Options for integrating CICS applications in an SOA.

By Mark Cocker, CICS technical strategy and planning,
IBM Hursley Lab

September 2007

Contents

2 Executive summary
3 Technical transformation
4 CICS applications that can be transformed
7 Web-based access to CICS programs
9 Use the right architecture to connect to CICS Transaction Server
13 Strategic access options
22 Summary
23 For more information

Executive summary

Today, more and more companies are evolving to embrace business principles that enable them to integrate business processes end to end across the company and with key partners, so that they can respond flexibly and rapidly to new circumstances. As an IBM CICS® software user, you probably have a huge investment in CICS applications that play a critical role in your business processes, and must play an equally vital role in your business initiatives.

In moving to an environment that allows this kind of agility, your business must undergo both organizational transformation (moving from independent departments to shared business resources) and technical transformation (moving from discrete applications to connected and interdependent IT components). By adopting open standards, you enable your client and server components to be hosted in the environment most appropriate to their requirements, while still being able to interact easily – independent of hardware, runtime environment and programming language.

However, most IT systems – including CICS applications – were not designed with these objectives in mind. These systems constitute tightly coupled and highly optimized core assets, supporting critical processes that must be protected from disruption. Such factors sometimes make it difficult to change these applications; yet rewriting them is generally too costly, time-consuming or risky to be a practical option.

This white paper shows how the robust CICS systems that you rely on today can easily deliver your business applications to modern service oriented architectures (SOAs), with first-class support for Web services, Java™ 2 Platform, Enterprise Edition (J2EE), IBM WebSphere® MQ and other access options. IBM CICS Transaction Server enables you to use programs without changing them, combining them into higher-level services using graphical mapping and modeling tools. Over time, you can develop brand-new services that meet your customers' demanding needs. By using an SOA approach, you can more-closely align your IT systems to the requirements of your business. You can also reap the benefits that come with more-adaptable IT systems together with cost savings and productivity improvements.

Technical transformation

A good starting point for this discussion is the IBM SOA reference architecture (shown in Figure 1), a technical framework for enterprise transformation that enables software to be delivered as reusable, shareable services. This architecture provides the ability to bridge disparate systems spread across your entire enterprise. And because its components are modular, you can start small and grow your implementation to cover your evolving integration needs, both internally and externally.

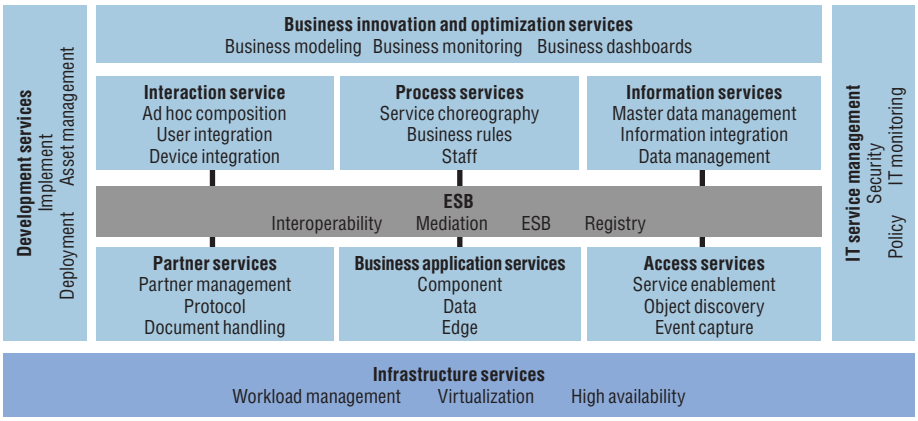

Figure 1. IBM SOA reference architecture

Within the reference architecture, there are three distinct styles of transformation.

User-interface modernization

This style transforms the user experience through facilities in the interaction services box. It aims to reach new customers while helping to improve productivity and reduce costs. Using this style of transformation can also help reduce training costs and increase overall user satisfaction. This method is the most accessible because it requires the lowest level of investment. You can achieve a rapid return on investment (ROI) through improved user interfaces with a modern interface design, and enhanced productivity with optimized interaction patterns.

Application integration

This style transforms application connectivity through facilities in business-application services, access services and process services. It aims to extend existing applications beyond their original designs to support integrated business processes, helping to reduce errors and development costs. You can turn existing applications into reusable services that can be accessed by a new set of users or reused to create new front-end business functions. The underlying principle – that you can reuse existing applications with little or no change – offers a lower-risk approach than a replacement strategy, which involves rewriting applications.

Service orientation

This style transforms the application architecture to provide greater responsiveness to business partners and customers. It involves some reengineering of the original application. Undoubtedly, this method requires higher investment of resources and time, but gives you the capability to create components from existing applications, which are more flexible and configurable for use in new applications. This reuse of business logic is called *componentization* and typically results in significant cost savings when compared with developing new application code.

This white paper describes the technical options that you can use to transform your CICS applications to deliver flexible reuse. It also explains the relative advantages of each option. Using these techniques, CICS Transaction Server enables you to progressively move your connectivity styles to suit different business and solution requirements.

