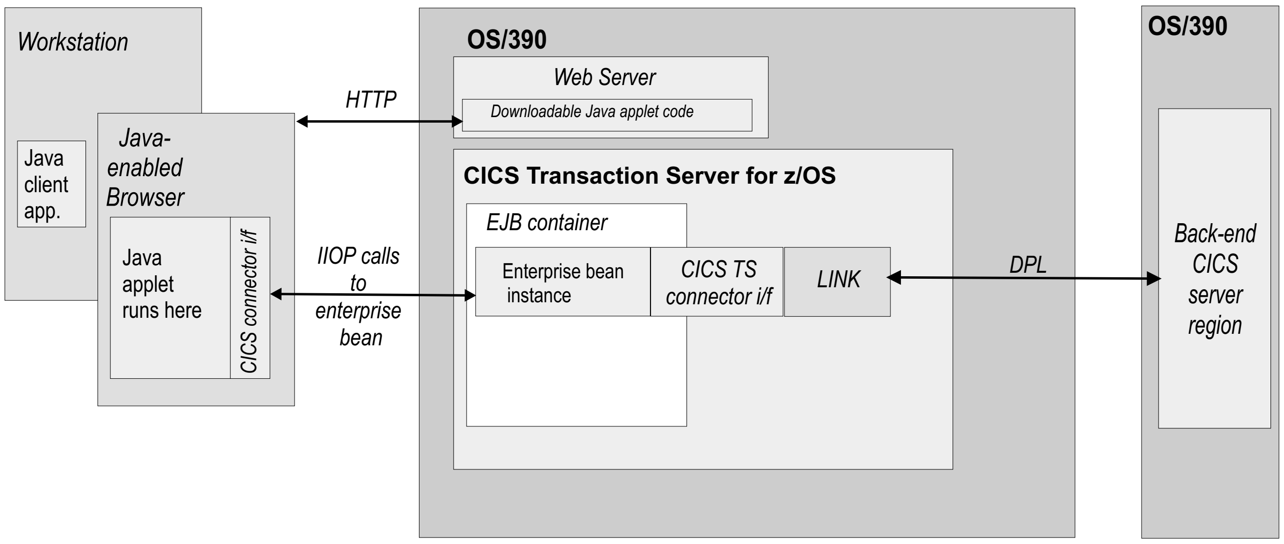


Figure 14. A CICS enterprise bean connects to a CICS server program. This method is only possible in CICS Transaction Server for z/OS.

A Java client application or applet uses RMI/IIOP to create an instance of an enterprise bean, which exists in a CICS EJB container. The enterprise bean uses the CICS Connector for CICS TS to link to a server program on a back-end CICS OS/390 region. The connector issues a DPL call to the back-end region.

There are two ways of using the CICS Connector for CICS TS:

1. Program to the connector's CCF Client Interface, using VisualAge for Java Enterprise Access Builder or a similar product. *This is the recommended method.*
2. Program to the connector's CTG API. Normally, you would use this method only if you do not have access to VAJ EAB or a similar product.

To use the CICS Connector for CICS TS to create an enterprise bean, the Java programmer requires a reasonable knowledge of CICS (although somewhat less than if he were using JCICS). However, the enterprise beans that he creates, using either the connector or the JCICS classes, can be used by Java programmers who have little knowledge of CICS.

Using the CICS Connector for CICS TS's CCF interface

The recommended way to create a Java application that uses any CICS connector—including the CICS Connector for CICS TS—is to use the VisualAge for Java Enterprise Access Builder, or a similar product, to program to the connector's CCF Client Interface. The VAJ Enterprise Access Builder provides visual tools and high-level constructs that mask the complexity of coding CCF objects by hand. VAJ is described in "Introducing VisualAge for Java Enterprise Access Builder" on page 74. The rest of this section describes what you need to know about the CCF Client Interface in order to use the Enterprise Access Builder effectively.

The CCF Client Interface consists of the following classes:

ConnectionSpec

A **ConnectionSpec** object holds all the connection-relevant attributes (for example, hostname and TCP/IP port number) necessary to drive an interaction with a server. It identifies a unique connection.

It is the factory for a **Communication** object.

The CICS Connector for CICS TS's **ConnectionSpec** class is called **CICSConnectionSpec**.

InteractionSpec
An **InteractionSpec** object holds all the interaction-relevant attributes (for example, the name of the target program and the mode of the interaction—send or receive) necessary for an interaction with a server. It is passed as an argument to a **Communication** object when a particular interaction is to be carried out.

The CICS Connector for CICS TS's **InteractionSpec** class is called **ECIInteractionSpec**.

Communication
The **execute** method of a **Communication** object allows you to drive an interaction with a server. The **execute** method takes three arguments—an **InteractionSpec** that specifies the type of interaction, and **input** and **output** objects that carry the exchanged data.

input/output
Input and **output** objects are beans that hold the data exchanged with the target program. The data is accessible via the bean's property access methods.

The implementation of the beans may be based on the Java Record Library, or may be proprietary to a particular CCF connector.

VAJ Enterprise Access Builder provides tools to import a data descriptor (such as a COBOL copy book) that represents the communications area of the target program. From the data descriptor, you can construct record beans that are used to build and decode the target program data.

Here's an outline of the program logic needed for a single interaction with an application server, using the standard CCF Client Interface classes supported by a CICS connector. This is the logic a CICS enterprise bean would use to access a back-end CICS program.

1. Create a **CICSConnectionSpec** object that includes a URL, which (in our case, because the CICS enterprise bean is running on the same operating system as the CICS Connector for CICS TS) should be set to `local://`.

   Note: How to set the attributes of the CCF Client Interface objects is described in more detail in "Setting the CCF interface attributes" on page 71.
2. Create a **Communication** object from the **CICSConnectionSpec** object, and execute its **connect** method.
3. Create an **ECIInteractionSpec** object that includes the name of the target program.
4. Run the **execute** method of the **Communication** object, passing the **ECIInteractionSpec**, and the **input** and **output** record beans, as arguments.
5. Retrieve the data returned by the target program from the **output** record bean.
6. Execute the **disconnect** method of the **Communication** object.

This sequence is illustrated in Figure 15 on page 71.

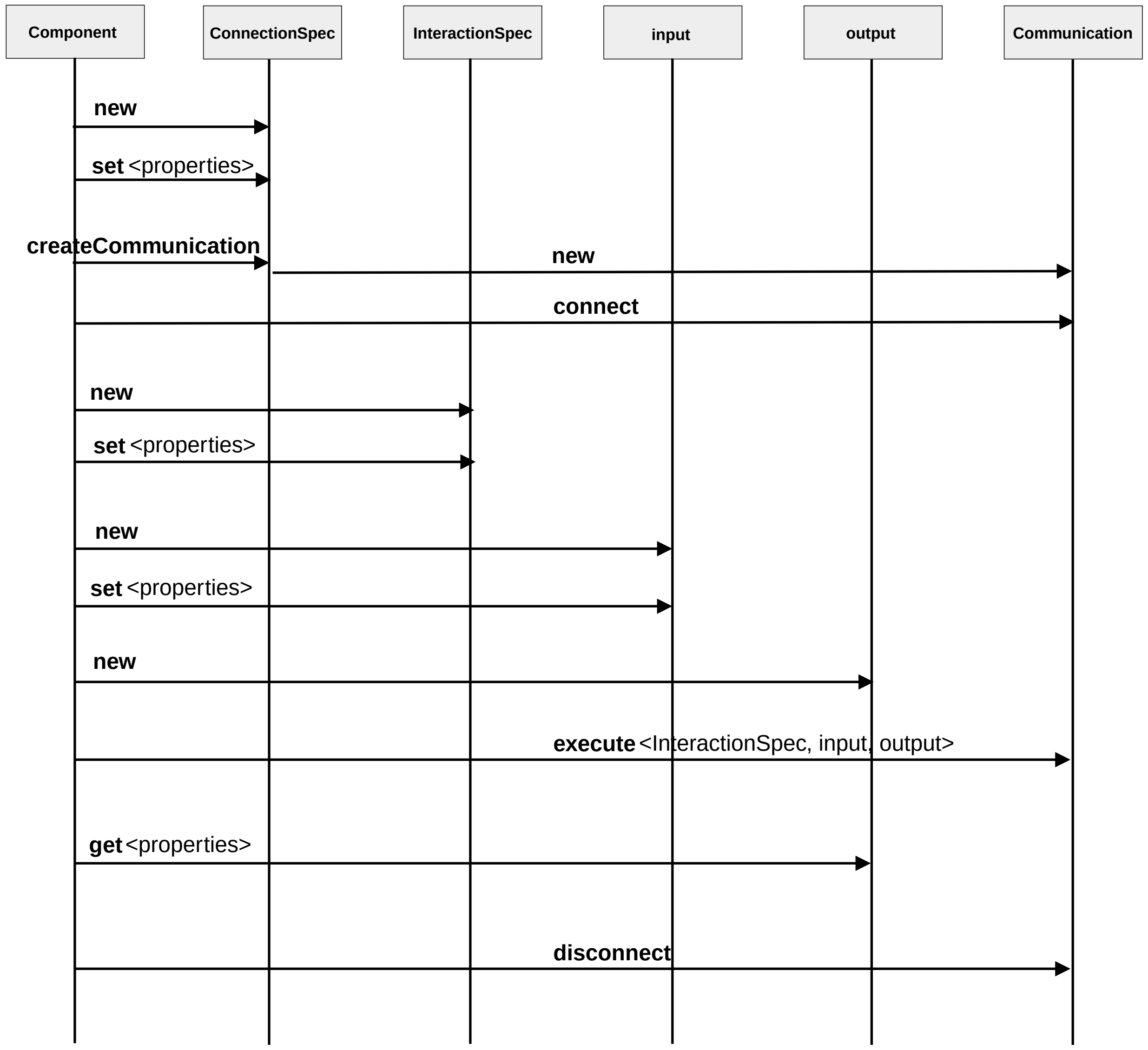


Figure 15. Using the CCF Client Interface classes

Setting the CCF interface attributes

This section describes how to set the attributes of the **CICSConnectionSpec** and **ECIInteractionSpec** classes that form part of the CICS Connector for CICS TS's Client Interface. (The **Communication**, **input**, and **output** classes are programmed in the same way as for other CICS CCF connectors.)

Note: Many of the attributes are ignored by the CICS Connector for CICS TS. This is because the CICSConnectionSpec and ECIInteractionSpec classes are also used by traditional CTG client applications, for some of which the attributes are meaningful. Ignoring the attributes, rather than introducing new types of CICSConnectionSpec and ECIInteractionSpec, means that existing CTG client applications can be ported more easily to CICS OS/390.

CICSConnectionSpec

Set the attributes of your CICSConnectionSpec object as follows:

CICSServer
The SYSID of the CICS region which owns the program to be linked to. You can set this value explicitly. However, the recommended method is to set a null value here, and to rely on the PROGRAM definition to specify the location of the server program, and whether or not dynamic routing should occur.

ClientSecurityClassName
Ignored by CICS.

connectionTimeout
Ignored by CICS.

GatewayURL
Set as follows:

auto://tcpipaddr:portno/
Supported, provided that the TCP/IP address is that of the host that CICS is running on (in which case, the local protocol is used).

local://
Supported.

http://tcpipaddr/
Not supported.

tcp://tcpipaddr:portno/
Not supported.

Normally (because the CICS enterprise bean you are creating will run on the same host as the CICS Connector for CICS TS) you would set strGatewayURL to `local://`.

logonLogoff
Ignored by CICS.

maxConnections
Ignored by CICS.

minConnections
Ignored by CICS.

reapTime
Ignored by CICS.

ServerSecurityClassName
Ignored by CICS.

terminalModel
Ignored by CICS.

unusedTimout
Ignored by CICS.

ECIInteractionSpec

Set the attributes of your ECIInteractionSpec object as follows:

CICSELUW
Set as follows:

False	The ECI call is not part of the CICS extended unit of work (UOW).
True	The ECI call is part of the CICS extended UOW.

Note: In traditional CTG applications, the CICS extended UOW encompasses a series of one or more ECI requests to a server program, each executed with SYNCONRETURN set *off*, followed by a final ECI request (to the same server program) that is executed with SYNCONRETURN set *on*. This final ECI call causes CICS to take a syncpoint on successful completion of the server program, and any changes to resources made by the server program to be committed.

When using the CICS Connector for CICS TS, all the LINK requests in the CICS extended UOW, *including the last*, are executed with SYNCONRETURN set *off*. Any changes to resources made by the server program are committed by CICS at end of task or if the application issues a syncpoint. This behavior is consistent with that of CICS DPL.

ECITimeout
Ignored by CICS.

Mode
Only sync (MODE_SEND_RECEIVE) is supported.

Password
Ignored by CICS.

ProgramName
Set to the name of the program to be linked to.

TPNTransactionName
Ignored by CICS.

TransactionName
Optionally, can be set to the name of the transaction to be used as the mirror transaction on the remote region. The default is an empty string (“”).

Userid
Ignored by CICS.

Data conversion

Java programs always use the Unicode character set. However, the communications area passed to the target program on the back-end CICS OS/390 region must be in EBCDIC. When writing your enterprise beans, you can handle data conversion using either of two methods:

Method 1

1. Convert from Unicode to ASCII in the **input** record bean.
2. Use the CICS conversion program, DFHCCNV, to convert between ASCII and EBCDIC. The connector calls DFHCCNV, before and after the program link call, provided that you have included a conversion template for the program’s communication area in the DFHCNV conversion table. Figure 16 shows an example conversion template, coded using DFHCNV macros, for the communications area of a program named `server_program`.

```
DFHCNV TYPE=ENTRY,RTYPE=PC,CLINTCP=437,RNAME=server_program,USREXIT=NO
       DFHCNV TYPE=SELECT,OPTION=DEFAULT
       DFHCNV TYPE=FIELD,OFFSET=0,DATATYP=CHARACTER,DATALEN=32767,   *
              LAST=YES
```

Figure 16. A conversion template to convert a program COMMAREA between ASCII and EBCDIC. This example converts all the data. (By coding multiple DFHCNV TYPE=FIELD macros, you can select which fields are converted, and what type of conversion is applied to each.)

For detailed information about the DFHCCNV conversion program, conversion templates, the DFHCNV conversion table, and the syntax of DFHCNV macros, see the *CICS Family: Communicating from CICS on System/390*.

Note: The connector calls DFHCCNV on the local CICS region—the region on which the enterprise bean runs—even if the target program is remote. So you must add your conversion template to the conversion table on the local region.

3. Convert from ASCII to Unicode in the **output** record bean.

Method 2

1. Convert directly from Unicode to EBCDIC in the **input** record bean.
2. Convert directly from EBCDIC to Unicode in the **output** record bean.

This method is more efficient because it misses out the intermediate conversion from Unicode to ASCII. However, the record beans must be coded to contain all the conversion logic.

Introducing VisualAge for Java Enterprise Access Builder

The VisualAge for Java Enterprise Access Builder provides a construct called a *Command*. A Command represents a single interaction with an application server. It is a Java bean with properties and an execute method—see Figure 17 on page 75. A Command is a composition—it encapsulates all the objects necessary for carrying out an interaction.

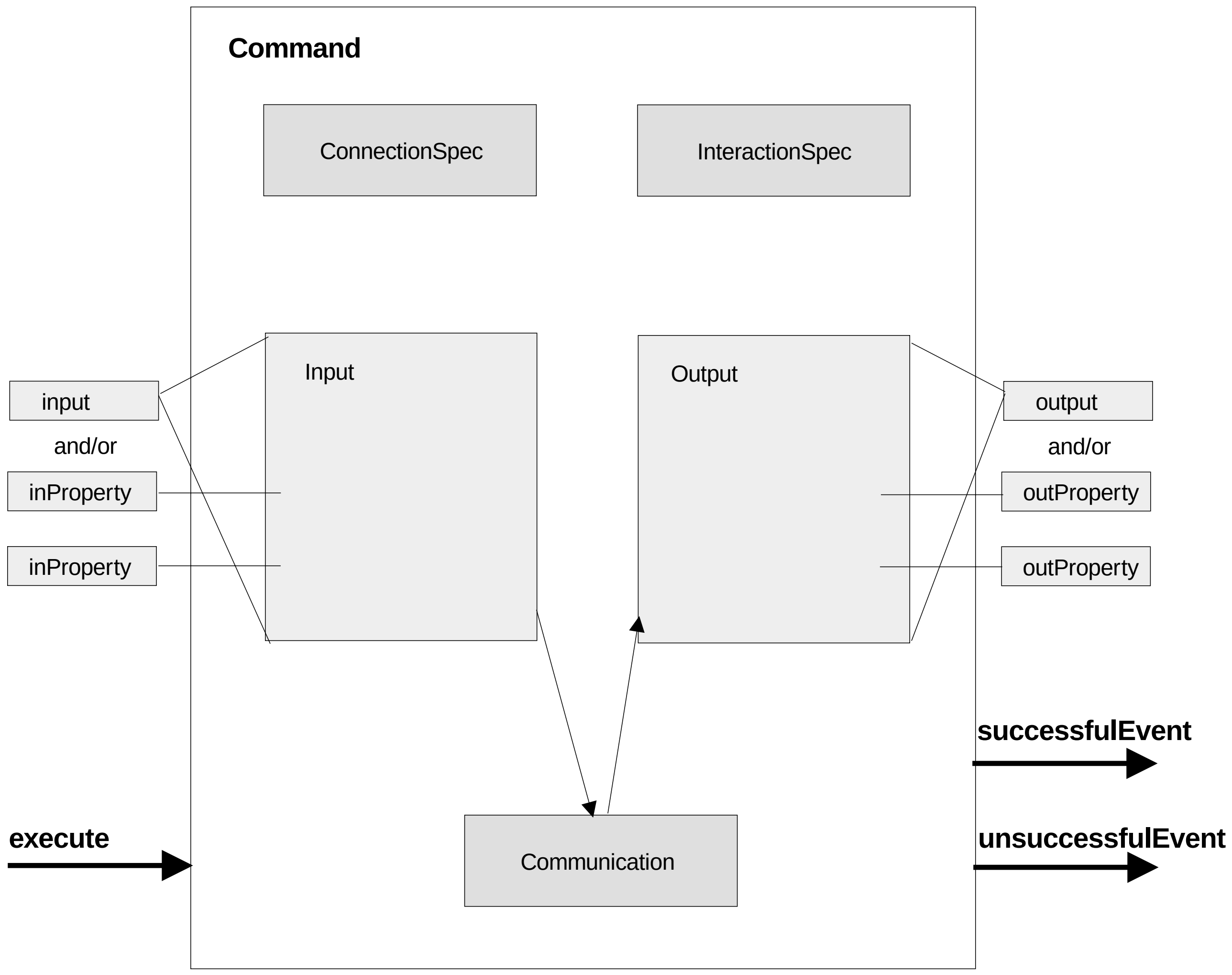


Figure 17. A Command bean produced by VisualAge for Java EAB

The Enterprise Access Builder also provides a high-level construct called a *Navigator*, which implements a sequence of interactions with an application server. A Navigator is simply a composition of Commands and Navigators. Figure 18 on page 76 shows a Navigator produced by the Enterprise Access Builder.

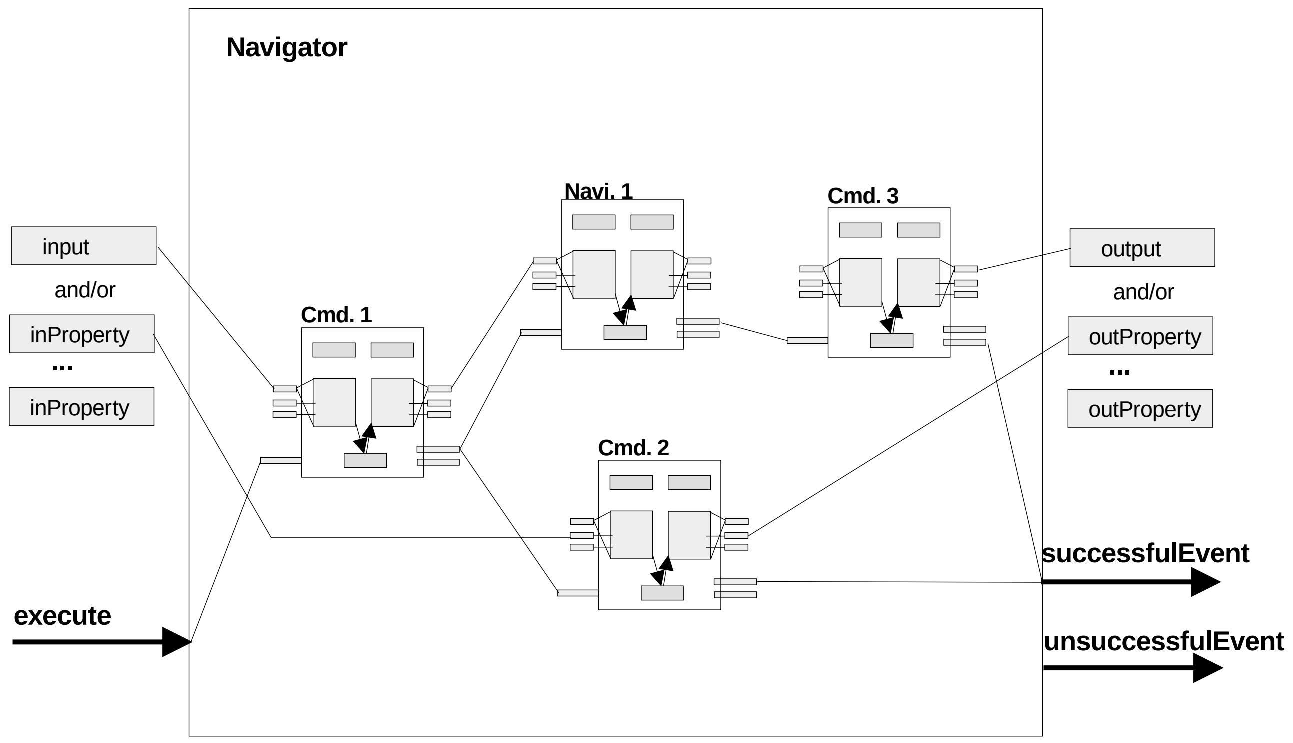


Figure 18. A Navigator produced by VisualAge for Java EAB

The EAB Command Editor tool allows you to build Commands and Navigators visually.

Using the CTG API

If you do not have VisualAge for Java Enterprise Access Builder or a similar product, your enterprise beans can use either JCICS or the CICS Transaction Gateway API to link to a CICS server program.

This section tells you how to program the CICS Connector for CICS TS's non-CCF interface, using the CTG API.

Using the CTG API to link from a CICS OS/390 Java program or enterprise bean to another, possibly remote, CICS program is possible only in CICS TS z/OS. (In previous releases of CICS, you could only use the CTG API to link to a CICS OS/390 program from a non-CICS Java client application.)

The CTG API is described in the *CICS Transaction Gateway Programming Guide*, SC34–5594–00, and in the HTML documentation shipped with the CICS Transaction Gateway. You should refer to those sources for general guidance. The rest of this section describes how to use the **ECIRequest** and **JavaGateway** classes with the CICS Connector for CICS TS.

Note: Many of the attributes of the ECIRequest and JavaGateway classes are ignored by the CICS Connector for CICS TS. This is because these classes are also used by traditional CTG client applications, for some of which the attributes are meaningful. Ignoring the attributes, rather than introducing new types of ECIRequest and JavaGateway, means that existing CTG client applications can be ported more easily to CICS OS/390.

ECIRequest

Set the attributes of your ECIRequest object as follows:

Abend_Code
Ignored by CICS.

Call_Type
Set to one of the following:

CICS_EciListSystems
The connector returns a list of CICS server regions.

Note: The list returned by the CICS Connector for CICS TS contains only one item—see the description of the SystemList attribute.

ECI_STATE_SYNC
The connector returns the status of the connection.

ECI_SYNC
The connector issues an EXEC CICS LINK SYSID() PROGRAM() call.

ECI_SYNC_TPN
The connector issues an EXEC CICS LINK SYSID() PROGRAM() call.

Cics_Rc
CICS returns a suitable return code on each request.

Commarea
Set to a communications area suitable for the server program to be linked to.

Commarea_length
Set to the length of the communications area.

ConnectionType
Returned by CICS on an ECI_STATE_SYNC call.

CicsClientStatus
Returned by CICS on an ECI_STATE_SYNC call.

CicsServerStatus
Returned by CICS on an ECI_STATE_SYNC call.

Extended_Mode
Supported by CICS.

Note: In traditional CTG applications, extended mode encompasses a series of one or more ECI requests to a server program, each executed with SYNCONRETURN set *off*, followed by a final ECI request (to the same server program) that is executed with SYNCONRETURN set *on*. This final ECI call causes CICS to take a syncpoint on successful completion of the server program, and any changes to resources made by the server program to be committed.

When using the CICS Connector for CICS TS, all the LINK requests in the CICS extended UOW, *including the last*, are executed with SYNCONRETURN set *off*. Any changes to resources made by the server program are committed by CICS at end of task or if the application issues a syncpoint. This behaviour is consistent with that of CICS DPL.

Luw_Token
In extended mode, CICS returns a unique number denoting the unit of work (UOW) token.

Message_Qualifier
Ignored by CICS.

Password
Ignored by CICS.

Program
Set to the name of the program to be linked to.

Server
Set to the SYSID of the CICS region which owns the program to be linked to. You can set this value explicitly. However, the recommended method is to set a null value here, and to rely on the PROGRAM definition to specify the location of the server program, and whether or not dynamic routing should occur.

SystemList
Contains a list of CICS server regions, each CICS denoted by its SYSID. This list is returned by the connector on a CICS_EciListSystems request.

The CICS Connector for CICS TS returns a list consisting of only one item—the "CICS default list entry", which has a SYSID of "   " (4 spaces). CICS treats this SYSID as null; setting the Server attribute of the ECIRequest object to this value causes CICS to run the server program locally.

Transid
Optionally, can be set to the name of the transaction to be used as the mirror transaction on the remote region. The default is an empty string ("").

Userid
Ignored by CICS.

JavaGateway

Set the attributes of your JavaGateway object as follows:

Port
Ignored by CICS.

Server
Set as follows:

auto://tcpipaddr/
Supported, provided that the TCP/IP address is that of the host that CICS is running on.

local://
Supported.

http://tcpipaddr/
Not supported.

tcp://tcpipaddr/
Not supported.

Normally (because the CICS enterprise bean you are creating will run on the same host as the CICS Transaction Gateway) you would set strGatewayURL to `local://`.

SetClientSecurity
Ignored by CICS.

SetServerSecurity
Ignored by CICS.

Benefits

1. The CICS Connector for CICS TS helps you to build powerful enterprise beans that make use of existing CICS programs.
2. The connector enables you to exploit VisualAge for Java Enterprise Access Builder, and similar tools, to develop enterprise beans rapidly.
3. The enterprise beans that you build:
   - Enable programmers of Java client applications, who typically have little or no knowledge of CICS, to add the power of CICS to their applications.
   - Can be used by Java client applications, applets, and servlets running on many platforms.

Requirements

The software requirements for the CICS Connector for CICS TS are:

1. The CICS Transaction Gateway for OS/390 Version 3.12. This is shipped with CICS.
2. If the CICS server program is to run on a separate back-end CICS region (the usual case), the back-end region must support distributed program link (DPL) calls.

Restrictions and recommendations for the CICS Connector for CICS TS

The following restrictions and recommendations apply to the CICS Connector for CICS TS:

1. **The CCF Interface**

   The values you can assign to some attributes of CCF objects are restricted:

   ECIInteractionSpec objects
   - **Mode** must be set to `MODE_SEND_RECEIVE`.
   - **ProgramName** must be set.

   This request maps to an `EXEC CICS LINK PROG() {SYSID()} {SYNCONRETURN}` command.

   CICSConnectionSpec objects

   GatewayURL must be either of the following:

   ```
   local://
   auto://tcpipaddr:portno/
   ```

   where `tcpipaddr` is the TCP/IP address of the host CICS.

2. **CTG API**

   The values you can assign to some attributes of CTG objects are restricted:

   ECIRequest objects

   Call_Type must be one of the following:

   ECI_STATE_SYNC

   Returns the status, which is a single value made up of the following components:

   ```
   ECI_CONNECTED_TO_SERVER
   ECI_SERVERSTATE_UP
   ECI_CLIENTSTATE_INAPPLICABLE
   ```

ECI_SYNC
Has the same semantics as ECI_SYNC_TPN. The program name must be set and, optionally, the server name (which is used as the SYSID).

This request maps to an `EXEC CICS LINK PROG() {SYSID()} {SYNCONRETURN}` command.

ECI_SYNC_TPN
Has the same semantics as ECI_SYNC. The program name must be set and, optionally, the server name (which is used as the SYSID).

This request maps to an `EXEC CICS LINK PROG() {SYSID()} {SYNCONRETURN}` command.

JavaGateway objects
Server must be either of the following:

```
local://
auto://tcpipaddr:portno/
```

where `tcpipaddr` is the TCP/IP address of the host CICS.

3. **Transaction commit point**

   If the target server program is local to the region on which the connector runs, or is remote and the request mode is extended (`ECI_EXTENDED`):
   - The transaction is committed at end-of-task or SYNCPOINT

   or
   - Can be backed out by a `SYNCPOINT ROLLBACK` request.

   If the target server program is remote and the request mode is not extended (`ECI_NO_EXTEND`):
   - The transaction at the back-end is committed on return (SYNCONRETURN).
   - The local transaction is committed at end-of-task or SYNCPOINT, *or* can be backed out by a `SYNCPOINT ROLLBACK` request.
4. The connector does not support explicit commit or backout commands.
5. Connector applications must be in JVM mode.
6. It is recommended that, for improved performance, you run the CTG classes on the trusted middleware classpath.
7. All connector requests must be made on the primary thread for the process.

Installation

The CICS Connector for CICS TS consists of two parts:
1. A platform-specific library of native functions, `libCTGJNI.so`
2. The CICS Transaction Gateway for OS/390 Java classes.

Both the `libCTGJNI.so` library and the CICS Transaction Gateway for OS/390 Java classes are automatically installed when you install CICS.

Check that the library path used by the CICS JVM includes the directory which contains the `libCTGJNI.so` library. CICS installs `libCTGJNI.so` into the `/usr/lpp/cicsts/cicsts21/lib` directory (where `cicsts21` is the value of the CICS_DIRECTORY variable used by the DFHIJVMJ job during CICS installation).

If you need to modify the library path used by the CICS JVM, edit the LIBPATH statement in your JVM profile.

Note: The LIBPATH statement in the default JVM profile, DFHJVMPR, already includes the `/usr/lpp/cicsts/cicsts21/lib` directory. If you are using the default profile no changes are necessary.

CICS installs the CICS Transaction Gateway for OS/390 Java classes in an HFS directory that is in the "trusted middleware" classpath used by the CICS JVM. (For CICS JVM purposes, the CICS Connector for CICS TS is classified as trusted middleware.).

To complete the installation, ensure that the `ccf.jar` file is in the trusted middleware classpath. The `ccf.jar` file contains utility classes required by the CTG for OS/390 Java classes. It is supplied with VisualAge for Java Enterprise Access Builder and as part of the EJB CICS sample application. If you have already installed the EJB CICS sample application you need do nothing more. If not, you can find the `ccf.jar` file in the EJB CICS sample directory on HFS. The default EJB CICS sample directory is:

```
/usr/lpp/cicsts/cicsts21/samples/bankaccount
```

where `cicsts21` is the value of the CICS_DIRECTORY variable used by the DFHIJVMJ job during CICS installation.

Note: The recommended way to add files and directories to the trusted middleware classpath is to specify them on either the TMPREFIX or TMSUFFIX initialization option in your JVM profile.

Changes to CICS externals

The CICS Connector for CICS TS introduces several changes to CICS external interfaces.

Messages

There are some new CICS messages in the range DFHCZ0150—DFHCZ0159. These are described in the *CICS Messages and Codes* manual.

Trace points

Trace data can be output from either or both parts of the CICS Connector for CICS TS:

1. From the library of CICS native functions, `libCTGJNI.so`. Use this trace information to help debug a failure within CICS.
2. From the CICS Transaction Gateway for OS/390 Java classes. Use this trace information to help debug a failure within the connector classes.

CICS trace

There are new CICS trace points in the range AP 21E7—AP 21E9. These are described in the *CICS Trace Entries* manual.

To manage the output of CICS trace information from the connector, use CICS trace control in the normal way.

CTG for OS/390 trace

How to switch on CICS Transaction Gateway for OS/390 tracing for the connector depends on which of the connector's interfaces you are using—the CCF interface or the CTG API.

Tracing the CCF interface: How to switch on CICS Transaction Gateway for OS/390 tracing when using the connector' CCF interface is described in the HTML documentation supplied with VisualAge for Java Enterprise Access Builder.

Tracing the CTG API: To switch on CICS Transaction Gateway for OS/390 tracing for the CTG API, append the following lines to the JVM properties file (the file pointed to by the JVMPROPS statement in your JVM profile).

```
gateway.T=on
gateway.T.entry=on
gateway.T.lines=on
gateway.T.exit=on
gateway.T.stack=on
gateway.T.trace=on
gateway.T.timing=on
```

Notes:

1. You can set some of these switches to `off`, as required. The meanings of the switches are as follows:

   gateway.T=on/off
   Set all debugging on or off.

   gateway.T.entry=on/off
   Set entry points on or off.

   gateway.T.lines=on/off
   Set general lines on or off.

   gateway.T.exit=on/off
   Set exit points on or off.

   gateway.T.stack=on/off
   Set stack dumps on or off.

   gateway.T.trace=on/off
   Set product-level tracing on or off.

   gateway.T.timing=on/off
   Set timing on or off.
2. In the CICS-supplied sample JVM properties files, `dfjjvmpr.props` (pointed to by the default JVM profile, DFHJVMPR) and `dfjjvmps.props`, the above lines are already present, with each switch set to `off`. If you use one of these property files, simply turn on the switches that you require.
3. The switches could be set by the Java application program. For example: `System.setProperty("gateway.T","on")`, and so on.

When trace is switched on, trace records are written to **stderr**. By default—that is, unless the CICS JVM profile specifies otherwise—the JVM directs **stderr** to **dfhjvmerr.*applid*.*time*.*taskid*.*txt***. The **dfhjvmerr** file is written to the sub-directory defined by the WORK_DIR entry in the JVM profile.

The EJB CICS sample application

The EJB CICS sample application uses the CICS Connector for CICS TS. The sample implements an enterprise bean that uses the connector to link to back-end CICS COBOL programs. The EJB CICS sample application is described in the *Java Applications in CICS* manual.

Chapter 5. CICS support for the IBM persistent reusable JVM

This chapter describes enhancements to CICS support for the IBM Java™ Virtual Machine. It covers the following topics:

- "Overview"
- "Benefits" on page 89
- "Requirements" on page 89
- "Changes to CICS externals" on page 89

Overview

CICS provides the support you need to run a Java application program in an OS/390 Java Virtual Machine (JVM) executing under the control of a CICS region. CICS support for the JVM allows you to run CICS application programs written in Java and compiled to bytecode by a Java compiler at the IBM Developer Kit for OS/390, Java 2 Technology Edition level.

