IBM CICS Transaction Gateway

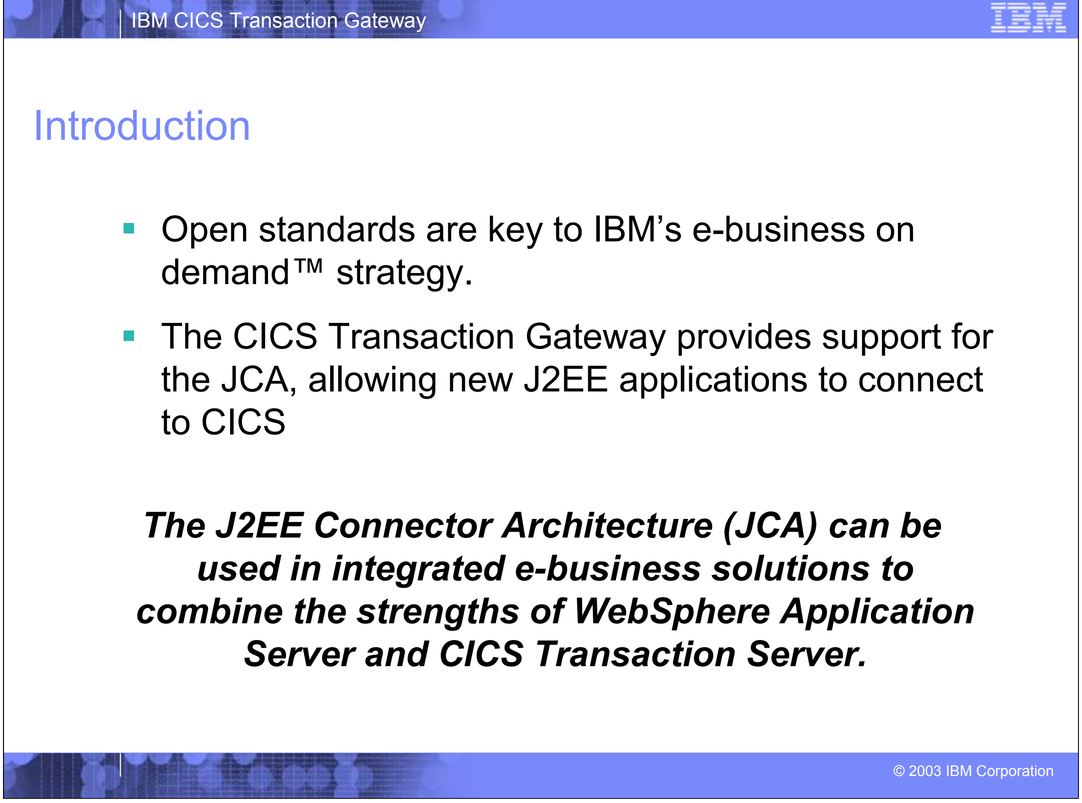
Integrating WebSphere® Application Server and CICS®, using the J2EE Connector Architecture

Agenda

- **Understanding the CICS Transaction Gateway**
- **J2EE Connector Architecture**
- **Application development**
- **Using the JCA with different CICS Transaction Gateway topologies**
- **Summary**

Part One

Understanding the CICS Transaction Gateway



Introduction

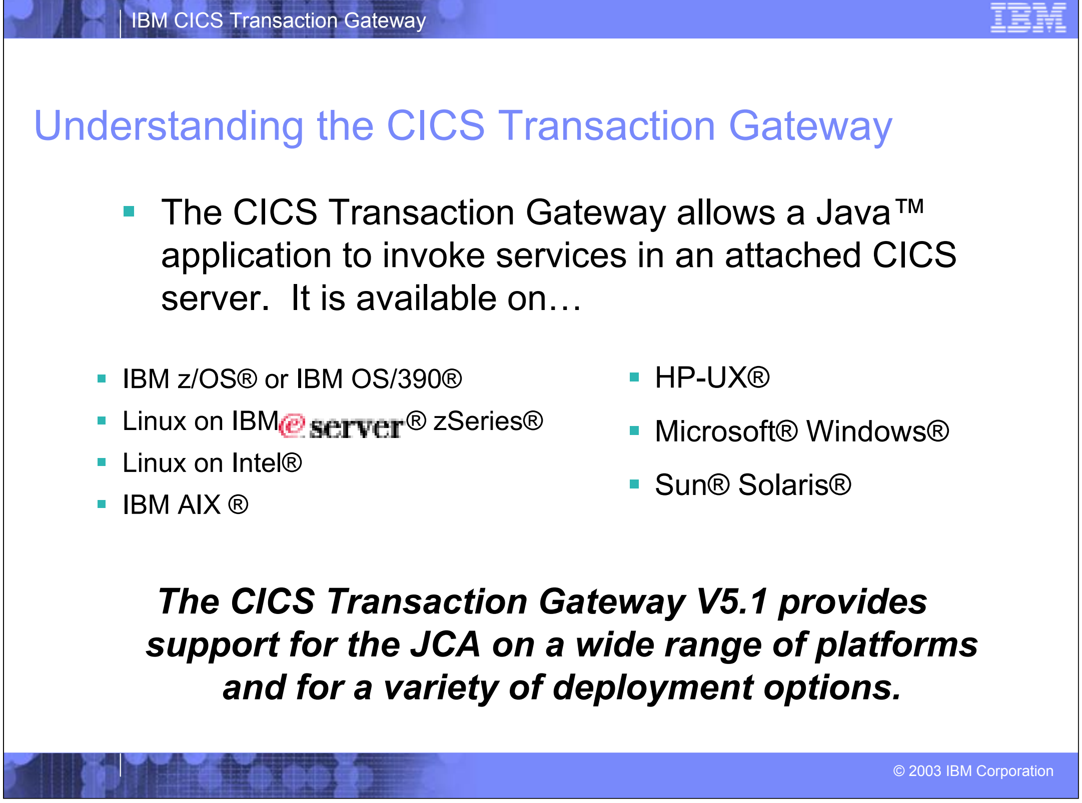
- Open standards are key to IBM’s e-business on demand™ strategy.
- The CICS Transaction Gateway provides support for the JCA, allowing new J2EE applications to connect to CICS

The J2EE Connector Architecture (JCA) can be used in integrated e-business solutions to combine the strengths of WebSphere Application Server and CICS Transaction Server.

Open standards are key to IBM’s e-business on demand™ strategy. End-to-end business processes require applications to be based on open standards like Java™ 2 Platform, Enterprise Edition (J2EE) in order to facilitate the integration of heterogeneous components. Because most of today’s existing IT applications run in an Enterprise Information System (EIS) such as IBM CICS® Transaction Server, you need an efficient way of reusing your existing assets and integrating mission-critical applications into new e-business solutions.

How can a new J2EE application connect to CICS? In many cases, the answer is to use the J2EE Connector Architecture (JCA), which defines a standard architecture for connecting the J2EE platform to heterogeneous EISs, such as CICS. The latest version of CICS Transaction Gateway provides support for the JCA, allowing the proven qualities of CICS to be exploited by J2EE applications running in IBM WebSphere® Application Server.

This presentation explains how using the JCA can provide benefits in terms of both application development and improved qualities of service in the system operating environment. The JCA can be used by business integration tools to allow application developers to compose new applications by mixing in pre-existing components, such as CICS programs, thus providing a rapid development capability for new J2EE e-business applications. The underlying infrastructure automatically provides the required qualities of service, including the management of connections, security and transactions.



Understanding the CICS Transaction Gateway

- The CICS Transaction Gateway allows a Java™ application to invoke services in an attached CICS server. It is available on…

- IBM z/OS® or IBM OS/390®
- Linux on IBM eserver® zSeries®
- Linux on Intel®
- IBM AIX ®
- HP-UX®
- Microsoft® Windows®
- Sun® Solaris®

The CICS Transaction Gateway V5.1 provides support for the JCA on a wide range of platforms and for a variety of deployment options.

The CICS Transaction Gateway is a set of client and server software components that allow a Java application to invoke services in an attached CICS server. The application can be an applet, a servlet, an enterprise bean or any other Java program. This presentation focuses on the use of Java servlets and enterprise beans running in IBM WebSphere Application Server.

The CICS Transaction Gateway, Version 5.1 is the most recent edition and currently supports the following platforms: IBM z/OS® or IBM OS/390®,Linux on IBM e-server® zSeries®, Linux on Intel®, IBM AIX® , HP-UX, Microsoft® Windows® and Sun Solaris operating environments. Connectivity is provided on these platforms from all supported WebSphere Application Server environments to all supported CICS servers.

Understanding the CICS Transaction Gateway

- The major components that comprise the CICS Transaction Gateway are:
  - Gateway daemon
  - Client daemon
  - Java class library
- The two main programming interfaces supported by the CICS Transaction Gateway are:
  - External Call Interface (ECI)
  - External Presentation Interface (EPI).

• Gateway daemon

A long-running process that functions as a server to network-attached Java client applications by listening on a specified TCP/IP port. The CICS Transaction Gateway supports a variety of TCP/IP-based network protocols, including Secure Sockets Layer (SSL) and HTTP variants.

• Client daemon

An integral part of the CICS Transaction Gateway on all distributed platforms. It provides the CICS client-server connectivity using the same technology as the CICS Universal Client.

• Java class library

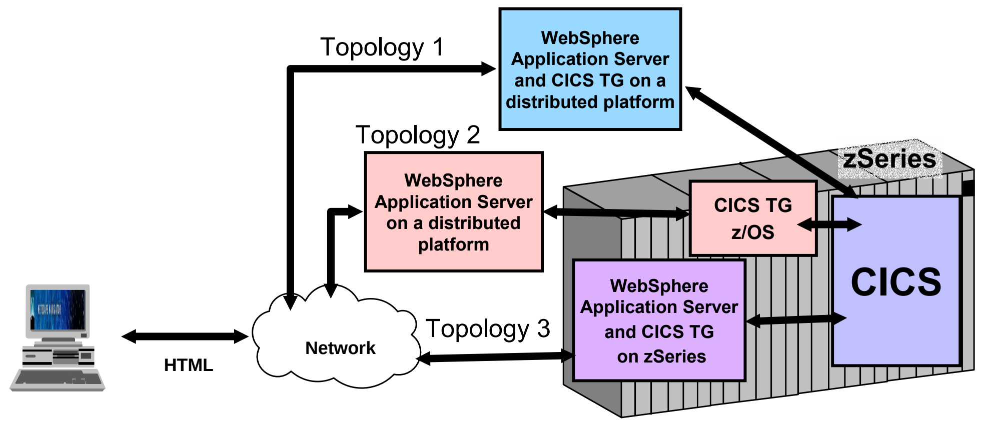
Java classes used by the application to invoke the services in an attached CICS server.

