IS GRID COMPUTING THE NEXT BIG THING?

August 2003 Volume 3 Issue 8

WebServices JOURNAL

.NET J2EE XML

WSJ2.COM

From the Editor
Double Standards
by Sean Rhody pg. 3

Industry Commentary
Infrastructure-Level Web Services
by Anne Thomas Manes pg. 58

RETAILERS PLEASE DISPLAY UNTIL SEPTEMBER 30, 2003
$6.99US $7.99CAN
08
0 71486 03420 9

SYS-CON MEDIA

FOCUS ON GRID COMPUTING

Introducing Open Grid Services *Savas Parastatidis*
An infrastructure built on existing technologies 10

Grid Services Extend Web Services *Andrew Grimshaw*
A solid foundation for service consumer reliability *& Steve Tuecke* 22

Identifying and Brokering Mathematical Web Services *Mike Dewar*
A formal language can move service discovery forward faster 44

WSJ Feature: Building a Business Logic Layer Over Multiple Web Services *Rajesh Zade*
Leveraging multiple Web Services *& Avinash Moharil* 16

WSJ Feature: Introducing BPEL4WS 1.0 *Mark Little*
Building on WS-Transaction and WS-Coordination *& Jim Webber* 28

Technology: Web Services Made Easy with Ruby *Aravilli Srinivasa Rao*
A simple development method 34

GLUE: Creating Web Services Using GLUE *Shannon Ma*
An easy development framework 38

Product Review: SoftArtisans FileUpEE from SoftArtisans *Joseph A. Mitchko*
Industrial-grade uploading/downloading for Web services 54

WebServices JOURNAL
.NET J2EE XML

EDITORIAL OFFICES
SYS-CON MEDIA
135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 FAX: 201 782-9637
WEB SERVICES JOURNAL (ISSN# 1535-6906)
Is published monthly (12 times a year)
By SYS-CON Publications, Inc.
Periodicals postage pending
Montvale, NJ 07645 and additional mailing offices
POSTMASTER: Send address changes to:
WEB SERVICES JOURNAL, SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645

SYS-CON MEDIA

Double Standards

Written by
Sean Rhody

Author Bio:
Sean Rhody is the editor-in-chief of Web Services Journal and managing editor of WebLogic Developer's Journal. He is a respected industry expert and a consultant with a leading consulting services company.
SEAN@SYS-CON.COM

In June I attended the JavaOne conference out in San Francisco, to keep up with what the Java world was doing, and to see how it impacted Web services. I see a number of parallels between Web services and the way that the various Java specifications have been created, and some key differences. But even further, I went to a number of sessions on Web services and was reminded, once again, that the lack of a single standards body is a serious roadblock to implementation of Web services.

Now, of course I'm not talking just about UDDI, WSDL, SOAP, and XML. These are pretty much okay for implementation, but when you have to have an entire standards organization such as the WS-I, whose only purpose is to make sure that Web services implementations interoperate, there's something wrong.

Not that I have anything against the WS-I, mind you – they're doing very necessary work. But it's only necessary because the standards stink. That's right. They stink. If my SOAP stack doesn't work with your SOAP stack (sounds like a bad prison movie) there's just something fundamentally flawed in how the specification was created. And there are plenty of those stories out there.

But I was further reminded of the mess we're in by some of the Web services presentations. While obviously biased toward Java (it was JavaOne, after all), what really got me was the way everyone needed to explain how this specification came from HP, that standard was developed by W3C, and OASIS has a competing specification to some other specification. It's clear that there are too many bodies producing standards, not to mention too many standards themselves.

The Java model works somewhat better, with a single standards organization and the JSR process. Rather than develop competing specifications (SAML or WS-Security, for example), the JCP provides guidance from multiple companies toward the creation of a single standard that all Java vendors will comply with. No one has to decide whether to use BPML or BPEL, or the Java equivalent.

Not that the model isn't without its flaws as well. I was speaking with several board members of the WS-I recently and in their view multiple standards are good, in that they are a sign of intense interest and usually of innovation.

And that's good to a point. That point being the place where people are paralyzed by confusion over which standard to use, what product to buy, and how to manage the process. The Java concept of a single standards body was an acknowledgement by Sun that to grow Java widely, they needed to give up a great deal of control to other organizations who would also profit from participation. The ingenious aspect of the development of Java was that the control was released gradually, and the single body retained all control over the platform.

For us, I fear that the cat is already out of the bag. With two major standards-making bodies, one standards integration body, and vendor companies with multiple divisions backing different horses, we're never going to get to the same level of cooperation as the JCP.

Instead, I would propose that WS-I become the central Web services body, and that the members of the other bodies treat them as the Supreme Court of Web services. Once they rule on a specification, let there be no further disputes. Let's limit the number of specifications so the innovations can go toward making a smaller set of standards better.

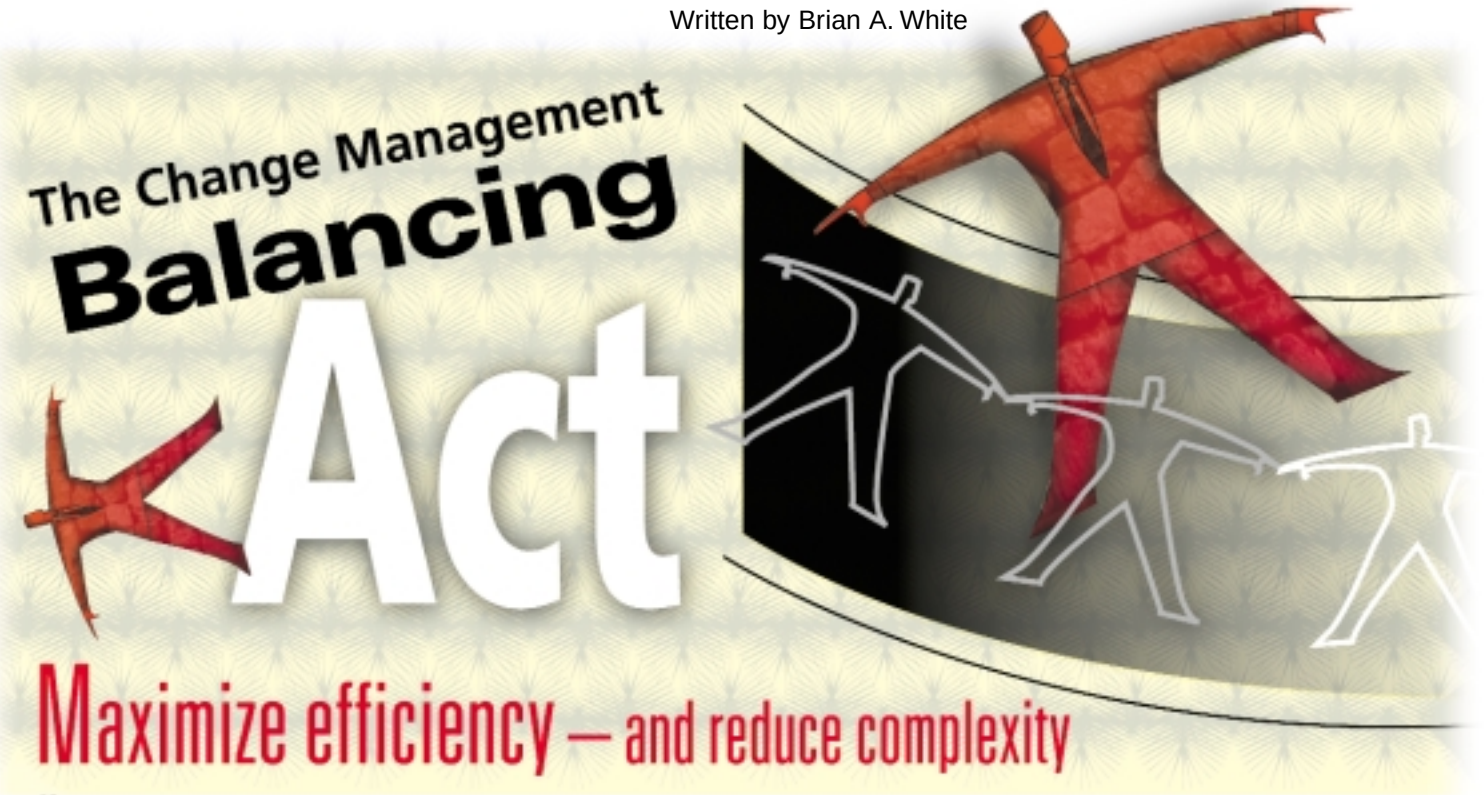
Of course the WS-I may not want to act as the final arbiter of Web services fate, and for various reasons, many vendors may not want the WS-I as currently constituted to be the sole determining body for Web services. That's fair. I suppose we can always create yet another body to do the task. We'll probably need a few more standards as well, just in case.