The JVM provided by the IBM Developer Kit for OS/390, Java 2 Technology Edition, with a special enhancement, is known as the persistent reusable JVM (resusable JVM for short), and it includes two optimizations designed for the execution of CICS transactions. These are:

- The serial reuse of a JVM for multiple transactions, avoiding most of the initialization costs. Serial reuse entails resetting the state of the JVM between uses.
- An optimized garbage collection scheme, enabled by the clean separation of short lived application objects from the long-lived classes, objects and native state (that is, non-Java or C language), which are reset.

These optimizations are explained in the following sections.

Enabling serial reuse

Serial reuse is enabled by dividing the classes contained in the JVM into three parts:

- The OS/390 JVM code, which provides the base services in the JVM.
- The middleware, which provides services that access resources. These include the JCICS interfaces classes, JDBC, JNDI, and so on.
- The user application.

Middleware classes have privileges that are not available to the application, and which enable optimizations through the caching of state (loading of classes and native libraries, for example) to be used by multiple applications. However, middleware is also responsible to reset itself correctly at the end of a transaction and, if necessary, to reinitialize at the beginning of a new transaction in order to isolate different applications from each other. Classes are classified as middleware by virtue of their inclusion on the `ibm.jvm.trusted.middleware.class.path`. The trusted middleware class path property is built automatically by CICS from the paths specified on the CICS_DIRECTORY, JAVA_HOME, TMPREFIX, and TMSUFFIX parameters defined in the JVM profile (see the JVM profile parameters under "Changes to JVM initialization parameters" on page 91).

Not all applications are able to exploit serial reuse. If the application uses Java interfaces that modify the state of the JVM in a way that can't be safely reset (such as changing system properties, closing the standard output stream, or loading a native library), the JVM is not reused. The storage used by the JVM is recovered and a new JVM is initialized to provide a safe environment for subsequent applications. The JVM monitors the use of interfaces that prevent safe resetting, and the events that prevent reuse are logged.

Enterprise beans and CICS programs that execute on a single Java thread using interfaces defined by the Enterprise JavaBeans specification, or by the JCICS classes, are normally able to exploit serial reuse.

You can run the JVM in a mode that does not attempt serial reuse by specifying `Xresettable=NO` in the JVM profile, but this should be necessary only if the application needs to use Java facilities that modify the state of the JVM in an uncontrolled way.

Application classes are defined as follows:

- For classes that are enterprise beans, CICS manages the loading of the JAR files by means of the DJAR definitions.
- # # For classes that are executed directly through a CICS program definition, or CORBA applications:
  - # By inclusion on the `CLASSPATH` setting in the JVM profile.
  - # # # By inclusion in the `ibm.jvm.shareable.application.class.path` system property, defined in the system properties file referenced by the JVMPROPS parameter in the JVM profile

  # # # # # Defining classes in the `ibm.jvm.shareable.application.class.path` system property provides additional optimization by caching the classes in the JVM and reinitializing them. This is the recommended configuration for the best performance. If you use the `CLASSPATH`, classes are reloaded from HFS files each time the JVM is reused.
- # For utility classes:
  - # # If the utility classes are used by enterprise beans, they must be defined in the `ibm.jvm.shareable.application.class.path` system property.
  - # # # If the utility classes are used by classes that are executed directly through a CICS program definition, the utility classes can be defined in CLASSPATH or in the `ibm.jvm.shareable.application.class.path` system property.

OS/390 JVM classes have a special status that allows the objects they create to be associated with middleware or the application, depending on the kind of class that invokes their construction. The arrangement of the objects at run-time is explained in more detail below. The OS/390 JVM classes do not need to be included on a classpath.

The run-time structure of the JVM

The IBM reusable JVM manages run-time storage in several segregated heaps whose characteristics can, to some extent, be individually tuned using parameters in the JVM profile. These are:

The transient heap

This heap contains objects constructed by application classes, and any objects constructed by OS/390 JVM classes as a result of calls from application classes. It also contains any application classes, including their static data, that are loaded from the `CLASSPATH`. Segregation of this heap from the middleware heap improves performance of garbage collection.

The middleware heap

This heap contains objects constructed by middleware classes and any objects constructed by OS/390 JVM classes as a result of calls from middleware classes. It also contains static data for the middleware classes and the OS/390 JVM classes and other string constant data.

The system heaps

The main system heap contains the class definitions for all the classes *except* those application classes loaded from the `CLASSPATH` that are reloaded every time they are used. This includes the middleware class definitions and the pooled string constant data.

The other system heap contains cached application class definitions. This is called the application class system heap (ACSH).

How CICS manages the JVMs

CICS maintains a pool of JVMs, in which JVMs may be in use or available for reuse, as shown in Figure 19.

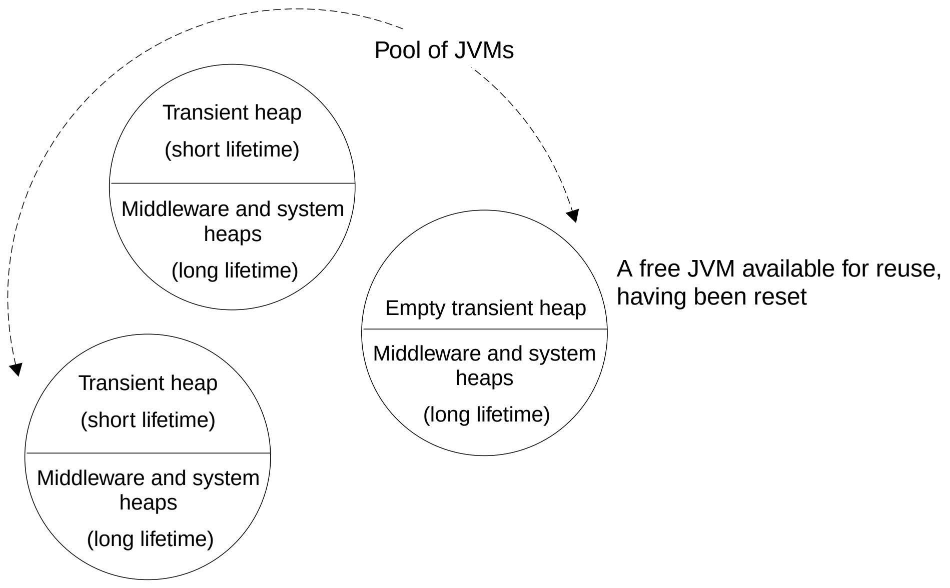


Figure 19. The reuse optimization from a JVM pool. In the figure, two of the JVMs shown are in use and contain application objects in the transient heap, which is separated from long lived objects in system and middleware heaps. The third JVM contains only long-lived objects and is available for reuse.

Managing the size of the pool

Each JVM begins execution on an MVS TCB allocated from a pool of open TCBs managed by CICS. Open TCBs can also be used for Java hotpooling, and the total number of TCBs is limited by the MAXOPENTCBS system initialization parameter. CICS controls the numbers of TCBs of each type and adjusts the number in response to the work load. You should adjust the MAXOPENTCBS setting according to the amount of storage below 16M that is available in your system. You should also restrict the number of active transactions in the system (through the MXT system initialization parameter, for example) to maintain a JVM pool that always has JVMs free to satisfy new requests. CICS reduces the number of active JVMs automatically if the work load does not require them.

Selecting the right type of JVM

JVMs are allocated to a CICS transaction that requests execution of a Java program. This is illustrated in Figure 20. The JVM characteristics (for example, heap size and class path) required by the Java program are defined by naming a JVM profile on the CICS program resource definition, which also indicates the `static main` method that is the entry point of the application program. In the case of enterprise beans, the entry point is the CICS CORBA request processor, supplied with CICS.

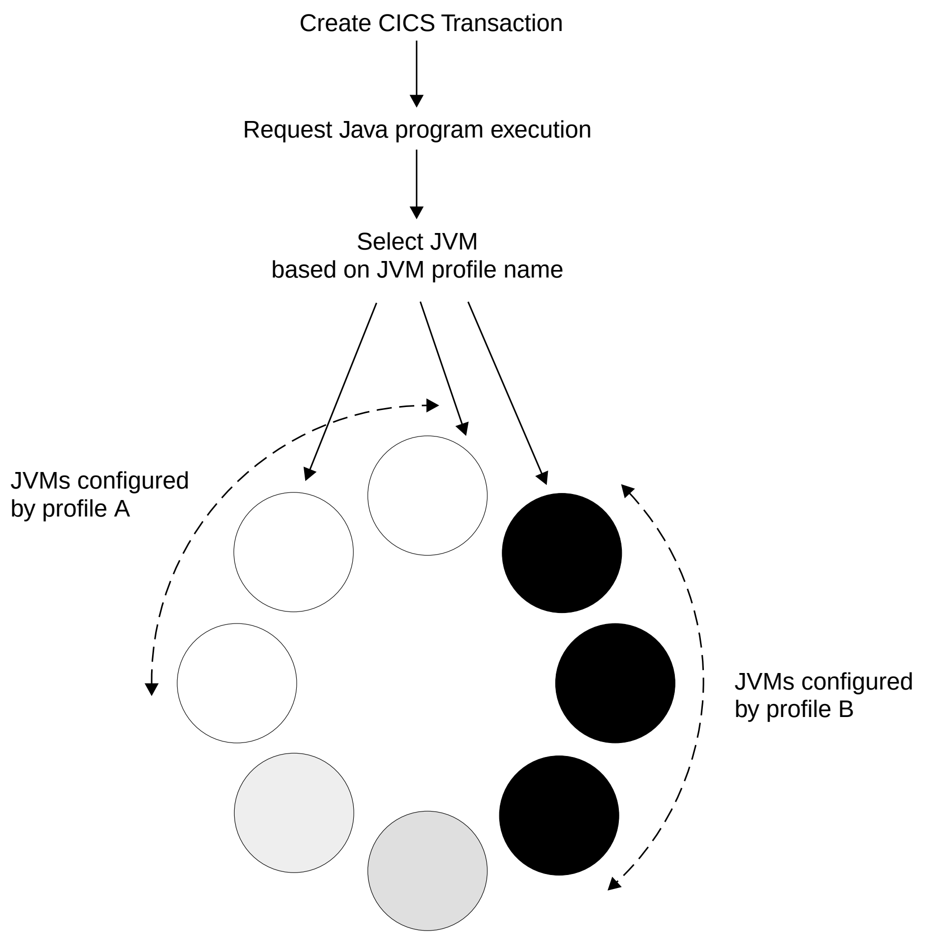

Create CICS Transaction
Request Java program execution
Select JVM based on JVM profile name
JVMs configured by profile A
JVMs configured by profile B

Figure 20. Selecting a JVM from the pool to match a request

The selection mechanism matches requests with a JVM that has the correct configuration to run them. The configuration is identified by the profile name that was used to create the JVM.

If CICS can’t find a JVM of the right type, and cannot create a new JVM because of the MAXOPENTCBS limit, CICS re-initializes an existing JVM to ensure the required characteristics. The best performance is achieved when there are many more JVMs than the number of JVM types. The fewer the number of JVM types you have, the more chance there is of an existing JVM matching the program requirement, thus avoiding the overhead of re-initialization.

Use of resource definitions for JVM selection

The specification of the JVM environment suitable for a request is found by reference to the CICS program definition. The program resource definition is determined from the request, which could originate as one of the following:

- An EJB request that matches a REQUESTMODEL, which specifies a transaction identifier
- A 3270 or START request that specifies a transaction identifier
- An EXEC CICS LINK request, or an ECI or EXCI call that names the program directly
- An entry in a program list table post-initialization (PLTPI)

The first two of these types of request are shown in Figure 21.

In each case, a program definition is used to specify the name of the Java class whose `public static main` method is to be invoked, and the characteristics of the JVM needed to run it.

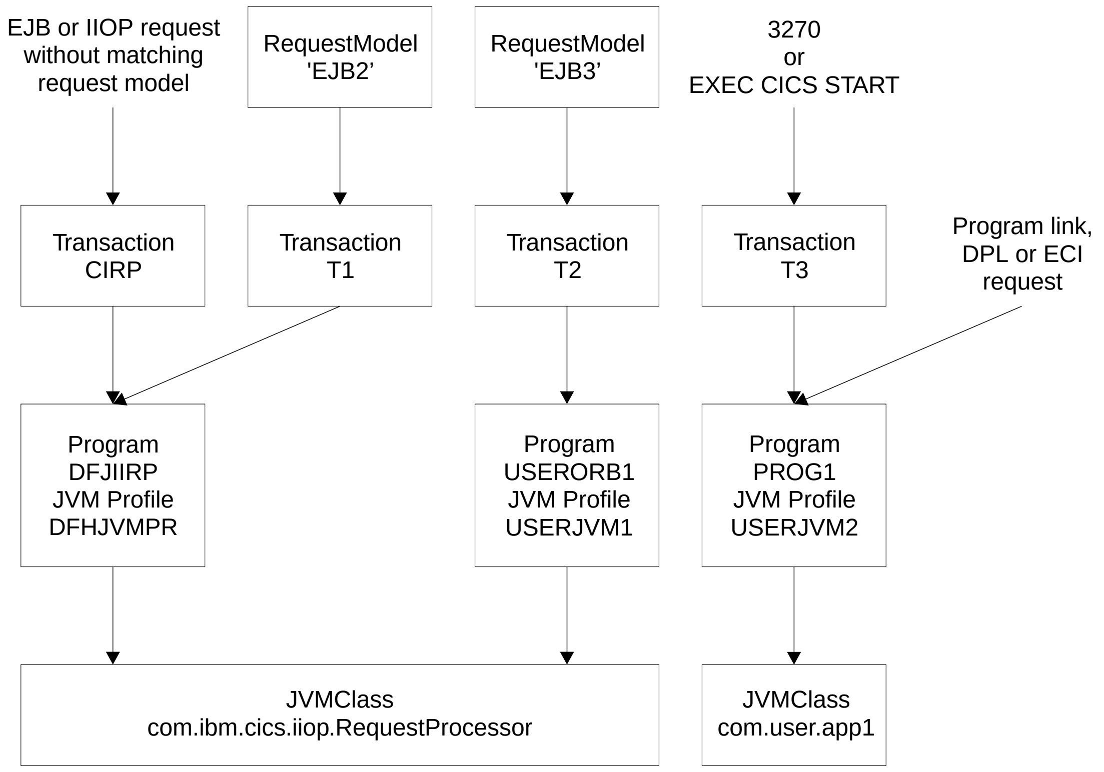

EJB or IIOP request without matching request model
RequestModel 'EJB2'
RequestModel 'EJB3'
3270 or EXEC CICS START
Transaction CIRP
Transaction T1
Transaction T2
Transaction T3
Program link, DPL or ECI request
Program DFJIIRP JVM Profile DFHJVMPR
Program USERORB1 JVM Profile USERJVM1
Program PROG1 JVM Profile USERJVM2
JVMClass com.ibm.cics.iiop.RequestProcessor
JVMClass com.user.app1

Figure 21. Determination of the JVM type

Figure 21 shows how various types of request are analyzed to determine the JVM needed for execution. This is done by the request model, transaction and program resource definitions, where the latter specifies the name of the JVM profile. The resolution of the Java class name containing the entry point for the program is also specified in the program resource definition, and is independent of the JVM type. Note that, if CICS cannot find a request model resource definition that matches an inbound request, CICS uses the CIRP transaction by default. The CICS-supplied CIRP transaction invokes DFJIIRP, which requires the default JVM profile, DFHJVMPR.

Debugging support in the CICS JVM

The JVM in CICS supports the Java Platform Debugger Architecture (JPDA), which is the standard debugging mechanism provided in the Java 2 Platform. This architecture provides a set of APIs that allow the attachment of a remote debugger to a JVM. A variety of third party debuggers are available that exploit JPDA and can be used to attach to and debug a JVM that is running an enterprise bean, CORBA object or CICS Java program. Typically the debugger provides a graphical user interface that runs on a workstation and allows you to follow the application flow, setting breakpoints and stepping through the application source code, as well as examining the values of variables.

You can find information about JPDA and JPDA-compliant applications at the web site `http://java.sun.com/products/jpda/`.

In addition to the standard JPDA debug interfaces in the JVM, CICS provides a set of interception points, which can be of value to the developers of debugging applications. These interception points (or plugins) allow additional Java programs to be inserted immediately before and after the application Java code is run. Information about the application (for example classname and method name) is made available to the plugin programs. The plugin programs can also use the JCICS API to obtain information about the application. These interception points can be used in conjunction with the standard JPDA interfaces to provide additional CICS-specific debug facilities. See *Java Applications in CICS* for more details.

Restrictions

The Java 1.1 JVM supported by CICS TS 1.3 is not supported, and any Java programs that executed under CICS TS 1.3 must be migrated to Java 2 to run under the reusable JVM. Application migration issues are contained in documents at `http://java.sun.com/j2se/1.3/compatibility.html`. Support for the reusable JVM completely replaces the JVM support provided in CICS TS 1.3, but configuration options allow the reusable JVM to be run in the same mode with small modifications to your customized initialization options. This might be necessary to execute programs that use Java interfaces that make the JVM non-resettable, such as multi-threading. It might also be necessary for compatibility reasons: for example, the old mode calls DFHJVMAT, which is not available in the new mode.

Native code migration issues are described by the `read.me` file that is in the `doc` subdirectory when you have installed the IBM persistent reusable JVM.

A stack of programs formed by a succession of EXEC CICS LINK commands, or JCICS program invocations within the same CICS task, cannot contain more than one JVM. Distributed program link (DPL) requests are not restricted in this way.

Note: This is a restriction for CORBA client applications executing in Java, which execute through the VisualAge for Java, Enterprise ToolKit for OS/390 bytecode binder in CICS TS Release 3 and can make local EXEC CICS LINK calls (directly or through an intermediate program) to a JVM. In CICS TS 2.1, the CORBA client applications execute in a JVM and cannot, therefore, make the same call.

Benefits

Support for the reusable JVM enables significant optimizations compared with CICS JVM support in CICS TS Release 3. The advantage is a reduction in the cost of initialization for a Java application program. This applies to Java programs that are invoked:
- As enterprise beans
- Through an EXEC CICS LINK command.
- Through an ECI, EXCI, or DPL call from outside CICS
- Through 3270 data streams.

Requirements

CICS JVM support requires a special enhancement to the IBM Developer Kit for OS/390, Java 2 Technology Edition. This enhancement uses new IBM technology that provides a persistent, reusable Java Virtual Machine. For information on how to obtain the IBM Developer Kit for OS/390, Java 2 Technology Edition., go to the IBM Web site at `www.s390.ibm.com/java`.

Changes to CICS externals

There are changes to a number of CICS external interfaces to enable CICS support of the persistent reusable JVM. These are:
- “Changes to resource definition” on page 90
- “Changes to JVM initialization parameters” on page 91
- “Changes to the application programming interface” on page 97
- “Changes to the system programming interface” on page 98
- “Changes to CICS-supplied transactions” on page 100
- “Changes to user-replaceable modules” on page 100
- “Changes to monitoring and statistics” on page 101
- “Changes to problem determination” on page 101

Changes to system initialization parameters

There is a change to the range of TCB modes supported by the MAXOPENTCBS system initialization parameter to support reusable JVMs. The details that follow also cover the changes introduced for Java hotpooling:

MAXOPENTCBS={5|*number*}
Specifies the maximum number of open TCBs that can exist concurrently in the CICS region. The open TCBs controlled by this parameter are as follows:

TCB mode	Used by
J8	Java programs in a JVM running in CICS key.
H8	Java program objects defined with HOTPOOL(YES), running in CICS key

When specifying the MAXOPENTCBS number, take into account TCB storage requirements: TCBs use real storage, and virtual storage below 16MB.

The default number is 5 open TCBs.

CICS manages a pool of open TCBs up to the limit set by MAXOPENTCBS. At any one time, the pool can consist of some TCBs that are allocated to tasks, and others that are free. For example, if the maximum number of open TCBs is set at 100, at a particular time the pool could consist of 50 open TCBs, not all of which are allocated. CICS attaches a new TCB when it can’t find a suitable match with a free TCB.

CICS dispatcher uses two techniques in its management of open TCBs: (1) to ensure that requests for an open TCB can be satisfied, and (2) minimize the impact on resources by reducing the number of free TCBs in the OTE pool:

- If the MAXOPENTCBS limit has been reached and there is not a free TCB of the required mode to satisfy a request, CICS looks for a free TCB of any other mode. If there is a free TCB of another mode, CICS detaches the free TCB and attaches a new TCB of the required mode. This technique is called stealing. Although costly on performance, it allows the request to be satisfied.
- To minimize the impact on storage, CICS attempts to balance the number of open TCBs against current needs by reducing the number of free TCBs. Thus, if CICS finds that there are free TCBs in the pool it gradually removes the excess number by detaching them, thereby freeing the resources used by the excess TCBs.

CICS maintains statistics of excess TCB management and TCB stealing activities.

Changes to resource definition

There are changes to the CICS PROGRAM resource definition. The JVMPROFILE attribute is added, and the DEBUG option is removed from the JVM attribute. The changes are illustrated in the following CEDA panel.

```
OVERTYPE TO MODIFY                                        CICS RELEASE = 0610
 CEDA  ALter PROGram( CBLTEST  )
  PROGram        : CBLTEST
  Group          : TEST
  DEscription  ==>
  Language     ==>                   CObol | Assembler | Le370 | C | Pli
   .
   .
 JVM ATTRIBUTES
  JVM            : No                No | Yes
  JVMClass     ==>
               ==>
               ==>
               ==>
               ==>

  JVMProfile   ==> DFHJVMPR

                                                    SYSID=HT61 APPLID=CICSHT61
```

The descriptions of the new and changed attributes are as follows:

JVM({NO|YES})
Specifies whether or not the program is to operate under the control of a Java Virtual Machine (JVM). The debugging option is removed from this parameter and is specified in the JVM profile. The supported values are:

NO
The program is a compiled program object and cannot be run in a JVM.

YES
The program is a Java bytecode program and executes in a JVM.

Note: The DEBUG option is no longer supported, and is interpreted by CICS as JVM(YES). You can continue to update definitions with the debug option for use on the earlier release by using the CSD ALTER command in compatibility mode.

JVMPROFILE(*name*)

Specifies the name of the data set member that contains the JVM profile. The name must be a member of the data set that is referenced by DFHJVM DD statement in the CICS startup JCL. The profile contains the JVM options for the execution of the program. The default name of the member is DFHJVMPR. You can customize this profile, and the member name, in a number of ways.

Changes to JVM initialization parameters

The JVM profile contains the initialization options that CICS uses to create the JVM. You define a JVM profile as a member of a PDS with a DD name of DFHJVM, which requires a DD statement in the CICS startup JCL. You can maintain JVM profile members in a PDS using MVS text editing facilities in TSO.

Note: Some of the JVM initialization options are the same as those required by the earlier version of the OS/390 JVM, some have changed in the reusable JVM, and others are completely new. For completeness, all the options are described in this section.

The profile specifies a set of values for JVM options that CICS uses to start the Java virtual machine. One of these parameters, JVMPROPS, indicates the name of an HFS file that contains the user-defined system properties.

The options defined in the JVM profile are divided into three groups: those that are required only by CICS; the standard JVM options used when starting a new JVM; and a set of nonstandard options, some of which are used for tuning, and others for debugging in a development environment.

Note: All parameter keywords and options specified in the JVM profile are case-sensitive, and must be specified exactly as shown in the following sections.

Options required by CICS

This set of JVM profile options is required only by CICS to enable CICS to start the JVM.

CICS_DIRECTORY=`/usr/lpp/cicsts/`*`cicsts21`*`/`

Specifies the path for the CICS Java `/lib` subdirectory and its JAR files. The `/usr/lpp/cicsts/` part of the path is fixed, and you are required to specify only the fourth part of the path. By default, `/lib` is installed in `/usr/lpp/cicsts/cicsts21/`, where `cicsts21` is defined by the USSDIR installation parameter when you installed CICS TS.

This parameter is required; together with the JAVA_HOME parameter, it is used by CICS to build the basic `ibm.jvm.trusted.middleware.class.path` property value.

INVOKE_DFHJVMAT={<u>NO</u>|YES}

Specifies whether or not the user replaceable module, DFHJVMAT, should be invoked before creating a new JVM. If `Xresettable=YES` is also specified, INVOKE_DFHJVMAT is ignored.

JAVA_HOME=`/usr/lpp/`*`java130s/J1.3`*`/`

Specifies the subdirectory into which IBM Developer Kit for OS/390, Java 2 Technology Edition subdirectories and JAR files are installed. By default, these are installed in `/usr/lpp/java130s/J1.3/`, where `/java130s/J1.3/` is defined when you install IBM Developer Kit for OS/390.

This parameter is required; together with the CICS_DIRECTORY parameter, it is used by CICS to build the basic `ibm.jvm.trusted.middleware.class.path` property value.

JVMPROPS=*file_name*
Specifies the name of an HFS file that contains a sequence of named values. For example, the file specified by JVMPROPS could set the `java.security.policy` property by specifying `java.security.policy=file:/usr/lpp/cicsts/cicsts21/lib/security/dfjejbpl.policy`.

Each property value is specified on a separate line and the property value is delimited by the end of the line. Property values are passed to the JVM for the construction of system properties as if specified by a `-D` option in a Java command. CICS provides two sample properties files, `dfjjvmpr.props` and `dfjjvmps.props`. These are referenced by the corresponding sample profiles, DFHJVMPR and DFHJVMPS, as follows:

- Profile DFHJVMPR specifies JVMPROPS=`dfjjvmpr.props`
- Profile DFHJVMPS specifies JVMPROPS=`dfjjvmps.props`

The first of these sample files, `dfjjvmpr`, defines the properties you need for a reusable JVM; the second is suitable only for a non-reusable JVM.

You should ensure that the file named by the JVMPROPS option is secure, with update authority restricted to system administrators. This is because the JVMPROPS file is typically used to define sensitive JVM configuration options, such as the security policy file and the trusted middleware classpath.

LIBPATH
Specifies the directory path to be searched for the following programs:

- The IBM persistent reusable JVM
- Native C dll files loaded by the JVM.

The directory path set in the CICS-supplied JVM profiles (in members DFHJVMPR and DFHJVMPS) includes the path to the native C dll files required to support JCICS.

STDERR={dfhjvmerr|*file_name***}[-generate]**
Specifies the name of the HFS file to be used for **stderr**. The file is created if it does not exist. If the file already exists, output is appended to the end of the file. On termination of the JVM, if the `stderr` file is empty, it is deleted.

The default name is `dfhjvmerr`. Note that for a fixed file name, the output from multiple JVMs is appended to the named file, in the WORK_DIR directory, and the output is interleaved.

-generate

Specifies that you want CICS to generate the output file name for a
specific JVM by appending the following qualifiers to the name supplied
on the generate option:

# *region*	The applid of the CICS region
# *time*	The current time in the form `yydddhhmmss`.
# **.txt**	A literal string suffix to indicate that the file contains readable data and should be transferred with character translation by tools such as FTP.

Note that -generate must be preceded by one blank.

Any `stderr` file that is empty at the end of the task is deleted.

STDIN={dfhjvmin|*file_name*}
Specifies the name of the HFS file to be used for **stdin**. The file is created if it does not exist.

STDOUT={dfhjvmout|*file_name*}[-generate]
Specifies the name of the HFS file to be used for output to the `stdout` file. The file is created if it does not exist. If the file already exists, output is appended to the end of the file. On termination of the JVM, if the `stdout` file is empty, it is deleted.

The default value is `dfhjvmout` in the directory specified on the WORK_DIR parameter.

The -generate option operates in the same way as for `stderr`.

TMPREFIX=*path_name*
Specifies paths to be added to the trusted middleware classpath that CICS generates automatically from the CICS_DIRECTORY and JAVA_HOME parameter. The paths specified on TMPREFIX are inserted at the *beginning* of the CICS-generated `ibm.jvm.trusted.middleware.class.path` system property.

TMSUFFIX=*path_name*
Specifies paths to be added to the trusted middleware classpath that CICS generates automatically from the CICS_DIRECTORY and JAVA_HOME parameter. The paths specified on TMSUFFIX are added to the *end* of the CICS-generated `ibm.jvm.trusted.middleware.class.path` system property.

WORK_DIR={.|*directory_name*}
Specifies the HFS directory that is used by the CICS JVM interface when creating the `stdin`, `stdout` and `stderr` files. A period (.) is defined in the CICS-supplied JVM profiles in XDFHENV (in member names DFHJVMPR and DFHJVMPS), which means you want to use the user directory of the CICS region userid. If the CICS region userid does not have an OMVS user directory, or if WORK_DIR is omitted altogether, /tmp is used as the HFS directory name.

Java standard options

The following Java standard options, which are included in the reusable JVM implementation, are passed by CICS to the JVM when the JVM is being launched:

CLASSPATH=*class_path*
Specifies the directory path to be searched by the JVM for application classes and resources.

SHOWVERSION={NO|YES}
Display version information and continue.

VERBOSE={NO|[class][,gc][,jni]}
Indicates whether or not the JVM should issue a message containing information about its activities. You can specify any or all of the three available options, or none. The options are:

NO Omit the `verbose` option so that the JVM does not report any of the information messages described by the following options.

class Specifies that the JVM is to report information about each class it loads. This equates to the standard JVM launcher option `-verbose` (or `-verbose:class`) defined in the reusable JVM Java 2 specification.

gc Specifies that the JVM should issue a message to report each garbage collection event. This equates to the standard JVM launcher option `-verbose:gc` defined in the Java 2 specification.

jni Specifies that the JVM should issue a message each time it performs one of the following Java native interface (JNI) operations:

- Dynamic link of a native method
- Registers a native method
- Loads a native library.

This equates to the standard JVM launcher option `-verbose:jni` defined in the Java 2 specification.

Coding examples:

```
verbose=class
verbose=class,jni
verbose=gc
```

Java nonstandard options

The following parameters are nonstandard options that are supported by the reusable JVM.

Xcheck={NO|[jni][,nabounds]}
Specifies whether or not you want the JVM to perform additional checks. There are two options, and you can specify either of these or both.

jni Means perform additional checks for JNI functions

nabounds
Means perform additional checks for JNI array operations.

If you omit the `jni` option, also omit the comma in front of `nabounds`.

Xdebug={NO|YES}
Specifies whether or not debugging support is to be enabled in the JVM. See also the `Xnoagent` and `Xrunjdwp` debug invocation options.

For more information, see the Java Platform Debugger Architecture (JPDA) description at `http://java.sun.com/products/jpda/doc/`.

To ensure clean termination of the debug session, you are recommended to specify Xresettable=NO.

Xinitacsh=*size*
Specifies the initial size of the application class system heap for resettable JVMs . This option is ignored for JVMs that are not reusable .

Specify *size* as a number of bytes, kilobytes, or megabytes (see “Specifying storage sizes” on page 97). The default is 128KB.

Xinitsh=*size*
Specifies the initial system heap size (class cache) for both resettable and non-resettable JVMs. There is no maximum heap size enforced by the JVM.

Specify *size* as a number of bytes, kilobytes, or megabytes (see “Specifying storage sizes” on page 97). The default size is 128KB.

Xinitth=*size*
Specifies the initial transient heap size as a number of bytes. This is ignored for non-reusable JVMs because all volatile objects are placed in the middleware heap.

Specify *size* as a number of bytes, kilobytes, or megabytes (see “Specifying storage sizes” on page 97). The default size is 500MB.

Xmaxe=*size*
Specifies the maximum heap expansion size for the middleware heap.

Specify *size* as a number of bytes, kilobytes, or megabytes (see “Specifying storage sizes” on page 97). The default size is 0 (there is no maximum heap expansion size).

Xmaxf=*percent*
Specifies the maximum free heap percentage size for the middleware heap. The default is 0.6 (60%)

Xmine=*size*
Specifies the minimum heap expansion size for the middleware heap.

Specify *size* as a number of bytes, kilobytes, or megabytes (see “Specifying storage sizes” on page 97). The default is 1MB.

Xminf=*percent*
Specifies the minimum free heap percentage size for the middleware heap. The default is 0.3 (30%)

Xms=*size*
Specifies the initial size of the middleware heap. A non-resettable JVM does not have a transient heap, therefore all instances of objects are allocated in the middleware heap.

Specify *size* as a number of bytes, kilobytes, or megabytes (see “Specifying storage sizes” on page 97). The default size is 500MB.

Xmx=*size*
Specifies the maximum total size of the middleware and transient heaps. In a non-resettable JVM this option sets the maximum Java middleware heap size only.

Specify *size* as a number of bytes, kilobytes, or megabytes (see “Specifying storage sizes” on page 97). The default is 64KB.

Xnoagent={NO|YES}
Specifies whether or not you want CICS to include the `Xnoagent` option on JVM launch command. `Xnoagent` disables the old `sun.tools.debug` agent. The Java Platform Debug Architecture (JPDA) attaches its debug agent in a different way. See the Java Debugger (JDB) description in the Java 2 SDK 1.3 specification for more information.

Xnoclassgc={NO|YES}
Specifies whether or not you want CICS to include the `Xnoclassgc` option on JVM launch command.

NO Omitting the option means that the JVM performs class garbage collection.

YES The option is specified on the JVM launch command, which means that class garbage collection is not performed.

Xoss=*size*
Specifies the maximum Java stack size for any thread.

Specify *size* as a number of bytes, kilobytes, or megabytes (see “Specifying storage sizes” on page 97).

Xresettable={YES|NO}
Specifies whether the JVM is eligible to be reused again for execution of other suitable Java programs.

YES

The JVM should be reused for execution of subsequent Java programs if possible. This avoids the overhead of starting the virtual machine for each execution of the program. The JVM profile must ensure, through the settings of the trusted and application class paths, that the application code is distinguished from trusted middleware (that is, services such as JCICS and EJB interface classes). If applications are not correctly distinguished, the state of the JVM at the termination of a program link requests is uncertain, and can cause unpredictable behavior during subsequent requests. The IBM persistent reusable JVM documentation explains the distinction between application code and middleware in more detail.

NO

The JVM is to be initialized specifically for execution of the requesting program.

Xrs={NO|YES}

Specifies whether or not you want CICS to include the `Xrs` option on the JVM launch command. `Xrs` means reduce the use of operating system signals.

Xrunhprof=(*suboption*=*string*,*suboption*=*string*...)

Enables CPU, heap, or monitor profiling. This option is typically specified as a list of sub-options of the form <suboption=*string*>, with each sub-option separated by a comma. Run the command `java -Xrunhprof:help` to obtain a list of sub-options and their default values.

Xrunjdwp=(*suboption*=*string*,*suboption*=*string*...)

Loads the JPDA reference implementation in-process debugging libraries and passes any -Xrunjdwp sub-options specified. This library resides in the target VM and uses Java virtual machine debug interface (JVMDI) and the Java native interface (JNI) to interact with it. It uses a transport and the Java Debug Wire Protocl (JDWP) to communicate with a separate debugger application. Run the command `java -Xrunjdwp:help` to obtain a list of sub-options and their default values.

