

# Integrating Legacy Applications as Web Services

---

A HostBridge™ White Paper



Toll-free: (866) XML-CICS (965-2427)  
Email: [info@hostbridge.com](mailto:info@hostbridge.com)

## **Copyright Notice**

Copyright © 2003 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: 5/13/2003

## **Trademarks**

HostBridge is trademarked by HostBridge Technology.

# Integrating Legacy Applications as Web Services

Web services promise to lower the costs of integration and help legacy applications retain their value. This white paper introduces you to the basics of web services, how they reduce costs, and how you can use them to integrate mainframe applications with other enterprise applications.

## Web Services in a Nutshell

- Programs, not documents
- Remote programs invoked through a loosely-coupled messaging system
- Application interface based on Internet standards (XML and HTTP)
- Meant for application-to-application communication
- A family of technologies and specifications (SOAP, WSDL, and UDDI)
- Platform-independent
- Broad vendor support

Web services are platform-independent interfaces that allow communication with other applications using standards-based Internet technologies, such as HTTP and XML. They provide an opportunity for organizations to reduce the costs and complexities of application integration inside the firewall and create new possibilities for legacy applications to participate in eBusiness.

Proponents of web services talk about a new era of interoperability. But, haven't we heard all of this before? Applications using standardized interfaces such as CORBA, J2EE, or Microsoft's DCOM were supposed to deliver the same benefits but fell short. However, each of those interfaces has platform-specific and vendor-specific components that keep them from being technology-neutral. Therefore, companies that adopted CORBA interfaces were not immediately compatible with partner networks that used either DCOM or J2EE. However, web services are standards-based, platform-independent, and the course web services take is controlled by independent standards bodies, not individual corporations pushing their own technologies. Web services simplify the complexity of integration by reducing the number of APIs to one and the number of data formats to one.

Web services sound promising for new applications that have support for Internet technologies built-in, but what about legacy systems? How can they benefit from web services integration and what does it take to integrate legacy applications as web services? First, let us discuss web services in general, and then we will delve into their relevance to legacy applications.

## Standards-based and Platform-independent

While applications have been able to communicate using Internet technologies for years, only recently have standards evolved to allow any distributed applications to talk to each other. Web services depend upon three key open standards to allow them to communicate regardless of the hardware they run on, operating system they run under, or programming language used to produce them.

**Table 1. Standards for web services**

Standard	Description
SOAP	The Simple Object Access Protocol specifies a format for messages passed between web services.

WSDL	The Web Services Description Language describes web services so that other web services know how to access them, what to send as input, and what to expect as output.
UDDI	The Universal Description, Discovery, and Integration standard is a searchable registry that allows web services to publish WSDL documents so other web services can find them by name or category. UDDI also provides specification information concerning the input data formats, security models, protocols they use, and response data formats.

“With a new web services coating, *existing systems, from CICS to client/server, will participate in new business scenarios.* And with that participation codified in standard interfaces, the apps themselves can be modified or swapped out without bringing operations to a halt.”

Ted Schadler  
Forrester Research, Inc.

Building on the foundation provided by SOAP, WSDL, and UDDI, there are emerging standards for handling complex business process workflow, authentication and message integrity, transactions that span multiple enterprises, and long lasting transactions to ensure that the outcomes of the transactions are reliable. Underlying all of this is the use XML as the message format and a standard protocol such as HTTP as the transport method.

Standards-based components diminish the costs and skills required to integrate applications within an organization or between partners. Because there is widespread vendor support for web services, interoperability among vendor solutions will improve. However, the greatest benefits of standards-based web services for developers and administrators are reduced complexity and increased flexibility of integration architectures.

## Reducing Complexity and Increasing eBusiness Flexibility

The number of application programming interfaces (APIs) and data formats that developers must learn and support affects the complexity and flexibility of integration. The more interfaces and formats the greater the costs to implement and maintain an integration framework. SOAP provides a common interface to applications and XML provides a common data format — together, they simplify integration and lower integration costs.