Mindreef

www.mindreef.com/nl0308

Written by Brian A. White

The Change Management Balancing Act



Maximize efficiency – and reduce complexity

Managing change in a software system is a lot like balancing your personal finances. With or without a resource allocation plan, the assets available and the demands placed on them change constantly. Whether it's your code or your checkbook, the result of mismanaging change over time is likely to be the same: disaster.

The complexity and volatility of today's business demands have rendered "traditional" (i.e., ad hoc) methods for software change management obsolete. Distributed teams, multiple platforms, strict cost accountability, compressed release cycles: these mounting pressures dramatically escalate the risks of deploying new applications and integrating existing ones. With technology now serving as the primary interface to customers and partners, an organization may literally "bet the business" on the outcome of a software development effort. Now more than ever, the need to effectively control the impact of change is among the most critical aspects of software project management.

> "A successful change management solution helps deal with ... complexity, while mitigating project risk"

Done right, change management reduces complexity and maximizes efficiency. Done wrong, change mismanagement can bust budgets, trash schedules, and undermine team efforts. The question is not whether to manage change, but how to do it in a way that facilitates rapid response while providing adequate control.

Change Management 101

The ultimate goal of change management tools and processes is to mitigate risk. The underlying key to risk reduction is the ability to make informed decisions about what changes are most critical, in line with the organization's resources and revenue goals. The laws of physics notwithstanding, finite resources equate to a finite potential for change per unit of time. Balancing that equation to maximize the return on investment is the essence of successful change management.

Managing change entails more than just tools and training – it also requires a commitment to process. Vastly outweighing these investments are bottom-line benefits like:

- Protection of code investments
- Improved system quality
- Faster time-to-market via parallel development

AUTHOR BIO:

Brian A. White has a decade of hands-on experience with software configuration management tools and methods. He is responsible for enterprise change management product strategy for Rational Software, IBM Software Group. Brian is the author of Software Configuration Management Strategies and Rational ClearCase: A Practical Introduction.

- Support for cross-functional collaboration and distributed development
- Reproducibility of deliverables
- Greater development scalability

Each organization must apply best practices in line with existing business and development processes. There is, however, a short list of factors that are common to nearly all successful change management solutions.

Success Factors

When faced with a complex challenge like managing software change, it's human nature to look for a silver bullet. A common mistake is to roll out a software configuration management (SCM) or defect tracking product, and call it a day. But even the best tools are only as good as the process behind them. Just installing Quicken, for instance, does not mean you have a sound financial plan. Each business should first examine how change takes place in its unique environment, and then apply the tools that best augment that process.

Take, for example, the management of software configurations. Here you must be concerned not only with storing files, but also with your overall integration strategy. If you place all your project code under version control, and 10 developers are making changes to that code base, how do you intend to build the next release of your product from the pool of versions that are being created? Control over checking files in and out is essential; but, in itself, that's not change management. You also need to implement an integration/build strategy. This could be as simple as building the latest code that has been checked in every Friday night, or organizing a multitier promotion model.

However your change management approach ultimately works, it must be flexible enough to strike an appropriate balance between efficient control and over-control. Many organizations attempt to address the problem of change by, in effect, resisting it with rigid controls. This usually hinders the individual's ability to efficiently make changes.

Change management tools and processes ought to be carefully tailored to the task at hand. If you've got three developers working on a prototype that's scheduled for delivery in six months, and they all sit next to one another, a fairly relaxed process is adequate. If you're an IT organization building a mission-critical application using distributed internal and outsourced development resources, you need to control change much more tightly.

These considerations lay the groundwork for the successful implementation of change management tools and processes. Some proven and practical ways to put change management into action include:

- Storing important project artifacts in a central repository, and controlling access to them via a straightforward and reliable process
- Implementing some form of version control to enable parallel development
- Grouping related files logically, such as by component, and managing those files together
- Tracking system requirements, defects, and requests for enhancement end-to-end to ensure traceability

Let's look at each of these in more detail.

Store Artifacts in a Central Repository

The most fundamental element in any change-management strategy is basic file management. You need to store software and related project artifacts in a central repository, such as a version control or SCM system. To ensure that valuable work is protected from loss or corruption, you also need to control access to the repository via a simple and reliable check-in/checkout process.

Think about what happens when teams don't have this basic level of file management in place. Critical project code and documents reside on various servers and on individuals' laptops. Even with centralized file storage, developers can accidentally copy over each other's changes, wasting time, reintroducing bugs, and causing testing nightmares. Versions needed later, such as to rebuild a previous release, may be unrecoverable.

Beyond basic version control, two additional SCM capabilities are extremely useful. The first is workspaces, which greatly simplify keeping even the smallest development teams in sync. A workspace is basically either a copy or (preferably) a "virtual view" of some part of the central code repository, stored on individual developers' desktop systems. Periodically, the version control system can automatically update each workspace with the latest versions of everyone's files. This saves developers time by providing the most up-to-date versions.

In addition to workspaces, the availability of an audit history is also important. Most SCM tools provide some level of auditing. Audit histories let team members quickly see who made what change to what artifact, and when they made it. This simplifies both traceability and reproducibility.

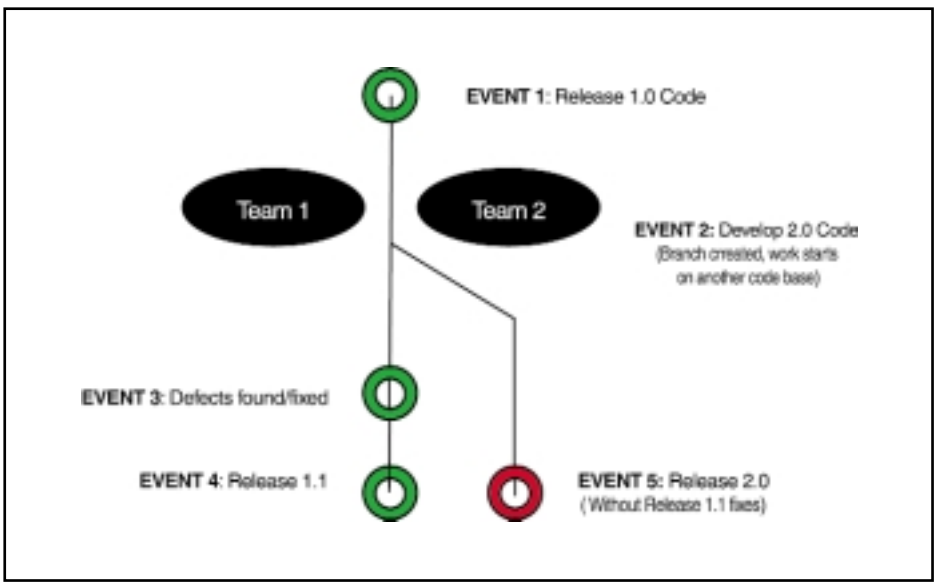


FIGURE 1 **Without SCM, multiple code bases are commonplace, costly, and confusing, especially as additional defects are discovered.**

Implement Version Control

In line with managing a central repository, a related necessity is some form of version control. This ability to manage concurrent changes to files is the foundation for parallel development, as well as baseline management. Without it, technical managers cannot shorten the "critical path" by breaking the serial task of developing complex software into multiple, parallel tasks.

Version control must be addressed on two levels:

- First, you need to control the process whereby two or more developers make changes to the same file at the same time. This requires a check-in/out mechanism, as well as automated support to resolve and merge multiple sets of changes.
- Second, once you've moved beyond the initial release, you'll most likely need procedures that enable two teams of people to work in parallel on two different releases of the system, using the same code base.

> "The ultimate goal of change management tools and processes is to mitigate risk"

Figure 1 illustrates a common example of the cost and risk associated with a lack of adequate version control. Two teams are working in parallel: one on Release 2.0, the other on bug fixes to Release 1.0. Unfortunately, some of the defects fixed in Release 1.0 were not properly merged into Release 2.0. As a result, customers uncover problems in Release 2.0 that they reported and found to be fixed in point Release 1.1.

