CICS solutions
White paper

Delivering e-business access to CICS: strategic options.

By Dr. Geoff Sharman, Mark Cocker,
IBM Software Group

June 2004

Contents

What is e-business access?

Companies today are increasingly evolving to on demand business: the ability to accelerate value by connecting business components with suppliers, partners and customers, enabling differentiation and rapid response to opportunities and threats. In moving to an on demand operating environment, 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 information technology components. The resulting on demand operating environment is based on open standards, integrated through loose coupling, virtualization and autonomic computing.

Most existing IT systems, however, were not designed with these objectives in mind. These systems constitute core assets that companies rely on to run their businesses, supporting mission-critical processes that must be protected from disruption. These factors make it difficult to change existing applications, yet rewriting these applications is generally too costly, time-consuming or risky to be a practical option.

Therefore, sound technical transformation strategies must include comprehensive plans for managing and reusing existing applications, data and skills. The most important goal for this transformation is to enable access to these applications through open and standard interfaces, enabling them to be integrated within a new enterprise IT architecture. This is referred to as *enterprise transformation*. Three possible styles of transformation include *improve, adapt and innovate.*

Improve, the first style of transformation, focuses on reducing costs and increasing productivity by enhancing the user interface. 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 end-user interfaces with a modern look-and-feel, and enhanced productivity with optimized interaction patterns. This can also result in lower training costs and increased overall user satisfaction.

Adapt, the second style of transformation, focuses on extending existing applications beyond their original design to create integrated solutions that yield significantly greater business value and process flexibility. Applications can be turned into reusable services accessed by a new set of users or reused from new front-end business functions. The underlying principle – that you can reuse existing applications with little or no change – means this is a lower-risk approach than a replacement strategy, which involves rewriting applications.

Innovate, the third style of transformation, 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 fully flexible and configurable for use in new applications. This reuse of business logic is called *componentization* and might result in significant cost savings when compared with developing new application code.

A key concept of the adapt style of transformation is that you can reuse applications through programmatic interfaces that enable the application to be invoked locally or remotely, either through a standardized application programming interface (API) or through a standardized network protocol.

This white paper offers an overview of the programmatic interfaces provided by IBM CICS® Transaction Server for z/OS and related products to support the adapt enterprise transformation style.

What CICS assets can be transformed?
Over the past 35 years, developers have created two major types of CICS applications, or assets.

- *CICS COMMAREA programs*
- *CICS terminal-oriented programs*

CICS COMMAREA programs receive requests and send responses through an area of memory called the COMMunications AREA. CICS programs can be written in COBOL, PL/I, C, C++, Java™ and other languages. In general, CICS COMMAREA programs are similar to subroutines in that they are unaware of how they were invoked. They are often stateless, with CICS – on behalf of the program – automatically managing the transactional scope and security context, which are typically inherited from the caller and a transaction definition.

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 an end-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 datastream, which the application acquires through a RECEIVE command. After processing, an output datastream is transmitted back to the terminal device through a SEND command. Terminal-oriented programs must be capable of analyzing device-specific input datastreams and building output datastreams to be transmitted to the terminal.

CICS also provides a service known as *basic mapping support (BMS)*, which simplifies application programming for terminals such as the IBM 3270 Display Station. This enables the programmer to define a static layout for each screen to be displayed, with identified fields for dynamic content acquired through a RECEIVE MAP command. This in turn causes BMS to analyze the datastream and to return 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. BMS is widely used because it frees the application programmer from needing knowledge of device specifics and enables applications to be device-independent to some degree.

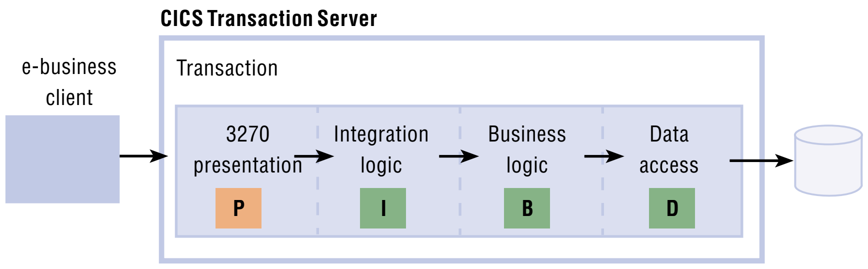

Figure 1. Separating key application elements helps enable maximum reuse and flexibility.

