

XML-Enabling CICS Applications for eBusiness

A HostBridge™ White Paper

1/8/2002



Toll-free: (866) 965-2427
Email: info@hostbridge.com

Copyright Notice

Copyright © 2001 by HostBridge Technology. All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without prior written permission. You have limited permission to make hardcopy or other reproductions of any machine-readable documentation for your own use, provided that each such reproduction shall carry this copyright notice. No other rights under copyright are granted without prior written permission. The document is not intended for production and is furnished "as is" without warranty of any kind. All warranties on this document are hereby disclaimed including the warranties of merchantability and fitness for a particular purpose.

Revision date: 8/12/2001

Trademarks

HostBridge is trademarked by HostBridge Technology.

XML-Enabling CICS Applications for eBusiness

HostBridge allows existing CICS 3270 transactions to be securely invoked via an HTTP request and delivers their output as a standard XML document — not a 3270 screen. As a result, HostBridge allows enterprises to integrate their existing CICS applications with new eBusiness applications. This document provides an overview of HostBridge and its value to the enterprise. It assumes a basic knowledge of concepts and technologies surrounding web-enablement of host applications.

Today, most corporate data and applications continue to reside on IBM-compatible mainframes. As corporations make the transition from "brick-and-mortar" to "click-and-mortar" enterprises, they must provide employees, partners, and customers with access to this data via intranets, extranets, and the Internet.

The late 1990's saw the emergence of a variety of "web-to-host" solutions that provide access to 3270-based host applications via the web. The most basic solutions merely offer thick-client Java applets that enable standard TN3270 connections to the host. Other solutions allow end-users to interact with text-based host applications through a more usable HTML or graphical user interface. The most advanced solutions convert the 3270 data stream or screen space to other types of data objects that can then be used within an entirely new web application.

Underlying all of these solutions is the use of 3270 data streams as the means of communicating between the host application and web-to-host gateway. The web-to-host gateway then uses a technique known as "screen scraping" to extract information out of the 3270 screen space. Whether screen scraping is performed on the client or the server, it involves the same process: using row & column coordinates to indicate the data areas to processed as input or output. However, screen-scraping techniques are particularly prone to scalability and application maintenance problems. Any changes to the host screens break the web-to-host applications. This means that for every change a CICS programmer makes to a host application, the change has to be communicated to the web developer who must make a corresponding change in the web-to-host application.

Historically, companies used screen-scraping because the only alternative was to re-engineer or re-write the host application to separate the presentation logic from the business logic — that is, to make it "non-visual." Now, HostBridge allows secure access to CICS transactions via an HTTP request and delivers their output as a standard XML document with no screen-scraping or host application reengineering. HostBridge circumvents the use of 3270 data streams and, as a result, allows enterprises to take a quantum leap forward in leveraging, enabling, integrating and scaling their existing CICS transactions into eBusiness applications.

Problem: Extracting Usable Data from Host Applications

Large enterprises have millions of dollars invested in information and data stored on mainframes hidden behind applications that are difficult to learn and use. Many companies are making the decision to move host data to the web for use by employees, or externally by customers and partners. As these companies prepare to make this data available, they face a unique challenge: how do web applications mine the host applications for data and retrieve that information in a usable form?

Learning Curves and Terminal Emulators

Many companies use web-based terminal emulators to allow end-user access to host applications. For users already familiar with how the host applications work, there are several Java or ActiveX products that perform terminal emulation in web browsers and present the host screens in a traditional 3270 terminal format: recreating a text-based screen with 24 rows and 80 columns and even presenting the familiar “green screen.” However, for new or casual users, this exposure to the host application interface is confusing and creates a learning curve that prevents this method of host access from being practical to anyone outside an organization.

Web-to-Host Screen Scrapers

To overcome the learning curve presented by web-based terminal emulators, several web-to-host products employ the technique of screen-scraping. These products generally access host applications and parse the contents of each screen looking for unique identifiers. In most cases, the web-to-host products will correlate the unique ID’s with a map to determine what information to extract from each host screen and how to present that information back to the user in HTML. The problem with this approach is that any change to a host screen breaks the web-to-host applications, so that every change to a host application must also involve a corresponding change in the web-to-host solution.

End-user Orientation

Current methods for accessing host applications serve the needs of organizations who want to allow end-user access to host applications. Current web-to-host products and services convert host data into HTML so that it is readable through web browsers and have no way to present host application data logic to other applications in a format that is meaningful and useful. This kind of application-to-application communication is becoming increasingly important as companies engage in business-critical B2B projects with their partners. For example, suppose a company needs to run a daily check with a distributor on the availability of all parts for a new widget. Using a standard web-to-host solution to translate the host application data to HTML, the results of a query to a CICS application after might look something like the following example.

```

<html>
<head>
<title>Untitled Document</title>
</head>
<body bgcolor="#FFFFFF">
<table width="75%" border="1">
  <tr>
    <td>Part #</td>
    <td>Item description</td>
    <td>Availability (units)</td>
  </tr>
  <tr>
    <td>123-ABC</td>
    <td>Component A</td>
    <td>12 </td>
  </tr>
  <tr>
    <td>456-DEF</td>
    <td>Component B</td>
    <td>5</td>
  </tr>
</table>
</body>
</html>

```

To parse this HTML and identify which items within the <td></td> tags are part numbers, item descriptions and number of units available can get very complex because HTML does not differentiate between the contents of each pair of tags. In fact, parsing HTML is actually more problematic than parsing 3270 datastreams because HTML has much less structure. Again, any changes in the host application could undo the web-to-host product's ability to retrieve data and the partner application's ability to check on part availability.

