Business Integration Solutions
November 2002

WebSphere® software

IBM

Using a message-based approach to integrate your CICS system with your entire IT infrastructure.

By Mike Brooks, Senior IT Specialist,
WebSphere Foundation, IBM Hursley, UK

Contents

Introduction

The programming components that comprise your IT system must exchange information when they collaborate in a complex business application. When they run on different operating systems and must communicate across your network, it presents a considerable integration challenge. A message-based approach can provide the solution. Programmed components can exchange information in the form of messages – a string of bytes meaningful to both applications that reference it. So you can focus on business logic instead of the complexities of different operating systems and network protocols.

Although IBM CICS® Transaction Server provides a wide variety of functionality, IBM WebSphere® MQ (formerly IBM MQSeries®) software enhances the server with realtime messaging. This white paper discusses how a message-based architecture can provide effective solutions to CICS Transaction Server integration issues. It offers scenarios to help you decide which solution is most appropriate to your business integration needs. All designed to help you realize the full potential of your transaction-processing environment.

A proven transaction-processing system

During the last 35 years, CICS has evolved into a widely used transaction-processing system on many companies' mainframes. Most businesses today use a variety of client devices to schedule work, which can then be serviced by one or more CICS regions collaborating with other products, like database managers and file systems. The latest release, IBM CICS Transaction Server for z/OS™, offers a broad range of interfaces that enables clients to connect to the server and to run applications. The clients include everything from dumb terminals that use 3270 data streams to Enterprise JavaBeans (EJB) components, which call CICS Transaction Server using Remote Method Invocation over Internet Inter-ORB Protocol (RMI over IIOP).

Realizing the full potential of your transaction-processing environment

Even with CICS connectivity functionality, combining CICS Transaction Server with the robust message-based architecture of WebSphere MQ software allows you to effectively address almost any CICS Transaction Server integration issues. When the programming components run on different operating systems and communicate across a network, you must address significant programming challenges to achieve the desired level of integration. A message-based approach enables programmed components to exchange information in the form of messages.

Instead of exchanging information directly from one application to another, messages are placed in queues – data structures that store them until they are retrieved by an application. A queue manager maintains the queue and is responsible for the integrity and persistence of the message. It can also deliver messages across a network to other queue managers. As you create new applications – or extend existing ones – to address changing business needs, you can only realize the full benefits that specific software solutions provide if you can integrate them with other disparate software components.

Mergers and acquisitions

When organizations merge, you must combine the software components you've used in isolation to provide a cohesive set of services across your entire enterprise. With CICS Transaction Server as a central component, the current set of connectivity options don't always allow it to integrate with software running on other platforms. WebSphere MQ technology offers a viable approach to resolving these issues with its ability to connect applications on disparate platforms and exchange messages between those applications and CICS Transaction Server. Providing assured, once-only delivery of important messages, workload balancing across its queues and failover functionality when a platform is unavailable.

Updating business-critical information

Traditionally, CICS has enabled companies to process enterprise data online during the workday—while batch updates to the data occur during off-peak hours. As your company extends its business across different time zones or increases the available period of service, the window for batch updates decreases. In fact, the higher number of updates to enterprise data may be so significant that you require a move away from batch processing altogether. A message-driven approach allows updates to be scheduled through CICS Transaction Server by asynchronously writing them to queues. They can then be processed when the opportunity arises, greatly reducing the need for a batch window. This means the system workflow does not have to be interrupted to run the batch update. Asynchronous processing means you can use spare capacity in the system to make updates.

Maintaining transaction integrity

One of the most robust features of messaging technology allows a program to work asynchronously. A client program can schedule a request using a message queue, and the queue manager can acknowledge acceptance of it. The client can then continue other work or disconnect—and the message can be processed later. The client receives a quick response and is assured that the updates will occur when the server which processes the message can do so. The queue manager delivers the message, once and only once, to a program that services the request. This program may reside on another system and may not be available at the time the client runs. Using CICS Transaction Server—as a client or a server—with WebSphere MQ gives you a fully integrated approach to provide this level of functionality. Operating much like CICS Recovery Manager, WebSphere MQ uses IBM Multiple Virtual Storage (MVS™) log streams to hold copies of messages when it accepts them from clients. If WebSphere MQ is restarted before the message is passed, that message persists on the log stream.

