

# SOAP and REST

---

## Choosing formal and informal Web services for CICS integration



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

## **Copyright Notice**

Copyright © 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, 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: 3/2/2009

## **Trademarks**

HostBridge is trademarked by HostBridge Technology.

# SOAP and REST: Choosing formal and informal Web services for CICS integration

In 2007, Gartner predicted that Web oriented architectures defined by REST and XML over HTTP would replace 90% of Web SOAP-only implementations by 2012<sup>1</sup>. You can see the move away from SOAP to RESTful architectures in Google APIs, Salesforce.com, and within large organizations like Wal-Mart. SOAP and REST will continue to play roles, but what will those roles be and when should you choose one over the other?

---

When HostBridge Technology launched itself into the CICS integration scene in 2001, our mantra was simple: “XML for CICS.” We made CICS transactions available as XML for use in application access and integration. It was, to us, an obvious marriage between the web and the mainframe made possible by advances in CICS Transaction Server. The next obvious step was to make CICS resources available as Web services. Continuing with the theme that made up popular, our Web services were plain old XML (POX<sup>2</sup>) over HTTP.

The years passed and people began to *talk* about Simple Object Access Protocol (SOAP). The dirty secret about SOAP is that it is anything but simple. There is a formal structure that promised to make it possible for any business to deploy, but differences between how application servers handled the requests, differences between how .NET and Java environments handled the documents, and lack of knowledge about best practices for its use made SOAP a huge headache. But, because it was the buzzword in every press and analyst conversation about integration, people began asking for it.

Eventually, we added SOAP support to HostBridge prior to the availability of native SOAP support within CICS TS. A few years later, a clear trend began to emerge: almost none of our customers were using SOAP; not even the customers who said they would like to see it added.

Like HostBridge, CICS Transaction Server has evolved with customer needs. IBM provided access to CICS via HTML, then XML, then SOAP, and now Representational State Transfer (REST). We will talk about these technologies in this white paper, but the net of it is that REST is a return to the basic architectural assumptions we made in HostBridge 1.0: XML over HTTP is flexible and easy to use, so why make it harder than you need to? We will discuss the differences between formal Web services (SOAP), informal Web services (RESTful), and when to use them to best effect.

---

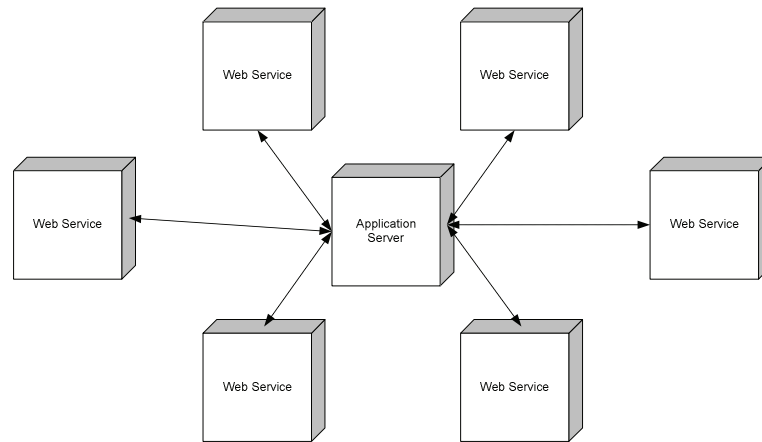
<sup>1</sup> “Applying WS-\* Based Web Services and WOA Standards to Enterprise Application-to-Application Interoperability Challenges,” Gartner 2007

<sup>2</sup> Yes, Plain Old XML, or POX, is a real industry term and a real acronym.