Best practice in CICS application design for a number of years has been to separate key elements of the application (see Figure 1).

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

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 presentation logic (for example, a Web service, Web browser or 3270 device). It also allows each program to be developed and optimized for the best ROI. Many CICS programs have already been written this way.

However, there are a number of programs that do not have such a clear separation of concerns, combining presentation logic (P) and business logic (B) into a single program for which there is only a 3270 interface.

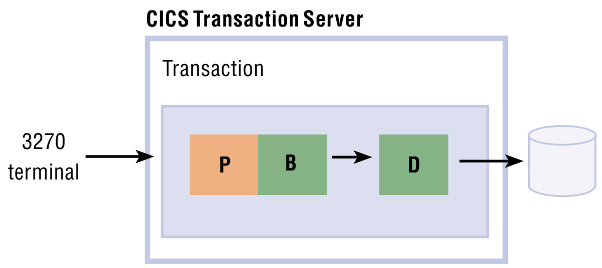

Figure 2. The Link3270 bridge function in CICS can reuse 3270 terminal-oriented programs.

CICS Transaction Server, Version 2 provides a Link3270 bridge function that neatly addresses this problem (see Figure 2). The client uses the Link3270 bridge to run 3270 transactions by linking to DFHL3270 and passing a COMMAREA that includes the transaction identifier and the data to be passed to the application. The response contains the 3270-screen data reply. If the target application used BMS, this is presented in the form of an application data structure (ADS), another name for the symbolic map that is generated by the BMS macros used to define the mapping of the 3270 screen. No changes are required for the existing application code, and knowledge of 3270 data streams is not generally needed. Thus, 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.

Historically, many 3270 transactions were written as pseudo-conversations, consisting of a number of terminal-oriented programs that execute in a defined sequence. Each program in a pseudo-conversation displays data to an end user and then terminates, leaving only a small amount of state data to be picked up by the next program in the sequence, which is initiated by input data received from the end user's terminal. The Link3270 bridge is able to fully reuse these pseudo-conversational transactions. There is also a companion product, IBM WebSphere® MQIntegrator® Agent for CICS, which has an intuitive visual interface that enables you to model a sequence of terminal-oriented programs and pseudo-conversational transactions that in turn makes use of the Link3270 bridge.

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 is to decide the best access options to support your e-business solution.

e-business access to CICS programs

Today CICS COMMAREA programs can be accessed in a variety of ways from client applications running on a wide range of platforms (see Figure 3). Typical e-business clients include:

- *Web service requester*
- *Java servlet or Enterprise JavaBeans (EJB) running in a Java 2 Platform, Enterprise Edition (J2EE) application server*
- *C# application running in a Microsoft® .NET environment*
- *Web browser*
- *Messaging middleware*

In most cases, connections from an e-business client will use a combination of:

- *External connectors*
- *Internal adapters (user written or generated by tools)*
- *Standard Internet Protocol (IP) based protocols*

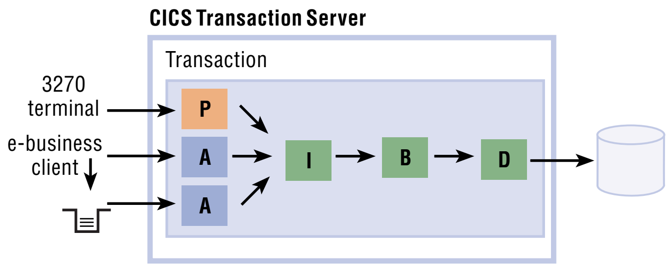

Figure 3. Access options provided by CICS facilitate effective reuse of existing business logic.

For example, a terminal-oriented program (P) and a Simple Object Access Protocol (SOAP) adapter (A) can access the same CICS business logic program (B). An adapter is simply a program that accepts the request and converts the data from an external format to the internal COMMAREA format used by CICS COMMAREA programs.

An external connector provides a remote call interface and implements a private protocol to invoke an application running under CICS. An external adapter must also be used. This converts data from its external format to the COMMAREA format used by CICS. The most well-known example of an external connector for CICS is CICS Transaction Gateway, which implements the Common Connector Interface (CCI) specified by the J2EE Connector Architecture (JCA), and is used with adapters implemented as Java beans.