CICS applications that can be transformed

Over the past 37 years, developers have created two major interfaces to CICS applications: CICS communications area (COMMAREA) programs and CICS terminal-oriented programs.

CICS COMMAREA programs receive requests and send responses through an area of memory called the *communications area*. CICS programs are primarily written in the COBOL language, but PL/I, C, C++, Java and REXX are also popular. CICS COMMAREA programs are similar to subroutines in that they are largely unaware of how they were invoked. As a result, they are often stateless, with CICS Transaction Server – on behalf of the program – managing the transactional scope and security context, which are typically inherited from the caller and a transaction definition. These programs typically expect the data in the COMMAREA to be formatted in their native language structures.

CICS Transaction Server for z/OS, Version 3 introduced the containers and channels programming model, which is more flexible and does not have the constraints of COMMAREAs.

CICS terminal-oriented programs are sometimes known as 3270 programs because they are designed to be invoked indirectly from an IBM 3270 display station or similar buffered terminal device. Invocation usually corresponds to a single interaction in a user dialog, starting with receipt of a message from the terminal and ending with transmission of a reply message to the same device. Input data from the terminal device is carried in a data stream, which the application acquires through a RECEIVE command. After processing, an output data stream is transmitted back to the terminal device through a SEND command. Terminal-oriented programs must be capable of analyzing device-specific input data streams and building output data streams to be transmitted to the terminal.

CICS also provides a facility known as *basic mapping support (BMS)*, which helps simplify application programming for terminal-oriented programs. This facility enables the programmer to define a static layout for each window to be displayed, with identified fields for dynamic content acquired through a RECEIVE MAP command. BMS then analyzes the data stream and returns record-formatted data to the application. Similarly, the application presents output data in record format using a SEND MAP command, which causes BMS to build an output data stream for the terminal. Application programmers widely use BMS because it frees them from having to know device specifics and as a result, enables applications to be device-independent to some degree.

Best practice in CICS application design for a number of years has been to separate an application into the following key elements (see Figure 2):

- *Client adapter or presentation logic*
- *Integration logic*
- *Business logic*
- *Data-access logic*

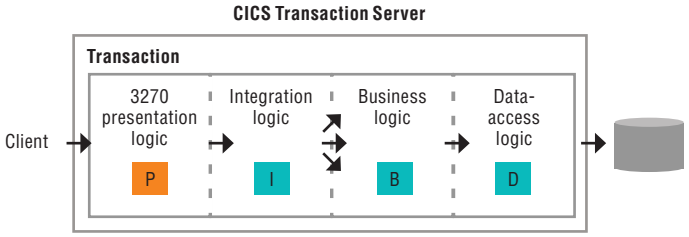

Figure 2. Separating key application elements promotes reuse and flexibility.

This separation provides a framework that enables reuse of business logic and data-access logic programs as subroutines within a larger application, as well as reuse with alternative implementations of client and presentation logic (for example, a Web service, Web browser or CICS terminal-oriented program). It also allows each program to be developed and optimized for the best ROI. Many CICS programs have already been written this way. However, you might have a number of programs that do not have such a clear separation of concerns, combining presentation logic (denoted as P in Figure 3) and business logic (B) into a single program for which there is only a CICS terminal-oriented-program interface. CICS Transaction Server provides a Link3270 bridge function that neatly addresses this problem (see Figure 3). The client uses the Link3270 bridge to run CICS terminal-oriented-program transactions by linking to the program DFHL3270 and passing a COMMAREA that includes the transaction identifier and the data to be passed to the application. The response contains the CICS terminal-oriented-program screen-data reply. If the target application used BMS, this information is presented in the form of an *application data structure (ADS)*, which is another name for the symbolic map that is generated by the BMS macros used to define the mapping of the CICS terminal-oriented-program window. No changes are required for the existing application code, and knowledge of CICS terminal-oriented-program data streams is usually not needed. As a result, the Link3270 bridge provides a programmatic interface for an important class of terminal-oriented programs, enabling them to be reused without resorting to less-efficient and more-fragile screen scraping.

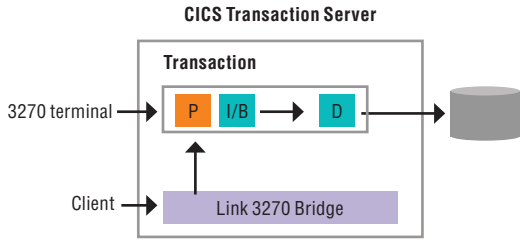

Figure 3. Access options provided by CICS Transaction Server facilitate effective reuse of existing terminal-oriented programs.

Historically, many CICS terminal-oriented-program transactions were written as pseudo-conversations, consisting of a number of terminal-oriented programs that run in a defined sequence. Each program in a pseudo-conversation displays data to a user and then stops, leaving only a small amount of state data to be picked up by the next program in the sequence, which is initiated by the next input data received from the user's terminal. The Link3270 bridge is able to fully reuse these pseudo-conversational transactions.

