Modernizing CICS® and System z with Web Services/SOA

The HostBridge Approach

A HostBridge White Paper

Modernizing CICS® and System z with Web Services/SOA

The HostBridge Approach

Many of the largest organizations in the world bet their business on IBM System z, and most, if not all of them depend on the power and reliability of CICS, DB2, VSAM, and other System z resources to process billions of information transactions worth tens of billions of dollars every day. Given the tremendous business value of System z information resources, enterprises continually seek ways to extend mainframe usability and add even greater value to this core system.

Mainframe modernization using Web services/SOA has, in recent years, become one of the most effective, efficient, and economical means to reach these goals. In manufacturing, telecommunications, energy, high tech, insurance, government, health care, education, and other industries, organizations have achieved remarkable results modernizing the mainframe with Web services/SOA. To cite just a few examples:

- Using a Web services engine, the largest credit union in the world presents CICS applications and System z data – Fidelity banking apps, VSAM files, and DB2 data – to millions of members worldwide via the company's Web site, to 7,000 employees through a Universal Agent Desktop client, and to thousands of ATMs in all 24 time zones.

- A top international auto maker uses lightweight RESTful services for internal integration of a CICS lease management application, including VSAM files, with a Siebel CRM system. By enabling customer service reps to interact with CICS from within the Siebel interface, the services-oriented integration helps the auto maker boost employee productivity, accelerate customer-centric processes, and improve customer satisfaction and loyalty.

- A European health insurer uses SOAP services to make CA Datacom-based policy information available 24/7 to policy holders, physicians, other health professionals, and company personnel through an eHealth Community Cloud that will ultimately serve more than 7 million users.

Despite the apparent benefits and advantages of Web services/SOA, many enterprises have not embraced the approach. An October 2010 Research Note from Gartner, entitled "Service-Oriented Architecture as a Mainframe Modernization Strategy," recognizes this somewhat surprising situation and reasserts the fundamental value proposition of modernizing the mainframe via Web services/SOA. The Note offers four key findings for organizations that rely on System z:

- Use of technologies to wrap existing mainframe transactions can offer significant new uses of extant business logic implemented in legacy environments.

- Restructuring screen-based transactions can make them more usable in a number of modern business scenarios.

- IBM Specialty Engines (System z9 Integrated Information Processor [zIIP] and System z Application Assist Processor [zAAP]) can be leveraged, where appropriate, to reduce consumption of general-purpose MIPS when wrapping existing transactions.

- Mainframe customers should consider mainframe-resident wrapping products to maximize reliability and performance of the mainframe, and eliminate unnecessary network traffic.[1]

Types of Modernization: Presentation and Integration

In the current technology conversation, the term "mainframe modernization" is used in two different, quite contrary ways. Some use it to describe *migration* of data and information processes to other platforms. Others use it to describe *reuse* of existing mainframe applications and data. Reuse is a type of modernization in that mainframe data and business logic are rendered "new" when they are "wrapped" in modern data formats – XML, Web services, HTML. Repurposed with standard languages and protocols, mainframe data becomes interoperable with any information system that supports the same open standards – as virtually all now do. *Wrap and reuse* is the phrase used most often to describe this modernization approach, and it is in this sense that this paper uses "modernization."[2]

Wrap-and-reuse modernization divides into two broad categories – *presentation* and *integration*.

- *Presentation* is the repurposing of mainframe applications and data for presentation in a Web interface. The objective is to put high-value business logic and data at the fingertips of more people – employees across an organization, partners at remote locations, and customers anywhere using the latest devices and clients. With the advent of Web 2.0 technologies, mainframe data repurposed for Web presentation can participate dynamically in the newest Web server-based composites and mashups.

- *Integration* is a matter of wrapping mainframe data in industry-standard Web services languages and protocols so that the mainframe and its resources can be readily and universally integrated with other platforms, systems,

---

[1] Gartner Research, by Dale Vecchio, "Service-Oriented Architecture as a Mainframe Modernization Strategy," ID Number: G00174569, 29 October 2010.