Addressing specific requirements

Some applications have a specific requirement—for auditing, logging or some other purpose—to queue requests before they are executed. By writing WebSphere MQ commands into your CICS applications, you can meet particular requirements when CICS Transaction Server creates or receives a request.

Event-driven transactions

Automation — where an action is started by an event instead of a person — has become an increasingly popular component of large, and sometimes long-lived, applications. Many of the techniques employed in this approach work well within a message-driven architecture. CICS Transaction Server can provide applications that form one part of the solution. However, those applications require the services that WebSphere MQ offers to help you automate event-driven transactions. CICS Transaction Server can put a message on a queue and continue with other tasks. The message is read and processed by the appropriate application, which can't directly interfere with CICS Transaction Server. And, through WebSphere MQ, the transaction process completes without any human intervention.

Connecting new and old applications

Over time, your organization may have developed an extensive inventory of software components to use within CICS systems. As new business needs arise, you want to reuse these components in new solutions — without having to restructure them. You can save money by reducing the need to create new applications and by shortening development time. Sometimes a CICS component provides a proven core business function that may be difficult or risky to re-create or modify for some new purpose. CICS Transaction Server workload components fall into two categories: business logic and IBM 3270 technology-based devices. In those cases, WebSphere MQ helps CICS Transaction Server integrate both pure business logic and 3270 technology-based transactions into a message-driven solution without the need to modify the existing function.

Making the connection across disparate applications

Business logic programs receive input and deliver output in a format independent of the client device from which they are run. A communications area (COMMAREA) can be used to input data to a program and to overwrite it, or a separate one can be used for output. For the purposes of this white paper, business logic programs refer only to those programs that can be called using a COMMAREA interface. WebSphere MQ allows you to invoke this business logic from software that can't otherwise interface with CICS Transaction Server.

The second set of programs has been written to run from 3270 technology-based devices—many of which existed from when individuals initiated almost all work from IBM VTAM® terminals. These programs frequently contain a mixture of presentation and business logic and are not easily restructured. And when you introduce new devices, you often have to reface these applications. WebSphere MQ software provides sample applications that you can customize to use within CICS Transaction Server to allow a 3270 program to interface with message queues. Applications that can interact with WebSphere MQ, but not directly with CICS Transaction Server can take advantage of the collaboration between WebSphere MQ and CICS Transaction Server—enabling you to reuse 3270 components without having to change them.

CICS adapter

WebSphere MQ software and CICS Transaction Server run at separate address spaces within IBM OS/390® or IBM z/OS systems, and they communicate using components provided by both products. CICS Transaction Server has a task-related user exit (TRUE) interface that allows it to forward executed information to another subsystem. WebSphere MQ software provides a CICS adapter—a set of components that you can run within a CICS region—to allow it to create a connection through a TRUE interface to a queue manager. It also allows application programs running within CICS Transaction Server to use the connection to send and receive messages. Figure 1 shows the main components in this architecture.

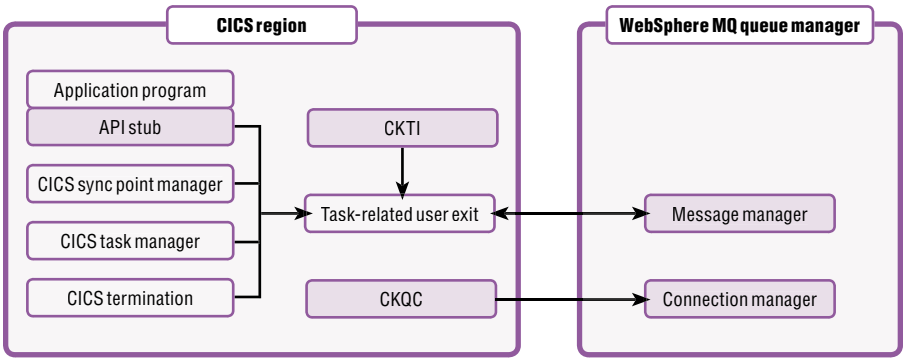

Figure 1. CICS adapter

A CICS region can use a single adapter to connect to one queue manager. And several CICS regions can share a workload through a common queue manager. The adapter provides CICS Transaction Server with a control transaction, or CKQC, to start, stop and customize the interface between CICS Transaction Server and the queue manager. You can run this transaction from a program list table (PLT) program during the initialization of the CICS region or from a 3270 terminal once the region has started.