Also, CICS programs are typically grouped into application suites, or components, for performing a common set of business actions. Identifying the CICS programs that provide flexible public interfaces and understanding these interfaces is the first key step in reuse. The next step is to choose the best access options to support your solution.

Web-based access to CICS programs

Today, CICS COMMAREA, and container and channel programs, can be accessed in a variety of ways from clients running on a wide range of platforms. Typical client types include:

- *A Web service requester*
- *A Java servlet or Enterprise JavaBeans (EJB) running in a J2EE application server*
- *A C++ application running in a Microsoft® .NET environment*
- *A Web browser*
- *IBM WebSphere MQ*

In most cases, connections from these clients use a combination of external connectors, internal adapters and Internet Protocol (IP)-based communications. For example, a terminal-oriented program and a Web service requester can access the same integration logic (see Figure 4). An adapter is simply a program that receives the request and converts the data from an external format to the internal format used by CICS COMMAREA programs.

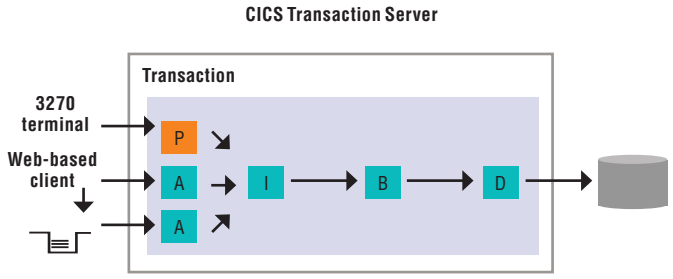

Figure 4. Access options provided by CICS facilitate effective reuse of existing business logic.

An external connector provides a remote call interface and implements a private protocol to invoke an application running under CICS Transaction Server. You must also use an external adapter to convert data from its external format to the COMMAREA format used by your programs in CICS Transaction Server. The most well-known example of an external connector is IBM CICS Transaction Gateway, which implements the Common Connector Interface (CCI) specified by the Java Connector Architecture (JCA), and is used with adapters implemented as Java beans.

An internal adapter is runtime code, possibly generated by a tool such as Rational Developer for System z™, that converts from one request format to another, such as converting XML to a COMMAREA. You can implement the adapter in any language supported by CICS and make it independent of the specific protocol used.

Along with these techniques, you can choose to create a standard IP-based adapter that uses a specific transport, such as WebSphere MQ, HTTP and TCP/IP sockets. This approach might be the only available option that supports unique types of clients, and it can permit greater control. However, this control must be balanced against additional development effort, and a loss of generality and reuse, because you can use the adapter only with a specific transport protocol.

Your preferred architectural approach is a key decision because of its effect on the costs of developing the solution and its long-term ROI. However, business factors such as existing development processes and the availability of skills might be as significant as technical factors influencing this decision. It is important to recognize that there is no one right answer suitable for all solutions.

Use the right architecture to connect to CICS Transaction Server

In investigating the appropriate architecture to use to connect to CICS Transaction Server, you must take into account business considerations, as well as the technical requirements of the solution, and compare these to the capabilities of the access options. Business considerations typically include:

- *Your organization's standards or reference frameworks*
- *Your organization's preferred application-development environment and tools*
- *The availability of skills*
- *Time constraints and required development effort for a solution*
- *Service-level agreements*

The technical requirements you should consider for the solution include:

- *Security*
- *Transactional scope*
- *Performance*
- *Reliability, availability and scalability*
- *Granularity of the interface*
- *Synchronous or asynchronous invocation*
- *Client-server coupling*
- *Inbound and outbound capability*
- *Data conversion*
- *State management*

Security

The most common security requirements are to authenticate users and middle-tier servers and to encrypt data flows. Simple user ID and password authentication is still widely used, although x.501 client certificates, Kerberos tickets and other schemes are becoming popular. Regardless of the technique you adopt, the user's credentials must eventually be mapped to an external security manager (ESM) user ID to support the authorization and auditing requirements that normally apply to CICS applications.

Transactional scope

This requirement refers to the capability of a given access option to support local transactions (one-phase commit), enabling a number of updates performed by CICS applications to be processed as a single unit of work; or global transactions (two-phase commit), enabling a client to coordinate updates performed by CICS Transaction Server with updates to resources made elsewhere. If neither is supported, updates performed by the CICS application are updated before returning the response to the client (referred to as *sync on return*).

Performance

Response time and cost-per-transaction are important aspects of performance in a production system. CICS Transaction Server helps minimize their effects and is highly optimized for traditional styles of access, such as CICS terminal-oriented-program access over a Systems Network Architecture (SNA) network. However, most solutions require other elements (such as connectors, adapters, encrypted data flows or new data-stream architectures), which impose an overhead on the implementation of the target business program. This overhead can be characterized by expressing it as a percentage of the base implementation cost of the target program, which is typically 1- to 2-million processor instructions million processor instructions for an average CICS transaction that is written in COBOL and that updates Virtual Storage Access Method (VSAM) records.