[2] The mainframe modernization lexicon continues to grow and includes such terms as Web enablement, Web presentation, XML enablement, SOA enablement, services enablement, application integration, data integration, and others.

applications, and data. Integration in this sense encompasses application integration, data integration, and SOA integration.[3]

HostBridge and Integration "Environments"

HostBridge was founded in 1999 by Russ Teubner and Scott Glenn, current CEO and CTO respectively. At that time, both had years of experience conceiving, designing, and developing integration software for the IBM mainframe – from point solutions enabling communication between mainframe and Unix systems to one of the first products to dynamically translate 3270 and 5250 data streams to HTML for Web presentation.

In 1999, forging a new path, the two set out to design a technology that would integrate the mainframe using emerging interoperability standards. Their ultimate goal: *integrate anything mainframe with anything distributed*.

As a first step toward this goal, they conferred with long-time customers frustrated by the then-current state of integration technology. Based on these discussions, they defined fundamental principles that the company still lives by.

- Exploit the best integration technologies offered in CICS and System z

- Adhere to industry-best interoperability standards – XML, HTTP, SOAP, etc.

- Track evolutions in distributed systems and design appropriate solutions.

HostBridge continues to follow these principles. As these three technology "environments" have evolved over time, HostBridge has added support for new mainframe tools and technologies, enhanced its Web services capabilities, and, as needed, developed specialty connectors for leading enterprise systems on the distributed side, e.g., the HostBridge PegaConnector and SiebelConnector.

While the effort to keep pace with technological change in never-ending, the clarity of the original vision has certainly been validated.

The Early Integration Problem

In those early conversations with customers, a frequent topic of discussion was a predominant problem that plagued middle-tier integration solutions – screen scraping.

Mainframe screen scrapers captured and used data from the 3270 data stream. Designed to enable display of mainframe data on fixed terminal screens, the data stream included both the backend data and business logic from source applications as well as presentation logic that told the terminal how to display the data on screen. To

---

[3] Through the years, organizations have employed many integration models and approaches. In the past, we would have included a review of these in a paper of this type. However, now that the Web services/SOA approach is generally understood (even though adoption may lag), we'll forego discussion of this well documented history.

this day (yes, there are still some out there), screen scrapers use this presentation logic to present data on desktop clients in rigidly defined rows and columns.

Problems occurred whenever mainframe applications were changed and those changes altered screen geometry. If a data field or input field was not in the location expected by the screen scraper, the connection was broken. The breakage could be fixed, but the repair would require code-level reconfiguration of the screen scraper every time the mainframe changed, a costly duplication of developer labor. And breakage was far too frequent.

Mainframe screen scrapers also had another characteristic problem. They installed in the middle tier. As more organizations adopted them, performance problems became increasingly apparent. Every transaction with the mainframe required intermediate processing in the middle tier.

For organizations used to – and reliant on – mainframe performance and reliability, these early integration solutions did not meet expected standards on either front.

HostBridge: XML Engine for CICS

At the time HostBridge was founded, two simultaneous developments opened the door to a different approach to mainframe integration. IBM introduced the Link3270 Bridge interface in CICS Transaction Server v1.2/1.3, and XML emerged as a new standard for system-to-system interoperability. By leveraging the Link3270 Bridge and XML together, HostBridge took a significant stride toward its ultimate goal of integrating anything mainframe with anything distributed.

The first release of the product HostBridge, in 2000, pioneered auto-generation of XML from CICS applications. Doing so, it redefined the benefits of mainframe integration, enabling mainframe levels of performance, bulletproof reliability, and unmatched fidelity to source applications.

HostBridge installs on the mainframe (current versions can run under CICS or inside z/OS) and performs like other mainframe applications (performance may be further enhanced with HostBridge for zIIP, discussed below). In integration environments, mainframe-resident HostBridge is able to deliver significantly higher performance than middle-tier products, with their unnecessary processing layer between mainframe and distributed system. With HostBridge, distributed systems talk directly with the mainframe.

By exploiting the Link3270 Bridge interface, HostBridge was able to intercept CICS data *before* the presentation layer was added, thus freeing the data from row-column rigidity, enabling dynamic integration of CICS, and eliminating the breakage that plagues screen scrapers. HostBridge also made dynamic integration a reality for the first time. Even as mainframe conditions change, HostBridge delivers data straight from the application layer, not the presentation layer. Because it captures and replicates actual application output in real time, HostBridge achieves the highest levels of reliability while eliminating the need to rewrite integration code every time the mainframe is changed.