To develop code in parallel, version control tools must be flexible enough to allow fine-tuning of those policies that balance individual isolation against the pace of integration. If each engineer saw every change that someone else made when they clicked "Save," the changes would pile up so quickly that a stable release could never be built. Likewise, working in isolation for long periods can make integration unacceptably difficult: issues like misunderstandings about interface design need to be rectified early and often. The deeper you get into the development life cycle, the more important it is to integrate changes frequently. When everyone is working on skeleton code, the ability to view or work with others' changes is less important than when you are fixing a defect two days before "code freeze."

Group Files Logically

Another factor in effective SCM is the logical grouping of files in support of component-based development, software reuse, and baseline control. A logical file organization simplifies the process of

integrating a complete change "activity," as opposed to working with random versions of individual files. The organization of artifacts around versioned components also enables a team to manage its code baseline relatively naturally after each testing/integration step, provided sufficient energy is devoted up-front to aligning the file organization with the system architecture.

In general, the more intuitively a code base is organized, the less process that is needed to ensure that whatever code changes comprise a fix or enhancement, they are baselined together and included (or not) in a build together. Good file organization also simplifies the tracking of defects and enhancements by making it much easier to tie the delivery of new functionality to changes in the code base. I once worked with a customer who had organized their code base alphabetically, by file name. You can imagine the uselessness of statements like, "We're going to baseline files F through H on Friday."

Track Defects and Enhancements

Most development organizations realize that defects and enhancements "should" be tracked. For some, it suffices to do this somewhat informally, such as through spreadsheets. But whatever it takes to do it – do it.

Note that the process implemented to track defects is likely to differ from the process for tracking enhancements. Enhancements often involve more up-front evaluation because of the amount of work they can require. Bug fixes become increasingly important in the end game. But in either case, the decision-making process is at least as important as the tools involved. What is critical to the customer? What is possible in the time allotted? What must the next release look like relative to the competition?

The relationship between traceability and product planning also revolves around change management information. What data do you need to determine when a product is ready to ship? What do you need to know to assess risk? Is a defect related to complex, breakable code; or to a more robust or isolated component?

What is the acceptable quality level you're striving for? This is where defect and enhancement tracking meets requirements management, because quality is ultimately measured by the relationship between the system's functionality and what the customer wanted in the first place.

The Balance Point

Software systems only grow in complexity. A successful change management solution helps deal with that complexity, while mitigating project risk. Remember: the question is not whether to manage change, but how to do it in a way that balances rapid response with adequate control. The challenge is to maintain that balance across systems both simple and complex, and across the entire development life cycle.

Introducing Open Grid Services

An infrastructure built on existing technologies

In June 2003, the Global Grid Forum (GGF) adopted the Open Grid Services Infrastructure (OGSI) specification as a GGF standard. OGSI is essential to the Open Grid Computing vision as it is the foundation on top of which the building blocks of future Grid applications will be placed. Those building blocks, the Grid services, are being defined by various GGF working groups, with the Open Grid Services Architecture (OGSA) working group orchestrating the entire process.

AUTHOR BIO:

Dr. Savas Parastatidis is the chief software architect at the North-East Regional e-Science Centre (NEReSC), Newcastle, UK. He is NEReSC's expert in Grid computing technologies and standards. Previously, Savas co-led an R&D team at HP's middleware division that produced the world's first XML-based transactioning system and represented HP during the early stages of the OASIS BTP standardization effort.

SAVAS.PARASTATIDIS@NEWCASTLEAC.UK

This article introduces OGSA, presents the OGSI specification, and discusses the significant role of Web service standards in Grid computing.

Grid Computing

Grid computing has been a hot topic in the news recently. Major vendors like IBM, HP, and Sun have not only been advertising their plans to support Grid computing, but they have also started rolling out their first e-business and e-science solutions.

All the interest in Grid computing led to the formation of the Global Grid Forum (GGF; www.ggf.org), a forum for the discussion of Grid-related ideas and the promotion of enabling technologies. One of its main activities is the creation of a standards-based platform for Grid computing with emphasis on interoperability.

But, what is the Grid? And, what does the term "Grid computing" mean? It could be about virtual organizations, or the integration of distributed resources, or a universal computer, or even about interconnecting supercomputers depending on who you talk to. A definition of the Grid is beyond the scope of this article. Your favorite Web search engine will provide you with many links to definitions and information. There are also a number of books out there from which you can draw information.

OGSA

The blueprint of the Grid architecture is defined by OGSA (www.ggf.org/ogsa-wg) as a set of fundamental services whose interfaces, semantics, and interactions are standardized by the GGF working groups. OGSA plays the coordinating role for the efforts of these groups. It identifies the requirements for e-business and e-science applications in a Grid environment and specifies the core set of services and their functionality that will be necessary for such applications to be built, while the technical details are left to the groups.

While OGSA has adopted a services-oriented approach to defining the Grid architecture, it says nothing about the technologies used to implement the required services and their specific characteristics. That is the task of OGSI.

Figure 1 shows the relationship between OGSA, OGSI, and the Web services standards. Also, a list, which is not exhaustive, of candidate core services is presented. It is expected that in the months to follow, OGSA will standardize this list.

OGSI

The OGSI working group decided to build the Grid infrastructure on top of Web services standards, hence leveraging the great effort – in terms of tools and support— that the industry has put into the field. Nevertheless, the group identified some key

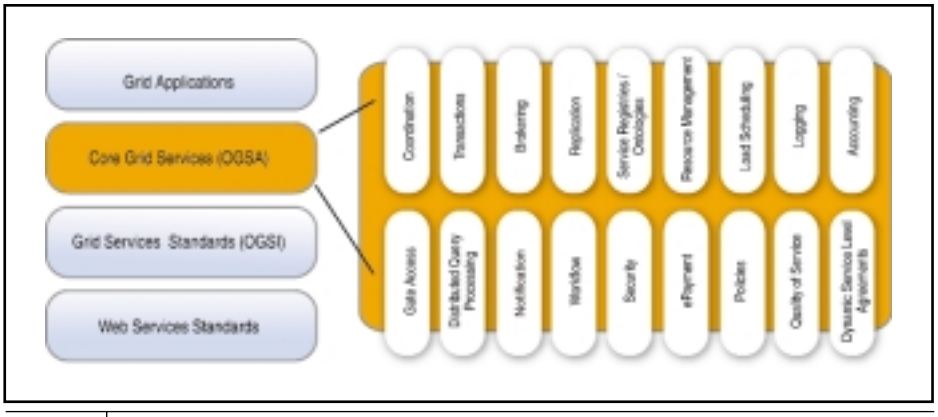


FIGURE 1 | **J2EE container environment**

characteristics missing from Web services standards that they thought would be instrumental in building Grid applications, including the ability to create new services on demand (the Factory pattern); service lifetime management; statefulness; access to state; notification; and service groups.

OGSI is based on the concept of a Grid service instance, which is defined as "a Web service that conforms to a set of conventions (interfaces and behaviors)" [www.ggf.org/ogsi-wg]. All services in the OGSA platform must adhere to the conventions specified by OGSI and, therefore, they are all Grid services. It is crucial to note that the term "Grid service" is used here to refer to all aspects of OGSI.

Grid Service Description

According to the OGSI specification, each Grid service instance is associated with some transient state and has properties that control its lifetime. Additionally, the specification defines the operations that must be supported and the way the state and lifetime-related information can be accessed. As with Web services, a WSDL document – with few noncompliant extensions to the WSDL standard – is used to describe the operations, messages, and portType (renamed to "interface" in WSDL 1.2) of a Grid service instance. In OGSI terms, the WSDL document is called the "Grid service description."

The specification mandates that the portType of a Grid service inherits (or "extends" in WSDL 1.2) the GridService portType defined by OGSI, guaranteeing that the OGSI-specific operations will be available. Although not part of WSDL 1.1, portType inheritance was deemed necessary by the working group so they not only made use of it in the specification but its importance was reflected in its introduction to the current WSDL 1.2 draft.

Service Data

In addition to operations, a portType of a Grid service instance may also include Service Data Declarations (SDDs), constructs describing the structure of the Service Data Elements (SDEs) that make up the instance's exposed state. The concept of SDDs and SDEs combined with a number of operations defined by OGSI provides a common mechanism to describe the structure of a Grid service instance's exposed state, to access or modify that state, and to use the state for notification purposes. For example, access to an SDE is made possible through appropriate get/set operations in the GridService portType. The inclusion of the SDDs and SDEs in the specification brings the idea of JavaBeans properties and C++/Java instance/class data members in Grid services.