### Fewer Application Interfaces

Traditionally, developers create interfaces for each application in an integration project to allow inter-application communication. Because the APIs for each application are different, companies integrating large numbers of applications develop complex integration frameworks that require specialized development skills and lead to increased integration costs. Increased complexity also makes integration architectures less flexible, so that changes in the network and business processes take longer and become more difficult.

“G3,500 firms have valuable business logic embedded in a hodgepodge of legacy systems - from CICS to customized provisioning apps. *Adding Web services interfaces to the mix makes those elements available for widespread reuse.*”

Laura Koetzle  
Forrester Research, Inc.

Reducing the number of APIs reduces complexity and increases flexibility. Web services based on Internet technologies are not specific to any platform and provide a single API for any application to use. They are loosely-coupled and can be invoked directly as traditional APIs or requests can be sent to a queuing system where transactions can occur at specified dates or times.<sup>1</sup> Web services also provide greater flexibility when designing integration architectures. Because web services use a common interface, changes to the network or business processes do not affect the ability of individual applications to communicate with each other.

<sup>1</sup> Integration using traditional APIs is tightly-coupled in that each application accepts input and output using unique data formats. The applications usually assume that they are invoked synchronously and they expect to receive an immediate response.

---

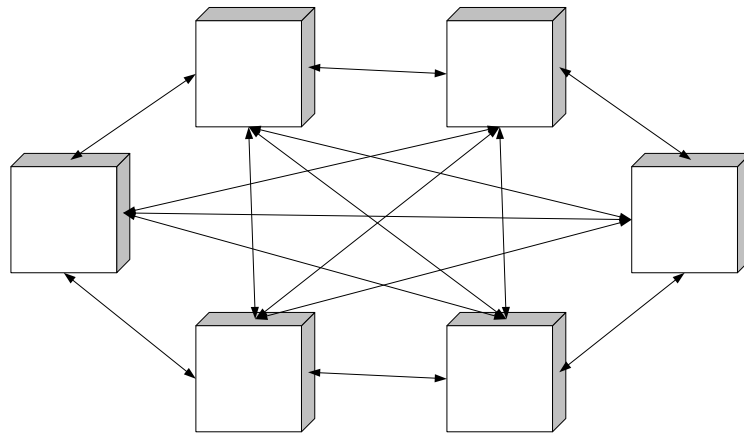
## Common Data Formats

Another source of complexity is the proliferation of data formats. In point-to-point integration, each application must transform its own data structures to those of every other application with which they share information. Reducing the number of data formats reduces the number of data transformations that take place as information passes through the integration framework.

---

## Traditional Integration vs. Web Services Integration

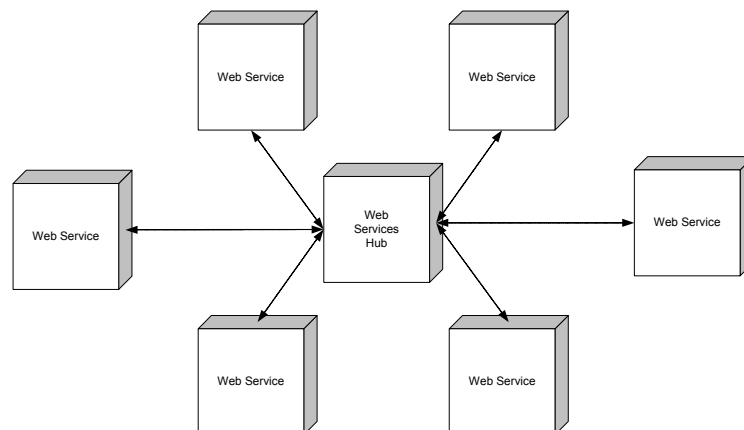
Figures 1 and 2 show how reducing the number of APIs and using a common data format based on XML can reduce the complexity in an integration framework. In traditional integration, an application communicates with another application through the target application's API. Data passing between applications undergoes a transformation from one format to another. Application programmers must be aware of each API and each data format in the framework, which creates a huge learning curve. As you add more applications to the framework, the complexity increases and the burden on developers grow.