An internal adapter is run-time code, possibly generated by a tool, that converts from one request format to another, such as converting SOAP over HTTP to a COMMAREA. The adapter can be implemented in any language supported by CICS and can be independent of the specific protocol used.

In addition to these techniques, customers can choose to create a standard IP-based protocol adapter that exploits a specific transport such as WebSphere MQ, HTTP and TCP/IP sockets. This approach might be the only available option that supports some types of e-business clients, and it permits greater flexibility in the functionality that can be implemented. However, this flexibility must be balanced against a loss of generality and reuse, because the adapter can be used only with a specific transport protocol.

The choice of architectural approach is a key decision because it might affect the costs of developing e-business applications and their long-term value. However, business factors, such as the availability of skills, might be as significant as technical factors influencing this decision. What is important to recognize is that there is no single right answer, just as there is no single programming language suitable for all applications.

The choice of architectural approach is a key decision because it might affect the costs of developing e-business applications and their long-term value. However, business factors, such as the availability of skills, might be as significant as technical factors influencing this decision. What is important to recognize is that there is no single right answer, just as there is no single programming language suitable for all applications.

Which architecture should be used to connect to CICS?

To answer this question, you need to articulate the application's functional requirements, and map these onto the capabilities of the access option. Your application is likely to be delivered across several e-business clients, and therefore, a range of access options is needed. The application functional requirements will typically encompass:

- *Security*
- *Transactional scope*
- *Performance*
- *Reliability, availability and scalability (RAS)*
- *Application interface*
- *Synchronous or asynchronous invocation*
- *Client/server coupling*

Security. The most common security requirement is to authenticate end users and middle-tier servers. Simple user ID and password authentication is still widely used, although x.501 client certificates, Kerberos tickets and other schemes are becoming popular. Whichever technique you adopt, the user's credentials must eventually be mapped to an external security manager (ESM) user ID to support the authorization and accounting 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 an external server to coordinate updates performed by CICS with updates to local resources held by that server.

Performance. Response time and cost per transaction are important aspects of performance in a production system. CICS seeks to minimize these and is highly optimized for traditional styles of access, such as 3270 terminal access over an System Network Architecture (SNA) network. However, most e-business solutions require additional elements (such as connectors, adapters, encrypted data flows or new datastream architectures), which are less optimized, and impose an overhead on the execution of the target business program. This overhead might be characterized by expressing it as a percentage of the base execution cost of the target program, which is typically one to two million processor instructions for a CICS/COBOL/VSAM terminal-driven program.

Reliability, availability and scalability (RAS). The access option should support the business goals and qualities of service requirements. The access option should also facilitate workload management.

Application interface. The access option needs to support the CICS program interface either directly or by using an adapter without imposing additional restrictions. The interface encompasses the data elements, the code page and the size of the message.

Synchronous or asynchronous invocation. The majority of access options support synchronous invocation, meaning that a client request receives a single reply from CICS and the client waits for the reply. In the alternative approach, known as asynchronous invocation, CICS 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, while others are described as loosely coupled. Tight coupling implies that the client and server share many assumptions and dependencies. Loose coupling is not a precise concept but refers to several possibilities:

- *The client and CICS may process concurrently.*
- *The client might use dissimilar technology from CICS.*
- *The client need not be maintained or upgraded in step with CICS.*

In general, access options that are loosely coupled are more robust. Planned or unplanned outages, software upgrades and other operational events have less impact on their ability to interoperate, while tightly coupled systems are optimized for run-time performance and operational control.

Strategic connectivity options
CICS has a range of access, or connectivity, options that are based on TCP/IP and support the diverse needs of e-business clients. For convenience they are divided here into:

Standard architectures that provide comprehensive development tools and runtime support in CICS.

- *SOAP*
- *JCA*
- *Enterprise JavaBeans*

Standard transports that 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 will assume more responsibility for systems management, security and recovery.

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

Other options are supported for their unique capabilities where the previous options cannot fulfill the requirements. You can refer to CICS documentation for details about these technologies.

- *APPC (advanced program-to-program communication)*
- *CICS EXCI (external CICS Interface)*
- *CICS FEPI (front-end programming interface)*
- *CICS ECI (external call interface)*
- *CICS EPI (external presentation interface)*

SOAP

The SOAP for CICS feature enables CICS programs to be a Web service provider or requester (see Figure 4). The SOAP service is fully defined in a Web Services Definition Language (WSDL) file. Usually, tools are used to import the WSDL file and generate a proxy for the e-business client to use to construct and send the SOAP message.