Most importantly, the HostBridge XML Engine makes CICS applications integration-ready with any distributed system that supports the XML standard. HostBridge transforms CICS application output to XML metadata (screen and field names) and transactional data (values). HostBridge generates XML from any CICS application – BMS, 3270, COBOL, PL/1, COMMAREA, CA Ideal, CA ADS/IDMS, and others. The engine also supports SOAP conversion, enabling distributed applications to invoke CICS transactions as SOAP services. In most circumstances, HostBridge manages mainframe-distributed communication using HTTP requests over TCP/IP. For mainframe applications that use COMMAREA, HostBridge also supports EXCI connectivity.
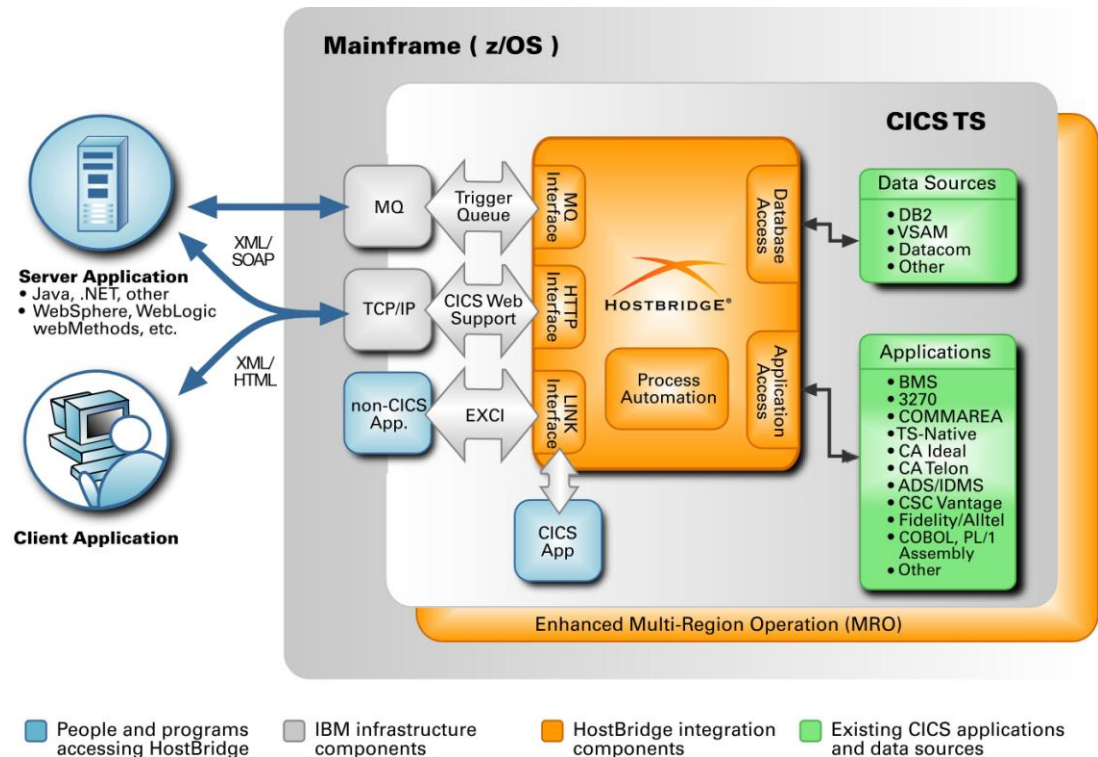

Figure 1. HostBridge Integration Architecture

XML generated by HostBridge is ready for consumption and use by virtually any Web services-based development technology. On the distributed end, it is fully interoperable with any distributed system that supports XML.

The HostBridge XML Engine, also known as HostBridge Base, remains a fundamental component of HostBridge, enabling customers to use XML auto-generated from CICS applications and also providing the display data for the HostBridge Eclipse IDE (discussion below).

HostBridge Process Automation Engine: JavaScript-Based Web Services

The HostBridge Process Automation Engine is a scripting and Web services engine based on industry-standard JavaScript. When HostBridge first introduced the engine under the name HostBridge Extended, or HBX, the team was responding to actual needs. When a few HostBridge customers asked about extending the capabilities of HostBridge Base with scripting or programming that would automate complex CICS transaction processes, the team faced a second major decision. The value of automating CICS processes was obvious; the question was what technology to standardize on.