---

**Figure 1. Traditional integration frameworks (6 applications, 6 APIs, and 6 data formats)**

Web services integration still requires that one application call another using an API and data transformation still occurs. However, each application uses SOAP as a common API and XML as a common data format. So, even as the number of applications in the integration framework increases the burden on the developers remains the same.



---

**Figure 2. Web services integration frameworks (6 web services, 1 API, and 1 data format)**

## Legacy Applications and Web Services Models

Mainframes are still the most reliable and scalable platforms for handling large amounts of data and large numbers of transactions, so keeping your legacy applications on the mainframe is a good technical decision. And, integrating your applications is almost always cheaper than rewriting them. Now, by enabling your legacy applications as web services you can deliver integration projects faster and cheaper because your legacy application groups and Internet groups use their existing knowledge to integrate the applications.

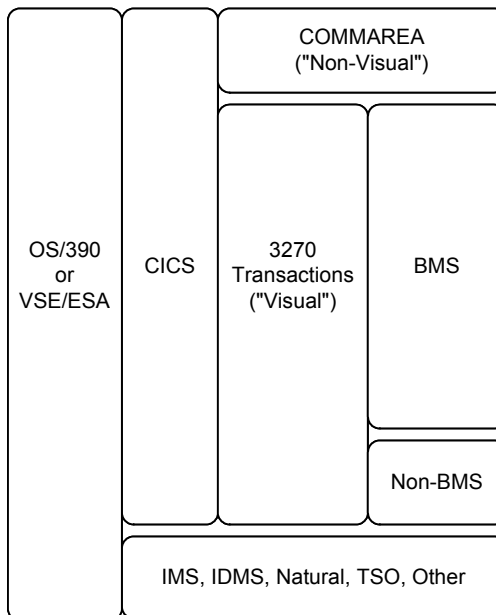
There are two basic models for integrating legacy applications as web services. The differences between these models depend upon where the web services exist, how they operate under the covers, and the types of applications you want to integrate. In this article we will refer to these models as Adapters and Gateways.

- *Adapters* run on the mainframe and can use interfaces that permit tight integration with the target application.
- *Gateways* run off the mainframe on middle tier servers and often use traditional methods, such as screen-scraping.

---

### Types of Legacy Applications

The choice between using adapters or gateways often depends upon the types of applications you need to integrate. Figure 3 shows a high-level taxonomy of mainframe applications.



**Figure 3. Legacy mainframe application taxonomy**

CICS represents the broadest category of OS/390 applications. CICS transactions fall into two sub-categories: “visual” and “non-visual.” A “visual” transaction is one that expresses a presentation interface to an end-user at a terminal. You could also refer to a “visual” transaction as a “terminal-oriented” transaction. In contrast, “non-visual” transactions do **not** interact with an end-user. Instead, another program invokes these transactions. (This type of transaction is also referred to as “COMMAREA transaction” because the input/output parameters are passed to/from the transaction using an area of storage referred to as the “communication area,” or COMMAREA.)

In the case of visual transactions, CICS application developers have always had a number of choices in how to design their transaction to interact with an end-user at a terminal. The majority of applications use a component of CICS called Basic Mapping Support (BMS). BMS essentially handles the presentation logic of the transaction and relieves the application developer from having to encode and decode 3270 terminal data streams. The remaining groups of applications that do not use BMS either include code to process 3270 data streams, or rely upon a non-IBM solution to handle presentation logic.

IBM has traditionally provided facilities that allow you to use gateways to access CICS. However, with the release of CICS Transaction Server, IBM began providing facilities that allow the use of adapters to access legacy applications, so you can choose between adapter and gateway models based on your needs. The recent availability of adapters that support the full range of CICS applications allows remote applications to directly invoke almost any CICS application as a web service.