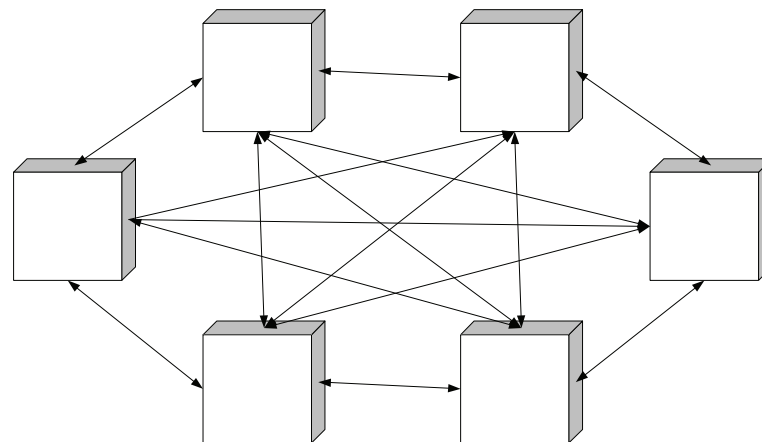
## Formal and Informal Web Services

There are a number of ways to implement program-to-program interactions and still consider them as Web services. For example, the Web service can be implemented as a “Web API that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services.” Alternatively, the Web service can be implemented according to a more formal approach: “clients and servers that communicate using XML messages that follow the SOAP standard.” This distinction is very important. However, we need some words to describe the difference. In this white paper we use the words “formal” and “informal” to denote two basic approaches to implementation of Web services, where the definitions for these approaches are:

- **Formal Web service** – an approach whereby clients and servers communicate using XML messages that follow the SOAP standard; WSDL is used to describe the operations supported by the server. As mentioned above, the use of formal Web services and SOAP lend themselves to a distributed computing model as shown below.



- **Informal Web service** – any approach to implementing a Web service that does not abide by the SOAP and/or WSDL specifications. In the case of REST, HTTP and XML are used but the formality of SOAP and WSDL are dispensed with. As mentioned above, the use of informal Web services and REST lend themselves to point-to-point communications as shown below.



## What is SOAP?

SOAP, or Simple Object Access Protocol, is a transport, platform, and language-agnostic designed as an alternative to interoperability technologies like CORBA and DCOM. A crowd of standards govern the use of SOAP, so much so that Gartner has begun calling Web services based on SOAP “WS-<sup>3</sup>”. The proliferation of standards makes SOAP increasingly complex to understand and use. Minus the standards, a basic SOAP message looks like the following:

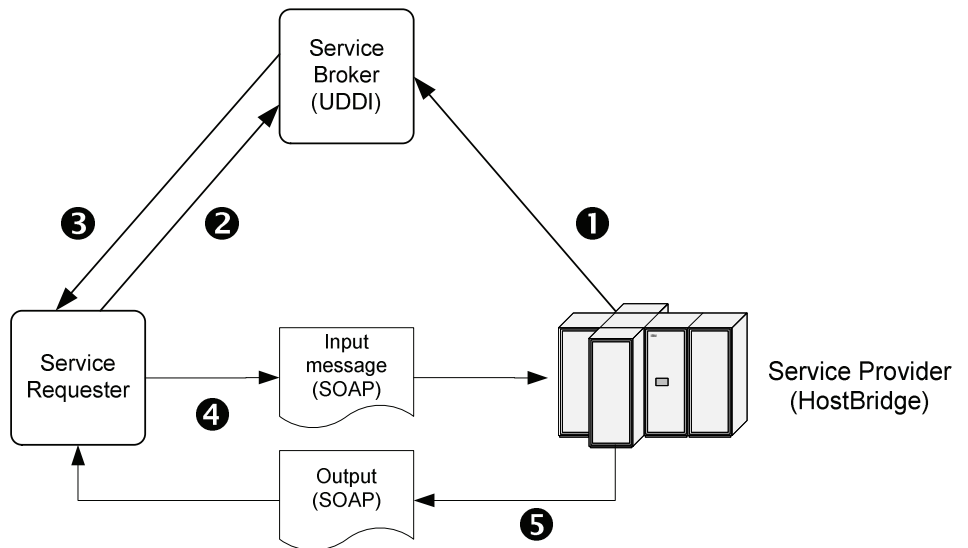
```
<soap-env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <soap-env:Body>
    <hb:payload />
  </soap-env:Body>
</soap-env:Envelope>
```

