

WebSphere software

IBM

e-business software

## Connecting WebSphere to CICS with CICS Transaction Gateway

Nigel Williams

September 2002

IBM Design Center for e-business infrastructure

Product Support and Solutions Center, Montpellier, France

nigel\_williams@uk.ibm.com

IBM Software Group

- CICS remains IBM's flagship TP monitor. It runs most of today's mission critical business applications and is used by over 14,000 companies.
- This presentation is not actually about CICS itself but how to connect to CICS from WebSphere using the CICS Transaction Gateway. The CTG is a product in it's own right. It's not as mature as CICS itself but it is being used today in a variety of ways and in some of these business critical applications.

# Trademarks

---

- The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:
  - ▶ AIX
  - ▶ CICS
  - ▶ CICS Transaction Gateway
  - ▶ DB2
  - ▶ DFSMS
  - ▶ IBM
  - ▶ IMS
  - ▶ Language Environment
  - ▶ OS/390
  - ▶ RISC System/6000
  - ▶ RACF
  - ▶ RMF
  - ▶ S/390
  - ▶ VisualAge
  - ▶ WebSphere
  - ▶ z/OS
  - ▶ zSeries
- Java and all Java based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- SUN is a registered trademark of Sun Microsystems, Inc. in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Linux is a registered trademark owned by Linus Torvalds.
- Microsoft, Windows, Windows NT, Windows 2000 and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Other company, product or service names may be trademarks or service marks of others.



# Agenda

---

- ▶ **What is the CICS Transaction Gateway ?**
- ▶ **Different interfaces**
- ▶ **How do you use the CTG with WebSphere ?**
- ▶ **Different deployment options**
- ▶ **Customer scenarios**
- ▶ **CTG V5**
- ▶ **CTG and the J2EE Connector Architecture**



CICS Transaction Gateway is IBM's strategic connector for WebSphere. It is a key enabling solution that provides a robust, flexible and scalable solution for companies who have existing business-critical processes in place and need to make these business systems accessible in new ways. It allows users to efficiently integrate applications that operate on the IBM WebSphere Application Server, or any Java-enabled application server with core business systems running on CICS servers. The latest version of CICS TG implements the **J2EE Connector Architecture**, allowing Enterprise Java technology to exploit the proven qualities of CICS.

This session will introduce you to CICS TG, and show you how it can be a significant factor in your WebSphere based e-business solution. The presentation is based on several redbooks and customer project experience whilst working in the IBM Design Center for e-business infrastructure.

# CICS Transaction Gateway

---

*"The purpose of CICS Transaction Gateway is to provide **efficient integration** of middle tier application servers with CICS ... by providing a **multi-user gateway** which supports different **programming interfaces**... for use by Web applications in Java and other languages."*

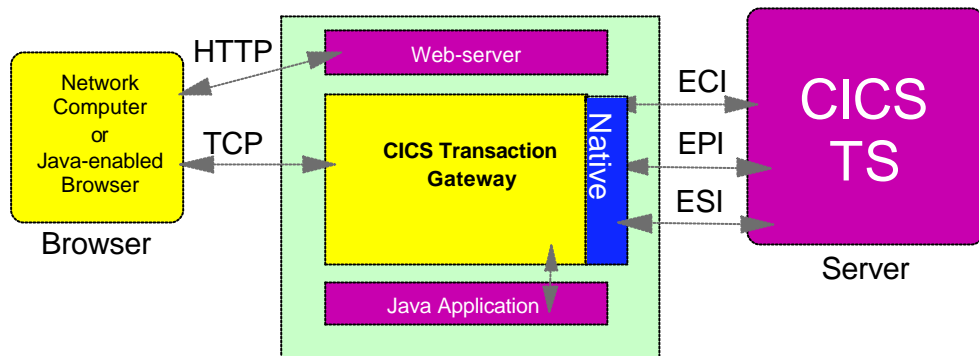
*"CICS Transaction Gateway enables customers to implement **robust and efficient** end-to-end configurations and... provides considerable **flexibility** in the configurations which may be adopted."*

*"The option to deploy CICS Transaction Gateway on OS/390 or z/OS, supporting Web applications locally on OS/390 or z/OS, or remotely on a physical middle tier server... enables **high scalability** to be achieved and has been tested at transaction rates in excess of 1,000 transactions per second."*



- These quotes are taken from the announcement letter for CICS Transaction Gateway Version 5.

# CICS Transaction Gateway - What's That ?



## CICS Transaction Gateway is:

- ▶ Gateway daemon
- ▶ A set of Java classes which can be used to connect to CICS (from applet, application or servlet)
- ▶ Native code which handles communication from Gateway to CICS server

Current Version  
for all platforms  
is **CTG V5**



© IBM Corporation 2002

IBM Software Symposium 2002

The CTG provides the following components:

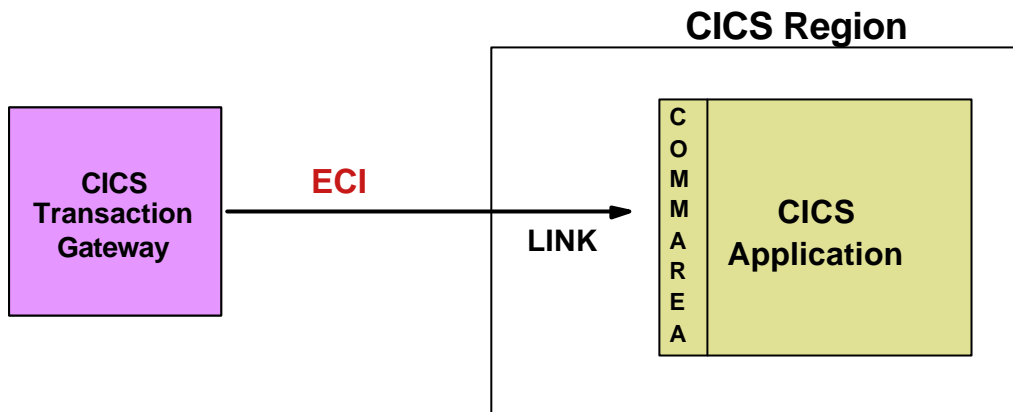
- ▶ A **Gateway daemon** that communicates with CICS applications as a result of receiving requests from Java applets (or perhaps other Java programs)
- ▶ A set of **Java classes**. For example, the class JavaGateway is used to establish communication with the Gateway process
- ▶ **Native code** which handles the communication from the Gateway process to the CICS server

The chart does not show CTG being used with WebSphere - next we will see some options for deploying CTG with WebSphere (we will see that in some cases it is not necessary to use all of the CTG components)

The CTG supports three programming interfaces, the ECI, EPI and ESI. These programming interfaces are provided in Java, C/C++, Visual Basic and COBOL. This presentation will only deal with scenarios in which a Java client is used.

The current version of CTG for all platforms is CTG V5 which has been available since July, 2002.

## CTG Interfaces - ECI



- The CTG has 3 external interfaces
- The **External Call Interface (ECI)**
  - ▶ Link to CICS programs
  - ▶ Available on all platforms



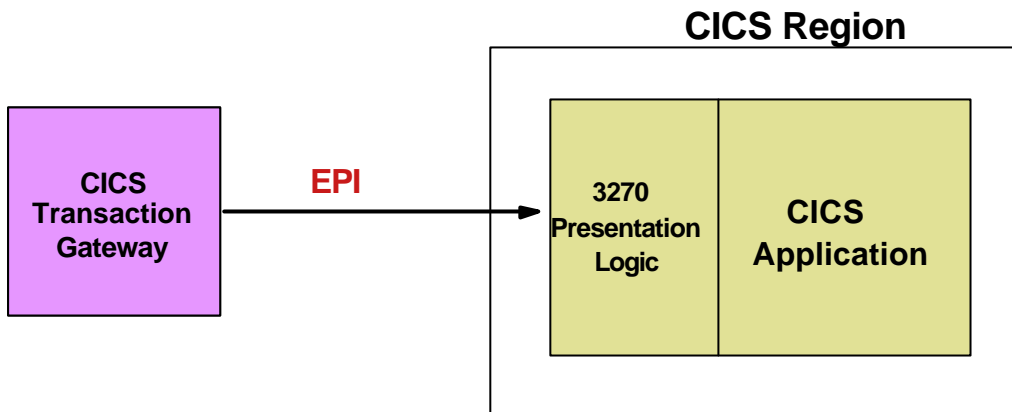
The CICS Transaction Gateway provides three external interfaces:

- ▶ External Call Interface (ECI)
- ▶ External Presentation Interface (EPI)
- ▶ External Security Interface (ESI)

The ECI is used for calling COMMAREA based CICS applications. The COMMAREA is the buffer used for passing the data between the client and the CICS server. CICS sees the client request as just another distributed program link (DPL) request.

ECI is supported on both distributed platforms and on z/OS.

# CTG Interfaces - EPI



- The **External Presentation Interface (EPI)**
  - ▶ Invoke 3270 based transactions
  - ▶ Available on distributed platforms only
  - ▶ EPI support classes and EPI beans provided
  - ▶ Do not need detailed knowledge of 3270 data streams to use



© IBM Corporation 2002

IBM Software Symposium 2002

The EPI is used for invoking 3270 based transactions. A terminal is installed in CICS, and CICS sees the request as running on a remote terminal controlled by the CTG.

EPI is supported only on distributed platforms. The CTG for z/OS does not support EPI, so can not be used to invoke 3270 based transactions.

EPI is useful where presentation logic and business logic are integrated, thus a COMMAREA interface cannot be used to drive the business logic.

**EPI support classes** are provided to make the task of programming the EPI simpler:

- ▶ High-level constructs for handling 3270 data streams (e.g AID,FieldData,Screen,Terminal, Map and MapData) . These are used to represent the interface to a CICS 3270 terminal and the resulting 3270 response.
- ▶ Do not need a detailed knowledge of 3270 data streams to use.

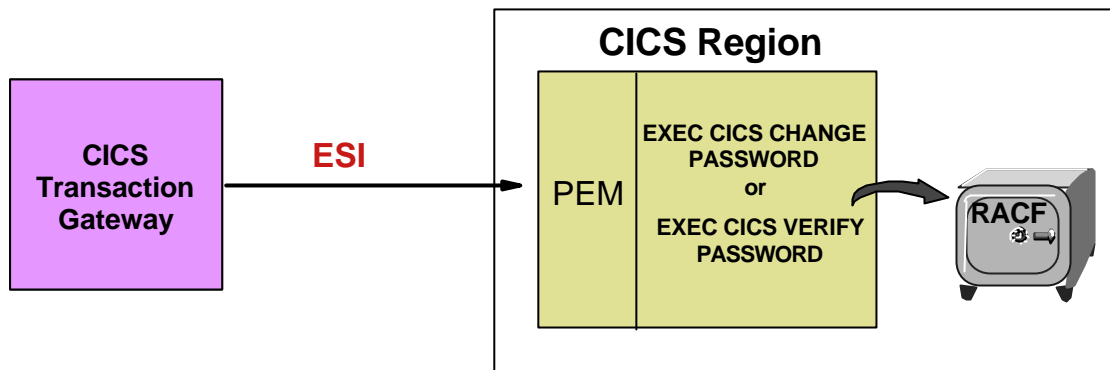
To initiate a 3270 transaction in CICS and receive the reply, the following series of commands is required:

1. Create a JavaGateway object and open a connection to the gateway.
2. Create a Terminal object and use this to install a terminal in CICS and to start a transaction on that terminal.
3. Retrieve the data sent to the 3270 terminal either using:
  - The Screen and Field objects
  - The BMSMapConvert utility and the Map and MapData classes which allow the EPI application to access BMS field names directly based on information in the BMS map.
4. Delete the terminal and close the connection to the JavaGateway.

**EPI beans**

- ▶ Based on the EPI support classes and JavaBean development environment.
- ▶ Allow creation of EPI applications in a visual development environment.

## CTG Interfaces - ESI



- The **External Security Interface (ESI)**
  - ▶ Verify and change password information
  - ▶ Available on distributed platforms only



© IBM Corporation 2002

IBM Software Symposium 2002

The ESI is used for verifying and changing userid and password information with CICS.