The OGSI working group felt that the SDD and SDE concepts were important enough to the Grid's service-oriented architecture to adopt them in their specification, even though they are not part of any other standard. As with portType inheritance, the group is actively lobbying for the introduction of service data in the next version of WSDL.

> "OGSI encourages the development of fine-grained, component-based architectures while Web services promote a coarse-grained approach"

Grid Service Handles, References, and Locators

Each Grid service instance is identified by a URI, which is called the Grid Service Handle (GSH). However, a GSH cannot be used to communicate with a Grid service instance. Instead, it has to be resolved to a Grid Service Reference (GSR), which contains a binding-specific endpoint that consumers can use. More than one GSR may exist at the same time pointing to the same Grid service instance through different binding-specific endpoints (e.g., SOAP endpoint or CORBA IOR).

Unlike a GSH, a GSR may expire throughout the lifetime of a Grid service instance. If that happens, the consumer of the service will have to get one (or more) new GSR(s). A HandleResolver portType is defined to support resolutions of GSHs to GSRs. There can be a number of reasons a service provider may want the GSRs to its Grid service instances to expire. If, for example, service mobility was to be implemented, consumers would have to refresh their GSRs to point to the new location of the service. For quality of service, access to service instance may only be allowed through a particular network binding only at certain times.

A *locator* is a container of GSHs and GSRs that point to the same Grid service instance. It may also contain the qNames of the portTypes that are supported by the referred instance. Many of the operations defined in the OGSI portTypes return or accept Locator constructs as arguments.

Creating Grid Service Instances

The Factory pattern enables the dynamic creation of Grid service instances based on a Grid service description. OGSI specifies the operations and semantics of a factory through the Factory portType. However, a Grid service instance does not have to be created through a service that supports the Factory portType. For example, a hosting environment may create Grid service instances when it starts or as a result of an event.

The implementation of the Factory portType by a Grid service instance is not mandated by the specification.

Notification and Service Groups

The OGSI specification includes portTypes to support notification and service groups. The notification pattern allows interested parties to subscribe to service data elements and receive notification events when their value is modified. Service group portTypes allow registries of grid service instances to be formed. Service groups may denote some kind of relationship between the instances they contain, but that is not a requirement. For example, a service group may contain all the instances that were created by a particular user.

The implementation of the notification and service group portTypes by a Grid service instance is not mandated by the specification.

An Example

The discussion of OGSI so far has given us enough ground to explore an example. We are going to write the Grid service description of

a very simple workflow execution Grid service and explore the steps for creating and consuming instances of it.

The Grid service description of our workflow execution Grid service is shown in Listing 1. It supports only a simple operation, "executeWorkflow", plus all the operations in the GridService portType that the workflowExecutionPortType inherits. Note that the gwsdl:portType element is used instead of the wsdl:portType one. This is the temporary solution that OGSI adopted until WSDL 1.2 – with the "extends" attribute for the wsdl:portType element – is published.

The Grid service description shown in Listing 1 does not include "binding" or "service" elements. This is because the document is like a template for the construction of new Grid service instances. The WSDL of those instances will, of course, have to contain binding related information. The OGSI specification says nothing about specific bindings.

So, how do we use this service? First, we need to get access to an instance of it. The hosting environment may create one automatically for us and make it available through a registry, or there may be a factory Grid service instance that we may be able to use. The approach will depend on the semantics of our service. We may want to have a Grid service instance representing the workflow execution engine as a resource, in which case all consumers will have to share it. Alternatively, we may require each consumer to create a new instance that will represent that consumer's interaction with the workflow execution engine, in which case the instance's state may represent interaction-specific information.

Let's assume that there is a factory that can create instances of our data access Grid service. Figure 2 illustrates the necessary steps. We need to obtain a GSH to a factory instance (e.g., by looking into an appropriate registry, Steps 1 and 2). Then, we can resolve the GSH to one or more GSRs, giving us access to the operations of that factory (this step can be part of the query to the registry). If the request to create a new instance (Step 3) of the service is successful (Step 4), we will be given a GSH to the new instance (Step 5). Again, we will have to resolve the GSH to a GSR so we can access the instance. This may happen automatically as part of the create operation or we may have to contact an instance that implements the OGSI HandleResolver portType (the former is assumed in Figure 2).

Once we have a GSR to the newly created Grid service instance, we can invoke operations on it (Step 6). We can submit a BPEL script for execution or call the operations that are defined by the GridService portType. Of course, a more useful workflow execution Grid service would probably have operations that stopped and paused the execution of the workflow, queried its state, supported notification, and so on.

Now that we have an idea of how instances are created and consumed, we can introduce service data into our example. We could expose the status of the execution of the submitted workflow (see Listing 2).

The sd:serviceData element declares the existence of the "wf:status" SDE as part of an instance's state. The value of the SDE can be retrieved through appropriate operations defined in the Grid-Service portType.

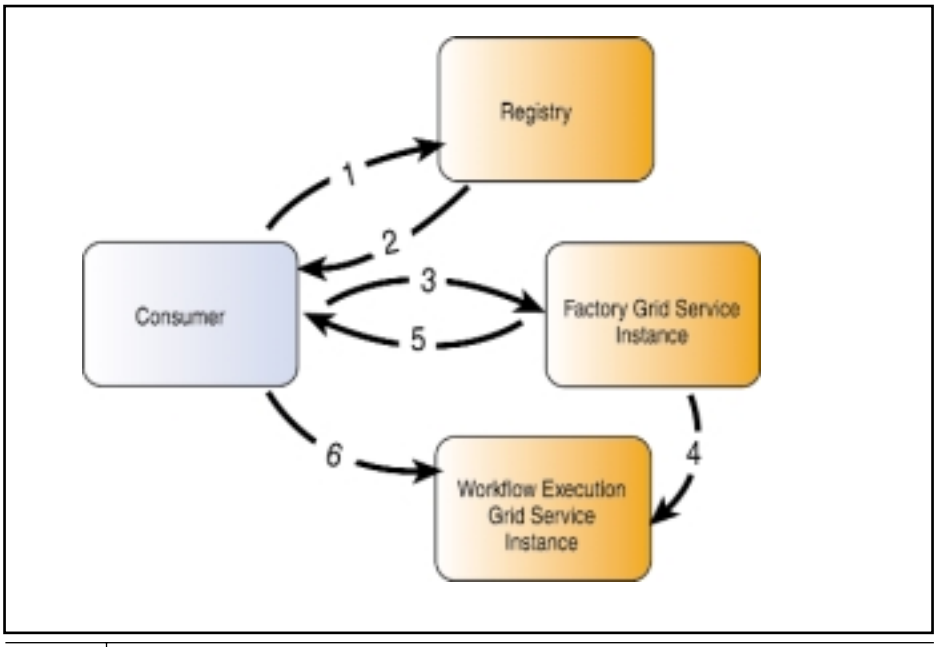


FIGURE 2 | **Steps to create and consume a workflow execution Grid service instance**

Comments on Grid Services

OGSA is the Grid community's effort to create a services-oriented platform for building large-scale distributed applications. OGSI constructs the foundations of the OGSA platform on top of Web services.

In making some of the characteristics it introduces a mandatory part of the infrastructure, OGSI has moved away from the vision of the Web services infrastructure as a collection of interoperable components that are built on top of widely accepted standards. Instead, OGSI brings a flavor of object orientation a la CORBA and J2EE into Grid services (e.g., service data and Grid service instances) and introduces noncompliant extensions to the WSDL standard (e.g., portType inheritance and service data).

It's my opinion that OGSI encourages the development of fine-grained, component-based architectures while Web services promote a coarse-grained approach. Due to its nonstandard features, like service data, mandatory statefulness, GSR, and factories, the OGSI specification deviates from common Web services practices. Then again, OGSI is an application of the Web services concept to the Grid application domain so experience may prove that the introduced characteristics are indeed required.

The Future

The OGSI working group has completed its work on version 1.0 of the specification. In the future, more effort will be focused on defining higher-level services for OGSA and, of course, building applications.

The Grid community is building its infrastructure on top of Web services technologies, making the Grid a big user of the emerging standards and an excellent evaluation platform for all the specifications and tools.

