

Composing CICS® Services

The Question Is Where?

A HostBridge™ White Paper

By Russ Teubner, CEO



866-965-2427

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: 1/15/2009

Trademarks

HostBridge and the HostBridge logo are trademarked by HostBridge Technology.

Composing CICS Services

The Question Is Where?

Preface

Several months ago I set about writing an “anti-white paper” – more relaxed in style, hopefully less boring than most (including a few I’ve authored), and honest about its marketing message. *The Role of HostBridge™ in the CICS® Services Architecture* remains a work in progress. Our plan is to keep revising until we have the issue of CICS integration *nailed*. With that goal in mind, I encourage you and all readers to share your thoughts and experiences. You can find the latest version and express your opinions at www.hostbridge.com/awpfeedback.

In the paper – and in the excerpt you’re reading now – I don’t try and fool anyone: I have a point to make (What a revelation!). My point is...

That HostBridge, CICS Transaction Server v3, and IBM’s Service Flow Feature, when used together, meet the broadest possible range of CICS application integration requirements – and in the most intelligent manner possible.

My hope is to make this point with technical accuracy, passion, humor, and a bit of irreverence. But there are a number of problems.

First, the point is amazingly difficult to make! Why? Because it relies on a clear understanding of what HostBridge, CICS TS v3, and SFF do individually and how they can work in concert. That involves a lot of puzzle pieces, and we have to look at them closely to figure out how they fit together.

Second, there’s you – the technology, architecture, or business integration professional. If you’re like many of your peers, you feel overwhelmed. Overwhelmed by the myriad business requirements and projects fighting for attention. Overwhelmed by the diversity of systems and applications in your organization. Overwhelmed by the fact that the technology innovation rate always seems to outpace the technology implementation rate (and the larger your organization, the more true this is). Whatever you do, it seems you can’t keep up or catch up. Technology professionals who support mainframe systems are acutely aware of this dynamic. They’ve been expected to do more with less for years.

Because you’re overwhelmed, it’s often too easy to embrace simple, all-encompassing marketing messages from software vendors. Which leads to problem No. 3. Simple, all-encompassing messages rarely tell the whole story. Unfortunately, *you* are the one who has to figure this out!

Hence, my anti-white paper. It’s not only “anti” in terms of style; it’s also “anti” in length¹ (call it the Big Paper) for a very good reason – because the story of CICS

¹ In view of its growth, we contemplate publishing it in book form as soon as it’s complete. Stay tuned.

integration is anything but simple. To really understand the CICS integration story, *you* have to understand a range of interlocking business and technical issues. And *you* have to understand how IBM products and complementary products like HostBridge can be used in concert to address those issues. CICS integration is complex, and its story is nuanced. An accurate understanding of such a story is always more difficult. But the thorough approach pays rich dividends.

Too many customers, including many of ours, have bought into simple messages and then bought a do-it-all CICS integration product before understanding the whole narrative, including the part played by specialty solutions like HostBridge. After their projects fail, they are usually very well informed about what products do or don't do. Unfortunately, they've wasted a lot of time and money on what amounts to an expensive learning experience.

So my goals in writing the Big Paper, and shorter segments like this one, are to provide some valuable education regarding CICS integration and save *you* time, expense, and agony in the bargain.

"Composing CICS Services: The Question Is Where?" – this paper – is an excerpt from the Big Paper, taken mostly from a single chapter, focused, naturally, on the Where question.

So let's peel back the onion and savor the nuance of *Where*...

Where to Begin: Terms and Assumptions

Before we set out, some basic orientation is in order, like perusing a map before exploring strange new worlds. Let's quickly consider *Why* you would compose CICS services and *What* you would compose...

Why Compose CICS Services

The *Why* is relatively easy...

You need to do more with CICS. And to borrow an evolving message from our friends at IBM, you need to do it better and you need to spend less. Practically speaking, you need to make operational processes more efficient, make your workforce more productive, cut your costs, and squeeze more revenue out of your System z mainframe and its software.

To accomplish this bigger *Why*, you probably have smaller, CICS-specific *whys* in mind. You want to extend CICS applications or data to more people in your own organization. You need to "modernize" your CICS apps so they are usable by today's employees, partners, or customers, who are far more comfortable working on the web than facing a terminal-oriented application. You know you'll benefit from automating complex CICS transaction processes into a single simple service. Or your goal is to make your critical CICS applications and business logic a key part of some larger integration architecture.