Reliability, availability and scalability

The access option that you choose should support your business goals and your organization's service-level agreements. You should also take workload management, monitoring, failover, automation and problem-analysis capabilities into consideration. CICS Transaction Server, Version 3.2, introduces supports for the new Enterprise Workload Manager (EWLM) for z/OS, which is part of the IBM Virtualization Engine™ platform. It enables you to define performance objectives and monitor and manage performance for workloads that run across different platforms in the IBM eServer® family. Because it runs on many types of servers, EWLM can be used for end-to-end workload monitoring in distributed environments that contain multiple, interacting server products. EWLM and z/OS Workload Manager (WLM) can run simultaneously; EWLM monitoring does not affect IBM z/OS® management and monitoring.

Granularity of the interface

Some access options, such as Web services, lend themselves to handling high-level, or coarse-grained, business requests. An example of this is a single request for all outstanding customer orders where the result includes the status of 30 orders and the details of each order. Other access options, such as JCA, lend themselves to handling fine-grained requests. An example of this kind of request is where the first request is sent to obtain a list of the 30 outstanding orders, then 30 additional requests are sent for the details of each order. Per request, fine-grained access is typically more efficient. However, when you take the end-to-end solution into consideration, processing a single coarse-grained request is likely to be more efficient in resources and response times than many fine-grained requests.

Specifically for CICS Transaction Server, IBM Rational Developer for System z features a new, intuitive service-flow modeler tool that enables one or more fine-grained COMMAREA programs and terminal-oriented programs to be aggregated into a single coarse-grained program. The resulting service flow can be deployed within CICS Transaction Server, Version 3 and exposed as a Web service or as a callable COMMAREA program.

Synchronous or asynchronous invocation

The majority of access options support synchronous invocation, meaning that a client request receives a single reply from CICS Transaction Server and the client waits for the reply. With asynchronous invocation, CICS Transaction Server generates an immediate response confirming that the request has been received, as well as one or more deferred responses containing replies to the original request. Typically, the client system continues processing as soon as the confirmation response is received and does not wait for the deferred response.

Client-server coupling

Some access options are described as tightly coupled, whereas others are described as loosely coupled. *Tight coupling* implies that the client and server impose dependencies and assumptions on each other that need to be managed and tested. For example, you must write requests for EJB components in CICS Transaction Server in Java, and the method signature of each request must exactly match that of the EJB public interface. *Loose coupling* implies that the client and server are free to use dissimilar technology and have very little dependency upon each other.

In general, access options that are loosely coupled are more robust. Outages, software upgrades and other operational events have less impact on the ability of applications to interoperate, whereas tightly coupled systems are optimized for runtime performance and control.

Inbound and outbound capability

Although it might be sufficient for CICS applications to simply be invoked as services by new applications that require access to existing and proven capabilities, the more-general case is for CICS applications to make calls to services hosted elsewhere, as well as being called themselves. This method provides a much more flexible and cost-effective approach to new business models.

Data conversion

Some access options require application data to be exchanged in specific formats, while other options negotiate at run time or leave it up to developers to enable the formats to interoperate. However, the IBM z/OS® operating system has historically processed and stored text data in EBCDIC codepages and numeric data in big-endian or packed-decimal format. As a result, most access options involve some data conversion. In some cases, the client is expected to convert data, for example using JCA, whereas other access options, such as Web services, can take advantage of CICS Transaction Server to efficiently to perform this conversion. In addition, modern databases can store data in the form of XML, and today's compilers enable applications to easily process XML documents.

State management

State management is the process by which you maintain information about the state of an application between multiple invocations of that application. Depending on the architecture or transport, CICS Transaction Server provides a means to preserve the state between invocations if the system design requires it.

Strategic access options

One of the great strengths of CICS Transaction Server is its flexibility to deliver your applications to a range of client types in harmony. CICS Transaction Server has a broad choice of access options that are based on TCP/IP and support the needs of the majority of clients. For convenience, they are divided here into standard architectures and standard transports.

Standard architectures provide comprehensive development tools and runtime support in CICS Transaction Server, including:

- *Web services*
- *JCA*
- *EJB*

Standard transports are suitable for use by applications that require greater control of the protocol and do not need the development tools or qualities of service provided by the standard architectures. These applications assume more responsibility for systems management, security and recovery. The standard transports include:

- *IBM WebSphere MQ*
- *HTTP*
- *TCP/IP sockets*

Other options continue to be supported for their unique capabilities and to maintain backward compatibility. Refer to CICS Transaction Server documentation for details about these technologies:

- *Advanced Program-to-Program Communication (APPC)*
- *External CICS interface (EXCI)*
- *Front-end programming interface (FEPI)*
- *External call interface (ECI)*
- *External presentation interface (EPI)*

Web services

The Web services support in CICS Transaction Server, Version 3 enables your programs to be Web service providers or requesters. This support provides a number of standards including SOAP, Web-services distributed transactions (WS-AtomicTransaction) and Web Services Security (WS-Security), and conforms to the Web Services Interface (WS-I) basic profiles. CICS Transaction Server provides tools to either produce a Web Services Description Language (WSDL) file from a language structure used by an existing COMMAREA program, or to produce a language structure from a WSDL file.

