The need for a general context definition in Web Services

Mark Little (*mlittle@redhat.com*)

It has long been recognized that the World Wide Web is probably the most successful distributed system created. It is inherently loosely coupled (clients and servers frequently interact across the globe) and highly scaleable (many thousands of Web sites). There are a number of factors that can be attributed to the Web's success, but two of the most important are:

- Sessions between clients and servers are maintained only long enough to transfer an HTML page and are dropped immediately afterward. This means that costly resources (e.g., TCP/IP connections, threads, processes) are not maintained for long durations, particularly when there are many users interacting with a service.
- Server interactions are either stateless, meaning that any instance of a Web server offering a particular service, e.g., airline reservation, can field the request, or information required to identify a previous user (and possibly state) is propagated with the invocation, e.g., the cookie.

Both of these factors mean that clusters of servers can relatively easily be used to distribute the load and provide improved availability/fault-tolerance to users. Web servers offering critical services are typically deployed over a cluster of machines. A locally distributed cluster of machines with the illusion of a single IP address and capable of working together to host a Web site provides a practical way of scaling up processing power and sharing load at a given site. Commercially available server clusters rely on a specially designed gateway router to distribute the load using a mechanism known as network address translation (NAT). The mechanism operates by editing the IP headers of packets so as to change the destination address before the IP to host address translation is performed. Similarly, return packets are edited to change their source IP address. Such translations can be performed on a per session basis so that all IP packets corresponding to a particular session are consistently redirected.

Most proponents of Web Services agree that it is important that its architecture is as scalable and flexible as the Web. As a result, the current interaction pattern for Web Services is based on coarse-grained services or components. The architecture is deliberately not prescriptive about what happens behind service endpoints: Web Services are ultimately only concerned with the transfer of structured data between parties, plus any meta-level information to safeguard such transfers (e.g., by encrypting or digitally signing messages). This gives flexibility of implementation, allowing systems to adapt to changes in requirements, technology etc. without directly affecting users. Furthermore, most businesses will not want to expose their back-end implementation decisions and strategies to users for a variety of reasons.

In distributed systems such as CORBA, J2EE and DCOM, interactions are typically between stateful objects that resided within *containers*. In these architectures, objects are exposed as individually referenceable entities, tied to specific containers and therefore often to specific machines. Because most Web Services applications are written using object-oriented languages, it is natural to think about extending that architecture to Web Services. Therefore a service exposes *Web Services resources* that represent specific states. The result is that such architectures produce tight coupling between clients and services, making it difficult for them to scale to the level of the World Wide Web.

Right now, there are two primary models for the session concept that are being defined by companies participating in defining Web services: the WS-Addressing EndpointReference with ReferenceProperties[1] and the WS-Context explicit context structure[2]. The WS-Addressing session model provides coupling between the web service endpoint information and the session data, which is analogous to object references in distributed object systems. WS-Context provides a session model that is an evolution of the session models found in HTTP servers, transaction, and MOM systems, allowing a service client to more naturally bind the relationship to the service dynamically and temporarily. The client's communication channel to the service is not impacted by a specific session relationship.

If a session-like model based on WS-Addressing were to be used when interacting with stateful services, then the tight coupling between state and service would impact on clients. As in other distribution environments where this model is used (e.g., CORBA or J2EE), the remote reference (address) that the client has to the service endpoint *must* be remembered by the client for subsequent invocations. If the client application interacts with multiple services within the same logical session, then it is often the case that the state of a service has relevance to the client only when used in conjunction with the associated states of the other services. This necessarily means that the client must remember each service reference and somehow associate them with a specific interaction; multiple interactions will obviously result in different reference sets that may be combined to represent each sessions.

For example, if there are N services used within the same application session, each maintaining m different states, the client application will have to maintain N*m reference endpoints. It is worth remembering that the initial service endpoint references will often be obtained from some bootstrap process such as UDDI. But in this model, these references are stateless and of no use beyond starting the application interactions. Subsequent visits to these sites that require access to specific states must use different references in the WS-Addressing model.

This obviously does not scale to an environment the size of the Web. However, an alternative approach is to use WS-Context and continue to embrace the inherently loosely-coupled nature of Web Services. As we have shown, each interaction with a set

[1] W3C WS-Addressing Working Group, http://www.w3.org/2002/ws/addr/

[2] OASIS WS-CAF Technical Committee, http://www.oasis-open.org/committees/ws-caf

of services can be modeled as a session, and this in turn can be modeled as a WS-Context activity with an associated context. Whenever a client application interacts with a set of services within the same session, the context is propagated to the services and they map this context to the necessary states that the client interaction requires.

How this mapping occurs is an implementation specific choice that need not be exposed to the client. Furthermore, since each service within a specific session gets the same context, upon later revisiting these services and providing the same context again, the client application can be sure to return to a consistent set of states. So for the N services and m states in our previous example, the client need only maintain N endpoint references and as we mentioned earlier, typically these will be obtained from the bootstrap process anyway. Thus, this model scales much better.

Although WS-Addressing and WS-Context compliment one another, it is an unfortunate fact that developers are using WS-Addressing for session-oriented interactions, with all of the associated implications on application brittleness. Another factor is political: despite significant positive user and developer feedback, WS-Context has not received as much backing from Web Services heavyweights.

We believe that the W3C and the Web Services community should address the lack of context and session management within the architecture before it is too late. Whether or not that is through WS-Context is immaterial. Although WS-Context is certainly one way of achieving the goals of loosely coupled session management and offers Web-type scalability for Web Services, there may be other approaches. We cannot afford to let the Web Services architecture degenerate into “CORBA with angle-brackets”, as it surely must if the WS-Addressing approach to session management gains momentum.

The recommendation of the authors is that the W3C should acknowledge that context and session management are a fundamental missing component of the Web Services architecture and solicit responses on how to best address it.