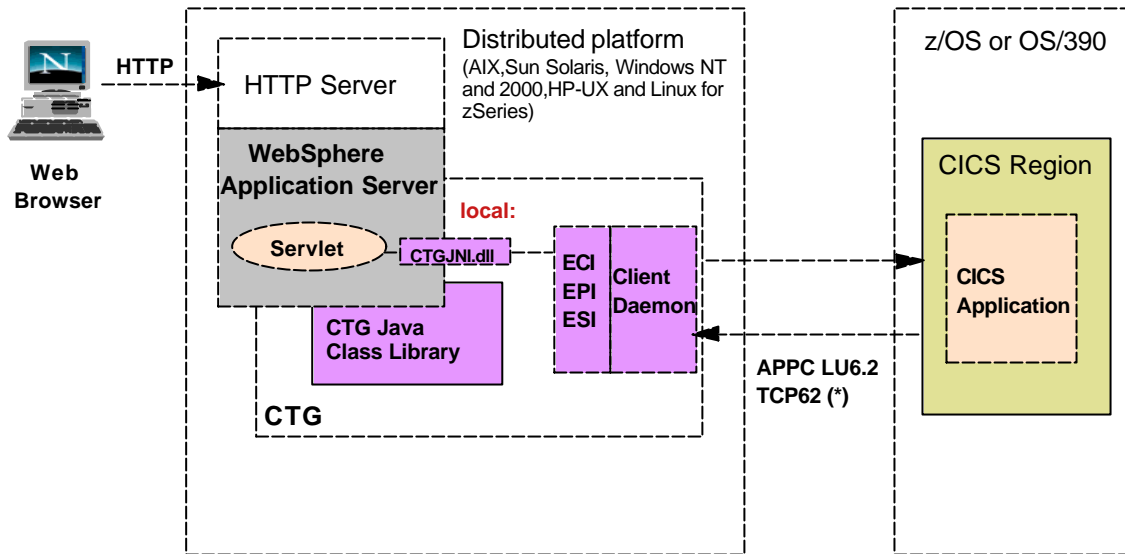
The ESI is based on the CICS Password Expiration Management (PEM) function.

The sample code below changes the CICSRS9 user's password from PASSWORD to JAN2002 on the CICS server SCSCPAA7 connecting through the CTG on tcp://gunner:2006/.

```
try {  
JavaGateway jg=new JavaGateway("tcp://gunner",2006);  
ESIRequest req=  
ESIRequest.changePassword("CICSRS9","NOV2001","PASSWORD","SCSCPAA7");  
jg.flow(req);  
System.out.println("Rc:"+req.getRc());  
jg.close();  
}catch (IOException ioe){  
System.out.println("Handled exception:"+ioe.getMessage());  
}
```

ESI is supported on distributed platforms only.

# Using CTG with WebSphere Distributed



(\*or ECI over TCP/IP can be used with CICS TS V2.2)

Servlet uses **local:** protocol so CTG native code CTGJNI.dll invokes CICS client directly

**ECI** (External Call Interface)  
**EPI** (External Presentation Interface)  
**ESI** (External Security Interface)



© IBM Corporation 2002

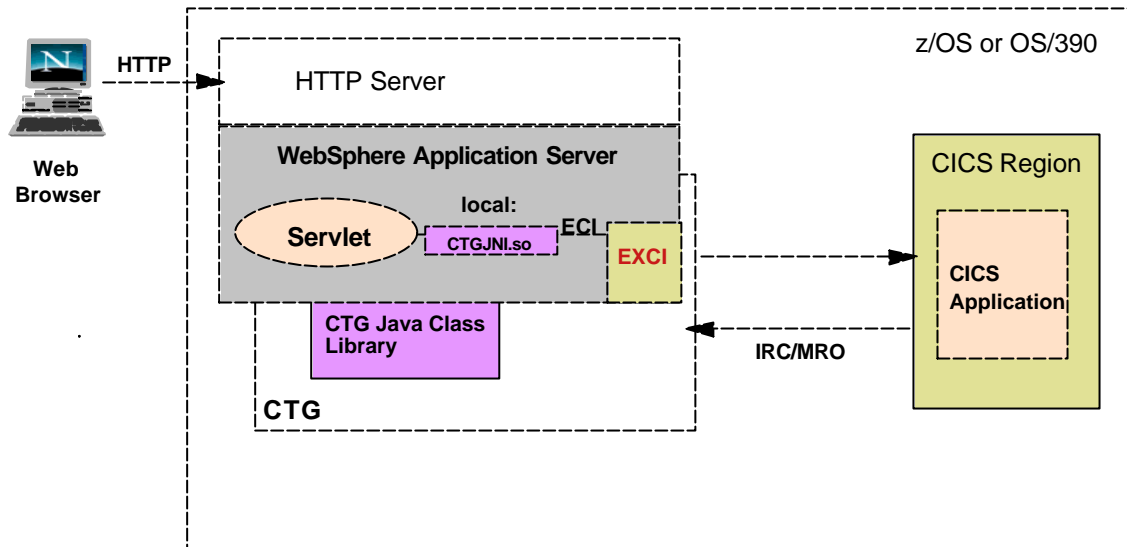
IBM Software Symposium 2002

This chart shows the use of CTG with WebSphere on a distributed platform.

In this deployment, the Gateway daemon is not used because the CTG local: protocol is being used to invoke the CICS Universal Client native libraries directly from the servlet. The CUC then flows the ECI, EPI, or ESI request to the CICS server using either an APPC Systems Network Architecture (SNA) connection or a TCP62 connection.

If the CICS region is at the V2.2 level, and the ECI is being used, then the request from the CICS client can be passed to CICS using TCP/IP directly (rather than using an SNA protocol).

# Using CTG with WebSphere on z/OS



- ▶ On z/OS the ECI is mapped onto the **EXCI** (External CICS Interface)
- ▶ The CTG native code (CTGJNI.so) invokes the EXCI directly



© IBM Corporation 2002

IBM Software Symposium 2002

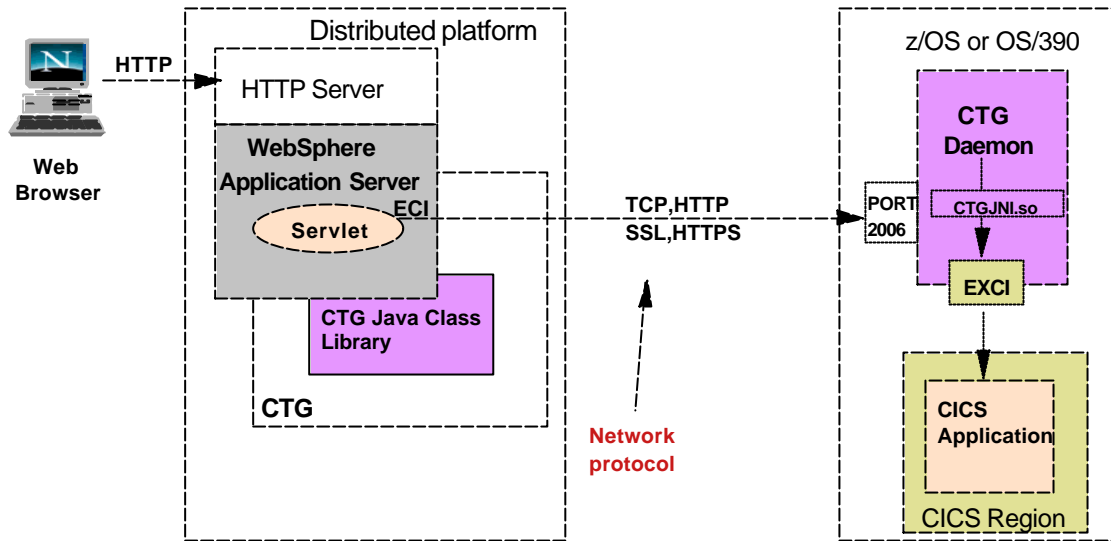
This chart shows the use of CTG with WebSphere on z/OS.

On z/OS, the servlet environment is provided by WebSphere Application Server for z/OS.

The CICS Transaction Gateway daemon is not usually required when running servlets within the WebSphere Application Server on z/OS since the CTG local: protocol can be used to directly invoke the native functions in the underlying EXCI. The EXCI passes the request onto the attached CICS region.

Since the CTG uses the CICS EXCI protocol, the CICS region and the Web Application server address space can be situated on any machine in a sysplex. However, the servlet is limited to using the subset of the ECI methods supported by the EXCI, cannot use the EPI or ESI, and must communicate with a CICS Transaction Server region at V1.2 or above.

# Using the CTG daemon on z/OS



- ▶ Servlet uses **network protocol** to communicate with CTG daemon
- ▶ CTG daemon native code (CTGJNI.so) uses EXCI to pass request to CICS



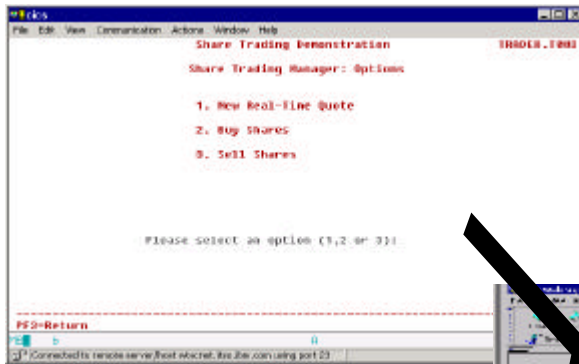
© IBM Corporation 2002

IBM Software Symposium 2002

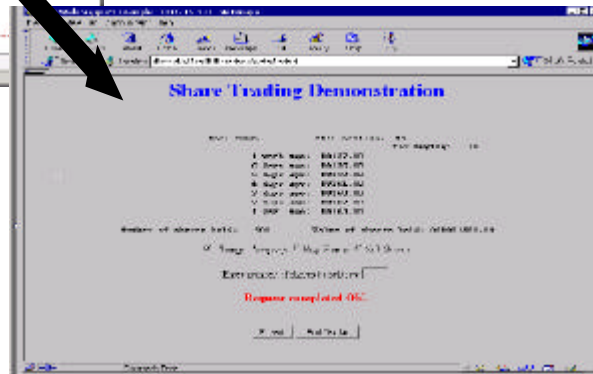
This chart shows the use of CTG on z/OS in conjunction with a WebSphere Application Server on another platform

This configuration's advantage is that a CTG network connection (TCP/IP, HTTP, SSL, or HTTPS) can be used from the Web server machine to the z/OS system, as opposed to an SNA connection. Also, the CICS Transaction Gateway daemon can exploit the scalability of S/390 by listening on multiple TCP/IP ports and balancing requests across multiple CICS regions. However, the same limitations apply to this scenario as when using a CTG entirely on OS/390. Namely, the servlet is limited to using the ECI classes (so it cannot use the EPI or ESI), and must communicate with a CICS TS region at V1.2 or above.