So much for *Why*. (See, that wasn't so bad.)

What You'll Compose, with Brief Musical Interlude

The *What* is pretty straightforward as well. You can achieve all the *Whys* most effectively, efficiently, and economically by composing a business service – e.g., retrieving a customer's purchase order history from a CICS application – in the form of a web service – a program-to-program interaction enabling completion of the business service over a network. (For those interested in deeper definitions and explanations, check out the Big Paper.)

So *what* is it in CICS that you are composing?

Cue musical interlude... (Get used to the music analogies; integration technology borrows heavily from the concepts and vocabulary of music.)

A piece of music is composed from notes that have a particular pitch and duration. The primary “notes” from which a CICS business service is composed are the existing CICS applications. As musical notes have unique characteristics, so do CICS resources. Broadly speaking, existing CICS application resources fall into two categories, and various words or phrases have been used to describe them. We'll use “visual” and “non-visual” primarily, as well as “fine-grained” and “coarse grained.”

- A “visual” CICS application expresses a presentation interface to an end-user at a terminal. You can also refer to a visual application as a “terminal-oriented” application.
- A “non-visual” CICS application does not interact with an end-user. Instead, it is designed to be invoked by another program. Non-visual programs are also referred to as “COMMAREA” programs because the input/output parameters are passed to/from the program using an area of storage referred to as the communication area, or COMMAREA.

I, along with IBM, often describe the difference between visual and non-visual CICS applications in terms of granularity. Terminal-oriented applications are often described as “fine-grained” because they are designed to deliver a sub-second response to a terminal user by executing a very specific processing step and exchanging small amounts of information in quick succession. By comparison, COMMAREA programs are usually considered more “coarse-grained” (or at least not fine-grained).

To summarize in terms of our music analogy, the two basic types of “notes” in a CICS service composition are terminal-oriented applications and COMMAREA programs. Due to differences in granularity, composing a series of interactions with a terminal-oriented application into a CICS business service is *fundamentally different and more difficult* than composing a CICS business service out of COMMAREA applications.

But there's a significant catch: *terminal-oriented applications are more common.*

All of these factors – different and difficult, yet more common – bear on the question we turn to now – *Where you'll compose your CICS services.* As you'll see in a moment, your choice of composition location will directly impact consumption of processing resources, application response times, and thus the overall usability and performance of your integration solution.

The Question Is Where?

When it comes to composing business or web services, *Where* is a pretty broad concept. There are usually lots of moving parts, and we can analyze many different attributes. I will focus primarily on two: (a) where the web service boundary exists and (b) where the business service composition is executed.

To illustrate my discussion of the most utilized configuration options, I'm going to use a series of diagrams like this one:

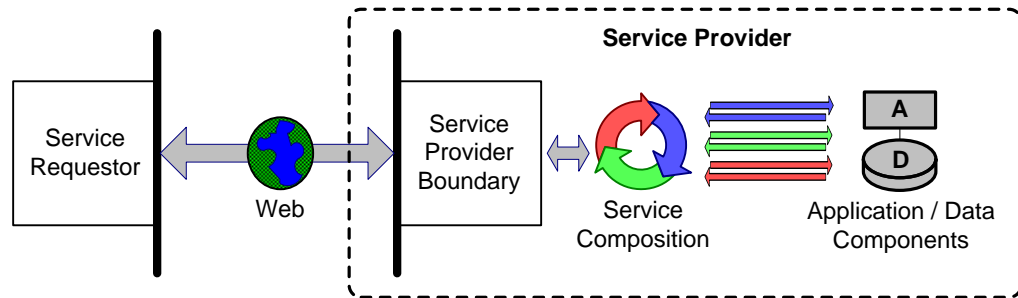


Figure 1. Basic Diagram Showing Service Requestor and Service Provider.

All the diagrams include the same key components:

- Service Requestor – The distributed system or program that expresses a request using standard web service methods (e.g., SOAP).
- Service Provider – The software components, collectively, that respond to the request, also using standard web service methods.

