

# High-Fidelity CICS® Integration

## HostBridge™ and IBM, A Brief History

---

A HostBridge White Paper

By Russ Teubner, CEO



866-965-2427

[info@hostbridge.com](mailto:info@hostbridge.com)

## **Copyright Notice**

Copyright © 2008-2009 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, 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: 7/7/2009

## **Trademarks**

HostBridge and the HostBridge logo are trademarked by HostBridge Technology. All other trademarks mentioned are property of their respective owners.

# High-Fidelity CICS® Integration

## HostBridge™ and IBM, A Brief History

---

Summer 2009 marks the 40th birthday of CICS®, and there's every reason to be confident that CICS will be around for a long time to come. The largest enterprises in the world bet their business daily on the robust, reliable transaction processing of CICS, which handles more than 30 billion transactions per day at a value of more than \$1 trillion per week. And increasingly, organizations are extracting greater value from trusted CICS applications and data thanks to services-based integration.

Of course CICS integration hasn't been a cakewalk, yet today there's great reason to celebrate. Although legacy CICS applications have posed any number of challenges on the road to services-based integration, we've come a very long way, and CICS can now legitimately claim as a full-fledged participant in services architectures of every shape and size.

Over the past decade, HostBridge™ has had a significant role in enabling Web-services integration for CICS. And our passion for high-fidelity CICS integration has reaped rewards for customers in every major business sector. So to honor CICS and because we hope the story has relevance and value for you, we offer this brief history of how HostBridge and IBM make high-fidelity CICS integration real.

### The Genesis of HostBridge – 1999-2002

HostBridge started as a "clean sheet" design to solve a specific CICS integration problem. In late 1999, on a sunny day in San Francisco, at a meeting with Scott Glenn and I (HostBridge co-founders), Marshall Gordon (then Chief IT Architect at Delta Dental of California), stated, "We will no longer deploy any CICS integration solutions that rely on 'screen scraping' techniques."

Marshall didn't need to elaborate. Having worked in the industry for years, Scott and I understood. Screen scraping was the fundamental flaw in many mainframe integration solutions.

The essence of screen scraping is its dependence on row/column coordinates to specify the location of data on a screen and create a relationship between two programs. Imagine a CICS transaction that displays data as part of a screen image, and a distributed program that masquerades as a terminal so it can capture the screen image in memory and copy (or "scrape") the data off it. By using row/column coordinates to reference the screen data, the distributed program creates a static, binding relationship between itself and the presentation logic output by CICS. If the CICS transaction is changed such that it alters the screen format in any way, then the binding will likely be incorrect and the distributed program will likely fail.

Because of this limitation, screen scraping led to what could best be described as "application rigor mortis," whereby applications could not evolve over time because

too many things broke when they did! Not to mention the performance of such solutions, especially in high-volume environments, was usually very poor.

During our meeting, we discussed two exciting new approaches to solving these problems – the use of XML to exchange information between an existing CICS transaction and a distributed system (remember, XML was still big news in 1999) and a new “bridging” feature that IBM introduced in CICS Transaction Server v1.2 and formalized in CICS TS v1.3. Using this feature in conjunction with a supported API, a program could intercept CICS Basic Mapping Support (BMS) SEND and RECEIVE commands *before* the terminal data stream, with its presentation logic, was generated as output or expected as input.

*The implication for integration of terminal-oriented CICS applications was huge. Interaction with CICS by distributed systems would no longer be rigidly dependent on screen geometry – i.e., screen-scraping.*

Later that evening, Scott and I completed the first HostBridge block diagram. The initial version of HostBridge would be designed to exploit the Start Bridge interface within CICS TS v1.3 for the purpose of XML-enabling CICS BMS applications.<sup>1</sup>

## Guiding Principles

This brief story illustrates the principles that have guided HostBridge the company, HostBridge the product, and HostBridge the partner since the beginning:

- Start with a full understanding of what *real* customers do with *real* CICS applications.
- Exploit the best of what CICS TS has to offer natively.
- Assess the latest trends in Web-based application development.
- Adhere to relevant integration standards (HTTP, XML, SOAP, REST, etc.).
- Retain our obsession with “high-fidelity” integration.