See the Java Debugger (JDB) description in the Java 2 SDK 1.3 specification for more information.

Xss=*size*

Specifies the size of stack for each new Java thread.

Specify *size* as a number of bytes, kilobytes, or megabytes (see "Specifying storage sizes" on page 97).

Note: The default Java thread stack size is 1MB.

Xverify={remote|all|none}

Specifies the level of verification you want the JVM to perform on classes to be loaded. When class files are loaded (possibly over the network or from an untrusted source) into a JVM, there is no way of telling how its byte codes were generated. The options are:

remote

Verify only those classes that are loaded over the network. The default verification setting means that anything installed locally, through the CLASSPATH, is not verified. This option equates to the Java command line option `Xverify:remote`.

all Verify everything. This option equates to the Java command line option `Xverify:all`.

none
Do not perform any verification. This option equates to the Java command line option `Xverify:none`.

Specifying storage sizes: Specify sizes in multiples of 1024 bytes. Use the letter K to indicate kilobytes, and the letter M to indicate megabytes. For example, to specify 6291456 bytes as the initial size of the middleware heap, code Xms in one of the following ways:

```
Xms=6291456
Xms=6144K
Xms=6M
```

Samples

CICS provides a number of samples to help you define or alter JVM initialization options. These are:

- DFHJVMPR, a JVM profile for a re-usable JVM, referenced by the program definition that defines the CICS request processor.
- DFHJVMPS, a JVM profile that executes Java programs in the same way as in CICS TS 1.3, without JVM reuse.
- DFHJVMAT, the sample user replaceable module, which resets `stdout` to a file specific to the new task.
- DFHAPH8O, the sample user replaceable module that enables you to alter Language Environment® run-time options.
- Two JVM properties files, dfjjvmpr.props and dfjjvmps.props, each of which specifies a trusted middleware class path for the JVM.

The sample user replaceable modules are provided in the CICS SDFHSAMP library. The JVM profiles and JVM properties files are supplied in XDFHENV.

Changes to the application programming interface

There are some additional RESP2 values and a new JVM plugin mechanism.

RESP 2 values

The new RESP2 values qualify the INVREQ response to EXEC CICS LINK requests to report the conditions:

- A Java program cannot be executed because the JVMPOOL is DISABLED.
- The required JVM profile cannot be found or is invalid.
- The JVMPROPS file cannot be found or is invalid.

JVM plugin mechanism

In order to facilitate the debugging of enterprise beans and other Java programs CICS provides a JVM debug plugin mechanism that allows the CICS Java middleware to be customized by passing control at particular points to a user or vendor written debugging program. There are three Java exit points:

- A CICS EJB container debug plugin providing methods that are called immediately before and after an EJB method is invoked.
- A CICS CORBA debug plugin providing methods that are called before and after a CORBA method is invoked.
- A CICS Java Wrapper plugin providing methods that are called immediately before and after a CICS Java program is invoked

The programming interface consists of two Java interfaces. **DebugControl** (full name: `com.ibm.cics.server.debug.DebugControl`) defines the method calls that can be made to a user-supplied implementation, and **Plugin** (full name: `com.ibm.cics.server.debug.Plugin`) provides a general purpose interface for registering the plugin implementation. These interfaces are supplied in `dfjwrap.jar`, and documented in JAVADOC HTML. See *Java Applications in CICS* for more information.

Changes to the system programming interface

There are two new commands and changes to a number of existing commands. These are:

- New commands for INQUIRE and SET JVMPOOL
- Changes to the INQUIRE and SET PROGRAM command
- Changes to the CREATE command
- A change to the COLLECT STATISTICS command
- A change to the PERFORM STATISTICS command.

INQUIRE JVMPOOL

The INQUIRE JVMPOOL command returns information about the pool of JVMs in the CICS region. There can be only one pool of JVMs in a CICS region, therefore there is no name or identifier required on this command.

CICS determines the information you request from the actual JVMs that are active in the CICS region.

The following options are available on this command:

PHASEOUT(*data-area*)
returns a fullword binary field giving the number of JVMs that are marked for removal from the JVM pool. These JVMs are still allocated to a task that is currently executing, or has executed, a Java program in the JVM.

JVMs are marked for removal as a result of a CEMT (or EXEC CICS) SET JVMPOOL PHASEOUT command. Note that the phaseout number also includes JVMs whose owning tasks are being purged as a result of a SET JVMPOOL PURGE or FORCEPURGE command but the tasks could not immediately be purged.

STATUS(cvda)
returns a CVDA indicating the overall status of the JVM pool. The CVDA values are:

ENABLED
The pool is enabled for use and Java programs can execute using JVMs from the pool. This is the normal status.

DISABLED
The pool is disabled, and new requests cannot be serviced from the pool. Programs can still be executing if they were started before the JVM pool became disabled.

TOTAL(*data-area*)
returns a fullword binary field giving the number of all JVMs that are pre-initialized to process CICS program link requests. This total includes JVMs that are in the process of being terminated and removed from the region and included on the PHASEOUT count.

SET JVMPOOL

The SET JVMPOOL command allows you to change the status of the pool of JVMs in the CICS region or to terminate the JVMs in the pool. There can only be one pool of JVMs in a CICS region, therefore there is no name or identifier required on this command.

The following options are available on this command:

STATUS(cvda)
specifies whether new Java requests can be accepted and serviced by the JVM pool. The CVDA values are:

ENABLED
The pool status is set to enabled for use and Java programs can execute using JVMs from the pool.

DISABLED
The pool status is set to disabled, preventing new requests from being serviced from the pool. Programs that were started before the command was issued are allowed to execute to completion.

TERMINATE(cvda)
specifies that the JVM pool is to be terminated. The CVDA values are:

PHASEOUT
All JVMs in the pool are marked for deletion. The JVMs are actually deleted when the CICS task is terminated.

PURGE
All JVMs in the pool are terminated using the CICS SET TASK PURGE mechanism, and the JVMs are terminated.

FORCEPURGE
All tasks using JVMs in the pool are terminated by the CICS SET TASK FORCEPURGE mechanism, and the JVMs are terminated.

INQUIRE PROGRAM

There are two changes to the INQUIRE PROGRAM command.

- The JVMPROFILE option is added to return the name of the PDS member that specifies the environment variables (the profile) for the JVM needed to execute the Java program.
- The JVMDEBUG option is obsolete. If you have applications programs that use this option, CICS returns NODEBUG as the only *cvda* value.

SET PROGRAM

There are changes to the SET PROGRAM command.

- The SET STATUS(ENABLED|DISABLED) options are honored for programs that are invoked through a CICS program link request. The command has no effect on the same programs if they are invoked by Java programs through a method call.

CREATE PROGRAM

The CREATE PROGRAM command is enhanced to support the new JVMPROFILE attribute that is added to the program resource definition.

COLLECT STATISTICS

The JVMPOOL option is added to the COLLECT STATISTICS command to enable you to collect statistics about a JVMPOOL. See “Statistics” on page 101 for details of the JVMPOOL statistics available.

PERFORM STATISTICS RECORD

The JVMPOOL option is added to the PERFORM STATISTICS RECORD command to enable you to record statistics for a JVMPOOL. See “Statistics” on page 101 for details of the JVMPOOL statistics available.

Changes to the exit programming interface (XPI)

There are new options on the INQUIRE_PROGRAM call for the attributes added to the program resource definition.

Changes to CICS-supplied transactions

The changes to the SPI are also applied to the equivalent CEMT INQUIRE and SET commands:

- CEMT INQUIRE and SET JVMPOOL commands are added.
- JVMPROFILE is added to the INQUIRE PROGRAM command.
- JVMPOOL is added to CEMT PERFORM STATISTICS.

See “Changes to the system programming interface” on page 98 for details.

Changes to global user exits

The global user exit task indicator field, addressed by UEPGIND, which is part of the DFHUEPAR standard parameter list, includes a new symbolic value for the open TCB used for Java hot-pooling, This is represented in DFHUEPAR as both a two-character code and a symbolic value, as follows:

Table 8. TCB indicators in DFHUEPAR. Description

Symbolic value	2-byte code	Description
UEPTH8	H8	A Java hotpooling mode TCB

Changes to user-replaceable modules

There are changes to the program autoinstall user-replaceable module (URM), the introduction of a new URM to define Language Environment run-time options for the Language Environment enclave in which the JVM runs, and a change affecting the invocation of DFHJVMAT.

DFHSJJ8O

This new URM contains Language Environment run-time options that are used to construct the environment (the Language Environment enclave) in which the JVM runs. It defines storage allocation parameters for heap and stack. You should need to change the supplied version of the program only in exceptional circumstances. Note that the initial heap size and the heap increment sizes defined by DFHSJJ8O are for minimal values, and less than the minimum heap size for a typical JVM. The actual amount of storage allocated for the Java heap is controlled by the Xmx initialization option specified in the JVM profile.

The source for DFHSJJ8O is supplied in CICSTS21.CICS.SDFHSAMP library. For information on how you can tailor this URM to your own requirements, see the *CICS Customization Guide*.

Program autoinstall user-replaceable module

There is a change to the parameter list of the program autoinstall URM to support the new parameter on a program resource definition. The addition to the parameter list is:

PGAC_JVM_PROFID
This is an 8-byte field that specifies the name of the JVM profile to be used for the JVM in which the program is to run. The profile is a member of the XDFHENV PDS.

DFHJVMAT

Invocation of DFHJVMAT is restricted to the same events as in the previous release of CICS; that is, initialization of a JVM that is to be destroyed at the end of the program link request. DFHJVMAT is able to configure a JVM based on task-lifetime data (for example, it can set `stdout` to a task-specific output stream).

DFHJVMAT is not called in the case of a JVM that is configured for reuse by multiple tasks, because this would be inconsistent with identifying the JVM characteristics with the profile name.

Changes to monitoring and statistics

There are changes to CICS monitoring and statistics in support of the reusable JVM.

Monitoring

There are new performance class data fields added to the DFHTASK group in support of the Java 2 JVM. See “Changes to monitoring data” on page 173 for details.

Statistics

CICS provides new global statistics to provide:

- Total requests to JVM programs
- Total requests to JVM programs that map to a JVM that is a candidate for reuse
- Total requests requiring new JVMs to be initialized
- Total requests requiring new JVMs to be initialized because no suitable JVM is available
- Total number of requests resulting in a JVM being terminated because of application activity that fails isolation checks and leaves its JVM in an unusable state.

All these statistics can be obtained by an EXEC CICS COLLECT STATISTICS JVMPOOL command, and reset by the EXEC CICS SET statistics command.

The sample statistics program, DFH0STAT, is updated to report the new statistics.

Changes to problem determination

There are new messages added to aid problem determination in connection with CICS JVMs. These new messages are identified by a new component code, SJ, giving messages of the form DFHSJ*nnn*.

See the *CICS Transaction Server for z/OS Migration Guide* for details of all new and changed messages.

There is also a new AJ*xx* abend code for a transaction that cannot execute because its initial program requires a JVM but the JVMPOOL is disabled.

Chapter 6. Enhancements to CICS support of TCP/IP

This chapter describes the enhancements to CICS functions that use TCP/IP through the CICS sockets domain, namely HTTP and IIOP:

- Socket management, which enables you to specify the maximum number of sockets that the CICS sockets domain should have active at one time.
- Functions that are used by the enhanced CORBA support:
  - Outbound socket support
  - Asynchronous receive
  - Lifetime management of sockets

The CICS Web interface was first introduced in CICS/ESA® 4.1, and significantly enhanced in CICS TS 1.3, when the CICS Web support and the TCP/IP services were segregated into their separate domains (the CICS Web (WB) domain and the CICS sockets (SO) domain). CICS IIOP support was introduced in CICS TS 1.3 as part of the AP domain, and this also uses the sockets domain services. Now, the CICS IIOP services are further enhanced, and encapsulated in their own domain (the CICS IIOP (II) domain).

The TCPIPSERVICE resource definition was introduced in CICS TS 1.3 to support both HTTP and IIOP requests. Also, to enable the CICS TCP/IP support for these protocols, the TCPIP system initialization parameter was added.

Note!

The TCPIP system initialization parameter and TCPIPSERVICE resource definitions are for use only with the CICS-provided TCP/IP services for HTTP and IIOP, and are not used by the TCP/IP Socket Interface for CICS (also referred to as CICS TCP/IP for short). CICS TCP/IP is a feature of TCP/IP for MVS that allows remote users to access CICS client/server applications over TCP/IP internets. This sockets interface for CICS is supplied with OS/390 Communications Server, an integral part of OS/390, is not part of the CICS product and does not use the CICS sockets domain.

The chapter covers the following topics:

Overview

The TCP/IP support provided by the CICS sockets domain is enhanced by the addition of the following function:

Socket management

You can now specify the maximum number of sockets that the CICS sockets domain can have active at one time. The number is specified initially in the SIT, and can be changed dynamically with CEMT or the SPI.

The following functions are used by the enhanced CORBA support:

Outbound socket support

CICS can now initiate an IP connection.

Note: The term **connection** is used here in its IP sense; that is, a path between two processes, rather than in its CICS sense of a collection of sessions between two systems.

Lifetime management of sockets

When a socket is created, CICS specifies a **lifetime** for the socket. The lifetime indicates to the Sockets domain how the socket should be shared by tasks, and when it should be closed. The lifetime can be:

task One task may use the socket. The task can close the socket explicitly at any time. If it does not do so, CICS closes the socket when the task ends.

shared
Several tasks may use the socket concurrently. When a task issues a **close** request for the socket, or when the task ends, CICS notes that the task is no longer using the socket. CICS closes a shared socket when no tasks are using it.

persistent
Several tasks may use the socket serially. A task may close a socket explicitly at any time. If it does not do so, the socket remains open when the task ends, and another task can use it. Normally CICS does not close persistent sockets unless a task issues a **close**request. However, when the number of active sockets reaches its maximum number, CICS will close persistent sockets that are not in use so that it can satisfy requests for new sockets.

Asynchronous receive support

A CICS task can now initiate a receive request without being suspended in the Sockets domain while the receive request completes. When sufficient data is available to satisfy the request, the Sockets domain issues a notification — normally to the domain that issued the original request. The original receive request is now re-issued, and is satisfied immediately.

Benefits

- The **socket management** function lets you constrain the use of resources by tasks that use TCP/IP sockets.
- The following enhancements to TCP/IP support in CICS contribute to the benefits of the enhanced CORBA support provided in this release:
  - Outbound socket support
  - Lifetime management of sockets
  - Asynchronous receive support

Requirements

There are no specific hardware or software requirements for Enhancements to CICS support of TCP/IP, over and above those for CICS Transaction Server for z/OS, Version 2 Release 1 itself.

Changes to CICS externals

There are changes to a number of CICS external interfaces in support of enhanced sockets support. These are:
- “Changes to system initialization parameters” on page 105
- “Changes to the system programming interface” on page 105

Changes to system initialization parameters

There is a new system initialization parameter for enhanced sockets support:

MAXSOCKETS={number|65535}

number

Specifies the maximum number of IP sockets that can be managed by the CICS Sockets domain.

If the userid under which the CICS job is running does not have superuser authority, the maximum number of sockets that can be managed by the sockets domain is limited to the value of the MAXFILEPROC parameter in SYS1.PARMLIB member BPXPRMxx. If you specify a value greater than this in the MAXSOCKETS system initialization parameter (or by letting CICS use the default), CICS issues a message indicating the value that CICS has used.

If the userid under which the CICS job is running has superuser authority, up to 65535 sockets can be managed by the sockets domain.

Note that sockets created by Java programs running on threads that are not managed by CICS do not count towards the MAXSOCKETS limit.

Changes to the system programming interface

INQUIRE TCPIP

Two options are added to the INQUIRE TCPIP command:

MAXSOCKETS(data-value)

returns a fullword binary field containing the maximum number of IP sockets that can be managed by the CICS sockets domain.

ACTSOCKETS(data-value)

returns a fullword binary field containing the current number of active IP sockets managed by the CICS sockets domain.

SET TCPIP

Two options are added to the SET TCPIP command:

MAXSOCKETS(data-value)

specifies, as a fullword binary field, the maximum number of IP sockets that can be managed by the CICS sockets domain.

If the userid under which the CICS job is running does not have superuser authority, the maximum number of sockets that can be managed by the sockets domain is limited to the number specified in the MAXFILEPROC parameter in SYS1.PARMLIB member BPXPRMxx. If you specify a greater value, CICS sets the limit to MAXFILEPROC.

If the userid under which the CICS job is running has superuser authority, up to 65535 sockets can be managed by the sockets domain.

Note that sockets created by Java programs running on threads that are not managed by CICS do not count towards the MAXSOCKETS limit.

If you reduce the limit to less than the number of sockets currently active, CICS prevents new sockets from being created until the number of active sockets falls below the limit.

NEWMAXSOCKETS(data-value)
returns, in a fullword binary field, the new value of MAXSOCKETS.

If the userid under which the CICS job is running does not have superuser authority, CICS may set the MAXSOCKETS limit to a smaller value than requested. NEWMAXSOCKETS tells you the limit that CICS has set.

There are two new combinations of conditions and RESP2 values:

INVREQ
RESP2 value:

16 MAXSOCKETS is not in the range 1 through 65535.

NOTSUPERUSER
RESP2 value:

15 CICS was unable to set MAXSOCKETS to the value you requested, because the userid under which the CICS job is running does not have superuser authority. CICS has set the limit to the value of the MAXFILEPROC parameter specified in SYS1.PARMLIB member BPXPRMxx.

The following combination of conditions and RESP2 values can be returned on the SET TCPIP command:

NORMAL
RESP2 value:

14 TCPIP has been opened, but some TCPIPSERVICEs have not been opened because the MAXSOCKETS limit has been reached.

The following combination of conditions and RESP2 values can be returned on the SET TCPIPSERVICE command:

INVREQ
RESP2 value:

14 The TCPIPSERVICE has not been opened because the MAXSOCKETS limit has been reached.

COLLECT STATISTICS

A new option is added to the COLLECT STATISTICS command:

SOCKETS
Request global statistics for IP sockets.

Changes to CICS supplied transactions

CEMT INQUIRE TCPIP

Two options are added to the CEMT INQUIRE TCPIP command:

MAXSOCKETS(data-value)
returns the maximum number of IP sockets that can be managed by the CICS sockets domain.

ACTSOCKETS
returns the current number of active IP sockets managed by the CICS sockets domain.

CEMT SET TCPIP

Two new options are added to the CEMT SET TCPIP command:

MAXSOCKETS(data-value)
specifies the maximum number of IP sockets that can be managed by the CICS sockets domain.

If the userid under which the CICS job is running does not have superuser authority, the maximum number of sockets that can be managed by the sockets domain is limited to the number specified in the MAXFILEPROC parameter in SYS1.PARMLIB member BPXPRMxx. If you use the SET SYSTEM command to specify a value greater than that in MAXFILEPROC, CICS resets the limit to MAXFILEPROC. The message ″EXCEEDS HARDLIMIT″ is displayed when the request is made.

If the userid under which the CICS job is running has superuser authority, up to 65535 sockets can be managed by the sockets domain.

Note that sockets created by Java programs running on threads that are not managed by CICS do not count towards the limit.

If you reduce the limit to less than the number of sockets currently active, CICS prevents new sockets from being created until the number of active sockets falls below the limit.

ACTSOCKETS
displays the current number of active IP sockets managed by the CICS sockets domain.

CEMT PERFORM STATISTICS

A new option is added to the CEMT PERFORM STATISTICS command:

SOCKETS
Request global statistics for IP sockets.

Changes to monitoring and statistics

There are changes to CICS monitoring data and statistics records as part of the enhancements to CICS support of TCP/IP. See “Chapter 13. Monitoring and statistics changes” on page 169.

Changes to samples

The statistics sample program, DFH0STAT, is changed to accommodate the new statistics collected.

Changes to CICS-supplied utilities

The statistics sample program, DFHSTUP, is changed to accommodate the new statistics collected.

Chapter 7. Java Naming and Directory Interface™ (JNDI)

This chapter describes the Java Naming and Directory Interface (JNDI) support. It covers the following topics:
- "Overview of JNDI in CICS"
- "Benefits of JNDI in CICS"
- "Requirements" on page 110
- "Changes to CICS externals" on page 110

Overview of JNDI in CICS

JNDI is an application programming interface specified in the Java programming language that provides directory and naming function for Java applications. JNDI also defines a service provider's interface that allows various directory and naming service drivers to be plugged in. See Figure 22.

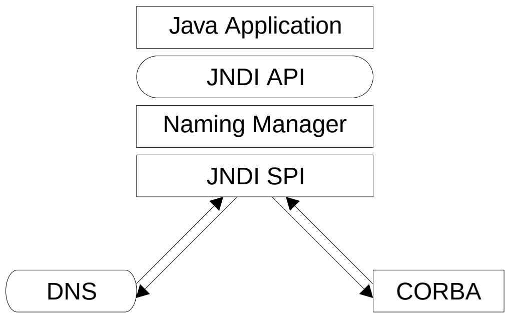


Figure 22. JNDI structure

With CICS support for Enterprise Javabeans technology, the JNDI application programming interface (API) enables enterprise beans and other Java programs running under CICS to look up a name or to locate an external enterprise bean in a nameserver, which can then be invoked. Client Java programs can also use JNDI to locate CICS enterprise beans and stateless CORBA objects.

The Java JNDI API and system programming interface (SPI) map to the CORBA object services (COS) naming directory server.

Security

JNDI does not define an external security interface for accessing naming and directory servers. Authentication or access control to the directory service are controlled by individual service providers.

Benefits of JNDI in CICS

The JNDI enables client programs to locate CICS server applications, such as enterprise beans and stateless CORBA objects, using a nameserver. Also, enterprise beans and Java programs can locate, amongst other things, an enterprise bean referenced by the local Java code so that a request can be sent to it.

Use of JNDI with a nameserver means that you do not have to transmit object references to all client locations.

Requirements

JNDI provides an interface to a COS naming directory server, such as the one provided by WebSphere Application Server Advanced Edition. This is supplied with CICS TS to ensure you have a suitable COS naming directory server for use on a Windows NT machine.

Changes to CICS externals

There are no changes to CICS externals in support of JNDI, but there are other external interfaces that you need to define and use.

Other external interfaces

Object references can be registered in a nameserver from CICS by issuing the commands PERFORM CORBASERVER PUBLISH, or PERFORM DJAR PUBLISH. This is done automatically for you as part of the deployment process for enterprise beans (see “Chapter 8. Deploying enterprise beans” on page 111), but you will need to issue the command PERFORM CORBASERVER PUBLISH for a CorbaServer used by stateless CORBA objects.

You need to define the CosNaming Server that CICS is to use. You do this on a JVM property definition in the system properties file, which is specified on the JVMPROPS parameter in the JVM profile.

The property name is `java.naming.provider.url`, and the value is specified as in the following example:

```
java.naming.provider.url=IIOP://servername.hursley.ibm.com:900
```

Note: The protocol is always IIOP.

There is an example of the `java.naming.provider.url` system property (specified in a comment line) in the sample `dfjjvmpr.props` system properties file supplied in `/usr/lpp/cicsts/cicsts21/props` in the CICS TS 2.1 HFS.

Application programming interfaces

The CICS API does not provide support for naming directory services, but the Java JNDI API contains provides directory and naming function for Java applications.

Chapter 8. Deploying enterprise beans

This chapter describes the features in CICS Transaction Server for z/OS provided to facilitate the process of deploying enterprise beans into the CICS EJB server. It covers the following topics:

- “Overview”
- “Benefits” on page 113
- “Requirements” on page 114
- “Changes to CICS externals” on page 115
- “CICS security considerations” on page 124
- “Installation and setup” on page 124

Overview

Deployment of enterprise beans has been introduced in the overview of the EJB ″story″ (see page 23 in Chapter 2. Introduction to Enterprise JavaBeans™). This section explains the process in more practical terms.

The term “deployment” used in the EJB specification describes a series of tasks that makes the enterprise beans in one or more JAR files available for use in a specific operating environment (in this case, the CICS EJB server).

Some of the steps involved might be regarded as application programming tasks (for example, code generation), and some as systems programming tasks (for example the modification of runtime options). Here, that distinction is ignored, and they are considered as steps in the deployment task.

Deployment begins with a JAR file that contains Java classes and can contain one or more (usually many) enterprise beans. The steps in the deployment process are as follows:

Step 1
Preparing an EJB 1.1 deployment descriptor in the JAR file.

Step 2
Adding generated code and CICS-specific bindings.

Step 3
Creating CICS resource definitions (either suitable for a CICS CSD, as CICS in-core definitions, or for a CICSPlex SM data repository).

Step 4
Making the enterprise beans accessible to CICS by storing the ejb-jar file in the HFS repository on z/OS.

Step 5
Publishing the CorbaServer name and enterprise bean names to an external namespace using JNDI.

At the end of this process the ejb-jar file is deployed to the CICS system, and available to be used.

CICS provides several tools to facilitate this process. These are illustrated in Figure 23 on page 112.

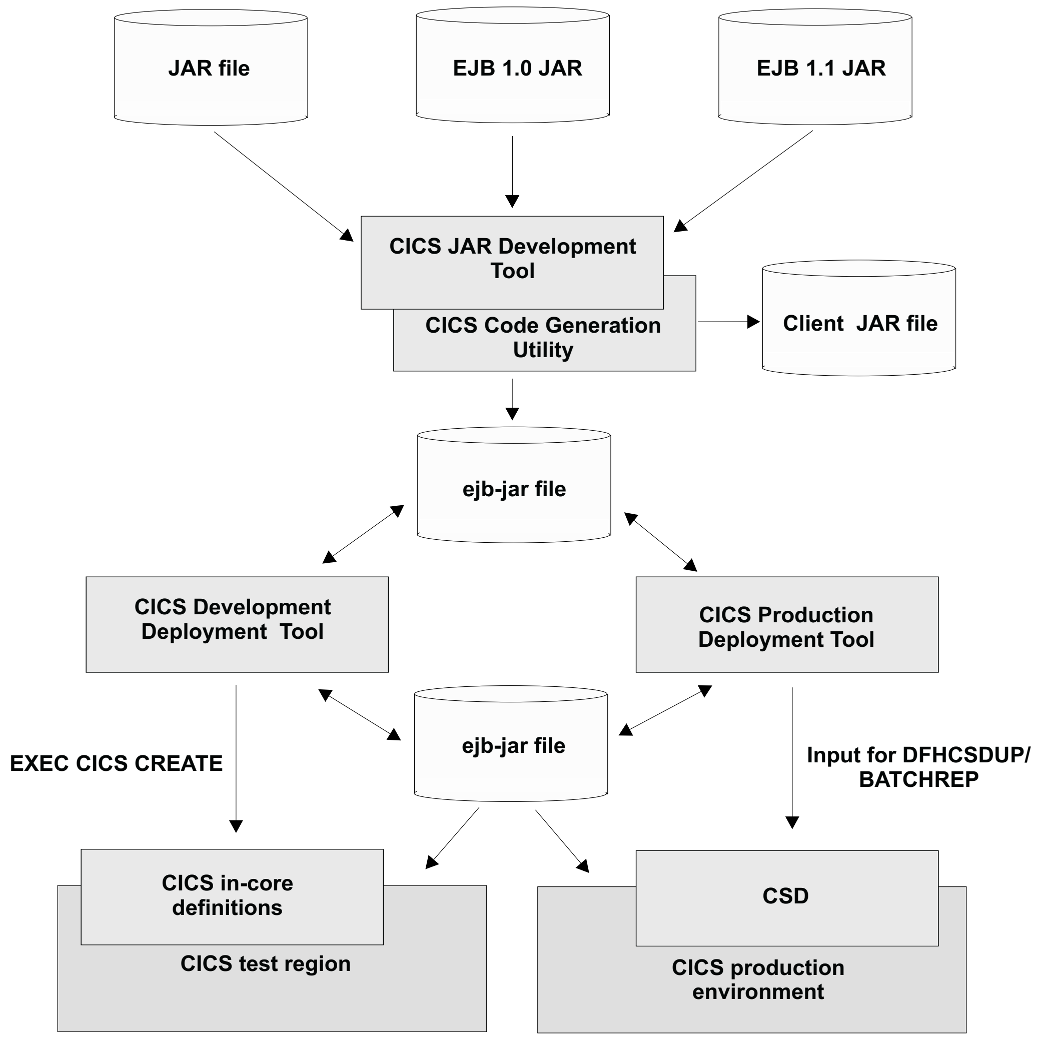

JAR file
EJB 1.0 JAR
EJB 1.1 JAR
CICS JAR Development Tool
CICS Code Generation Utility
Client JAR file
ejb-jar file
CICS Development Deployment Tool
CICS Production Deployment Tool
ejb-jar file
EXEC CICS CREATE
Input for DFHCSDUP/ BATCHREP
CICS in-core definitions
CICS test region
CSD
CICS production environment

Figure 23. The process of preparing and deploying an ejb-jar file.

The starting point is a JAR file containing one or more enterprise beans with or without a deployment descriptor. If you are using an integrated development environment, you can generate a deployment descriptor automatically. VisualAge for Java Enterprise Edition version 3.5, for example, incorporates an EJB development environment that can create ejb-jar files with EJB version 1.0 deployment descriptors.

The first part of the deployment process is the preparation of the deployment descriptor. The **CICS JAR development tool for EJB technology** is an editing tool designed to help you do this. It provides a GUI interface that enables you to create or edit the ejb-jar file's deployment descriptor together with some optional CICS specific customizations. It also converts EJB 1.0 serialized deployment descriptors into the EJB 1.1 XML equivalents required for the next stage of deployment.

The next stage is code generation. Incorporated into the CICS JAR development tool is the **CICS code generation utility for EJB technology**. This tool automatically generates the code required to tailor a generic enterprise bean into one that can run in an EJB server including the CORBA stubs and ties needed for RMI/IIOP communication. It can optionally produce a ClientJAR file, which contains only the home and remote interfaces for the enterprise beans in the JAR file, together with their RMIC stub classes. This enables client applications to install only this smaller ejb-jar file instead of an instance of the whole output ejb-jar file.

The CICS code generation utility can also be run separately in an MS-DOS command prompt window and can be used in a batch process. The result of this process is one or more EJB 1.1 JAR files.

The final stages of deployment involve the creation of CICS resource definitions, publishing bean references to an external namespace, and making the ejb-jar file accessible to CICS. There is a choice of tools for these operations.

The **CICS development deployment tool for EJB technology** provides a route that simplifies the creation of the CICS definitions in order for application programmers with a minimum of CICS expertise to test enterprise beans in a CICS test environment. The tool stores the ejb-jar file in HFS on z/OS and creates a set of generic CICS resource definitions using EXEC CICS CREATE commands. It does not update the CICS CSD. CICS resource definition information is stored in the ejb-jar file for possible later reuse by this or the alternative deployment tool. It also uses the CICS API to store a reference to each enterprise bean in an external namespace on the WebSphere Application Server using JNDI.

The alternative is the **CICS production deployment tool for EJB technology**. This tool enables you to specify in detail the necessary CICS resource definitions. This tool can create ejb-jar files ready to store on HFS. It also produces a DFHCSDUP input stream that you can use to define the necessary CICS resource definitions on the CICS CSD or a BATCHREP input stream to create definitions on the CICSPlex SM data repository. Like the CICS development deployment tool it stores CICS resource definition information in the ejb-jar file for use in a later deployment process. You can run this tool either via a GUI on your workstation, or as an off-line utility.

Enterprise bean development might involve successive uses of the development deployment tool and the production deployment tool. JAR files created using the CICS development deployment tool and run on a CICS test region can be passed to the CICS JAR development tool or the CICS production deployment tool for refinement before deployment to a CICS production environment. It is also possible to pass ejb-jar files from the CICS production deployment tool to the CICS JAR development tool or the CICS development deployment tool for modification or further testing. Both deployment tools can reuse CICS resource definitions stored in the ejb-jar file in previous sessions.

Benefits

Deployment is an essential part of the overall benefit brought to CICS application development by EJB support. This has been described already (see page 31).

The following tools provide support to help you complete the various sub-processes of deployment.

The CICS JAR development tool for EJB technology

- Accepts JAR files that have an EJB version 1.0 deployment descriptor, or an EJB version 1.1 deployment descriptor, or no deployment descriptor.
- Displays the content of the deployment descriptor, if any.
- Enables you to supply or alter deployment descriptor information.
- Enables you to save the result, and saves it as an EJB version 1.1 deployment descriptor.

The CICS code generation utility for EJB technology
Automatically generates the code required to tailor a generic enterprise bean into one that can run in an EJB server. It is incorporated into the CICS JAR development tool, but can also be run in an MS-DOS command prompt window, and can be used in a batch process.

The CICS development deployment tool for EJB technology
Makes it possible for an application programmer to deploy and test enterprise beans in CICS without having to understand the details of the CICS-specific customization required in the deployment descriptor or the CICS resource definitions that enable the enterprise beans to run in the CICS EJB server. It also stores CICS resource definition in the ejb-jar file for later reuse.

The CICS production deployment tool for EJB technology
- Enables you to add CICS resource definition information to the deployment descriptor in the ejb-jar file.
- Enables you to provide values for the environment entries in the EJB attributes in the deployment descriptor in the ejb-jar file, or to change them.
- Enables you to provide values for the Container Binding entries in the deployment descriptor in the ejb-jar file, or to change them.
- Enables you to store your ejb-jar file in HFS ready for use. (There are limitations on this action which are described in the detailed description of the tool).
- Enables you to generate resource definition statements in either or both of the following formats:
  – Ready to be processed by DFHCSDUP.
  – Ready to be processed by BATCHREP.

Requirements

Hardware

The additional hardware requirements for the CICS EJB deployment tools are as follows:

- A client PC platform capable of running Microsoft Windows NT or Microsoft Windows 2000 together with your Java development environment, such as VisualAge for Java. The recommended minimum is 266MHz, 64Mb.
- A Windows NT or Windows 2000 server platform capable of running WebSphere Application Server.

Software

The additional software required for the CICS EJB deployment tools is as follows:

- One of the following operating systems for your workstation
  – Microsoft Windows NT, version 4.0, or later
  – Microsoft Windows 2000.
- Websphere Application Server version 3.5 (This is required for the CICS Development Deployment tool and for the run time operation of the CICS EJB server. It is not required for the operation of the other CICS deployment tools.)
- IBM Developer Kit for Windows, Java 2 Technology Edition, Version 1.3, at Service Release 6 (or later), plus the EJB 1.1 standard interface classes, provided in `javax.ejb.zip` and `j2ee.jar`. (The IBM Developer Kit for Windows, Java 2 Technology Edition, Version 1.3 at Service Release 6, is supplied with CICS TS on a CD-ROM labeled CICS Tools for EJB Technology, LCD4–4355.)

- One of the follow Web browsers:
  - Netscape Communicator version 4.5, or later
  - Microsoft Internet Explorer version 5.00, or later

Changes to CICS externals

CICS support for the EJB deployment process involves only minor changes to CICS external interfaces. Instead the deployment process is facilitated by several new stand-alone workstation-based tools. These are:

- "The CICS JAR development tool for EJB technology"
- "The CICS code generation utility for EJB technology" on page 118
- "The CICS development deployment tool for EJB technology" on page 120
- "The CICS production deployment tool for EJB technology" on page 121

Other changes are described in "The final stages of deployment" on page 122, "JVM plugin mechanism" on page 97, and "Problem determination" on page 123.