The two main programming interfaces supported by the CICS Transaction Gateway are the External Call Interface (ECI) and External Presentation Interface (EPI). The ECI provides a call interface to COMMAREA-based CICS programs, whereas EPI provides an API to invoke 3270-based programs.

Where possible it is recommended for the business logic program to maintain a COMMAREA interface to allow flexibility and in the future to support open standards as they evolve. COMMAREA-based programs are the best reuse option for this reason.

CICS Transaction Gateway - Deployment Options

- There are a number of deployment options available, each with their own differences. They can be classified into three distinct topologies:


Topology 1
WebSphere Application Server and CICS TG on a distributed platform
Topology 2
WebSphere Application Server on a distributed platform
zSeries
CICS TG z/OS
CICS
WebSphere Application Server and CICS TG on zSeries
Topology 3
Network
HTML

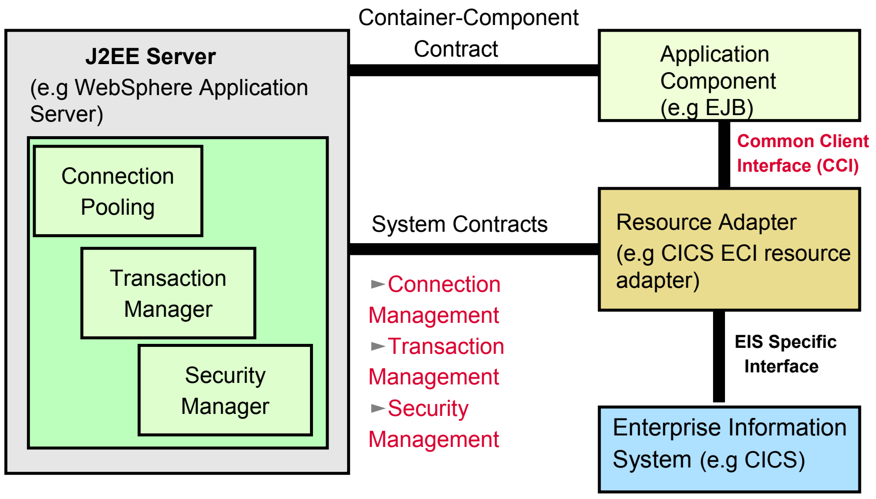
The CICS Transaction Gateway offers several different topologies to choose from. For example, when using WebSphere Application Server, connections can be local (the CICS Transaction Gateway communication is invoked through the Java Native Interface) or remote connections can be made to a Gateway daemon. The Gateway daemon may be running on the same machine as WebSphere Application Server or on another machine.

The diverse range of supported environments and the different deployment options can be classified into three distinct topologies. Later in this presentation, these options are outlined and the principal differences among them are described.

Part Two

J2EE Connector Architecture

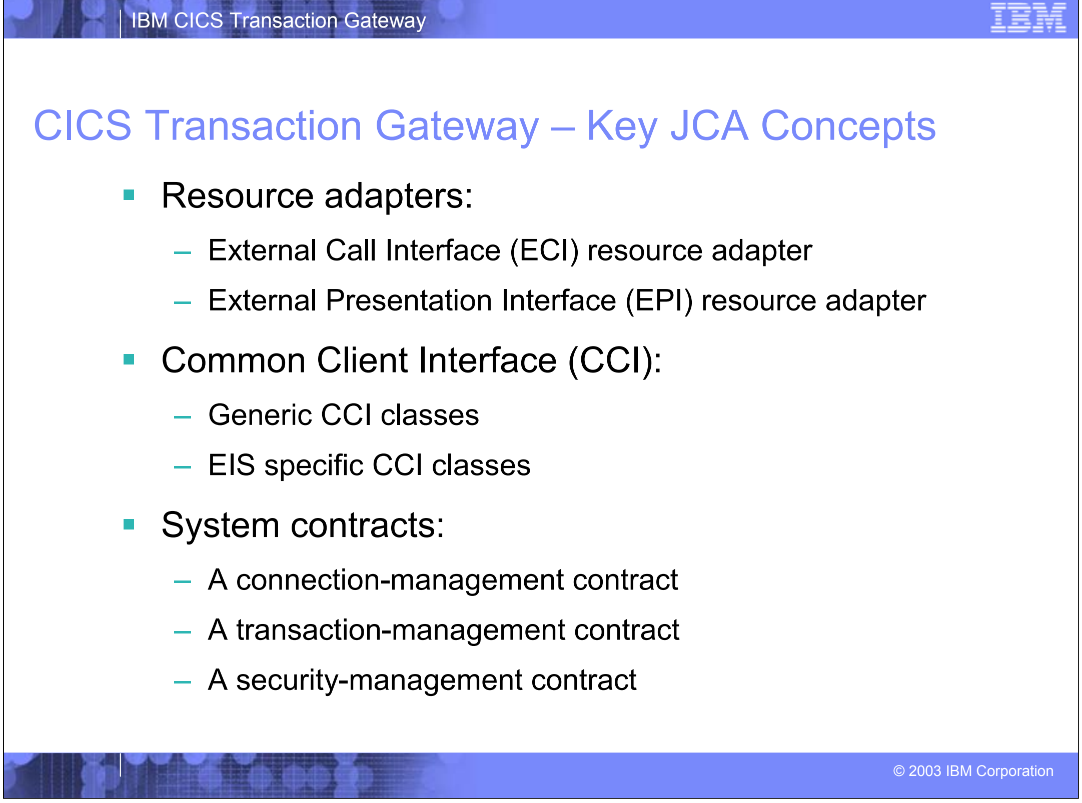
An introduction to the J2EE Connector Architecture


Container-Component Contract
J2EE Server
(e.g WebSphere Application Server)
Connection Pooling
Transaction Manager
Security Manager
Application Component (e.g EJB)
Common Client Interface (CCI)
System Contracts
►Connection Management
►Transaction Management
►Security Management
Resource Adapter (e.g CICS ECI resource adapter)
EIS Specific Interface
Enterprise Information System (e.g CICS)

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

JCA, Version 1.0 defines a number of components that make up this architecture, shown in the diagram on this slide.

Incidentally, before the existence of the JCA, IBM recognized a need for a common way to connect to EIS systems and introduced the Common Connector Framework (CCF). The JCA provides similar function to the CCF, but it is an open specification and provides stronger emphasis on system contracts and qualities of service.



CICS Transaction Gateway – Key JCA Concepts

- Resource adapters:
  - External Call Interface (ECI) resource adapter
  - External Presentation Interface (EPI) resource adapter
- Common Client Interface (CCI):
  - Generic CCI classes
  - EIS specific CCI classes
- System contracts:
  - A connection-management contract
  - A transaction-management contract
  - A security-management contract

The JCA specification states that the way an application server communicates with an EIS is through a resource adapter. A resource adapter is written to support a specific EIS.

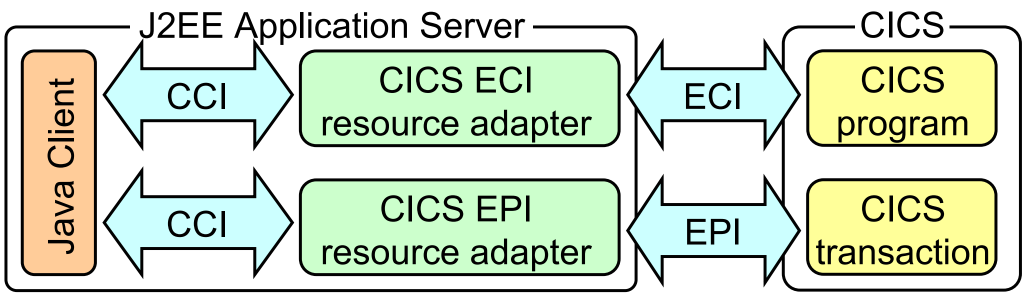
The CICS Transaction Gateway provides two resource adapters: the ECI resource adapter, which provides access to COMMAREA-based CICS programs, and an ECI resource adapter, which provides access to 3270-based programs.

All resource adapters expose common interfaces:

1) A common API for a Java component to communicate with a resource adapter. This API is called the Common Client Interface (CCI). The CCI is comprised of two sets of classes, generic classes applicable to any EIS, and a set of classes specific to the EIS (in this case CICS specific classes).

2) A common set of system contracts that allow the application server to manage connections, transactions, and security propagation with the resource adapter.

Resource Adapters



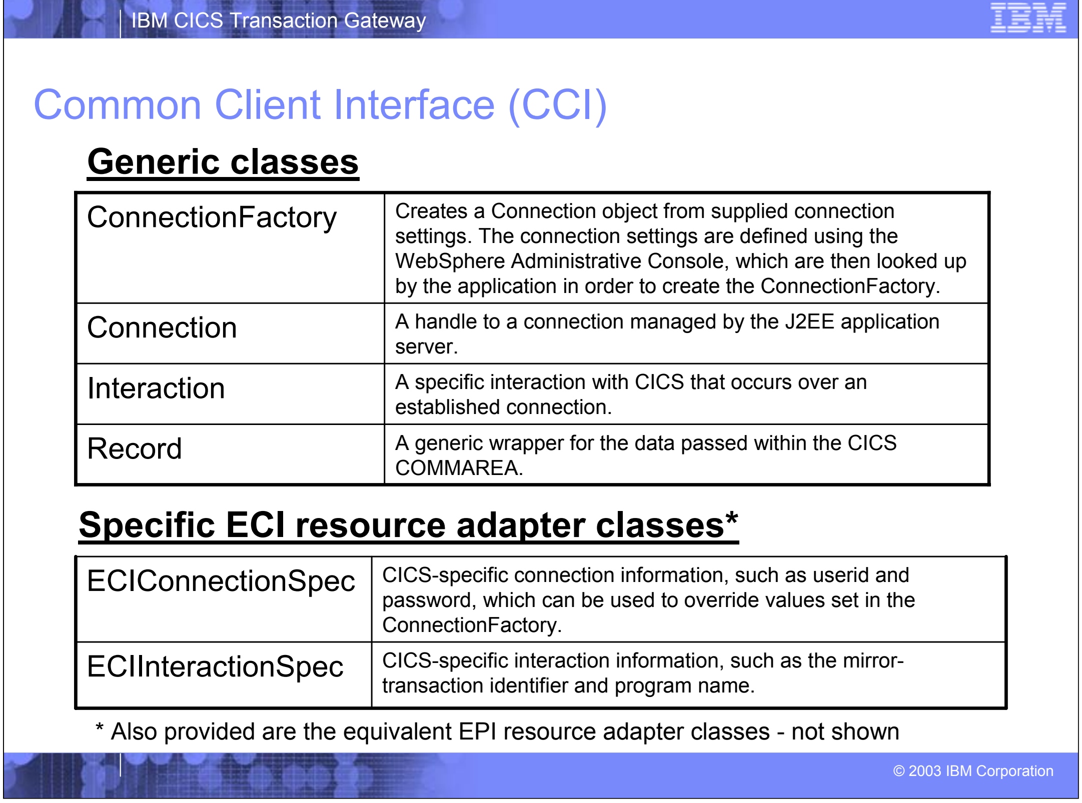
The CICS ECI resource adapter is the most commonly used CICS Transaction Gateway resource adapter.