# Trader Application



TRADER is a sample application used to demonstrate CICS Web enablement



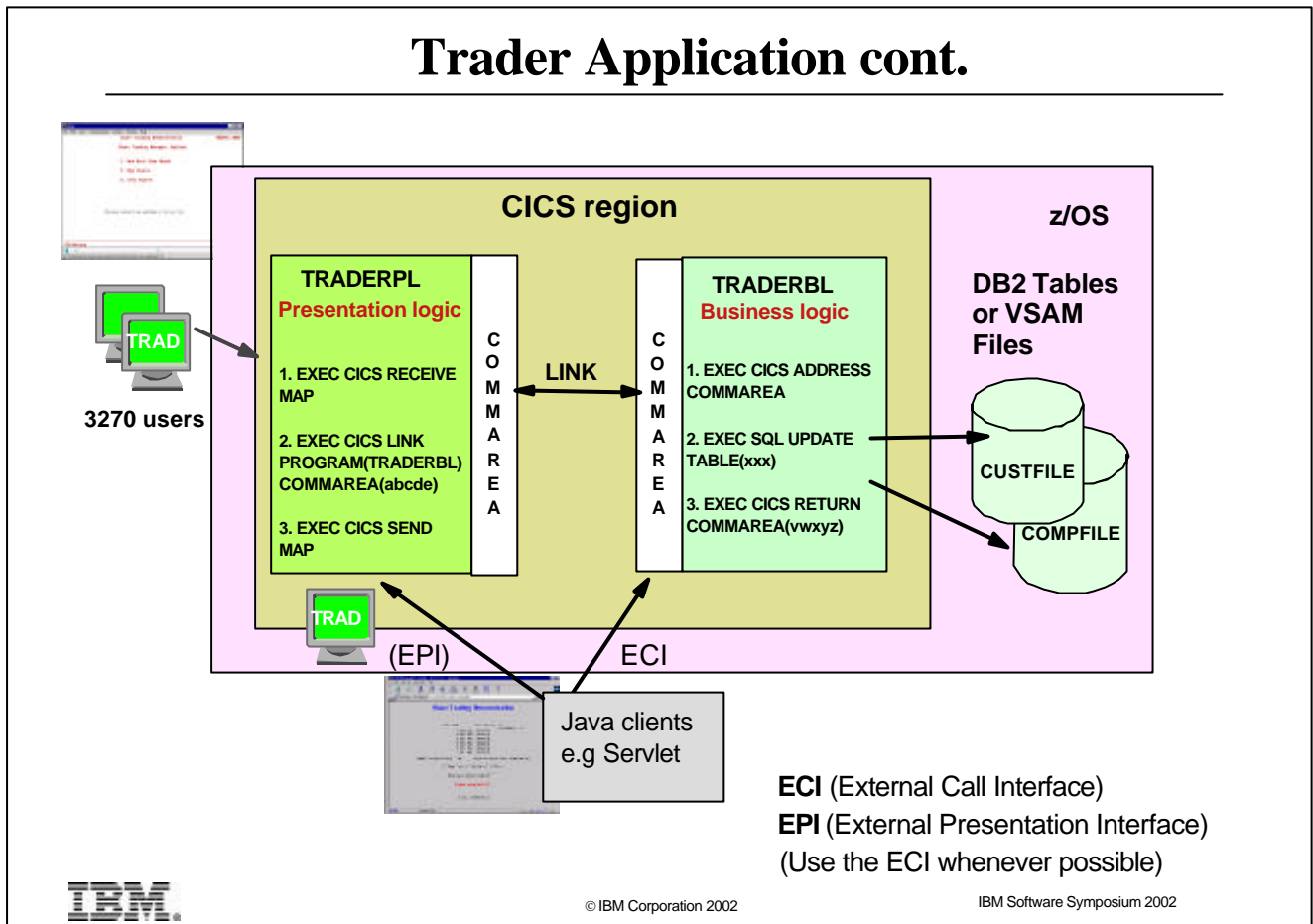
This chart introduces a sample application called TRADER which is referenced in many of the CICS Web Enablement redbooks.

Using TRADER as an example, the redbooks show how a new Web front-end can be added to an existing CICS application quite easily using new Java presentation logic and the CTG for connecting to the TRADER application.

The functions of TRADER are:

- ▶ Initial request sent
- ▶ Logon/Password page
- ▶ List of Companies to choose
- ▶ Quote/Buy/Sell
- ▶ Results

# Trader Application cont.



This chart shows that the Trader application is composed of two programs

- TRADERPL - presentation logic
- TRADERBL - business and data access logic

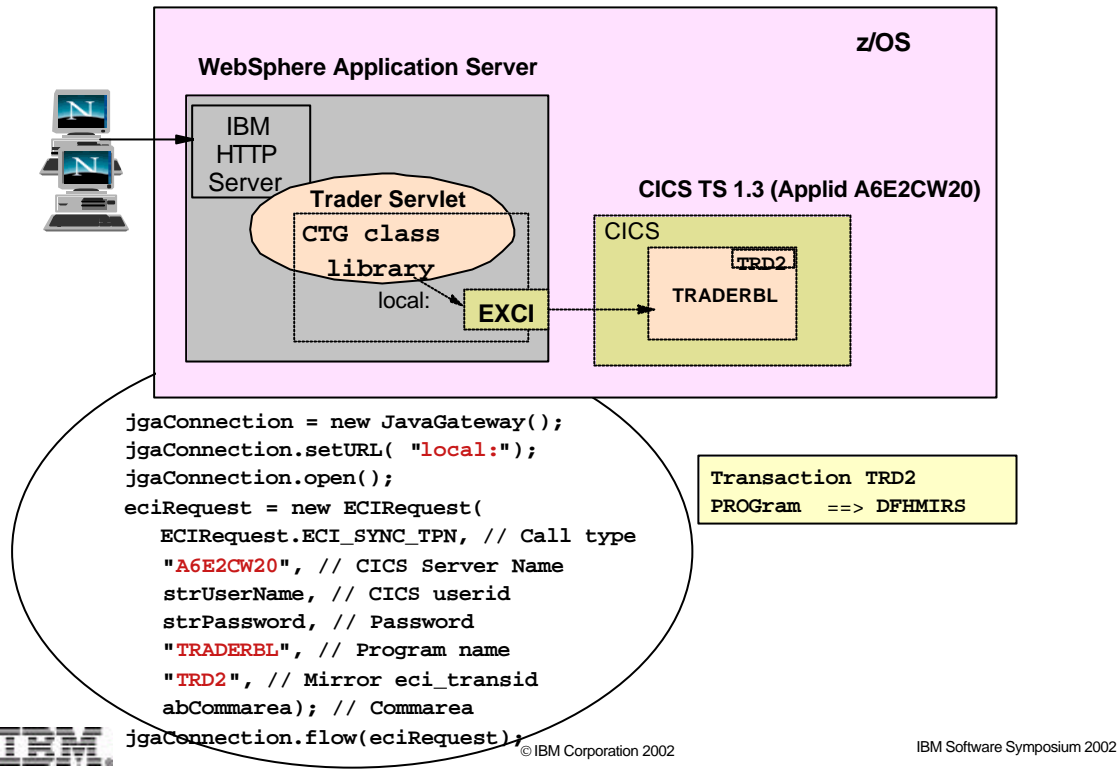
The correct approach to integrate the Trader application in a Web application is to access the business logic program (TRADERBL) from a Java client program such as a servlet.

The ECI can be used from the Java client program to call TRADERBL passing a COMMAREA.

The chart also shows how the EPI could be used to invoke the TRAD transaction (and program TRADERPL) directly. This is shown for illustration purpose only - you should always use the ECI in preference to the EPI. The EPI is useful where presentation logic and business logic are integrated, thus a COMMAREA interface cannot be used to drive the business logic

# Trader Servlet

http://<ip\_address>:port/servlet/Trader

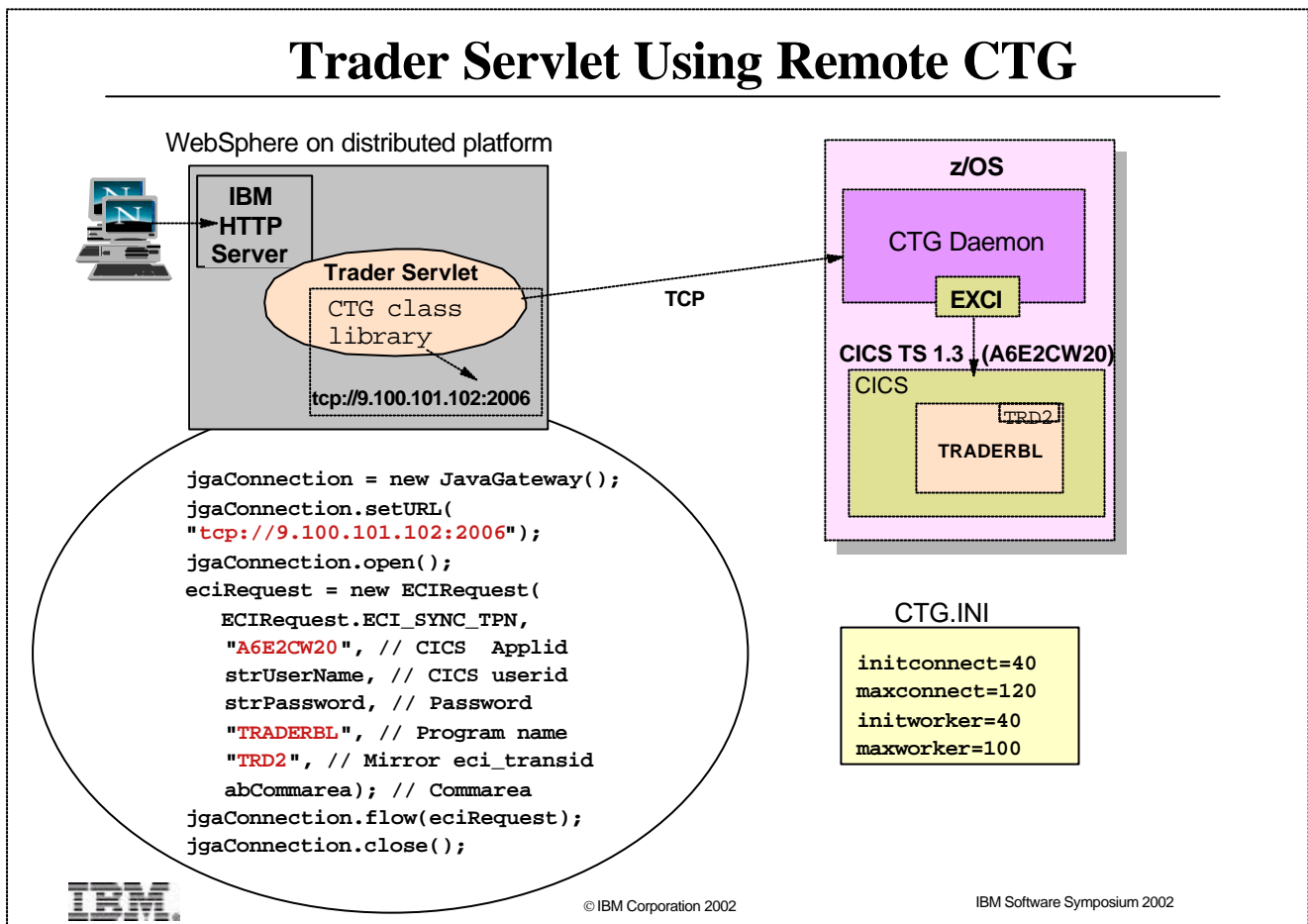


This Chart shows a Servlet (Trader) calling a CICS program using CTG classes:

- The JavaGateway and ECIRRequest are CTG classes
- Local mode specified - this means that CTG native code libCTGJNI.so is called directly
- TRADERBL (business logic program) is called
- The Trader servlet is new Web presentation logic
- An EXCI connection needs to be defined in CICS
- The ECIRRequest specifies the name of mirror transaction

Other parameters can be coded on ECIRRequest, for example, ECIRRequest.ECI\_NO\_EXTEND or ECIRRequest.ECI\_EXTEND to control the whether the call results in a single UOW or is one call in a series of calls which in aggregate constitute a UOW.