The JavaScript Standard

In weighing options, the HostBridge team was determined to remain true to its founding principles – exploit the best mainframe technologies and the best interoperability standards so long as they would not sacrifice performance, reliability, and fidelity. As analysis proceeded, potential new benefits became development principles – flexibility, functionality, and improved standards-based interoperability.

The team investigated languages like REXX (Restructured Extended Executor). Even though open source REXX interpreters existed, open source was not the same as open standard. As an IBM proprietary language, it would have inherent limitations and would tend to exclude a broad and growing population of integration architects and programmers who favored open standards.

Modeling technologies were also considered, but they had limitations as well, particularly with respect to terminal-oriented CICS applications, the type most customers were using and sought to automate. Designed for interactions with human beings at terminal screens, these applications are notoriously fine-grained, often including highly complex transaction sequences with additional exceptions and error conditions. While human beings could learn to recognize and respond to these very detailed sequences, modeling software was too coarse-grained, too high-level. It was good at the big picture, less so at the brushstroke level.

From the perspective of HostBridge principles – choosing the interoperability technologies that would work best in both the mainframe and the distributed worlds – the best and only choice was JavaScript. The choice was questioned by some (the time was 2004 after all, and JavaScript was perceived as a Web language good on the client side), but the tune soon changed.

Some reasons for the choice were true at the outset and remain true today. JavaScript is widely known to diverse sets of developers (.NET, Java, Web). It is easier to master and use than almost any other programming language, especially mainframe languages. JavaScript is flexible, "speaking" all the industry-standard languages like HTTP, XML, SOAP, REST Web services, and it is supported by every major platform is use today. JavaScript is also standardized by ECMA, the European Computer Manufacturer's Association (the latest specification being ECMAScript Edition 3/JavaScript 1.8).

Other reasons for the choice were prescient. The client-side technology has matured and emerged as a powerful option on the server side, with significant advances in server-side engines/interpreters. Its strong following also continues to grow, even including some mainframe developers. In addition, JavaScript is now fully supported by System z and CICS, and is in fact a full-functioning citizen of the mainframe world.[4]

The HostBridge JavaScript Engine

The JavaScript-based HostBridge Process Automation Engine installs with HostBridge Base on the mainframe, under CICS or inside z/OS (it can also install as an Enclave SRB, important for HostBridge for zIIP, discussed below). Both a development facility and runtime engine, it allows integration architects, process designers, and application developers to use industry-standard ECMAScript/JavaScript to write integration scripts that expose mainframe applications and data as Web services or as new dynamic applications able to carry out high-order business processes.

As its name implies, HostBridge Process Automation was originally designed as a way to orchestrate, automate, and integrate complex CICS transaction processes – i.e., transaction micro flows – as a single Web service.

Terminal-oriented CICS applications – those requiring interaction by a human being at a terminal screen – typically feature transaction processes involving dozens or even hundreds of screens. Integrating these micro flows was problematic, especially for middle-tier screen scrapers. Whether the "user" was a human being working at a computer or a screen scraper sending requests to the application, interaction proceeded one screen at a time. The process was inherently slow, and when developers tried to automate the process, execution of multiple transactions resulted in response latency and poor performance. If a process involved twenty screens, twenty full transits of the network were needed to complete the process.

Running on the mainframe, HostBridge was literally in a better position to automate these processes, and the HostBridge JavaScript engine made it possible for developers to write scripts that could orchestrate any number of transactions in the application micro flow into a single service.

A comprehensive JavaScript facility, the HostBridge Process Automation Engine fully supports object-oriented scripting/Web services. Supported services types and protocols include XML, SOAP, SOAP 1.2, REST/RESTful services, Ajax, JSON (JavaScript Object Notation), Atom, HTML, and E4X (ECMAScript for XML). The HostBridge engine also includes predefined objects that can perform virtually any operation a COBOL program can perform. These include CICS command-level access, CICS transaction execution, COMMAREA program execution, a Simple API for XML (SAX) parser, SOAP RPC response writing support, and more.

Over and above the orchestration and automation of CICS processes, the HostBridge Process Automation Engine significantly extends HostBridge access capabilities. Via

---

[4] For more on JavaScript and its benefits, including dramatic productivity gains by developers and several real-world use cases, see the recent three-part article series by Russ Teubner, available at www.hostbridge.com/index.php/news/articles.