Because it allows for optimal component reuse, the CICS ECI resource adapter is the focus of this presentation.

A major rationale for the JCA is to simplify the job of the EIS provider so that he has to provide a single resource adapter which will be able to be plugged into any applications server. In addition the application server needs to extend its system only once to be assured of connecting to multiple EIS's.

The CICS Transaction Gateway provides two separate resource adapters, one for ECI and one for EPI:

1) ECI resource adapter - This adapter is required for making calls to CICS COMMAREA-based programs. It is the most commonly used CICS Transaction Gateway resource adapter, and because it allows for optimal component reuse, is the focus of this presentation.
2) EPI resource adapter - This adapter is required for making calls to CICS 3270-based programs. It provides a record-based interface to terminal-oriented CICS applications.



Common Client Interface (CCI)

Generic classes

ConnectionFactory	Creates a Connection object from supplied connection settings. The connection settings are defined using the WebSphere Administrative Console, which are then looked up by the application in order to create the ConnectionFactory.
Connection	A handle to a connection managed by the J2EE application server.
Interaction	A specific interaction with CICS that occurs over an established connection.
Record	A generic wrapper for the data passed within the CICS COMMAREA.

Specific ECI resource adapter classes*

ECIConnectionSpec	CICS-specific connection information, such as userid and password, which can be used to override values set in the ConnectionFactory.
ECIInteractionSpec	CICS-specific interaction information, such as the mirror-transaction identifier and program name.

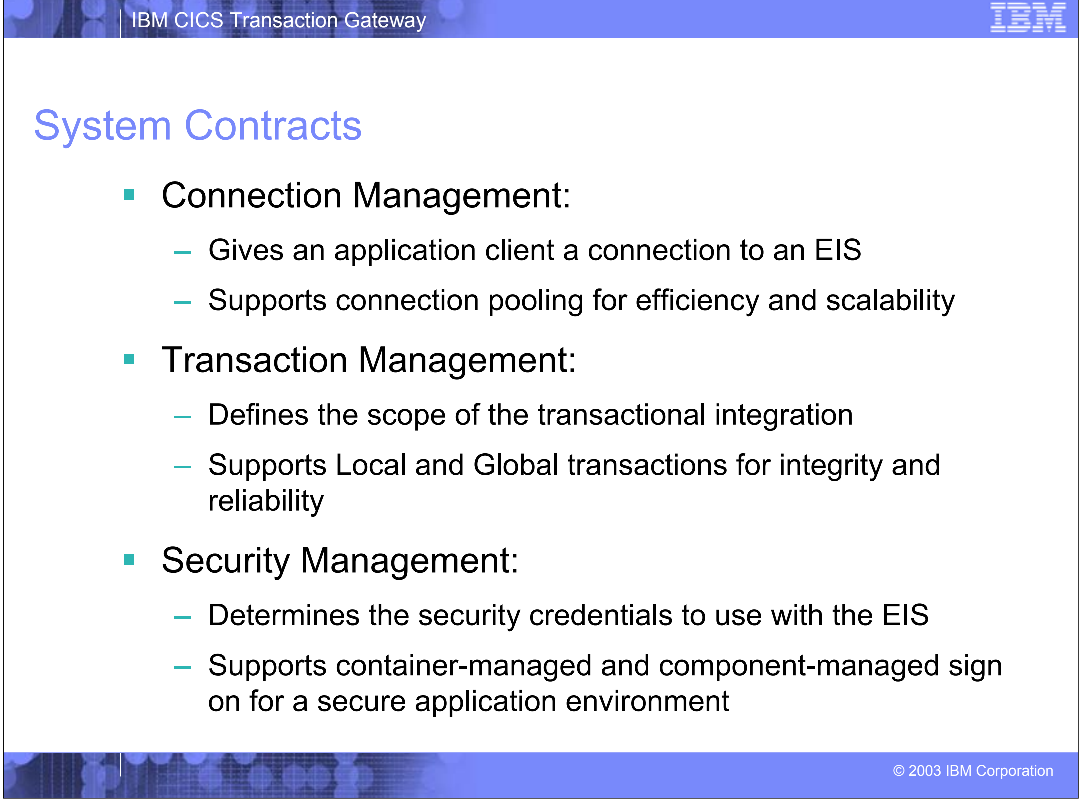
* Also provided are the equivalent EPI resource adapter classes - not shown

This table provides a brief description of the main generic CCI classes (as they apply to CICS) and the specific ECI resource adapter classes.

The Common Client Interface (CCI) defines a common application programming model for interacting with resource adapters and is independent of any specific EIS. This does not mean, however, that a developer can write exactly the same code to access one EIS (for example, CICS) as he writes to access another EIS (for example, IBM IMS™ systems).

The generic CCI classes (such as Connection or Interaction) are the same in that they are independent of the EIS, whereas specific EIS classes (such as ECIConnectionSpec) cater to the differences. For example, the parameters used to call a CICS program are different compared with those used to invoke an IMS transaction, but the programming model is the same — independent of the EIS. As a result, you can increase developer productivity when developing applications to communicate with multiple EISs.

The CCI programming interface is similar to other J2EE interfaces, such as the JDBC (Java Database Connectivity) interface.



System Contracts

- Connection Management:
  - Gives an application client a connection to an EIS
  - Supports connection pooling for efficiency and scalability
- Transaction Management:
  - Defines the scope of the transactional integration
  - Supports Local and Global transactions for integrity and reliability
- Security Management:
  - Determines the security credentials to use with the EIS
  - Supports container-managed and component-managed sign on for a secure application environment

The JCA defines a standard set of system-level contracts between a J2EE application server and a resource adapter. These system-level contracts define the scope of the managed environment that the J2EE application server provides for JCA components. The standard contracts include:

• A connection-management contract

Provides a consistent application programming model for connection acquisition and enables a J2EE application server to pool connections to a back-end EIS. This leads to a scalable and efficient environment that can support a large number of components requiring access to an EIS system.

• A transaction-management contract

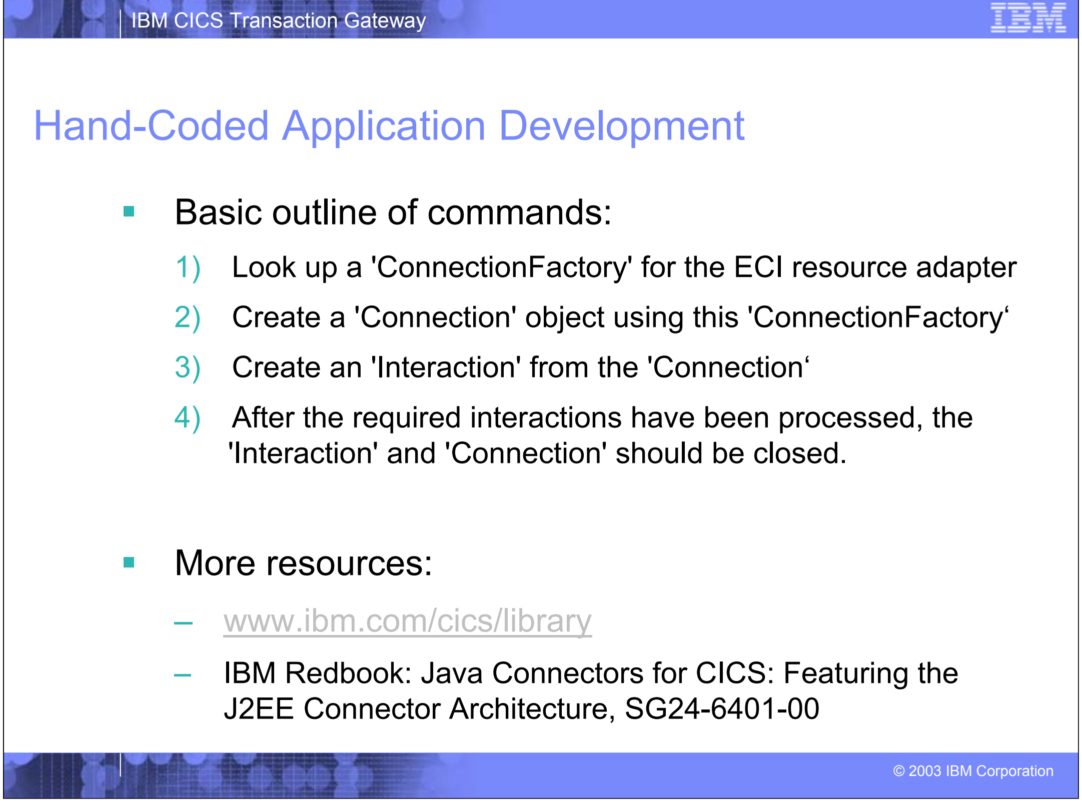
Defines the scope of transactional integration between the J2EE application server and an EIS that supports transactional access. This contract allows a J2EE application server to use a transaction manager to manage transactions across multiple resource managers (known as global transactions). This contract also supports the LocalTransaction interface, which refers to transactions that are managed internally to a resource manager without the involvement of an external transaction manager. Within WebSphere Application Server these transactions are known as Resource Manager Local Transactions.

