Use RosettaNet-based Web services, Part 2: BPEL4WS and RosettaNet

Standing on the shoulders of a giant: Web services with RosettaNet

Suhayl Masud

July 22, 2003

Sir Isaac Newton once wrote, "If I have seen farther than others, it is because I was standing on the shoulders of giants". We should pay great care to this wisdom and ensure that Web services leverage existing knowledge and expertise to "see farther". The shoulders on which Web services can stand belong to "giants" like RosettaNet, which has already spent the past 5 years creating and implementing automated e-business dialogues between partners. Web services can take these efforts further by applying ubiquitous technologies to the e-business dialogue and bringing the e-business dialogue within affordable reach, enabling widespread use.

This is the second article in a series that demonstrates the true potential of Web services by creating an e-business dialogue that can be used to conduct real business. This e-business dialogue will be based on a real-world business problem, which will be addressed by using a proven solution from RosettaNet. In this series, I will show you that the most important aspect of Web services is the e-business dialogue. In this article, I will cover the following:

- Challenges in using Web services for e-business dialogues
- The components of RosettaNet
- How to translate RosettaNet into WSDL and BPEL
- The complementary nature of RosettaNet and BPEL4WS.

In Part 1 of the series (see Resources), I explained that the true value of Web services lies in creating collaborative e-business dialogues between business partners. I explained what e-business dialogues were, and how they worked. The following articles will look at creating an extended e-business scenario using the RosettaNet PIP3A4 and BPEL4WS.

Tuning the Web services mindset

Realizing the true potential of Web services -- affordable and widespread e-business dialogues based on ubiquitous technologies -- requires slight philosophical changes to the current Web services mindset. Luckily, BPEL4WS provides the necessary changes.
E-business dialogues between peers

First, organizations interacting via Web services need to be treated as peers operating on an equal basis. This means that a service requester and a service provider simply become partners where the service provider engages in a proper dialogue, unlike its current role of simply responding to requests. To do so, the service provider must have a way to communicate back with the service requester. Currently, this communication happens either on the same http connection that the request came in on, or by stripping out the technical address from the http headers of the request. To enable a proper dialogue between peers, the address must have business semantics; it must belong to a business transaction. With BPEL4WS, Web services get this necessary boost which enables peer dialogues.

Interactions must be stateful

In the current Web services world, most Web services are synchronous. This means that the requester must be put on hold for the response. This works great if the response is available immediately, which is the case in simplistic request/response implementations like a browser asking a Web server for a Web page, or an application looking up the trading value of a stock. However, in longer running business transactions, it is not possible to continue to wait for a response that might take hours or even days. Imagine if you called up a business colleague on the phone, and you were put on hold for a few hours while the colleague prepared a response for your request.

In the current request/response setup, the service provider is only concerned about responding on the same open connection; the service provider does not store means to contact the requester on what state the dialogue has reached. In the (synchronous) request/response setup, there is no state of the dialogue; it is simply based on a request, to which there is a response. To conduct proper e-business dialogues, the interactions must be asynchronous and state-ful.

Realistic e-business scenarios

While BPEL4WS provides Web services with the necessary means to enable e-business dialogues between peers and create asynchronous stateful interactions, it is still necessary to use these means to create realistic e-business scenarios. In other words, BPEL4WS provides all the tools to create robust executable business processes; it is up to the partners to express meaningful e-business dialogues using these tools.

While you could cook up our own e-business dialogues, you can do a much better job by borrowing from experts that have already spent a great deal of time and effort creating expert e-business dialogues. As I explained earlier, this is where you can leverage RosettaNet components, to build your own real-world e-business dialogues. The actual details and code will be revealed in the next two articles in the series. In this article, I will talk about the conceptual steps required to create RosettaNet-based Web services.

Leveraging RosettaNet

Recall that a good e-business dialogue has the following features:
• Open and accessible public interfaces
• Partner roles
• Standard messages exchanged in agreed upon choreography
• Standard vocabulary
• An environment of security and trust.

For the e-business dialogue being created in this series, the PIP3A4 purchase order process from RosettaNet will help define the public interface. It will use the partner roles and choreography defined in the PIP as well as the standard messages from the RosettaNet PIP and use them as the BPEL4WS messages.

Now let us look at the components of RosettaNet and how RosettaNet-based e-business dialogues work.

**Figure 1: How RosettaNet works (Source RNIF 2.0)**

RosettaNet standards have three components:

• Partner Interface Processes (PIPs)
• Dictionaries
• RosettaNet Implementation Framework (RNIF).

The PIPs are specifications for standard messages and choreography to conduct a specific business activity. The dictionaries standardize the words used in the messages, and the RNIF contains specifications to enable trust and security in the message exchange. Our interest lies in selecting relevant components from RosettaNet and implementing them in the Web services environment, and, for this article, I will focus only on how to understand a PIP specifications and which components you can use from it.
RosettaNet PIPs

RosettaNet PIPs create standard e-business dialogues for common business activities like order and inventory management, transportation, sales forecasting, etc. The PIPs are organized in functionally logical groupings of segments and clusters. For example, the e-business dialogue this example will conduct is PIP3A4 *Purchase Order Request*, found in the *Quote and Entry Segment* grouping, which belongs to the *Order Management* cluster. All PIPs are available for public download at the RosettaNet Web site.