The CICS JAR development tool for EJB technology

The CICS JAR development tool is a Java-based GUI editor that runs on your workstation. It is used to edit an ejb-jar file containing one or more enterprise beans in order to prepare a deployment descriptor. This tool also invokes the CICS code generation utility, which generates additional code required for deployment to the CICS EJB server.

For each enterprise bean, the ejb-jar file must include the following:

- The enterprise bean implementation
- The remote interface
- The home interface

The ejb-jar file can also contain the class files for all the classes and interfaces that the enterprise bean class, and the remote and home interfaces depend on. This includes their super-classes and super-interfaces, and the classes and interfaces used as method parameters, results, and exceptions. These class files can be made available from another JAR file (not necessarily an ejb-jar file).

Before the ejb-jar file can be deployed to the CICS EJB server it must also contain the deployment descriptor, bindings for any resources and references contained in the deployment descriptor, and all the generated code required for RMI/IIOP and EJB operation. You can use the CICS JAR development tool to create this extra information.

Preparing the deployment descriptor

The deployment descriptor is an important feature of the EJB specification. It allows the declarative specification of run time attributes for the enterprise beans avoiding the need to code the equivalent information into the beans themselves. This means that you can change these values at deploy time without having to rewrite and compile the enterprise bean classes.

In the EJB version 1.0 specification a deployment descriptor is a number of serialized Java objects (one per enterprise bean) stored in the ejb-jar file. In EJB 1.1, a deployment descriptor is a single XML document containing the deployment data for all the enterprise beans in the ejb-jar file. This tool creates only EJB 1.1

style deployment descriptors but it can read either style. You can use it to create or edit deployment descriptors, and to convert EJB 1.0 deployment descriptors to the EJB 1.1 equivalent.

A deployment descriptor contains two kinds of information:

- Structural information that describes the structure of an enterprise bean and declares the bean's external dependencies.
- Application assembly information that describes how the enterprise bean (or beans) in the ejb-jar file is composed into a larger application deployment unit.

You should not normally change structural information because doing so could alter the enterprise bean's function. You can change assembly level information without altering the enterprise bean's function, although doing so may alter the behavior of an assembled application.

Code generation

Code generation is the process of creating in the ejb-jar file the following generated code:

- RMI/IIOP stub classes based on the home and remote interfaces within the source ejb-jar file. These interfaces allow enterprise beans to be invoked remotely.
- Additional classes used by the container to honor the information stored in the deployment descriptor including the classes that implement the bean's home and remote interfaces.
- The communication stubs used on the client.

Code generation is carried out by the CICS code generation utility. There are two ways to invoke this utility

- From the CICS JAR development tool GUI using the **Generate** command on the **File** menu.
- Directly from an MS-DOS command prompt window on your workstation, or from the run option on the Windows start menu.

See "The CICS code generation utility for EJB technology" on page 118 for more information about using the CICS code generation utility as a stand-alone tool.

Using the CICS JAR development tool

Before starting: The enterprise beans in your ejb-jar may have dependencies on other Java classes. These will often be in other JAR files. In order to analyze the enterprise beans, and in particular in order to perform code generation, these other JAR files need to be available to the CICS JAR development tool and to the CICS code generation utility. Typical examples would be the `ccf.jar`, `recjava.jar` and `eablib.jar` files exported from VisualAge for Java which contain the Connector, Record Framework and Enterprise Access Builder information that people use in their beans.

These additional JARs have to be on your classpath before you invoke the tool. ″Start->Settings->Control Panel->System->Environment″ is one way to do it.

You will need the Sun standard EJB interfaces (2 jar files) and add them to your classpath before starting the tools.

- The EJB 1.1 standard interface classes are required on your classpath before running the CICS JAR development tool or the CICS code generation utility. These classes are available either in file `ejb11.jar` which is shipped in the IBM

Developer Kit (at the SDK 1.3 level) Service Release 6 or later , or in file `j2ee.jar` which is shipped in the Sun J2EE SDK.

- The EJB 1.0 standard interface classes are required on your classpath before running the CICS JAR development tool if the facility to migrate EJB 1.0 enterprise beans to EJB 1.1 is required. These classes are available in file `javax.ejb.zip` which can be obtained from `http://java.sun.com`

Note: These files are also available on the CICS deployment tools CD-ROM.

Using the tool: To invoke the CICS JAR development tool, use the menu option of the Windows NT (or Windows/2000) Start menu, Programs, IBM CICS TS 2.1 Tools, CICS Jar Development Tool for EJB Technology.

Alternatively the CICS JAR development tool may be started from an MS-DOS command prompt window using the command:

```
CICSJDT  [filename.jar [-auto]]  [-font(xx)]
```

where:

CICSJDT
: is the batch file to be run. Before using the CICSJDT command, ensure that the directory which contains `CICSJDT.BAT` file is available on your system's PATH statement. It is normally found in `C:\Program Files\IBM\CICS TS 2.1 Tools\JAR Development Tool\bin`

[*filename.jar*]
: is your input JAR file. If you do not specify it on the command line you can specify it in the GUI.

[-auto]
: indicates that the CICS JAR development tool is to load and then immediately save the specified JAR file. This can be useful for converting any EJB 1.0 deployment descriptors in the jar file to an EJB 1.1 XML deployment descriptor without user interaction. The parameter is optional, but only available if you specify an input JAR file. It is only relevant if the CICS JAR development tool is run from the command prompt.

[-font(xx)]
: indicates the font size which is to be used by the GUI instead of the default size. `xx` is the desired font size, the minimum value permitted is *10*. The parameter is optional. It is only available if the CICS JAR development tool is run from the command prompt.

The GUI main panel has options to load and save JAR files, and it displays a scrollable list of the enterprise beans in the current loaded JAR file. Each enterprise bean is identified by its bean name.The panel has buttons providing the following operations:

New - Create a new enterprise bean entry in the deployment descriptor, with default properties.

Copy - Create a new enterprise bean entry in the deployment descriptor, with properties copied from an existing entry.

Edit - Edit the properties of an enterprise bean entry in the deployment descriptor

Delete - Delete a deployment descriptor

JAR Details
: - Edit JAR file level attributes in the deployment descriptor

Bindings
: - Bring up a view that allows a JNDI binding to be specified for any enterprise bean reference or resource in the deployment descriptor.

CICS Options
: - Bring up a view that allows you to specify a CICS TRANSID against any part of the deployment descriptor that requires a REQUESTMODEL resource definition later in the deployment process.

The panel also has a message and status line, with a button that brings up a status log dialog displaying a scrollable message history. This can be saved to a file if required.

There is online help available while using the CICS JAR development tool, and for further detail about the tool, see *Java Applications in CICS*

The CICS code generation utility for EJB technology

The CICS code generation utility opens an ejb-jar file, verifies each Enterprise bean in the file, then creates the additional EJB and RMI classes for each Enterprise bean, that will be needed at runtime. It also excludes entity beans from the output ejb-jar file.

Refer to “Before starting” on page 116 for information about your classpath before proceeding.

You can invoke the CICS code generation utility from the CICS JAR development tool, or you can run it stand-alone, either in an MS-DOS command prompt window on your workstation, or from the run option on the Windows start menu.

The command to start the CICS code generation utility as a stand-alone tool is:

```
CICSCGU inputjarfile [workingdir] [outputjarfile] [clientjarfile] [options]
```

where:

CICSCGU	is the batch file to be run. Before using the CICSCGU command, ensure that the directory which contains `CICSCGU.BAT` file is available on your system’s PATH statement. It is normally found in `C:\Program Files\IBM\CICS TS 2.1 Tools\JAR Development Tool\bin`
inputjarfile	is the filename of the input ejb-jar file.
[workingdir]	is the directory name for temporary storage of extracted files and generated classes. It is not required if the -analyze option is selected.
[outputjarfile]	is the filename of the output ejb-jar file, which contains the additional code generated by this tool. It is not required if the -analyze option is selected.
[clientjarfile]	is the filename of an output ejb-jar file, which contains only the home and remote interfaces for the enterprise beans in the JAR file, together with their RMIC stub classes. It is not required if the -analyze option is selected, and is only required if you want to produce a ClientJAR file.

The ability to produce a ClientJAR file means that client applications no longer need to have the whole output ejb-jar file installed on their client machines with their client applications, just this smaller ejb-jar file.

The ClientJAR file needs no further processing by the CICS deployment tools for EJB technology. It is ready to be placed in the client according to the requirements of the client environment.

[Options]

-analyze
only verify Enterprise beans in input jar file, do not generate additional classes. If you specify -analyze, [workingdir] [outputjarfile] and [clientjarfile] are not required.

Default action is: Both analyze and generate classes.

-codegen
only generate EJB server implementation classes, do not verify Enterprise beans

Default action is: Both analyze and generate classes.

-force
ignore verification errors

Default action is: On finding errors, issues messages and does not proceed with code generation.

-J<suboption>
pass suboptions to the JVM which is used to invoke the RMIC compiler (for example: —Jmx128M, to set maximum heap size)

Default action is: No additional options are passed to the JVM.

-keep
preserve the working directory and generated files

Default action is: Delete the working directory and contents.

-nocompress
do not compress the deployed jar file

Default action is: Compress the JAR file.

-noRMIC
do not generate RMIC stubs or ties

Default action is: Generate the RMIC stub and ties.

-verbose
turn on tracing

Default action is: Suppress the verbose output.

When invoked from the CICS JAR development tool by using the **Generate** command on the **File** menu, the **inputjarfile** value is supplied by the CICS JAR development tool, you can specify the **outputjarfile**, and a **clientjarfile**, and, if you require, the **verbose** and **keep** options. The utility chooses a **workingdir** and the remaining options take their default values.

The tool issues error messages if it encounters entity beans in the ejb-jar file, but it continues to process any session beans.

The CICS development deployment tool for EJB technology

The CICS development deployment tool provides a method of deploying a 1.1 ejb-jar file into CICS directly from a development workstation without the need to define detailed CICS resource definitions. The purpose of this is to allow EJB application developers with minimal or no CICS knowledge to deploy enterprise beans into a CICS unit test environment. It is designed as an alternative to the CICS production deployment tool but is not appropriate for deploying enterprise beans into a CICS production environment.

The interface between the application programmer and this tool is a standard Web browser. The application programmer enters user ID and password information, the name and location of the 1.1 ejb-jar file to be deployed, and selects the CICS CorbaServer into which the enterprise beans are to be deployed.

The system programmer maintains control of the environment by means of the deployment configuration file (DCF). This is an XML file which specifies user and CorbaServers information as well as a number of other parameters that relate to the operation of the tool.

In operation the tool starts by initializing a Web application running on the Websphere Application Server. The Web application parses the DCF to load and validate information specified by the system programmer. Once this is successful, the application developer can connect to WebSphere to communicate with the Web application. Information entered by the application developer is validated and the enterprise beans are uploaded to the WebSphere platform.

The Web application extracts from the DCF the transaction ID associated with the selected CorbaServer. The WebSphere-specific bindings information is extracted from the ejb-jar file together with any CICS resource definition information that might be present. If there is no CICS resource definition information in the ejb-jar file, the tool creates it using default values then stores it in the ejb-jar file for use in subsequent deployment sessions. The Web application uses RMI to pass all of this information, along with the HFS name, to the CICS component of the development deployment tool; the DeployEJBJarBean session bean. The ejb-jar file is transferred to HFS storage on z/OS using FTP.

DJAR naming is handled by the CICS application, DFHADJAR, which is called using the JCICS API. The CICS applications DFHADSTR and DFHADINS carry out resource definition and REQUESTMODEL naming using the EXEC CICS CREATE API. The results of these CREATE commands are returned to DeployEJBJarBean and then passed back to the Web application where they are displayed for the application developer on the Web user interface. DeployEJBJarBean also stores a copy of the successful CICS resource definitions in the copy of the ejb-jar file on z/OS.

This deployment process is summarized in Figure 24 on page 121.

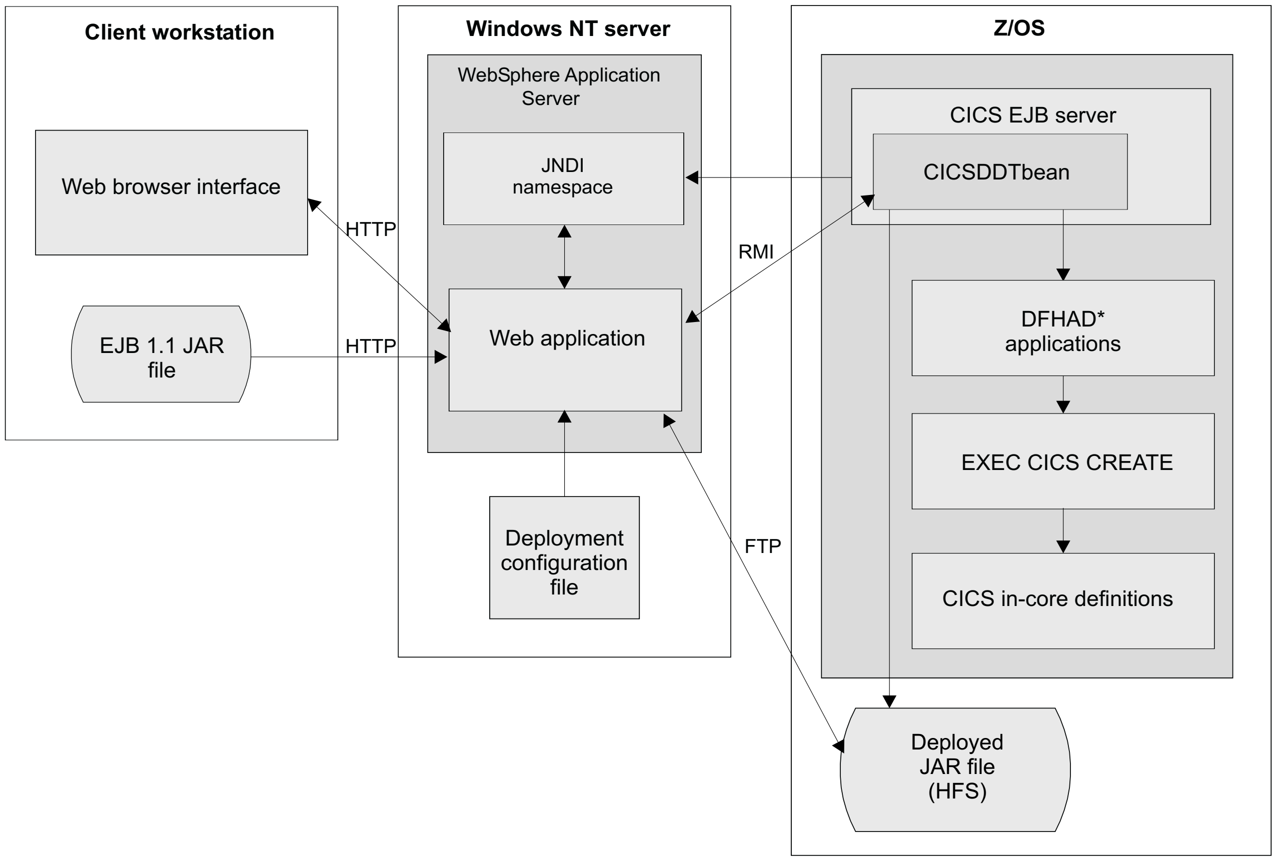


Figure 24. Deploying an EJB 1.1 JAR file from a workstation using the CICS development deployment tool.

Using the CICS development deployment tool

Operation of the CICS development deployment tool is straightforward. Starting the tool is simply a matter of invoking the Web application from your application development workstation. To do this start your Web browser and enter the URL of the Web application to open the tool's User Login page, which prompts you for your MVS user ID and password. The URL of this page can also include the name of the ejb-jar file to be deployed.

The next page is the JAR selection page. Choose whether you want to deploy or undeploy an ejb-jar file, then select the CorbaServer from a list those specified in the DCF file. Lastly, enter the path to the ejb-jar file if it was not supplied on the Web application's URL.

The final page summarizes the result of the deployment operation. The button on the page acts as a toggle, allowing the you to turn on or off the information relating to any CICS resource definitions created

The CICS production deployment tool for EJB technology

The CICS production deployment tool is the tool for system administrators to perform the tasks required to deploy Enterprise JavaBeans in an ejb-jar file into CICS regions.

The CICS production deployment tool requires as input an ejb-jar file that has an EJB 1.1 deployment descriptor.

When you want to deploy ejb-jar files as part of your testing, you will probably use the CICS development deployment tool.

The CICS production deployment tool can work with a local ejb-jar file (one stored on the workstation where the CICS production deployment tool is executing), or it can establish a connection with HFS on the OS/390 host system. The HFS data can be accessed via FTP, or as a Network drive. It is only possible to work with local and HFS files at the same time and thus move or copy files between the two storage locations when HFS is accessed as a Network drive.

The CICS production deployment tool makes the required changes to the ejb-jar file and is able to produce as output:

- A deployed ejb-jar file, containing all the modifications required before it may be used. If stored in HFS this ejb-jar file is ready to be used.
- A DFHCSDUP input stream to define in the host system CSD the CICS resource definitions relevant to the Enterprise JavaBeans in this ejb-jar file.
- A BATCHREP input stream which may be used in a CICSPlex SM environment to fulfill an equivalent function to the DFHCSDUP input stream.

Each use of the CICS production deployment tool can generate either a DFHCSDUP input stream, or a BATCHREP input stream, or both, as the user chooses.

Note: It is possible to define the required resources to CICS without the use of the CICS production deployment tool, but you are recommended to make use of the tool's ability to generate appropriate resource definition commands.

The CICS production deployment tool is provided in two forms:

- As a graphical user interface (GUI) enabled as a Java and XML application.
- As an off-line utility enabled as an MVS UNIX System Services command.

Information about installing and using the CICS production deployment tool is available in the following places:

Installation of CICS production deployment tool GUI
see *Java Applications in CICS*

Using the CICS production deployment tool GUI
see *Java Applications in CICS*

Using the CICS production deployment tool offline utility
see *Java Applications in CICS*

.

The final stages of deployment

If you use the CICS development deployment tool for EJB technology, there is nothing left to do—the development deployment tool has done it all. However, if you use the CICS production deployment tool for EJB technology, the following tasks remain to be completed.

Applying generated resource definitions

The resource definition statements that were generated by the CICS production deployment tool, either as a DFHCSDUP input stream, or as a BATCHREP input stream, (or both, if you so specified), and which were stored in HFS, now have to be applied to the systems for which they are intended. You do this using DFHCSDUP or BATCHREP. Sample JCL for each is available in *Java Applications in CICS*

Publishing names to the JNDI
Finally the names of the DJARs and the CORBASERVERs involved must be published to the JNDI.For details of this, see *Java Applications in CICS*

Ensuring that the JAR file is stored in HFS
If you did not use the CICS production deployment tool to store its output ejb-jar file in the HFS of the MVS image in which the required CICS region is run, you must now make sure that it is stored there, either by reuse of the CICS production deployment tool, or otherwise. Remember that the CICS production deployment tool can only work with local and HFS files at the same time and thus move or copy files between the two storage locations when HFS is accessed as a Network drive.

The ejb-jar file is then deployed to the CICS system, and is available to be used.

Problem determination

Error handling for the development deployment tool
Problems in the CICS development deployment tool are handled as follows:

- When the tool detects an error, it generates an exception. This causes the Web application to issue a message, which includes any associated variables.
- Error conditions and error objects are traced to CICS if the error occurs in an enterprise bean.
- If the error is in the Web application, trace is sent to a file determined in the DCF and messages are sent to the WebSphere console.

Any problems in the CICS components of the development deployment tool, cause an error to be returned to the caller indicating the failing instructions and the associated RESP and RESP2 values. These values are displayed, or translated to a more meaningful message where the error could be expected (such as the DFHADJM file not being available, or there being no available DJAR names remaining).

All error conditions are traced along with any available data in order to aid debugging.

Messages
The following new CICS messages help determine the cause of failures in enterprise bean deployment:

- DFHAD0200 to DFHAD0402

Messages intended for the application developer are displayed by the deployment tools within each tool’s user interface.

Messages intended for the system programmer are output either to the WebSphere Application Server console or to CICS, depending on the issuing program, and the location of the event that caused the message to be issued. CICS messages are sent to the CICS console and to a new transient data queue CADO.

Trace
Additional trace points are provided before and after the following external exit points (Java plugins):

- In the EJB container
- In the ORB
- In the JCICS wrapper.

The CICS development deployment tool Web application running on WebSphere Application Server outputs trace data to a collection of files identified in the DCF. This trace output is controlled by a flag in the deployment configuration file. This is not CICS system trace and this trace information is not passed to CICS.

CICS security considerations

Only users of the CICS development deployment tool require direct access to CICS. Security for this tool is handled as follows.

All deployment operations relating to the CICS test region use a single predetermined user ID. This is achieved using a sample URM that can be defined on the TCPIP service definition related to the enterprise bean's specified CORBASERVER. If the URM is not included on the TCPIP service definition, the CICS default user ID is used.

If CICS security is active, the user ID requires access to the resources used by the development deployment tool and also authority to create, discard and inquire on the necessary CICS RDO objects.

A further level of security is provided by the development deployment Web application. Application developers using this tool are required to have an entry in the DCF matching an MVS user ID. This entry specifies the CorbaServers that can be used for deployment. An MVS password are also used for validation.

The system administrator can control whether or not a particular connection should use SSL certificates. Use of SSL certificates is recommended in order to prevent user ID and password information being sent over an insecure connection.

Installation and setup

The steps involved in setting up the application development infrastructure that will allow you to develop and deploy enterprise beans to the CICS EJB server are as follows:

1. Make sure the prerequisite hardware and software are in place. See *Java Applications in CICS* for guidance on this.
2. Install the tools on their target workstations. You do not have to install the tools in any particular order.
3. Configure the tools according to the detailed setup instructions. See *Java Applications in CICS* for detailed guidance.

The CICS JAR development tool incorporating the CICS code generation utility, and the CICS production deployment tool need to be set up on your application development workstation. The CICS development deployment tool needs to be installed on a workstation running the WebSphere Application Server, which may or may not be on the same machine as the other tools.

The workstation and WebSphere components of the deployment tools are supplied on a CD as a set of InstallShield packages.

To install one or more of the deployment tools, run the **Setup** program from the CD on the target workstation. This starts the InstallShield Wizard, which leads you through the rest of the installation process.

Select **Complete** to install all three deployment tools, or **Custom** to control which tools and features to install. Follow the on-screen prompts to install the selected tools. The Wizard also creates a CICS IBM group in your workstation's Start menu from which you can start the CICS JAR development tool or the CICS production deployment tool. (The CICS development deployment tool is run from a Web browser.)

You can download any service updates to the CICS EJB deployment tools from the following IBM Web site:

```
http://www.software.ibm.com/software/ts/cics/support/
```

Setting up the development deployment Web application

Several tasks are involved in setting up the Windows NT server: The major one is installing, configuring and testing the WebSphere Application Server. You then need to install the WebSphere component of the CICS development deployment tool and configure the deployment configuration file.

The Web application is configured using the wizards available within the WebSphere Application Server console You can also control the start up and shut down of the Web application from the console

Controlling deployment using the deployment configuration file

The deployment configuration file (DCF) provides the CICS system programmer with the means to control the development deployment process. It is an XML file located in the DCF directory on the WebSphere Application Server platform. Its location is specified as an init parameter (configDefLoc) to the Web application. The DCF is designed to be edited in a standard text or XML editor. After editing the DCF, you must restart the CICS development deployment tool for any changes to take effect.

Figure 25 on page 126 contains an example of a deployment configuration file. It contains sample data showing support for 4 users, 10 CICS CorbaServers (2 of which point to the same server but have different transaction IDs associated with them), and 2 logical name bindings. Three of the four users have a CorbaServer, which is intended for his or her exclusive use. Preventing users from sharing CorbaServers is generally good practice. It makes beans more easily identifiable and avoids the need to administer a naming convention for beans.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DeploymentConfig SYSTEM "DCF.DTD">
<DeploymentConfig>
  <ConfigDefaults MaxJARSize="1000" LocalJARBase="E:/DJARS" AdminContact="John Brown, extension 123455"
 MasterTrace="ALL"  TraceLogPath="C:/temp/logfile.log" MaxActionWaitPeriod="600"/>
  <OS390Server DeployJarBase="/usr/deployedJARs" ServerName="winmvs2c.hursley.ibm.com"
UserIDIgnoreCase="true" FTPPort="21" NamingServiceURL="iiop://pushmepullyou.hursley.ibm.com:900/"
JNDIPrefix="DFHD"/>
  <CorbaServers>
    <CorbaServer CICSName="CON1" FriendlyName="CorbaServer 1" TransID="TRN1"/>
    <CorbaServer CICSName="CON1" FriendlyName="CorbaServer 2" TransID="TRN2"/>
    <CorbaServer CICSName="CON3" FriendlyName="CorbaServer 3" TransID="TRN3"/>
    <CorbaServer CICSName="CON4" FriendlyName="CorbaServer 4" TransID="TRN4"/>
    <CorbaServer CICSName="CON5" FriendlyName="CorbaServer 5" TransID="TRN5"/>
    <CorbaServer CICSName="CON6" FriendlyName="CorbaServer 6" TransID="TRN6"/>
    <CorbaServer CICSName="CON7" FriendlyName="CorbaServer 7" TransID="TRN7"/>
    <CorbaServer CICSName="CON8" FriendlyName="CorbaServer 8" TransID="CIRP"/>
    <CorbaServer CICSName="CON9" FriendlyName="CorbaServer 9" TransID="CIRP"/>
    <CorbaServer CICSName="TEST" FriendlyName="Test CorbaServer" TransID="TRNT"/>
  </CorbaServers>
  <Users>
    <User Userid="DSMITH" Trace="OFF">
        <CorbaServerRef Name="CON1"/>
    </User>
    <User Userid="MGREEN" Trace="1">
  <CorbaServerRef Name="CON2"/>
    </User>
    <User Userid="SWHITE" Trace="1,2,3">
  <CorbaServerRef Name="TEST"/>
    </User>
    <User Userid="JBROWN" Trace="1,3">
  <CorbaServerRef Name="CON1"/>
  <CorbaServerRef Name="CON3"/>
  <CorbaServerRef Name="CON5"/>
  <CorbaServerRef Name="CON7"/>
  <CorbaServerRef Name="CON9"/>
     <CorbaServerRef Name="TEST"/>
    </User>
  </Users>
  <Bindings>
    <ResourceMapping LogicalName="Resource1" Value="jdbc/Resource1"/>
    <ResourceMapping LogicalName="Resource2" Value="jdbc/Resource2"/>
  </Bindings>
</DeploymentConfig>
```

Figure 25. Deployment configuration file example

The DCF structure must match that specified in the document type definition (DTD) named DCF.DTD. This DTD defines the XML tags that must be used in the DCF, and the order in which they appear. If these rules are violated, parsing and validation fails and the deployment tool cannot start.

The DCF.DTD should reside in the same directory on the WebSphere Application Server platform as the DCF file itself. It should not be changed in any way as this would cause DCF validation to fail.

Setting up the CICS components of the development deployment tool

To set up the CICS unit test environment used to run enterprise beans you should configure a CICS EJB server, then set up the CorbaServers and transaction IDs as specified in the deployment configuration file.

CICS definitions for the tool are in 3 groups:

DFHADPD
This is the program definition group. It contains definitions to the required CICS programs and transactions. See *CICS Resource Definition Guide* for guidance on how to install program definitions.

DFHADFD
This is the file definition group. It contains the DFHADJM file. DFHADJM is a file-control-managed VSAM key-sequenced data set (KSDS) used by the tool to map CICS DJARs to the associated HFS JAR files. This group is unlocked so you can redefine the DSNAME of the file. Alternatively, this can be defined by a deployment descriptor card in the JCL in the same way as other CICS data sets. DFHADJM should not be shared across regions. See *CICS Resource Definition Guide* for guidance on how to install file definitions.

DFHADBD
This is the bean definition group. It contains the bean runtime requirements. It is not locked to allow you to redefine the definitions. It contains:

DFHADTCP
This is the TCP/IP service used to receive requests for the bean component of the tool. You can alter the port number as required, but it must match the port number of CorbaServer DFHD. See *CICS Resource Definition Guide* for more guidance.

DFHD This is the CorbaServer used to hold the DJAR containing the bean component. It is advisable to alter the host name of the MVS system on which the CICS is running. Also, ensure that the port number matches that defined in TCP/IP service DFHADTCP. You can alter the JNDI prefix, but if you do this, it must match the JNDI prefix stated in the JNDIPrefix element of the DCF. You can also change the shelf. See *CICS Resource Definition Guide* for more guidance.

DFHADJAR
This is the DJAR definition of the jar containing the bean component. The HFS file defaults to match the default installation of the UNIX System Services component of CICS. If the `dfjadmvs.jar` file is not located in the default location, adjust the HFS file to match. See *CICS Resource Definition Guide* for guidance on how to install DJAR definitions.

DFHADRM
This is the request model definition for the CICSDDT bean. One request model is used for all operations. You should not need to alter this unless you alter the CorbaServer used to hold the DJAR containing the bean component. See *CICS Resource Definition Guide* for guidance on how to install request model definitions.

These groups are not included in the default CICS startup group list, DFHLIST, so they are not automatically installed on start up. Be aware that as the definitions are in a DFH group, any DFHCSDUP upgrades will overwrite any updated information with the supplied defaults. To avoid this possibility copy DFHADBD and DFHADFD to another group before you edit them. Once the definitions have been tailored, edit your start up JCL to install this group list along with any existing lists. .

Once the tool has been installed, the bean needs to be published to the JNDI name server. Before you do this make sure that you have specified the nameserver on the `NamingServiceURL` field in the DCF. You need to publish the bean only once during the lifetime of the JNDI using the command `CECI PERFORM DJAR(DFHADJAR) PUBLISH`. This is not done automatically as part of the CICS start up procedure. See *CICS System Programming Reference* for more guidance on using this command.

Setting up user IDs for the development deployment tool

All deployment operations relating to the CICS test region use a single predetermined user ID. This is achieved using a user replaceable module (URM) that can be defined on the TCP/IP service definition related to the enterprise bean's specified CorbaServer. See *CICS Resource Definition Guide* for more guidance. If the URM is not included on the TCP/IP service definition, the CICS default user ID is used.

If CICS security is active, the user ID requires resource level access to the resources in DFHADPD, DFHADFD and DFHADBD, which are used by the development deployment tool, and also command level access to create, discard and inquire on the necessary CICS RDO objects.

A further level of security is provided by the development deployment Web application. Application developers using this tool are required to have an entry in the DCF matching an MVS user ID. This entry specifies the CorbaServers that can be used for deployment.

The system administrator needs to do the following:

- Set up a RACF® group and add a GID (group ID) to give full access to UNIX System Services.
- Set up the CICS user ID and add it to the RACF group so that the required HFS directories can be created. See *CICS Transaction Server for z/OS Installation Guide* for guidance on this.
- Set up MVS user IDs for all users who have access to the CICS region for the purpose of deploying and running enterprise beans and add them to the RACF group.
- Ensure that FTP services are available on MVS UNIX System Services. See the *OS/390 IBM CS IP User's Guide*, GC31–8514 for information on MVS FTP services.

The system administrator can control whether or not a particular connection should use SSL certificates. Use of SSL certificates is recommended in order to prevent user ID and password information being sent over an insecure connection.

Chapter 9. CICSPlex SM management of enterprise beans

This chapter describes CICSPlex SM management of enterprise beans. It covers the following topics:
- “Overview”
- “Changes to CICSPlex SM externals”
- “CICSPlex SM workload management of enterprise beans” on page 141

Overview

Enterprise beans can be managed by CICSPlex SM providing:

- Support for resource definition with:
  - New BAS resource definition types for CorbaServers and CICS-deployed JAR files.
  - # # Modified definition types for programs, transactions, request models, and TC/IP services.
- Support for CICS resources with:
  - New resources for CorbaServers, CICS-deployed JAR files, and beans.
  - # # Modified resources for programs, request models, TCP/IP services, and units of work.
- Workload management of enterprise beans using both workload balancing and workload separation techniques.
- Support for including the following EJB resources in real-time analysis:
  - EJCOSE
  - EJDJAR
- Changes to the EUI by the provision of:
  - New menus and views to support the definition and operation of CorbaServers, CICS-deployed JAR files, and beans.
  - Modified views to support the changes to the definition and operation of transactions, request models, TCP/IP services, and resource descriptions.
- Changes to the API by the provision of:
  - New resource tables to support the definition and operation of CorbaServers, CICS-deployed JAR files, and beans.
  - Modified resource tables to support the changes to the definition and operation of transactions, request models, and TCP/IP services.
- Changes to the Web User Interface by the provision of:
  - Support for the operation of CorbaServers, CICS-deployed JAR files, and beans, including new sample views in the supplied Starter Set.
  - Support for the modified transactions, request models, TCP/IP services, and resource descriptions, including modified sample views.

Note: There is no support for enterprise bean monitoring or statistics using CICSPlex SM.

Changes to CICSPlex SM externals

The changes to CICSPlex SM externals for enterprise beans comprises:

- Two new CICSPlex SM Business Application Services (BAS) resource definition types:
  - CorbaServer resource definition using the new EJCODEF resource
  - CICS-deployed JAR file resource definition using the EJDJDEF resource

  To use these resources, see:
  - For the EUI,“New BAS resource definition views”.
  - For the API, “New BAS resource definition tables” on page 140.
- Modifications to the CICSPlex SM BAS resource definition types:

  #
  - Program definition (PROGDEF)
  - Request model definition (RQMDEF)
  - TCP/IP service definition (TCPDEF)
  - Transaction definition (TRANDEF)
  - Resource description definition (RESDESC)

  To use these resources, see:
  - For the EUI,“Modified BAS resource definition views” on page 132.
  - For the API, “Modified BAS resource definition tables” on page 141.
- Four new CICSPlex SM operations resource types:
  - CorbaServer resources using the EJCOSE resource
  - CICS-deployed JAR file resources using the EJDJAR resource
  - Enterprise bean resources using the EJCOBEAN and EJDJBEAN resources

  To use these resources, see:
  - For the EUI, “New operations views” on page 135.
  - For the Web User Interface, “Web User Interface changes” on page 140.
  - For the API, “New operations resource tables” on page 141.
- Five modified CICSPlex SM operations resource types:

  #
  - Program resources (PROGRAM)
  - Request model resources (RQMODEL)
  - TCP/IP services resources (TCPIPS)
  - Unit of work resources (UOW)
  - Unit of work link resources (UOWLINK)

  To use these resources, see:
  - For the EUI,“Modified operations views” on page 136.
  - For the Web User Interface, “Web User Interface changes” on page 140.
  - For the API, “Modified operations resource tables” on page 141.

EUI changes

The EUI changes are covered in two sections::

- “New BAS resource definition views”
- “Modified BAS resource definition views” on page 132

New BAS resource definition views

The two new BAS resources are:

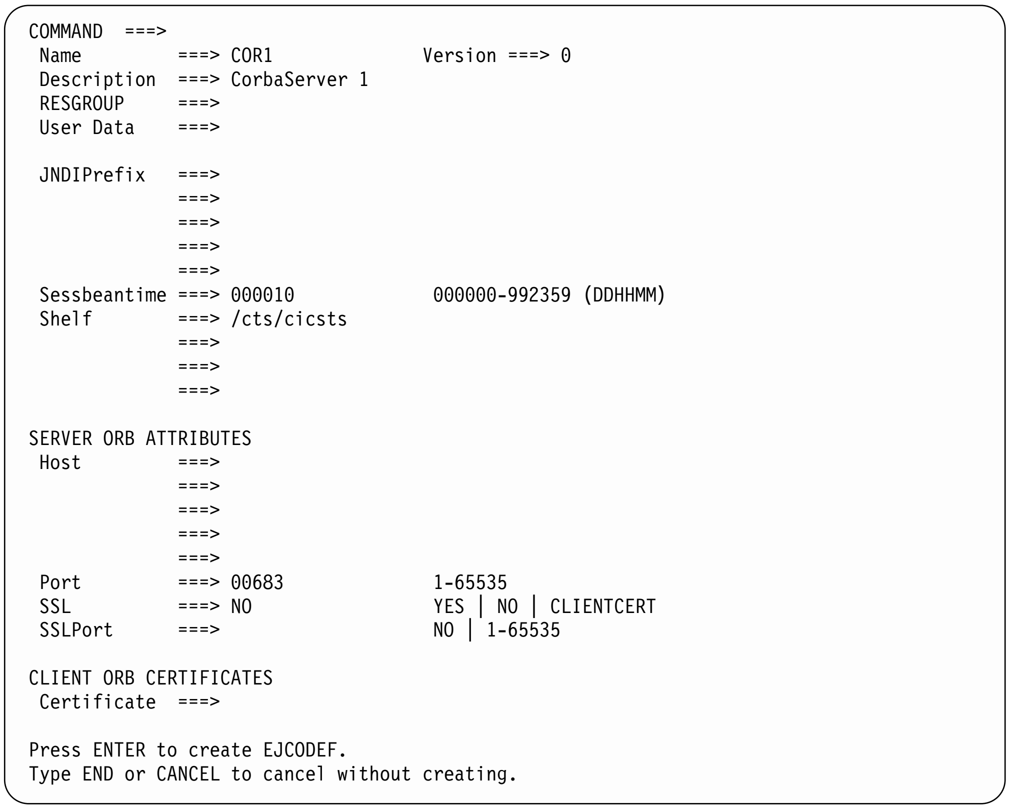
EJCODEF (CorbaServer definition)

EJCODEF definitions describe the physical and operational characteristics of CorbaServers. To display information about existing CorbaServer definitions, either issue the command:

```
EJCODEF [resdef]
```

or select EJCODEF from the ADMRES menu.

Figure 26 shows the format of the panel produced when you use the create primary (CREate) or line (CRE) action command from the EJCODEF view.