There are solutions on the market that support COMMAREA, BMS, and non-BMS transactions all through a single adapter. Because most shops have a mix of application types, companies should seek out this kind of adapter to avoid multiple software licenses and additional training on how to integrate the different application types within your organization.

Companies wanting to integrate IMS DC applications will most likely need to use gateway methods. While IMS DC applications usually separate application data and presentation logic, IMS does not provide an interface that allows an adapter to capture application data before it is sent and converted to XML/SOAP. Thus, the only other viable solution is to reengineer the application or use a gateway to access the application from a middle tier.

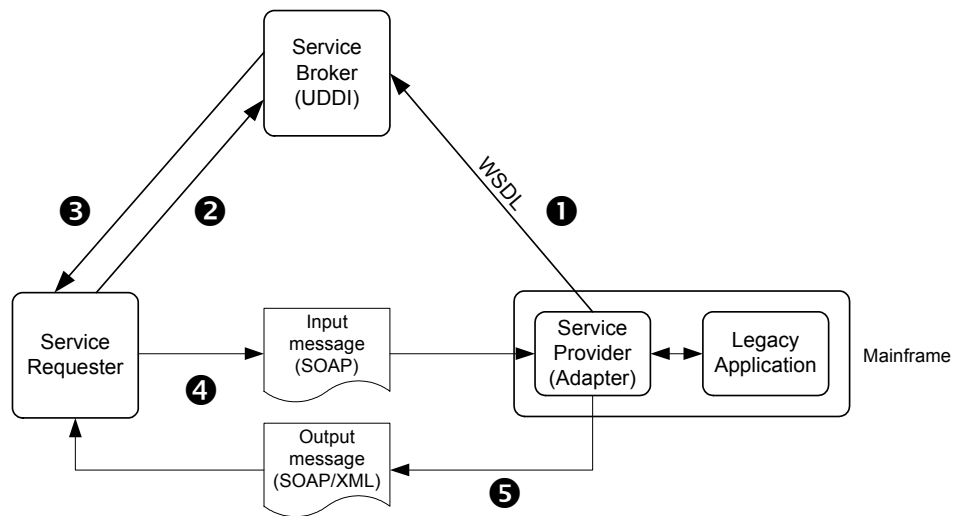
In the adapter and gateway examples below, we focus on using web services with visual (terminal-oriented) applications because they are typically more difficult to integrate than non-visual applications.

---

### **Adapter Model for Web Services**

Adapters allow you to transform your legacy applications into web services without requiring the use of additional hardware, without changes to the legacy application, and without falling back upon brittle techniques like screen scraping as the access method. Compared to gateways, adapters yield better performance by running on the host and more reliable operation due to the elimination of the many layers data must pass through due to screen-scraping. Adapters also provide enhanced access to application information such as state and error codes. This information is lost when data is sent to a terminal emulation technology such as a 3270 emulation client or FEPI.

Figure 4 below shows the basic model for accessing legacy applications as web services directly through adapter technologies.



**Figure 4. Adapter model**

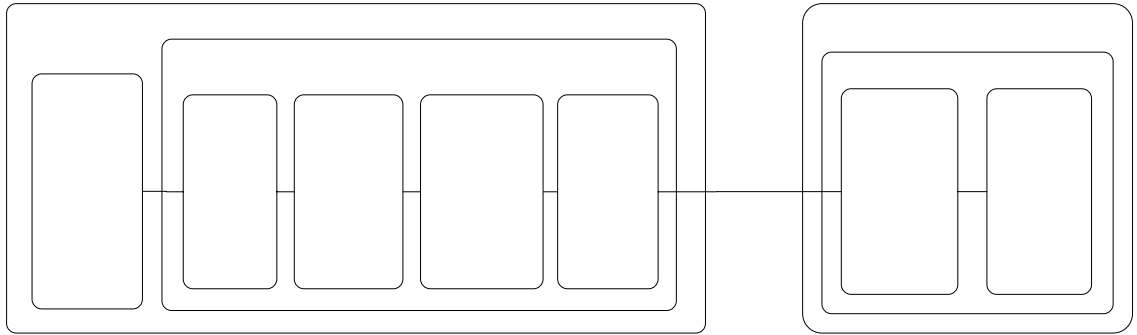
The diagram in Figure 4 illustrates a web services architecture: a Provider that supplies the web service, a Requester that uses a web service, and a Broker that finds Providers for Requesters. In this model, the legacy application is the Provider. The following steps represent how to find and use a web service adapter. (Steps 1-3 are optional.)