**Figure 1. Basic SOAP message format**

The SOAP message is an XML document, but the format is different. The basic XML document is “wrapped” in a pair of elements: <SOAP-ENV:Envelope> and <SOAP-ENV:Body>. According to the SOAP specification, SOAP messages wrap their contents in an envelope and use the body to identify the beginning and ending of the payload. The addition of the “hb:” prefix to the standard XML elements illustrates the use of namespaces. Namespaces provide a tool for developers to use to identify which elements are HostBridge input elements.

### SOAP Model

Below is the basic model for accessing CICS applications as web services through HostBridge.



**Figure 2. HostBridge and the SOAP web services model**

The diagram shows three web services: a Provider (HostBridge) that provides the web service, a Requester that uses a web service, and a Broker that finds

<sup>3</sup> Why does Gartner call SOAP services WS-\*? Here is a list of the specifications surrounding the use of SOAP: WS-coordination, WS-atomic transaction, WS-business activity, WS-policy, WS-policy assertions, WS-policy attachment, WS-discovery, WS-metadata exchange, WS-management, WS-I basic profile, WS-addressing, WS-enumeration, WS-events, WS-transfer, WS-security(SOAP message security), WS-security(Username token profile), WS-security (X.509 token profile), WS-secure conversation, WS-security policy, WS-trust, WS-federation, WS-federation active requestor profile, WS-federation passive requestor profile, WS-security (Kerberos binding), WS-reliable messaging, UDDI, WSDL, XML, Namespaces in XML, XML information set, SOAP, SOAP-over-UDP, and more...

Providers for Requesters. The following steps are required to find and use a web service. (Subsequent requests do not require steps 1-3.)

1. HostBridge uploads a WSDL specification to publish its web service with a Broker.
2. The Requester 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. HostBridge returns an XML document to the Requester with the CICS data enclosed

---

## SOAP Example

In this example, HTTP will be the transport protocol. The SOAP request-response sequence represents a call to CICS to call a Web service that retrieves a stock quote for IBM.

```
GET /StockPrice HTTP/1.1
Host: hostbridge.com
Content-Type: application/soap+xml
Content-Length: nnn

<?xml version="1.0"?>
<soap-env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:hb="http://www.hostbridge.com/stock-service">
  <soap-env:Body>
    <hb:getQuote>
      <hb:companyName>IBM</s:companyName>
    </hb:getQuote>
  </soap-env:Body>
</soap-env:Envelope>
```