• A security-management contract

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. Both container-managed sign-on (in which the J2EE application server is responsible for flowing security context to the EIS) and component-managed sign-on (in which the application is responsible for flowing security context to the EIS) are supported.

These system contracts are transparent to the application developer, which means they do not have to implement these services themselves. In a managed environment such as WebSphere Application Server, system contracts are contractual agreements between the J2EE application server and the resource adapter on how to manage connections, transactions and security. It is these system contracts that make the JCA such a powerful solution for integrating CICS applications with J2EE applications running in WebSphere Application Server.

Part Three
Application Development



Hand-Coded Application Development

- Basic outline of commands:
  1) Look up a 'ConnectionFactory' for the ECI resource adapter
  2) Create a 'Connection' object using this 'ConnectionFactory‘
  3) Create an 'Interaction' from the 'Connection‘
  4) After the required interactions have been processed, the 'Interaction' and 'Connection' should be closed.

- More resources:
  – www.ibm.com/cics/library
  – IBM Redbook: Java Connectors for CICS: Featuring the J2EE Connector Architecture, SG24-6401-00

The following list shows the basic outline of the commands for using the CCI with the CICS ECI resource adapter:

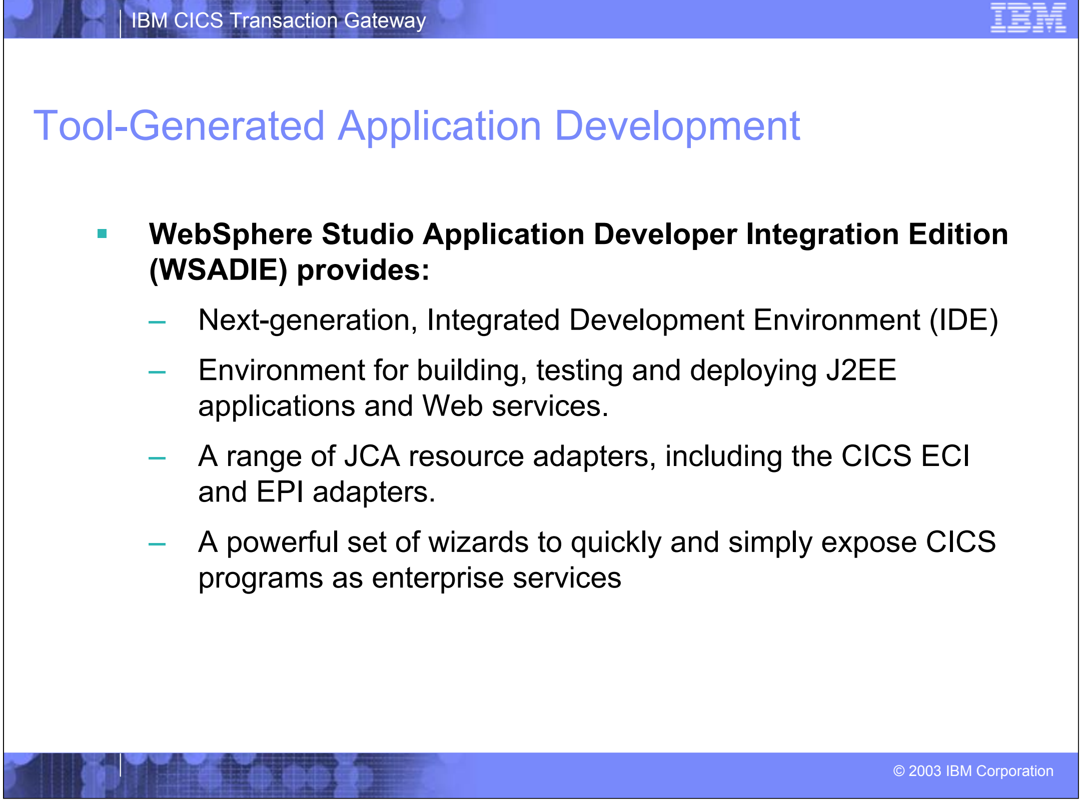
1. Look up a 'ConnectionFactory' for the ECI resource adapter.

2. Create a 'Connection' object using this 'ConnectionFactory'. A 'Connection' is a handle to the underlying network connection to the EIS. Specific connection properties, such as a user name and password, can be passed using an 'ECIConnectionSpec' object.

3. Create an 'Interaction' from the 'Connection'. Specific interaction properties can be passed using an 'ECIInteractionSpec' object. The call to the EIS is initiated by invoking the 'execute()' method on the

'Interaction' , passing data as input and output records.

4. After the required interactions have been processed, the 'Interaction' and 'Connection' should be closed.



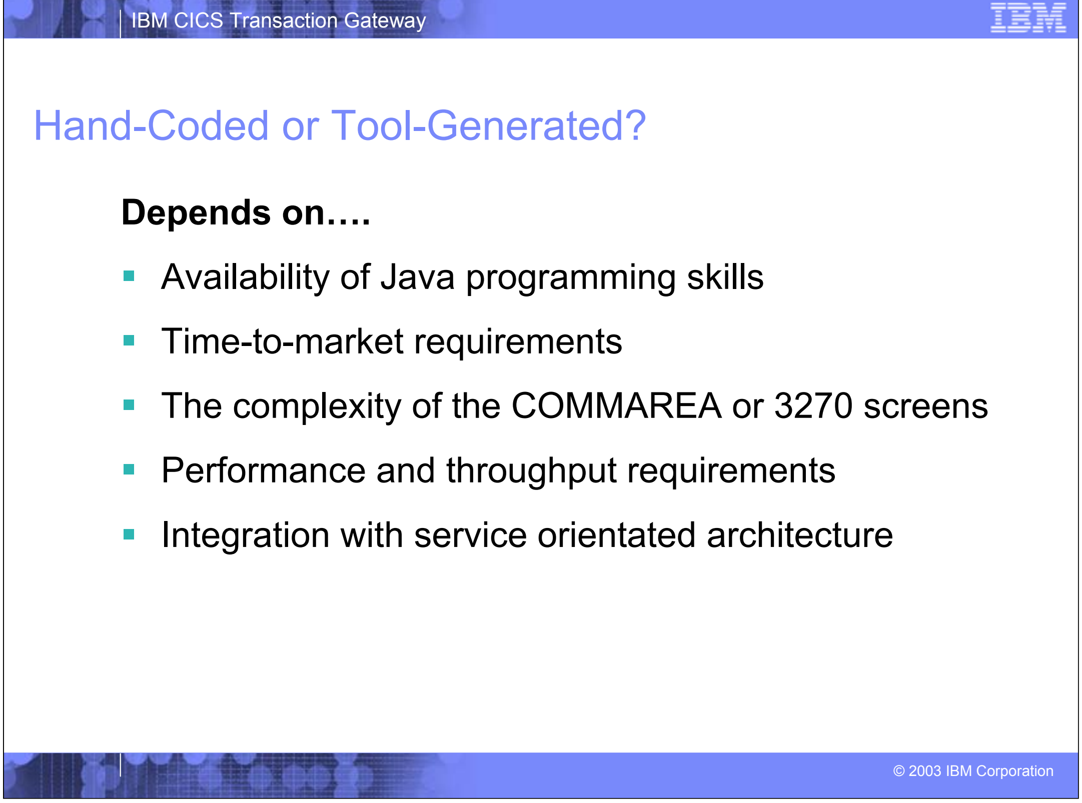
Tool-Generated Application Development

- **WebSphere Studio Application Developer Integration Edition (WSADIE) provides:**
  - Next-generation, Integrated Development Environment (IDE)
  - Environment for building, testing and deploying J2EE applications and Web services.
  - A range of JCA resource adapters, including the CICS ECI and EPI adapters.
  - A powerful set of wizards to quickly and simply expose CICS programs as enterprise services

The CCI is targeted primarily toward application development tools and enterprise application integration frameworks. IBM WebSphere Studio Application Developer Integration Edition provides a development environment that contains a range of JCA resource adapters, including the CICS ECI and EPI adapters.

WebSphere Studio Application Developer Integration Edition is targeted to development projects requiring integration with back-end systems and provides support for a variety of client types, including Web services clients. It also allows existing software components, such as CICS programs, to be wrapped as services, which can communicate with other applications through a common interface known as Web Services Invocation Framework (WSIF).

A CICS ECI service can be generated from an existing CICS COMMAREA-based program using a wizard, and the service can then be deployed in WebSphere Application Server.



Hand-Coded or Tool-Generated?

Depends on….

- Availability of Java programming skills
- Time-to-market requirements
- The complexity of the COMMAREA or 3270 screens
- Performance and throughput requirements
- Integration with service orientated architecture

Several factors should be considered when choosing between a hand-coded development process or a tooled solution like WebSphere Studio Application Developer. These include:

• The availability of Java programming skills.

• Time-to-market requirements. (Tool-generated applications may take less time to develop.)