1. The Provider uploads a WSDL specification to publish its web service with a Broker.
2. The Requester (usually a Java or .NET application) queries the Broker for a web service by name or category.
3. The Broker selects a Provider and returns the Provider information to the Requester.
4. The Requester uses the information from the Broker to format and send a SOAP message to the HostBridge.
5. The Provider returns a SOAP/XML response to the Requester with the legacy application data enclosed.

### Gateway Model for Web Services

Unlike adapters, gateways typically run on a physical or logical middle tier. Where the gateway runs is important because there are so few options for accessing the host from the middle-tier servers, which means gateways usually involve some form of screen scraping. The integration is tightly-coupled between the gateway and a specific application, so changes to the application will break the integration.

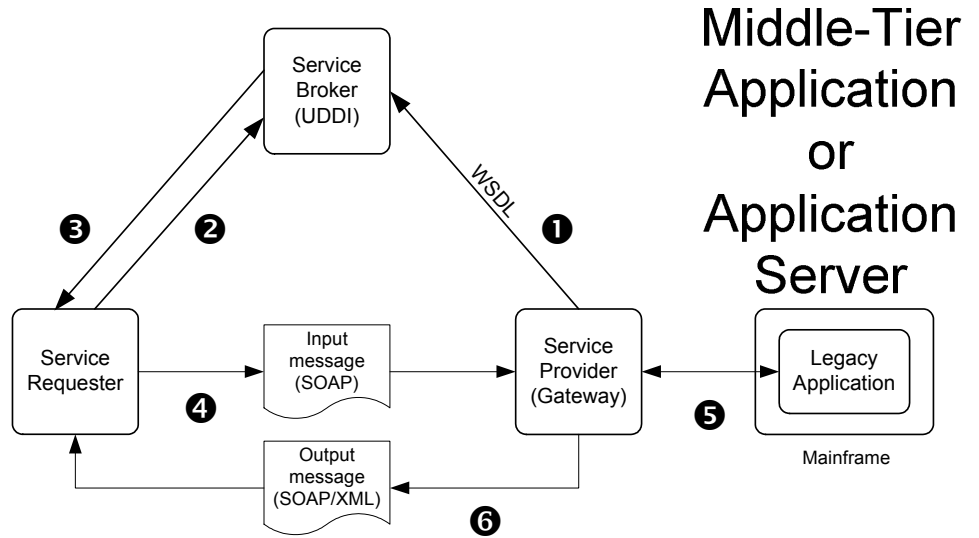
When gateways communicate with terminal-oriented legacy applications they open a terminal session with the legacy application, send a request to the application, receive the terminal datastream, use HLLAPI to capture the screen data, process the screen data, convert the contents to XML, and ship the XML document to the requester. (A variation of the gateway model is to use FEPI on the mainframe instead of a middle-tier terminal emulation client. This simply moves the middle tier onto the mainframe.) The most common components of the gateway model appear in the Figure 5, below.



**Figure 5. Typical middle-tier gateway architecture**

Gateways allow you to get an application into the web services mix, but screen scraping creates performance bottlenecks and multiple points of failure between the legacy application and the web service. For this reason, gateways are probably best for short-term projects, either as a transition to using adapters or as a stop-gap measure during application reengineering or platform migration.

The figure below shows the basic model for accessing legacy applications indirectly using a gateway technology such as a screen scraper.