Scenario 1: WebSphere MQ aware applications

A CICS application program is considered to be *WebSphere MQ aware* when it contains commands provided by the WebSphere MQ application programming interface (API). Figure 1 shows this as the application program that interacts with the TRUE interface to access the services of the queue manager. These programs must be link-edited with a special stub provided by the CICS adapter, so they can use this interface. It also shows other components of CICS Transaction Server involved in the execution of the transaction interacting with the TRUE interface as part of this process.

This scenario offers the most appropriate solution when you are creating a new CICS application or extending an existing one so that, using WebSphere MQ as the transportation mechanism, it can send requests to another trading partner application running outside of CICS. The CICS application will contain WebSphere MQ API (MQI) commands, and, as a result, it will have limited potential for reuse for future projects without WebSphere MQ. This application will also be tightly coupled at the message-content level to the one it communicates with because both must share the same command structure so that each application can deconstruct received messages.

The CICS component can send one or more messages without requiring a reply. If it does this, it can rely on WebSphere MQ to retain the message until it is read by the partner application – even if this occurs after the CICS application has completed its transaction or after the CICS subsystem has been shut down. Or the CICS component can wait for a reply message from the partner system. However, when using this approach, you should consider that the application must be able to deal with instances where the reply message does not arrive immediately or is never received. The four remaining scenarios allow WebSphere MQ clients to communicate with CICS Transaction Server and schedule the execution of various types of transactions.

Scenario 2: WebSphere MQ aware programs started through a task-initiator transaction

If your solution involves creating new components on both the client side and within CICS, consider this scenario or scenario 3. Figure 1 also shows the use of another component supplied as part of the CICS adapter. A task-initiator transaction, or CKTI, can monitor a specific message queue across the connection to the queue manager. Several CKTI commands can monitor a set of queues, although no more than one instance can be used with a particular queue from within a single CICS region. Each instance of CKTI is defined, started and stopped using the CKQC adapter-controller transaction. When one or more messages arrive in a message queue to meet a particular trigger condition, the CKTI transaction detects this event. You can configure the CKTI to start a separate named CICS transaction to process the message(s) on this queue. Having started a new instance of this named transaction, CKTI then continues to monitor the queue for other trigger events.

The named transaction started by CKTI is, in effect, the same as the one described in the first scenario. The transaction contains the stub, which allows it to access the CICS adapter through the TRUE interface and typically contains commands to retrieve one or more messages from the queue. It may also contain instructions to put messages onto other queues or carry out any of the functions supported by standard CICS APIs.

With both of these scenarios, the application programs must contain MQI commands and, as a result, may have a limited level of reuse. You can't invoke the task-initiator transaction as part of an extended unit of work. It can't modify the security context for any named transactions it starts. So scenarios 2 and 3 offer a specialized solution for using WebSphere MQ with CICS.

The WebSphere MQ-CICS bridge

Another approach you can use to run a CICS application from a message to overcome these restrictions is through the WebSphere MQ-CICS bridge. The WebSphere MQ-CICS bridge allows a WebSphere MQ application to call an application running under CICS Transaction Server in one of two ways. The first involves using a message queue as the input and output to a distributed programming link (DPL) request running within CICS Transaction Server.

The second allows the invocation of a CICS 3270 transaction and provides input and output in the same way. In either case, the programs running within CICS Transaction Server do not contain MQI calls and are unaware that they have been invoked by WebSphere MQ, making them fully reusable components.