On the service provider side, the action “under the covers” can be broken out in a number of different ways. Because we are discussing the composition of services from existing CICS applications and data, we will highlight three key components:

- Service Provider Boundary – A program that receives and processes the web service request. The boundary function passes control to the program that implements the business service. In the context of our discussion, we will assume that this is a composed service.
- Service Composition – A program that contains the logic describing how the existing CICS applications and data are to be executed or accessed. The service composition *is* the business service.
- Application/Data Components – The existing CICS transactions, programs, and/or data services that will be executed or accessed as part of the service composition.

Since we are focused on CICS integration, we will assume that the application/data components always reside in a CICS TS region on the System z mainframe (duh!). However, the service provider boundary and the service composition can be deployed in various ways, or, more to the point, on different platforms: either on a server or on

the System z mainframe.² So in my diagrams, platforms on the service provider side may be separated as depicted here:³

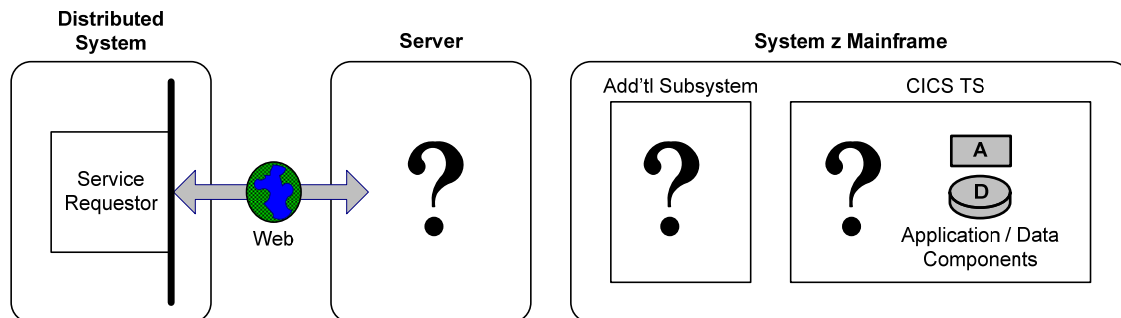


Figure 2. Platforms on the Service Provider Side

The question we want to focus on is this: on the service provider side, *what works best where?*

Relative Position of the Service Boundary and Service Composition

Our objective in this section is to examine where the web service boundary exists and where the business service composition is executed. If you do the math, you can figure out that there are quite a few alternatives. However, there are four common approaches. We will discuss each turn. Note that they all have a common characteristic: the service composition function is always *inside* the service provider boundary. *That's because doing otherwise is a bad idea.* The first three approaches also have something else in common: the web service boundary exists on the same platform where the business service composition is executed. The fourth approach highlights a case in which these two functions are split across platforms.

Back in the old days (before the introduction of the HostBridge scripting engine, if I may say so), customers would typically write a server-based process to automate a series of requests to CICS. However, when a distributed program is used to automate the micro flows required to implement a CICS business service, there are certain ramifications.⁴ If the composition involves many steps, numerous requests/responses must be exchanged between the distributed system and the mainframe. As a result, the response time associated with execution of the business service can become excessive.

² By "System z mainframe" I mean the zOS operating system environment running on a z Series mainframe. By "server" I mean an execution platform that is distinct from the zOS environment running on the z Series mainframe. By this definition, WebSphere running under the Linux operating system environment on a System z mainframe could be a server. I'm not going to worry about these distinctions further because we have enough details to contemplate already.

³ Note also that these diagrams assume that the service requestor exists on a "distributed system." This would usually connote a physically distinct hardware/software platform of some ilk. While this is typical, it does not have to be the case. For example, the "distributed system" could be WebSphere running on zOS or zLinux. In the modern world of IBM System z, zOS, zLinux, and virtualization, what happens where can be rather fluid. Please consider this a diagrammatic convenience rather than a limiting assumption.

⁴ The simple definition of a "micro flow" is: the detailed sequence of requests and responses that are performed within a complete CICS business service.

And if the distributed program and the mainframe communicate using a web services approach, resource consumption on both the server and mainframe can be high.

The following paragraph from IBM's announcement letter regarding Service Flow Feature (SFF) captures this precise thought very well:

In order to implement an optimal method of integration between modern enterprise solutions and an existing enterprise system, the number of interactions between the systems should be minimized. This reduces the volume of request data being exchanged over the transport and the cost of data content processing in each system.

Enough said. It's just a bad idea to automate fine-grained transactions and programs across a web services boundary. *You want this type of orchestration/composition activity to be as close to the underlying components as possible.*⁵

Now that you know why the service composition will always be behind the service boundary in the following diagrams, let's get on with it.