Acknowledgments

Thanks to Prof. Paul Watson, Dr. Jim Webber, OGSI working group, and OGSI-Primer working group.

Listing 1: Grid service description of the simple workflow execution grid service

```
<wsdl:definitions

xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"

xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions"

xmlns:wsdl="http://schemas.xmlsoap.org/wsdl"

xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns:bpel="http://schemas.xmlsoap.org/ws/2003/03/business-process"
   xmlns:wf="http://example.com/work-flow"

targetNamespace="http://example.com/workflow ">

   <wsdl:types/>

   <wsdl:message name="executeWorkflowMsg">
      <wsdl:part name="process" element="bpel:process"/>
   </wsdl:message>

   <gwsdl:portType name="workflowExecutionPortType"

extends="ogsi:GridService">
      <wsdl:operation name="executeWorkflow">
         <wsdl:input message="executeWorkflowMsg"/>
      </wsdl:operation>
   </gwsdl:portType>
</wsdl:definitions>
```

Listing 2: Grid service description with service data declarations of the simple workflow execution Grid service

```
<wsdl:definitions

xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"

xmlns:sd="http://www.gridforum.org/namespaces/2003/03/serviceData"

xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions"

xmlns:wsdl="http://schemas.xmlsoap.org/wsdl"

xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns:bpel="http://schemas.xmlsoap.org/ws/2003/03/business-process"
   xmlns:wf ="http://example.com/work-flow"

targetNamespace="http://example.com/workflow">

   <wsdl:types>
     <xsd:schema>
      <xsd:complexType name="status">
       <xsd:choice>
        <xsd:element name="running"/>
              <xsd:element name="paused"/>
        <xsd:element name="completed"/>
            </xsd:choice>
         </xsd:complexType>
     </xsd:schema>
   </wsdl:types>

   <wsdl:message name="executeWorkflowMsg">
      <wsdl:part name="process" element="bpel:process"/>
   </wsdl:message>

   <gwsdl:portType name="workflowExecutionPortType"

extends="ogsi:GridService">
      <wsdl:operation name="executeWorkflow">
         <wsdl:input message="executeWorkflowMsg"/>
      </wsdl:operation>

      <sd:serviceData name="status" type="wf:status"
        minOccurs="1" maxOccurs="1"

mutability="mutable"/>
   </gwsdl:portType>
</wsdl:definitions>
```

Download the code at sys-con.com/webservices

WebServices .NET J2EE XML JOURNAL

PRESIDENT AND CEO
Fuat A. Kircaali fuat@sys-con.com

VP, BUSINESS DEVELOPMENT
Grisha Davida grisha@sys-con.com

ADVERTISING
SENIOR VP, SALES & MARKETING
Carmen Gonzalez carmen@sys-con.com
VP, SALES & MARKETING
Miles Silverman miles@sys-con.com
ADVERTISING DIRECTOR
Robyn Forma robyn@sys-con.com
DIRECTOR, SALES & MARKETING
Megan Ring-Mussa megan@sys-con.com
ADVERTISING SALES MANAGER
Alisa Catalano alisa@sys-con.com
ASSOCIATE SALES MANAGERS
Carrie Gebert carrie@sys-con.com
Kristin Kuhnle kristin@sys-con.com

SYS-CON EVENTS
PRESIDENT, SYS-CON EVENTS
Grisha Davida grisha@sys-con.com
CONFERENCE MANAGER
Michael Lynch mike@sys-con.com

CUSTOMER RELATIONS / JDJ STORE
CIRCULATION SERVICE COORDINATORS
Niki Panagopoulos niki@sys-con.com
Shelia Dickerson shelia@sys-con.com
Edna Earle Russell edna@sys-con.com
JDJ STORE MANAGER
Rachel McGouran rachel@sys-con.com

SYS-CON.COM
VP, INFORMATION SYSTEMS
Robert Diamond robert@sys-con.com
WEB DESIGNERS
Stephen Kilmurray stephen@sys-con.com
Christopher Croce chris@sys-con.com
ONLINE EDITOR
Lin Goetz lin@sys-con.com

ACCOUNTING
FINANCIAL ANALYST
Joan LaRose joan@sys-con.com
ACCOUNTS RECEIVABLE
Kerri Von Achen kerri@sys-con.com
ACCOUNTS PAYABLE
Betty White betty@sys-con.com

SUBSCRIPTIONS
SUBSCRIBE@SYS-CON.COM
1-888-303-5282
FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS, PLEASE SEND YOUR LETTERS TO SUBSCRIPTION DEPARTMENT
COVER PRICE: $6.99/ISSUE
DOMESTIC: $69.99/YR (12 ISSUES)
CANADA/MEXICO: $89.99/YR
ALL OTHER COUNTRIES: $99.99/YR
(U.S. BANKS OR MONEY ORDERS)

WorldWide Newsstand Distribution
Curtis Circulation Company, New Milford, NJ

SYS-CON MEDIA

written by Rajesh Zade & Avinash Moharil

Building a Business Logic Layer OVER Multiple Web Services

Leveraging multiple Web services to build a truly distributed Web services architecture

Over the last few years, there have been significant developments in the Web services world. Many enterprises have embraced Web services to build business-to-business transactions and a uniform communication layer among applications over corporate intranets. This article discusses how to leverage application resident business logic by building a business logic layer over multiple Web services.

Many businesses are adopting Web services to gain access to applications and legacy databases that reside inside corporate networks (usually behind corporate firewalls). Web services have changed the B2B model from *centrally located, exchange-oriented B2B APIs* to *distributed, corporate network–resident APIs*. This model has gained huge popularity because it gives corporations complete control over the applications they run and also allows them to expose only those areas that are of interest to their business partners and those they deem appropriate to expose to the world. In the future it will be hard to imagine any businesses running all the applications internally without communicating with any other businesses or partners. Since Web services allow businesses to gain access to partial business logic and data residing within other businesses, the technology also opens up a whole new area for building a business logic layer that can operate over several other Web services in real time. The central idea of this article is how to build a logical layer serving as a Web service over other multiple Web services accessing remote applications and analyzing responses obtained from those Web services in real time. We use a simple case study based on JAX-RPC to build two Web services, "Timesheet" and "Insurance," and also build a logical layer, "Payroll," to evaluate responses from these two Web services. The "Payroll" client can easily be represented as another Web service that satisfies requests from such parties as your payroll processing company. We will also discuss EAI/BPM service integration techniques.

Web services, broadly speaking, are services offered via the Web. It is a platform-, vendor-, and language-independent, request-response model. In a typical Web services scenario, a business application sends a request to a service at a given URL using the SOAP protocol over HTTP. The service receives the request, processes it, and returns a response. The examples pre-

AUTHOR BIOS:

Rajesh Zade has over 14 years of experience in the computing field and has been working in various Java, eCommerce, and Web services technologies since 1996. Currently, he works as a chief technical architect for NetCliff, Inc. RAJESH.ZADE@NETCLIFF.COM.

Avinash Moharil has more than 14 years of experience in the computing field and has been working as a senior technology consultant and project manager in Silicon Valley. AVINASH.MOHARIL@NETCLIFF.COM.

sented here are based on Sun Microsystems' Java Web Services Developer Pack 1.1 (http://java.sun.com/Webservices/Webservicespack.html). You will need to download and install JWSDP-1.1 on your machine to run the examples. (The code is online at www.sys-con.com/webservices/sourcec.cfm. Please read readme.txt carefully before running these examples.)

> "Many enterprises have embraced Web services to build business-to-business transactions and a uniform communication layer among applications over corporate intranets"

Using Multiple Web Services Within One Container

Consumers and providers of Web services are typically businesses, making Web services predominantly business-to-business transactions. An enterprise can be a provider of Web services and also a consumer of other Web services. For example, an HR department could be in the consumer role of Time Tracking and Insurance Tracking services provided by outsourcing companies, and a provider of a Payroll service to payroll processing companies. To be a provider of a Web service, a provider service doesn't have to be a service that processes just local application logic or fetches local legacy resident databases, but can very well work as simply an adaptor to other consumer Web services.

TABLE 1: Applications as both provider & consumer

Industry	Use
Mortgage Processing	Loan applications can be processed based on services such as "credit history", "account information", "credit card information", and "debt information". Information obtained from these services can be evaluated in real time to screen prospective loan applicants.
Payroll Processing	Our case study demonstrates a payroll processing application that is based on "Time Tracking" and "Insurance" services.
Retail	Appropriate product prices can be distributed as a service that is based on other services such as "Product Information", "State and Local Taxes", "Shipping And Handling".
Government	Many government agencies such as IRS can process tax information based on several other services provided by federal and state government departments.