Solution: HostBridge and XML

HostBridge solves the problem of data recognition by allowing users to access host applications via an HTTP request and by converting returned data from CICS applications into XML for use by the web applications. Using XML preserves the business logic of the CICS applications and provides an industry-standard means of exchanging data between CICS and any other XML-enabled applications.

About XML

XML stands for eXtensible Markup Language, and is a data format for structured document exchange between applications. Like HTML, it is a markup language derived from SGML. However, unlike HTML, which the Internet community created to format information and display it across multiple platforms, XML is best suited to organize data for structured document exchange between applications. While HTML specifies how a document should appear, it does not describe what kind of information the document contains or how it is organized. XML allows you to organize information in a standard way that can enable back-end systems to conduct business transactions in a known format. For example, business partners can standardize on specific XML syntax to describe purchase orders and can then

automate the transfer of that information across otherwise incompatible systems.

The HTML document above could be rendered in XML as follows:

```
<?xml version="1.0" ?>
<response>
  <catalogue_item>
    <partnum>123-ABC</partnum>
    <description>Component A</description>
    <units>12 </units>
  </catalogue_item>
  <catalogue_item>
    <partnum>456-DEF</partnum>
    <description>Component B</description>
    <units>5</units>
  </catalogue_item>
</response>
```

The XML syntax is not only readable by humans, but the data can be easily parsed by applications or imported directly into databases, since all major databases now support the import and export of data in XML.

Benefits of XML

The key benefit of retrieving host data in XML is that the information is completely independent of how you wish to display it. Because XML is an industry-standard markup language, it can be used for multiple purposes. For example, an eXtensible Style Language Transformation (XSLT) allows you to map XML tags and transform them to any other format. In the XML samples we have already seen, you could transform the XML into formats suitable for use with BizTalk, ebXML, UDDI, or any other growing eBusiness exchange technologies.

How Does HostBridge Work?

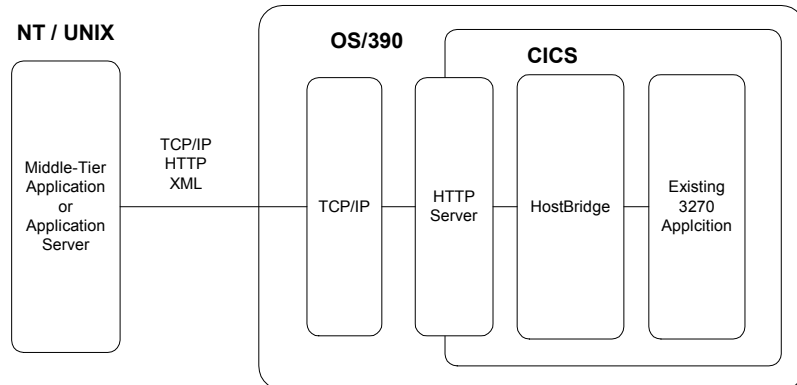
Host Bridge runs on the mainframe under OS/390. It is built on the foundation of two features that IBM has recently added to CICS Transaction Server: CICS Web Support and 3270 Bridge

- **CICS Web Support (CWS)** enables an HTTP client (e.g., a Web browser) to communicate directly with mainframe CICS application programs without an intermediate gateway or a separate Web server (you can also communicate through the OS/390 HTTP Server or WebSphere 390).
- **3270 Bridge** makes it possible to intercept the flow of data into, and out of, a CICS transaction *before* a 3270 data stream is generated as output or expected as input. 3270 Bridge works by intercepting the flow of control between the user transaction and BMS, thereby allowing another software component, such as HostBridge, to handle input/output operations for the transaction.¹

¹ The vast majority of CICS transactions rely upon a component within CICS referred to as BMS (Basic Mapping Support) to interact with a 3270 terminal. When a transaction calls BMS, it specifies the name of a 3270 screen "map" and a set of fields and values to be used with it. The map determines where on the screen the fields are to be placed and BMS generates the resulting 3270 data stream. BMS also accepts the 3270 terminal input and returns it to the program as values in the corresponding field(s). Thus, an often-overlooked fact is that CICS transactions that use BMS to interact with a 3270 terminal "think" in field name/value pairs - not in 3270 data streams.

Architecture

The diagram below shows the basic architecture of a system using HostBridge.



The basic components include a client application that often resides on UNIX or Windows NT. A typical transaction using HostBridge consists of the following steps.