Option 1: Service Provider Boundary and Service Composition Outside the Mainframe

This configuration relies on a middle tier server to express the web service boundary to the outside world and to execute the service composition.

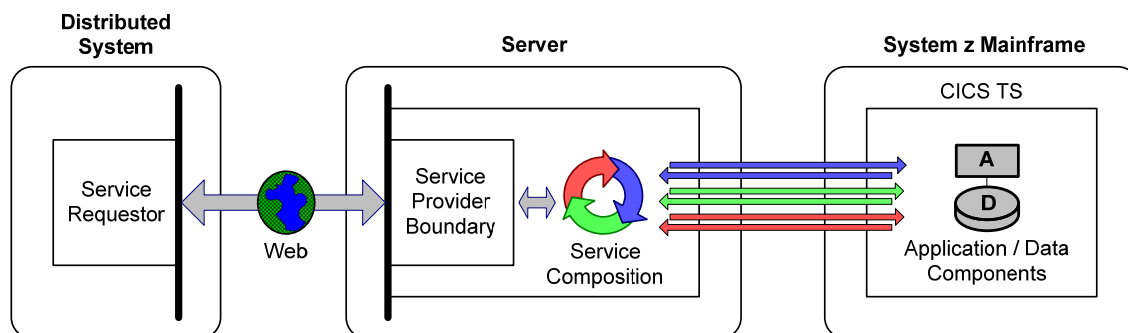


Figure 3. Service Provider Boundary and Service Composition Outside the Mainframe.

This model was popular during the early days of web services, no doubt because web services capabilities were available on server-based platforms before they were on the mainframe. And if you combined a server-based web services implementation with a traditional server-based screen scraping product – presto! – you got something that looked like a web services gateway for the mainframe.

Fortunately, those days are mostly over, although this configuration sometimes still appeals to organizations that, for whatever reason, have issued edicts like “thou shalt

⁵ I also want to clarify something I'm NOT saying. While orchestrating *micro* flows across a web services boundary is a bad idea, orchestrating *macro* flows across a web services boundary is normal and to be expected. In fact, a whole category of orchestration products is growing up, centered on BPEL (Business Process Execution Language), that let you compose high-level web services from lower-level web services.

not put any new software on the mainframe.” They may think such a configuration saves them money, but in my experience, it doesn’t. In fact, it’s often counterproductive.

The most egregious implication of this configuration is that each interaction with a CICS application/data component requires a full travel on the network that connects the server and the mainframe. (Groan.)

As you may have surmised from my comments, this approach has the following attributes, all negative:

- Very high response time due to request/response latency
- Potentially high CPU utilization on the mainframe, depending on how the server and mainframe exchange requests/responses
- Integration with CICS terminal-oriented applications is usually accomplished via screen-scraping (in the Big Paper I wax eloquent about that nastiness)
- Integration with COMMAREA applications is usually limited to the exchange of 32 KB of data per interaction
- Limited or, more likely, no access to CICS managed resources.

Not a pretty picture.

Option 2: Service Provider Boundary and Service Composition Inside the Mainframe, but Outside CICS

This second configuration jettisons the middle tier server but replaces it with an additional, non-CICS mainframe subsystem. However, this mainframe subsystem is responsible for the same tasks as the middle tier server in Option 1 – express the web service boundary to the outside world and execute the service composition.

As you consider this option, keep one thing in mind: software vendors who promote this approach often have a big investment in code that was developed *before* CICS TS was introduced. As a result, some of these products turn a blind eye to the capabilities that are now built into CICS TS.