The SOAP request is received by the HTTP listener in CICS Transaction Server or the WebSphere MQ trigger monitor (see Figure 5). In both cases, the request is passed to a pipeline to process the SOAP headers and set up the transaction and security environment. CICS Transaction Server can convert the incoming XML request into a language structure (such as COBOL record format), or you can use tools to generate a custom data mapping or have CICS Transaction Server pass the XML data to the target CICS program. After the program has completed the request, the response is passed back to the pipeline to be wrapped in SOAP headers and returned to the client.

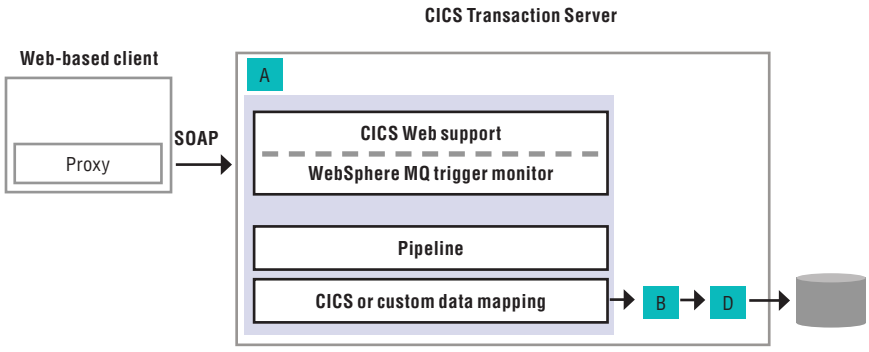

Figure 5. Web services capabilities are now fully integrated into CICS Transaction Server, Version 3.

Where advanced data mapping is required between XML data types and COBOL language structures, Rational Developer for System z provides a visual editor. It also provides the ability to generate WSDL files, CICS WSBIND resources and programmatic XML converters.

You can use Web services to interact in a highly secure and reliable manner, independent of platform, environment or application language. Developers can rapidly build open-standards-based applications independent of the CICS business-logic program they will interact with. This logical separation makes Web services a loose-coupling architecture. In addition, IBM WebSphere Service Registry and Repository can be used to host WSDL descriptions of services and provides governance features to manage its access and life cycle.

JCA

JCA defines a standard for connecting from the J2EE specification to heterogeneous enterprise information systems (EISs), such as CICS Transaction Server. CICS Transaction Gateway provides a JCA connector for CICS Transaction Server as a resource adapter (see Figure 6). The resource adapter plugs into the J2EE application server, providing connectivity between the J2EE application, the application server and CICS Transaction Server.

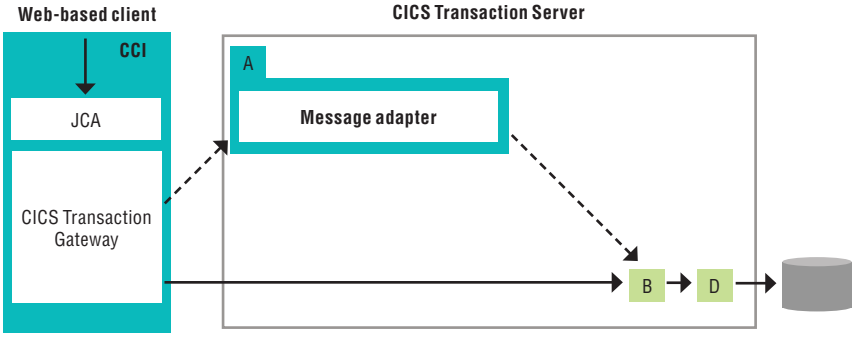

Figure 6. CICS Transaction Gateway provides JCA access to CICS Transaction Server.

JCA defines the CCI that the client uses to drive interactions. It also supports transactional coordination either by XA global transactions (two-phase commit) or by resource manager local transactions (one-phase commit). The CICS ECI resource adapter supports XA global transactions when used in conjunction with CICS Transaction Gateway, Version 6.1 on z/OS or when used with IBM WebSphere Application Server for z/OS. For further details about JCA and transaction integration with CICS Transaction Server, refer to the paper *Integrating WebSphere Application Server and CICS using the JCA*, available at **ibm.com**//software/htp/cics/library/rmp/overviews/ibm_ rmp_ integrating_was_and_cics_using_jca.html.

The J2EE application can invoke the CICS business-logic program directly if no message transformation is required. In this case, you can use IBM Rational® Application Developer software to create a Java bean to represent and convert data to the COMMAREA of the program, with Java methods for getting and setting field values.

A message adapter in CICS Transaction Server is required only if the message is to be transformed: for example, if the request is in XML and the CICS business logic program requires a COBOL record format. The JCA connector provided by CICS Transaction Gateway is an effective replacement for ECI Java classes, and has a 32 KB limit on message size. JCA is considered a medium-coupling architecture, because messages are normally flowed as COBOL types.