Web services and/or JavaScript scripts, it can invoke CICS and IMS applications as well as DB2, VSAM, Datacom, and DL/I data (via HostBridge Data Access Modules, predefined objects that are callable within a HostBridge script or Web service).

HostBridge Eclipse IDE

The HostBridge Eclipse IDE – an Eclipse plug-in that is an integral component of the HostBridge Process Automation Engine – provides developers with a familiar, easy-to-use tool for developing process automation scripts and/or Web services that expose mainframe data to distributed systems.

As was the case when standardizing on XML and JavaScript for the HostBridge XML Engine and Process Automation Engine respectively, the HostBridge team made a purposeful decision when standardizing on an integrated development environment. They chose Eclipse because, on the one hand, the environment is particularly suited to work with System z resources since both Rational Application Developer for System z and CICS Explorer are Eclipse-based; on the other, Eclipse is a widely adopted standard and it designed for extensibility, e.g., supporting many programming languages by means of its plug-in architecture. With its broad appeal to both mainframe and distributed developers, it was the right choice for HostBridge and HostBridge customers.

Developers use the HostBridge Eclipse IDE to author and edit scripts; interactively compile, execute, and debug scripts; save scripts to the mainframe as VSAM files; convert COBOL copybooks to JavaScript and XML; and generate Web services (SOAP, WSDLs, and multiple RESTful services).

A unique and powerful capability of the HostBridge Eclipse IDE is its use of HostBridge XML to provide a dynamic view of live CICS application output. Called the HB Transaction Explorer, this feature makes it possible for integration developers to write scripts and services that replicate source applications and transaction sequences with unparalleled fidelity. Using the Explorer's visual interface, developers can actually see and interact with the most complex transaction micro flows – including all error and exception conditions – and develop services at a level of granularity that development products like Service Flow Modeler cannot begin to match. The result is integration with the highest reliability and highest fidelity to source applications.

HostBridge for zIIP

HostBridge for zIIP – HostBridge integration software running on the IBM System z Integrated Information Processor (zIIP) – represents another choice to meet evolving customer needs. The zIIP specialty engine was first introduced by IBM to help reduce the costs of System z processing by shifting DB2 workloads to the zIIP. More workloads were added to the eligible list, and now they include many integration workloads .

Organizations using zIIP specialty engines can employ HostBridge for zIIP to transition significant portions of their integration workloads to the zIIP. The HostBridge XML Engine and Process Automation Engine can both run on the zIIP, and all scripts/Web

services that customers write with HostBridge, including dynamic, script-based applications, are zIIP-eligible.

With HostBridge for zIIP, all the benefits of the zIIP accrue to eligible HostBridge processes and integration/SOA workloads. These benefits include reduced processing costs, lower mainframe TCO, and potentially improved performance depending on an organization's general purpose processor capacity (shifting workloads from a lower-capacity GPP to the zIIP may result in higher processing performance).

Mainframe Presentation: HostBridge WIRE

Thus far, we have focused on industry-standard Web services technologies – XML, HTTP, JavaScritpt, SOAP, REST, etc. – for "backend" integration, i.e., *anything mainframe with anything distributed*. But "anything distributed" is an inclusive category, and we have left out one of the most important distributed technologies – the Web.

We did so because Web integration is often seen as integration of a different type – not because the technologies are all that different but because the value proposition is unique. When we talk about mainframe Web presentation – the integration of System z applications and data in modern Web interfaces – the conversation shifts from programs to people.

In those olden days of power users at fixed terminals, no one imagined the mainframe as a platform accessible from anywhere by anyone. Times have changed. Where power users had to train extensively before gaining privileged access to the mission-critical mainframe via dumb terminals, today's employees work at desktop and Web clients that can display any number of applications – enterprise, Internet, and composites of both – presented in intuitive, immediately usable interfaces. Where power users in call centers were once the only conduit between caller and data, today's customer expects 24/7 self-service through a Web interface that mashes up all sorts of relevant feeds. Today, navigating and managing the variety of resources that employees, customers, and partners expect depends less on the user's training than on the work performed by the Web integration/application architect or developer.