1. The client application sends an HTTP request via the TCP/IP layer on the host.
2. An HTTP Listener within CICS Web Support monitors TCP/IP and receives the request from TCP/IP.
3. If the Listener receives the HTTP request on a specified port, it passes the request to the HostBridge analyzer.
4. HostBridge authenticates the URL from the client. If the URL from the client contains the correct authorization header, HostBridge assigns a userid and begins a transaction with the host application under that userid.
5. The user transaction returns the requested data to HostBridge.
6. HostBridge converts the data to XML and passes it to CWS.
7. CWS then returns an HTTP response to TCP/IP.
8. TCP/IP then sends the HTTP response with the XML data back to the client application.

HostBridge XML Conversion

```
Share Trading Demonstration                                TRADER.T004
Share Trading Manager: Real-Time Quote

User Name:      RUSS
Company Name:   Casey_Import_Export

Share Values:
NOW:            00079.00
1 week ago:    00059.00
6 days ago:    00063.00
5 days ago:    00065.00
4 days ago:    00070.00
3 days ago:    00072.00
2 days ago:    00078.00
1 day ago:     00077.00

Commission Cost:
for Selling:    007
for Buying:     010

Number of Shares Held: 0500
Value of Shares Held: 000039500.00

Request Completed OK
-----
PF3=Return                                           PF12=Exit
```

In the screen above, a trader named Russ is retrieving a stock quote for Casey Import and Export. HostBridge identifies each screen and field in a CICS application. Notice the "User Name" field with the value "RUSS." Let's take a look at how HostBridge displays the CICS data in XML after an external application sends the following HTTP request to the mainframe web server.

[http://company.com:\[port\]/hostbridge?HB_TRANID=\[CICS transaction\]](http://company.com:[port]/hostbridge?HB_TRANID=[CICS transaction])

Below is the HostBridge output sent back to the requesting application.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!-- HostBridge Copyright 2000, 2001 HostBridge Technology, U.S. Patent
Pending -->
<hostbridge>
  <token>73722b93</token>
  <timestamp>20010702154224</timestamp>
  <status>
    <response>0</response>
    <desc>ok</desc>
  </status>
  <transaction facility="}AAB" next_tranid="TRAD">
    <status>
      <cics_resp>0</cics_resp>
      <cics_resp2>0</cics_resp2>
      <cics_desc>ok</cics_desc>
      <task_end>endtask</task_end>
      <abend_code />
    </status>
    <parameters>
      <tranid>TRAD</tranid>
      <userid>DSI1</userid>
    </parameters>
    <command>
      <send_map erase="y" erase_unp="n" unlock_kb="y"
        alarm="n" reset_mdt="n">
        <mapset>TRADEMS</mapset>
        <map>T004</map>
        <data_indicator>map_and_data</data_indicator>
        <fields count="15">
          <field name="USER41" index="0">
            <name len="6">USER41</name>
```

```

        <value maxlen="20"
            len="4">RUSS</value>
        <attr byte="00" justify="1"
            disp="n" prot="n" num="n"
            int="n" mdt="n" />
    </field>
    .
    .
    .
    <field name="MESS4" index="0">
        <name len="5">MESS4</name>
        <value maxlen="79" len="20">Request
            Completed OK</value>
        <attr byte="00" justify="1"
            disp="n" prot="n" num="n"
            int="n" mdt="n" />
    </field>
</fields>
</send_map>
</command>
</transaction>
</hostbridge>

```

The entire contents of a transaction appears within the `<transaction></transaction>` tags. Notice that the tags `<name>USER41</name>` and `<value>RUSS</value>` identify the “User Name” field indicated in our example screen above. Also within the `<field name="USER41"></field>` tag, we find other attributes of the “User Name” field defined within the XML, such as field length and attributes. This information appears for each field on the CICS application.

While this information is descriptive enough so that humans can understand what is happening within a transaction, the important point for B2B and application integration projects is that other programs can easily find and extract the data they need.

Accessing CICS Applications with HostBridge

To interact with the CICS application, the URL in the HTTP request needs to include several pieces of information. Let’s look at a URL that could be sent to the CICS application and change the contents of the USER41 field from “RUSS” to “MICHAEL”.