With this scenario, you can easily set up and maintain your messaging infrastructure. Ideal when you want a basic solution that doesn't rely on more elaborate messaging formats. By avoiding these complexities, scenario 2 cannot propagate a user ID from the client to a CICS application or participate in an extended unit of work. But some solutions may not require this functionality.

Scenario 3: WebSphere MQ-CICS bridge and DPL requests

A CICS business logic program is linked together by the linkage editor, a tool that pulls together all the compiled programming components and converts them into a load module that can then run in the deployed environment. When this occurs, the CICS business logic program receives input in a COMMAREA and returns its output in the same way. The WebSphere MQ-CICS bridge provides a CICS transaction and an associated CICS program that can monitor a particular message queue across the connection between the local CICS region and a queue manager. A WebSphere MQ client application can write a structured message to this queue. This message must contain information in a predefined format that the monitoring transaction can use to decide how to handle the message. Several formats are possible, each starting with a block of data called an MQMD header. This field contains control information used by the monitoring transaction like the message format type, along with optional information, such as a reply-queue identifier and a user ID.

The information that follows the MQMD field can simply be the name of the application program to run within CICS, with the option to include data that the program can receive as its COMMAREA. You can include other control information with an MQCIH header, which must follow the MQMD field. The MQCIH field can include unit-of-work information for a sequence of DPL requests or pseudo-conversational controls needed when running a 3270 application, as outlined in scenario 4. The bridge DPL transaction can set some of the MQCIH fields in any response message for use by the client application when the request has been serviced.

Figure 2 shows the main components of this architecture. A connection must exist between the queue manager and CICS region, both running in the same MVS system. You can achieve this connection using the adapter, under the control of the CKQC transaction, as described earlier in this paper.

You must start a bridge-monitoring transaction (CKBR) to look for messages arriving on a particular queue. Several monitoring transactions can run concurrently to manage a set of queues in this way. When the message arrives, the monitoring transaction detects it and starts a bridge DPL transaction to process the message. The monitoring transaction continues to look for other messages arriving on the queue. You can set up the bridge DPL transaction to run requests using a system user ID or the user ID of the requester.

The DPL bridge task reads the message from the queue. From within the message, the task finds the name of a CICS application program, any input data it requires and, optionally, the name of a message queue to send a response to. Next, the DPL bridge transaction sets up a COMMAREA containing the input data and links to the named program, returning any output COMMAREA to the reply queue after the program has ended. The WebSphere MQ client application can use this mechanism either synchronously or asynchronously. It can wait for a response by monitoring a particular queue, or it may send a message and continue processing other transactions or even terminate without waiting for CICS Transaction Server to process the request. If the client runs on the same system as CICS Transaction Server, it can monitor the transmission queue for a response. If it is remote to CICS Transaction Server, the client can use WebSphere MQ to route the response message from the transmission queue to one defined to its local queue manager.

Each DPL request executes in isolation, and no state is preserved in CICS to tie up a series of requests that a client might make. So, if the client calls CICS Transaction Server more than once, each piece of work is treated separately because the server doesn't retain any information to tie them together. The WebSphere MQ monitoring transaction and DPL bridge transaction run within the same CICS region. The program targeted by the DPL request can be made eligible for routing to another CICS region. All to help balance the workload more effectively.

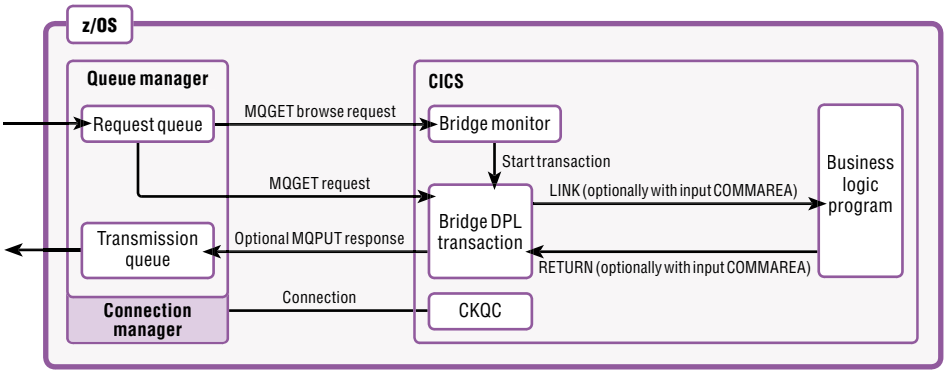

Figure 2. WebSphere MQ-CICS bridge with DPL[1]