Understanding that Web presentation of mainframe resources requires particular attention to the latest Web technologies, HostBridge set out to develop an engine specifically for Web presentation. Once again, choices had to be made – platform, development environment, mainframe access, and so on. The result was HostBridge WIRE, the Web Interface Rules Engine.

WIRE is a .NET programming and runtime environment enabling presentation of mainframe applications and data in three different modes, from Classic Mode for power users who still do mountains of crucial work, and Standard Mode with a more Web-oriented display for less experienced users, to Enhanced Mode for users who demand and gain productivity benefits from a rich, full-featured Web or Web 2.0 experience.

Uniquely among HostBridge products, WIRE does not run on the mainframe but on Microsoft IIS Web Server. This architecture was chosen deliberately to maximize benefits to the WIRE customer. Here's how. Consuming HostBridge XML by design,

WIRE delivers all the performance, reliability, and fidelity advantages of HostBridge.[5] At the same time, as a .NET application running under Visual Studio 2008, WIRE also offers one of the most powerful interface tools of the Web development world.
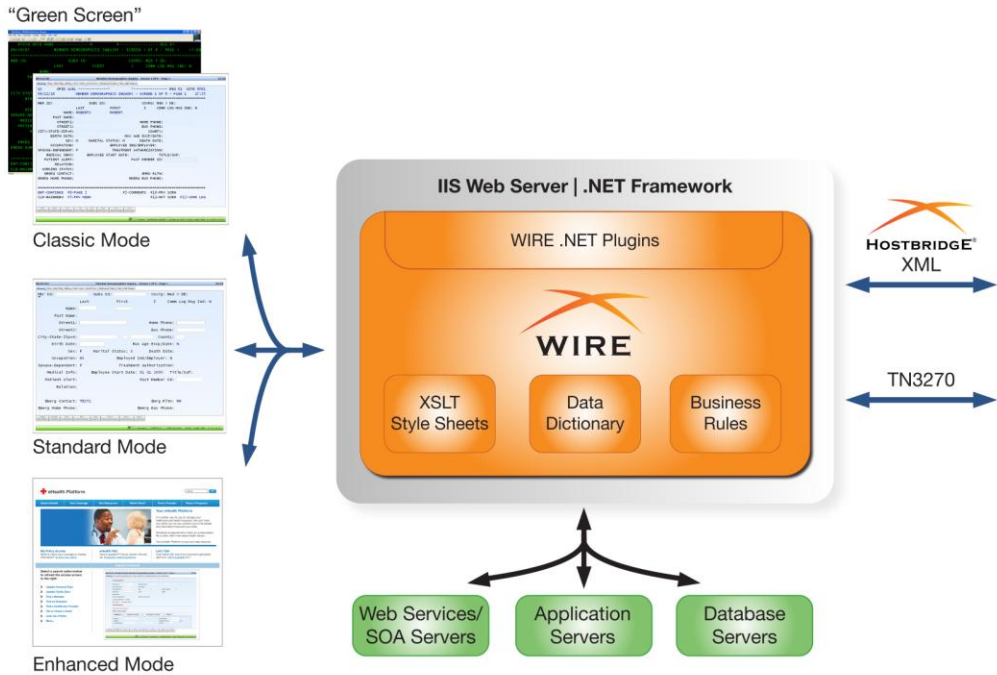

Figure 2. WIRE Architecture.

With the WIRE Toolbox, Data Dictionary, and ASP.NET custom controls, Web programmers can easily develop and deploy graphical user interfaces, or composite applications or mashups, from HostBridge-supported mainframe resources as well as other distributed Web, application, and data resources accessible through the .NET framework.

WIRE extends all of the HostBridge integration benefits – performance, reliability, fidelity, and flexibility – to mainframe Web presentation.

Conclusion

Organizations that build their business on the System z platform trust its power, performance, and reliability. Those who seek to modernize the mainframe – who need to integrate *anything mainframe with anything distributed* – trust HostBridge to extend the same benefits to critical integration projects. Whether customers are considering integration or Web presentation, HostBridge has been thoughtfully designed and developed to extend the mainframe's power, performance, and reliability – and add the flexibility and fidelity that result from choosing the best industry-standard interoperability technologies.

---

[5] WIRE can also consume TN3270 data streams for customers preferring that alternative.