```
http://company.com:4041/hostbridge?HB_TRANID=TRAD&HB_TOKEN=73722b93&USER41=MICHAEL
```

This part of the URL...	Does this...
http://company.com:4041	Identifies the address and port on HostBridge runs. The HTTP listener passes URLs on this port to the HostBridge analyzer.
hostbridge?	Identifies this as a HostBridge session.
HB_TRANID=TRAD	Specifies a CICS transaction that HostBridge should execute.
HB_TOKEN=73722b93	<p>After an initial connection to the host application, HostBridge returns a session ID called a state token. Seen in the XML above as:</p> <pre><token>73722b93</token></pre> <p>Subsequent transactions with the host must include this token in the URL so CICS will recognize the transaction as part of an existing session.</p>

USER41=MICHAEL

Sends the field name and the data value we want to enter in the transaction. HostBridge interprets any name/value pair sent along the URL as input to a field unless the name starts with "HB_" (e.g., HB_TOKEN).

Flexibility and Ease of Use

Because HostBridge uses XML to allow external applications to access host data using tags, there is no need to identify locations of fields and text on a screen, as screen scrapers do. This freedom from screen-scraping provides several key benefits in a production environment.

Screen-scrapers create dependencies between CICS applications and the client applications that access them. If a change in the host application shown in the previous examples shifts the position of the field labeled "USER41," the CICS developers must document the changes for the web application developers so they can modify the screen-scraping tool to grab data from the new field location. If the web developers do not change their applications, the changes in the host application will create errors when the web application scrapes the screens looking for the fields in the old locations. HostBridge removes this dependency because it converts identifying elements for host fields — in this case `<name>USER41</name>` — regardless of their physical location, so web applications simply find the information for the fields they need in the XML files.

Scalability

Because it converts CICS application data directly to XML, HostBridge eliminates the need to scrape or parse 3270 data streams to find relevant host data and convert it into a usable form. And, because it runs on the mainframe, there is no need to incur overhead by passing data through SNA stacks and HLLAPI running on an NT or UNIX box. All of this allows HostBridge to run at the full speed of your CICS applications.

Security Overview

Running a CICS transaction using CICS Web Support and the XML application connector is as secure as running the transaction from a 3270 terminal. The XML application connector works with standard mainframe and Internet security methods so that it works within whatever security model you have in place to protect your applications and data.

HostBridge combines standard mainframe and Internet security methods to create a security model that ensures your applications and data are protected from end-to-end of each transaction.

- **Username/password protection** through RACF, ACF/2, and TopSecret maintains your existing host security that limits user access to resources based on userid authorizations.
- **Client authentication** through CICS Web Support or OS/390 UNIX System Services ensures that clients and external applications are authorized to connect to the host application.
- **Data encryption** with secure sockets layer (SSL) protects data that passes between the web server and the client application using 56-bit or 128-bit encryption.

Attempts to access host applications from a 3270 terminal or emulator only provide username/password protection. Moreover, terminal emulators actually allow users to login and conduct transactions by passing text between the host and the client in clear text. Because the XML application connector works with web servers that authenticate clients and encrypt data, it is a more secure solution than 3270 emulation for accessing CICS application.

The following information provides a more detailed overview of the technical aspects of the HostBridge security model.

CICS Web Support vs. OS/390 Web Server

CICS Web Support is part of CICS TS 1.3. It enables HTTP clients, such as web browsers, to communicate directly with mainframe CICS application programs without an intermediate gateway or a separate Web server. CWS includes an HTTP server that determines how to process HTTP requests based on the port number on which the request connects. Rather than entering the CICS Web Support environment directly through its integrated HTTP server, you can also enter through an OS/390 Web Server, such as Domino Go. The OS/390 Web Server runs on the UNIX System Services (USS) platform and can also provide secure access to CICS services.