```
COMMAND  ===>
 Name         ===> COR1              Version ===> 0
 Description  ===> CorbaServer 1
 RESGROUP     ===>
 User Data    ===>

 JNDIPrefix   ===>
              ===>
              ===>
              ===>
              ===>
 Sessbeantime ===> 000010             000000-992359 (DDHHMM)
 Shelf        ===> /cts/cicsts
              ===>
              ===>
              ===>

SERVER ORB ATTRIBUTES
 Host         ===>
              ===>
              ===>
              ===>
              ===>
 Port         ===> 00683              1-65535
 SSL          ===> NO                 YES | NO | CLIENTCERT
 SSLPort      ===>                    NO | 1-65535

CLIENT ORB CERTIFICATES
 Certificate  ===>

Press ENTER to create EJCODEF.
Type END or CANCEL to cancel without creating.
```

Figure 26. Creating a CorbaServer definition

After installation of a EJCODEF resource definition, you can enquire about the resultant object using:

- The CICSPlex SM EJCOSE command.
- The CICS CEMT INQUIRE CORBASERVER command
- The EXEC CICS INQUIRE CORBASERVER command.

For descriptions of the CorbaServer resource definition attributes, see the *CICSPlex System Manager Managing Business Applications*.

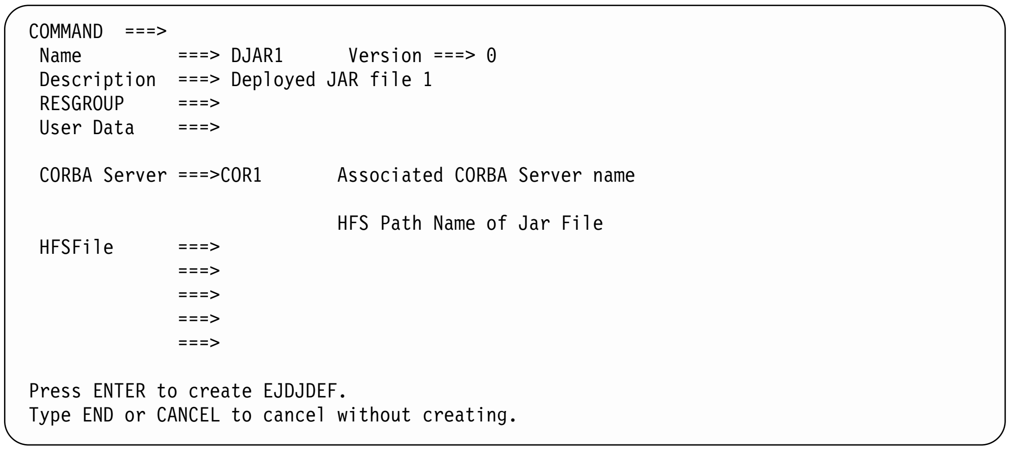
EJDJDEF (CICS-deployed JAR file definition)

EJDJDEF definitions describe the physical and operational characteristics of CICS-deployed JAR files. To display information about existing CorbaServer definitions, either issue the command:

```
EJDJDEF [resdef]
```

or select EJDJDEF from the ADMRES menu.

Figure 27 shows the format of the panel produced when you use the create primary (CREate) or line (CRE) action command from the EJDJDEF view



```
COMMAND  ===>
 Name         ===> DJAR1      Version ===> 0
 Description  ===> Deployed JAR file 1
 RESGROUP     ===>
 User Data    ===>

 CORBA Server ===>COR1       Associated CORBA Server name

                             HFS Path Name of Jar File
 HFSFile      ===>
              ===>
              ===>
              ===>
              ===>

Press ENTER to create EJDJDEF.
Type END or CANCEL to cancel without creating.
```

Figure 27. Creating a CICS-deployed JAR file definition

After installation of a EJDJDEF resource definition, you can enquire about the resultant object using:

- The CICSPlex SM EJDJAR command.
- The CICS CEMT INQUIRE DJAR command.
- The EXEC CICS INQUIRE DJAR command.

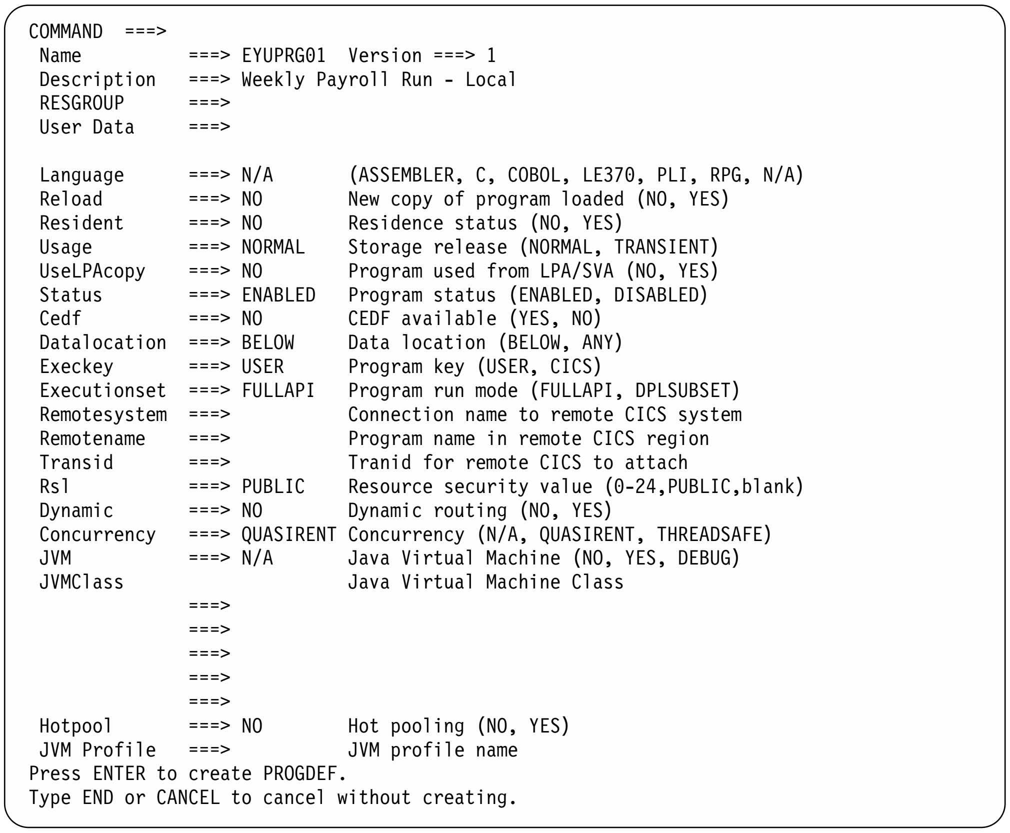
For descriptions of the CICS-deployed JAR file resource definition attributes, see the *CICSPlex System Manager Managing Business Applications*.

Modified BAS resource definition views

The BAS views modified to support enterprise beans are:

PROGDEF (program definition)

The PROGDEF panel has a new attribute: JVM Profile.
#

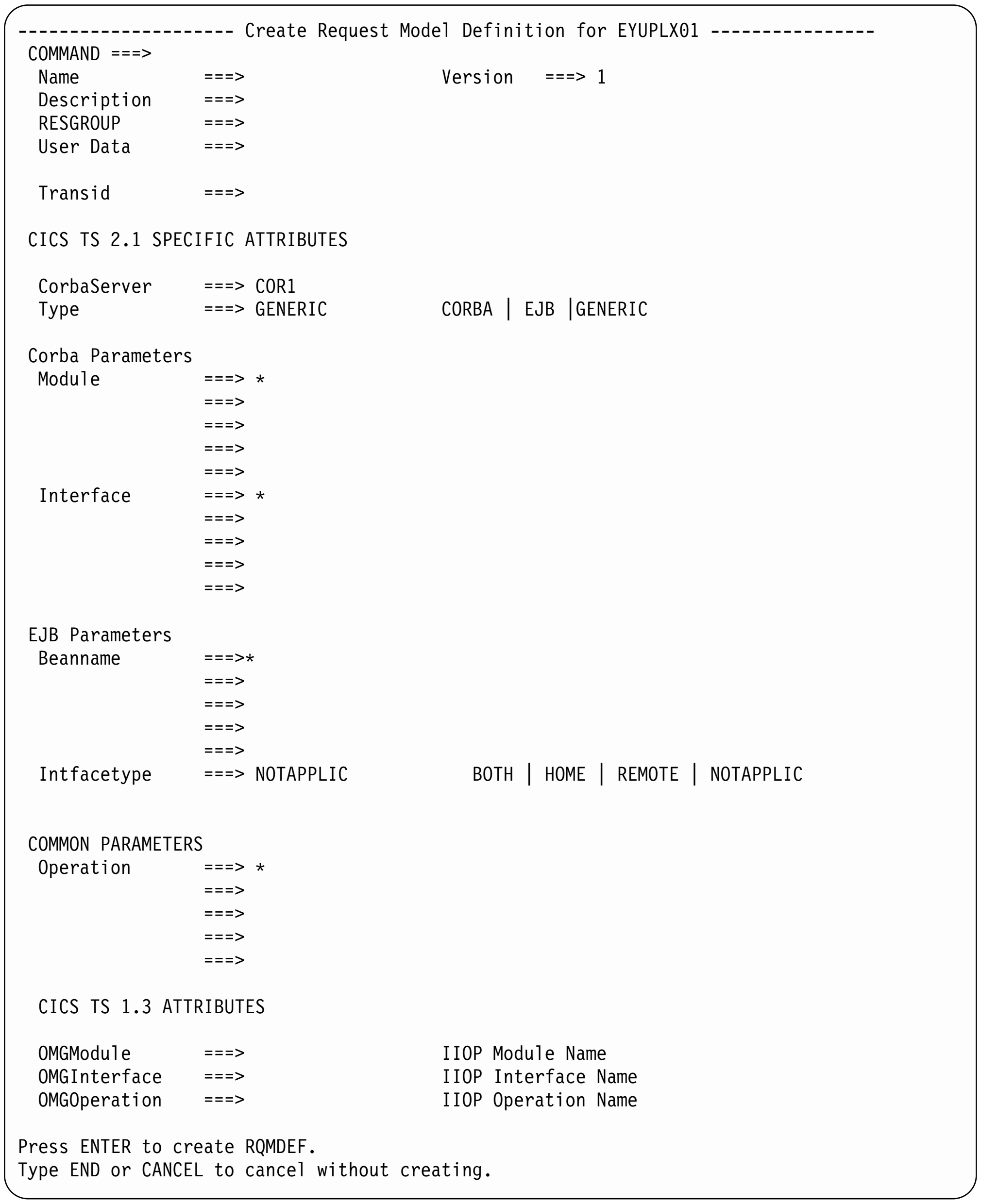


```
COMMAND  ===>
 Name          ===> EYUPRG01  Version ===> 1
 Description   ===> Weekly Payroll Run - Local
 RESGROUP      ===>
 User Data     ===>

 Language      ===> N/A       (ASSEMBLER, C, COBOL, LE370, PLI, RPG, N/A)
 Reload        ===> NO        New copy of program loaded (NO, YES)
 Resident      ===> NO        Residence status (NO, YES)
 Usage         ===> NORMAL    Storage release (NORMAL, TRANSIENT)
 UseLPAcopy    ===> NO        Program used from LPA/SVA (NO, YES)
 Status        ===> ENABLED   Program status (ENABLED, DISABLED)
 Cedf          ===> NO        CEDF available (YES, NO)
 Datalocation  ===> BELOW     Data location (BELOW, ANY)
 Execkey       ===> USER      Program key (USER, CICS)
 Executionset  ===> FULLAPI   Program run mode (FULLAPI, DPLSUBSET)
 Remotesystem  ===>           Connection name to remote CICS system
 Remotename    ===>           Program name in remote CICS region
 Transid       ===>           Tranid for remote CICS to attach
 Rsl           ===> PUBLIC    Resource security value (0-24,PUBLIC,blank)
 Dynamic       ===> NO        Dynamic routing (NO, YES)
 Concurrency   ===> QUASIRENT Concurrency (N/A, QUASIRENT, THREADSAFE)
 JVM           ===> N/A       Java Virtual Machine (NO, YES, DEBUG)
 JVMClass                     Java Virtual Machine Class
               ===>
               ===>
               ===>
               ===>
               ===>
 Hotpool       ===> NO        Hot pooling (NO, YES)
 JVM Profile   ===>           JVM profile name
Press ENTER to create PROGDEF.
Type END or CANCEL to cancel without creating.
```

Figure 28. The PROGDEF panel

RQMDEF (request model definition)

The modified RQMDEF panel is:

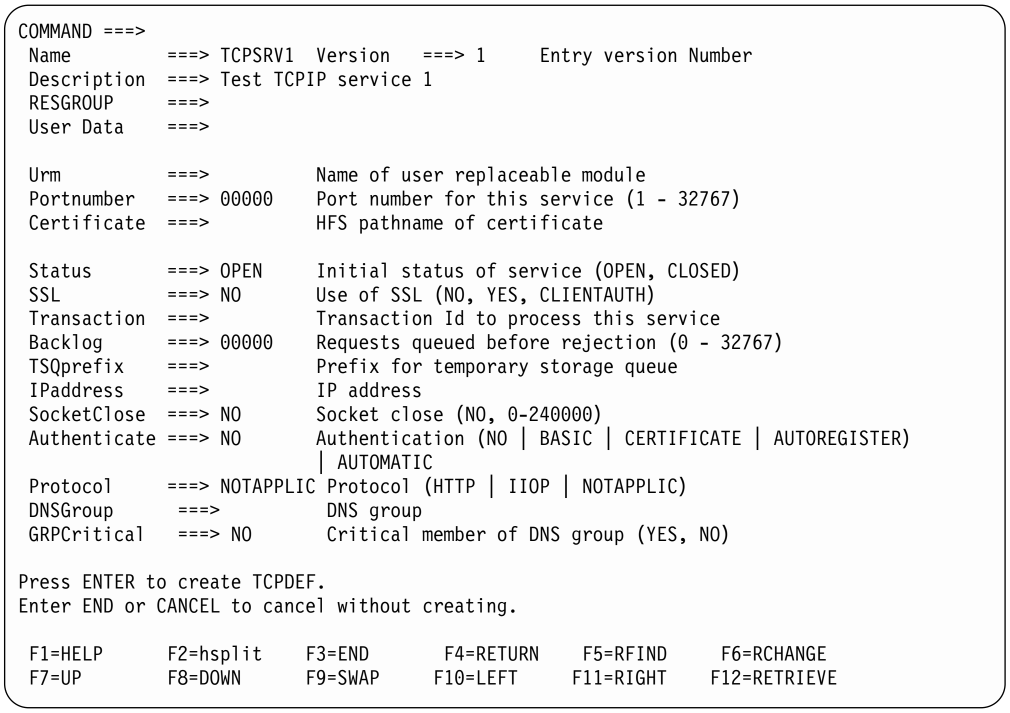


```
-------------------- Create Request Model Definition for EYUPLX01 ----------------
 COMMAND ===>
  Name            ===>                   Version   ===> 1
  Description     ===>
  RESGROUP        ===>
  User Data       ===>

  Transid         ===>

 CICS TS 2.1 SPECIFIC ATTRIBUTES

  CorbaServer     ===> COR1
  Type            ===> GENERIC           CORBA | EJB |GENERIC

 Corba Parameters
  Module          ===> *
                  ===>
                  ===>
                  ===>
                  ===>
  Interface       ===> *
                  ===>
                  ===>
                  ===>
                  ===>

 EJB Parameters
  Beanname        ===>*
                  ===>
                  ===>
                  ===>
                  ===>
  Intfacetype     ===> NOTAPPLIC            BOTH | HOME | REMOTE | NOTAPPLIC


 COMMON PARAMETERS
  Operation       ===> *
                  ===>
                  ===>
                  ===>
                  ===>

  CICS TS 1.3 ATTRIBUTES

  OMGModule       ===>                   IIOP Module Name
  OMGInterface    ===>                   IIOP Interface Name
  OMGOperation    ===>                   IIOP Operation Name

Press ENTER to create RQMDEF.
Type END or CANCEL to cancel without creating.
```

Figure 29. The RQMDEF panel

For descriptions of the request model resource definition attributes, see the *CICSPlex System Manager Managing Business Applications*.

TCPDEF (TCP/IP services definition)
There are four new attributes: Protocol, Authenticate, DNSGroup, and GRPCritical. The changed panel is shown in Figure 30 on page 135.



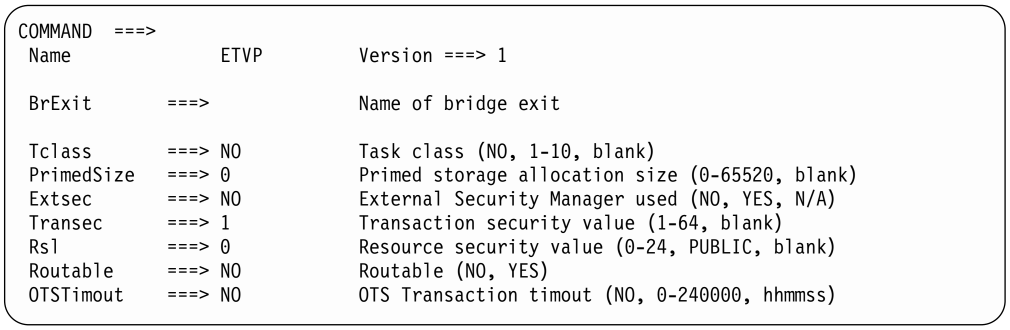
```
COMMAND ===>
 Name         ===> TCPSRV1  Version   ===> 1     Entry version Number
 Description  ===> Test TCPIP service 1
 RESGROUP     ===>
 User Data    ===>

 Urm          ===>         Name of user replaceable module
 Portnumber   ===> 00000    Port number for this service (1 - 32767)
 Certificate  ===>         HFS pathname of certificate

 Status       ===> OPEN     Initial status of service (OPEN, CLOSED)
 SSL          ===> NO       Use of SSL (NO, YES, CLIENTAUTH)
 Transaction  ===>          Transaction Id to process this service
 Backlog      ===> 00000    Requests queued before rejection (0 - 32767)
 TSQprefix    ===>          Prefix for temporary storage queue
 IPaddress    ===>          IP address
 SocketClose  ===> NO       Socket close (NO, 0-240000)
 Authenticate ===> NO       Authentication (NO | BASIC | CERTIFICATE | AUTOREGISTER)
                            | AUTOMATIC
 Protocol     ===> NOTAPPLIC Protocol (HTTP | IIOP | NOTAPPLIC)
 DNSGroup      ===>          DNS group
 GRPCritical   ===> NO       Critical member of DNS group (YES, NO)

Press ENTER to create TCPDEF.
Enter END or CANCEL to cancel without creating.

 F1=HELP      F2=hsplit    F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 30. Creating a TCP/IP service definition

For descriptions of the TCP/IP service resource definition attributes, see the *CICSPlex System Manager Managing Business Applications*.

TRANDEF (transaction definition)

The OTSTimeout attribute is added to the fourth detail panel of the TRANDEF resource definition:



```
COMMAND  ===>
 Name             ETVP         Version ===> 1

 BrExit       ===>             Name of bridge exit

 Tclass       ===> NO          Task class (NO, 1-10, blank)
 PrimedSize   ===> 0           Primed storage allocation size (0-65520, blank)
 Extsec       ===> NO          External Security Manager used (NO, YES, N/A)
 Transec      ===> 1           Transaction security value (1-64, blank)
 Rsl          ===> 0           Resource security value (0-24, PUBLIC, blank)
 Routable     ===> NO          Routable (NO, YES)
 OTSTimout    ===> NO          OTS Transaction timout (NO, 0-240000, hhmmss)
```

Figure 31. The TRANDEF fourth panel

For descriptions of the transaction resource definition attributes, see the *CICSPlex System Manager Managing Business Applications*.

New operations views

The new operations views to support enterprise beans are:

EJCOBEAN

A general view of beans within a CorbaServer.

EJCOBEAD
A detailed view of a bean within the specified CorbaServer.

EJCOBEAS
A summary view of beans within a CorbaServer.

EJCOSE
A general view of CorbaServers within a CICS system.

EJCOSED
A detailed view of a CorbaServer within a CICS system.

EJCOSE2
A detailed view of the JNDIPrefix and Shelf attributes of a CorbaServer within a CICS system.

EJCOSE3
A detailed view of the Host and Certificate attributes of a CorbaServer within a CICS system.

EJCOSES
A summary view of CorbaServers within a CICS system.

EJDJAR
A general view of CICS-deployed JAR files within a CorbaServer.

EJDJARD
A detailed view of a CICS-deployed JAR file within a CorbaServer.

EJDJARS
A summary view of CICS-deployed JAR files within a CorbaServer.

EJDJBEAN
A general view of beans within a CICS-deployed JAR file.

EJDJBEAD
A detailed view of a bean within the specified CICS-deployed JAR file.

EJDJBEAS
A summary view of beans within a CICS-deployed JAR file.

Modified operations views

The operations details views modified to support enterprise beans are:

PROGRAM
The PROGRAMD view has a new attribute: JVM Profile.
#

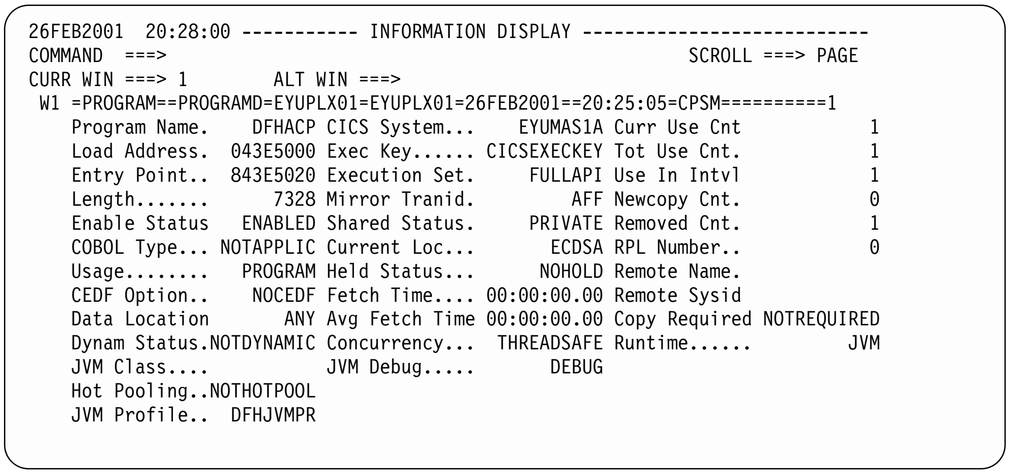

```
26FEB2001  20:28:00 ----------- INFORMATION DISPLAY ---------------------------
COMMAND  ===>                                                 SCROLL ===> PAGE
CURR WIN ===> 1        ALT WIN ===>
 W1 =PROGRAM==PROGRAMD=EYUPLX01=EYUPLX01=26FEB2001==20:25:05=CPSM=========1
    Program Name.    DFHACP CICS System...    EYUMAS1A Curr Use Cnt            1
    Load Address.  043E5000 Exec Key...... CICSEXECKEY Tot Use Cnt.            1
    Entry Point..  843E5020 Execution Set.     FULLAPI Use In Intvl            1
    Length.......      7328 Mirror Tranid.         AFF Newcopy Cnt.            0
    Enable Status   ENABLED Shared Status.     PRIVATE Removed Cnt.            1
    COBOL Type... NOTAPPLIC Current Loc...       ECDSA RPL Number..            0
    Usage........   PROGRAM Held Status...      NOHOLD Remote Name.
    CEDF Option..    NOCEDF Fetch Time.... 00:00:00.00 Remote Sysid
    Data Location       ANY Avg Fetch Time 00:00:00.00 Copy Required NOTREQUIRED
    Dynam Status.NOTDYNAMIC Concurrency...  THREADSAFE Runtime......         JVM
    JVM Class....           JVM Debug.....       DEBUG
    Hot Pooling..NOTHOTPOOL
    JVM Profile..  DFHJVMPR
```

Figure 32. The PROGRAMD panel

RQMODEL

The new attributes (Module, Interface, Operation, Beanname, Type, Intfacetype, and CORBA server) are added to the RQMODELD view:

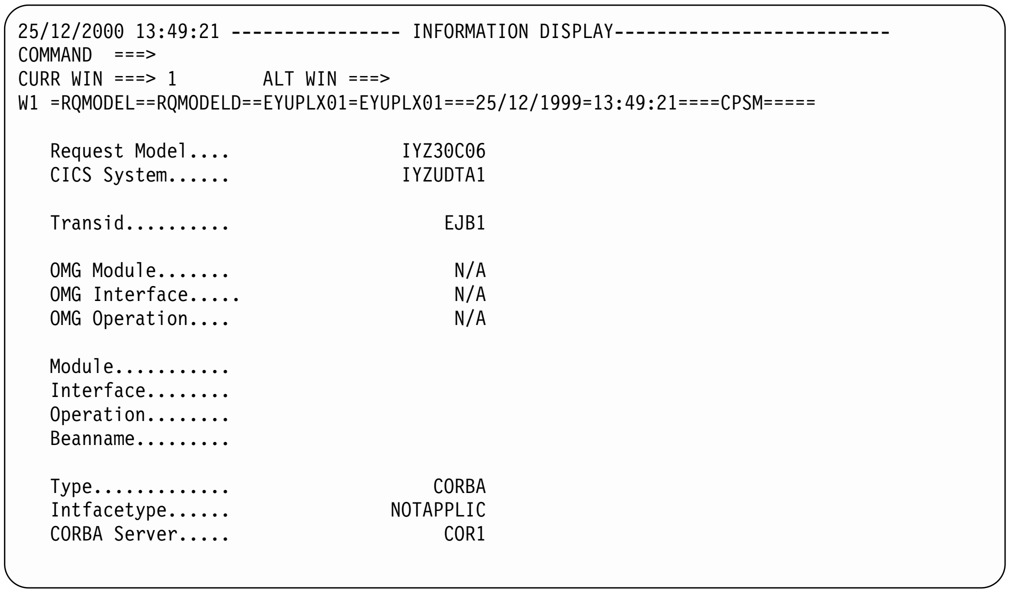
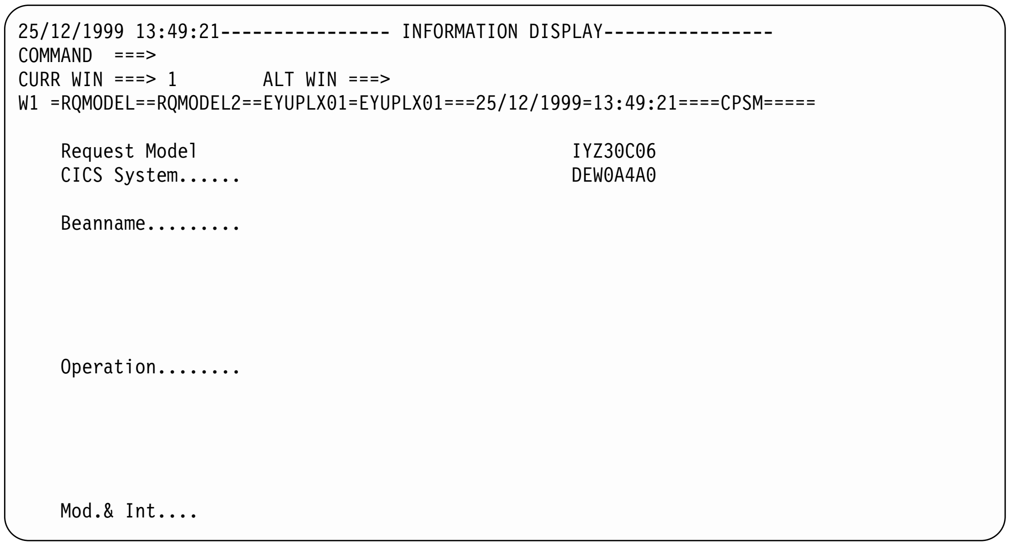

```
25/12/2000 13:49:21 ---------------- INFORMATION DISPLAY--------------------------
COMMAND  ===>
CURR WIN ===> 1        ALT WIN ===>
W1 =RQMODEL==RQMODELD==EYUPLX01=EYUPLX01===25/12/1999=13:49:21====CPSM=====

   Request Model....                IYZ30C06
   CICS System......                IYZUDTA1

   Transid..........                    EJB1

   OMG Module.......                     N/A
   OMG Interface.....                    N/A
   OMG Operation....                     N/A

   Module...........
   Interface........
   Operation........
   Beanname.........

   Type.............                   CORBA
   Intfacetype......               NOTAPPLIC
   CORBA Server.....                    COR1
```

Figure 33. The RQMODELD panel

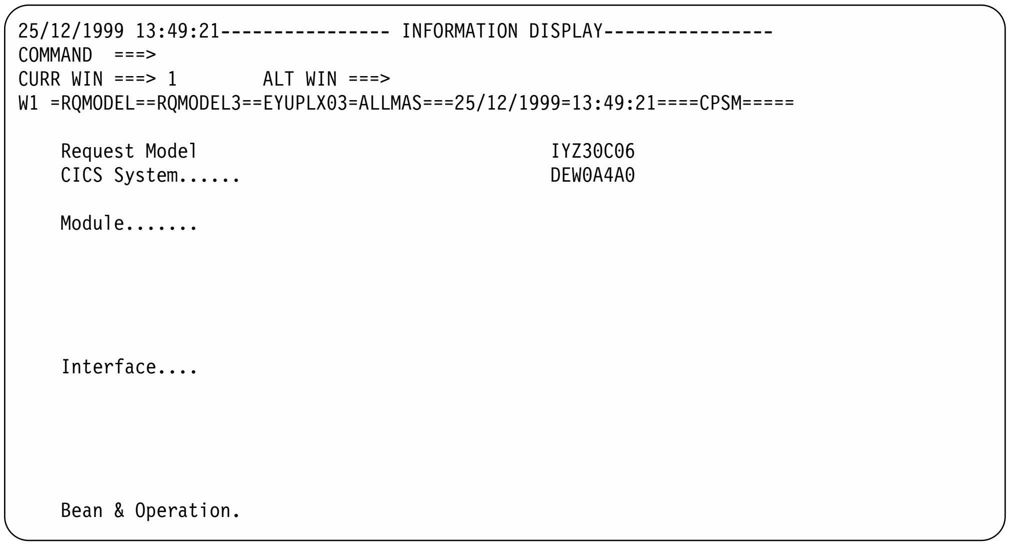
The Beanname and Operation fields hyperlink to the new RQMODEL2 view:



```
25/12/1999 13:49:21---------------- INFORMATION DISPLAY----------------
COMMAND  ===>
CURR WIN ===> 1        ALT WIN ===>
W1 =RQMODEL==RQMODEL2==EYUPLX01=EYUPLX01===25/12/1999=13:49:21====CPSM=====

    Request Model                                   IYZ30C06
    CICS System......                               DEW0A4A0

    Beanname.........




    Operation........




    Mod.& Int....
```

Figure 34. The RQMODEL2 panel

The Module and Interface fields hyperlink to the new RQMODEL3 view:

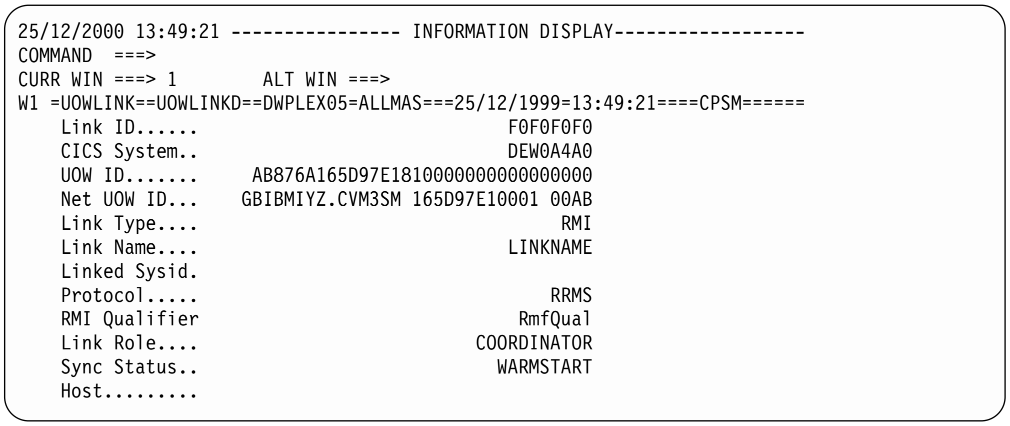


```
25/12/1999 13:49:21---------------- INFORMATION DISPLAY----------------
COMMAND  ===>
CURR WIN ===> 1        ALT WIN ===>
W1 =RQMODEL==RQMODEL3==EYUPLX03=ALLMAS===25/12/1999=13:49:21====CPSM=====

    Request Model                                 IYZ30C06
    CICS System......                             DEW0A4A0

    Module.......




    Interface....




    Bean & Operation.
```

Figure 35. The RQMODEL3 panel

UOWLINKD

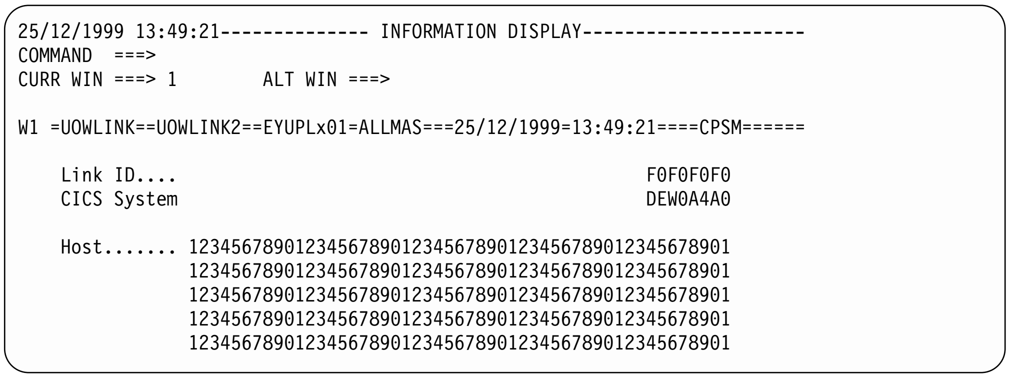
The new attribute (Host), is added to the UOWLINKD view:



```
25/12/2000 13:49:21 ---------------- INFORMATION DISPLAY------------------
COMMAND  ===>
CURR WIN ===> 1        ALT WIN ===>
W1 =UOWLINK==UOWLINKD==DWPLEX05=ALLMAS===25/12/1999=13:49:21====CPSM======
    Link ID......                          F0F0F0F0
    CICS System..                          DEW0A4A0
    UOW ID.......     AB876A165D97E1810000000000000000
    Net UOW ID...    GBIBMIYZ.CVM3SM 165D97E10001 00AB
    Link Type....                               RMI
    Link Name....                          LINKNAME
    Linked Sysid.
    Protocol.....                              RRMS
    RMI Qualifier                           RmfQual
    Link Role....                       COORDINATOR
    Sync Status..                         WARMSTART
    Host.........
```

Figure 36. The UOWLINKD panel

The Host field hyperlinks to the new UOWLINK2 view:

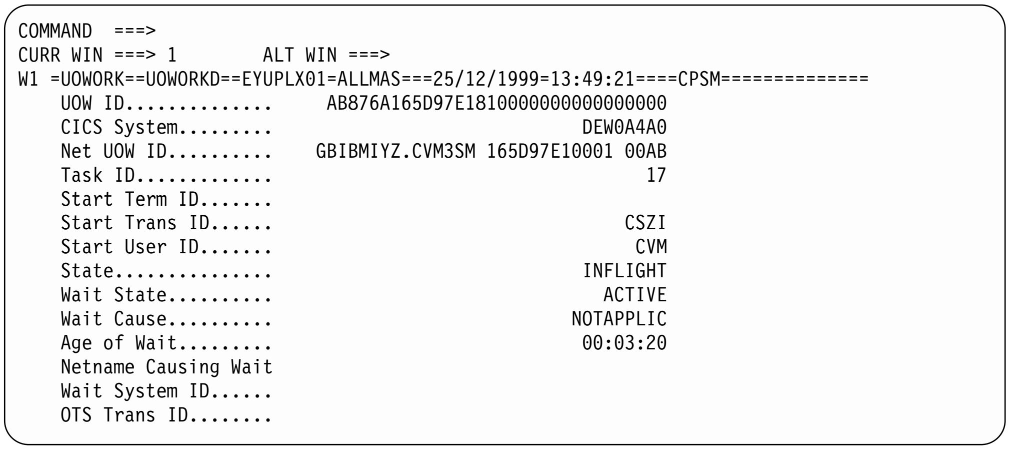


```
25/12/1999 13:49:21-------------- INFORMATION DISPLAY---------------------
COMMAND  ===>
CURR WIN ===> 1        ALT WIN ===>

W1 =UOWLINK==UOWLINK2==EYUPLx01=ALLMAS===25/12/1999=13:49:21====CPSM======

    Link ID....                                         F0F0F0F0
    CICS System                                         DEW0A4A0

    Host....... 12345678901234567890123456789012345678901234567890 1
                12345678901234567890123456789012345678901234567890 1
                12345678901234567890123456789012345678901234567890 1
                12345678901234567890123456789012345678901234567890 1
                12345678901234567890123456789012345678901234567890 1
```

Figure 37. The UOWLINK2 panel

UOWORKD

The new attribute (OTS Trans ID) is added to the UOWORKD view: # #



```
COMMAND  ===>
CURR WIN ===> 1        ALT WIN ===>
W1 =UOWORK==UOWORKD==EYUPLX01=ALLMAS===25/12/1999=13:49:21====CPSM=============
    UOW ID.............     AB876A165D97E1810000000000000000
    CICS System.........                            DEW0A4A0
    Net UOW ID.........    GBIBMIYZ.CVM3SM 165D97E10001 00AB
    Task ID............                                   17
    Start Term ID.......
    Start Trans ID......                                CSZI
    Start User ID.......                                 CVM
    State...............                            INFLIGHT
    Wait State..........                              ACTIVE
    Wait Cause..........                           NOTAPPLIC
    Age of Wait.........                            00:03:20
    Netname Causing Wait
    Wait System ID......
    OTS Trans ID........
```

Figure 38. The UOWORKD panel

The OTS Trans ID field hyperlinks to the new UOWORK2 detail view:

```
25/12/1999 13:49:21-------------- INFORMATION DISPLAY-------------
COMMAND  ===>
CURR WIN ===> 1        ALT WIN ===>
W1 =UOWORK==UOWORK2==EYUPLX01=ALLMAS===25/12/1999=13:49:21==CPSM===
    UOW ID.....                AB876A165D97E1810000000000000000
    CICS System                                         DEW0A4A0

    OTS Trans ID                   12345678901234567890123456789012
                                   12345678901234567890123456789012
                                   12345678901234567890123456789012
                                   12345678901234567890123456789012
```

Figure 39. The UOWORK2 panel

Web User Interface changes

The CICSPlex SM Web User Interface can be used to manage the new resource tables and the new resource table attributes.

The Starter Set provided includes changed views and new views for the new operations resource tables. These changes and additions are similar to those provided by the EUI for new and changed resources.

CICSPlex SM API changes

The CICSPlex SM API has been enhanced by:

- New BAS resource definition tables for the new operation objects; see “New BAS resource definition tables”.
- Modified BAS resource definition tables for the changed operation objects; see “Modified BAS resource definition tables” on page 141.
- New operations resource tables; see “New operations resource tables” on page 141.
- Modified operations resource tables; see “New operations resource tables” on page 141.

New BAS resource definition tables

The CICSPlex SM API supports enterprise beans with resource tables for the new objects:

EJCODEF resource table
A CICS definition that describes the physical and operational characteristics of a CorbaServer.

EJCINGRP resource table
A BAS definition that describes the membership of a CorbaServer definition (EJCODEF) in a resource group (RESGROUP).

EJDJDEF resource table
A CICS definition that describes the physical and operational characteristics of a CICS-deployed JAR file.

EDJINGRP resource table
A BAS definition that describes the membership of a CICS-deployed JAR file definition (EJDJDEF) in a resource group (RESGROUP).

Modified BAS resource definition tables

The PROGDEF, TRANDEF, RQMDEF, and TCPDEF resource tables are enhanced
to include the new attributes.

The RESDESC resource table is enhanced to include attributes for the new resource definition types:

CorbaServers	EJCDEFRG	CorbaServer definition resource group
	EJCDEFTS	CorbaServer definition target scope
	EJCDEFRS	CorbaServer definition related scope
CICS-deployed JAR files	EJDDEFRG	CICS-deployed JAR file definition resource group
	EJDDEFTS	CICS-deployed JAR file definition target scope
	EJDDEFRS	CICS-deployed JAR file definition related scope

New operations resource tables

EJCOBEAN resource table
A CICS resource that describes an enterprise bean object in a CorbaServer being managed by CICSPlex SM.

EJCOSE
A CICS resource that describes a CorbaServer installed in an active CICS system being managed by CICSPlex SM.

EJDJAR resource table
A CICS resource that describes a CICS-deployed JAR file in an active CICS system being managed by CICSPlex SM.

EJDJBEAN resource table
A CICS resource that describes an enterprise bean object in a CICS-deployed JAR file being managed by CICSPlex SM.

Modified operations resource tables

The PROGRAM, RQMODEL, TCPIPS, UOW and UOWLINK resource tables are
enhanced with the new attributes.

CICSPlex SM workload management of enterprise beans

This chapter describes CICSPlex SM workload management of enterprise beans. It covers the following topics:
- "Introduction to workload management of enterprise beans" on page 142
- "Programming interfaces" on page 145
- "Workload balancing" on page 142
- "Workload separation" on page 143
- "Transaction affinities" on page 145

Introduction to workload management of enterprise beans

CICSPlex SM provides dynamic workload management of enterprise beans executing in CICS-provided CorbaServers. CICS Transaction Server for z/OS, Version 2 Release 1 extends the distributed routing program (DSRTPGM) routing model.

For workload management of enterprise beans, the roles of the CICS regions are:

Requesting region
: The region in which the routing request originates. For enterprise bean invocations, this may be external client code, or another EJB logical server that invokes an enterprise bean.

Routing region
: The CICS region in which the decision is taken on where the transaction identified in the request model should be run. The routing region must be running CICS® Transaction Server for z/OS™, Version 2 Release 1.

Target region
: The CICS region in which the transaction identified in the request model runs. For enterprise bean invocations, this is typically an AOR. The target region must be running CICS® Transaction Server for z/OS™, Version 2 Release 1.

In order to manage enterprise bean workloads, you need to create a *logical EJB server*, which will typically consist of a number of cloned routing regions and cloned target regions. (See “Logical servers — enterprise beans in a sysplex” on page 25 for details.) A CICSplex involved in the workload management of enterprise beans may contain one or more logical EJB servers, and regions that are not involved with processing enterprise bean invocations.

Each cloned target region may run a number of *CorbaServers*. A CorbaServer provides the execution environment for enterprise beans and stateless CORBA objects. See “The execution environment” on page 13 for more information.

CICS notifies the distributed routing program EYU9XLOP of all enterprise bean routing requests. EYU9XLOP passes control to EYU9WRAM for:

- In the routing region:
  - Notification
  - Route selection
  - Route selection error
  - Route attempt complete
- In the target region:
  - Transaction initiation
  - Transaction termination
  - Transaction abend

Workload management processing is unchanged.

Workload balancing

Workload balancing of enterprise beans can be achieved using the queue and goal algorithms. The process is shown in Figure 40 on page 143. The inbound IIOP work request is received by a routing region (listener) in system group EYUSGEJ1 and is matched to an enterprise bean name, an operation and a CorbaServer using a request model definition. The routing region routes the transaction identified in the request model to a target region in the CICS system group. The transaction runs in the CorbaServer corresponding to the installed request model instance..

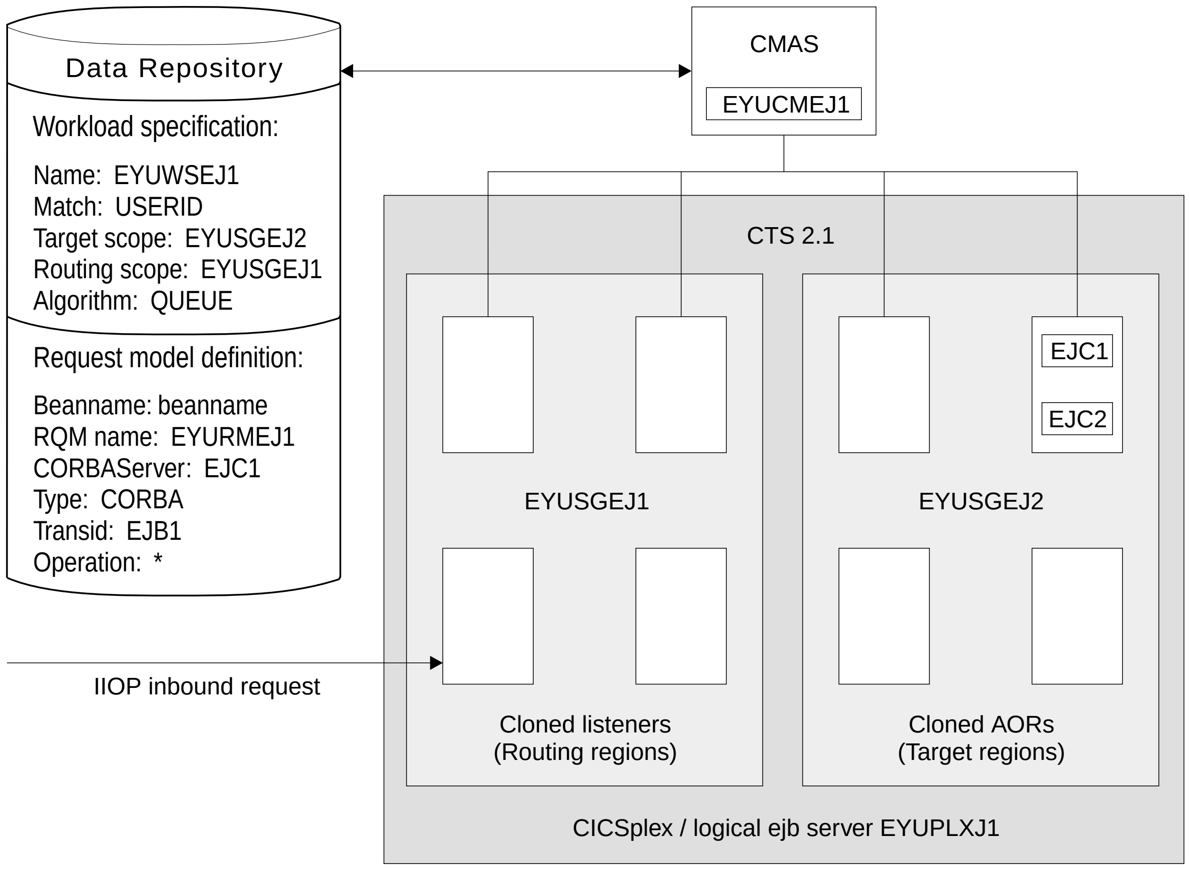


Figure 40. Sample workload balancing definition for dynamic routing of enterprise beans

The routing regions and targets regions must be connected via MRO.

Workload separation

Workload separation of enterprise beans is by transaction group or id. Figure 41 on page 144 illustrates what such a workload might look like.

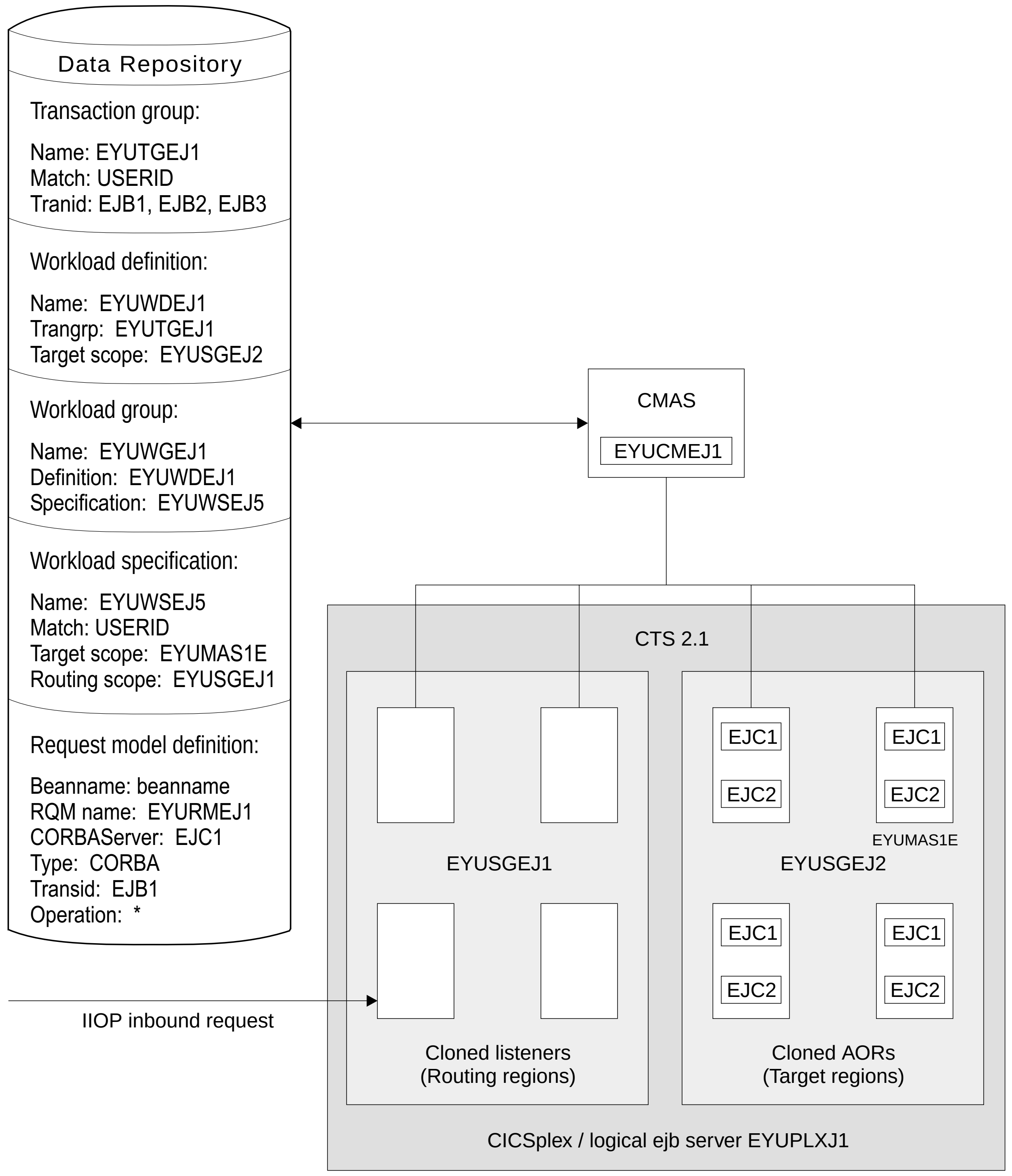


Figure 41. Sample workload separation definition for dynamic routing of enterprise beans

An incoming IIOP enterprise bean request to CICS includes the enterprise bean name, the operation, and (via the enterprise bean naming prefix) the CorbaServer. This information is matched to a predefined request model definition that is installed in the routing regions (cloned listener regions) and the target regions (cloned CICS AORs). The request model identifies the CICS transaction to be run.

The routing program identifies the workload definition that names the transaction group containing the transaction. After determining the appropriate set of target regions, CICSPlex SM selects one based on the status of the active target regions in that set. The CICS system name is returned to the routing region, which routes the work request to the named target. The transaction is run in the CorbaServer corresponding to the installed request model instance.

Transaction affinities

Transaction affinities have no meaning in the workload management of enterprise beans. Transaction affinity relation and lifetime fields in the workload management views should be left blank.

Programming interfaces

For workload management of enterprise beans, you must set, in all listener and target regions, the DSRTPGM SIT parameter:

```
DSRTPGM=EYU9XLOP
```

You must also set routing support active in all listener and target regions, by using the CPSM CICSSYS view to set ROUTING SUPPORT ACTIVE = YES for each region.

Finally, as workload management of enterprise beans follows the Distributed Routing model for workload management, you must ensure that all the regions (both listener and target regions) are correctly associated with an appropriate workload management specification.

The dynamic workload management for enterprise beans is identified by a new routing type (DYRTYPE) in the EYURWCOM and EYURWTRA communication areas. CICSPlex SM accepts and processes this new routing type. All the existing workload management facilities provided by CICSPlex SM (workload balancing, workload separation, and abend avoidance) are supported for this new routing type. The EYURWCOM communication area has been enhanced by the introduction of a new routing type:

WCOM_DYRTYPE
Specifies the type of routing request:

7	For routing of an IIOP request.

The EYURWTRA communication area has been enhanced by the introduction of a new routing type:

WTRA_DYRTYPE
Specifies the request type:

7	The request type is an IIOP request.

CICSPlex SM provides a user replaceable module (URM) EYU9WRAM that allows you to customize your routing logic. The control blocks passed by CICSPlex SM to this URM contain the new information provided to CICSPlex SM by CICS.

Part 3. System management and Parallel Sysplex support

This Part describes all the new function included in CICS to provide enhancements to CICS system management and Parallel Sysplex support. It covers the following topics:

Chapter 10. CICS support for VTAM alias facility

This chapter describes CICS support for VTAM LU alias names. It covers the following topics:

Overview

VTAM LU alias facilities provide improved connectivity and inter-operability for terminal networks. Using LU aliases provides support for expanding SNA networks and allows for greater integration of multiple enterprises. Adding support in CICS for the VTAM alias facility enables CICS to use an LU alias for autoinstalled terminals and work stations and thus ensure unique names in a CICSplex comprising terminal-owning and application-owning regions (TORs and AORs).

CICS supports both forms of the VTAM alias function—predefined and dynamic—only where shown in the following table:

Table 9. Devices for which CICS supports VTAM dynamic LU aliases

	CICS-to-CICS APPC connections (APPL definitions)		**APPC devices (LU definitions)**				**Terminals**	
	Synclevel 1	**Synclevel 2**	**Synclevel 1**		**Synclevel 2**			
	Predefined alias only		**Predefined alias**	**Dynamic alias**	**Predefined alias**	**Dynamic alias**	**Predefined alias**	**Dynamic alias**
VTAM	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CICS	Yes	No	Yes	Yes	No	No	Yes	Yes

The LU alias is used as the NETNAME for terminals and work stations that logon to a CICS region. However, the network qualified name is supplied for problem determination on the CEMT and EXEC CICS INQUIRE TERMINAL, INQUIRE CONNECTION, and INQUIRE NETNAME commands, on a new option called NQNAME. This is also added to the CEST command, and supplied on selected messages and trace.

CICS does not support LU alias for synclevel 2 connections (LUTYPE 6.1 and 6.2) and ignores any LU alias for these LU types, and continues to use the network name defined in the VTAM APPL statement.

Dynamic LU alias support

CICS supports the use of a dynamic LU alias for CICS terminals and workstations that are autoinstalled only. You enable dynamic LU alias support by specifying LUAPFX on the VTAM APPL definition for any CICS terminal-owning region that could receive duplicate netnames. Also, when starting VTAM, specify the following options on the START command:

- NQNMODE=NQNAME
- CDRSCTI=*n* to specify the length of time that the session name should last after the last session has logged off.

VTAM generates a dynamic LU alias only if LUAPFX is specified on the CICS APPL statement and the resource comes from another network. That is, it has a different network name from the network to which the CICS region belongs.

Predefined LU alias support

CICS supports the use of a predefined LU alias for CICS terminals and workstations that are explicitly defined and those that are autoinstalled. You can also use a predefined LU alias for CICS regions that communicate using CICS intersystem communication (ISC). You enable predefined alias support by specifying LUALIAS=*alias* on any cross-domain resource (CDRSC) that needs a specific alias.

Note: A terminal or APPC sync level 1 work station that is defined to CICS on an explicit resource definition (that is, it is not autoinstalled) and is in a different network, requires a CDRSC definition with a specific alias on the LUALIAS parameter. This overrides the dynamic generation of an alias where LUAPFX is specified on the CICS region's APPL statement. To ensure that CICS can match the VTAM LU alias with the installed terminal definition, the LUALIAS value must match the NETNAME specified on the CICS TERMINAL resource definition.

An LUALIAS option in the CDRSC is effective if the resource comes from another VTAM domain (or network). That is, it is not used if the resource comes from the same MVS image, but is used if the resource comes from another MVS image regardless of whether it is from the same sysplex, another sysplex in the same network, or from a different sysplex. If an LU alias is predefined, a dynamic LU alias is not generated.

When to use VTAM LU aliases with CICS

Use VTAM dynamic or predefined LU aliases as described in the following topics.

Using dynamic LU alias

Use dynamic LU alias where:

- Your cross-network terminals and workstations that logon to CICS are mainly autoinstalled.

  The CICS region receives logons from terminals and synclevel 1 connections (both parallel and single sessions) and those logons (or binds) are from cross-network resources that might have duplicate network names.

  However, be aware that synclevel 1 connections could become synclevel 2 in the future. For example, if you have a connection between a TXSeries™ CICS and CICS/ESA it will be synclevel 1, but if you change to using TXSeries CICS with a PPC gateway, synclevel 2 will be used. CICS does not support dynamic LU aliases for synclevel 2 APPC connections.
- An AOR receives shipped terminals or connections with duplicate network names from different TORs.

Using predefined LU alias

Use predefined LU alias where:

- Dynamic LU alias is in operation in a CICS region and your terminals or workstations are explicitly defined on CICS terminal resource definitions with explicit terminal identifiers. In this case, you use predefined LU aliases to override the generation of dynamic LU aliases, which CICS would fail to match with any installed resource definition.
- Dynamic LU alias is not in operation in a CICS region, to avoid any conflict with duplicate network names.

Cross-network devices that need predefined LU alias

If the following VTAM cross-network resources are to be connected to a CICS region that is defined to VTAM with LUAPFX specified on its APPL statement, they must each have a CDRSC LUALIAS=netname entry:

- CICS RDO-defined terminals connected from another network. These include VTAM terminals that cannot be autoinstalled:
  - Pipeline terminals
  - Automatic teller machines (3614 and 3624)
  - Devices for which CICS does not receive logons, such as printers.
- LUTYPE6.2 synclevel 1 connections that may be bound using limited resources.

  Like other LUTYPE6.2 connections limited resource connections release their dynamic LU alias when CDRSCTI expires after the last session is unbound. However, these sessions are unbound whenever they are not in use, and if they rebind after the dynamic LU alias is released, CICS would install another connection, potentially with a different LU alias.
- CICS RDO-defined work stations (LUTYPE6.2 synclevel 1 connections) connected from another network.
- Resources that require an LU name in a RACF profile definition, or resources for which prior knowledge of the LU name is required.

Considerations when using VTAM LU aliases

When planning to use the VTAM LU alias facility, consider the following points:

CDRSCTI timeout values

Make the time specified on CDRSCTI long enough to cover any time interval specified on CICS START commands that are issued against a terminal resource that uses a dynamic LU alias.

This applies to STARTS with a delay that run on both a TOR or AOR. If the CDRSCTI time is not long enough, a resource could log off and then log back on again with a different network name and thus a different TERMID.

The CDRSCTI time interval should also be greater than that specified on the CICS AILDELAY system initialization parameter.

However, if your applications have no dependency on the network name or termid, you can disregard CDRSCTI or set it to 1.

Predictable termids

If you need autoinstalled terminal resources to have a predictable and reproducible TERMID for such things as temporary storage queue names and START requests, you may need to modify your autoinstall user-replaceable module (URM) to select a reproducible TERMID from the network qualified name (NQNAME) supplied in the CINIT or the BIND.

There is an example of such code (commented-out) in the sample autoinstall URM, which extracts the network qualified name from the CINIT and BIND. The example illustrates how to create a TERMID from the last non-blank character of the NETID and the last 3 non-blank characters of the *real* network name (NETNAME).

MVS workload management

If your MVS workload policies specify LU name classifications, remove the LU name for any cross-network resources that are autoinstalled by CICS.

Recovery and persistent sessions support

Resources for which CICS uses any VTAM LU alias (predefined or dynamic) and which come from a different network are not cataloged by a CICS region that is not using persistent session. This means the terminal sessions for the resources cannot be recovered during an emergency restart.

Resources for which CICS uses any VTAM LUALIAS (predefined or dynamic) and which come from a different network are catalogued if CICS is using persistent sessions. This enables CICS to restore resource terminal session information from the CICS catalog pending recovery of the session from VTAM. However, if the resource does not persist, the resource is deleted during an emergency restart.

This action is necessary because VTAM may have been restarted, which would cause dynamic LU aliases to be reissued to different sessions. CICS is unable to tell if VTAM has been restarted, and CICS cannot tell the difference between a predefined and a dynamic LU alias.

CLSDST PASS
If you ISSUE PASS (CLSDST PASS) for a terminal that uses a dynamic LU alias to pass control to another CICS region in another MVS image, the resource will be known by a different network name in the receiving CICS. This is true if the APPL statement of only one or both the CICS regions specify LUAPFX to activate dynamic LU alias.

Generic resources
If a number of generic resource TORs are in two different MVS images, a terminal or work station that logs on to one image is assigned a different network name if it logs off and logs on to a TOR in another image.

TX Series resources
A TX Series resource can use APPC LUTYPE6.2 synclevel 1 or synclevel 2 depending on whether or not it uses a PPC Gateway

- If the PPC Gateway is not used, synclevel 1 communication is used and LU aliases are used.
- If the PPC Gateway is used synclevel 2 communication is used and LU aliases are not used.

FEPI
FEPI front end systems are not supported by VTAM LU alias.

Other factors to consider

Before you decide to enable dynamic LU aliasing, give some thought to the consequences this might have, and review the timing values already in use to ensure that your current operation can benefit without any undesirable side affects. Your choice of what LU alias prefix to use for the CICS APPL definition, and the choice of CDRSC time-out value, might be affected by the following considerations:

- For an LU that originates from a remote network, the alias name bears no relation to the LU’s real name and it has a shorter lifetime such that you cannot use it as a permanent reference to the LU. Weigh this factor against the advantages by understanding what use is made currently of the netname by your systems. If the name acquires meaning only while the LU is logged on to CICS, an LU alias does not pose a problem. If the LU’s name is used in some way as a reference to information after the LU has logged off (perhaps when it logs on again later), take into account the logged-off period when determining the value of CDRSCTI. The question to ask is this: if another LU logs on and is assigned the same name (of an LU that logged off earlier), would that affect the integrity of any data?

Check whether your applications use the EXEC CICS ASSIGN NETNAME command, and for what purpose.

- The LU name might be used by support groups or help-desks in resolving problems. The extent to which such groups can rely on using a dynamically-allocated LU name while the LU is logged off is governed by the CICS AILDELAY and AIRDELAY system initialization parameters.

  Check whether CEMT INQUIRE NETNAME or CEMT INQUIRE TERMINAL NETNAME commands are used for problem determination.
- Alias names created by VTAM are unique only within an MVS image. This means that CICS regions in different images rely on different LUAPFX values to maintain this uniqueness across a Parallel Sysplex, but CICS regions in the same image can share the same LU alias prefix. A consequence of using different prefixes in the same MVS image is that the same LU receives a different name when logging on to two CICS regions when it could have been given the same name. The same prefix in different images only becomes an issue if there is intercommunication between CICS regions across the sysplex.
- The netname is one of the attributes of an LU that is shipped to an AOR by a TOR. If the TOR in one MVS image routes transactions to an AOR in another MVS image, you may need to maintain uniqueness of LU names throughout the CICSplex.
- Consider whether damage could result from a clash of names. Although CICS does not allow two LUs with the same netname to be logged on at the same time in the same region, with dynamic LU aliases the netname can be reused for different LUs. Hence the recommendation that the CDRSCTI value should be greater than that specified on the CICS AILDELAY system initialization parameter. This avoids the situation where a terminal control table entry (TCTTE) created for an LU that has since logged off is reused for another LU because it has been allocated the same LU alias name. Although historically netname has been considered to be unique this is no longer the case. If an LU alias name times out and is reused, any data associated with that netname (or termid) can inadvertently be available to the new LU.

Benefits

Support for VTAM predefined and dynamic LU alias allows CICS regions that are owned by different enterprises, and which may use the same VTAM network names, to interconnect, overcoming the problems of duplicate network names that have occurred in the past.

VTAM dynamic LU alias benefits the interconnection of enterprises for organizations that offer bureau services. Incidents of conflict tend to grow when the number of interconnected networks increases for any reason, which could be a consequence of network restructure, mergers or acquisitions.

Another benefit of the dynamic LU alias facility is that it eliminates the need to predefine cross domain resource definitions (CDRSCs) for CICS autoinstalled devices.

Requirements

The hardware and software requirements for VTAM dynamic LU alias are the same as for CICS TS generally.

Changes to CICS externals

There are changes to a number of CICS external interfaces in support of the VTAM LUALIAS function. These are:

- "Changes to system definition"
- "Changes to resource definition"
- "Changes to the application programming interface" on page 155
- "Changes to the system programming interface" on page 155
- "Changes to CICS-supplied transactions" on page 155
- "Changes to user-replaceable modules" on page 156
- "Changes to monitoring" on page 157
- "Changes to problem determination" on page 157

Changes to system definition

There are changes to the way you define the VTAM APPL definition for your CICS regions to enable VTAM dynamic LU alias. If a CICS region might receive autoinstall requests from another VTAM network, include the LUAPFX parameter in the region's VTAM APPL definition.

LUAPFX=*string*

specifies the prefix characters of the LUALIAS to be assigned when a dynamically generated cross-network CDRSC (with NQNMODE=NQNAME) is created for a session with CICS. VTAM concatenates the characters specified with the next sequential number available to form a VTAM-generated LUALIAS name for the cross-network dynamic CDRSC.

string

indicates the two characters to be used as the prefix for all dynamically generated LUALIAS names for dynamic cross-network CDRSCs in session with the CICS region defined by the APPL statement. Remember to take into account the VTAM naming conventions when choosing this prefix. CICS considerations when specifying the LU alias string are given in the *CICS Transaction Server for z/OS Installation Guide*.

Note: VTAM deletes a dynamically-generated LU alias after a terminal session is closed, or the last session of an APPC parallel sessions connection is closed, and the CDRSCTI-specified timeout interval has expired. The permitted range of timeout values is 1 second to 7 days, but generally the default of 8 minutes is acceptable in most situations. The CDRSCTI timer doesn't start until there are no more sessions involving the resource represented by a CDRSC.