## High Fidelity Integration

I often use the phrase “high-fidelity” when discussing CICS integration, but it may be new to some people, so a brief bit of explanation is in order. “High-fidelity,” when applied to video or audio, implies the reproduction of sights or sounds with minimal distortion and maximum resolution.

---

<sup>1</sup> **The HostBridge Patent:** On December 27, 2005, HostBridge was awarded US Patent No. 6,981,257 B2. In the wonderfully arcane language of “patent-speak,” the HostBridge patent describes: “A system, method and apparatus to facilitate the invocation of existing CICS BMS transactions and deliver the executed transaction output to a requesting application as a standardized XML document.” The broad objectives of the patent are to: allow enterprises to integrate CICS transactions into scalable e-business applications; facilitate communications between CICS and non-CICS platforms and applications without screen scraping; provide for XML-based communications between CICS transactions and client applications without CICS application, transaction, or system modification.

The concepts of minimal distortion and maximum resolution are directly applicable to CICS integration as well. Integration technologies can “reproduce” CICS applications with either a high degree of clarity and resolution – i.e., fidelity – or with lower resolution and more distortion. Screen scraping injects a high degree of distortion and/or loss of resolution with respect to originating CICS applications and integration. Other integration techniques can diminish fidelity as well.

At HostBridge, high fidelity has always been our goal. And as this brief history shows, our efforts over nearly a decade have paid off, so much so that today we confidently say we *make high-fidelity CICS integration real*.

## **Evolving Support for CICS – 2001-2005**

As IBM has evolved the bridging capabilities within CICS TS (e.g., Link Bridge), HostBridge has been fully supportive. In fact, because HostBridge customers are among the heaviest users of the Link Bridge interface, and because HostBridge probably has more experience with the bridging capabilities of CICS TS than any other ISV, we have helped drive Link Bridge functional requirements of CICS TS.

A primary focus of HostBridge has always been on integration of CICS terminal-oriented transactions with distributed systems. Initially we focused on BMS-based applications, and later expanded to fully support non-BMS applications as well – even non-CICS applications. Working closely with the CICS product development team, we have also been able to add support for capabilities that were once considered “impossible” (e.g., the PAGE, ACCUM, and PAGING options of the EXEC CICS SEND command). Along the way we did not limit ourselves to terminal-oriented applications. Real customers normally rely on a mix of application types, and in response to their needs, we added support for LINKable COMMAREA programs early on.

Of course we continue to make sure our products support the broadest range of terminal-oriented applications. HostBridge is the only product of its type that is CA smart-certified for use with all the CA application platforms that run on the mainframe – CA Ideal™, CA IDMS™, CA Telon®, and CA Gener/OL™. HostBridge also supports a number of CICS-based third-party applications, such as Fidelity/Alltel and CSC Vantage, that use proprietary techniques for screen description/mapping (other than BMS), once again eliminating the need to scrape screens and overcoming breakable relationships based on screen geometry.