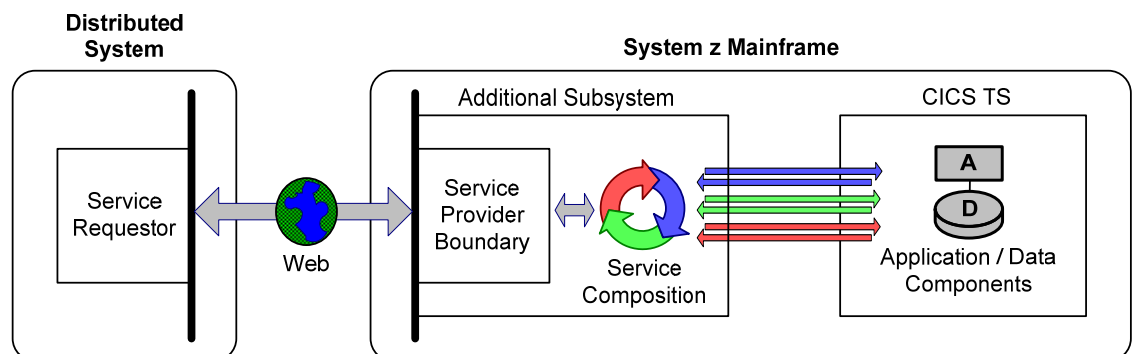


Figure 4. Service Provider Boundary and Service Composition Inside Mainframe, Outside CICS.

This approach has the following attributes:

- Lower response time because the service composition and application/data components are on the mainframe
- CPU consumption on the mainframe will depend on the nature of the service composition and the operational characteristics of the additional System z subsystem.
- Integration with CICS terminal-oriented applications is frequently still accomplished via screen-scraping (can we not rid ourselves of this infernal scourge?!)⁶
- Integration with COMMAREA applications is usually limited to the exchange of 32 KB of data per interaction
- Capabilities built into CICS TS are usually *not* exploited (as far as I'm concerned, you're paying for them, so you might as well use them)
- No integration with the CICS TS operational environment (e.g., security)
- Limited or no access to CICS managed resources

To me, this approach is better, but it still has serious issues that can compromise the overall fidelity of the integration solution. It also begs an important question: if your objective is to expose CICS applications/data as a web service, why would you ignore the capabilities included in CICS TS and implement the web services boundary outside of the CICS TS environment? Why would you relegate CICS TS to be anything other than a first-class web service end point?

Option 3: Service Provider Boundary and Service Composition Inside CICS TS

Our third configuration jettisons the non-CICS mainframe subsystem and exploits the capabilities of CICS TS v3. Now we're talking! CICS TS v3 expresses the web service boundary to the outside world. Furthermore, software running inside of CICS describes and executes the service composition. By collapsing everything within CICS TS, we can achieve optimal performance and maximum sensitivity to the application/data components.

⁶ Some have attempted to exercise the CICS Link Bridge interface across an EXCI boundary. Others implement a Link Bridge "proxy" within CICS and use cross-memory services to pass data between address spaces. In my experience, both techniques yield lower fidelity solutions because important information about the context of the CICS application is either ignored or unavailable.

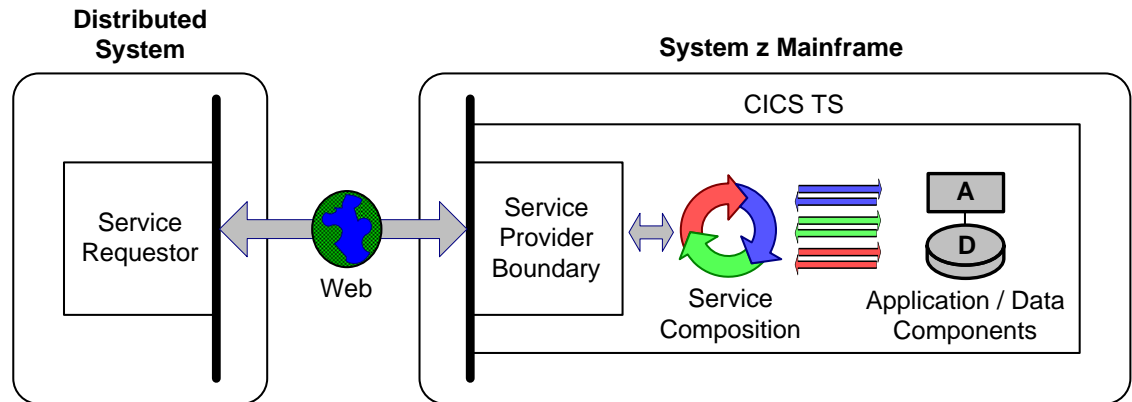


Figure 5. Service Provider Boundary and Service Composition Inside CICS TS.