The OS/390 Web Server includes some functionality that CICS Web Support does not. For example, OS/390 Web Server supports logging, filtering and redirection. However, “beginning with CICS Transaction Server 1.3, CICS Web Support provides the same secure access possibilities as when going through an OS/390 server.”² Both provide better access control than when using real or emulated 3270 terminals.

² CICS Transaction Server for OS/390 Version 1 Release 3: Web Support and 3270 Bridge (SG24-5480), page 3

HostBridge will work with either CICS Web Support or the OS/390 Web Server.³ Given the simplicity of deploying CICS Web Support, we suggest that organizations use CWS's integrated web server for initial development and testing with HostBridge. The choice between CWS and OS/390 Web Server for production use should be based upon which tool best meets the organization's production requirements. Regardless of whether the installation uses CWS or OS/390 Web Server, the principles of creating a secure transaction environment are the same.

Creating a Secure Transaction Environment

Effective authorization and access control requires that every transaction run under a userid. The userid then becomes the basis for performing all kinds of security checking. Most common is resource security, where the userid is used for checking that a user is authorized to use CICS resources that the currently executing transaction uses. Examples of CICS controlled resources are files, transient data and temporary storage queues, referenced journals, and other transactions that an application wants to start. To create and enforce a secure environment, CICS relies on an External Security Manager (ESM) such as RACF.

Security in the 3270 Terminal Environment

In a terminal environment, if a user is not signed on, the transactions started from that terminal run under the userid of the "default user." This userid is specified in the System Initialization Table (SIT) using parameter DFLTUSER. Typically, this user has limited (or no) authority and cannot start transactions. Thus, in a secure environment, users must sign on with the CESN transaction, which allows them to enter a userid and a password. Another way to sign on is to execute an installation-written transaction that contains an EXEC CICS SIGNON command. All transactions started from the terminal by the signed-on user will run under their userid.

The above description of security in a 3270 terminal environment is predicated on the existence of a "principal facility." Certain CICS commands, such as EXEC CICS SIGNON, require that a principal facility exists. It is easiest to think of a principal facility as CICS's abstract representation of a physical terminal. When CICS initiates a task, it assigns the principal facility to the task and the task "owns" the facility for its duration. No other task can use that terminal until the owning task ends. CICS allows a task to communicate directly with only one terminal, namely its principal facility.

Security in the CICS Web Support Environment

In the CICS Web Support environment, there is no principal facility. Instead, CWS defines the notion of a "bridge facility". Whereas a principal facility represents a physical terminal, a bridge facility represents a virtual terminal.⁴ Since there is no principal facility in the CICS Web Support environment, a CESN transaction or an EXEC CICS SIGNON command cannot be used to obtain and validate userid information. However, CWS defines a different mechanism to achieve the same result.

³ Both CICS Web Support and OS/390 Web Server rely on the MVS TCP/IP program product for network connectivity.

⁴ Essentially, a bridge facility is a control block that the existing 3270-based transaction sees as its principal facility. A bridge facility is modeled on an existing, installed terminal. By default, CICS uses the CICS-supplied terminal definition CBRF (essentially a basic 3270 LU2 terminal definition). Most existing CICS transactions expect to be invoked by unsolicited input from a 3270 terminal, and then issue RECEIVES and SENDS to that terminal, its principal facility. This is exactly the emulation that the bridge creates to the existing code. However, even though the task thinks it has a principal facility and can issue RECEIVES and SENDS, all other tasks, and CICS itself, see the task as a non-terminal task, that is, one running without a principal facility.

