

<http://myarch.com/reliability-of-soap-over-http-web-services>

Reliability of SOAP over HTTP Web Services

Posted on 10/29/2006 , Alexander Ananiev, [7 Comments \(Add \)](#)

HTTP or HTTPS can be viewed as the current de-facto standard transport binding for Web services. It is frequently said, however, that HTTP is inherently unreliable and not appropriate in situations where guaranteed delivery and other Quality of Service (QoS) characteristics are required. To deal with this issue, many Web services products, commercial and open-source, offer a non-standard (meaning that it is not part of a WS-I profile) transport binding using JMS or other messaging APIs. Alternatively, [WS-ReliableMessaging \(WS-RM\)](#) specification defines a reliable messaging protocol for Web services independently of the transport binding. Currently, WS-RM is supported by several Web services products.

It is tempting to standardize on one of these alternatives for an enterprise SOA implementation in order to meet QoS requirements. However, we need to realize that both of these options add complexity to Web services implementation and greatly impair interoperability. JMS binding requires that all Web service consumers must support a particular messaging provider. Additionally, appropriate messaging resources (queues, connection factories) must be configured on the server and, sometimes, on the client's side (e.g., MQ channels). A Web service provider has to be able to communicate with a messaging provider e.g., to consume a message from a queue (which calls for a Message Driven Bean in J2EE environment). So while these issues are not insurmountable, they certainly make things more complicated and, among other things, require full J2EE stack for Web services implementation.

WS-RM is currently being standardized by [WS-I](#), there is also [project "Tango"](#) that specifically addresses interoperability between Sun and Microsoft implementations. Unfortunately, WS-RM support is still far from being uniformed among Web services vendors. Also, from what I understand, WS-RM by itself is not sufficient since it does not define how (or if) a WS-RM conversation participates in a transaction. So then WS-Transaction or some proprietary vendor extension is needed, and that in turn calls for the ability to support XA. I also suspect that the use of a persistent storage on the client is the only way to support all WS-RM requirements (e.g., what if the client goes down in the middle of a message exchange?), so this makes clients "fatter" and more complex.

The bottom line is that "good old" SOAP over HTTP is the easiest and the most interoperable way to implement Web services today. So how much can we trust SOAP/HTTP Web services and should HTTP even be considered in an enterprise setting where QoS is almost always a requirement?

First, we need to remember that HTTP as well as virtually any other communication protocol today (including IIOP and protocols used by messaging providers) is based on TCP. The whole purpose of TCP is to be able to transfer data reliable and so it employs [acknowledgements and sliding window](#) mechanism to guarantee the delivery of data. What does it mean in terms of Web services? Say, we have a one-way service. If we invoked a service and received a successful reply (no SOAP faults or HTTP errors), we can be confident that our SOAP message reached its destination. But what if the connection went down in the middle of the call and we received a timeout error? Our message exchange is now in an ambiguous state. If the message has not been received, we need to invoke the service again, but, on the other hand, if it has been received, the duplicate message may lead to an inconsistent state if the service provider is not able to handle duplicates correctly.

QoS provided by messaging systems or WS-RM helps us in this situation by ensuring that the message will only be delivered once; messaging systems can also handle re-delivery of messages and "store and forward" (there are other QoS policies besides "exactly once", but "exactly once" is the most stringent one). Messaging systems also provide another important benefit by allowing a Web services call to participate in an atomic transaction. This allows service consumers to keep in synch multiple resources (including the ones provided by Web services) thus improving data integrity and reliability.

So where does it leave SOAP/HTTP services? Based on the explanation above, SOAP/HTTP is not the best fit when:

- Service providers can't handle message duplicates (in other words, an operation performed by the service is not [idempotent](#)).
- Different data resources owned by service provider and service consumer must be in synch at all times.

However, we still might be able to use SOAP/HTTP if we make our service consumers a little smarter. Specifically, service consumers must be able to meet the following requirements:

- Consumers must be able to retry a Web service call in case of a failure due to a connectivity error. In the simplest case, a consumer's application may choose to show the error message to an end user and let the user press "submit" button again. However, most consumers will probably choose to automate the retry logic (e.g., have X number of attempts) and at least log unsuccessful attempts and potentially alert an application administrator (note that some Web services clients have built-in retry capabilities).
- Consumers must be able to uniquely identify the data that they are sending to providers. Suppose we have "addEmployee" service that takes an employee XML message as an input. The employee message must have unique ID set by the consumer of the service, so that when the invocation is retried, the service will be able to detect that the employee was already added as part of the previous call.

This means that popular techniques using database sequences or autoincrement fields for generating unique IDs do not work with SOAP/HTTP Web services. Service consumers must implement their own way of creating unique IDs (perhaps relying on some kind of a natural key or using UUID).

- Consumers must be able to handle certain application-specific errors (SOAP faults). For example, "addEmployee" service may return "Employee with this ID already exists" error after "addEmployee" call was retried in response to a connectivity failure. The consumer's application will have to stop retrying after catching this error. This situation may also require an end user (or an administrator) involvement to verify that the employee was indeed added.

As an example, let's take a look at "Create/Read/Update/Delete" (CRUD) operations. While real-life services oftentimes do much more than just "CRUD" (e.g., calculations), it is a valid simplification for our purposes.

Operation	Is Idempotent?	Required Service Consumer Behavior
Create	No	Must be able to retry the call Must be able to handle "record already exists" error.
Read	Yes	None
Update	Yes, unless update involves calculating new values based on the existing values, e.g., incrementing a counter	Must be able to retry the call
Delete	No	Must be able to retry the call Must be able to handle "record does not exist" error.

So what is the takeaway? SOAP/HTTP can be used in many (if not most) situations, however, implications of this decision must be fully understood. All service consumers must be made fully aware of these implications. Most importantly, service consumers must implement proper logic for handling connectivity failures and application errors. In some cases, users of service consuming application may need to be instructed about how to react to certain application errors.