CICS Transaction Gateway is a high-performing, highly secure and scalable access option with tight integration to existing CICS applications. Because CICS Transaction Gateway is easy to install, has flexible configuration options and requires minimal changes to CICS Transaction Server and in most cases, no changes to existing CICS applications, it provides an attractive option for integrating existing CICS applications into your Web-based architecture. Also, CICS Transaction Gateway supports a range of non-Java clients, including C, C++, COBOL and COM.

EJB components

EJB components are able to invoke methods on remote EJB objects across a network. The EJB client can call a remote object after it has obtained a reference to it, either by looking it up in a naming service, such as Lightweight Directory Access Protocol (LDAP), or by receiving the reference as an argument, a return value or from a cache. The Java Object Request Broker (ORB) uses object serialization to transparently marshal and unmarshal parameters supporting true object-oriented polymorphism. EJB support in CICS Transaction Server can include Secure Sockets Layer (SSL) encryption, full transaction coordination (two-phase-commit protocol) and the flowing of a user's J2EE security role.

When an EJB request is received, a Java virtual machine (JVM) is started in CICS Transaction Server, and the ORB built into the JVM decodes the Remote Method Invocation (RMI). The ORB, in turn, calls the appropriate methods of the EJB session bean to process the request (see Figure 7). If the session bean is to call a business-logic program, the CCI application programming interface (API) can be used. It is the same CCI API as provided by CICS Transaction Gateway, but in this case, the JCA resource adapter is provided by CICS Transaction Server and does not involve network flows. If the message needs to be transformed, a message adapter can be called before calling the business logic.

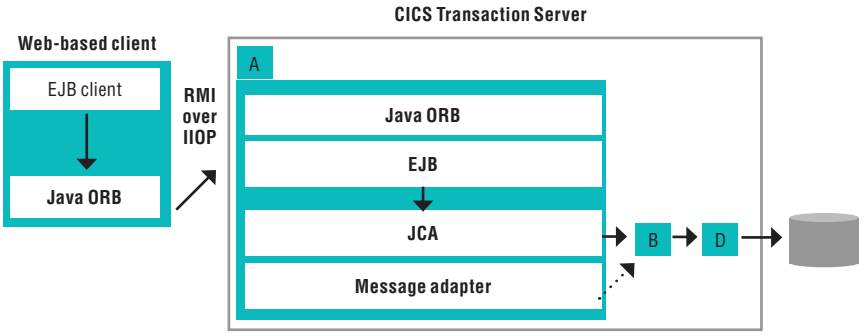

Figure 7. EJB support in CICS Transaction Server allows Java programs on any platform to easily invoke EJB components in CICS Transaction Server.

You can choose to have the session bean in CICS Transaction Server be stateless between invocations, or stateful, in which case CICS Transaction Server saves the session bean data (passivate) into a VSAM file and restores the state (activate) automatically. EJB components provide a tightly coupled connection because both ends must be implemented by compatible J2EE technologies and EJB interfaces.

WebSphere MQ

WebSphere MQ software enables you to easily exchange information across different platforms, integrating existing business applications in the process. WebSphere MQ provides the assured delivery of messages, dynamically distributes workload across available resources and helps make programs portable (see Figure 8). CICS Transaction Server for z/OS, Version 3.2 now makes it easier to install and more efficient to run WebSphere MQ workloads.

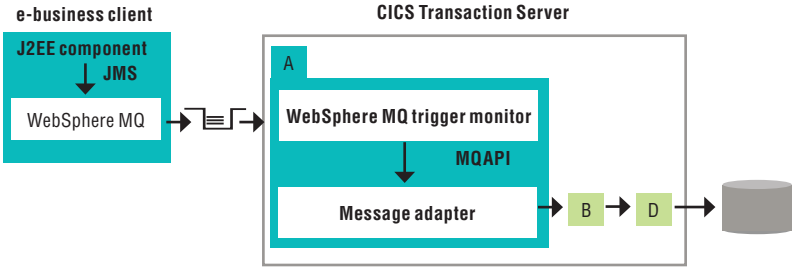

Figure 8. WebSphere MQ provides assured delivery of messages from many platforms to enable efficient, asynchronous access to CICS Transaction Server.

WebSphere MQ provides Java Message Service (JMS) APIs and native WebSphere MQ APIs that can be used by clients on a wide variety of platforms, with many options for routing and encrypting messages prior to arriving on IBM WebSphere MQ for z/OS. The WebSphere MQ trigger-monitor program runs in CICS Transaction Server, and according to the queue definitions, as messages arrive, it starts the appropriate message-adapter program in a new transaction. The message adapter uses WebSphere MQ native APIs to receive the message, transform it if required and call the business-logic program. A reply message can be sent using the reply-to queue defined in the message. For efficiency, the message-adapter program usually continues to process messages on the inbound queue until it is empty. You can also use WebSphere MQ for pseudo-synchronous messaging, but you might need to consider how to handle error and compensation situations.