## **HostBridge Process Automation – 2004**

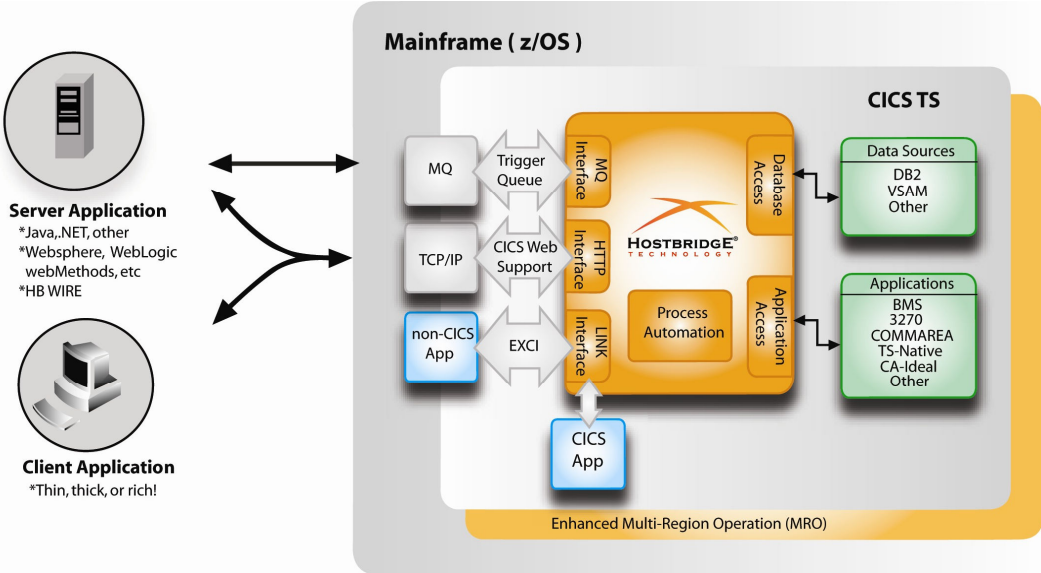
Customers were pleased with the value proposition of HostBridge – XML-enabling CICS applications and data – and its technical merits. As more customers implemented the product, however, needs and issues continued to emerge, particularly involving orchestration of the “micro flows” intrinsic to CICS transactions and sequences of HTTP/XML requests and responses.

Terminal-oriented applications were designed for use by human operators. As a result, a single interaction with such an application rarely embodies the full scope of

work that must be performed to accomplish a business process. The business process usually requires the execution of multiple transactions as part of a conversation or “pseudo-conversation” (a CICS technical concept). Thus, a detailed sequence of steps must usually be performed. These steps are often referred to as the “micro flows” necessary to accomplish a business service.

In the early days of HostBridge, customers would typically write a server-based process, such as a Java application running under WebSphere, to automate the exchange of a series of HTTP/XML requests and responses with HostBridge. However, when a distributed program was used to automate the micro flows required to implement a business service, there were certain ramifications. First, if the process was complicated, involving many steps, it would require numerous HTTP requests/responses, and the aggregate latency associated with execution of the overall business service could become excessive. Second, CPU consumption associated with the processing of individual HTTP requests/responses on the z/OS system could be high.

In response, we introduced a CICS-based Process Automation Engine as part of HostBridge version 4 in 2004. This component (initially referred to as HostBridge Extended) allows a customer to describe/codify a sequence of interactions using ECMAScript (a.k.a., JavaScript).<sup>2</sup>



To some industry observers, implementing a native ECMAScript engine within CICS seemed odd, and the question arose: why scripting? This paper answers the question in full, but now is a good time to recall some basic characteristics of scripting. A scripting language is a programming language that is intended to sequence, control, or otherwise “glue together” various software applications or data sources. Scripts enable the behavior of the underlying application/data to be adapted to a particular need. Scripting languages enable programming at a higher

<sup>2</sup> The name "ECMAScript" was a compromise between the organizations involved in standardizing the language, especially Netscape and Microsoft. Brendan Eich, the creator of JavaScript, is on record as saying that "ECMAScript was always an unwanted trade name that sounds like a skin disease."

level of abstraction than languages like COBOL, C, or Java. Scripting languages are also easier to learn and faster to code.

Equally significant, the scripting approach was perfectly consistent with the principles that have always guided our thinking about CICS integration. Real customers were using it to overcome real integration challenges, and their insight became our direction. As it turns out, we scripting advocates were prescient about its capabilities and benefits in the CICS environment. As scripting has gained acceptance, HostBridge Process Automation has added more objects that allow a HostBridge script to interact with terminal-oriented transactions, COMMAREA-based programs, files, databases, and any CICS-managed resource.

## CICS TS v3 and HostBridge – 2005-2007

We are now at the point in this history where we can begin to consider how HostBridge complements the capabilities of CICS TS v3. We can consider it a second phase in the evolution of CICS integration as we move from XML-enablement to Web services-enablement.