# Trader Servlet Using Remote CTG



This Chart shows

- The Trader servlet running on distributed platform
- The JavaGateway specifies the tcp address and port of the CTG daemon
- A TCP/IP connection established to CTG daemon
- The CTG daemon uses EXCI to pass the request onto CICS

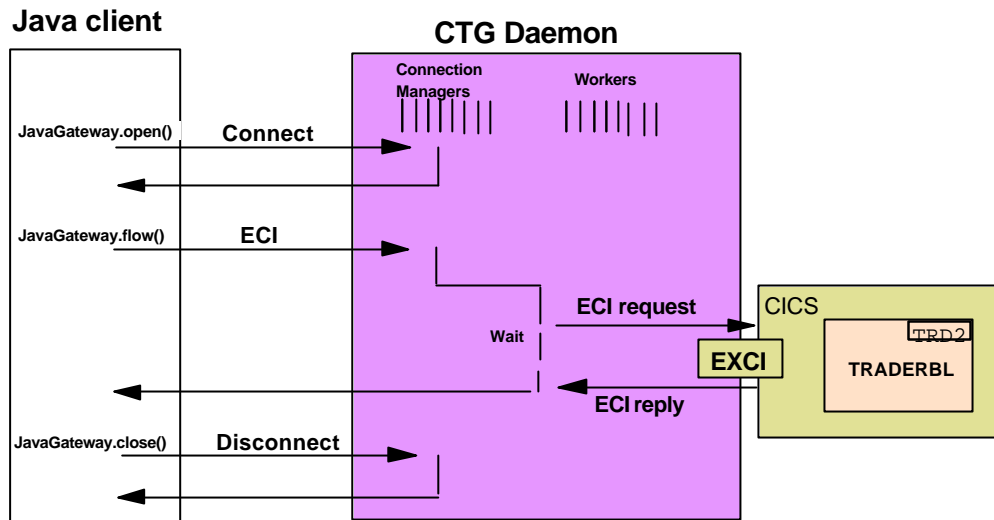
Like a Web server the CTG daemon has thread parameters

Connection manager (based on number of clients connecting)

Worker (these perform the work i.e connecting to CICS via EXCI)

Max is maximum, init are those created during CTG initialisation

# CTG/390 Threading Model



## Recommendations:

- `maxconnect` = maximum number of concurrent clients to be supported by this CTG daemon
- `maxworker`  $\leq$  100
  - ▶ Prevents errors from trying to exceed EXCI session limit



© IBM Corporation 2002

IBM Software Symposium 2002

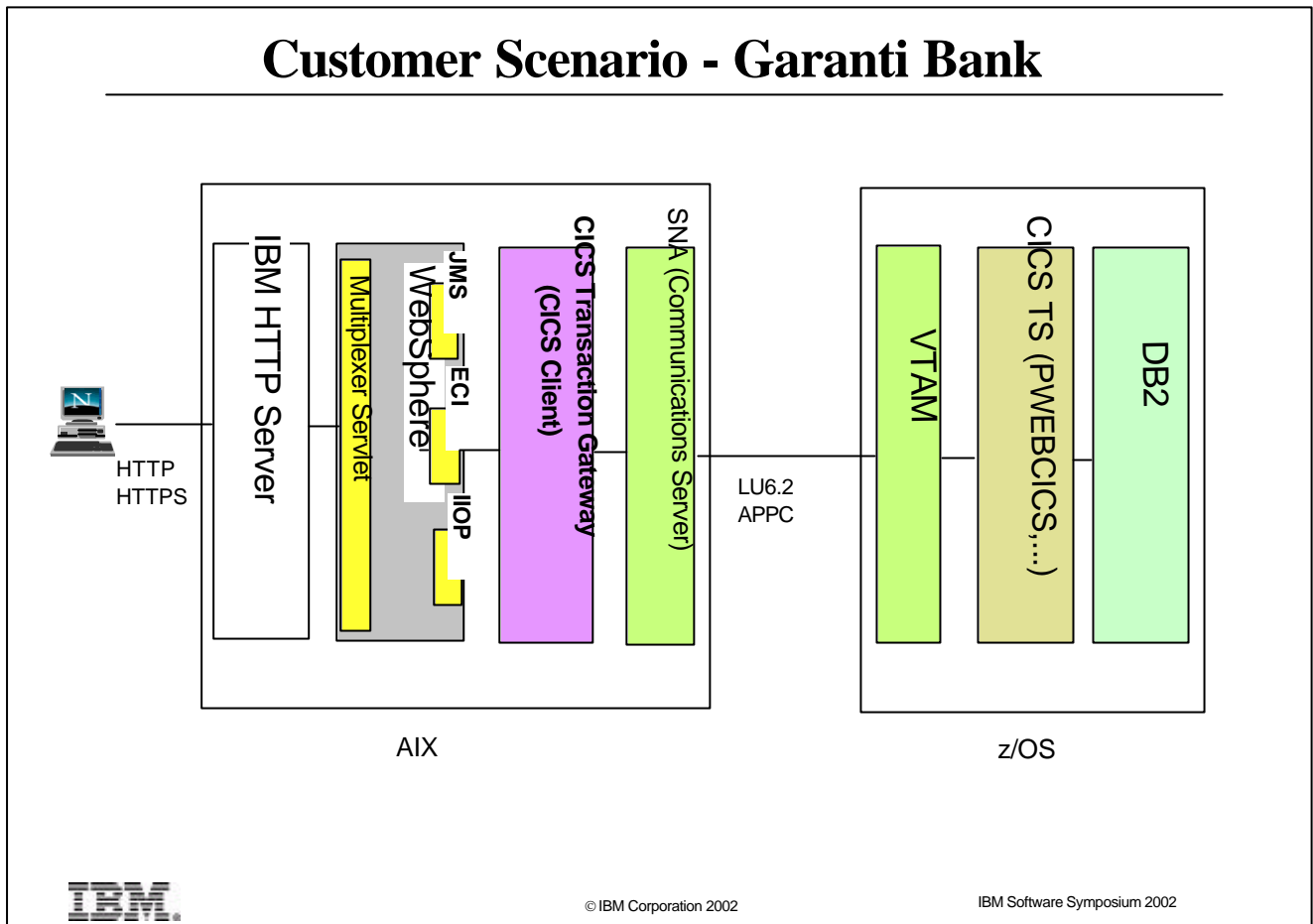
This Chart shows the different threads that are used by the CTG daemon:

- **Connection Manager threads**  
A Connection manager thread is in use for as long as client is connected
- **Worker threads**  
A Worker thread is in use for as long as EXCI request takes

A full description of the recommendations for defining the threading parameters (init, max and timeout parameters) can be found in the redbooks. In summary:

- `maxconnect` should be set to the maximum number of concurrent clients to be supported by this CTG daemon
- `maxworker` should be set  $\leq$  100 as this prevents errors from trying to exceed EXCI session limit

## Customer Scenario - Garanti Bank

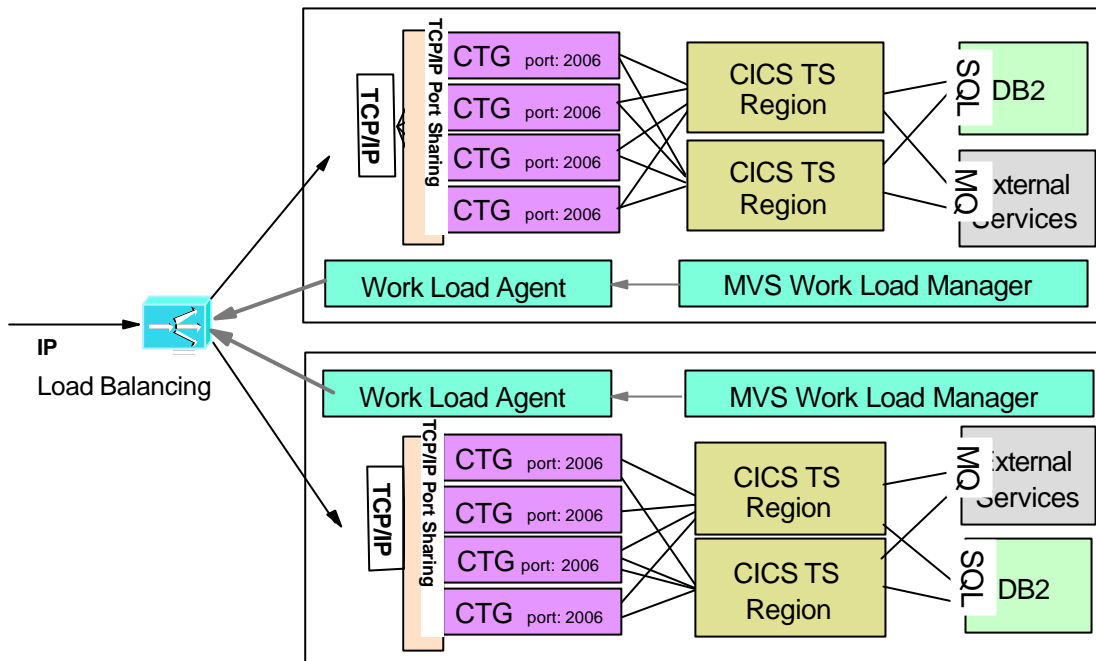


This chart was provided by one of the customers that we are currently working with in the Design Center. Garanti bank is one of the largest banks in Turkey.

Garanti have enabled intranet banking and insurance applications using WebSphere, CICS Transaction Gateway and CICS.

The chart shows the different components of their solution, which is a 3 tier solution based on a browser thin client, WebSphere on AIX and CICS on z/OS. The CTG for AIX is used for the communication between WebSphere and CICS.

## Customer Scenario - US Financial Institution



IBM

© IBM Corporation 2002

IBM Software Symposium 2002

This chart shows another configuration resulting from a Design Center project for a US financial institution.

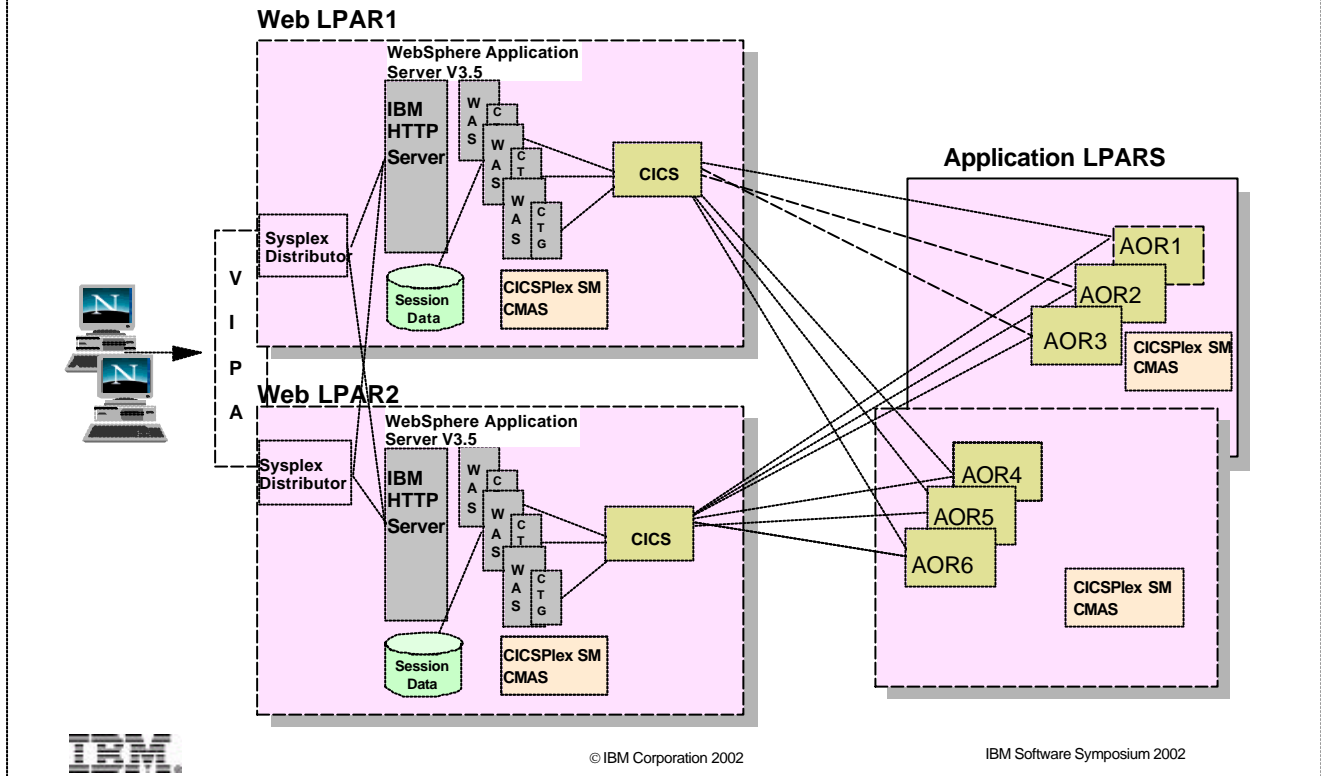
In this case:

- ▶
- ▶ The application is running on a distributed platform
- ▶ The CTG daemon is running on z/OS
- ▶ A load balancing solution (such as Network Dispatcher or Cisco's Mutil Node Load Balancer) is implemented external to the sysplex

In this project, a target transaction rate of more than 400 transactions per second was achieved.