Access control in a CICS Web Support environment begins with the CICS TCPIPSERVICE definition.⁵ The TCPIPSERVICE definition specifies a port number to be monitored by CICS, the authentication procedure to be used for requests received on this port, and the name of an “analyzer” program to receive control after the HTTP request has been authenticated. CICS Web Support supports three authorization models: (1) none, (2) HTTP basic authentication or (3) Secure Sockets Layer (SSL). HostBridge is insensitive to the authorization mechanism selected. The following example assumes that BASIC authentication was selected.

CWS Access Control Process: BASIC Authentication

Upon receiving an HTTP request, CWS analyzes it to find an “AUTHORIZATION: BASIC” header. If CWS cannot find an authorization header, then it rejects the request with a “401 unauthorized” return code. If CWS finds an authorization header, then it extracts the userid and password from the header and uses the EXEC CICS VERIFY function to validate the userid and password. As a result, CICS performs the same userid and password validation as if the user were logging on from a 3270 terminal.

If the userid and password combination is invalid, CWS rejects the HTTP request with a “401 unauthorized” return code. If the userid and password combination is valid, CWS passes control to the HostBridge analyzer specified in the TCPIPSERVICE definition. All transactions subsequently executed via HostBridge run under the now authenticated userid.

Note that the userid and password are *never* sent as plain text in the HTTP basic authentication header. Rather, the HTTP basic authentication standard dictates that they are always encoded. Furthermore, the encoded userid and password are included in *every* request. By way of comparison, when accessing CICS using a terminal emulator and the TN3270 protocol, userids and passwords are *always* sent as plain text and are only exchanged *once*, at the beginning of the session.

⁵ The following discussion assumes the use of CICS TS v1.3 with APAR PQ36169 applied. This APAR added the AUTHENTICATE parameter to the TCPIPSERVICE definition. Permissible values for AUTHENTICATE are NO, BASIC, CERTIFICATE, AUTOREGISTER and AUTOMATIC). This APAR also causes CWS to implement/enforce the selected security process prior to the analyzer receiving control.

System Configurations

HostBridge configurations depend on how client applications connect to CICS. Supported connection methods include the following:

- Middle-tier HTTP Servers (with SNA or TCP/IP Connections to OS/390)
- OS/390 HTTP Servers
- CICS Web Support
- MQSeries

There are two types of HTTP servers found in most networks: basic web servers and application servers⁶. Web servers from Apache, Microsoft, and Netscape power most of the UNIX and NT servers that comprise the middle-tier between client applications and CICS. The OS/390 HTTP Server and CICS Web Support power most of the mainframe web servers. Companies use C++, Java, and various popular scripting languages to add custom XML support to these web servers.

Application servers, such as WebSphere, SilverStream, or webMethods power much of the eBusiness world. These servers collect data from disparate sources (such as databases, text files, and other applications) and combine them into a single datastream to be viewed in a browser or passed to another application. Most application servers support the ability to import and transform XML data for use in the datastream.

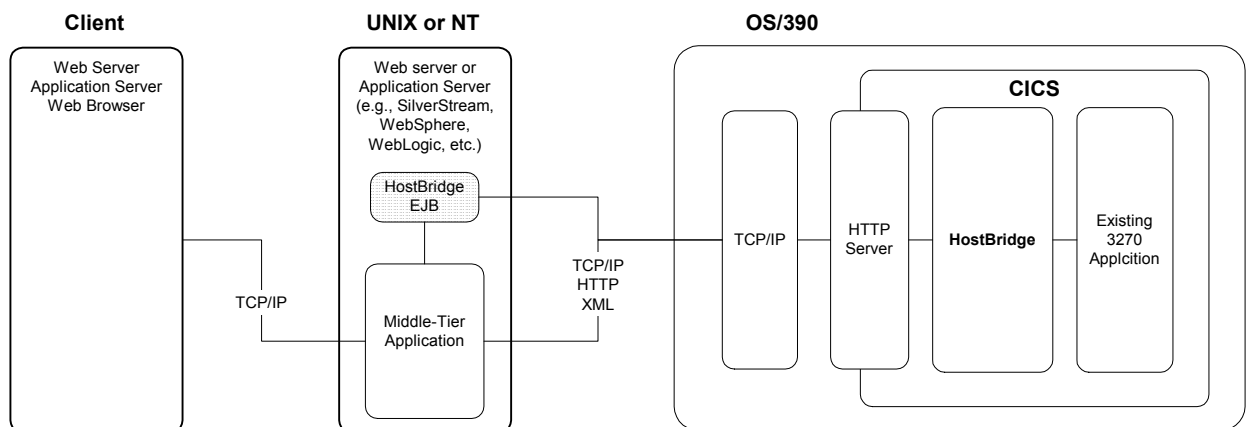
Now, let's look at each of the basic HostBridge system configurations.

Middle-Tier HTTP Servers

For organizations that connect to CICS from HTTP servers on the middle tier, HostBridge allows connections using both TCP/IP and SNA. The connection type you choose can depend on whether or not TCP/IP is running on OS/390.

TCP/IP Connectivity

Most physical middle-tier connections to CICS will take place using TCP/IP rather than SNA in a typical configuration as shown below.



⁶ A third type of HTTP server combines enterprise application integration (EAI) and business-to-business integration (B2B) to enable businesses to share corporate data in the form of XML. Microsoft® BizTalk™ Server 2000 and servers supporting the ebXML standard set by OASIS are the pioneering efforts in this area. These servers are not yet widely used, but they will become a major part of eBusiness infrastructure. Because these servers use HTTP and XML to communicate, HostBridge will play a key role by unlocking CICS data for use in new eBusiness configurations.