The WebSphere MQ DPL bridge for CICS Transaction Server provides a second option (see Figure 9). This generic adapter passes a message from a named input queue to a business-logic program through the COMMAREA. This is ideal in the situation where the client can format the message into a form acceptable by the business-logic program.

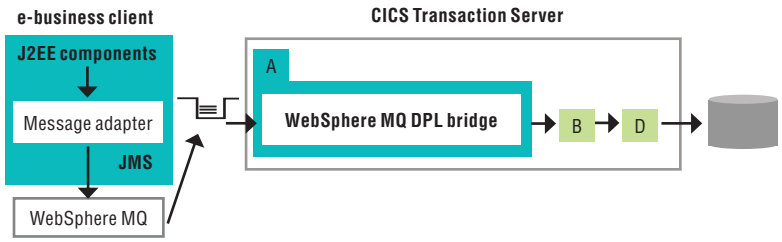

Figure 9. The WebSphere MQ DPL bridge can link to existing CICS programs without the need for a WebSphere MQ trigger-monitor program.

HTTP

The adapter component is made up of the HTTP listener in CICS Transaction Server (called *CICS Web support*) and a message adapter. CICS Transaction Server supports HTTP basic authentication for user identification, or the more-secure SSL encryption and authentication with client and server certificates

CICS Transaction Server, Version 3 introduces Uniform Resource Identifier Map (URIMAP) resource definitions to decide how to process the request, including which message adapter to call, and what transaction and security environment is required. The message adapter uses CICS Web APIs to extract the HTTP user data, which is usually formatted as XML or HTML form data. The message adapter has access to the HTTP and TCP/IP headers if required. The message adapter transforms this information into a COMMAREA and calls the business-logic program.

The response message is also normally formatted as HTML or XML, and the message adapter can use the CICS document APIs to easily merge static HTML or XML with dynamic data from the business-logic program. The response is returned to the client for display or processing. To support Web browsers, the solution should include Web servers to support static content such as pictures and cascading style sheets (CSSs).

HTTP is synchronous and stateless. However, if state management is required, CICS Transaction Server provides a utility for storing state data indexed by a state-management token that the HTTP client can return on subsequent calls to retrieve the state.

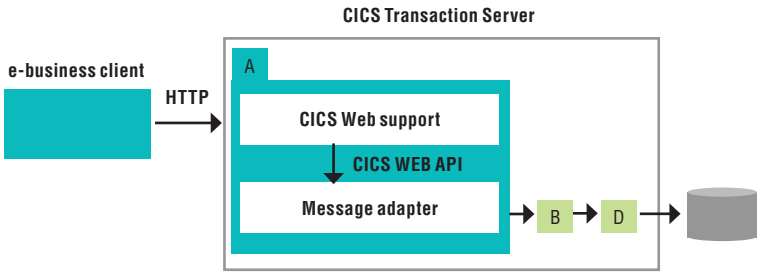

Figure 10. CICS Transaction Server supports HTTP clients and Web browsers with user-friendly Web and TCP/IP APIs, and it can construct HTML and XML responses with the document API.

TCP/IP sockets

The TCP/IP socket interface for CICS Transaction Server (referred to as CICS sockets) is provided by IBM z/OS Communications Server and supports peer-to-peer applications in which both ends of the connection are programmable. CICS sockets provide a variant of the Berkeley Software Distribution 4.3 Sockets interface, which is a low-level API with built-in support for SSL and Transport Layer Security (TLS), but it does not support distributed transactions or systems management.

CICS sockets provide a concurrent listener, EZACIC02, or you can write your own concurrent or iterative listener to meet your needs. The listener and the child server use the CICS sockets APIs to receive and send data, and perform general communications-control functions (see Figure 11). You can write the programs in COBOL, PL/I, assembler language or C. You can also write client adapters to create new outbound connections.

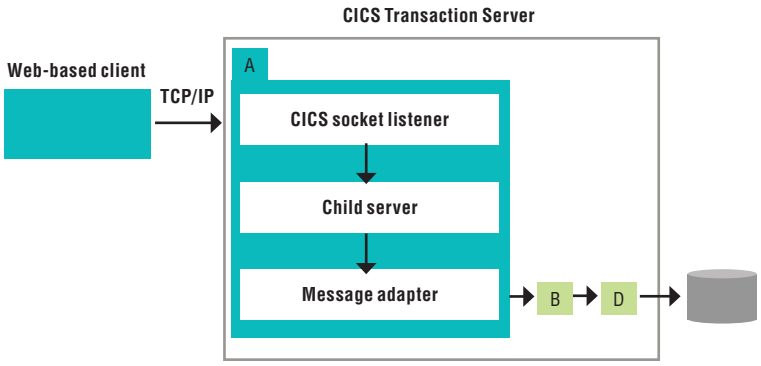

Figure 11. CICS sockets provide a completely programmable solution where other access options are not suitable.