# Customer Scenario - Luxembourg Government

See [www.ibm.com/eserver/zseries/zbulletin/pdf/issue30/cieproject.pdf](http://www.ibm.com/eserver/zseries/zbulletin/pdf/issue30/cieproject.pdf)



This chart shows the resulting configuration implemented by a European Government following a Design Center project to develop an e-business solution for accessing tax information on the Web.

As illustrated in the drawing, HTTP requests are received by specific Web logical partitions (LPARs). The business logic, encapsulated in CICS applications, runs in different application LPARs. Communication between Web LPARs and application LPARs is done using XCF, allowing a protocol switch from the TCP/IP incoming requests. All the partitions belong to the same sysplex in order to benefit from the Workload Manager (WLM).

A CICSplex is created with CICS "listener" regions running in the Web LPARs and CICS Application Owning Regions (AORs) running in the application LPARs. The listener regions receive servlet requests and dynamically route the requests to the AORs. CICSplex Systems Manager balances the workload using information from WLM.

The Sysplex Distributor, a function of the SecureWay® Communications Server for OS/390 V2R10 is used inside the Web partitions to dynamically distribute HTTP requests across the IBM HTTP Servers running in the Web LPARs. The distribution is done through WLM and the Quality of Service policy agent.

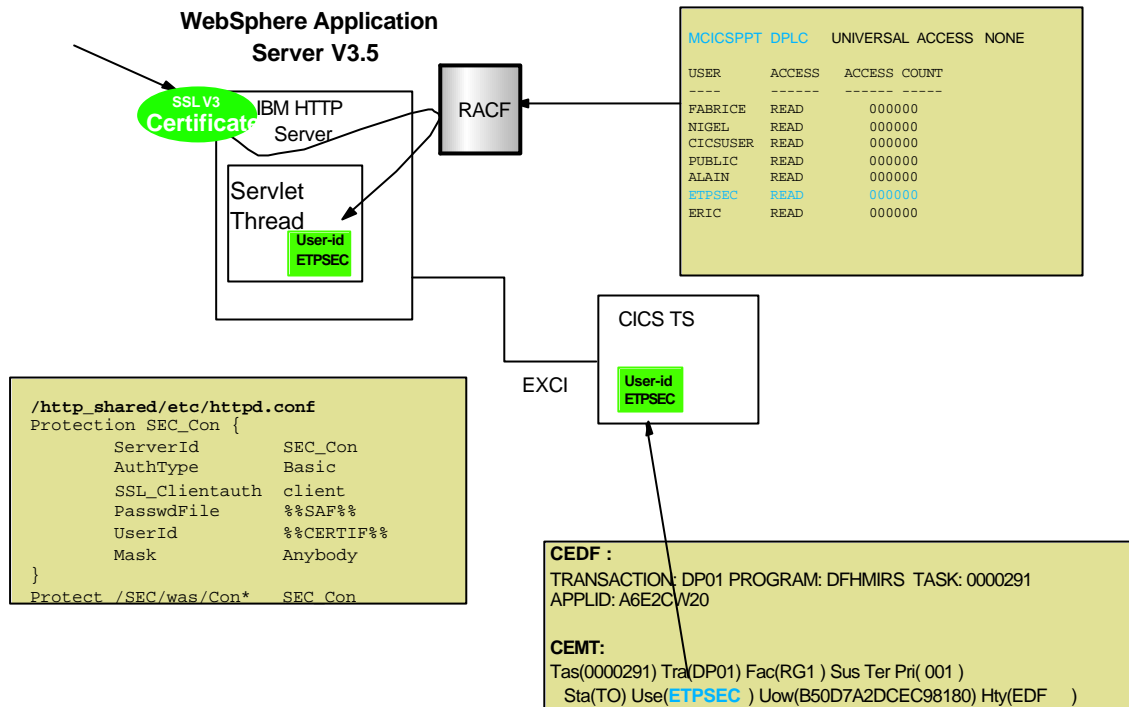
The HTTP Servers may run in 'scalable mode'. WLM can then manage the number of Queue Servers to satisfy the business goals assigned to the different applications.

This design provides the necessary degree of redundancy (multiple Web LPARs, multiple application LPARs, and multiple CICS regions) in order to satisfy the need for high availability.

Another important Web integration aspect for projects like this one is security, that is, how to ensure that the userids authenticated by the Web server are automatically propagated to the CICS regions. The reference article (see link below) deals with how a secure model for integration of WebSphere and CICS was created for this project.

Note: The Web location of this article is:  
[ibm.com/eserver/zseries/zbulletin/pdf/issue30/cieproject.pdf](http://ibm.com/eserver/zseries/zbulletin/pdf/issue30/cieproject.pdf)

# Customer Scenario - End to End Security



- One important aspect of many Design Center projects is end to end security.
- The chart shows how an authenticated userid can be automatically delegated to CICS when using the IBM HTTP Server and WebSphere Application Server V3.5 on z/OS.
- Note that propagation of the authenticated userid changes considerably when migrating to WebSphere Application Server V4.01 for z/OS. The process is also very different for WebSphere on the distributed platforms.

# CICS Transaction Gateway V5

---

- Generally available July 2002

Enhancements include:

- Support for the J2EE Connector Architecture
  - First introduced in CTG V4.01 (dist. platforms) and CTG V4.02 (for use with WebSphere for z/OS)
  - Support for J2EE Asynchronous ECI calls on z/OS
- Operational improvements e.g.
  - Dynamic control of tracing
  - ARM restart for CTG/390
- JSSE support (Java Secure Sockets Extension) for 128-bit encryption
- Enhanced logging
  - No default logging of client connections
  - Logging of EXCI return codes on z/OS
- Enhanced TCP62 support
  - Including support for TCP62 on the Linux for S/390 platform



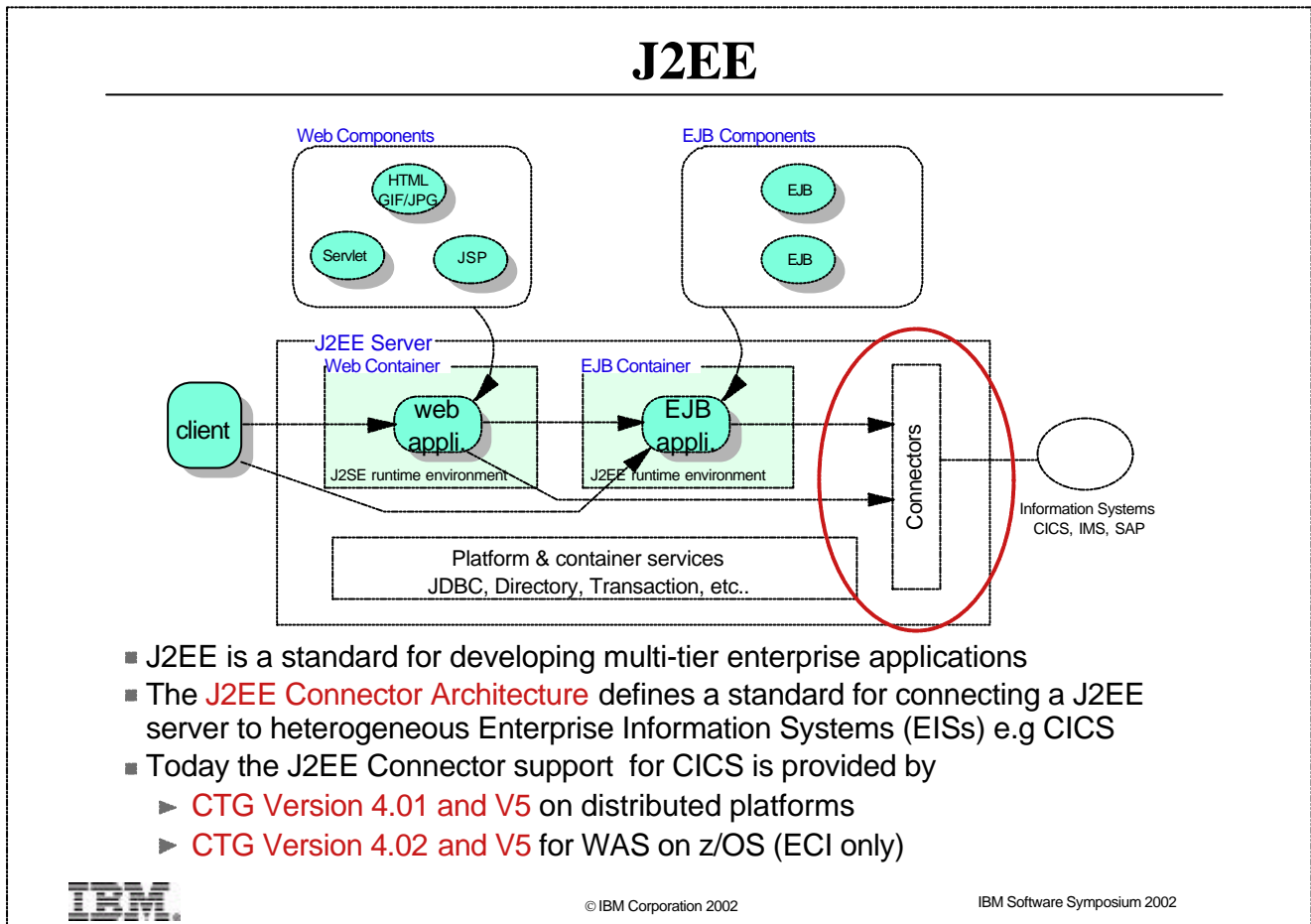
© IBM Corporation 2002

IBM Software Symposium 2002

CTG V5 became generally available in July, 2002.

Enhancements include:

- Support for the J2EE Connector Architecture
  - First introduced in CTG V4.01 (dist. platforms) and CTG V4.02 (for use with WebSphere for z/OS)
- Operational improvements e.g.
  - Dynamic control of tracing (java -jar ctgadmin.jar -ctg=tcp://myctg.com:2810 -a=setgwtrace -tlevel=4 -tfile=c:\traces\ctg.trc)
  - ARM restart for CTG/390
- JSSE support (Java Secure Sockets Extension) for 128-bit encryption
- Enhanced logging (ability to split console output stdout and stderr messages and so you can redirect appropriately)
  - No default logging of client connections
  - Logging of EXCI return codes on z/OS
- Enhanced TCP62 support - including support for TCP62 on the Linux for S/390 platform



The Java 2 Platform, Enterprise Edition (J2EE) is a JavaSoft standard for developing multi-tier enterprise applications. J2EE simplifies enterprise applications by basing them on standardized, modular components. The J2EE spec requires servers to support a specific set of protocols and Java enterprise extensions, including JDBC 2.0, Enterprise JavaBeans 1.1, Java Servlets 2.2, Java Server Pages 1.1. J2EE applications are portable across J2EE servers.

The J2EE Connector Architecture defines a standard architecture for connecting the Java 2 Platform Enterprise Edition (J2EE) platform to heterogeneous Enterprise Information Systems (EISs). Examples of EISs include transaction processing systems such as CICS Transaction Server, and Enterprise Resource Planning systems such as SAP

An application server vendor only needs to extend its system once to support the J2EE Connector Architecture and is then assured of connectivity to multiple EISs. Likewise, an EIS vendor provides a standard resource adapter (see next chart) and it has the capability to plug in to any application server that supports the J2EE Connector Architecture.