**Figure 3. SOAP request for stock quote**

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml
Content-Length: nnn

<?xml version="1.0"?>
<soap-env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:hb="http://www.hostbridge.com/stock-service">
  <soap-env:Body>
    <hb:getQuoteResponse>
      <hb:price>45.25</s:price>
    </hb:getQuoteResponse>
  </soap-env:Body>
</soap-env:Envelope>
```

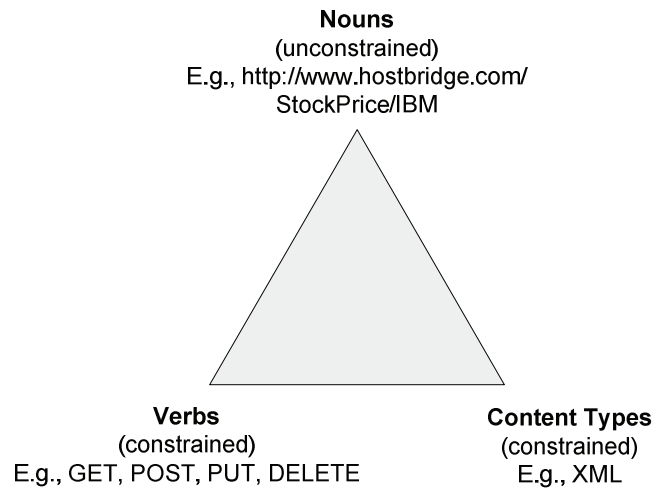
**Figure 4. SOAP response with stock quote**

Notice that both the client and server must understand and adhere to the SOAP messaging format.

## What is REST?

RESTful Web services are a lightweight alternative to the heavy, SOAP-based standards. In RESTful web services, the emphasis is on simple point-to-point communication over HTTP using XML. The origin of the term "REST" comes from the thesis from Roy Fielding describing the concept of *Representative State Transfer*. REST is an architectural style where distributed systems are built on a shared model and have agreement between nouns (resource names as URIs), verbs (HTTP methods used), and content types (usually XML or

JSON). The REST model for the relationship is depicted as a triangle shown below.



**Figure 5. REST relationship model**

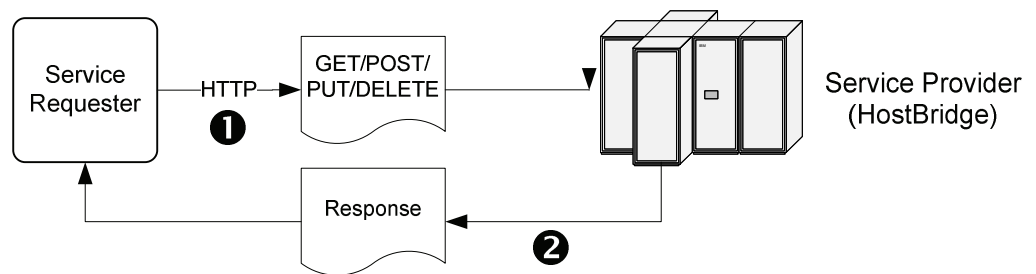
Verbs are operations applied to resources. The REST Web services model assumes universal verbs applied to all nouns based on methods in HTTP 1.1 which correspond to the CRUD (create, read, update, delete) database operations.

HTTP Method	CRUD Operation	Description
GET	Read	Fetches a resource
PUT	Create	Transfers a resource to the server and overwrites existing resources
POST	Update	Adds to an existing resource on the server
DELETE	Delete	Discards resources; names cannot be used

Nouns are URIs to resources. In our examples, a service to request a stock price for the company 'IBM', for example, would be handled using an HTTP GET to `http://www.hostbridge.com/StockPrice/IBM`.

## REST Model

Below is the basic model for RESTful Web services.



**Figure 6. HostBridge and the RESTful web services model**

The diagram shows two web services: a Provider (HostBridge) that provides the web service and a Requester that uses a web service. In the RESTful model, the services are self-describing, so there is no need for WSDL and a UDDI server to act as a Broker. The following steps are required to find and use a web service. (Subsequent requests do not require steps 1-3.)

1. The Requester queries the Provider over HTTP using a URI. The request uses one of the HTTP methods to determine whether the request is intended to create, read, update, or delete data.
2. HostBridge returns an XML document to the Requester with the CICS data enclosed.

As you can see, the simplified architecture and process not only makes RESTful Web services easier to use and deploy, but without all the middleware calls, the RESTful services are much more scalable.

---

## REST Example

The stock quote service above rewritten as a RESTful Web service might adopt the following request-response model.

```
GET /StockPrice/IBM HTTP/1.1
Host: hostbridge.com
Accept: text/xml
Accept-Charset: utf-8
```

**Figure 7. REST request for stock quote using HTTP GET**

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<hb:quote xmlns:hb="http://hostbridge.com/stock-service">
  <hb:companyName>IBM</s:companyName>
  <hb:price>45.25</s:price>
</hb:quote>
```

**Figure 8. REST response with stock quote**

The RESTful version is simpler and more concise than the RPC-style SOAP version. RESTful Web services are intentionally closer in design and style to the Web itself.