In CICS TS v3, IBM added a number of significant features under the heading of “Web services enablement.” For example, CICS TS v3 added an elegant and robust transport management architecture built around new CICS components (i.e., pipelines, channels, and containers). Support for HTTP 1.1 was added, and support for SOAP was formalized. Layered on top of these capabilities were new features to implement formalized Web services. These features were implemented via new resource definitions (e.g., WEBSERVICE), EXEC CICS commands (e.g., EXEC CICS WEBSERVICE), and the CICS Web Services Assistant utility program. Collectively, these infrastructure improvements laid the foundation for CICS TS to function as a first class citizen in the Web services world.<sup>3</sup>

The explosion of new features in CICS TS v3 became the foundation of numerous HostBridge enhancements, thus providing additional value to HostBridge customers. For example, HTTP 1.1 support improved HostBridge throughput; channels and containers provided new means to handle large blocks of data within HostBridge; HTTP and SOAP enhancements made it easier to issue outbound service requests from HostBridge scripts; and the additional transport layer security options gave HostBridge customers many new options.

However, for all its new features and capabilities, CICS TS v3 is not a “full weapons platform” when it comes to CICS application integration. To illustrate this fact, let’s pose and reflect on the answer to a simple question. Without modifying your existing applications, what kinds of Web services can you build using CICS TS v3? The answer is simple. A Web service that invokes a single COMMAREA program. That’s good, but it’s just a start!

---

<sup>3</sup> I used the word “infrastructure” deliberately, because it describes the correct relationship between HostBridge and CICS TS: HostBridge *builds upon* the Web services infrastructure provided by CICS TS. Our focus has always been on the layers *above* Web services infrastructure. Our focus is *application integration* – not SOA infrastructure.

The component of CICS TS v3 that facilitates deployment of a single COMMAREA program as a Web service is the Web Services Assistant (WSA). WSA provides two utility programs: one that generates a WSDL from a language structure (e.g., a COBOL copybook) and one that generates a language structure from a WSDL. Additionally, these utility programs assist with the generation of the CICS artifacts/definitions that permit deployment of the Web service (e.g., the WSBIND file).

WSA is a great tool for those organizations (a) that have lots of COMMAREA programs and (b) whose individual COMMAREA programs fully implement the required business service. But again, as real users will attest, there's a lot more to CICS. It is crucial to remember that *WSA does not allow you to build a Web service from multiple COMMAREA programs or from terminal-oriented applications*. So how do you do that? Enter tools like Service Flow Feature and HostBridge.

## Service Flow Feature and Service Flow Modeler

On November 22, 2005, IBM announced the CICS Service Flow Feature (SFF). The title of the announcement is important: "IBM CICS Service Flow Feature enables composition of CICS applications to create CICS business services." The title reflects SFF's primary role as service composition, *without* dictating how the resulting service is deployed. To quote the announcement letter:

CICS Service Flow Feature, comprising a tooling component and a run-time component:

- Is a business service integration adapter for CICS applications.
- Enables integration developers to create CICS business services by composing a sequence of CICS application interactions. These can be integrated in Service-Oriented Architectures (SOA) or business process collaborations.
- Delivers a graphical integrated development environment for composing the CICS application interactions, a generation facility for creating a run-time application, and a run-time component that exploits CICS interfaces to support the service flow.

Keep in mind, the driving idea behind Service Flow Feature is the composition of a "coarse grained" service from existing "fine grained" CICS applications, i.e., terminal-oriented applications with their intrinsic micro flows.

The tooling component of SFF is Service Flow Modeler (SFM). Here's the IBM explanation of SFM: "The CICS Service Flow Modeler delivers the flow modeling and mapping capabilities required to transform the behavior of the application components to form a business service without changing the underlying implementation." At first blush, it sounds like SFM is some sort of a really big magic wand – it can transform the behavior of an existing CICS application without having to change the actual application. Cool! Sign me up!... Back to reality. SFM doesn't auto-magically transform anything in the sense that your existing CICS applications behave or operate any differently. The magic (and I'm not opposed to calling it that) is in orchestrating the execution of existing CICS applications according to a specified flow.

That's why another description of SFM sounds more correct: "a graphical modeling integrated development environment that enables the creation of CICS business services by composing a flow of CICS application interactions."