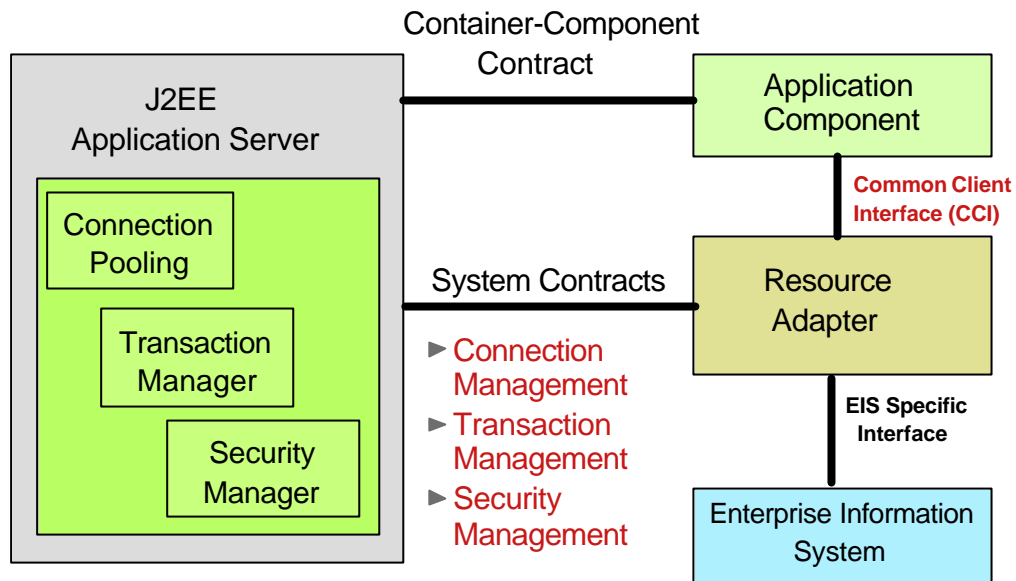
Note: The complete J2EE Connector Architecture specification defined by the Java Community Process can be downloaded at <http://java.sun.com/j2ee/download.html#connectorspec>

The J2EE Connector Architecture is a part of the J2EE 1.3 specification - although it is supported by WAS V4 which is a J2EE 1.2 compliant server.

**CTG V5 and CTG V4.01 provides J2EE connector support for the distributed platforms**

**CTG V5 and CTG V4.02 for z/OS provides J2EE connector support for use with WAS V4.01**

# J2EE Connector Architecture



© IBM Corporation 2002

IBM Software Symposium 2002

Version 1.0 of the J2EE Connector Architecture defines a number of components that make up this architecture:

## **Common Client Interface (CCI)**

The CCI defines a common API for interacting with resource adapters. It is independent of a specific EIS. A Java developer communicates to the resource adapter using this API. See next chart.

## **System contracts**

The Connector architecture defines a standard set of system-level contracts between a J2EE server and a resource adapter. The standard contracts include:

- A **connection management contract** that lets a J2EE server pool connections to an underlying EIS, and lets application components connect to an EIS. This leads to a scalable application environment that can support a large number of clients requiring access to EIS systems.
- A **transaction management contract** between the transaction manager and an EIS that supports transactional access to EIS resource managers. This contract lets a J2EE server use a transaction manager to manage transactions across multiple resource managers. This contract also supports transactions that are managed internal to an EIS resource manager without the necessity of involving an external transaction manager.
- A **security contract** that enables secure access to an EIS. This contract provides support for a secure application environment, which reduces security threats to the EIS and protects valuable information resources managed by the EIS.

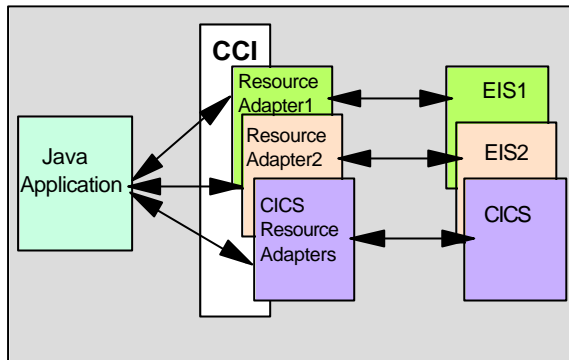
These system contracts are transparent to the application developer, meaning they do not have to implement these services themselves.

## **Resource adapter deployment and packaging**

The connector architecture enables an EIS vendor to provide a standard resource adapter for its EIS. A resource adapter is a middle tier between a Java application and an EIS, which permits the Java application to connect to the EIS.

A resource adapter provider develops a set of Java interfaces/classes as part of its implementation of a resource adapter.

# Using the CCI with the CICS resource adapters



## CCI pseudo code

```
ConnectionFactory cf =<lookup from JNDI namespace>
Connection connection =cf.getConnection();
Interaction interaction =connection.createInteraction();
interaction.execute(<input and output data>);
interaction.close();
connection.close();
```

- ▶ CCI provides a standard client API to interface with resource adapters
- ▶ CTG provides CICS resource adapters for the **ECI** and **EPI** interfaces
- ▶ CICS resource adapters supplied in **cicseci.rar** and **cicsepi.rar** files (**cicseciRRS.rar** for use with WebSphere on z/OS)
- ▶ CCI strategically replaces IBM's Common Connector Framework (CCF)



© IBM Corporation 2002

IBM Software Symposium 2002

The **Common Client Interface (CCI)** defines a standard client API for application components to access multiple resource adapters. This API can be used directly, or enterprise application integration frameworks can be used to generate EIS access code for the developer. The CCI is designed to be an EIS independent API, such that an enterprise application development tool can produce code for any J2EE compliant resource adapter that implements the CCI interface. Such tools include the VisualAge for Java Enterprise Access Builder, and WebSphere Studio Application Developer Integration Edition.

The CCI has the following characteristics:

- ▶ It is independent of a specific EIS. It forms a base level API for EIS access on which higher level functionality specific to an EIS can be built.
- ▶ It provides an API that is consistent with other APIs in the J2EE platform, such as JDBC.
- ▶ It is targeted primarily towards application development tools and enterprise application integration frameworks, rather than Java developers using the CCI API directly.

**The CCI strategically replaces the CCF (Common Connector Framework), and should be used for all new applications.**

The Java interfaces/classes provided in a resource adapter are packaged together with a deployment descriptor to create a Resource Adapter Archive (represented by a file with an extension of rar ). This Resource Adapter Module is used to deploy the resource adapter into the application server.

The CTG provides CICS resource adapters for ECI and EPI interfaces  
CICS resource adapters supplied in **cicseci.rar** and **cicsepi.rar** files  
(**cicseciRRS.rar** for use with WebSphere on z/OS) .

## CCI used to connect to TRADERBL

```
//Create and set values for ECI managed connection factory
ECIManagedConnectionFactory mcf=new ECIManagedConnectionFactory();
mcf.setConnectionURL("tcp://9.100.101.102");
mcf.setPortNumber("2006");
mcf.setServerName("A6E2CW20");
//Create a connection factory connection object
ConnectionFactory cxnf=(ConnectionFactory)mcf.createConnectionFactory();
Connection cxn=cxnf.getConnection();
//create an interaction with CICS to start program TRADERBL
Interaction ixn=cxn.createInteraction();
ECIInteractionSpec ixnSpec=new ECIInteractionSpec();
ixnSpec.setInteractionVerb(ixnSpec.SYNC_SEND_RECEIVE);
ixnSpec.setFunctionName("TRADERBL");
//Create a new record for handling the COMMAREA byte array
GenericRecord record =new GenericRecord(("abcde").getBytes("IBM037"));
//Finally execute and flow the request to CICS
ixn.execute(ixnSpec,record,record);
//Close the interaction and the connection
ixn.close();
cxn.close();
```



The chart shows an example use of the CCI to connect to the CICS program TRADERBL.

The following parameters of the ECIManagedConnectionFactory are set:

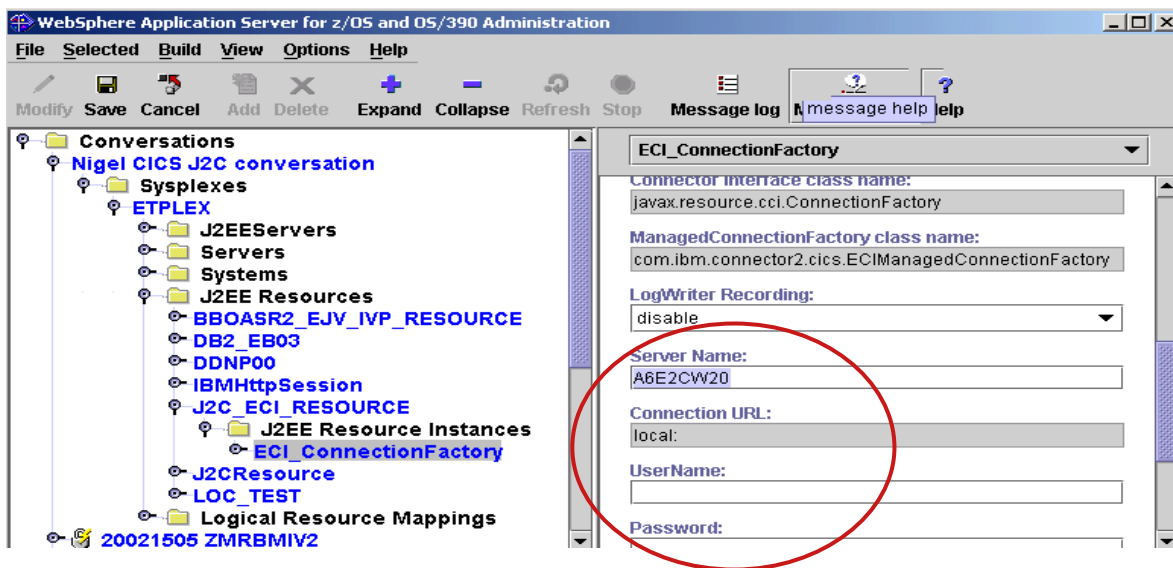
- ▶ **tcp://9.100.101.102** to be used as the Connection URL (note that the only option for this setting when deploying in WAS for z/OS is to use **local:** )
- ▶ 2006 for the port number (this is ignored if the protocol is local)
- ▶ **A6E2CW20** as the CICS server name -

In addition, the **TRADERBL** program is specified as the function name on the ECIInteractionSpec.

Note that, in this example, the connection factory is not retrieved from a JNDI name server, but is defined manually using the **ManagedConnectionFactory** class. This is done for illustrative purposes only - in a managed environment, the connection factory should be looked up from the JNDI namespace.

We will see how the parameters of the managed connection factory can be defined in the JNDI namespace for WebSphere on z/OS in the next chart.

# Deploying TraderBean to WebSphere for z/OS



- ▶ When deploying on WebSphere for z/OS, the Systems Management User Interface (SMUI) is used to:
  - ▶ Define the CICS ECI resource adapter
  - ▶ Create new resource instances
  - ▶ Set properties of the resource instance e.g CICS server name



The chart shows a screenshot from the WebSphere for z/OS Systems Management User Interface (SMUI).

The SMUI is used to:

- ▶ Define the CICS ECI resource adapter
- ▶ Create new resource instances
- ▶ Set properties of the resource instance e.g CICS server name

Applications then perform a JNDI lookup (based on the name of the J2EE resource name) to retrieve the necessary connectivity information about the CICS region that is to be accessed.

# Application Development Tools

- J2EE Connection Architecture spec states that AD tools will be the primary way to work with the CCI (rather than hand-coding CCI code)
- IBM offers two J2EE CA AD tools:
  - ▶ **VisualAge for Java Enterprise Edition**
    - ▶ Uses the Enterprise Access Builder component
    - ▶ Creates Record and Command beans
  - ▶ **WebSphere Studio Application Developer Integration Edition**
    - ▶ Part of the new WebSphere Studio family
    - ▶ Wizards to creates Enterprise Service which can be deployed as a session bean or Web service
    - ▶ Ships with the CICS ECI resource adaptor (non z/OS version)

**Service level 4 for WebSphere Application Server Advanced Edition V4 and WebSphere V4 for z/OS enables deployment of enterprise services created by WSAD**



© IBM Corporation 2002

IBM Software Symposium 2002

CCI applications are intended to be developed using tooling. IBM provides a set of development tools to help develop CCI applications:

## **VisualAge for Java Enterprise Edition**

This tool provides a component called the Enterprise Access Builder which can automatically generate Common Client Interface code. This code is stored within a command bean. The command bean is built using a set of wizards - no Java programming is required.

## **WebSphere Studio**

Certain configurations of this tool contain a full J2EE development environment, allowing the development and test of session beans in a WebSphere environment.