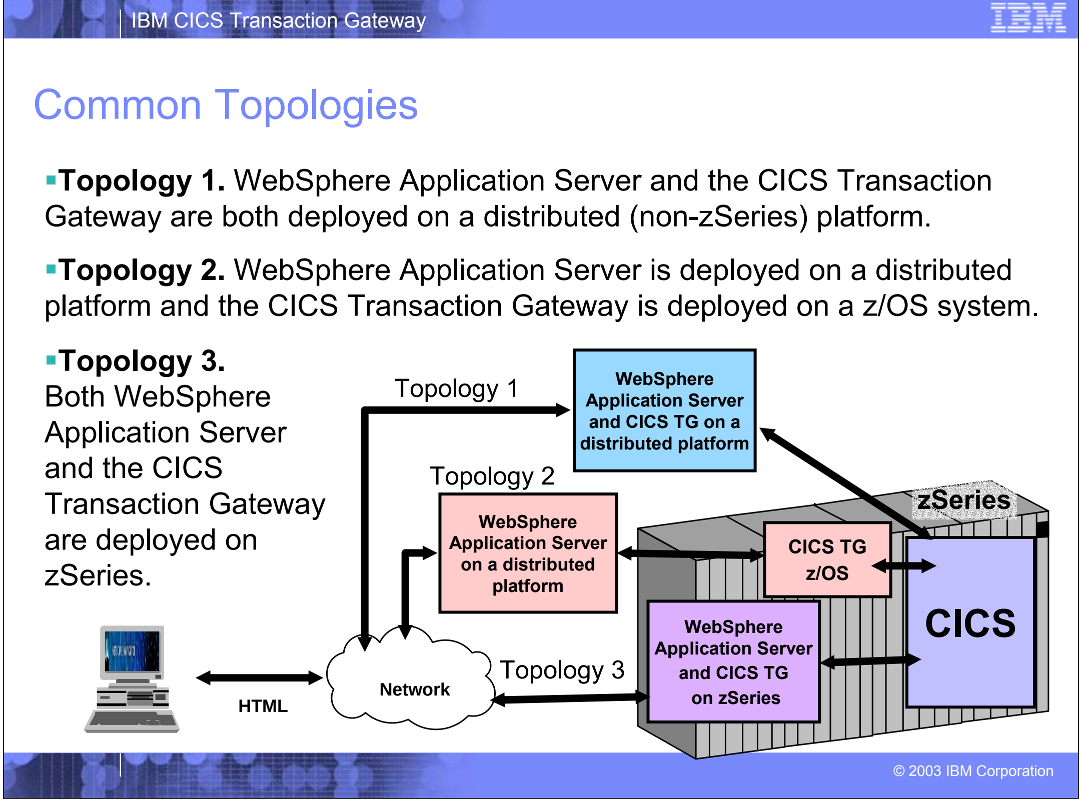
• The complexity of the COMMAREA or 3270 screens. (Hand-coded solutions require the programmer to manage the conversion of COMMAREA structures or 3270 screens to JCA records, which can be quite complicated and time-consuming. A tooled solution can automate this process.)

• The performance and throughput requirements. (The performance of hand coded CCI applications is likely to be better than tool-generated applications because of the additional abstraction and run-time requirements of the tool-generated applications.)

• The overall system architecture requirements. The tooling capability of WebSphere Application Developer Integration Edition is based on the generation of enterprise services to be integrated within a service-oriented architecture.

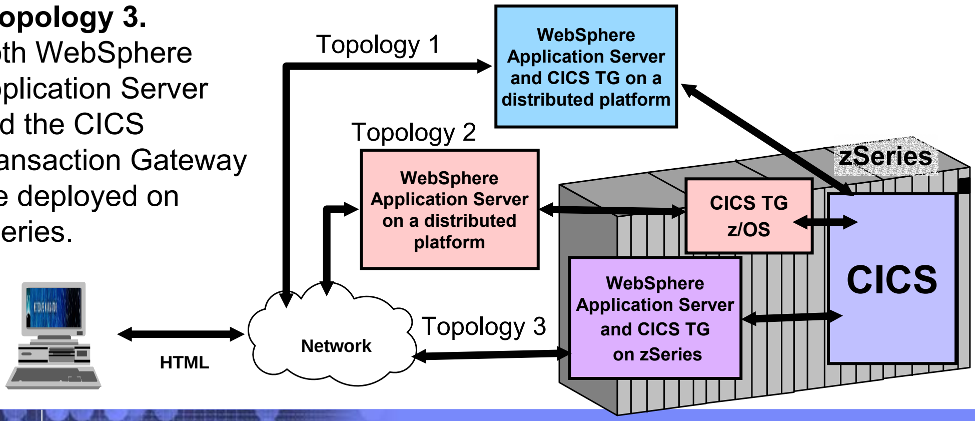
Part Four

Using the JCA with different CICS Transaction Gateway topologies



Common Topologies

- **Topology 1.** WebSphere Application Server and the CICS Transaction Gateway are both deployed on a distributed (non-zSeries) platform.
- **Topology 2.** WebSphere Application Server is deployed on a distributed platform and the CICS Transaction Gateway is deployed on a z/OS system.
- **Topology 3.** Both WebSphere Application Server and the CICS Transaction Gateway are deployed on zSeries.


Topology 1
WebSphere Application Server and CICS TG on a distributed platform
Topology 2
WebSphere Application Server on a distributed platform
zSeries
CICS TG z/OS
CICS
WebSphere Application Server and CICS TG on zSeries
Topology 3
Network
HTML

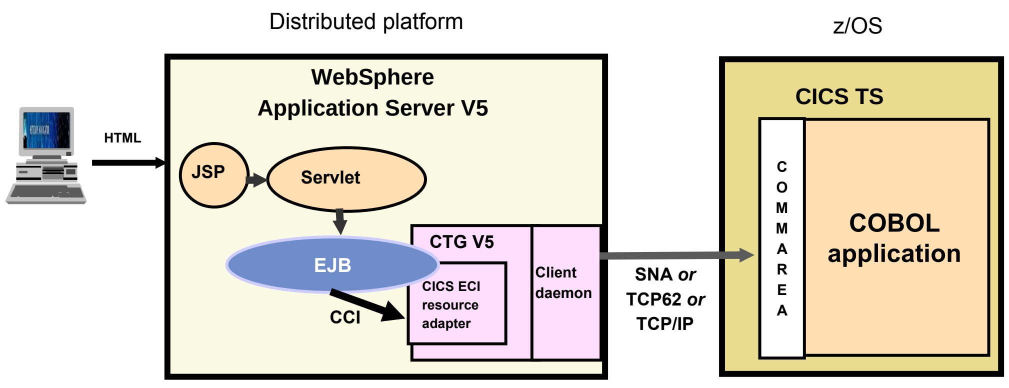
The JCA system contracts are the key to qualities of service provided by WebSphere Application Server and the CICS ECI resource adapter. They provide the mechanisms by which the management of connections, security and transactions are performed. However, the qualities of service vary depending on the topology in use. The three most common topologies are shown on this slide. They are:

Topology 1. WebSphere Application Server and the CICS Transaction Gateway are both deployed on a distributed (non-zSeries) platform.

Topology 2. WebSphere Application Server is deployed on a distributed platform and the CICS Transaction Gateway is deployed on a z/OS system.

Topology 3. Both WebSphere Application Server and the CICS Transaction Gateway are deployed on zSeries.

Topology 1: WebSphere Application Server and CICS Transaction Gateway deployed on distributed platforms

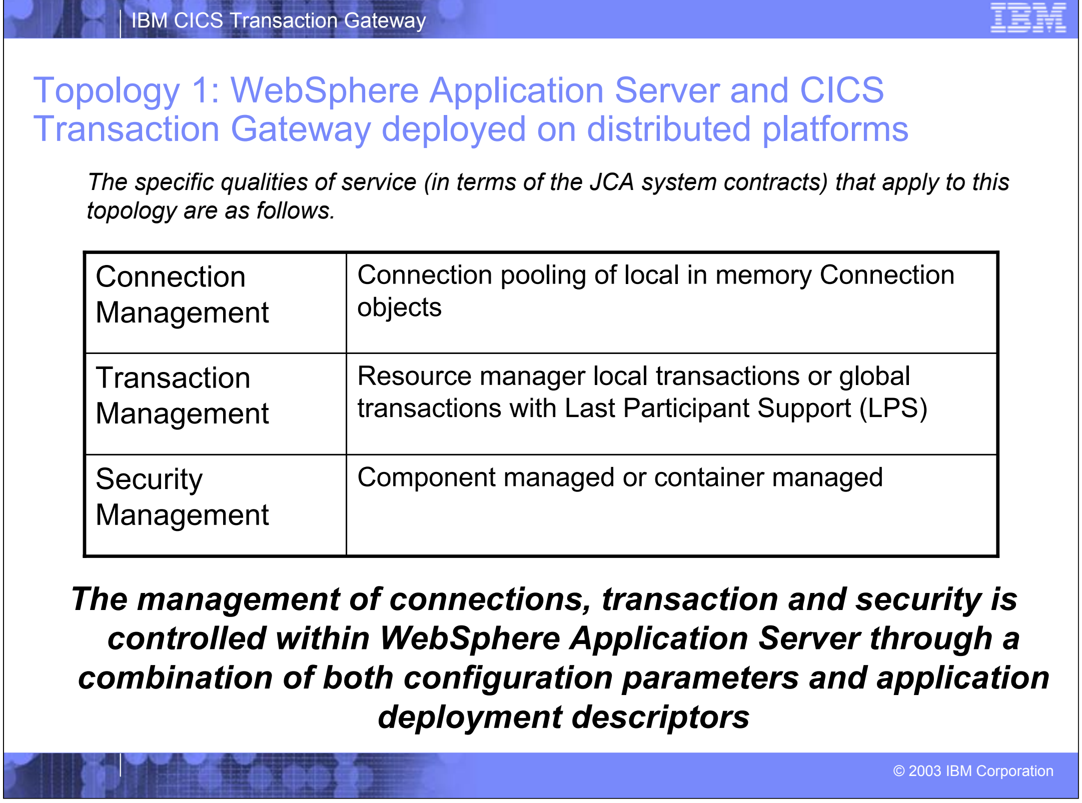


Topology 1: WebSphere Application Server and CICS Transaction Gateway deployed on distributed platforms

In this topology, both WebSphere Application Server and CICS Transaction Gateway are deployed on one of the distributed platforms, such as a Windows or UNIX® platform. Both ECI and EPI resource adapters can be used in this configuration. This slide shows an Enterprise JavaBeans (EJB) application using the ECI resource adapter to access a COMMAREA-based CICS COBOL application.

In this configuration, the Gateway daemon is not required because the CICS Transaction Gateway local protocol is used to invoke the Client daemon native libraries directly from the enterprise bean. The protocol is specified on the gatewayURL parameter in the connection factory custom properties using the WebSphere Administration Console. The Client daemon flows the ECI request to the CICS server using either a Systems Network Architecture (SNA) connection or a TCP62 connection. The TCP62 protocol allows SNA flows to be encapsulated in TCP/IP packets. If the CICS server is at the CICS Transaction Server, Version 2.2 level, the request from the Client daemon can be passed to CICS using TCP/IP directly (rather than using an SNA protocol).

A variation of topology 1 is where WebSphere Application Server and CICS Transaction Gateway are installed on one of the distributed platforms but reside on different machines. The main difference in this variation is that the connection pool represents physical network connections between WebSphere Application Server and the Gateway daemon. As a result, the connection pooling may provide greater benefit because it can reduce the network I/O overhead as well as memory and CPU consumption.