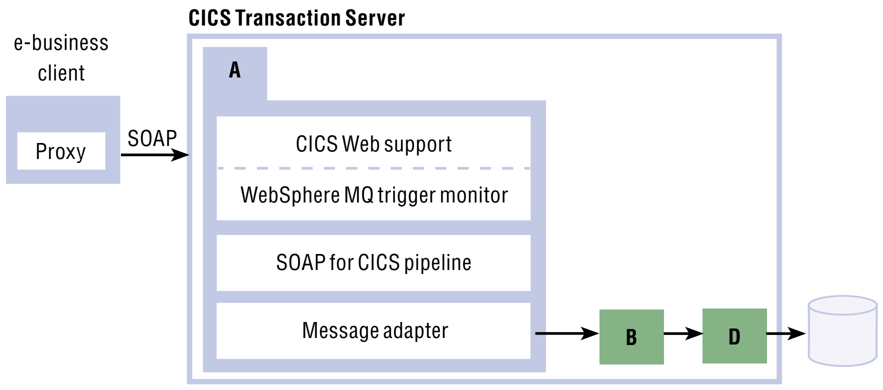

Figure 4. The SOAP for CICS feature provides direct access from Web service requesters and access to Web service providers.

The SOAP request is received by the HTTP listener (called CICS Web support) or the WebSphere MQ trigger monitor for CICS. In both cases, the request is passed to the SOAP for CICS pipeline to process the SOAP-architected headers, set up the transaction and security environment, and call the target message adapter with the XML request message. The message adapter transforms the XML message into a COMMAREA before calling the business logic. After the business logic completes, the message adapter creates an appropriate XML response message, which is passed back to the pipeline to be wrapped in SOAP headers and returned to the e-business client.

You can create the request and response XML schema definitions (XSDs) directly from the COMMAREA definition of the business logic program with tools such as IBM WebSphere Studio Enterprise Developer. A visual editor is provided in WebSphere Studio Enterprise Developer for the situation where a more complex mapping is required between an existing XSD file and an existing CICS COMMAREA program. WebSphere Studio Enterprise Developer also provides the means to generate the WSDL file and the SOAP message adapter to convert between XML and the COMMAREA.

The SOAP for CICS feature allows CICS programs to interact securely and reliably with Web services, independent of platform, environment or application language. Developers can rapidly build open-standards-based applications independent from CICS business logic. Loose coupling and interoperability are inherent in a service-oriented architecture (SOA) and make it a natural choice for many enterprise applications.

JCA

JCA defines a standard for connecting from the J2EE platform to heterogeneous enterprise information systems, such as CICS. The CICS Transaction Gateway provides the CICS JCA connector as a resource adapter (see Figure 5). The resource adapter plugs into the J2EE application server, providing connectivity between the J2EE application, the application server and CICS.

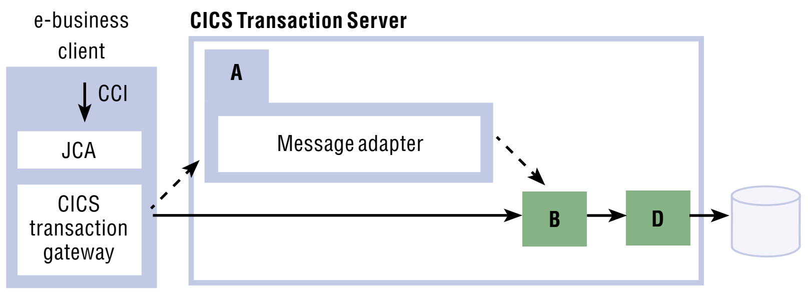

Figure 5. The CICS Transaction Gateway provides JCA access to CICS.

JCA defines the common client interface (CCI) for the e-business client to use to drive interactions. JCA supports transactional coordination either by resource manager local transactions or J2EE global transactions. The CICS Transaction Gateway JCA resource adapter supports two-phase commit global transactions in conjunction with IBM WebSphere Application Server for z/OS. In all other circumstances, it supports resource manager local transactions, which are one-phase commit transactions local to the CICS Transaction Gateway and the associated CICS region. For further details about JCA and transaction integration with CICS, refer to the paper "Integrating WebSphere Application Server and CICS using the JCA."