## When to use formal and informal Web services

Before we make some recommendations on when to use SOAP and REST for CICS integration, keep in mind these points for comparison.

Service	Pros	Cons
SOAP	Language, platform, and transport agnostic Designed to handle distributed computing environments Is the prevailing standard for web services, and hence has better support from other standards (WSDL, WS-*) and tooling from vendors Extensibility	Conceptually more difficult, more "heavy-weight" than REST Harder to develop, requires tools

RESTful	Language and platform agnostic Simpler to develop than SOAP Small learning curve, less reliance on tools Concise, no need for additional messaging layer	Assumes a point-to-point communication model--not usable for distributed computing environment where message may go through one or more intermediaries Lack of standards support for security, policy, reliable messaging, etc., so services that have more sophisticated requirements are harder to develop ("roll your own") Tied to the HTTP transport model
---------	---	---

When someone uses the phrase “Web services” or “Services Oriented Architecture” (SOA) they often assume the formal approach (use of SOAP, WSDL, etc.). But the practical reality is that a formal approach to Web services requires more human planning, as well as machine processing, on both sides of the connection. Thus, deploying a formal Web services approach may be more appropriate when:

- You don’t own/control both the requesting and responding system (e.g., inter-company connections)
- The client and server are in different security zones (e.g., outside and inside the firewall, respectively)
- Volume is low to medium
- The business relationship between the two parties is “loosely coupled.”
- An industry standard exists for the use of SOAP that multiple organizations agree to use in private communities.

Alternatively, deploying an informal Web services approach may be appropriate when:

- You do own/control both the requesting and responding system (e.g., intra-company connections)
- The client and server are in the same security zone
- Volume is medium to high.

The distinction between informal and formal Web services is important. With the release of Transaction Server 4.1, support for Web services through REST is built into CICS alongside the existing SOAP pipeline. This means CICS integration teams are able to choose the right style of Web services for their needs, and an understanding of formal and informal services can help guide the decision-making process.

## HostBridge, REST, and CICS Transaction Server

IBM added RESTful Web services to CICS TS with the release of SupportPac CA1S and includes this support as part of CICS Transaction Server 4.1 REST support comes with the addition of PHP running under the Java Virtual Machine. As IBM describes it: “Facilities provided by this SupportPac allow RESTful services to be created easily that exploit existing CICS COMMAREA applications. As well as existing application components, this SupportPac allows PHP scripts to access DB2 tables via PHP Data Objects (PDO) using the CICS Java Database Connectivity (JDBC) driver.” In the same way that the SOAP Pipeline provides access to CICS resources using formal Web services, PHP running under Java provides access to CICS resources using REST.

This creates yet another avenue to CICS. However, it still does nothing to make your 3270 BMS and non-BMS transactions available as Web services or XML. *You still need HostBridge to express application logic as business services, though now you have another way to call the services.*

## Conclusion

HostBridge customers have always been at the leading edge of implementing CICS-based Web services. While some use HostBridge as part of a formal Web services approach, the majority use HostBridge to implement informal Web services. Specifically, the most common methodology is using HTTP GET or POST requests, whereby the input parameters are specified using a URL query string or POST data, and the output data is an XML document (using a customer-specific or service-specific XML vocabulary). Why do customers do this? Because it is simple and efficient.

HostBridge makes CICS resources callable as RESTful Web services or SOAP objects. Regardless of the model you choose, HostBridge allows you to include CICS in your SOAs, mashups, or any other Web service implementations.

### Contact Information

For more information on HostBridge or to inquire about our free 30-day trial, please contact us using the information below.

Toll-free:  
1.866.XML.CICS (965.2427)

International:  
1.405.533.2900

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





100 E Seventh Ave, Stillwater, OK 74074  
Ph. 405.533.2900  
[www.hostbridge.com](http://www.hostbridge.com)