## Composing Services

When considering CICS services composition, the first question you have to ask is *what exactly are we composing?* It's not difficult to answer, but it is critical because the *what* answer affects the answers to every other question you will ask, particularly whether modeling is the right *how*.<sup>4</sup> So what are we composing? The two main CICS application types – COMMAREA programs and terminal-oriented programs.

COMMAREA programs – with their well-defined inputs, outputs, and operational characteristics – may, in many situations, be well-suited to a modeling approach. However, as the operational level of these programs becomes more complex or detailed, modeling tools start to get in the way.

CICS terminal-oriented applications are another story. Remember, terminal-oriented applications evolved over many years with an eye toward solid business value, optimal execution efficiency, and acceptable productivity of reasonably well-trained end users. These applications often defy the black-and-white, square-edged, if-then-else mentality of modeling tools. They were not written to be sympathetic to, or viewed through the lens of, a modeling tool. When modeling tools are used to describe the operation of such applications, the tool itself usually becomes the limiting factor as to what can be accomplished.<sup>5</sup>

A modeling tool is perfect for describing the macro-level flows of a business service. Macro flows include the top-level flow of the service, top-level decision points (conditional processing and looping constructs), and the invocation of coarse-grained CICS programs. However, using a modeling tool to describe the operation of fine-grained CICS terminal-oriented transactions is like trying to use a hammer to turn a screw.

You can't get away from the fact that CICS terminal-oriented applications are rather unique programmatic "things." They operate in certain odd ways, though for very good reasons. And they have all the operational consistency of the human being who authored them. No degree of "modeling" will change these facts. The only issue is how to embrace these applications most effectively and efficiently.

---

<sup>4</sup> *What* and *how* are just two of the questions addressed in recent HostBridge white papers, notably "The Role of HostBridge in the CICS Services Composition," an anti-white paper from which the other papers named in this note, and the paper you are reading, are derived. Answers to the question, where are we deploying the service? can be found in "Composing CICS Services: The Question Is, Where?" Answers to *how* can be found in "Composing CICS Business Services: SFM Modeling and HostBridge Scripting." You'll find these white papers at [www.hostbridge.com/index.php/library/whitepapers](http://www.hostbridge.com/index.php/library/whitepapers).

<sup>5</sup> In the interest of technical accuracy, I'll qualify this statement, a tiny bit. If all you are going to do is invoke one or two visual transactions and extract a few data items off each screen (the sort of scenario that vendors like to use during a sales demo), then you can probably "model" that activity. However, if you go beyond such simple scenarios (as most real-world applications do), you've got a problem.

Micro flows include the low-level flows of the service (the specific steps required to operate or interact with a terminal-oriented application) as well as the low-level data gathering, conditional processing, and looping constructs that are also required. In such cases, a simpler and more direct approach is called for.

If your business processes are based on CICS terminal-oriented transactions, you've got a radically unique problem to solve, and it requires the appropriate tool.

## **The Solution: Modeling AND Scripting**

So what is the right way to describe the operation of a CICS terminal-oriented application? Take a deep breath. The answer is... SCRIPTING! But not just scripting per se.... *It is scripting used to intelligently operate a series of terminal-oriented transactions and access data within the context of a modeled CICS business service.*

This should come as no surprise since I've already described that a key feature of HostBridge is its Process Automation Engine, a facility for developing and executing integration scripts within the CICS environment. A HostBridge integration script can access any CICS visual transaction, non-visual program, or data source – and many non-CICS resources. Using the HostBridge Process Automation Engine to compose integration scripts, system architects, process designers, and application developers can create services that automate and aggregate existing fine-grained transactions and data sources with the highest degree of fidelity.

Why is scripting rather than modeling the right choice for terminal-oriented CICS applications? Scripts are very concise. They are able to codify the well-defined steps that a human operator goes through to operate a terminal-oriented transaction (e.g., enter value in field “x,” press Enter key, get value from field “y,” etc., etc.). Scripts allow a human to break down a problem in a way that makes sense. Each step is visible and can be easily comprehended as part of the overall process. On the other hand, modeling diagrams with any degree of real-world detail or complexity tend to become large and unwieldy. The only way to deal with this is to create embedded models (models within models). But when you are working on the lowest level model, it is almost impossible to comprehend how it corresponds and interacts with a higher level model. As a result, the overall process becomes obscured.