This approach has the following attributes:

- Lowest possible response time because the service composition and application/data components are combined under the control of a single subsystem on the mainframe
- CPU consumption on the mainframe still depends on the nature of the service composition, but we have eliminated the overhead of an additional subsystem, and we can exploit the efficiencies of the CICS environment.
- Integration with most CICS terminal-oriented applications can be accomplished *without* screen-scraping (finally!)
- Integration with COMMAREA applications can exploit CICS TS capabilities and is *not* limited to the exchange of 32 KB of data per interaction
- Complete exploitation of CICS TS capabilities and full integration within the CICS TS operational environment (e.g., security)
- Full access to CICS managed resources

Perhaps it's no surprise, but to me this is the preferred approach. By a long shot!

At this point you may be tempted to think: "Of course he likes this approach because HostBridge is a service composition tool that runs inside of CICS TS." Actually, such musing raises an interesting question. Which came first? Did we design HostBridge and then declare that we liked this approach? Or did we decide that this approach was the best and then design HostBridge to exploit it? Allow me to shout out the answer loud and clear: **we decided that this approach was best and designed HostBridge to exploit it!**

Please understand, HostBridge began life as a clean sheet design in a CICS Transaction Server world. We built HostBridge from the ground up because it made perfect sense to us to exploit the capabilities that IBM was adding to CICS TS. We hope it makes the most sense to you too.

Option 4: Service Provider Boundary Outside the Mainframe and Service Composition Inside CICS TS

This fourth configuration is a hybrid, and there are some real-world situations in which it makes good sense.

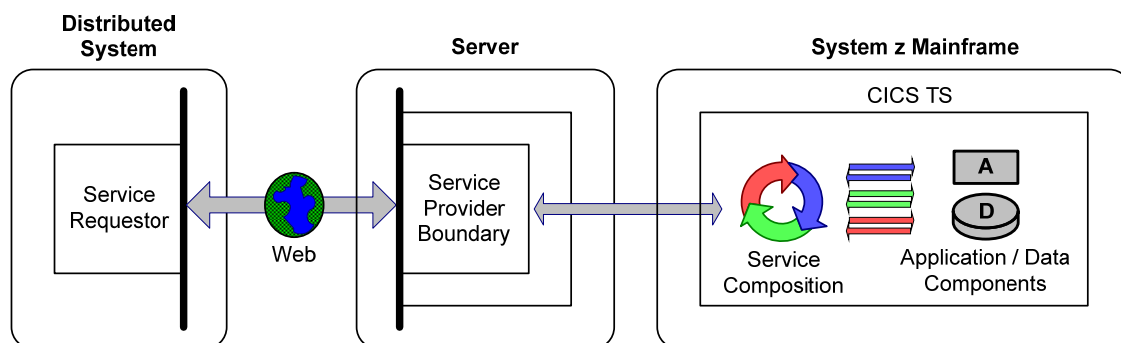


Figure 6. Service Provider Boundary Outside Mainframe, Service Composition Inside CICS TS.

For the reasons already cited, this configuration benefits from executing the service composition on the mainframe and within the CICS TS environment. But this leads us to another important fact:

Where the service composition runs is more important than where the service provider boundary exists.

Imagine that the server is WebSphere and that a Java application invokes a CICS-based service composition (hopefully implemented using HostBridge). It might be very inefficient to use a formal web services boundary between the distributed system and the server, and then implement another formal web services boundary between WebSphere and CICS TS.⁷ Another scenario in which this configuration surfaces is when an Enterprise Service Bus (ESB) or integration appliance (e.g., Cast Iron Systems) is used.

The point here is pretty simple: in many real-world situations it's very convenient (and in some cases more efficient) to implement the service provider boundary outside the zOS environment.

The Point of *Where*

If you've stayed with me through this discussion, commend yourself. Your reward is that you now have a good tool to analyze almost every conceivable approach, solution, or product from the perspective of *Where-to-compose-CICS-services*. While I spared both of us the pain of wading through *all* the possible combinations and permutations, we have covered the most common. I do hope you come away from this discussion with one settled conclusion:

⁷ In high-volume (and "high-sanity") environments, there are only so many formal web services boundaries that one may wish to implement!

To achieve the highest performance, and to ensure the highest fidelity to your CICS application, it is critical that the service composition execute as close to the CICS application context as possible.

There you have it. In forthcoming installments, we'll continue our exploration of the wilds of CICS integration, driving deep into *Formal vs. informal services*, *How to compose services*, *What technologies to employ*, and other informative nuances.

Please remember, you are invited to share your thoughts and observations at www.hostbridge.com/awpfeedback.

Until next time...

Russ