The following two configurations of WebSphere Studio can be used to develop J2EE enterprise applications:

WebSphere Studio Application Developer

WebSphere Studio Application Developer Integration Edition

Both of these configurations provide tools to generate session beans, and to test these beans in a powerful WebSphere Application Server test environment. For our purposes, the difference between these two configurations lies in the quality of the test environment:

## **WebSphere Studio Application Developer**

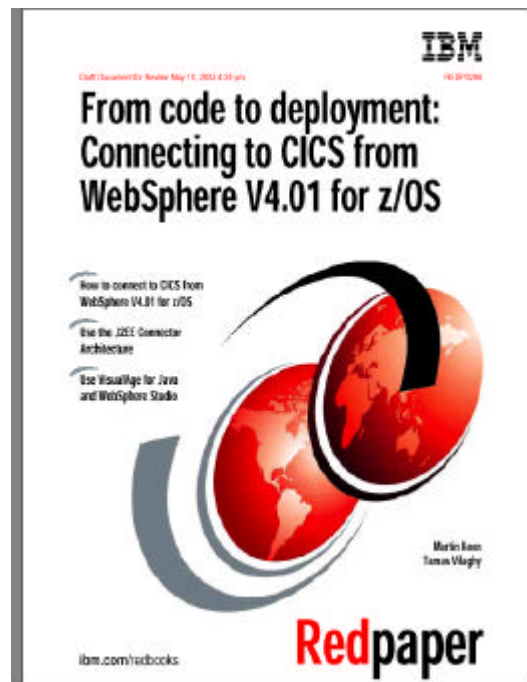
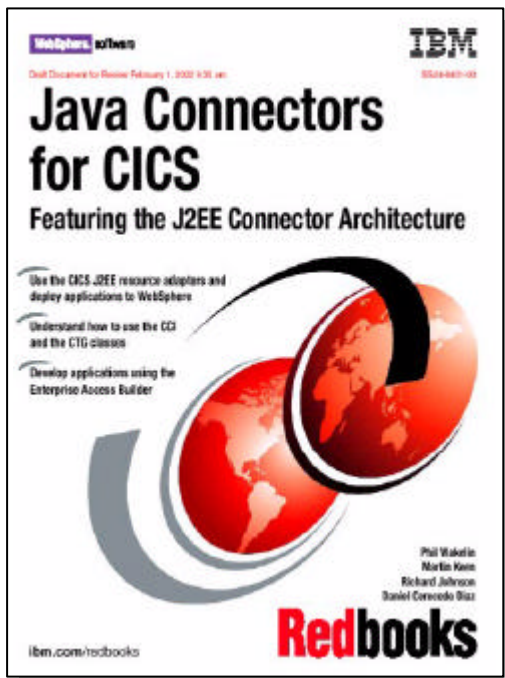
This configuration does not contain a J2EE Connector Architecture compliant test environment. Therefore the CICS ECI resource adapter classes must be imported into the WebSphere test environment, and the connections to the resource adapter will not be managed by the test environment. This is a non-managed environment.

## **WebSphere Studio Application Developer Integration Edition**

This configuration contains a J2EE Connector Architecture compliant test environment, and ships with the CICS ECI resource adapter (the non z/OS version) already installed into the WebSphere test environment. Connections to the resource adapter are managed by the test environment. This is a managed environment.

**Important:** Prior to WebSphere V4 Service Level 4, code that is generated using the Enterprise Services feature of WSAD-IE (which generates CCI code to connect to CICS) can only be deployed in WAS EE and not WAS AE or WAS for z/OS.

# New redbook and redpaper



© IBM Corporation 2002

IBM Software Symposium 2002

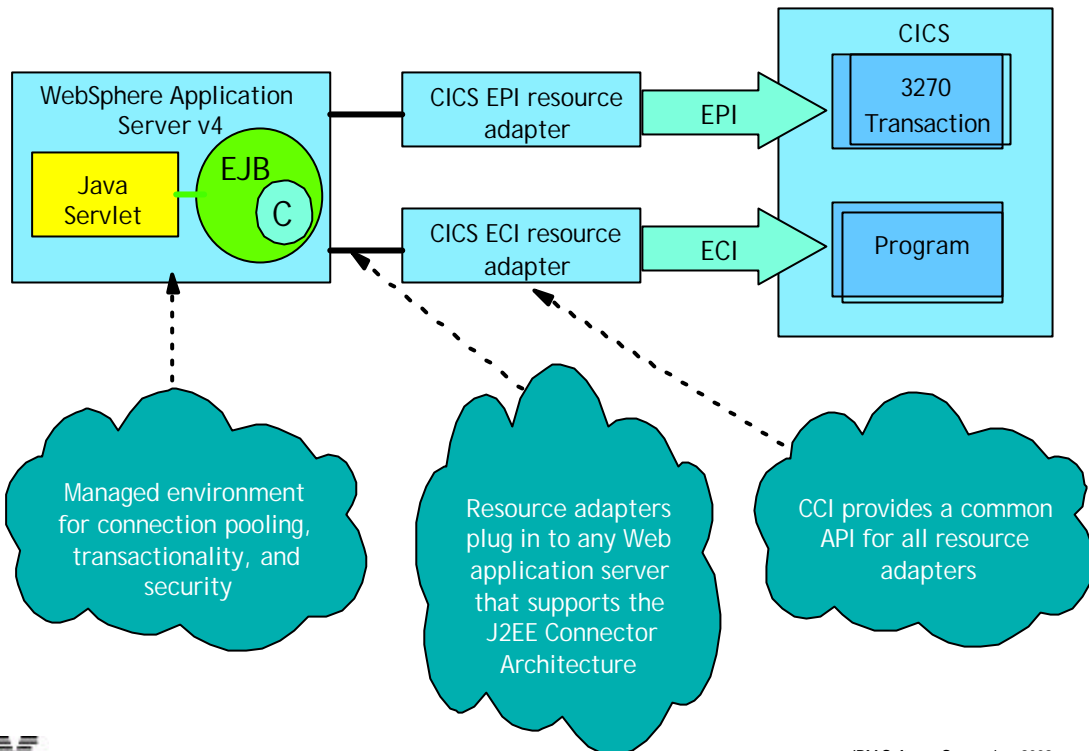
The J2EE connector support provided by the CTG is explained in detail in a new redbook called 'Java Connectors for CICS Featuring the J2EE Connector Architecture' (SG24-6401)

You can download this, and other redbooks, at <http://www.redbooks.ibm.com>

Contents of the redbook:

- Part 1 - Introduction
  - Chapter 1 - Java connectors for CICS
  - Chapter 2 - CICS Transaction Gateway
  - Chapter 3 - CICS and the J2EE Connector Architecture
- Part 2 - Connecting to COMMAREA based CICS programs
  - Chapter 4 - ECI and ESI applications
  - Chapter 5 - CCI applications -- ECI based
  - Chapter 6 - CCI applications -- managed environment
- Part 3 - Connecting to 3270 based CICS transactions
  - Chapter 7 - EPI support classes
  - Chapter 8 - CCI applications -- EPI based
- Part 4 - Appendices
  - Appendix A - Sample CICS programs
  - Appendix B - Configuring the CICS connectors in VisualAge for Java
  - Appendix C - Data conversion
  - Appendix D - Additional material

# Why use the J2EE Connector Architecture with CICS ?



IBM

© IBM Corporation 2002

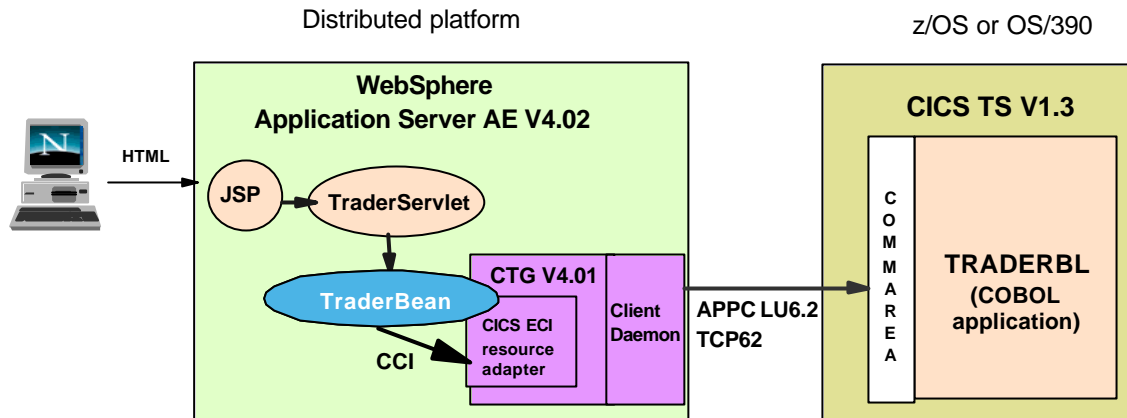
IBM Software Symposium 2002

The J2EE Connector Architecture offers an attractive way of accessing CICS resources:

- ▶ A managed environment can greatly simplify development, freeing developers from having to spend a great deal of time dealing with transactionality, security context, and connection pooling.
- ▶ The CICS ECI and EPI resource adapters plug in to any Web application server that supports the J2EE Connector Architecture. Web application servers can be selected based upon the quality of managed environment they provide.
- ▶ The CCI provides a common API for developers to connect to any enterprise system. This is useful for a developer coding to multiple enterprise systems (as there is only one API to learn).

**Note:** Although the benefits of using the J2EE connection architecture are more significant when using EJBs, we should also consider the use of CCI from servlets rather than the base APIs. CCI is J2EE aware and the base APIs are not.

# Trader EJB using CCI



- WebSphere provides a managed environment (connections, transactionality and security)
  - ▶ Connection pooling support provided by WebSphere - connection object should not span transaction boundary
  - ▶ CICS ECI adapter supports **local transactions** (and '**Last Participant Support**') - LPS not currently supported by WebSphere V4
  - ▶ Component managed signon is supported (requires userid and password to be set on ConnectionSpec or ManagedConnectionFactory and verified by CICS )



© IBM Corporation 2002

IBM Software Symposium 2002

The chart shows how the TRADER application can be implemented using an Enterprise JavaBean (TraderBean) and the J2EE connector support in CTG V5 to connect to the TRADER business logic program TRADERBL. The basic flow of the application is as follows:

1. The end user presses a button on a Web page that submits a form to the servlet.
2. The servlet receives the request for action and calls an appropriate method on the remote interface of the Trader session bean.
3. The session bean connects uses the J2EE CCI to call the CICS program TRADERBL, using the facilities of the CTG and the CICS resource adapter.
4. The session bean returns any output data from TRADERBL back to the servlet in the form of a data bean object.
5. The servlet forwards the request to a JSP which displays the contents of the data bean to the end user.

The J2EE Connector Architecture support provided by WebSphere Application Server Advanced Edition V4 provides the support for

## Connection Management

Connection pooling is implemented, allowing the reuse of CICS Transaction Gateway connections. Connection pool settings are specified in the Advanced panel of the connection factory in the Administrative Console.

Support is provided for the get/use/close programming model. This model dictates that a connection object to a resource adapter is retrieved, used, and closed with a transaction. The connection object should not exceed transaction boundaries. The get/use/cache programming model, which caches connections across transaction boundaries, is not currently supported. In Local mode there is little overhead for this type of get/use/close programming model.