Topology 1: WebSphere Application Server and CICS Transaction Gateway deployed on distributed platforms

The specific qualities of service (in terms of the JCA system contracts) that apply to this topology are as follows.

Connection Management	Connection pooling of local in memory Connection objects
Transaction Management	Resource manager local transactions or global transactions with Last Participant Support (LPS)
Security Management	Component managed or container managed

The management of connections, transaction and security is controlled within WebSphere Application Server through a combination of both configuration parameters and application deployment descriptors

Connection management

Pooling of connections is provided seamlessly by the pool manager component of WebSphere Application Server. This allows the reuse of connections to the resource adapter between multiple J2EE components, such as enterprise beans or servlets. The pool parameters are configured through the connection pool settings for the connection factory. Because the CICS Transaction Gateway and WebSphere Application Server both reside on the same host in this configuration, these pooled connections are actually a set of connection objects, each representing a local connection between the WebSphere Application Server and the co-resident CICS Transaction Gateway. As a result, the connection pooling provides a benefit in terms of reduced memory and CPU utilization. The network connections from the CICS Transaction Gateway into CICS Transaction Server are managed and pooled independently by the CICS Transaction Gateway Client daemon component.

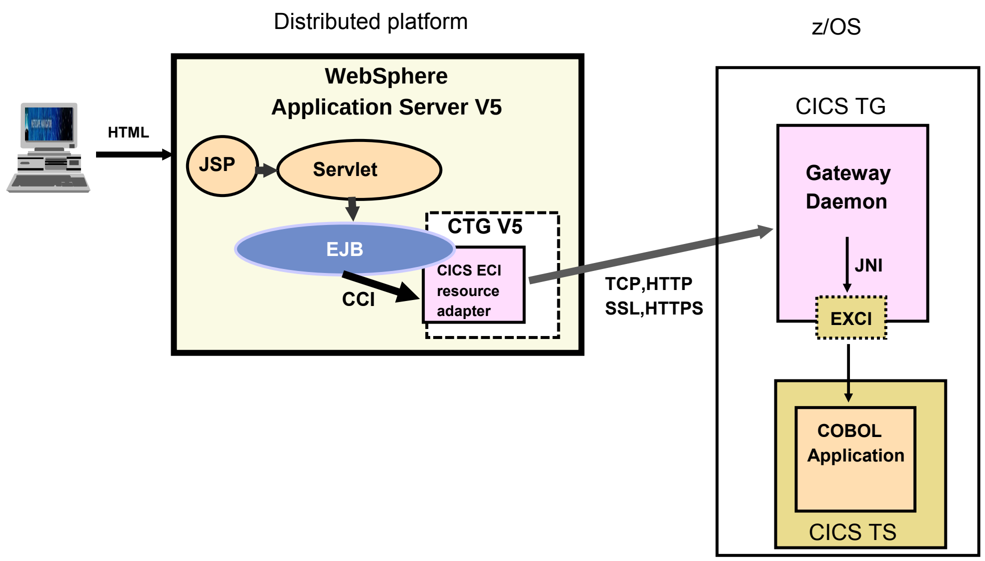
Transaction management

The ECI resource adapter only supports the LocalTransaction interface. As a result, the scope of the transaction is limited to the Resource Manager (that is, the associated connection factory and the specified CICS server). Such Resource Manager Local Transactions only support one-phase commit processing. So, if the enterprise bean shown on the previous slide attempts to access another recoverable resource (for example, a JDBC call to update a DB2® table), an exception will occur when the transaction manager tries to commit both updates. However, if you use a J2EE application server that supports the JCA option of last-resource commit optimization, an interaction using an ECIConnectionFactory can participate in a global transaction, provided that it is the only one phase-commit resource in the global transaction. This function is provided by the Last Participant Support in WebSphere Application Server Enterprise, Version 5.

Security management

Security credentials (userid and password) propagated through to CICS from WebSphere Application Server can be determined by the application (component managed) or by the Web or

Topology 2: Remote Gateway daemon on z/OS



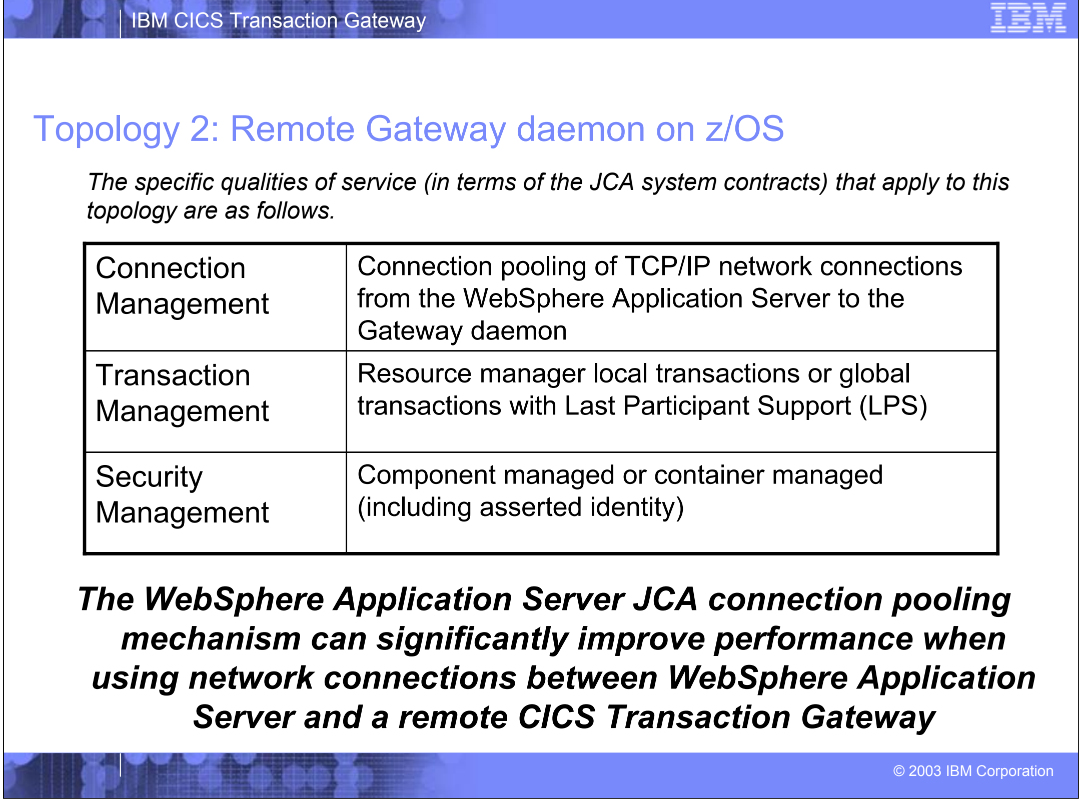
Topology 2: WebSphere Application Server deployed on a distributed platform and the CICS Transaction Gateway deployed on z/OS

When WebSphere Application Server is deployed on one of the distributed platforms, it is possible to access CICS through a Gateway daemon running on z/OS, as shown on this slide.

In this configuration, the protocol specified in the connection settings of the connection factory is one of the remote protocols (TCP, HTTP, SSL or HTTPS). The communication from the CICS Transaction Gateway on z/OS to the CICS server utilizes the External CICS Interface (EXCI). The EXCI is a function of CICS Transaction Server for z/OS that allows non-CICS address spaces on z/OS systems to link to programs in a CICS Transaction Server region. This configuration is being widely used today because topology 1 only supports native TCP/IP connections into zSeries when the CICS Transaction Server is at the Version 2.2 or 2.3 level. Topology 2 supports TCP/IP connection into zSeries systems for both CICS Transaction Server, Version 2 and CICS Transaction Server, Version 1.3.

When CICS security is enabled, topology 1 requires that a userid and password are flowed with each ECI request. In some situations, for example, when authentication is being undertaken using a different mechanism, such as with client certificates, userid authentication by CICS does not easily fit within the security design of a project. Using topology 2 can help to avoid this problem because CICS Transaction Gateway for z/OS supports the concept of asserted identity, whereby a pre-authenticated userid is flowed into CICS without a password.

Topology 2 enables integration with z/OS IP workload-management functions, including Sysplex Distributor and TCP/IP port sharing. Use of these technologies allows an individual Gateway daemon to be removed as a single point of failure and enables incoming work to be efficiently balanced across multiple Gateway daemons in multiple z/OS logical partitions (LPARs). Topology



Topology 2: Remote Gateway daemon on z/OS

The specific qualities of service (in terms of the JCA system contracts) that apply to this topology are as follows.

Connection Management	Connection pooling of TCP/IP network connections from the WebSphere Application Server to the Gateway daemon
Transaction Management	Resource manager local transactions or global transactions with Last Participant Support (LPS)
Security Management	Component managed or container managed (including asserted identity)

The WebSphere Application Server JCA connection pooling mechanism can significantly improve performance when using network connections between WebSphere Application Server and a remote CICS Transaction Gateway