Both modeling and scripting have their place in a high-volume, high-fidelity CICS integration architecture. But when it comes to intelligently operating a series of terminal-oriented transactions, scripting wins hands down.

## **HostBridge and SFM – 2007**

During 2007, IBM and HostBridge collaborated to bring these two approaches – modeling and scripting – together. First, we asked what would it look like if SFM and HostBridge could work together, allowing their respective strengths to be leveraged? The obvious answer was this: allow a HostBridge script to be invoked as part of an SFM flow. Specifically, allow a HostBridge script to be the implementation of an “Invoke node” within an SFM flow. EUREKA! This would be a wonderful combination of product strengths.

But how should we do it? Technically, a HostBridge script could already be invoked as part of an SFM flow in either of two ways: (1) invoke a HostBridge script as a LINKable COMMAREA program or (2) invoke a HostBridge script as a Web service. Both approaches would work, but both were ruled out because they were deemed to be either too crude or too resource-intensive.<sup>6</sup>

We – HostBridge and IBM – wanted a more ideal approach that would reconcile SFM’s bias toward LINKable COMMAREA programs and HostBridge’s thirst for metadata. We agreed that the most elegant and generic way to integrate a HostBridge script into an SFM flow would be to implement a new concept within SFM: a generic node that receives both data and metadata.

The starting point of such an ideal solution was another feature that IBM added to SFM in v7: support for LINKable programs that exchange data using CICS TS Channels and Containers. Within SFM, such a program is modeled by an Invoke node with multiple input/output messages. At run time, when the SFM-generated COBOL program LINKs to the target program, it passes to the target program a channel containing multiple containers (one message per container).

This mechanism allows SFM to pass a collection of information to a LINKable program that is more robust than a simple COMMAREA. If HostBridge were the target program, all sorts of interesting information (e.g., metadata) could be exchanged between SFM and HostBridge at run time via containers. Cool! But...

At design time, how do you specify to SFM that a HostBridge script is the implementation of such a node? How do you specify the name of the HostBridge script? And how do you specify the arguments to be exchanged between the SFM flow and the HostBridge script? The answers to these questions led us directly to the joint development work performed by IBM and HostBridge.

## **SFM Importer Extension Point and API**

Within SFM, “importing” is an important concept. Every Invoke node within an SFM flow is described by an operation file (a .wsdl file), along with one or more message files (a .mxsd file). These files are very complex XML documents. An “importer” is a program that takes an existing artifact (e.g., a COBOL copybook) and creates a corresponding operation and/or message file. IBM provides a number of importers with SFM (e.g., to create a message file from a COBOL copybook).

As part of our joint development work, IBM enhanced SFM to allow third parties (like HostBridge) to provide custom importers. A custom importer is invoked via the Importer Extension point. The term “generic node” was adopted to describe an SFM Invoke node that is created via the Importer Extension point. To complete the picture, IBM defined an API – SFAPI – that allows a custom importer to build the operation and message files associated with the node.

---

<sup>6</sup> The two options not chosen are analyzed more thoroughly in “The Role of HostBridge in the CICS Services Architecture,” available at [www.hostbridge.com/index.php/library/whitepapers](http://www.hostbridge.com/index.php/library/whitepapers).

Using these new SFM features, HostBridge created a custom importer that makes it extremely easy to include a HostBridge script within an SFM flow. Depending on the context, I refer to this component as either the HostBridge SFM Importer or SFM Plugin.<sup>7</sup>

## IBM and HostBridge Partnership Positioning

So what exactly has happened? I'll use IBM's words:

IBM and HostBridge Technology have worked to develop an API for the Service Flow Modeler (SFM) component of Rational Developer for System z (RDz) to enable software vendors to *extend SFM with complementary capability*. The new API allows IBM Business Partners and customers to take full advantage of the benefits of using RDz as a development environment for CICS integration. Using this new API business partners are able to access vendor runtimes from nodes within SFM. Thus, *IBM customers will be able to select from a range of nodes that suit their application needs*, enabling reuse of CICS resources as business services without any changes to existing code. This collaboration demonstrates a *commitment from IBM to work with business partners* to the benefit of mutual customers. Similarly, HostBridge Technology is committed to support IBM tooling for the purpose of CICS integration.