This scenario allows you to create or reuse business logic components and integrate them with client applications that can build and optionally receive sent messages through WebSphere MQ. From the CICS perspective, this is the recommended approach to create new applications, regardless of whether they are invoked from WebSphere MQ or from some other mechanism. The client must be aware of any COMMAREAs used by the CICS component, but the CICS program doesn't need to know anything about the client side. And to understand and use WebSphere MQ headers, you only need a minimum level of skill to write to the client side. As a result, maintenance is considerably simplified. Scenario 3 also has other security and transactional controls that the previous scenarios do not. Many enterprises have numerous CICS applications written specifically to run on 3270 devices. These applications are often difficult to rewrite. However, CICS Transaction Server has introduced functionality that allows it to run on a range of client types. Scenarios 4 and 5 describe this approach.

Scenario 4: WebSphere MQ-CICS bridge and 3270 transactions

Reusing 3270 technology-based applications has been a challenge with CICS Transaction Server. It was initially addressed in IBM CICS/ESA®, Version 3 with Front-End Programming Interface (FEPI) and then through the 3270 bridge component—introduced in CICS Transaction Server, Version 1.2. The WebSphere MQ-CICS bridge provides components that use the underlying 3270 bridge function to address the challenge more effectively. As with DPL requests, you can use the same WebSphere MQ-CICS bridge-monitoring transaction to detect messages arriving on a particular queue.

This monitoring transaction starts a separate transaction to process each message request. The initiated transaction reads the message from the queue and within that message finds the identity of a CICS 3270 transaction to invoke. Other components in the message are used as input to the application program associated with the 3270 transaction, which is then scheduled to be executed. Not only does this solve the problem of reusing 3270 applications, it does so without CICS Transaction Server being aware that the transaction is being handled separately.

When the CICS 3270 application runs, it executes in a special bridge-transaction environment. Within this, the bridge-exit component – provided by WebSphere MQ – intercepts most API calls from the 3270 application to either basic mapping services (BMS) or terminal control. CICS Transaction Server processes all other API requests to its resource managers, exactly as though the transaction had been started from a 3270 device. If the 3270 application issues a receive command, the bridge exit maps the input message to an application data structure for the program to use. If the application issues a send command, the bridge exit takes the application data structure of the program and maps it to a WebSphere MQ message. When the 3270 application terminates, this message is written to a reply queue, where the application that initiated the request can read it.

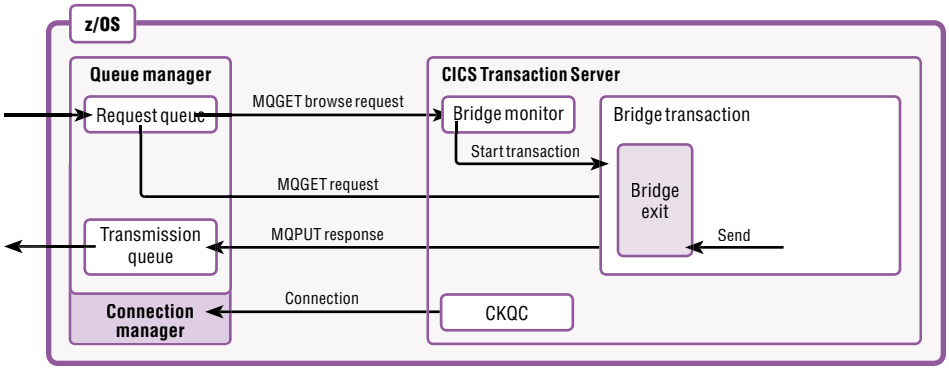

Figure 3. WebSphere MQ-CICS bridge with 3270 transactions

Unlike DPL requests, 3270 transactions can be pseudo-conversational. This means CICS Transaction Server maintains internal conversation state in a control-block structure representing a virtual 3270 terminal device, similar to those used by VTAM terminals connected to CICS Transaction Server. You can specify timeout values in the System Initialization Table or in the message headers, so that CICS Transaction Server can detect inactive conversations and discard unwanted control blocks associated with them. The primary restriction with this method is that the 3270 transaction must run within the same region as the monitoring transaction, which means workload balancing is not supported. However, it is totally recoverable in the event of a transaction abend, or system outage. Figure 3 shows a sample bridge-exit program.