Connection management

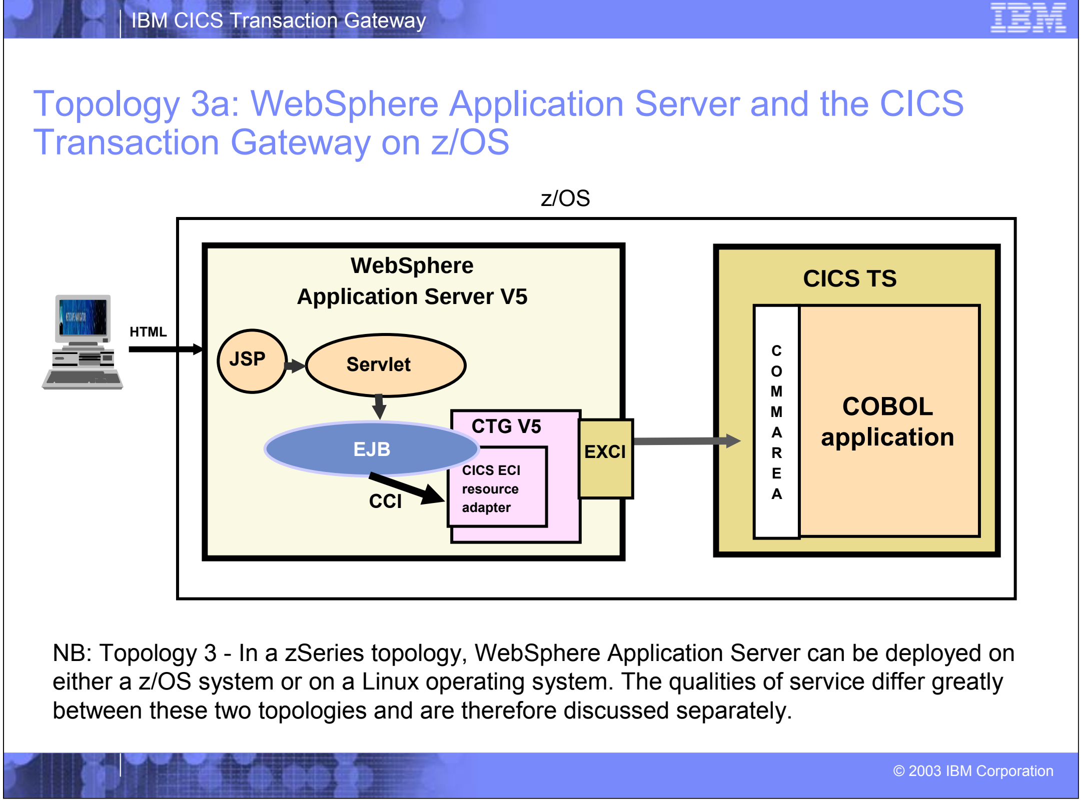
The connection pool represents physical network connections between WebSphere Application Server and the Gateway daemon on z/OS. In such a configuration it is essential to have an efficient connection-pooling mechanism because otherwise, a significant proportion of the time from making the connection to receiving the result from CICS and closing the connection can be in the creation and destruction of the connection itself. The JCA connection-pooling mechanism mitigates this overhead by allowing connections to be pooled by the WebSphere Application Server pool manager. This is particularly important when encrypted SSL connections are required, because the cost of SSL connection handshaking is typically the most expensive component in SSL communications.

Transaction management

Although the Gateway daemon is deployed on z/OS, the ECI resource adapter is installed into WebSphere Application Server on the distributed platform, so only the LocalTransaction interface is supported. Note that if the enterprise bean is deployed with a transactional deployment descriptor (for example, a value of REQUIRED), the resulting ECI request to CICS will use the extended unit of work mode. In this case, both the CICS Transaction Gateway and CICS Transaction Server need to be configured to register with IBM MVS™ Resource Recovery Services (RRS).

Security management

In this configuration, the Gateway daemon is the entry point to the zSeries system in which your CICS system is running, so it is normal for the Gateway daemon to perform an authentication check for incoming ECI requests from clients. However, after the end user has authenticated to WebSphere Application Server, there may not be a password available to send to the Gateway daemon. In this case, therefore, you will need to devise a way to establish a trust relationship between WebSphere Application Server and the Gateway daemon so that WebSphere Application Server can be trusted to flow only the userid on the request through to the CICS



Topology 3a: WebSphere Application Server and the CICS Transaction Gateway on z/OS

NB: Topology 3 - In a zSeries topology, WebSphere Application Server can be deployed on either a z/OS system or on a Linux operating system. The qualities of service differ greatly between these two topologies and are therefore discussed separately.

Topology 3

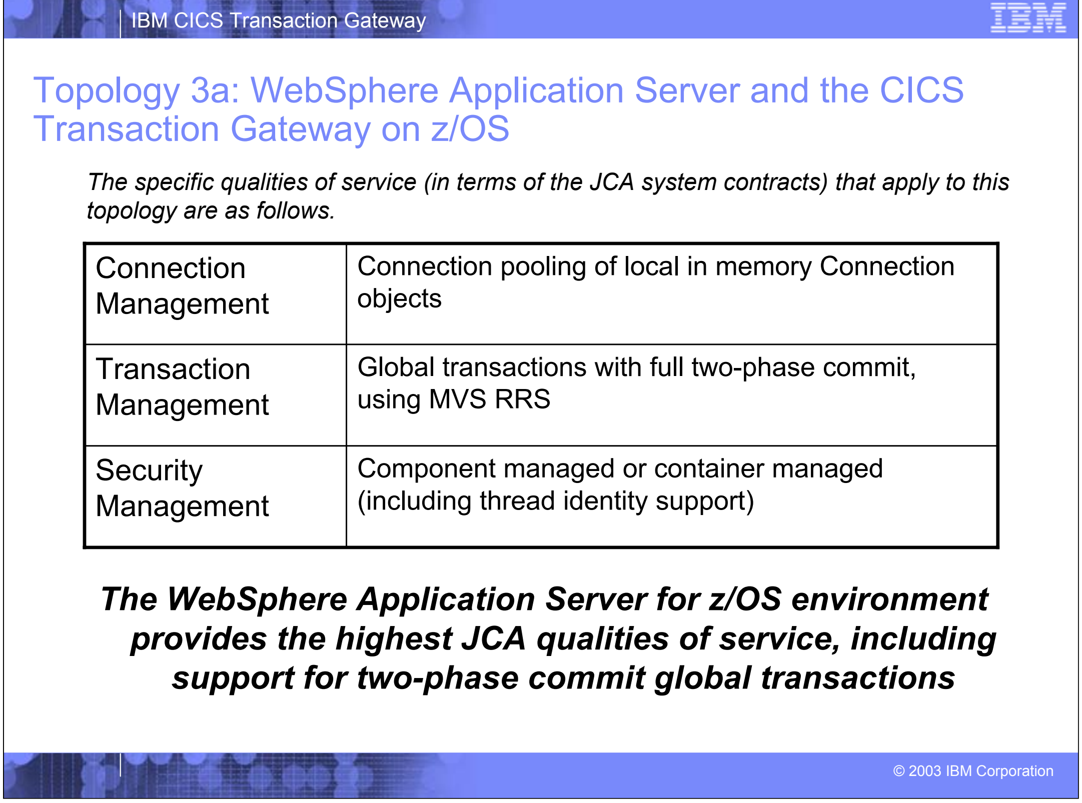
In a zSeries topology, WebSphere Application Server can be deployed on either a z/OS system or on a Linux operating system. The qualities of service differ greatly between these two topologies and are therefore discussed separately.

Topology 3a: WebSphere Application Server and the CICS Transaction Gateway on z/OS

In this topology only the CICS ECI resource adapter is supported. As in topology 1, the most common z/OS configuration makes use of a local CICS Transaction Gateway. On z/OS, this results in a direct cross-memory EXCI connection between WebSphere Application Server and CICS. The diagram on this slide shows the application deployed to WebSphere Application Server for z/OS, Version 5.

The CICS Transaction Gateway, Version 5.01 introduced remote connection support for this topology, which specifically targets z/OS.e customers*. However, the highest qualities of service can be achieved when a local connection between WebSphere Application Server and CICS Transaction Gateway is used (as shown in the diagram). The next slide looks at these services in relation to the three JCA system contracts.

*z/OS.e is a specially priced offering of z/OS that provides select z/OS function for the new zSeries 800 system. Traditional workloads, such as CICS Transaction Server, are not licensed for z/OS.e. Because the CICS Transaction Gateway relies on the EXCI interface provided by CICS Transaction Server for z/OS, the CICS Transaction Gateway must be installed in a full-function z/OS LPAR. This means a remote CICS Transaction Gateway connection must be used from WebSphere Application Server into the CICS Transaction Gateway.



Topology 3a: WebSphere Application Server and the CICS Transaction Gateway on z/OS

The specific qualities of service (in terms of the JCA system contracts) that apply to this topology are as follows.

Connection Management	Connection pooling of local in memory Connection objects
Transaction Management	Global transactions with full two-phase commit, using MVS RRS
Security Management	Component managed or container managed (including thread identity support)

The WebSphere Application Server for z/OS environment provides the highest JCA qualities of service, including support for two-phase commit global transactions

Connection management

The connection pool is a set of connection objects managed by WebSphere Application Server, which is not directly associated with the EXCI pipes used for communication to CICS.

Transaction management

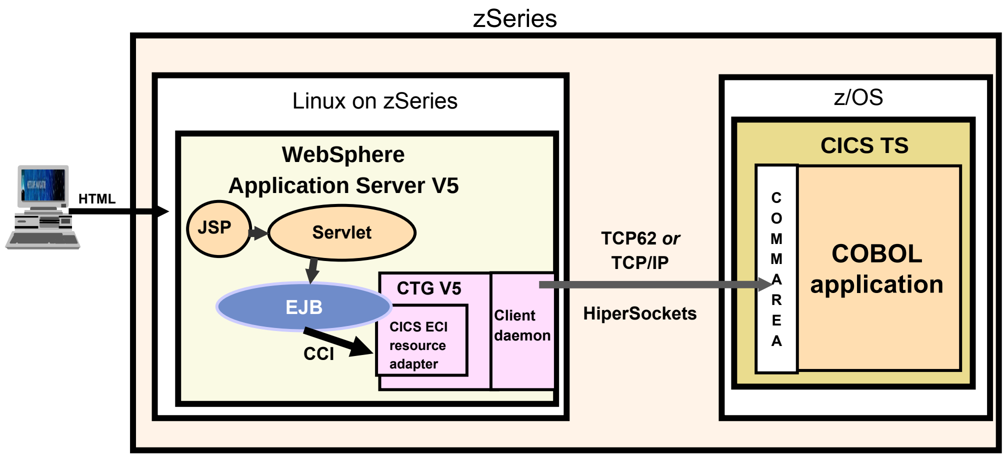
A two-phase commit capability is provided through RRS, an integral part of z/OS. RRS acts as the external transaction coordinator for z/OS in managing the transaction scope between WebSphere Application Server, CICS and other z/OS subsystems, including IMS/TM, IMS/DB, IBM DB2 and WebSphere MQ systems. This is the only scenario in which the two-phase commit transactional scope can fully extend to recoverable CICS resources.