Needless to say, HostBridge Technology is very pleased to have the opportunity to partner with IBM in this manner. Plus, we get to work closely with some really talented IBMers in the CICS TS and RDz product groups.

## High Fidelity Delivered

Perhaps it's also a good time to reiterate why IBM and HostBridge have collaborated in this manner. Again, I'll use IBM's words:

Enterprise customers running CICS have a broad spectrum of integration needs: for applications that range from relatively straightforward to extremely complex. IBM and HostBridge Technology have collaborated to offer an integration solution that deals with the full spectrum of integration needs. Rational Developer for System z's CICS Service Flow Modeler provides immediate access to both COMMAREA programs and *simple, well-behaved terminal oriented transactions* using high-level modeling of service flows. HostBridge extends this scope by adding provision for access to *complex terminal oriented transactions, direct connections to data sources* (such as DB2), and transactions making extensive use of MRO. *Together, RDz's CICS SFM, HostBridge, and CICS TS V3 provide access to all your CICS resources using common tooling without requiring any changes to your existing applications.*

---

<sup>7</sup> HostBridge has two Eclipse-based plugins – our IDE Plugin and our SFM Plugin. They are *not* the same thing. The IDE Plugin, a Process Automation component, is used to author, test, and deploy HostBridge scripts; it can be used in any Eclipse-based environment. The HostBridge SFM Plugin exploits these new features of SFM and therefore requires the RDz environment. The SFM Plugin and Eclipse IDE's in general are discussed in "The Role of HostBridge in the CICS Services Architecture," available at [www.hostbridge.com/index.php/library/whitepapers](http://www.hostbridge.com/index.php/library/whitepapers).

That last sentence says it all: “Together, RDz's CICS SFM, HostBridge, and CICS TS V3 provide access to all your CICS resources using common tooling without requiring any changes to your existing applications.”

Earlier we posed a question: Without modifying your existing applications, what kinds of Web services can you build using CICS TS v3? The answer was simple: A Web service that invokes a single COMMAREA program. Now we can pose another question: Without modifying your existing applications, what kinds of Web services can you build using CICS TS v3, Service Flow Feature, and HostBridge? The answer is far more interesting: ***You can do ANYTHING and EVERYTHING!***

With CICS TS v3, SFF, and HostBridge, you can build business/Web services from any combination of COMMAREA programs, terminal-oriented transactions (BMS or non-BMS), data sources (DB2, VSAM, DLI, etc.), and even non-CICS services.

CICS TS v3 provides the necessary infrastructure to enable robust and scalable services deployment. SFF provides a graphical modeling environment that enables the creation of CICS business services by composing a flow of CICS application interactions. HostBridge provides the ability to incorporate fine-grained terminal-oriented transactions within a service flow, as well as CICS-controlled data sources.

From the perspective of HostBridge and our customers – better yet, from the perspective of most organizations that rely on CICS – terminal-oriented applications are crucial. To do integration with the highest degree of fidelity – fidelity that ensures your terminal-oriented applications and data are working as they should in web integrations, assembled or composite applications, SOAs, or any new context – you need HostBridge Process Automation as one of your integration engines.

HostBridge, CICS TS v3, and SFF are *complementary*. But really, a stronger word than “complementary” is called for. For customers who want to compose high-fidelity web/business services from their existing terminal-oriented applications, we think HostBridge is practically a *requirement*.

## **Coda – 2009-2010**

A great deal of thought and analysis go into any integration project. We are currently preparing a follow-on white paper with that simple fact in mind. It will focus on questions you should ask – and then answer – in the process of making informed decisions about integration solutions based on the CICS applications you use to run your business.

I'll wager that before all is said and done, you'll find that high fidelity will be one of your top goals as well.