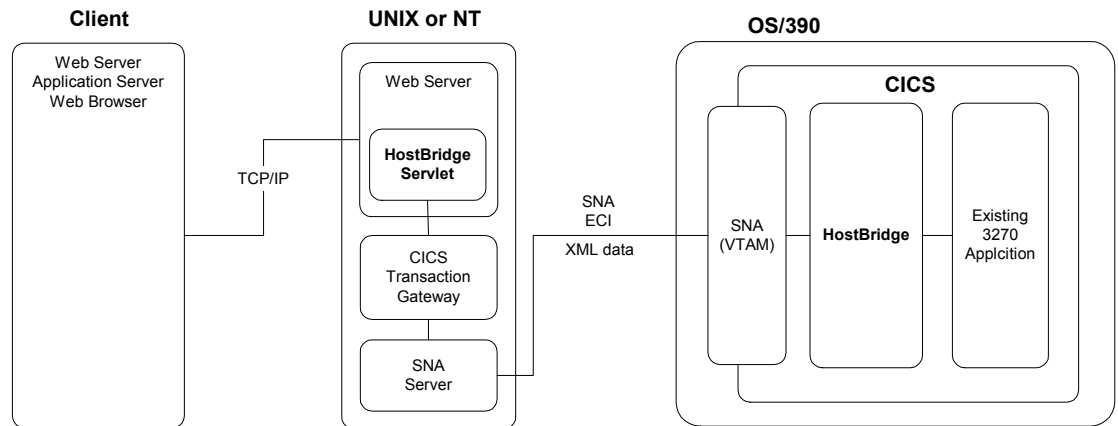
This configuration requires TCP/IP and an HTTP server on OS/390. The HTTP server can be the OS/390 web server, an application server like WebSphere, or CICS Web Support (CWS). In the diagram above, the following sequence of events results in the retrieval of XML data from CICS.

1. A client application requests data from the middle-tier application by sending a URL that identifies the CICS host and HostBridge as the application to execute.
2. The middle-tier application calls a HostBridge enterprise Java Bean (EJB) which generates and sends a URL to the HTTP server running on OS/390.
3. The HTTP server passes the request to HostBridge.
4. HostBridge initiates a CICS transaction and returns the transaction data as XML to the HTTP server.
5. The HTTP server passes the XML back to the middle-tier application, which then passes the data on to the client application.

This is the preferred method for enabling CICS applications to return XML data. It does not require development on the host and there is no need for resource intensive use of an SNA stack on the middle-tier.

SNA Connectivity

A combination of HostBridge servlets and the CICS Transaction Gateway (CTG) provides an extremely flexible solution for providing SNA access to CICS applications from the middle-tier. A basic SNA HostBridge configuration appears below.



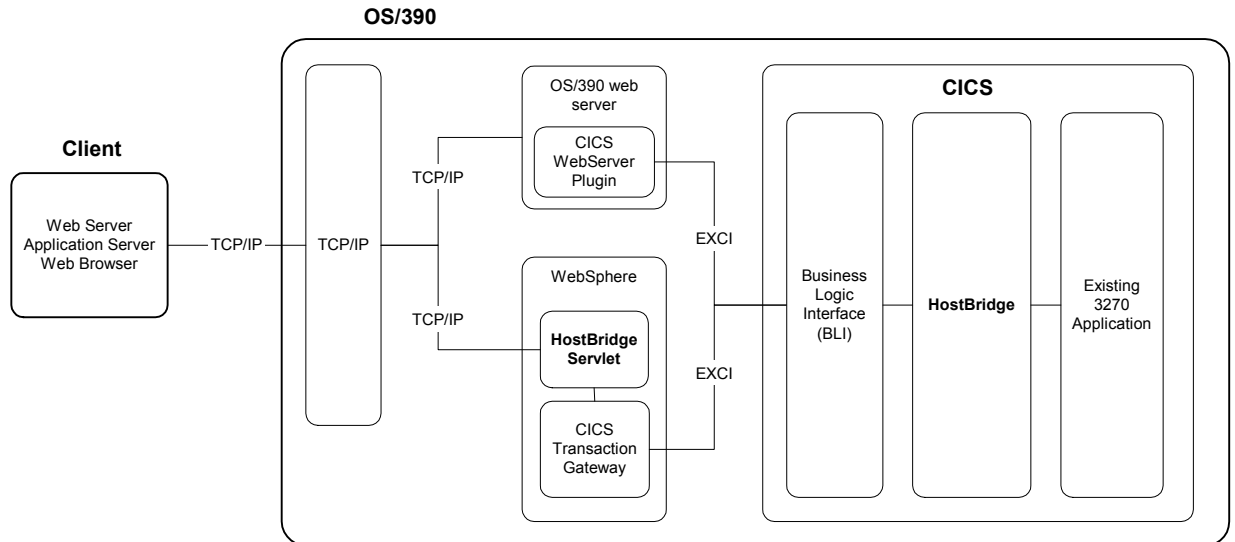
This configuration requires a HostBridge servlet, CTG, and an SNA stack on the middle tier. The HTTP server can be any web server or application server like WebSphere that supports Java servlets. In the diagram above, the following sequence of events results in the retrieval of XML data from CICS.

1. A client application requests data from the middle-tier application by sending a URL that identifies the CICS host and HostBridge as the application to execute.
2. The middle-tier application passes the request URL to the HostBridge servlet, which in turn passes the URL to the CICS Transaction Gateway (CTG).
3. The CTG uses the External Call Interface (ECI) to call HostBridge as a subroutine over an SNA connection.
4. HostBridge returns the CICS transaction data as XML.

For organizations that do not run TCP/IP on the host, the diagram above shows the preferred method for enabling CICS applications to return XML data. Although this configuration does not require host programming or TCP/IP on the host, the SNA stack on the middle tier may hinder scalability of the overall system.

OS/390 HTTP Servers

By using web servers or application servers that run on the host, you can eliminate the middle tier, as shown below.



This configuration requires TCP/IP and an HTTP server on the host. The method for retrieving XML from CICS applications differs slightly depending on whether your HTTP server is one of the OS/390 web servers or the WebSphere Application Server.

OS/390 Web Server

To retrieve XML from CICS using an OS/390 web server:

1. A client application connects to the host using TCP/IP and a URL that identifies the CICS host and HostBridge as the application to execute.
2. The web server passes the URL to the CICS Web Server Plugin.
3. The plugin uses the External CICS Interface (EXCI) to connect to the Business Logic Interface of CICS Web Support (CWS).
4. The BLI passes the URL to HostBridge.
5. HostBridge executes the CICS transaction and returns the transaction data as XML.

For organizations that have no need for the advanced features of WebSphere and do not want to offload the web server to a middle tier, this configuration provides an efficient method for XML-enabling CICS applications. This configuration may require programming on the host, depending on what functionality you want to include in a common gateway interface (CGI) program or Java application running under the web server.

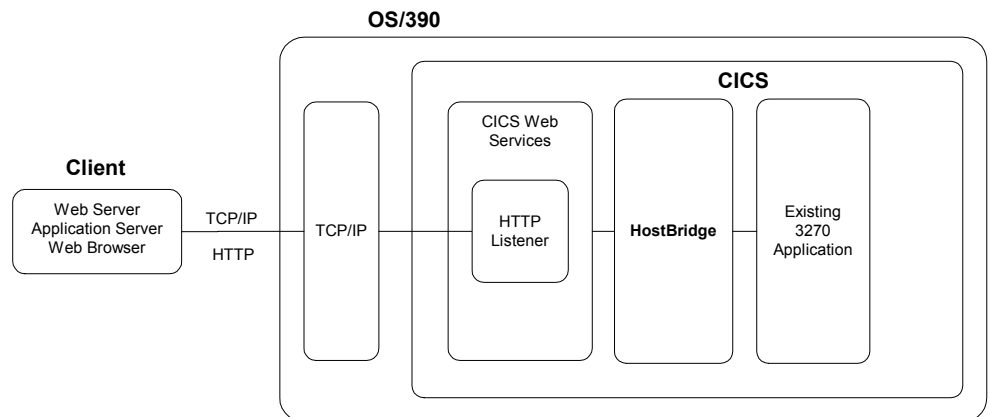
To retrieve XML from CICS using WebSphere:

1. A client application connects to the host using TCP/IP and a URL that identifies the CICS host and HostBridge as the application to execute.
2. WebSphere passes the URL to the CICS Transaction Gateway (CTG).
3. The CTG uses the External CICS Interface (EXCI) to connect to the Business Logic Interface of CICS Web Support (CWS).
4. The BLI passes the URL to HostBridge.
5. HostBridge executes the CICS transaction and returns the transaction data as XML.

For organizations that need the advanced features of WebSphere and do not want to offload the application server to a middle tier, this configuration provides an efficient method for XML-enabling CICS applications. This configuration may require programming on the host, depending on what functionality you want to include in the application running under WebSphere.