Summary

To achieve the best reuse of your CICS programs and to support multiple access options, build clear and concise business-logic interfaces of the right granularity for your solution. CICS Transaction Server and other tools help transform and aggregate your programs to achieve this goal. CICS Transaction Server provides a broad choice of access options based on TCP/IP, open-standards-based connectivity architectures and transport mechanisms. These capabilities enable applications in CICS Transaction Server to be full participants in your SOA environment.

You should consider business and technical requirements for an end-to-end solution when selecting the most appropriate connectivity option. CICS developers and system programmers are likely to find it easy to take advantage of new access options and make existing applications available for reuse.

CICS and WebSphere products are strategic middleware products that:

- *Interoperate through Web services, JCA, EJB and WebSphere MQ.*
- *Use and complement z/OS qualities of service.*
- *Provide high quality of service, low cost per transaction and top-notch security.*

To help with your CICS integration projects, consider the following two tables which are designed to encapsulate the options available to you, and recommendations for each one. Table 1 summarizes the capabilities of each integration option. Compare your business environment and technical end-to-end solution requirements with the information about each access option to select the most appropriate one for your business. Table 2 presents general recommendations for each access option.

Table 1. Comparison of capabilities for access options

Standard architecture	Capabilities	Security to IBM eServer® zSeries®	Transactional scope	Interface	Coupling
Web services	• Inbound and outbound • Synchronous (HTTP) • Asynchronous (WebSphere MQ) • EWLM	• Web services* • WS-TRUST • WS-Security • SSL • User ID and password	• Web services • Sync on return	• COMMAREA • CONTAINER	Low
JCA	• Inbound only • Synchronous • Asynchronous • 32 KB maximum message size	• SSL • User ID and password • Thread identity	• Local • Global • Sync on return	COMMAREA	Medium
EJB	• Inbound and outbound • Synchronous • EJB state management • EWLM	• EJB security roles • SSL	• Sync on return • Global	EJB session bean	High
Standard transport					
WebSphere MQ	• Inbound and outbound • Asynchronous • Assured delivery	• SSL • User ID and password	Sync on return	• COMMAREA • WebSphere MQ API	Medium
HTTP	• Inbound and outbound • Synchronous • EWLM	• SSL • User ID and password	Sync on return	CICS Web API	Medium
TCP/IP sockets	• Inbound and outbound • Synchronous and asynchronous	• SSL • User ID and password	Sync on return	CICS sockets API	High

Table 2. Recommendations for integrating CICS applications into your business

Standard architecture	Description	Positioning	Recommendation
Web services	Comprehensive World Wide Web Consortium (W3C) standards for messaging over the Web, supporting SOA to and from CICS Transaction Server	Industry-wide, open-standards-based integration technology that includes CICS connectivity; designed to improve quality of service, features and performance	Establish plans to transform CICS applications so they can participate in an SOA pattern with Web services
JCA	Lightweight J2EE standard for calling CICS and other EISs	Widely adopted CICS connectivity option with high qualities of service	Continue to use JCA and CICS Transaction Gateway within an SOA and ESB where fine-grained access and high qualities of service are required
EJB	Comprehensive J2EE standard for J2EE components, including EJB in CICS Transaction Server	Niche technology that provides highly functional, standards-based connectivity to CICS Transaction Server	Limit to applications that can benefit from consistent J2EE connectivity; continue to use Java as an application language
Standard transport			
WebSphere MQ	Comprehensive industry standard for assured messaging	Widely adopted business-to-business integration technology that includes CICS connectivity	Continue to use WebSphere MQ for basic messaging and flowing Web services
HTTP	Lightweight W3C standard for communications over the Web	Industry-wide, open-standards-based technology; ubiquitous for direct Web-browser connection and the basis for Web services	Use for Web services and to support browsers for niche applications
TCP/IP sockets	Lowest common denominator for CICS connectivity	Mature technology that provides basic and flexible connectivity to CICS Transaction Server	Limit to specialized applications; plan to adopt Web services

For more information

To learn more about IBM CICS Transaction Server, contact your IBM representative or IBM Business Partner, or visit:

ibm.com/cics

To learn more about the topics raised in this paper, please refer to the IBM Redbooks® publication *Architecting Access to CICS within an SOA*, available at:

ibm.com/redbooks/abstracts/sg245466.html?Open

To learn more about IBM WebSphere Application Server, contact your IBM representative or IBM Business Partner, or visit:

ibm.com/software/webservers/appserv/was/

To learn more about IBM WebSphere MQ, contact your IBM representative or IBM Business Partner, or visit:

ibm.com/software/integration/wmq/

To learn more about IBM Rational Developer for System z, contact your IBM representative or IBM Business Partner, or visit:

ibm.com/software/awdtools/devzseries

For a detailed description of integrating WebSphere Application Server and CICS Transaction Server using the JCA, visit:

ibm.com/software/htp/cics/ctg/library/#wpapers

IBM United Kingdom Limited
Hursley Park
Winchester
Hampshire
SO21 2JN
United Kingdom

Produced in the United States of America
09-07

WSW11339-USEN-00