Changes to resource definition

The description of the NETNAME parameter on the TERMINAL and CONNECTION resource definitions is extended to cover VTAM LU alias support:

NETNAME

The following is added to the description of the NETNAME parameter of the TERMINAL resource definition:

If the CICS region supports VTAM dynamic LU alias (that is, LUAPFX=*xx* is specified on the CICS region's APPL statement) and the terminal is in another network, it must be defined to VTAM on a CDRSC definition with a predefined LUALIAS (LUALIAS=*netname*) to override VTAM dynamic allocation. In this case, *netname* on the VTAM LUALIAS parameter must match the NETNAME defined on this terminal resource definition.

CONNECTION

The following is added to the description of the NETNAME parameter of the CONNECTION resource definition:

APPC synclevel 1: If the CICS region supports VTAM dynamic LU alias (that is, LUAPFX=*xx* is specified on the CICS region's APPL statement) this NETNAME is assumed to be in the same network as the CICS region. If it is not the resource must have a local VTAM CDRSC definition with LUALIAS=*netname* defined, where *netname* must match the NETNAME defined on this CONNECTION definition. Synclevel 1 APPC connections are generally work stations.

APPC synclevel 2 and LUTYPE6.1: This NETNAME is assumed to be unique. CICS matches it against the network name defined in the VTAM APPL statement. These connections are generally CICS-to-CICS but could, for example, be TXSeries-connected through a PPC gateway.

Changes to the application programming interface

On an ASSIGN NETNAME command, the NETNAME returned by CICS could be an LU alias, either dynamically allocated by VTAM or predefined on an LUALIAS parameter on a CDRSC definition.

Changes to the system programming interface

There are changes to the following system programming commands:

- INQUIRE TERMINAL | NETNAME
- INQUIRE CONNECTION.

INQUIRE TERMINAL | NETNAME

The 17-character network qualified name, NQNAME, is added for any terminal that received an NQNAME from VTAM at logon time.

Remote terminals do not have NQNAME.

INQUIRE CONNECTION

The network qualified name, NQNAME, is added for any connection that received an NQNAME from VTAM at bind time.

The NQNAME option is supported for problem determination purposes only, and is returned for autoinstalled and RDO-defined resources if it has been supplied by VTAM. However, it is not catalogued for RDO-defined resources and is therefore not available on a restart until that resource logs on again.

If the resource is non-VTAM or a remote terminal, NQNAME is blank. If the resource is a VTAM resource but has not yet received an NQNAME, CICS returns the known netname.

Changes to CICS-supplied transactions

NQNAME option is added to the CEMT INQUIRE TERMINAL, NETNAME, and CONNECTION commands, returning the 17-character network qualified name. See "Changes to the system programming interface" for information about this new option.

CETR

If dynamic LU alias is in operation for the CICS region, and you want to use VTAM exit tracing to trace the bind flows for an autoinstalled terminal, the NETNAME you specify on the CETR Transaction and Terminal Trace panel should be the real network name. However, if you want to trace once the LU alias is known, specify the LUALIAS name. If the real network name is used, and there is more than one network using that name, the VTAM exit tracing is activated for each occurrence of the network name.

Changes to user-replaceable modules

There are changes affecting the node error program (NEP) and the terminal autoinstall program.

Node error program

A new action of print NQN is added to the action flags set by DFHZNAC. Print NQN causes the network qualified name to be printed after any message that contains this flag. The action flag is TWAOPT1, flag 7, set to X'02'. This can be set and unset in the same way as print TCTTE. Print NQN is added as the default action flag for all the following DFHZC messages:

```
0125 0131 0144 0145 0146 0147 0148 0149 0150 0155 0156 0157
2117
2400 2401 2403 2404 2407 2408 2409 2410 2411 2416 2417 2418
2419 2420 2421 2423 2424 2425 2435 2443 2444 2446 2448 2449
2452 2456 2457 2460 2462 2467 2468 2470 2471 2490
3405 3407 3409 3417 3418 3419 3420
3421 3422 3424 3429 3433 3434 3435 3444 3445 3446 3447 3453 3454 3455
3461 3462 3464 3465 3466 3468 3469 3470 3471 3474 3475 3476 3477 3479
3480 3481 3485 3486 3487 3488 3489 3490 3491 3495 4902 4903 4904 4905
4906 4907 4909 4910 4911 4912 4913 4914 4915 4916 4917 4918 4919
4920 4922 4929 4924 4925 4926 4927 4928 4930 4931 4932 4934 4935
4936 4937 4938 4939 4940 4941 4942 4943 4944 4945 4946 4947 4949
6591 6594 6595 6596
```

Note that DFHZC2411 is not related to a specific node. That is, the TCTTE has not yet been created and the message is printed against a dummy TCTTE. The NEP is not called in this case, therefore the default setting cannot be overridden, hence the network qualified name is always printed for DFHZC2411 messages

When DFHZC2410 is issued against the dummy TCTTE, the network qualified name is not printed.

Terminal autoinstall URM

Review your terminal autoinstall URMs, including the default DFHZATDX and DFHZATDY modules, for each CICS region that is using dynamic LU alias (where LUAPFX=*xx* is specified).

In particular, the termid produced by the autoinstall URM should be reviewed for the following reasons:

- The default programs use the last 4 characters of the NETNAME, which will not produce a repeatable TERMID for an LU that is assigned a dynamic LU alias. Consider using the network qualified name in the CINIT or BIND if it is important that the termid is repeatable.
- If you use the last 4 characters of the NETNAME, a dynamic LU alias will produce a termid of 0001, 0002, and so on. Check that your RDO defined terminals do not have such names, and if necessary change your URMs' logic. You could possibly use the last character of the NETID concatenated with the last 3 from the real network name.
- There is some new sample code in DFHZATDX and DFHZATDY that extracts the network qualified name from the CINIT or BIND and uses the last character of the NETID and the last 3 characters of the real network name to provide an alternative TERMID.

If this logic fails to create a termid for any reason it drops through to create the termid from the network name as usual. Note this code is within comments and is supplied only to illustrate how to extract the required information from the CINIT and BIND '0E' control vectors

- The sample code (within comments) is also added to the C, COBOL, and PL/I versions of DFHZATDX. If you use these, note that:
  - The PL/I sample, DFHZPTDX, must be compiled with the PL/I compiler option LANGLVL(SPROG).
  - The COBOL sample, DFHZCTDX, must be compiled with compiler option TRUNC(OPT).

Changes to monitoring

If you are using VTAM dynamic alias support in CICS, do *not* use LUNAME classifications in your workload management (WLM) policies. Alternatively, do not use dynamic LU alias support if you have to use a WLM policy that uses LUNAME classifications.

The LUNAME (network name) passed to WLM is the NETNAME in use by CICS. This might be an LU alias. Additionally, CICS monitoring records include the network qualified name in the form of two 8-character fields containing NETID and real NETNAME. These two fields are valid if the resource had received an NQNAME during BIND or LOGON. If the resource has not logged on, or no NQNAME has been received, NETID is blank and real NETNAME is the same as NETNAME.

There are new monitoring data field changes for VTAM LU alias support:

Additional performance class data fields in DFHTERM

197 TYPE-C, 'NETID', 8 BYTES
NETID if a network qualified name has been received from VTAM. If it is a VTAM resource and the network qualified name has not yet been received, NETID is 8 blanks. In all other cases it is nulls.

198 TYPE-C, 'RLUNAME', 8 BYTES
Real network name if a network qualified name has been received from VTAM. In all other cases this field will be the same as DFHTERM_111.

For non-VTAM resources it is nulls.

Changes to problem determination

There are changes to messages, dump, and trace to include the network-qualified name.

Messages

A new message, DFHZC6907,is issued each time CICS autoinstalls a terminal or connection. This message shows the netname by which CICS will know the device, and the network-qualified name (*netid.realnet*) to show the origin of the resource.

The DFHZNAC message texts are not changed by CICS, but the network-qualified name can be added to the end of the message by your node error program (see “Node error program” on page 156).

Dump

NQNAME is included as part of the TCTTE and TCTSE in the TCP section of CICS dumps.

Trace

The following changes are made to CICS trace entries:

- NQNAME is included in trace point AP FC8A (by module DFHZATA)
- A new trace point, AP FCEE, is created to help you find the VTAM origin of the resource. This trace point contains:

  ```
  Data 1 = NETNAME
  Data 2 = NETID NETNAME
  ```

  This new trace entry is created at logon when the OPNDST is issued.

Chapter 11. Domain name system (DNS) connection optimization

This chapter describes enhancements to CICS support for connection optimization using the domain name system (DNS). It covers the following topics:

Overview

Connection optimization is a technique that uses DNS to balance IP connections and workload in a sysplex domain. In DNS terms, a sysplex is a subdomain that you add to your DNS name space. Connection optimization extends the concept of a DNS host name to clusters, or groups, of server applications or hosts. Server applications within the same group are considered to provide equivalent service. Connection optimization uses load-based ordering to determine which addresses to return for a given cluster.

DNS provides hostname-to-IP address mapping through network server hosts called domain name servers. This support extends the connection optimization introduced in CICS TS 1.3.

DNS registration

Server applications register with the MVS Workload Manager (WLM), which quantifies the availability of server resources within a sysplex. WLM must be configured in goal mode on all hosts within the sysplex. TCP/IP stacks can also register with WLM to provide information on the started IP addresses, or static definitions can be used if stacks do not support registration. When registering, server applications provide the following information:

Group name
This is the name of a cluster of equivalent server applications in a sysplex. It is the name within the sysplex domain that client applications use to access the server applications. CICS uses the DNSGROUP parameter of the TCPIPSERVICE resource definition as the group name to register with WLM.

Server name
This is the name of the server application instance. The server name must be unique among all servers that share the same group name. A server application instance can belong to more than one group. CICS registers with WLM using the specific APPLID of the region as specified by the APPLID system initialization parameter.

Host name
This is the host name of the TCP/IP stack on which the server application runs. During startup, CICS calls the TCP/IP function *gethostbyaddr* to determine the host name of the machine on which it is running, and passes it WLM for registration.

Name resolution example

The following diagram shows an example CICSplex consisting of 4 CICS regions, each executing on separate OS/390 machines within a sysplex.

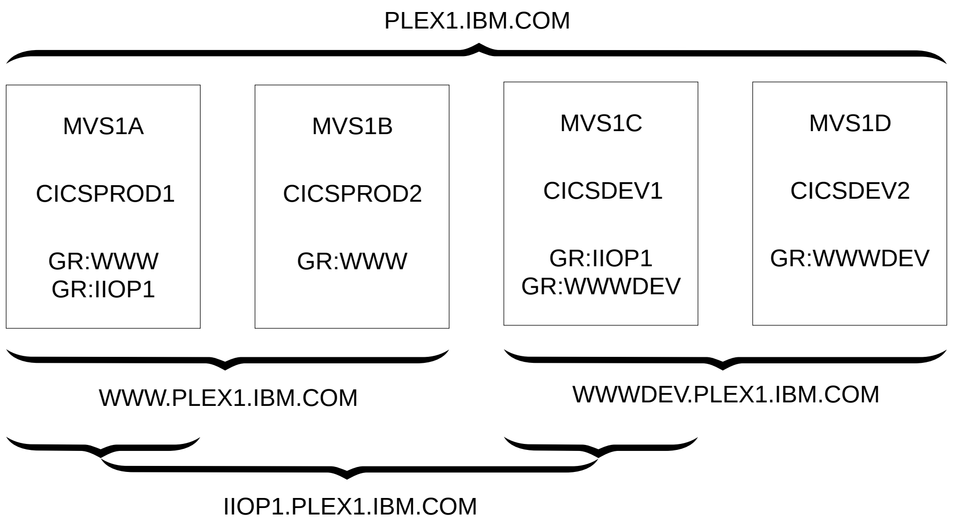


Figure 42. CICSPLEX using DNS connection optimization

The systems are named MVS1A, MVS1B, MVS1C and MVS1D, with the CICS regions having APPLIDs of CICSPROD1, CICSPROD2, CICSDEV1 and CICSDEV2

The sysplex is defined to the DNS to have the name PLEX1 and each MVS machine has a single IP address. The above diagram describes the names that a client machine could use to access the CICS regions based on the following resource definitions installed on each CICS:

- The region CICSPROD1 running on machine MVS1A has 2 TCPIPSERVICE definitions, one specifying a group_name of WWW and the second specifying a group_name of IIOP1
- The region CICSPROD2 running on machine MVS1B has 1 TCPIPSERVICE definition specifying a group_name of WWW
- The region CICSDEV1 running on machine MVS1C has 2 TCPIPSERVICE definitions, one specifying a group_name of IIOP1 and the second specifying a group_name of WWWDEV
- The region CICSDEV2 running on machine MVS1D has 1 TCPIPSERVICE definition specifying a group_name of WWWDEV

The names that a client can access are:

- PLEX1.IBM.COM will return the IP address of any of the machines in the plex
- WWW.PLEX1.IBM.COM will return either the address of MVS1A or MVS1B
- IIOP1.PLEX1.IBM.COM will return either the address of MVS1A or MVS1C
- WWWDEV.PLEX1.IBM.COM will return either the address of MVS1C or MVS1D

You can also address individual CICS regions within a group by using their APPLIDs (or server names). For example, CICSPROD1.WWW.PLEX1.IBM.COM will return the address of MVS1A. This is equivalent to MVS1A.PLEX1.IBM.COM, but the client does not have to know the machine on which the CICSPROD1 server is running, only that CICSPROD1 is part of the WWW group.

Since these names dynamically become available as CICS regions register with the WLM, adding more CICS regions and more MVS machines does not result in any more administration. Using the generic host names (such as WWWDEV.PLEX1.IBM.COM) decouples client applications from specific CICS regions and MVS hosts which enhances availability and scalability

Benefits

This enhancement to CICS DNS support improves the resource definition options and operator interfaces that allow CICS TCP/IP services to register and deregister with OS/390 work load manager (WLM)to take part in DNS connection optimization.

DNS connection optimization provides workload balancing for TCP/IP requests outside CICS, which can complement CICS managed workload balancing within the CICSplex.

Requirements

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

Changes to CICS externals

There are changes to a number of CICS external interfaces in support DNS connection optimization. These are:
- “Changes to system definition”
- “Changes to resource definition” on page 162
- “Changes to the system programming interface” on page 162
- “Changes to CICS supplied transactions” on page 164
- “Changes to problem determination” on page 164

Changes to system definition

To exploit DNS connection optimization, you might need to reconfigure the DNS server that CICS uses to resolve hostnames. In particular, CICS resolves its own hostname at startup using a call to the *gethostbyaddr* function. CICS needs to resolve its own hostname using the DNS server configured for connection optimization in the sysplex. This may not be the system configured name server if the sysplex is already configured for TCP/IP operation. The system name server may not even be on OS/390 or on any of the systems in the sysplex.

You can change the resolver configuration for CICS either by altering system TCP/IP configuration files or by adding or changing the DD statement SYSTCPD in the CICS start-up JCL. This DD name refers to the resolver configuration data set and in this data set is a reference to the DNS server’s IP address. If the DD name is not included in the startup JCL, a number of system files are searched until one is found.

The standard TCP/IP search order to locate the resolver configuration file is defined as follows:
1. The MVS data set or HFS file that is identified in the RESOLVER_CONFIG environment variable
2. The `/etc/resolv.conf` file that resides in HFS
3. The data set specified on the //SYSTCPD DD statement in the CICS startup JCL.
4. *jobname*.TCPIP.DATA for batch jobs, or *userid*.TCPIP.DATA for TSO users or Unix System Sevices shell users
5. The SYS1.TCPPARMS(TCPDATA) data set
6. The TCPIP.TCPIP.DATA data set.

CICS sets the value of the RESOLVER_CONFIG environment variable to the data set referenced by the SYSTCPD DD statement in the CICS JCL. This ensures that the data set referenced by SYSTCPD effectively becomes first in the search order and will take precedence to any existing `/etc/resolv.conf` file.

If the SYSTCPD DD statement is not defined in the startup JCL, RESOLVER_CONFIG is not set, and the normal TCP/IP search order applies. See the *OS/390 IBM Communications Server: IP Configuration Guide* for more information about system TCP/IP configuration files.

Changes to resource definition

Two options, DNSGROUP and GRPCRITICAL, are added to the TCPIPSERVICE definition:

DNSGROUP
specifies the location parameter passed on the IWMSRSRG register call to the OS/390 Workload Manager. The value may be up to 18 characters, and any trailing blanks are ignored. This parameter is referred to as group_name by the TCP/IP DNS documentation and is the name of a cluster of equivalent server applications in a sysplex. It is also the name within the sysplex domain that clients use to access the CICS TCPIPSERVICE.

More than one TCPIPSERVICE is allowed to specify the same group name. The register call is made to WLM when the first service with a group name specified is opened. Subsequent services with the same group name do not cause more register calls to be made. The deregister action is dictated by the GRPCRITICAL attribute. It is also possible to explicitly deregister CICS from a group by issuing a master terminal or SPI command.

GRPCRITICAL(NO|YES)
marks the service as a critical member of the DNS group, meaning that this service closing or failing causes a deregister call to be made to WLM for this group name. The default is NO, allowing two or more services in the same group to fail independently and CICS still remains registered to the group. Only when the last service in a group is closed is the deregister call made to WLM, if it has not already been done so explicitly. Multiple services with the same group name can have different GRPCRITICAL settings. The services specifying GRPCRITICAL(NO) can be closed or fail without causing a deregister. If a service with GRPCRITICAL(YES) is closed or fails, the group is deregistered from WLM.

To implement DNS for IIOP requests (including Enterprise beans), the following CORBASERVER options must be defined:

- The HOSTNAME option of the CORBASERVER definition must have a HOSTNAME defining the corresponding generic host name. This generic hostname is the DNSGROUP value from the TCPIPSERVICE definition, suffixed by the domain or subdomain name managed by the nameserver on MVS. This domain name is established by the TCP/IP administrator.
- The CORBASERVER (or any DJARs within it) must be published to the nameserver (The COS Naming Server) with the generic hostname.

Changes to the system programming interface

Three options are added to the EXEC CICS INQUIRE TCPIPSERVICE command:

DNSGROUP(data-area)
returns the 18–character DNS group name that this TCPIPSERVICE registers with WLM.

DNSSTATUS(cvda)
returns the current state of WLM/DNS registration of this TCPIPSERVICE. The CVDA values are:

NOTAPPLIC
This service is not using DNS connection optimization. No DNSGROUP attribute was specified when the resource was installed.

UNAVAILABLE
Registration is not supported by OS/390.

UNREGISTERED
Registration has not yet occurred (this is the initial state of any service).

REGISTERED
Registration has completed successfully.

REGERROR
Registration has failed with an error.

DEREGISTERED
Deregistration has completed successfully.

DEREGERROR
Deregistration has failed with an error.

GRPCRITICAL(cvda)
returns a CVDA value specifying whether or not this TCPIPSERVICE is a critical member of the DNS group. The CVDA values are:

CRITICAL
If this TCPIPSERVICE is closed, or abnormally stops listening for any reason, the group name specified in the DNSGROUP attribute is deregistered from WLM.

NONCRITICAL
If this TCPIPSERVICE is closed, or abnormally stops listening for any reason, the group name specified in the DNSGROUP attribute is not deregistered from WLM, unless this is the last service in a set with the same group name.

One option is added to the EXEC CICS SET TCPIPSERVICE command:

DNSSTATUS
changes the DNS/WLM registration status of this service. This can be done independently of changing the open or closed status of the service.

To account for timing delays in the deregister request reaching the WLM and the DNS updating its tables, it is advisable to deregister a service before setting it closed. This ensures that client applications do not encounter ″Connection Refused″ situations during the time between the deregister call being issued and the DNS server actually updating its tables. The valid CVDA values are:

DEREGISTER
causes CICS to deregister the group name specified by the DNSGROUP attribute of this TCPIPSERVICE. The WLM macro IWMSRDRS is called and CICS will no longer be a part of the DNS

connection optimization. Any other TCPIPSERVICES that are in the same group (that is, share the same DNSGROUP attribute) are also deregistered.

Changes to CICS supplied transactions

Three options are added to the CEMT INQUIRE TCPIPSERVICE command:

DNSGROUP(data-area)
returns the 18–character DNS group name that this TCPIPSERVICE registers with WLM.

DNSSTATUS(cvda)
returns the current state of WLM/DNS registration of this TCPIPSERVICE. The CVDA values are:

NOTAPPLIC
This service is not using DNS connection optimization. No DNSGROUP attribute was specified when the resource was installed.

UNAVAILABLE
Registration is not supported by OS/390.

UNREGISTERED
Registration has not yet occurred (this is the initial state of any service).

REGISTERED
Registration has completed successfully.

REGERROR
Registration has failed with an error.

DEREGISTERED
Deregistration has completed successfully.

DEREGERROR
Deregistration has failed with an error.

GRPCRITICAL(cvda)
returns a CVDA value specifying whether or not this TCPIPSERVICE is a critical member of the DNS group. The CVDA values are:

CRITICAL
If this TCPIPSERVICE is closed, or abnormally stops listening for any reason, the group name specified in the DNSGROUP attribute is deregistered from WLM.

NONCRITICAL
If this TCPIPSERVICE is closed, or abnormally stops listening for any reason, the group name specified in the DNSGROUP attribute is not deregistered from WLM, unless this is the last service in a set with the same group name.

One option is added to the CEMT SET TCPIPSERVICE command:

DNSSTATUS
allows deregistration of this TCPIPSERVICE.

Changes to problem determination

Sockets domain dump formatting is extended to include details about the DNS registration. This includes the GRPNAME, GRPCRITICAL, and DNSSTATUS attributes for each TCPIPSERVICE.

Chapter 12. Automatic restart of CICS data-sharing servers

This chapter describes automatic restart for CICS data-sharing servers. It covers the following topics:

Overview

All three types of CICS data-sharing server—temporary storage, coupling facility data tables, and named counters—now support automatic restart using the services of the MVS automatic restart manager (ARM). The servers also have the ability to wait during start-up, using an event notification facility (ENF) exit, for the coupling facility structure to become available if the initial connection attempt fails.

Automatic restart

During initialization, a data-sharing server unconditionally registers with ARM, except when starting up for unload or reload. A server does not start if registration fails, with return code 8 or above.

If a server encounters an unrecoverable problem with the coupling facility connection, consisting either of lost connectivity or a structure failure, it cancels itself using the server command `CANCEL RESTART=YES`. This terminates the existing connection, closes the server and its old job, and starts a new instance of the server job.

You can also restart a server explicitly using either the server command `CANCEL RESTART=YES`, or the MVS command `CANCEL` *jobname*`,ARMRESTART`

By default, the server uses an ARM element type of SYSCICSS, and an ARM element identifier of the form DFH*xxnn_poolname* where *xx* is the server type (XQ, CF or NC) and *nn* is the one- or two-character &SYSCLONE identifier for the MVS system. You can use these parameters to identify the servers for the purpose of overriding automatic restart options in the ARM policy.

Waiting on events during initialization

If a server is unable to connect to its coupling facility structure during server initialization because of an environmental error, such as no coupling facility being available, or loss of connectivity to the structure, the server uses an ENF event exit to wait for cross-system extended services (XES) to indicate that it is worth trying again. The event exit listens for either:

- A specific XES event indicating that the structure has become available, or
- A general XES event indicating that some change has occurred in the status of coupling facility resources (for example, when a new CFRM policy has been activated).

When a relevant event occurs, the server retries the original connection request, and continues to wait and retry until the connection succeeds. A server can be cancelled at this stage using an MVS `CANCEL` command if necessary.

Benefits

Automatic restart of the CICS servers ensures that any interruption to service is kept to a minimum by ensuring that a failed server restarts without operator intervention.

Requirements

The hardware and software requirements for automatic restart of data-sharing servers are the same as for CICS TS generally.

Changes to CICS externals

There are changes to a number of CICS externals in support automatic restart of CICS data sharing servers. The externals affected are:

- “Changes to server initialization”
- “Changes to server commands”
- “Changes to problem determination” on page 167

Changes to server initialization

A new category of parameters, for automatic restart manager, is added to the existing set that you use for server startup. These new server initialization parameters enable you to override default processing for the automatic restart function.

Automatic restart manager (ARM) parameters

The new server startup parameters for ARM support are:

ARMELEMENTNAME=*elementname*

specifies the automatic restart manager element name, up to 16 characters, to identify the server to ARM for automatic restart purposes. The permitted characters for the element name are A to Z 0-9 $ # @ and the underscore symbol (_)..

The default identifier is of the form DFH*xxnn_poolname*, where *xx* is the server type (XQ, CF, or NC), *nn* is the &SYSCLONE value for the system (which can be either one or two characters), and *poolname* is the name of the pool served by the server.

This parameter is only valid at server initialization.

This keyword can be abbreviated to **ARMELEMENT** or **ARMELEMNAME**.

ARMELEMENTTYPE=*elementtype*

specifies the automatic restart manager element type, up to 8 characters for use in ARM policies as a means of classifying similar elements. The permitted characters for the element type are A to Z 0-9 $ # and @.

The default element type is SYSCICSS. This parameter is only valid at server initialization.

This keyword can be abbreviated to **ARMELEMTYPE**

Changes to server commands

There are some new options on server commands:

CANCEL RESTART={NO|YES}
terminates the server immediately, specifying whether or not automatic restart should be requested. The default is RESTART=NO.

You can also enter RESTART on its own for RESTART=YES, NORESTART for RESTART=NO.

ARMREGISTERED
shows whether ARM registration was successful (YES or NO).

ARM
This keyword, in the category of display keywords that represent combined options, can be used to display all ARM-related parameter values. It can also be coded as **ARMSTATUS**.

Changes to problem determination

There are new and changed messages to provide information about servers. See the *CICS Transaction Server for z/OS Migration Guide* for details of all new messages.

Chapter 13. Monitoring and statistics changes

This section describes the monitoring and statistics changes in CICS. It contains the following topics:

Overview

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

- A number of additions and improvements have been made to performance class records. They are aimed primarily at improving the way that the monitoring data can be used for offline performance analysis and tuning.
- There are new CICS statistics for CorbaServer resources, derived mainly from the installed CORBASERVER resource definitions.
- There are new CICS global statistics for the pool of JVMs.
- There are new CICS statistics for the request model resources, derived mainly from the installed REQUESTMODEL resource definitions.
- There are new CICS global statistics for TCP/IP support.
- The statistics utility program (DFHSTUP) is enhanced to provide reports for the new CorbaServer, JVM pool, request model, and TCP/IP statistics data as well as enhancements to some existing reports in support of new function provided in this release.
- The statistics sample program (DFH0STAT) is enhanced to provide reports for the new CorbaServer, JVM pool, request model, and TCP/IP statistics data, as well as enhancements to some existing reports in support of new function.

Monitoring

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

Additions and changes to monitoring data

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

- Open transaction environment
- Java Virtual Machine (JVM)
- Enterprise JavaBeans support
- TCP/IP support enhancements
- CICS Web support enhancements

Revised sample monitoring control tables (MCTs) are provided for a terminal-owning region (TOR), an application-owning region (AOR), and a file-owning region (FOR). These show you the types of fields that can be excluded to reduce the size of the performance class record output by CICS monitoring. Using these sample MCTs, you can reduce the size of a performance record by:

- 688 bytes for a TOR
- 96 bytes for an AOR
- 800 bytes for an FOR.

Statistics

There are additional statistics for CorbaServer resources, known by the resource type name CORBASERVER, and mapped by a new copybook, DFHEJRDS.

There are additional statistics for the JVM pool, known by the resource type name JVMPOOL, and mapped by a new copybook, DFHSJGDS, with statistics record ID 117.

There are additional statistics for request model resources, known by the resource type name REQUESTMODEL, and mapped by a new copybook, DFHIIRDS.

There are additional statistics for TCP/IP, known by the resource type name TCPIP, and mapped by a new copybook, DFHSOGDS.

There are changes to a number of CICS statistics copybooks to provide additional information about resources. The changed copybooks are:
- Dispatcher statistics (DFHDSGDS)
- File statistics (DFHA17DS)
- TCP/IP service statistics (DFHSORDS)

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

Changes to CICS externals

The general changes to CICS monitoring and statistics result in a number of changes to CICS externals:
- “Changes to the system programming interface”
- “Changes to CICS-supplied transactions” on page 172
- “Changes to sample programs” on page 172
- “Changes to utility programs” on page 173
- “Changes to monitoring data” on page 173.

Changes to the system programming interface

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

EXEC CICS COLLECT STATISTICS

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

COLLECT STATISTICS

```
►►──COLLECT STATISTICS──┬─────────────────────────┬──────────────────────►◄
                        ├─ ...────────────────────┤
                        ├─CORBASERVER(data-value)─┤
                        ├─ ...────────────────────┤
                        ├─JVMPOOL─────────────────┤
                        ├─ ...────────────────────┤
                        ├─REQUESTMODEL(data-value)┤
                        ├─ ...────────────────────┤
                        ├─TCPIP───────────────────┤
                        └─ ...────────────────────┘
```

You can request specific statistics only for the CORBASERVER resource type, using the option CORBASERVER(*data-value*), where *data-value* is the 4-character name of the Corba server.

You can request global statistics only for the JVMPOOL resource type, using the option JVMPOOL.

You can request specific statistics only for the REQUESTMODEL resource type, using the option REQUESTMODEL(*data-value*), where *data-value* is the 8-character name of the request model.

You can request global statistics only for the TCPIP resource type, using the option TCPIP.

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

ASSEMBLER	CICSTS21.CICS.SDFHMAC
COBOL	CICSTS21.CICS.SDFHCOB
PL/1	CICSTS21.CICS.SDFHPL1

A full list of the statistics data copybooks that you can use is provided in the *CICS Performance Guide*.

- The new copybook for the CorbaServer resource statistics is DFHEJRDS.
- The new copybook for the JVM pool global statistics is DFHSJGDS.
- The new copybook for the request model resource statistics is DFHIIRDS.
- The new copybook for the TCP/IP global statistics is DFHSOGDS.

COLLECT STATISTICS exception conditions

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

EXEC CICS PERFORM STATISTICS RECORD

The PERFORM STATISTICS RECORD command causes the statistics for a named resource to be written immediately to the SMF data set.

PERFORM STATISTICS RECORD

```
►►──PERFORM STATISTICS RECORD──┤ PERF1 ├─────────────────────────────────►◄
```

PERF1

```
├──┬─────────────┬──────────────────────────────────────────────────────┤
   ├─...─────────┤
   ├─CORBASERVER─┤
   ├─...─────────┤
   ├─JVMPOOL─────┤
   ├─ ...────────┤
   ├─REQUESTMODEL┤
   ├─...─────────┤
   ├─TCPIP───────┤
   └─...─────────┘
```

Specify CORBASERVER for the Corba server resource statistics to be written immediately to the SMF data set.

Specify JVMPOOL for the JVM pool global statistics to be written immediately to the SMF data set.

Specify REQUESTMODEL for the request model resource statistics to be written immediately to the SMF data set.

Specify TCPIP for the TCP/IP global statistics to be written immediately to the SMF data set.

PERFORM STATISTICS exception conditions

There are no new exception conditions for the PERFORM STATISTICS command. The existing IOERR, NOTAUTH, and NOTFND can occur for the new CORBASERVER, JVMPOOL, REQUESTMODEL, and TCPIP resource names.

Changes to CICS-supplied transactions

The CORBASERVER, JVMPOOL, REQUESTMODEL, and TCPIP options are added to the CEMT PERFORM STATISTICS RECORD command. These options cause statistics for the named resources to be written immediately to the SMF data set, as in the EXEC CICS PERFORM STATISTICS RECORD command.

Changes to sample programs

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

DFH$MOLS sample monitoring program

DFH$MOLS is enhanced to:

- Handle SMF 110 monitoring data records for CICS TS Version 2 and CICS TS Version 1, in addition to CICS/ESA® Versions 3 and 4

DFH0STAT sample statistics program

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

- TCP/IP
- Data set names
- The JVM pool
- The EJB system data sets DFHEJDIR and DFHEJOS
- CorbaServers and DJARs
- DJARs and enterprise beans
- Requestmodels

Various improvements are made to the following statistics reports:

- System status
- Transaction manager and dispatcher
- Storage manager
- Files
- TCP/IP services
- Terminal autoinstall
- Page index.

There are also major changes to the structure of DFH0STAT, and it is supplied in executable form in SDFHLOAD. See “Changes to DFH0STAT” on page 180 for details.

Changes to utility programs

DFHSTUP, the statistics utility program, is enhanced to

- Produce additional statistics reports for:
  - CorbaServers
  - The JVM pool
  - Request models
  - TCP/IP
- Produce improved statistics reports for:
  - Dispatcher
  - Files
  - TCP/IP services

Changes to monitoring data

The following sections describe the changes to monitoring data fields.

Additional performance class data fields

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

Table 10. Additional performance class data fields

Group Name	Field-Id	Description
DFHSOCK	245	TCP/IP service name
DFHSOCK	246	TCP/IP service port number
DFHSOCK	289	Socket extract request count
DFHSOCK	290	Create non-persistent socket request count
DFHSOCK	291	Create persistent socket request count
DFHSOCK	292	Non-persistent socket high-water-mark
DFHSOCK	293	Persistent socket high-water-mark
DFHSOCK	294	Socket receive request count
DFHSOCK	295	Socket characters received
DFHSOCK	296	Socket send request count
DFHSOCK	297	Socket characters sent
DFHSOCK	298	Socket total request count
DFHSOCK	299	Outbound socket I/O wait time
DFHSYNC	199	OTS indoubt wait time
DFHTASK	192	Request receiver wait time
DFHTASK	193	Request processor wait time
DFHTASK	194	OTS Transaction id (Tid)
DFHTASK	262	User-task Key 8 TCB dispatch time
DFHTASK	263	User-task Key 8 TCB CPU time
DFHTASK	273	CICS JVM initialize elapsed time
DFHTASK	275	CICS JVM reset elapsed time
DFHTERM	197	Network qualified name network ID
DFHTERM	198	Network qualified name network name
DFHWEBB	224	WEB read request count
DFHWEBB	225	WEB write request count

Table 10. Additional performance class data fields (continued)

Group Name	Field-Id	Description
DFHWEBB	238	WEB extract request count
DFHWEBB	239	WEB browse request count

Changed performance class data fields

The following table shows the changed performance class data fields.

Table 11. Changed performance class data fields

Group Name	Field-Id	Description
DFHSOCK	241	Inbound socket I/O wait time
DFHTASK	164	Transaction flags

Part 4. Miscellaneous changes

This Part describes a number of miscellaneous changes to CICS TS. They are described in the following chapter:

- “Chapter 14. Other changes and enhancements” on page 177

Chapter 14. Other changes and enhancements

This chapter describes a number of small changes to CICS and CPSM. It covers the following topics:
- “Integrated CICS translator”
- “Sample program for the XLGSTRM global user exit” on page 178
- “Removal of support for the DFHDCT macro” on page 179
- “Changes to CICS file control” on page 179
- “Changes to DFH0STAT” on page 180
- “Changes to other sample programs” on page 183
- “Changes to CEOT transaction” on page 183
- “Changes to CICSPlex SM support” on page 184
- “Changes in the visual presentation of the CICSPlex SM Web user interface” on page 185
- “Support for additional code pages” on page 185

Integrated CICS translator

This section describes the integration of the CICS command translator with Language Environment-conforming COBOL and PL/I compilers.
- “Overview”
- “Benefits”
- “Requirements”
- “Change to CICS externals” on page 178

Overview

In earlier releases, CICS application programs have to be translated before they can be compiled. The translators find EXEC CICS commands, make them into comments, and generate CALLs appropriate to the language. The CICS-supplied jobs for compiling user application programs all contain an initial job step that invokes the translator appropriate to the compiler invoked in the following job step.