The PIP specification package is a zipped file that contains three types of documents. The specification itself is a Word document, help and guidelines for the messages are in HTML documents, and the message structure and content is captured in the XML DTDs. For the e-business dialogue, I will extract the process definition and choreography information from the UML diagram and associated tables in the PIP3A4 specification Word document, and I will use the DTDs to help create the messages and operations.

**Figure 2: The RosettaNet PIP Specification Package**

If you look at *figure 3*, you see how the *Request Purchase Order PIP* works. The partners exchange two messages and two receipts; the two messages are the Purchase Order Request and the Purchase Order Acceptance, with receipt acknowledgment signals in between two signal the receipt of messages.
I want to model the messages as WSDL operations. What is missing from the PIP Package is the DTD for the receipt messages. Since this message is common to all PIPs, it is part of the implementation framework, and I will simply pull it out of the RNIF package (also available on the RosettaNet Web site, see Resources).

To model the PIP receipt and messages as WSDL operations, I need to convert the DTDs to XML Schema documents; this is a simple task since most XML authoring tools can perform this conversion for you. However, there is a small problem with the DTDs in the RosettaNet specification that is due to an improper object model of the elements; each DTD appears to be stand-alone and not part of a bigger model, and therefore elements at times are duplicated or re-defined in different DTDs. While this is not a big problem in the RosettaNet environment, with Web services, the schemas from these DTDs need to be combined into the same document. RosettaNet has promised these problems will go away when they roll out a re-modeled, well-connected schema. In the two DTDs I converted to Schema, I manually renamed a couple of objects and changed references to fix this problem. Having extracted the messages I want to use in the e-business dialogue, my first step is to translate these messages into Web service operations. The next step will be to define the business dialogue as a BPEL4WS definition.

**Mapping RosettaNet into WSDL and BPEL**

As there are already several articles on developerWorks introducing WSDL (see Resources), I will assume that you are familiar with WSDL. The mapping between RosettaNet PIP and WSDL is as follows:
• RosettaNet message DTDs contain the types used in the message and these elements map to the types definitions of WSDL.
• RosettaNet message definitions, including the message names, are mapped to the message definitions in WSDL.
• The actions in a RosettaNet PIP are mapped to operations in WSDL.
• The partner roles from RosettaNet PIPs are mapped to Partners in BPEL4WS.
• Choreography from the RosettaNet PIP is implemented in the choreography of the abstract and executable business process in BPEL4WS.
• Exception messages from RosettaNet RNIF are mapped to the exception handling mechanisms of BPEL4WS.

Some concepts have no direct mapping between RosettaNet and BPEL4WS and require workarounds, as follows:

• The message exchange control parameters from RosettaNet (number of retries, hours to complete an activity etc); the workaround is to agree on these issues outside of the business process description.
• The messaging layer from RosettaNet RNIF provides methods for packing and unpacking messages and transporting them in a secure and reliable manner. The BPEL4WS and WSDL paradigm provides its own methods for encoding, transporting, and securing messages. The workaround is to use best practices from RosettaNet in the Web services paradigm.
• RosettaNet offers a third-party view of the e-business dialogue while BPEL4WS encodes it from a specific party's point of view, the workaround is for partners to create the e-business dialogue from a particular point of view and then to create and agree to its mirror image as well.

On the other hand, BPEL4WS offers the following advanced features that RosettaNet does not yet have:

• The ability to compose larger e-business scenarios by including several partners in the dialogue
• Logic statements like WHILE and SWITCH that allow flexible business processes that can change process flow based on the specific case
• The ability to tie in back-end systems into the business process
• A robust set of control activities for creating sophisticated workflows.

Are RosettaNet and BPEL4WS competing?

While RosettaNet and BPEL4WS have a slight overlap, they are certainly not competitors. Together, they are ideally suited for use as a powerful combination, with BPEL4WS as a language to define e-business processes, and RosettaNet to create standardized industry e-business processes. Thus, each handles the other's shortcomings and thus do not compete directly.

In RosettaNet, the e-business dialogues are defined with UML activity diagrams, text tables, and XML documents. RosettaNet does not currently use an overall e-business dialogue specification language that can be used to express the entire e-business dialogue. RosettaNet has been
eagerly exploring the use of a standard e-business process definition language to describe its e-
business processes, and BPEL4WS could be the useful candidate it is looking for.

The BPEL community, on the other hand, could benefit a great deal from leveraging RosettaNet
as well, as it would provide them with proven e-business dialogues, built and used by industry
experts. The conceptual functions of RosettaNet PIPs and BPEL4WS are complementary, and
using them in combination will certainly enrich Web service definitions.

Conclusion

In this article, I have discussed some challenges in creating and using real e-business dialogues
with Web services, and how BPEL4WS solves them. I introduced the specification of RosettaNet
and how to leverage them. I showed you at an abstract level how to map RosettaNet to WSDL
and BPEL4WS, and I ended the article by discussing the complementary nature of RosettaNet
and BPEL4WS. This article has been high level, but the next two articles contain a lot of code.
The following articles will look at creating an extended e-business scenario using the RosettaNet
PIP3A4 and BPEL4WS.

© Copyright IBM Corporation 2003
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)