In Figure 1, many Web services are in the provider role (time tracking, insurance, and others), and one is in both the provider and consumer roles (payroll).

The Payroll service consumes the Time Tracking and Insurance service responses to produce a response requested by external businesses in the form of a Payroll service.

A number of business applications can utilize such a design approach (see Table 1).

Case Study: "Build Payroll Service"

Let's build a service layer as described above to produce payroll information. This service is based on the JAX-RPC API from Sun Microsystems. A similar service layer can also be built based on the JAXM API or by using a combination of two APIs. The "Payroll" service presented here is a client that utilizes the two consuming Web services, "Time Tracking" and "Insurance." It can also be built and published as a "producer" service that responds to a service request from a payroll-processing agency.

We will create the following components to build this service layer:

- ***Insurance Service:***
  - InsuranceIF: Insurance interface that defines the service methods
  - InsuranceImpl: Actual implementation of an InsuranceIF interface
- ***Time Tracking Service:***
  - TimesheetIF: Timesheet interface that defines the service methods
  - TimesheetImpl: Actual implementation of the TimesheetIF interface
- ***Payroll Service (Client):***
  - PayrollClient: Client that invokes the above services to generate final paycheck amount

These services are deployed on the Tomcat server. The project is built using the Ant build system. Each service code resides in its own folder and each folder has the following set of files to build and deploy services on the Tomcat server:

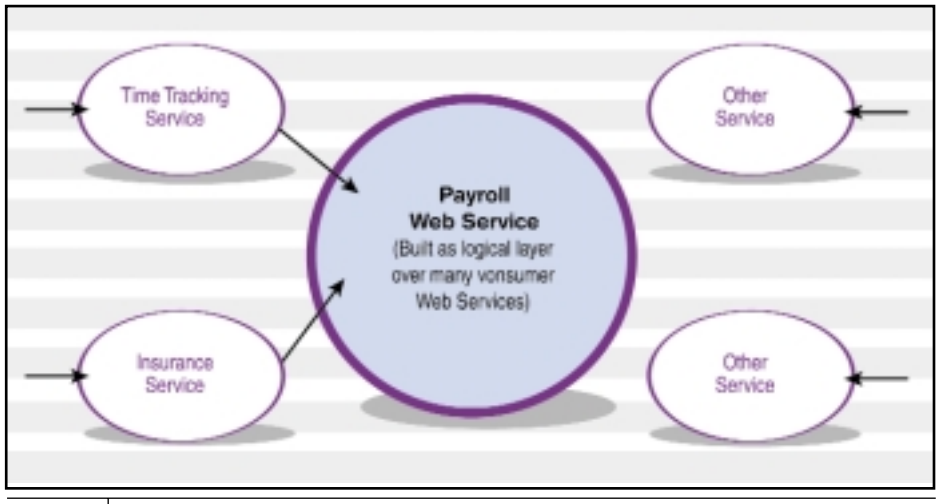


FIGURE 1 | **J2EE container environment**

- ***build.xml:*** Ant build for each service and client (uses targets.xml from "common" folder)
- ***config.xml:*** A configuration file read by the wscompile tool
- ***jaxprc-ri.xml:*** A configuration file read by the wsdeploy tool
- ***web.xml:*** A deployment descriptor for the Web component (a servlet) that dispatches to the service
- ***build.properties:*** Property file for the ant build system

Service Invocation Flow

When it receives a request for gross pay, the payroll service invokes calls to get the total hours and the rate information from the Timesheet service. Then it makes calls to the Insurance service to get health, life, and disability insurance premium information. Finally, it calculates the gross pay and returns it as a response to the calling service.

Let's discuss the Timesheet service further. We will not discuss the Insurance service since the implementation of both the services is very similar.

The Timesheet interface (see Listing 1) defines two methods, getHours(int empId) and getRate(int empId). Both methods take empId as a parameter and return hours billed for the current period and the consulting rate of the employee. A simple implementation of the service is described in Listing 2.

Listing 3 describes the configuration file for the Timesheet service needed for the wscompile tool and Listing 4 describes a configuration file read by the wsdeploy tool. The wscompile tool automatically generates the WSDL file when the service is compiled and deployed on the server. A complete listing of the WSDL file can be browsed when you visit the http://localhost:8080/time sheet-jaxrpc/timesheet and click the WSDL link on that page. JAX-RPC takes care of marshalling and unmarshalling the data returned by the service methods.

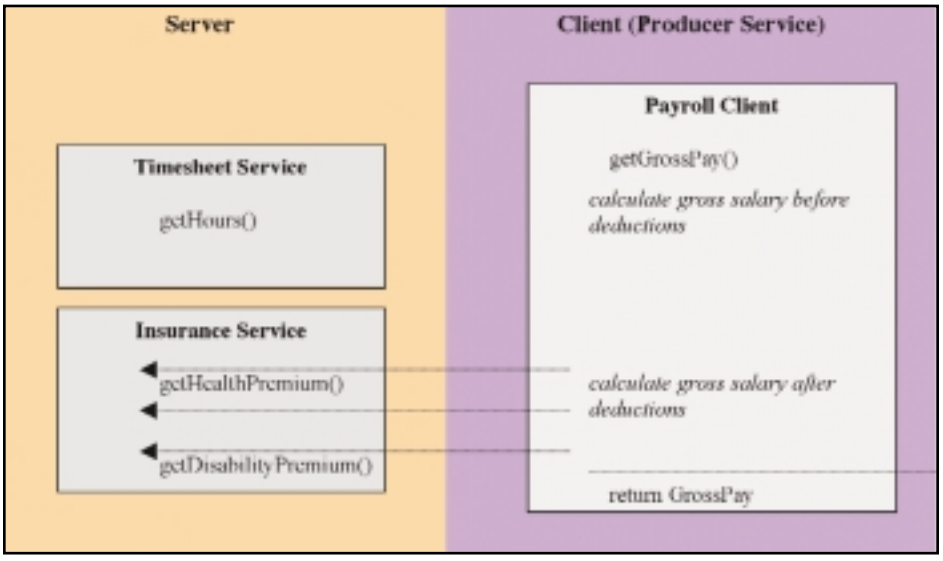


FIGURE 2 | **Service invocation flow**

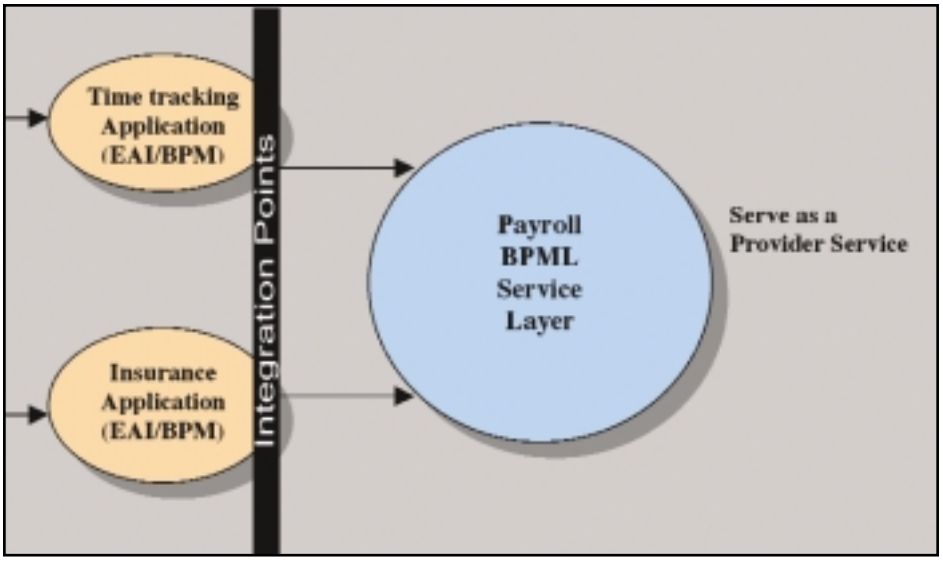


FIGURE 3 | **BPML environment**

> "Enterprises are exploring Web services as a viable option for EAI and BPM solutions to leverage their existing software assets"

Listing 5 describes the partial code of the payroll client that uses both Timesheet and Insurance services. As mentioned before, the payroll client can very well be built as a "Payroll" service that invokes Timesheet and Insurance services to compute employee salary.