The J2EE application can invoke the CICS business logic program directly if no message transformation is required. In this case, IBM WebSphere Studio Application Developer Integration Edition can be used to create a Java bean to represent a COMMAREA formatted as COBOL types, with Java methods for getting and setting fields.

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

The CICS Transaction Gateway is the preferred implementation for JCA connectors to access all CICS servers from WebSphere Application Server, for e-business applications that require a high-performing, highly secure and scalable access option with tight integration to existing CICS applications. The CICS Transaction Gateway benefits from ease of installation and flexible configuration options and requires minimal changes to CICS and in most cases, no changes to existing CICS applications. In addition, the CICS Transaction Gateway supports a range of non-Java clients, including C, C++, COBOL and COM.

Enterprise JavaBeans

Enterprise JavaBeans enables Java clients to invoke methods on remote Java objects across a network. The Java 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 optionally supports Secure Sockets Layer (SSL) encryption, full transaction coordination (two-phase commit protocol) and the flowing of a user's J2EE security role.

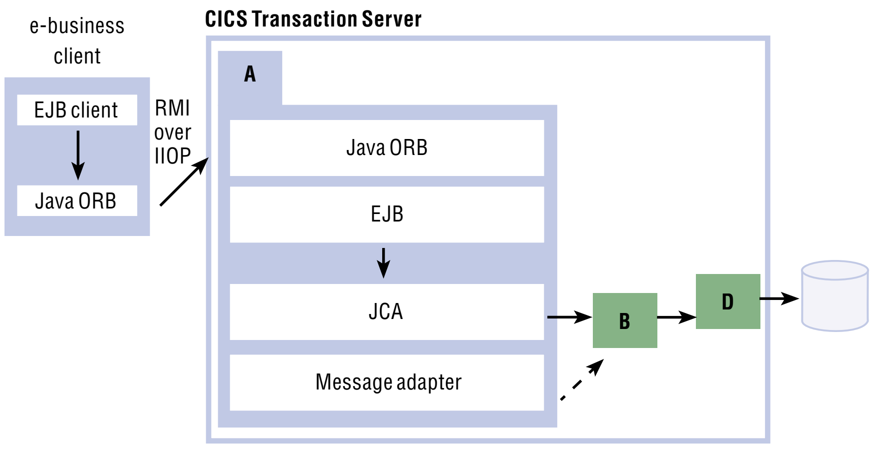

Figure 6. EJB support in CICS Transaction Server Version 2 allows any Java program to easily invoke EJB in CICS.

When an EJB request is received, a Java Virtual Machine (JVM) is started in CICS, 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 6). If the session bean is to call a business logic program, the CCI API can be used; the same CCI API as provided by the 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 could be called prior to calling the business logic.

The session bean in CICS has the option to be stateless between invocations or state-full, in which case CICS will save the session bean data (passivate) into a VSAM file and restore the state automatically (activate).

Enterprise JavaBeans provides a tightly coupled connection because both ends need to be implemented by compatible J2EE technologies and EJB interfaces.

WebSphere MQ
IBM WebSphere MQ allows you to easily exchange information across different platforms, integrating existing business applications in the process. WebSphere MQ assures reliable delivery of messages, dynamically distributes workload across available resources and helps make programs portable (see Figure 7).

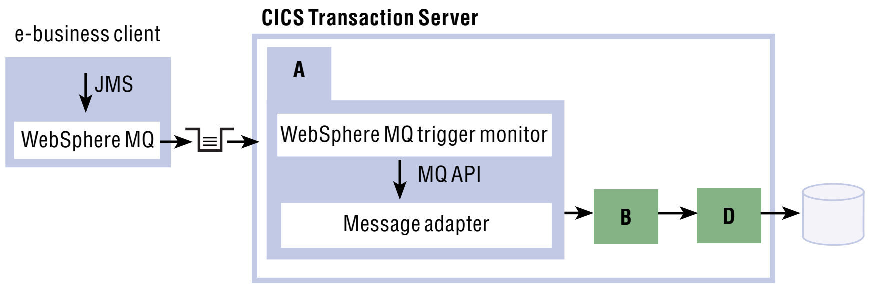

Figure 7. WebSphere MQ provides assured delivery of messages from many platforms to efficiently access CICS asynchronously.