Security management

Both container-managed and component-managed sign-on are supported. When using container-managed sign-on, a z/OS system-specific functionality known as thread identity support is provided by WebSphere Application Server for z/OS, Version 5. WebSphere Application Server for z/OS can be configured to use a local OS registry (such as IBM RACF®) as its user registry. When using RACF as the user registry, the security identity established after authentication in WebSphere Application Server will be an RACF userid if you use basic authentication or form-based login. If an SSL client certificate is used to authenticate, you can configure RACF to map that certificate to a RACF userid. This means the Java thread in WebSphere Application Server on z/OS will have a security identity that is a RACF userid. The unique thread identity support in WebSphere Application Server for z/OS allows WebSphere Application Server to automatically pass the userid of the thread (the caller's identity) to CICS when using the ECI resource adapter. This satisfies a common end-to-end security requirement of automatically propagating the authenticated caller's userid from WebSphere Application Server to CICS.

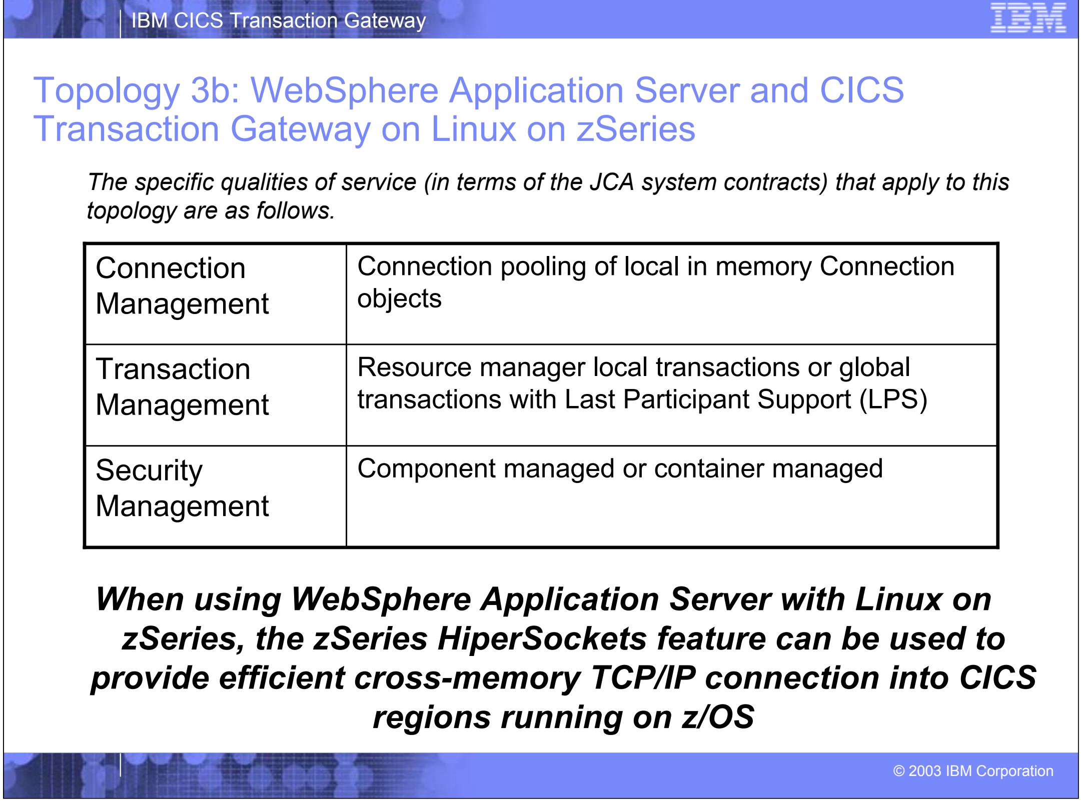
In this topology, the ECI resource adapter flows only the userid to CICS with the ECI request — it

Topology 3b: WebSphere Application Server and CICS Transaction Gateway on Linux on zSeries



NB: Topology 3 - In a zSeries topology, WebSphere Application Server can be deployed on either a z/OS system or on a Linux operating system. The qualities of service differ greatly between these two topologies and are therefore discussed separately.

When WebSphere Application Server is deployed within Linux on zSeries, this configuration provides a highly flexible and scalable environment based on the virtualization capabilities of IBM z/VM® and Linux systems.



Topology 3b: WebSphere Application Server and CICS Transaction Gateway on Linux on zSeries

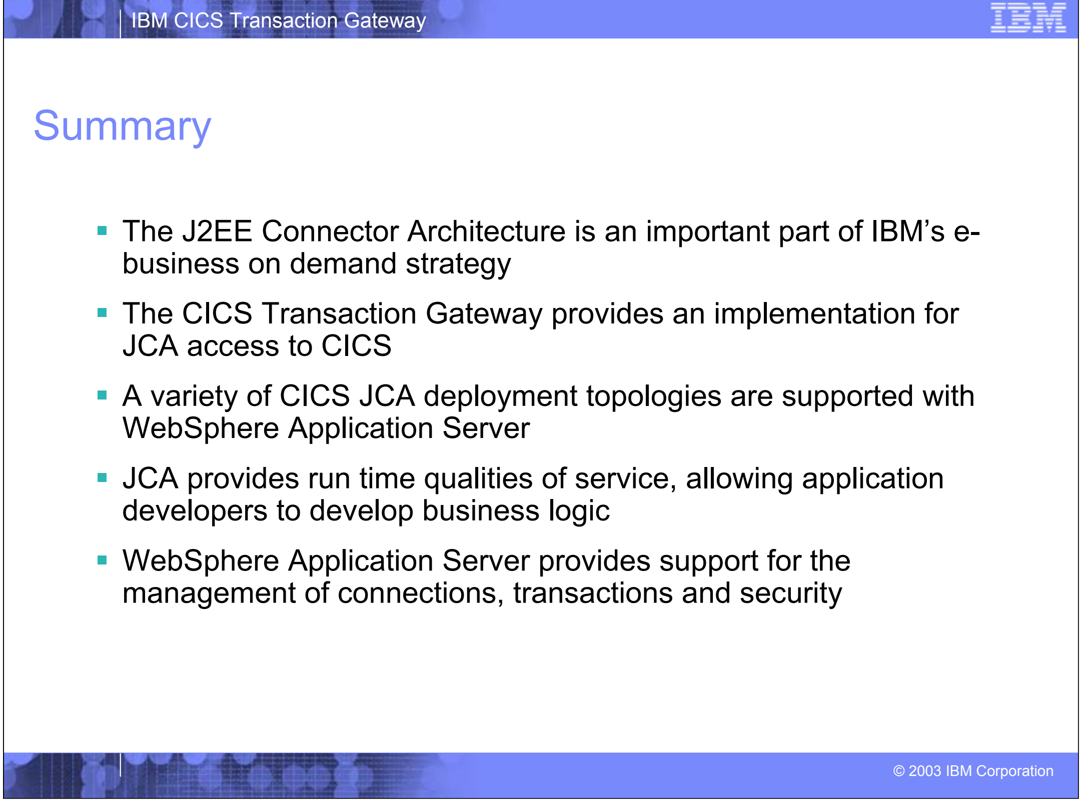
The specific qualities of service (in terms of the JCA system contracts) that apply to this topology are as follows.

Connection Management	Connection pooling of local in memory Connection objects
Transaction Management	Resource manager local transactions or global transactions with Last Participant Support (LPS)
Security Management	Component managed or container managed

When using WebSphere Application Server with Linux on zSeries, the zSeries HiperSockets feature can be used to provide efficient cross-memory TCP/IP connection into CICS regions running on z/OS

The JCA qualities of service for this configuration are almost identical to those described for topology 1 because Linux on zSeries (within a JCA and CICS Transaction Gateway scenario) can be treated as a distributed platform. A significant exception to this generalization is that IBM HiperSockets™ can be utilized to provide a highly efficient cross-memory transport for TCP/IP-based communication into CICS (using the ECI over TCP/IP function of CICS Transaction Server, Version 2.2 or 2.3).

In addition, a direct SNA connection is not supported when deploying the CICS Transaction Gateway on Linux. As a result, the TCP62 protocol must be used if support is required for EPI requests.



Summary

- The J2EE Connector Architecture is an important part of IBM’s e-business on demand strategy
- The CICS Transaction Gateway provides an implementation for JCA access to CICS
- A variety of CICS JCA deployment topologies are supported with WebSphere Application Server
- JCA provides run time qualities of service, allowing application developers to develop business logic
- WebSphere Application Server provides support for the management of connections, transactions and security

The J2EE Connector Architecture is an important part of IBM’s e-business on demand strategy. The CICS Transaction Gateway provides a JCA CICS ECI resource adapter that enables you to easily integrate J2EE applications running in WebSphere Application Server with existing CICS applications. The tooling capability of WebSphere Studio Application Developer Integration Edition builds on this capability by allowing integration of existing components as enterprise services in an overall service-oriented architecture. A variety of CICS JCA deployment topologies are supported with WebSphere Application Server, and the operational benefits of each environment are controlled by the system contracts specific to each topology. However, in all topologies the underlying infrastructure provided by WebSphere Application Server provides support for the management of connections, transactions and security. This support allows your application developers to focus solely on creating the business logic required for the integration of multiple components into an e-business on demand application.

More resources

This presentation is based on the White paper “Integrating WebSphere Application Server and CICS using the J2EE Connector Architecture” written by:

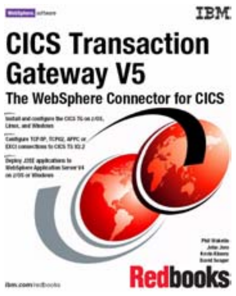
- Phil Wakelin, IBM Software Group - **Phil_Wakelin@uk.ibm.com**
- Nigel Williams, IBM Design Center for e-business on demand - **nigel_williams@uk.ibm.com**

This White paper will be available from the **www.ibm.com/cics/library**

Web site shortly after publication in March 2004.

More resources

- **There are a number of technical resources including Redbooks, manuals, configuration guides and data-sheets available from the following Web sites:**
  - CICS homepage www.ibm.com/cics
  - CICS Transaction Gateway homepage www.ibm.com/cics/ctg
  - WebSphere homepage www.ibm.com/websphere
- **The following IBM Redbooks may be of particular Interest:**
  - Java Connectors for CICS: Featuring the J2EE Connector Architecture, SG24-6401-00
  - CICS Transaction Gateway V5 The WebSphere Connector for CICS, SG24-6133-01



Trademarks