EAI/BPM Techniques

Let's look at how we can use our business logic layer along with EAI (enterprise application integration) and BPM (business process management). We will look at these options briefly and then apply the concepts to our case study. Business Process Modeling Language (BPML) is the language of choice to implement complex business processes as declarative business logic.

Today's enterprises also consider Web services while implementing BPM and EAI. Enterprises are exploring new avenues to automate business processes and share business data across the enterprise, allowing departments to be more

Kenetiks

www.kenetiks.com

efficient and productive. There is a dire need for a full duplex, bidirectional solution to share business processes seamlessly and exchange data among ERP, CRM, SCM, databases, data warehouses, and other important internal systems within the company. Enterprises have been using EAI technologies that enable unrestricted data sharing throughout networked applications within the company. EAI is the process of creating an integrated infrastructure for linking disparate systems, applications, and data sources across the corporate enterprise. The implementation of EAI technology has been an expensive affair with proprietary interface implementations to interact with various applications in real time. With the nightmare of managing disparate applications, data, data transformations, and proprietary interfaces, corporations are hoping SOAP-based architecture of Web services will be a breather.

The emerging BPM breed of software provides analysis, design, automation, and optimization of business processes. They allow business managers and analysts to change the business process logic without changing the underlying IT programs. Many enterprises have applications where changing business processes involves engineers and programmers. BPM is the foundation of next-generation EAI. BPML is an XML-based business process definition language. The Business Process Management Initiative's (www.bpmi.org) mission is to promote and develop the use of BPM through the establishment of standards for process design, deployment, execution, maintenance, and optimization. BPMI.org develops open specifications, assists IT vendors in marketing their implementations, and supports businesses using BPM technologies. Enterprises are exploring Web services as a viable option for EAI and BPM solutions to leverage their existing software assets. The use of standard XML protocols makes Web services platform, language, and vendor independent, and an ideal candidate for use in EAI and BPM solutions.

Declarative programming techniques are used to capture the business logic of your application as business rules and process models in the transaction logic and process logic layers. Transaction rules can be compiled into reusable components and executed within the Transaction Logic Engine. Process models are executed within the Process Logic Engine.

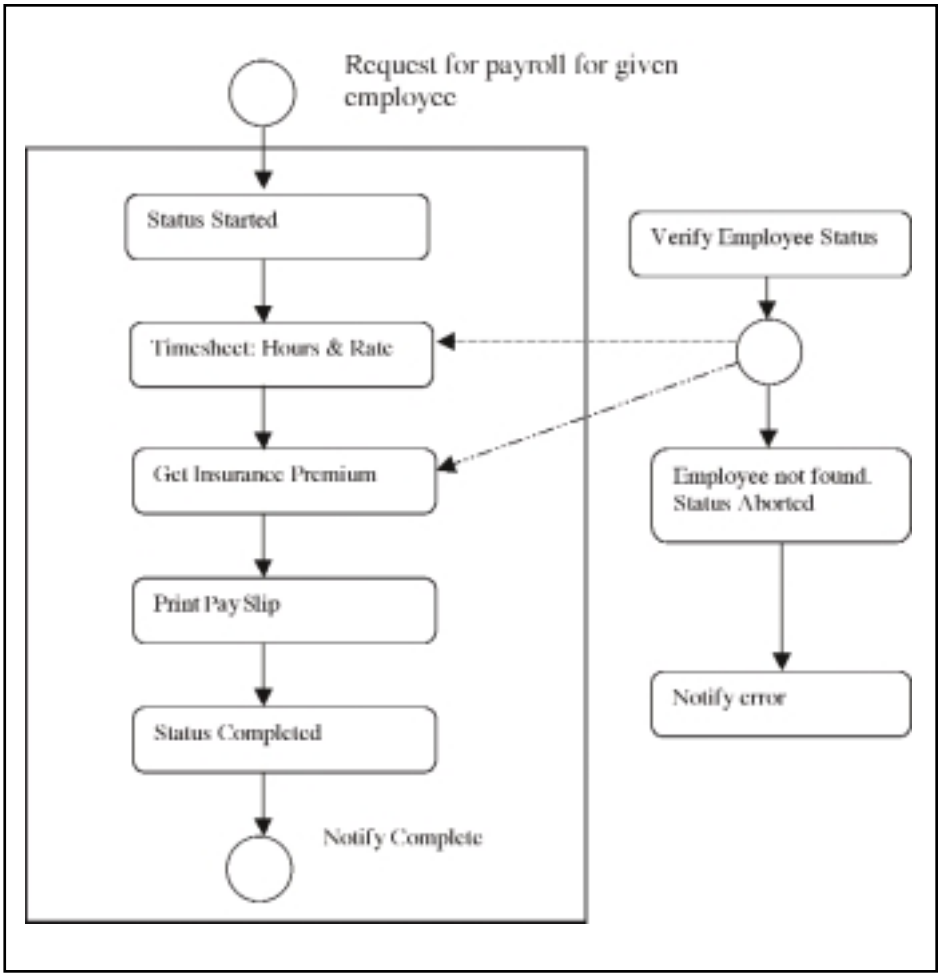


FIGURE 4 | **Process diagram**

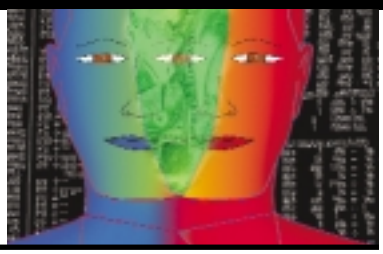
In this case study, we've implemented business process logic for payroll functionality. A business analyst can easily change the business parameters (integration points) regarding time tracking and insurance data extraction without changing the application programs to implement new business logic. Figure 3 is a re-representation of the case study from a BPML point of view. Instead of integrating with time tracking and insurance Web services, as in the case study, we are now integrating with enterprise products based on EAI or BPM. These products provide integration points that can hook into their applications. Once the data is obtained from these products, it can be utilized in real time by building a BPML layer. Figure 4 is a process diagram for our Payroll layer.

Conclusion

Web services as stand-alone processes in themselves are powerful interfaces to complete B2B transactions. By building a logical layer over multiple Web services, you can leverage application-resident business logic in real time and take Web services technology to another level. Web services can also be built over enterprise products based on EAI and BPM. Business Process Modeling Language (BPML) is an emerging language to build such a logical layer over enterprise applications.

Resources

- *W3C Web services architecture domain*: www.w3.org/2002/ws
- *JWSDP-1.1*: http://java.sun.com/Webservices/Web servicespack.html
- *The Business Process Management Initiative*: www.bpmi.org



Grid Services Extend Web Services

A solid foundation for service consumer reliability

We are often asked by people who are trying to understand the value Grid technology brings to Web services, "What is the significance of Grid services? They look like Web services." The answer to this question is superficially straightforward: mechanically, the differences between Grid services (as defined in the Open Grid Services Infrastructure [OGSI] V.1.0 specification) and Web services are few: a Grid service is simply a Web service that conforms to a particular set of conventions. For example, Grid services are defined in terms of standard WSDL (Web Services Definition Language) with minor extensions, and exploit standard Web service binding technologies such as SOAP (Simple Object Access Protocol) and WS-Security (Web Services Security). So yes, Grid services do look like Web services.