**Figure 6. Gateway model**

Again, the diagram in Figure 6 illustrates a web services architecture: a Provider that supplies the web service, a Requester that uses a web service, and a Broker that finds Providers for Requesters. The legacy application is not itself a web service, but is accessed by an off-host Provider. The following steps represent how to find and use a web service gateway. (Steps 1-3 are optional.)

1. The Provider uploads a WSDL specification to publish its web service with a Broker.
2. The Requester (usually a Java or .NET application) queries the Broker for a web service by name or category.
3. The Broker selects a Provider and returns the Provider information to the Requester.

4. The Requester uses the information from the Broker to format and send a SOAP message to the Provider.
5. The Provider starts an emulation session and conducts a series of transactions with the legacy application to collect the requested data.
6. The Provider converts the legacy datastream into an appropriate returns a SOAP/XML response and returns the response to the Requester with the legacy application data enclosed.

---

## Dynamic Adapters versus Code Generators

As we have seen, adapters provide several advantages over gateways when it comes to web services integration. How those adapters operate can be just as important. Products that implement the adapter model for web services usually fall into two camps: those that use code generation and those that are dynamic. Code generators have been around for some time and require developers to adhere to a fairly structured process. In the case of CICS applications, something like the following takes place:

1. Download program source code, copybooks, or screen maps to a workstation running proprietary tool.
2. Using a proprietary tool to generate web services “wrapper” code.
3. Upload generated code to mainframe
4. Compile, install and test generated code.
5. Repeat process for each program and whenever the programs change.

While generated adapters can jumpstart the initial development process, they create several maintenance problems. First, the integration developer is generating code off the host and uploading the code to the mainframe. In most cases, this developer will know a lot about web services and XML, but will know little about the mainframe. Thus, any changes to the mainframe application require coordination between the web developer, the application developer, and the CICS administrator. Second, generated adapters create “net new code” that must be managed. Changes to either the legacy application or to the web service will require repetition of the code generation process to keep the generated code in sync with the integrated application. Without proper management, you are likely to drown in a sea of generated code.

These issues led to the development of dynamic adapters. Dynamic adapters operate with little or no configuration and changes to legacy applications are automatically incorporated into the SOAP/XML output. In many cases there is no configuration required, while some cases may require a single step process to specify web service information for each application. As a result, there is no generated code and there is a clear division of labor: the CICS administrator installs the adapter and the web developer simply invokes the adapter as a web service.

## Conclusion

As we have seen, one problem with traditional integration techniques is the proliferation of point-to-point communication and data conversions that must change as new applications are integrated or data formats change. The problem gets complicated quickly when you add business partners, subsidiaries, mergers or acquisitions into the integration mix. Web services reduce the costs and complexities of integrating applications. By using standards-based technologies and widely available skill sets, web services allow companies to develop flexible integration solutions that can evolve as needs change.

The subsystem under which your legacy application runs determines your top-level integration choices. IMS provides limited facilities to create integration adapters, so IMS shops are often relegated to using screen scrapers and gateways. CICS Transaction Server includes facilities that allow third-party vendors to create adapters that can immediately enable legacy applications as web services. These facilities also provide additional benefits over gateways, such as improved performance and increased stability versus their screen scraping counterparts. By using the same industry standard technologies as web services, some adapters make it possible for applications to transparently invoke CICS transactions within a web services architecture and receive the resulting data as well-formed XML. For organizations that want to retain the value of their CICS applications, the combination of XML-enabling adapters and web services offers a practical and powerful integration solution.

Web services are not a trend, but an industry-wide movement that can provide a long-term solution for companies who want to integrate legacy applications and data with new eBusiness processes. In the end, companies need to assess the value of the data contained in their legacy applications. Most companies have already determined that such data is highly valuable and they are looking for ways to preserve their investments. Given that recent surveys show the top strategic priorities of CIOs and CTOs is integrating systems and processes, the use of web services will grow rapidly and you need to be ready for them.