If you want to integrate a non-CICS platform with CICS 3270 applications using WebSphere MQ and you're running on CICS Transaction Server, Version 1.2 or 1.3 for IBM OS/390, then you should use this scenario. As with scenario 3, you must use message headers to control the pseudo-conversation between the client and the 3270 application and to propagate any security context. By running the CICS bridge, you can't route the 3270 application to another CICS region. Instead, it must run alongside the monitoring transaction listening for messages arriving on a particular queue. Despite this restriction, it still remains a robust solution. You can also use this solution if you want totally automated application recovery in the event of a system outage.

Scenario 5: WebSphere MQ-CICS and Link3270 bridge with 3270 transactions

The release of CICS Transaction Server, Version 2.2 offers new functionality, called Link3270 bridge. This function allows a 3270 technology-based application to be wrapped with a layer that receives input and sends output in COMMAREAs. The new mechanism can be driven as a DPL request using the MQ monitoring transaction. The Link3270 bridge provides support for dynamic-transaction workload balancing by using transaction routing, making it suitable to balance workloads within an IBM CICSPlex® system. It is completely recoverable in the event of a transaction abend, providing superior performance to the WebSphere MQ-CICS bridge.

Scenario 5 uses the Link3270 bridge, available only with CICS Transaction Server, Version 2.2 for z/OS. This solution overcomes the restrictions associated with scenario 4. If you want to integrate WebSphere MQ clients with CICS 3270 applications, this scenario has the potential to become the solution of choice.

Summary

As your organization continues to integrate disparate systems to build more complex solutions—or as the result of mergers or acquisitions—WebSphere MQ can provide a proven, reliable option to connect many clients to CICS Transaction Server. With the reduction in operational batch windows, you need an asynchronous messaging mechanism to move from batch to online processing with your CICS systems. Some applications require that requests be reliably queued prior to execution, typically for logging or auditing purposes. WebSphere MQ has the robust functionality you need to make this happen. Using automation in complex, long-lived business applications means that recoverable asynchronous messaging is an essential part of these solutions, which you can achieve by combining WebSphere MQ with CICS. Instead of building completely new applications, you can also effectively reuse legacy CICS application components in a message-driven architecture—either as business logic or to be run from 3270 devices—using the components provided by WebSphere MQ.

References

To download the IBM manuals listed below, visit:
ehone.**ibm.com**/public/applications/publications/cgibin/pbi.cgi

WebSphere MQ for z/OS V5.3 Concepts and Planning Guide: GC34-6051
WebSphere MQ for z/OS V5.3 System Setup Guide: GC34-6052
WebSphere MQ for z/OS V5.3 System Administration Guide: GC34-6053
WebSphere MQ Application Programming Guide: SC34-6064
CICS Transaction Server for z/OS V2.2, CICS External Interfaces Guide: SC34-6006

To download the IBM Redbooks listed below, visit: www.redbooks.ibm.com

CICS Transaction Server for OS/390 V1.3, Web Support and 3270 Bridge: SG24-5480

Revealed! Architecting Web Access to CICS: SG24-5466

e-business Cookbook for z/OS Volume II: Infrastructure: SG24-5981

For more information

To learn more about the value of using WebSphere MQ software with CICS Transaction Server, visit **ibm.com**/webspheremq/messaging.

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

Produced in the United States of America
11-02

[1] The CICS region where the bridge runs must be on the same z/OS system image as the queue manager. The request queue must be under the control of that queue manager, but the output message can be forwarded to a remote queue.

G325-2113-00