WebSphere MQ provides Java Message Service (JMS) APIs and native WebSphere MQ APIs for use by e-business clients on a wide variety of platforms, with many options for routing and encrypting messages prior to arriving on WebSphere MQ for z/OS.

The WebSphere MQ trigger monitor program runs in CICS 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 will usually continue to process messages on the inbound queue until it is empty.

Although WebSphere MQ can be used for pseudo-synchronous messaging, you need to ensure that appropriate error handling and compensating business logic programs are in place.

IBM WebSphere MQ DPL bridge for CICS provides a second option (see Figure 8). 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 e-business client can format the message into a form acceptable by the business logic program.

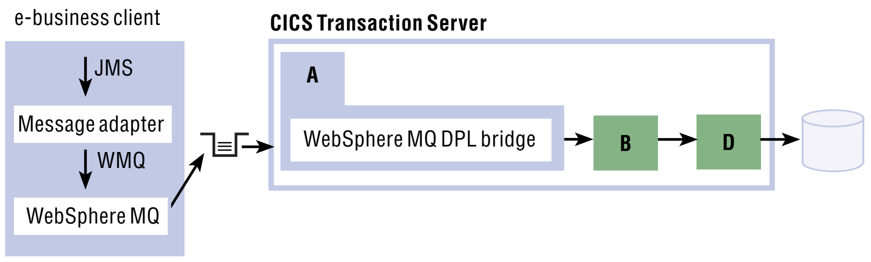

Figure 8. The WebSphere MQ DPL bridge can link to existing CICS programs without having to write an MQ trigger monitor program.

HTTP

The adapter component is made up of the HTTP listener in CICS (called CICS Web support) and a message adapter. CICS supports HTTP basic authentication for user ID identification or the more secure SSL encryption and authentication with client and server certificates (see Figure 9). CICS Web support has an analyzer that decides which message adapter to call.

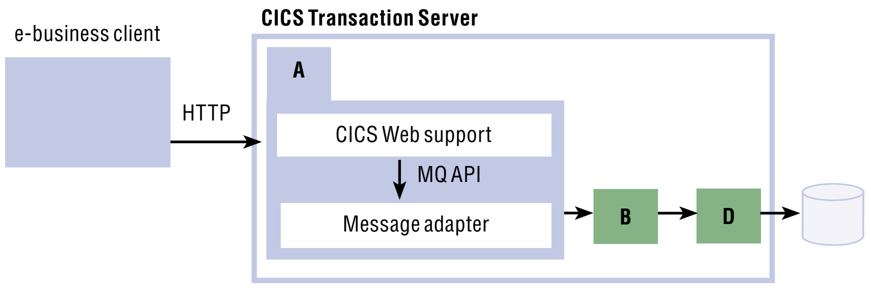

Figure 9. CICS supports HTTP clients and Web browsers with easy-to-use WEB and TCP/IP APIs and can construct HTML and XML responses with the DOCUMENT API.

CICS Web support sets up the transaction and security environment and calls the message adapter. The message adapter uses the CICS WEB APIs to extract the HTTP user data, which is usually formatted as an HTML form. 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.

If the e-business client is a Web browser, the response message will normally be formatted as HTML or XML. 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.

CICS Web support is typically complemented by a combination of static Web servers, servlets and Enterprise JavaBeans, and HTML cascading stylesheets.

HTTP is synchronous and stateless. However, if state management is required, CICS 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.

TCP/IP sockets

The TCP/IP Socket Interface for CICS (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 provides a variant of the Berkeley Software Distribution 4.3 Sockets interface, which is a low-level API with no built-in support for transactions or security.