Now, if you use one of the Language Environment-conforming compilers that has integrated the CICS translator, translation of the EXEC CICS commands takes place during program compilation.

Benefits

The stand-alone translators change the line numbers in source programs, which means that an intermediate listing, with the translator-generated CALLs, must be used when debugging an application program. With the integrated translator, application development is made easier because there is only one listing—the original source statements.

The process of translating and compiling is also less error-prone because it is no longer necessary to translate included members separately.

Requirements

The integrated CICS translator is supported by the following releases of the COBOL and PL/I compilers.
- IBM COBOL for OS/390 & VM, Version 2 Release 2, program number 5648-A25, with the PTF for APAR PQ45462.
- IBM VisualAge PL/I for OS/390, Version 2 Release 2.1, program number 5655-B22, with the PTF for APAR PQ45562.

Change to CICS externals

There are no changes to CICS external interfaces for the integrated CICS translator, but compilers that support the integrated translator make some CICS-supplied procedures redundant. Also, if you are using the integrated translator through one of the new compilers, you specify CICS translator options on a new compiler option:
- "Using the CICS-supplied procedures"
- "Specifying CICS translator options"

Using the CICS-supplied procedures

The CICS-supplied procedures that can be used to translate, compile, and link-edit application programs continue to be supported. However, it is recommended that these procedures are not used if the integrated CICS translator is supported by the language compiler.

The Language Environment-conforming language compilers that support the integrated CICS translator scan the application source and call the integrated CICS translator at relevant points.

Specifying CICS translator options

To specify CICS translator options when using PL/I with the integrated translator, specify the compiler preprocessor option, PP(CICS), followed by the CICS translator options inside parentheses. For example:

```
PP(CICS('opt1 opt2 optn ...'))
```

where `opt1 opt2 optn` are options to be passed to the CICS translator. You can specify the PP compiler option wherever PL/I compiler options can be specified.

To invoke the integrated translator when using the COBOL compiler, specify the compiler option CICS, and follow this with the CICS translator options inside parentheses. For example:

```
// PARM='NODYNAM,LIB,OBJECT,RENT,MAP,XREF,CICS(''NOEDF,SP'')'
```

Note: If specified on the PARM string, the CICS translator options must be enclosed in double apostrophes.

Sample program for the XLGSTRM global user exit

If a log stream connection request from CICS to the MVS system logger fails because the log stream is not defined to MVS, CICS issues a request to the MVS system logger to create the log stream dynamically, using a model log stream definition.

The XLGSTRM global user exit, in the log manager domain, is invoked when the CICS log manager detects that a log stream does not exist and before it calls the MVS system logger to define the log stream dynamically. You can use an XLGSTRM exit program to modify the request to MVS to create the new log stream. You could, for example:

- Change the model log stream name passed to the MVS system logger
- Change some of the values in the log stream parameter list used by the MVS system logger to define the log stream.

To help you customize log manager requests, CICS provides the source of a new sample program, DFH$LGLS, which you can use as a skeleton on which to base your own XLGSTRM global user exit program. The sample program, which is supplied in the CICSTS21.CICS.SDFHSAMP library, shows you how to access and change some of the parameters passed to your exit program. Specifically, it:

- Changes the model log stream name referenced by the UEPMLSN exit-specific parameter
- Uses the IXGINVNT macro to change the value of the HIGHOFFLOAD parameter in the log stream definition parameter list pointed to by the UEPIXG exit-specific parameter.

Note: To run the sample program as supplied, first create a model log stream called CICSAD01.DEPT0001.MODEL100. However, you will probably want to tailor the sample to suit your own environment. The source code contains comments to help you do this.

Removal of support for the DFHDCT macro

Runtime support in CICS for the destination control table (DCT) is removed:

- The DCT system initialization parameter is obsolete as a system initialization table (SIT) option and as a system initialization override.
- Transient data (TD) destinations can be defined only in the CSD, as TDQUEUE resource definitions. Migrate your existing DCTs to TDQUEUE resource definitions using the DFHCSDUP MIGRATE command.
- The DFHDCT macro is supported only for migration purposes.

See the *CICS Operations and Utilities Guide* for information on how to migrate DCTs to the CSD using the DFHCSDUP utility program.

See the *CICS Operations and Utilities Guide* for information on how to define TDQUEUEs.

Changes to CICS file control

In earlier releases of CICS, you can define application files as remote files, but you cannot define CICS system files as remote files. For example, a CICS business transaction services (BTS) repository data set may need to be shared between a number of CICS regions in a sysplex. In CICS TS 1.3, BTS repository data sets can only be shared by using them in RLS mode; that is, by defining them as RLSACCESS(YES).

The changes to file control permit a CICS system file (but not the CSD) to be defined as a remote file, enabling it to be shared by any number of application-owning regions (AORs). This change enables some CICS system files to be shared through a file-owning region (FOR), providing an alternative to RLS. The change is effective for the following system data sets only:

- CICS BTS repository data sets
- The EJB directory file, DFHEJDIR
- The EJB object store file, DFHEJOS.

CICS file control requests to these data sets can now be function-shipped, just like any remote VSAM file used by an application program.

Changes to file control API

The restriction that prevents RESP2 values for a remote file exception condition being returned to an AOR is removed. Where applicable, RESP2 values are returned for all file control exception conditions for local and remote files.

Removal of VSAM support in the DFHFCT macros

All VSAM file support is removed from the DFHFCT macros, leaving the file control table (FCT) as a BDAM-only table.

A consequence of removing the VSAM support from the FCT means that you cannot use the CICS TS 2.1 macros to assemble an old FCT with the MIGRATE option in readiness for migrating the VSAM file definitions to the CSD. Also, the CSD utility program, DFHCSDUP, no longer supports the use of the MIGRATE command for the FCT.

Changes to global user exits in the file control domain

Note: A service PTF for the following APAR is required to support this function:
- PQ51277 (Introduce XFCFRIN and XFCFROUT exits)

Two new global user exits are introduced in the file control domain:

XFCFRIN
is invoked on entry to the main file control request gate, FCFR. It allows you to:
- Monitor file control requests and allow them to continue, to be processed by CICS file control
- Intercept file control requests and bypass CICS file control processing altogether
- Redirect the request to a remote region.

XFCFROUT
is invoked on exit after successful or unsuccessful completion of a file control request. It allows you to monitor the results of completed file control requests. This exit is also invoked after exit XFCFRIN determines that CICS file control processing should be bypassed.

The XFCREQ and XFCREQC exits are no longer invoked, on the target region, for function-shipped requests. To intercept a function-shipped file control API request on the target region, use the XFCFRIN exit.

Changes to DFH0STAT

The sample statistics program, DFH0STAT, produces a report showing comprehensive system information about CICS resources (except for terminals and FEPI resources), and an overview of the MVS storage in use. The program demonstrates how you can use EXEC CICS INQUIRE and EXEC CICS COLLECT STATISTICS commands to produce an analysis of your CICS regions. You can use the sample program as supplied, or modify it to suit your needs.

In earlier releases of CICS, DFH0STAT is supplied in source form only, with supporting subroutines in assembler. Although the main programs are still written in COBOL and supplied in source form in the CICSTS21.CICS.SDFHSAMP library, it is now also supplied in pregenerated form in CICSTS21.CICS.SDFHLOAD. The executable version of the program is compiled using a Language Environment-conforming compiler (IBM COBOL for OS/390 & VM 2.1.1) and link-edited with the Language Environment resident routines using the SCEELKED library of OS/390 Release 8.

There are also HTML versions of the BMS maps supplied with the sample application, to enable you to run the STAT transaction using the CICS Web interface.

The restructured sample statistics program consists of the following components, all of which are defined in the CSD group DFH$STAT:

COBOL programs

There are five COBOL programs: DFH0STAT, the main program, which is link edited with four other COBOL modules:

DFH0STAT

This is the main program, which handles all BMS screen input/output, and the open and close of the JES SPOOL. It links to DFH0STLK, which controls all the other routines.

DFH0STLK

This COBOL module is called from DFH0STAT. DFH0STLK performs the following functions:

- Initializes the page numbers
- Links to DFH0STSY.
- Links to DFH0STTP.
- Links to DFH0STPR.
- Prints the page index if selected.

DFH0STPR

This COBOL module is called from DFH0STLK to print the collected statistics for:

- Program and terminal autoinstall and VTAM
- Connections and modenames
- TCP/IP
- TCP/IP services
- JVMPOOL
- EJB system data sets
- CORBAServers and DJARs
- DJARs and enterprise beans
- REQUESTMODELs
- Files and data set names
- Data tables
- DB2 connection and entries
- User exit programs and global user exits
- Enqueue manager and recovery manager.

DFH0STSY

This COBOL module is called from DFH0STLK to print system status and the collected statistics for:

- Transaction manager
- Dispatcher
- Storage manager (DSAs)
- Loader

DFH0STTP

This COBOL module is called from DFH0STLK to print the collected statistics for:

- Transaction classes
- Transactions
- Programs (and programs by DSA and LPA)
- Temporary storage (global)
- Temporary storage queues
- Transient data (global and resource)
- Journal names and log streams

DFH0STCM
The communications area (COMMAREA) used for communication between all the COBOL programs in the DFH0STAT suite.

DFH$STAS
The assembler language subroutine called by the COBOL module DFH0STSY.

DFH$STCN
The assembler language subroutine called by three COBOL modules: DFH0STPR, DFH0STSY, and DFH0STTP.

DFH$STTB
The assembler language table of global user exit names, loaded by the COBOL module DFH0STPR.

DFH0STM
This is the name of one of the map set source files supplied in SDFHSAMP, and also the name of one of the physical mapsets, used by STAT transaction in program DFH0STAT, supplied in SDFHLOAD.

DFH0STS
This is the name of one of the map set source files supplied in SDFHSAMP, and also the name of one of the physical mapsets, used by STAT transaction in program DFH0STAT, supplied in SDFHLOAD.

DFH0STMU
This is the name of the HTML version of the map set DFH0STM, supplied in SDFHSAMP.

DFH0STSU
This is the name of the HTML version of the map set DFH0STS, supplied in SDFHSAMP.

STAT
The transaction that invokes DFH0STAT.

Note: The DFH$STAT CSD group also defines programs DFH$STED and DFH$STER, but these are not part of the DFH0STAT sample application.

The sample program can be invoked as follows:

- As a program list table post-initialization (PLTPI) program, after the DFHDELIM statement.
- As a program list table shut-down (PLTSD) program, before the DFHDELIM statement.
- As a conversational transaction from a CICS terminal
- From a console
- As a started transaction using the EXEC CICS START command from a user-written application program
- By a distributed program link request from a user-written application program

To enable you to run the pregenerated sample statistics program from a CICS terminal:

- Ensure SPOOL=YES is specified as a system initialization parameter for the CICS region.
- Ensure the OS/390 Release 8 Language Environment (or later) run-time libraries are included in the STEPLIB and DFHRPL concatenations, as required.

  All the required executable code and map sets are supplied ready for use in CICSTS21.CICS.SDFHLOAD.

To customize the sample statistics application programs:

- You can use the pregenerated map sets. The following map objects are supplied:
  - Physical map sets, as load modules in CICSTS21.CICS.SDFHLOAD, which you can use unchanged.
  - Symbolic map sets, named DFH0STMD and DFH0STSD, for use as COBOL copybooks in DFH0STAT to enable you to recompile the sample program. These are supplied in CICSTS21.CICS.SDFHSAMP.
  - Map set source macros DFH0STM and DFH0STS, in CICSTS21.CICS.SDFHSAMP, that you can modify if you decide to customize the maps as well as the sample application programs.
  - HTML versions of the maps to enable you to run the sample application using the CICS Web interface. For information on how to create and load the HTML versions of the maps into a template data set, see the *CICS Transaction Server for z/OS Installation Guide*. See also the sample data set creation job, DFHDEFDS, supplied in CICSTS21.CICS.SDFHINST.
- If your COBOL compiler does not have the integrated CICS translator, first translate the customized COBOL program source code, using the translator options COBOL3 and SP.
- Compile the translated output to produce object code.
- Link-edit the object module to produce a load module, which you store in an application load library that is concatenated to the DFHRPL DD statement of the CICS startup job stream.

Changes to other sample programs

The sample CSD extract exit programs, DFH$FORA, DFH0FORC, and DFH$FORP, are updated to handle changes and additions to CSD resource definitions.

Changes to CEOT transaction

There are new options added to CEOT that allow you to alter the uppercase translation status (UCTRAN) for your own terminal, for the current session only.

This new option enables you to switch between the uppercase translation options to suit a specific requirement. For example, you might want to suppress CICS uppercase translation before using the CEDA transaction to define resources with attributes that require mixed-case input, such as HFSFILE, JVMCLASS, SHELF, CERTIFICATE, or DESCRIPTION.

The new keywords are NOUCTRAN, UCTRAN or TRANIDONLY, as shown in the following syntax diagram:

CEOT

```
►►──CEOT──┬──────────────┬──┬───────┬──┬───────┬──┬────────────┬──────────────►◄
          ├─PAgeable─────┤  ├─ATi───┤  ├─TTi───┤  ├─UCtran─────┤
          └─AUtopageable─┘  └─NOAti─┘  └─NOTti─┘  ├─NOUctran───┤
                                                  └─TRanidonly─┘
```

UCTRAN
: The uppercase translation status of your terminal is set to ON for the current session.

NOUCTRAN
: The uppercase translation status of your terminal is set to OFF for the current session.

TRANIDONLY
: The uppercase translation status of your terminal is set to translate only transaction identifiers entered at the terminal for the current session.

Changes to CICSPlex SM support

CICS TS 2.1 CICSPlex SM does not support the following CICS products that were supported by previous levels of CICSPlex SM:

- CICS/MVS® Version 2.1.2 (5665-403)
- CICS/ESA Version 3.3 (5685-083)
- CICS/VSE® Version 2 (any release) (5686-026)
- CICS Transaction Server for VSE/ESA™ Version 1 (5648-054)
- CICS for OS/2 Version 2.0.1 (5648-036)

However, for all except CICS on VSE, you can obtain CICSPlex SM support by using a CMAS at an appropriate lower level. Normally, all communicating CMAS regions should be at the same level, but CICS TS 2.1 CICSPlex SM enables the following CICS products to be controlled through an appropriate lower level of CMAS:

- CICS/MVS Version 2.1.2 (5665-403)
- CICS/ESA Version 3.3 (5685-083)
- CICS for OS/2 Version 2.0.1 (5648-036)

CICSPlex SM is a Tivoli-Ready® product that includes Tivoli® Global Enterprise Manager (GEM) CICSPlex SM Instrumentation. This is a Tivoli GEM agent that uses the CICSPlex SM API to:

- Gather information about CICS regions that are managed by CICSPlex SM.
- Monitor the operational state of the managed CICS regions and display their status
- Provide notification when the state of a managed CICS region changes
- Raise Tivoli events on error conditions
- Exploit real-time analysis (RTA) definitions.

Previous levels of CICSPlex SM have provided the ability to populate the Resource Object Data Manager (RODM) component of NetView with CICS resource state data. This support for NetView is now stabilized, and new object data is no longer added to this interface.

Changes in the visual presentation of the CICSPlex SM Web user interface

The CICSPlex SM Web user interface is changed:

- The colors are modified.
- The navigation icons are moved from the navigation frame to the assistance frame to enhance usability.
- The visual presentation is modified.

Support for additional code pages

CICS regions on System/390® store character data in EBCDIC format. When they exchange character data with ASCII-based systems such as CICS for Windows NT or CICS on Open Systems, the data must be converted between ASCII and EBCDIC formats. For standard conversion of character data between ASCII and EBCDIC, client and server code pages are used. (For explanatory information about data conversion, see the *CICS Family: Communicating from CICS on System/390*.)

CICS TS supports additional client and server code pages. Many of the code pages include the new euro currency symbol.

All the client and server code pages supported by CICS are listed in the *CICS Family: Communicating from CICS on System/390*.

Part 5. Requirements

This Part describes the hardware and software requirements for CICS TS in the following chapter:

- "Chapter 15. Prerequisite hardware and software for CICS Transaction Server for z/OS" on page 189

Chapter 15. Prerequisite hardware and software for CICS Transaction Server for z/OS

This chapter gives some information about related IBM program products that you need to use the CICS and CICSPlex SM elements of CICS Transaction Server for z/OS, and exploit the new and changed function. It covers the following topics:

Hardware prerequisites

To run CICS TS with the base (minimum) functional dependency on OS/390 Version 2 Release 8, you need a System/390 processor that supports that release of OS/390, and which has enough processor storage to meet the combined requirements of the host operating system, CICS, CICSPlex SM, the access methods, and the application programs.

Parallel Sysplex

A Parallel Sysplex environment is required by each of the data-sharing facilities supported by CICS, and by the MVS system logger’s log stream merging facility. If you use any of these facilities, you need:

- One or more coupling facilities with their associated coupling links installed
- An IBM sysplex timer
- Sufficient DASD paths to support the number of central processor complexes (CPCs) in the sysplex.

You can use CICS support for data sharing to access the following forms of data:

- IMS databases
- DB2 databases
- VSAM data sets
- CICS temporary storage
- Coupling facility data tables
- Named counters.

Sysplex timer

A Parallel Sysplex requires an IBM sysplex timer to provide a common external time source.

DASD paths

A Parallel Sysplex requires either DASD controllers with enough paths to dedicate one to each CPC in the sysplex, or an ESCON® director to provide the paths.

Katakana terminal devices

Old-style Katakana terminals that support only single-byte character sets (SBCS) cannot display lower-case Western characters. Therefore, because of the requirement on CICS to issue certain messages in mixed-case, CICS cannot support display or terminal devices that are restricted to the SBCS Katakana only part of code page 930.

Operating system

The OS/390 base requirement for CICS TS is OS/390 Version 2 Release 8 (5647–A01) or later, and all the major elements of OS/390 used by CICS must be at this release level. These are:

- DFSMS/MVS®
- High Level Assembler for MVS & VM
- JES2 or JES3
- Language Environment, plus required service PTFs (see the *CICS Transaction Server for z/OS Program Directory* for details).
- MVS base control program (BCP)
- Communications Server
- Security Server
- SMP/E
- TSO/E
- UNIX System Services, plus required service PTFs (see the *CICS Transaction Server for z/OS Program Directory* for details).

For the following specific functions, you need the additional software shown in the table:

CICS TS Function	Required software
Support for the Java Virtual Machine (JVM)	IBM Developer Kit for OS/390, Java 2 Technology Edition, Version 1.3 (referred to as the IBM persistent reusable JVM), product number 5655-D35.
EJB support tools	IBM Developer Kit for Windows, Java 2 Technology Edition, Version 1.3, at service level 6, plus the EJB 1.1 standard interface classes, provided in `javax.ejb.zip` and `j2ee.jar`. See the note 1 below.
Support for JNDI COS naming directory server	WebSphere Application Server Advanced Edition for Windows NT, V3.5, which requires Windows NT V4.0 or later, or Windows 2000. See note 2 below.
CICS TS Information Center	To read the online information, a Web browser that: • Supports HTML 4.0 • Supports the Document Object Model (DOM) standard • Has frames support enabled. # # # Windows NT or Windows 2000 are the recommended operating environments for the Information Center.To view and print PDF files, Adobe Acrobat Reader, 4.0.

Notes:

1. All three of the EJB support tools (CICS JAR development tool for EJB technology, CICS development deployment tool for EJB technology, and CICS production deployment tool for EJB technology), and IBM Developer Kit for

Windows, Java 2 Technology Edition, Version 1.3, are all supplied with CICS TS on a CD-ROM labeled CICS Tools for EJB Technology, LCD4–4355.

2. A subset of WebSphere Application Server Advanced Edition for Windows NT V3.5 is shipped (on CD-ROM) with CICS TS to provide the required COS naming directory server support. You are recommended to apply Service Pack 3.

IBM database products

CICS supports IMS/ESA® Database Manager and IBM DATABASE 2™ (DB2) as described in this section.

IMS/ESA Database Manager

CICS application programs can access IMS databases, through the DBCTL interface only, using IMS/ESA Database Manager Version 5 Release 1 (5695–176) or later.

IBM DATABASE 2 (DB2)

CICS application programs can access DB2 databases using DB2 Version 5 (5655–DB2) or later.

For CICS JVM programs that use the JDBC and SQLJ APIs, you need PTF UQ49041 for APAR PQ39420 on DB2 Version 5, or PTF UQ49039 for APAR PQ39411 on DB2 Version 6.

IBM telecommunications access methods

VTAM and TCP/IP are included as exclusive elements of OS/390 Release 8.

TCP/IP OS/390 Release 8 includes TCP/IP Socket Interface for CICS, which enables network users access to CICS regions. CICS programs can use the TCP/IP sockets application programming interface (API) to communicate with TCP/IP devices. TCP/IP also enables access to CICS through:

- The CICS ONC RPC support, which enables CICS as a server for Remote Procedure Call (RPC) requests using the Open Network Communication (ONC) standard protocol
- DCE/MVS, which enables CICS as a server for Remote Procedure Call requests using the Distributed Computing Environment (DCE) standard protocol

CICS also uses TCP/IP services for the following protocols:

- Hypertext Transfer Protocol (HTTP), which is supported by the CICS Web API through the CICS sockets domain.
- Internet InterORB Protocol (IIOP), which is supported by CICS IIOP services through the CICS sockets domain.

You can access CICS Transaction Server for z/OS using ACF/TCAM (DCB) Version 2.4 (5735–RC3) plus PTFs, or ACF/TCAM (DCB) Version 3.1 (5665–314) plus PTFs.

MQSeries® for OS/390

If you use MQSeries with CICS TS you need MQSeries for OS/390, Version 2 Release 1 with the PTF for APAR PQ35501, or later.

OS/390 Security Server (RACF)

The Security Server (RACF) available with OS/390 Release 8 provides for all CICS TS security needs.

CICS VSAM Recovery

If you use CICS VSAM Recovery (CICSVR) as your VSAM forward recovery utility, CICSVR Version 2.3 (5695–010) is required.

Tivoli Performance Reporter for OS/390

CICS TS is no longer supported by the earlier versions of performance reporter products (IBM SystemView® Enterprise Performance Data Manager/MVS (EPDM) or IBM SystemView Performance Reporter for MVS). For CICS TS Version 2, you need Tivoli Decision Support for OS/390 (5698-TD9) Version 1.5, with a PTF.

NetView® for MVS/ESA™

For a resource object data manager (RODM) repository that CICSPlex SM exploits through NetView MultiSystem Manager Version 2 Release 2 (5655–126), CICS TS supports NetView for MVS/ESA Version 3 Release 1 (5655–007).

Programming languages

CICS Transaction Server for z/OS supports the following programming languages and environments:

- High Level Assembler/MVS (5696–234)
- IBM VisualAge PL/I for OS/390, Version 2 (5655–B22)

  Note: You need Version 2 Release 2 Modification 1, with the PTF for APAR PQ45562 for the integrated CICS translator.
- IBM PL/I for MVS & VM (5688–235)
- OS PL/I Optimizing Compiler Version 2 Release 1 (5668–910)
- OS PL/I Optimizing Compiler Version 1 Release 5 (5724–PL1)
- IBM COBOL for OS/390 & VM, Version 2, (5648–A25)

  Note: You need Version 2 Release 2, with the PTF for APAR PQ45462 for the integrated CICS translator.
- IBM COBOL for MVS & VM (5688–197)
- VS COBOL II (5668–958 and 5668–023) (requires PTF for APAR 43097)
- C/370™ (5688–040 and 5688–187)
- IBM C/C++ for MVS (5655–121)
- CSP Version 3 or later
- SAA AD/Cycle® COBOL/370™ (5688–197)
- SAA AD/Cycle PL/I (5688–235)
- SAA AD/Cycle C/370 (5688–216)
- VisualAge for Java, Enterprise Toolkit for OS/390 (5655-JAV)

CICS components in object-code-only (OCO) form

Some of the functional areas of CICS are provided, either completely or partially, in object-code-only form (OCO), without licensed source materials. These areas include:

- Authorized cross-memory (AXM) server environment

- Autoinstall terminal model manager, AITM
- 3270 Bridge
- Business application manager domain
- Catalog domains
- Common Programming Interface functions
- Coupling facility data tables
- Coupling facility data table server
- Directory manager domain
- Dispatcher domain
- Document handler domain
- Dump domain
- Enqueue domain
- Enterprise Java domain
- Event manager domain
- EXEC CICS system programming command support
- File control RLS support
- IIOP domain
- JVM domain
- Kernel domain
- Loader domain
- Lock manager domain
- Log manager
- Message domain
- Monitoring domain
- Named counter server
- Object transaction services domain
- Offline statistics utility
- Offline system dump formatting routines
- Parameter manager domain
- Partner resource manager
- Program manager domain
- RDO for VSAM files and LSR pools
- Recovery manager
- Request streams domain
- Resource recovery services (RRS) interface
- SAA communications and resource recovery interfaces
- Scheduler services domain
- Security domain
- Shared data tables
- Sockets domain
- Statistics domain
- Storage manager domain
- Temporary storage data sharing server
- Temporary storage domain
- Timer domain
- Trace domain
- Transaction manager domain
- User domain
- Web domain.

Part 6. Appendixes

Glossary

CICS-deployed JAR file. A deployed JAR file, produced specifically (via several intermediate stages) for the CICS EJB server, which has been stored on the hierarchical file system (HFS) used by OS/390. This name is reserved for the original **deployed JAR file** on the HFS of a CICS system. (There are no specific names for JAR files in the various intermediate stages of deployment).

CICS EJB server. One or more CICS regions that support enterprise beans. A logical CICS EJB server typically consists of multiple (cloned) CICS listener regions and multiple (cloned) CICS AORs. The listener regions and AORs can be combined into listener/AORs.

container. The code that creates and manages enterprise bean instances at run-time, and provides the services required by each enterprise bean running in it. The EJB container supports a number of implicit services, including lifecycle, state management, security, transaction management, and persistence:

CorbaServer. The execution environment defined by a CORBASERVER definition. A CICS EJB server can contain multiple CorbaServers.

CORBASERVER. a CICS resource definition that defines an execution environment for enterprise beans and CORBA stateless objects. You can install multiple CORBASERVER definitions into the same CICS region (giving you different execution environments for enterprise beans). All the AORs in a logical CICS EJB server must contain identical CORBASERVER definitions.

deployed JAR file. A generic term for a file produced from the ejb-jar file. It contains the XML deployment descriptor and enterprise bean classes from the ejb-jar file, plus additional classes generated to support the chosen EJB container

deployment. In the EJB development environment, the act of generating EJB deployed classes for a specific container

deployment descriptor. An XML file defining a collection of enterprise beans and specifying how an EJB server should process the beans.

DJAR. A CICS resource definition that defines a CICS-deployed JAR file. It is not a synonym for the deployed JAR file itself. Installing the DJAR definition into CICS causes CICS to:

- Copy the CICS-deployed JAR file (and the classes it contains) to an HFS ″shelf″ directory that is specific to the CICS region;
- Read the deployed JAR file from the shelf, parse its XML descriptor, and store the information it contains;
- Generate the home interface class for each bean in the JAR file.

EJB. See Enterprise JavaBeans.

enterprise bean. An EJB component that implements a business task or business entity. See**entity bean** and **session bean**.

Enterprise JavaBeans. A component architecture for the development and deployment of object-oriented, distributed, enterprise-level applications.

entity bean. An enterprise bean that represents persistent data maintained in a database. An entity bean can manage its own persistence or it can delegate this function to its container. An entity bean is identified by its primary key. If the container in which an entity bean is hosted crashes, the entity bean, its primary key, and its remote references survive the crash. Entity beans are not supported in this CICS release. See also **session bean**.

extensible markup language (XML). A standard markup language that allows you to define the tags (markup) needed to identify the data and text in XML documents. EJB component architecture uses XML to describe its deployment properties.

factory. An enterprise bean that dynamically creates instances of beans.

JAR file (Java archive). A platform-independent file format that aggregates many files into one. Multiple Java applets and their requisite components (.class files and other resource files) can be bundled in a JAR file and subsequently downloaded in a single HTTP transaction.

Java database connectivity (JDBC). A standard Java API for accessing databases.

Java naming and directory interface (JNDI). A set of APIs that assist with the interfacing to multiple naming and directory services.

Java virtual machine (JVM). A software implementation of a central processing unit (CPU) that runs compiled Java code.

JPDA. See Java Platform Debug Architecture.

JVM. See Java virtual machine.

MOF. Meta-Object Facility; a standard for the definition of information models and the subsequent mapping of these models to CORBA interfaces.

Remote method invocation (RMI). A protocol that allows objects to be distributed over the network; that is, a Java program running on one computer can call the methods of an object running on another computer.

servlet. Server-side Java class that responds to HTTP requests.

session bean. An enterprise bean that is created by a client and that usually exists only for the duration of a single client/server session. A session bean performs operations such as calculations or accessing a database for a client. While a session bean can be transactional, it is not recoverable in the event of a system crash. See also **entity bean**.

XMI. XML metadata interchange; a standard for exchanging metadata information using XML technology. See also **extensible markup language**.

Bibliography

CICS Transaction Server for z/OS

CICS Transaction Server for z/OS Release Guide	GC34-5701
CICS Transaction Server for z/OS Migration Guide	GC34-5699
CICS Transaction Server for z/OS Installation Guide	GC34-5697
CICS Transaction Server for z/OS Program Directory	GI10-2525
CICS Transaction Server for z/OS Licensed Program Specification	GC34-5698

The above titles are the only unlicensed books available in hardcopy for CICS Transaction Server for z/OS Version 2 Release 1. All the remaining CICS and CICSPlex SM books are supplied in softcopy only in the CICS Information Center, which is distributed on CD-ROM.

CICS books for CICS Transaction Server for z/OS

General	
CICS User's Handbook	SX33-6116
CICS Transaction Server for z/OS Glossary	GC34-5696
Administration	
CICS System Definition Guide	SC34-5725
CICS Customization Guide	SC34-5706
CICS Resource Definition Guide	SC34-5722
CICS Operations and Utilities Guide	SC34-5717
CICS Supplied Transactions	SC34-5724
Programming	
CICS Application Programming Guide	SC34-5702
CICS Application Programming Reference	SC34-5703
CICS System Programming Reference	SC34-5726
CICS Front End Programming Interface User's Guide	SC34-5710
CICS C++ OO Class Libraries	SC34-5705
CICS Distributed Transaction Programming Guide	SC34-5708
CICS Business Transaction Services	SC34-5704
Java Applications in CICS	SC34-5881
Diagnosis	
CICS Problem Determination Guide	GC33-5719
CICS Messages and Codes	GC34-5716
CICS Diagnosis Reference	LY33-6097
CICS Data Areas	LY33-6096
CICS Trace Entries	SC34-5727
CICS Supplementary Data Areas	LY33-6098
Communication	
CICS Intercommunication Guide	SC34-5712
CICS Family: Interproduct Communication	SC34-0824
CICS Family: Communicating from CICS on System/390	SC34-1697
CICS External Interfaces Guide	SC34-5709
CICS Internet Guide	SC34-5713
Special topics	
CICS Recovery and Restart Guide	SC34-5721
CICS Performance Guide	SC34-5718
CICS IMS Database Control Guide	SC34-5711
CICS RACF Security Guide	SC34-5720

CICS Shared Data Tables Guide	SC34-5723
CICS Transaction Affinities Utility Guide	SC34-5728
CICS DB2 Guide	SC34-5707

CICSPlex SM books for CICS Transaction Server for z/OS

General	
CICSPlex SM Concepts and Planning	GC34-5732
CICSPlex SM User Interface Guide	SC34-5743
CICSPlex SM Commands Reference Summary	SX33-6117
CICSPlex SM Web User Interface Guide	SC34-5403
Administration and Management	
CICSPlex SM Administration	SC34-5729
CICSPlex SM Operations Views Reference	SC34-5739
CICSPlex SM Monitor Views Reference	SC34-5738
CICSPlex SM Managing Workloads	SC34-5735
CICSPlex SM Managing Resource Usage	SC34-5734
CICSPlex SM Managing Business Applications	SC34-5733
Programming	
CICSPlex SM Application Programming Guide	SC34-5730
CICSPlex SM Application Programming Reference	SC34-5731
Diagnosis	
CICSPlex SM Resource Tables Reference	SC34-5741
CICSPlex SM Messages and Codes	GC34-5737
CICSPlex SM Problem Determination	GC34-5740

Other CICS books

Designing and Programming CICS Applications	SR23-9692
CICS Application Migration Aid Guide	SC33-0768
CICS Family: API Structure	SC33-1007
CICS Family: Client/Server Programming	SC33-1435
CICS Transaction Gateway for OS/390 Administration	SC34-5528
CICS Family: General Information	GC33-0155
CICS 4.1 Sample Applications Guide	SC33-1173
CICS/ESA 3.3 XRF Guide	SC33-0661

Note: The *CICS Transaction Server for OS/390: Planning for Installation* book that was part of the library for CICS Transaction Server for OS/390, Version 1 Release 3, is now merged with the *CICS Transaction Server for z/OS Installation Guide*. If you have any questions about the CICS Transaction Server for z/OS library, see *CICS Transaction Server for z/OS Installation Guide* which discusses both hardcopy and softcopy books and the ways that the books can be ordered.

Determining if a publication is current

IBM regularly updates its publications with new and changed information. When first published, both hardcopy and BookManager® softcopy versions of a publication are usually in step. However, due to the time required to print and distribute hardcopy books, the BookManager version is more likely to have had last-minute changes made to it before publication.

Subsequent updates will probably be available in softcopy before they are available in hardcopy. This means that at any time from the availability of a release, softcopy versions should be regarded as the most up-to-date.

For CICS Transaction Server books, these softcopy updates appear regularly on the *Transaction Processing and Data Collection Kit* CD-ROM, SK2T-0730-xx. Each reissue of the collection kit is indicated by an updated order number suffix (the -xx part). For example, collection kit SK2T-0730-06 is more up-to-date than SK2T-0730-05. The collection kit is also clearly dated on the cover.

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

Index

Notices

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

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

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

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

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

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

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

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

Trademarks

Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To ask questions, make comments about the functions of IBM products or systems, or to request additional publications, contact your IBM representative or your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER
  Hampshire
  SO21 2JN
  United Kingdom

- By fax:
  - From outside the U.K., after your international access code use 44–1962–842327
  - From within the U.K., use 01962–842327
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink™: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Program Number: 5697-E93

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC34-5701-00

Spine information:

IBM CICS Transaction Server Release Guide

Version 2
Release 1