CICS Web Support

By using the CICS Web Support, a client application can make a direct HTTP connection to CICS without the intervention of an HTTP server on the middle tier or OS/390, as shown below.



This configuration requires TCP/IP on the host. In the diagram above, the following sequence of events results in the retrieval of XML data from CICS.

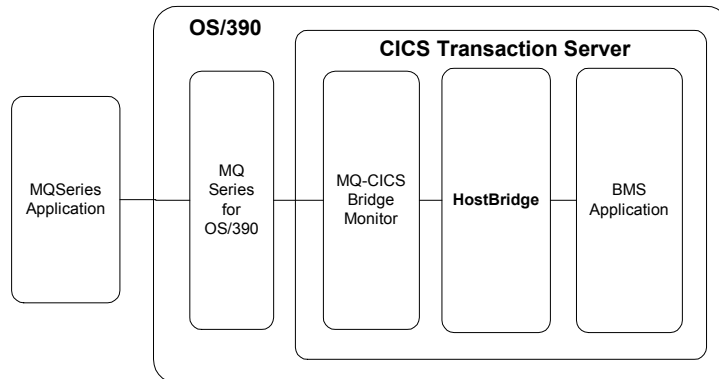
1. A client application connects to the host using assigned TCP/IP ports and a URL that identifies the CICS host and HostBridge as the application to execute.
2. CICS relays the request to the HTTP Listener.
3. The Listener and CICS Web Support translate the HTTP request from ASCII to EBCDIC and pass the data to HostBridge.
4. HostBridge executes the CICS transaction and returns the transaction data as XML to CICS Web Support.
5. CICS Web Support converts the XML data from EBCDIC to ASCII and returns the data to the client application.

CWS does not require programming, a web server on the host, or administration and support of a middle-tier server. However, CWS is

specifically designed to provide access to CICS applications and does not offer the full range of capabilities that web servers and application servers offer, such as support for CGI or Java applications.

MQSeries for OS/390

If MQ is present on both the client application and the host, then the client application can communicate with HostBridge using MQ messages, as shown below.



This configuration requires the client application to have either MQSeries Server or Client installed and MQSeries for OS/390 Version 2.1 or greater running on the CICS host with the MQ-CICS Bridge Monitor installed. In the diagram above, the following sequence of events results in the retrieval of XML data from a CICS application.

1. The client application issues an MQPut to the MQSeries queue manager running on OS/390 with a request to run a CICS transaction.
2. The MQ-CICS Bridge Monitor browses the queue and starts the HostBridge transaction.
3. HostBridge removes the message from the queue and executes the CICS transaction.
4. HostBridge builds the XML into a single message and places it in the queue.
5. The client application browses the queue and gets the message containing the XML from the queue.

Companies with existing installations of MQSeries can use this method to XML-enable their CICS applications without having to write their own application-specific bridge code or making any changes to their MQSeries applications.

Conclusion

HostBridge provides a flexible, scalable, secure, and easy-to-use solution that makes CICS applications usable in eBusiness by converting application data to XML. Unlike screen-scrapers, there is no need to identify field locations on a screen, so if CICS developers make changes to their applications by adding or removing data from fields on a screen, web applications that use HostBridge to access the CICS applications will not be affected. This saves time and money for development staff and reduces the chances that users will receive errors when they access your web applications.

Many current host application access solutions run on intermediate UNIX or NT web servers. Running alongside an SNA stack and a web server, these products receive a 3270 datastream, parse it, and then convert the datastream to HTML for presentation in web browsers. Because so much work is done on the single UNIX or NT machine, these products are unable to scale. They are sufficient for simple business-to-consumer (B2C) applications where occasional users logon to check account status or to see if a library book is on the shelves, but when it comes to the needs of B2B applications that produce hundreds of thousands of transactions each day, traditional web-to-host technology simply cannot keep up.

HostBridge combines mainframe and Internet security technologies into a single solution, so that using it to access CICS applications is actually more secure than using a TN3270 client. This means you can ensure safety of corporate data while simultaneously making that data available for B2C and B2B transactions.

HostBridge is a solution that allows other applications to access legacy data and use it in innovative ways to reduce training costs associated with teaching employees how to use complicated host applications, improve business efficiency, and unlock the valuable data companies spend years accumulating in their mainframes.