## Transaction Management

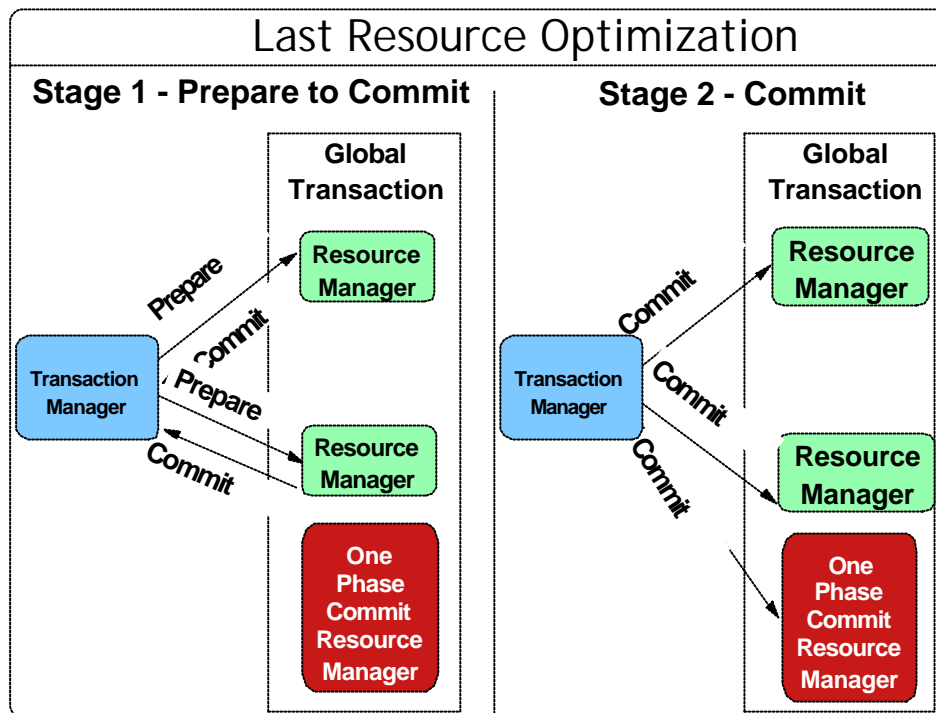
Support is provided by WebSphere Application Server for NoTransaction, Local transaction, and global transaction. The CICS ECI resource adapter supports **Local transaction**. The CICS EPI resource adapter supports **NoTransaction**.

Last Participant Support is supported by the CICS ECI resource adapter but it is not currently supported in WebSphere. If a future release of WebSphere has support for Last Participant Support then, in some circumstances (i.e. when there is only one resource manager in the global transaction that only support local transactions), it will be possible to coordinate the TRADERBL updates with other updates made by TraderBean.

## Security Management

Only component managed signon is currently supported. This requires the component to pass in userid and password credentials through the ConnectionFactory to CICS.

# What's Last Participant Support ?



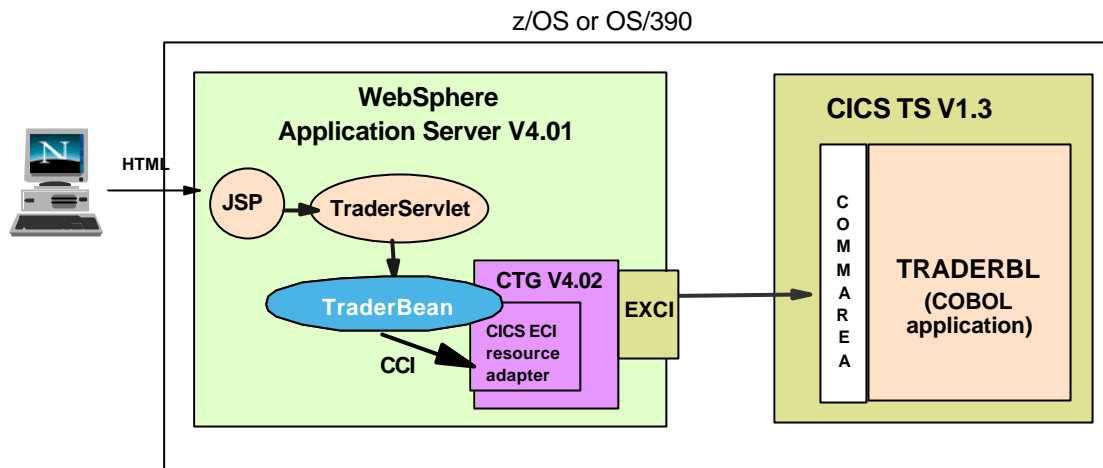
IBM

© IBM Corporation 2002

IBM Software Symposium 2002

- Last Participant Support allows a resource adapter that does not implement the XAResource interface to participate in a global transaction if it is the only resource manager (which does not implement the XAResource interface) included in the global transaction.

# CCI applications deployed in WebSphere for z/OS



- ▶ Connection pooling support provided by WebSphere
- ▶ **CTG/390 CICS ECI adapter supports two-phase commit processing**
  - ▶ Uses Resource Recovery Services (RRS)
  - ▶ WebSphere J2EE server and CICS must run in same LPAR and use **local:** protocol
  - ▶ CTG V5 (or V4.02) and WebSphere V4.01 with APAR PQ55873 required
- ▶ Container managed signon supported - pass security credentials to CICS based on **RunAS** deployment descriptor specified for TraderBean methods (**Caller, Server or Role**)



© IBM Corporation 2002

IBM Software Symposium 2002

The chart shows how the TRADER EJB application using the J2EE connector support can be deployed to WebSphere for z/OS.

The J2EE Connector Architecture support provided by WebSphere Application Server for z/OS provides the support for:

## Connection Management

Sysplex-wide pooling of connections is provided.

## Transaction Management

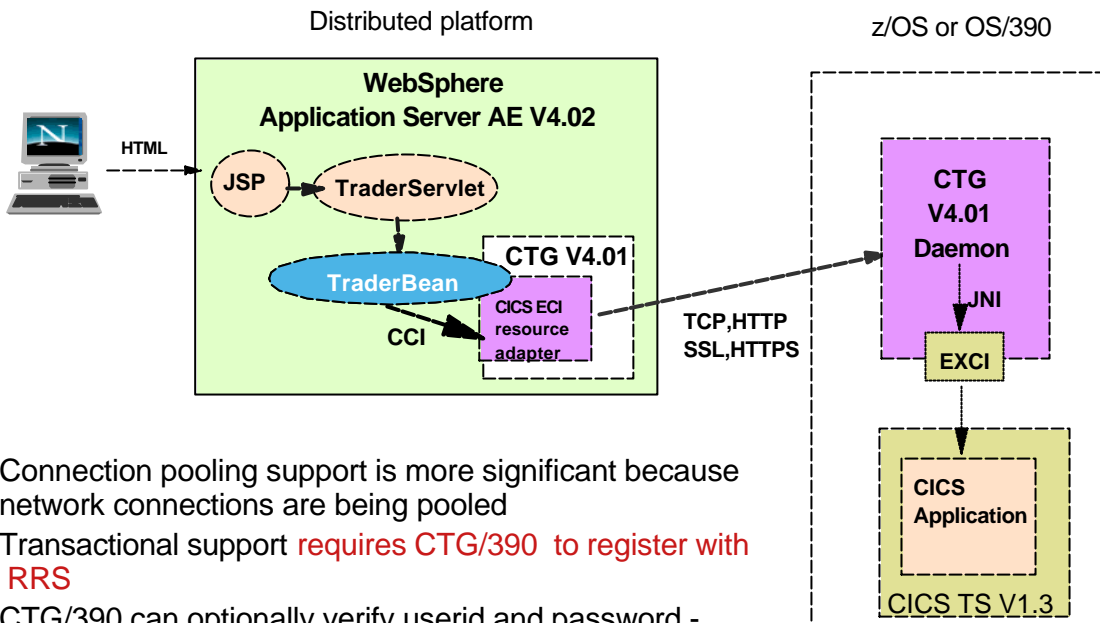
**Important:** This model provides support for two phase commit processing using RRS

## Security Management

The security credentials flowed from WAS to CICS are based on deployment information specified in the enterprise application. The userid you flow to CICS can come from:

- The enterprise application
- The userid of the J2EE server running the enterprise application
- The userid of the caller of the enterprise application
- A userid mapped to an EJB security role

# Trader EJB using CCI to connect to CTG/390



- ▶ Connection pooling support is more significant because network connections are being pooled
- ▶ Transactional support **requires CTG/390 to register with RRS**
- ▶ CTG/390 can optionally verify userid and password - only the userid is passed to CICS



© IBM Corporation 2002

IBM Software Symposium 2002

This chart shows the use of CTG on z/OS in conjunction with a CCI application running in WebSphere Application Server on another platform.

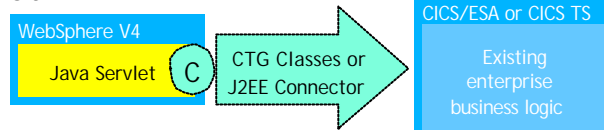
The Design Center is working with several customers who are prototyping this model.

As in the case of using the CTG classes directly, this configuration's advantage is that a CTG network connection (TCP/IP, HTTP, SSL, or HTTPS) can be used from the Web server machine to the z/OS system and the CICS Transaction Gateway daemon can exploit the scalability of S/390 by listening on multiple TCP/IP ports and balancing requests across multiple CICS regions.

If the TraderBean is deployed with a transactional deployment descriptor of REQUIRED, then the resulting ECIRRequest to CICS will use Extended UOW mode. In this case, the CTG and CICS need to be configured to register with RRS.

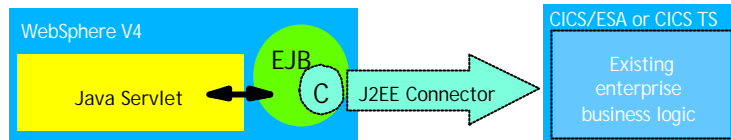
# Conclusion - main ways to use the CTG

## From servlet



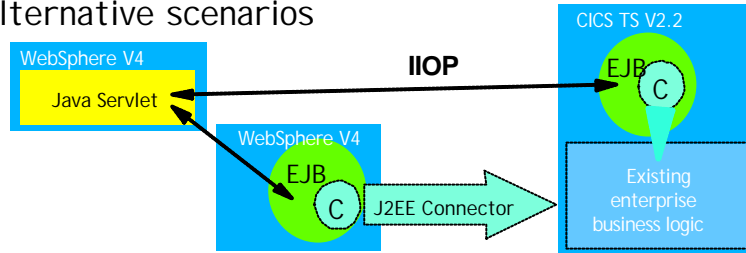
- ▶ J2EE Connector should also be considered for servlets

## From EJB



- ▶ J2EE Connector fits naturally with EJB model

## Alternative scenarios



- ▶ EJBs can also be deployed in CICS
- ▶ Connectors can be used **inside** CICS



This chart summarises the main ways that the CTG can be used to connect WebSphere to CICS.

- ▶ The most common approach today is to use the CTG classes (or CCF), most likely the ECI, from a servlet. For migration reasons, you should consider using the J2EE Connector support for new servlet applications.
- ▶ The most significant benefits of the J2EE Connector support come from using the connector from an EJB i.e transaction management, connection pooling and security management.
- ▶ Alternatively, you can deploy EJBs in CICS and use IIOP to access the EJBs from servlets. The EJB architecture provides the best technology for calling business logic in CICS. Today, an EJB deployed in CICS can use the CTG and CCF for accessing CICS procedural programs. Future support is planned to provide support for using the J2EE connector inside CICS.

# Resources

---

- Sources of planning information:
  - ▶ **SG24-6133-01 CICS Transaction Gateway V5, The WebSphere Connector for CICS**
  - ▶ SG24-5275 Java Application Development for CICS
  - ▶ SG24-5277 Revealed! CICS Transaction Gateway with more CICS Clients Unmasked
  - ▶ **SG24-5466 Revealed! Architecting Web Access to CICS**
  - ▶ SG24-5748 A Performance Study of Web Enabling CICS
  - ▶ SG24-5756 Securing Web Access to CICS
  - ▶ SG24-6118 Workload Managing Web Access to CICS
  - ▶ **SG24-6401 Java Connectors for CICS Featuring the J2EE Connector Architecture**
  - ▶ REDP0206 From code to deployment: Connecting to CICS from WebSphere V4.01 for z/OS
  - ▶ Designing & Programming CICS Applications, O'Reilly ISBN 1-56592-676-5

Find out more about these redbooks at <http://www.redbooks.ibm.com/>



# IBM Design Center for e-business infrastructure

*Discover how the Design Center has helped customers to implement e-business solutions using CICS*

## Connecting WebSphere to CICS

- Which connector ?
- Development tools
- Security
- Workload Management

Centre Informatique de l'Etat  
du Luxembourg (CIE)



## Enterprise JavaBeans



## XML and Web Services



## CICS and MQSeries



**Contact us at:**

[www.ibm.com/servers/design\\_center/](http://www.ibm.com/servers/design_center/)