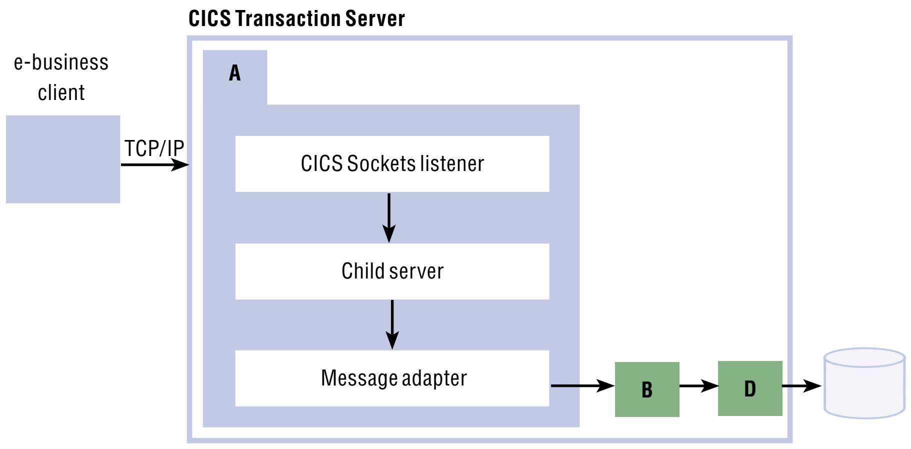

Figure 10. CICS Sockets provides a completely programmable solution where other access options are not suitable.

CICS Sockets provides a concurrent listener, EZACIC02, or you can write your own concurrent or iterative listener to meet your needs. The listener and child server uses the CICS Sockets APIs to receive and send data, and perform general communications control functions (see Figure 10). The programs can be written in COBOL, PL/I, assembler language or C. Client adapters can be written to create new outbound connections.

Summary

CICS provides a range of access options to support modern connectivity architectures, such as Web services and J2EE, and other standard transport mechanisms. With the right external connectors and internal adapters, you can maximize the reuse of your existing mission-critical CICS assets.

Table 1 summarizes the strategic options previously discussed for e-business clients to access CICS programs. Application functional requirements should be compared with the information about each access option. You are encouraged, for the business logic program, to maintain a COMMAREA interface for maximum flexibility to support multiple access options now and in the future as open standards evolve.

CICS and IBM WebSphere Application Server are strategic middleware products that interoperate well using technologies, such as Web services, to support end-to-end on demand systems. They exploit and complement z/OS qualities of service, such as high availability and scalability at low cost per transaction, with a high level of security. In combination, WebSphere and CICS support almost any mission-critical e-business solution.

Strategic options for e-business clients to access CICS programs

e-business client				**CICS Transaction Server**	
Standard architecture	**Capabilities**	**Security to IBM @server® zSeries®**	**Transactional scope**	**Interface**	**Coupling[1]**
SOAP[4]	Synchronous (HTTP) Asynchronous (WebSphere MQ) Inbound and outbound	User ID + password SSL	CICS transaction	XML in a CONTAINER	Low
JCA	32KB maximum message size Inbound only Synchronous and asyn-chronous	User ID + password SSL	Local transaction[3] Global transaction[3]	COMMAREA[2]	Medium
Enterprise JavaBeans[4]	EJB state management Inbound and outbound Synchronous	EJB security roles SSL	CICS transaction Global transaction	EJB (session bean)	High
Standard transport					
WebSphere MQ	Inbound and outbound Asynchronous Assured delivery	User ID + password SSL	CICS transaction	WebSphere MQ API or COMMAREA[2]	Medium
HTTP	Inbound and outbound Synchronous	User ID + password SSL	CICS transaction	CICS WEB API	Medium
TCP/IP sockets	Inbound and outbound Synchronous and asynchronous	User ID + password	CICS transaction	CICS sockets API	High

1. Coupling: Low indicates endpoints do not share assumptions about implementation technologies; high indicates tight integration optimized for run-time performance with many assumptions and dependencies.
2. COMMAREA is a message formatted as XML, simple text or a binary structure such as a COBOL record.
3. Dependent on topology used. Refer to the white paper "Integrating WebSphere Application Server and CICS using the JCA."
4. SOAP and EJB support is provided in CICS Transaction Server, Version 2.

For more information

To find more information about IBM CICS Transaction Server, visit:
ibm.com/cics

To find more information about IBM WebSphere Application Server, visit:
ibm.com/software/webservers/appserv/was/

To find more information about IBM WebSphere MQ, visit:
ibm.com/software/integration/mqfamily/index.html

To find more information about IBM WebSphere Studio Enterprise Developer, visit:
ibm.com/software/awdtools/studioenterprisedev/

For a detailed description of "Integrating WebSphere Application Server and CICS using the JCA," visit:
ibm.com/software/htp/cics/library/cicstgv5.html#wpapers

ON DEMAND BUSINESS™

G224-7324-00