AUTHOR BIOS:

Considered one of the true visionaries in the distributed computing movement, Dr. Andrew Grimshaw is the founder and CTO of Avaki. As the chief designer and architect of both the Mentat and Legion projects, and the author of over 50 publications and book chapters, Dr. Grimshaw is a strong believer in open standards and industry-wide collaboration. He is a founding member of the Global Grid Forum, where he serves on the steering committee and works within the Open Grid Service Infrastructure Working Group. Dr. Grimshaw has spoken at hundreds of conferences and tradeshows worldwide.
AGRIMSHAW@AVAKI.COM

Steve Tuecke is the lead software architect in the Distributed Systems Laboratory (DSL) in the Mathematics and Computer Science Division at Argonne National Laboratory, working primarily on Grid research and development. From 1996 until 2001, he was also the manager of the DSL, responsible for building the organization from scratch into a world-class research group. Tuecke's DSL work includes being chief architect of the Globus Toolkit. He is also actively involved in standards work at the Global Grid Forum and Internet Engineering Task Force.
TUECKE@MCS.ANL.GOV

What is the significance of Grid services? Well, these Grid service conventions are not superficial in their function; they address fundamental issues in distributed computing relating to how to name, create, discover, monitor, and manage the lifetime of stateful services. More specifically, these conventions support:

- Named service instances and a two-level naming scheme that facilitates traditional distributed system transparencies (see sidebar)
- A base set of service capabilities, including rich discovery (reflection) facilities
- Explicitly stateful services with lifetime management

Before proceeding further, let's state that clearly it's possible, using standard Web services, to manage and name stateful services using ad hoc methods, e.g. extra characters placed in URLs or extra arguments to functions. Similarly, you can define an introspection service interface and publish it via WSDL. However, the point is not whether it can be done, but whether it is done in a uniform and consistent manner that everyone understands so that the full power of traditional distributed system naming and binding techniques can be brought to bear on the widest possible set of services. Grid services are important because they provide uniformity and consistency for vital distributed system functions.

Named Service Instances

At the heart of the Grid service specification is the notion of a named service instance. A *service instance* is named by a Grid Service Handle (GSH). The GSH is a classic abstract name – it is not necessarily possible to determine by examination of a GSH such things as the location, number, implementation, operational status (e.g., up, down, failed), and failure characteristics of the associated named object. GSHs by themselves are not useful. Instead they must be resolved into a Grid Service Reference (GSR). A GSR contains sufficient information to communicate with the named grid service instance. Thus, GSHs and GSRs form a two-level naming scheme in which GSHs serve as abstract names and GSRs provide the method and address of delivery. Let's first look at the benefits of having abstract names.

Why Abstract Names?

What is an abstract name? An *abstract name* refers to a "thing" or "things." In the case of Grid services, the "things" are *Grid service instances*, which may be stateless, i.e., pure functions that transform or map inputs to outputs in which the outputs are independent of the history of calls on the service instance; or stateful, in which case the output may depend on the history of calls on the service. The advantage of using abstract names rather than addresses is that it permits a level of indirection between the representation of the name and the code that actually implements the service.

Note that the form of the GSH is flexible

Merant

www.merant.com

Transparencies

Distributed systems research has a rich literature where virtualization of resources and objects is a common solution to many problems. This virtualization results in a "transparency." Nine of these show up again and again. They are used with respect to accessing a remote service or object. Usually the intent is that the programmer/user need not know or deal with something, but can if they want to. The transparencies are:

- ***Access:*** The mechanism used for a local procedure call is the same as for a remote call. Many have argued that this is a special case on location transparency.
- ***Location:*** The caller need not know where the object is located, California or Virginia – it makes no difference.
- ***Failure:*** If the object fails the caller is unaware. Somehow the requested service or function is performed, the object is restarted, or whatever is needed happens.
- ***Heterogeneity:*** Architecture and OS boundaries are invisible. However, objects cannot migrate without limitation. At a bare minimum communication with objects on other architectures requires no data coercion.
- ***Migration:*** The caller need not know whether an object has moved since they last communicated with it.
- ***Replication:*** Is there one object or many objects behind the name? The caller need not know or deal with coherence issues.
- ***Concurrency:*** Are there other concurrent users of an object? The caller need not be aware of them.
- ***Scaling:*** An increase or decrease in the number of servers requires no change in the software. Naturally, performance may vary.
- ***Behavioral:*** Is it live or not? Whether an actual object or a simulation of the object is used is irrelevant. For example, am I talking to a host object, or a virtual host object?

– syntactically, a GSH is represented as a URI, allowing for many different name schemes. This allows for greater flexibility in choosing how GSHs are resolved into GSRs. For example, you could choose to implement a GSH name space that uses DNS addresses as part of the GSH. However, such a scheme is not required, nor does the inclusion of a DNS address in a GSH tell you anything per se about the location of the named service instance.

Why Use a Two-level Name Scheme Instead of Addresses?

Recall that GSHs (names) resolve to GSRs (addresses). Because GSHs are abstract names, the mapping of GSHs to GSRs need not be static – it is possible that a given GSH may resolve to different GSRs over time. For example, a Grid service instance (named by a GSH) may migrate from location to location in response to load, anticipated failure, or to improve performance by colocating with a client or frequently used service. Furthermore, the mapping may not only vary over time but need not be one-to-one. Other mappings are possible: one-to-many where the same GSH resolves to different GSRs – e.g., for load balancing.

The point is that by utilizing a two-level naming scheme, Grid services enable flexible realization of the traditional distributed system transparencies (see the sidebar: Transparencies). This has been done in many distributed systems in the past, as well as in contemporary Grid systems such as Legion and Avaki.

> "The importance of SDEs in the Grid cannot be underestimated"

Base Set of Interfaces

The second important aspect of Grid services is that a Grid service must export (expose) a minimum set of mandatory interfaces and a minimum set of what are called *service data elements* (SDEs) – essentially service metadata and state. These mandatory services and data elements are interfaces that all GSs can be assumed to have. They include functions and data elements for lifetime management (e.g., set termination time) and functions to query and manipulate SDEs.

The importance of SDEs in the Grid cannot be underestimated. They provide the basic mechanism by which discovery and monitoring of Grid services is achieved, including interface discovery, discovery of other metadata that might be used in services such as scheduling (e.g., policy information), and even the current state of the service instance (e.g., current load).

SDEs that convey dynamic state merit particular attention, for they distinguish Grid services from what is typically found in Web services. While Web service technologies such as WSDL and UDDI (Universal Description, Discovery, and Integration) allow for introspection and discovery of static information such as service interfaces and associated policy, dynamic SDEs provide a much richer basis for discovery and monitoring. This dynamicity is crucial when building dynamic systems in which the set of service interactions may be impossible to know a priori (at configuration time), in which decisions about which services to use may depend heavily on dynamic service state, and in which monitoring of a service's dynamic state is necessary to respond to unpredictable events in the overall system. Service data exploits XML strengths, including XML Schemas for rich modeling of the state and extensible support for rich query languages such as XPath and XQuery for introspection on that modeled state, to provide for a consistent and powerful basis on which discovery and monitoring can be built across all Grid services.

By clearly defining a minimum mandatory set of interfaces that all Grid services must support, as well as optional interfaces that play core roles, OGSI specifications make it possible to write higher-level software services and applications that can make assumptions about both the services and the infrastructure of the Grid. This uniformity simplifies the definition and implementation of the various higher-level services that must be defined to construct a complete Open Grid Services Architecture (OGSA). In addition, by defining base service type interfaces, the OGSI sets the tone and

Macromedia

www.macromedia.com/go/certification1/

philosophy of both how the Grid will operate and how future extensions will be shaped.

> "Grid services are important because they provide uniformity and consistency for vital distributed system functions"

Instantiation and Lifetime Management

The final major extension is the explicit concept of state in Grid services, as described earlier. While Web services may have state, it's not explicit, nor are different discrete units of state (i.e., instances) named in a consistent manner. Once there is namable state the obvious issues are (1) how are new instances created?; and (2) how long do they last?

To address these two issues the Grid service specification defines factories and lifetime management services.

New Web services are instantiated in the case of Grid services by *factories*. The Factory PortType defines an operation that creates a new service instance and returns its GSH (handle) and its initial termination time.

TerminationTime is a powerful concept supported by the OGSI. A significant problem in distributed systems in general, and Grid systems in particular, is the issue of reclamation of resources associated with services in the event of failures (e.g., loss of network connectivity) or lack of interest by any relevant clients (e.g., a service is no longer referenced by any other active process or service).

TerminationTime addresses this problem by setting a time when a service will self-destruct unless kept alive by subsequent increases in its termination time. Thus, a service can be started, and the progenitor need not explicitly destroy the service. Instead, it will be terminated automatically.

Summary

The OGSI specification provides the foundation upon which a wide range of Grid services will be defined, built, and interconnected. Grid services marry important concepts from the Grid computing community with Web services. They extend basic Web services by defining a two-layer naming scheme that enables support for the conventional distributed system transparencies, by requiring a minimum set of functions and data elements that support discovery, and by introducing explicit service creation and lifetime management.

These extensions are more than syntactic sugar. They extend and enhance the capabilities of Web services by providing common, powerful mechanisms that service consumers can rely upon across all services, instead of the service-specific, often ad-hoc approaches typically employed in standard Web services.

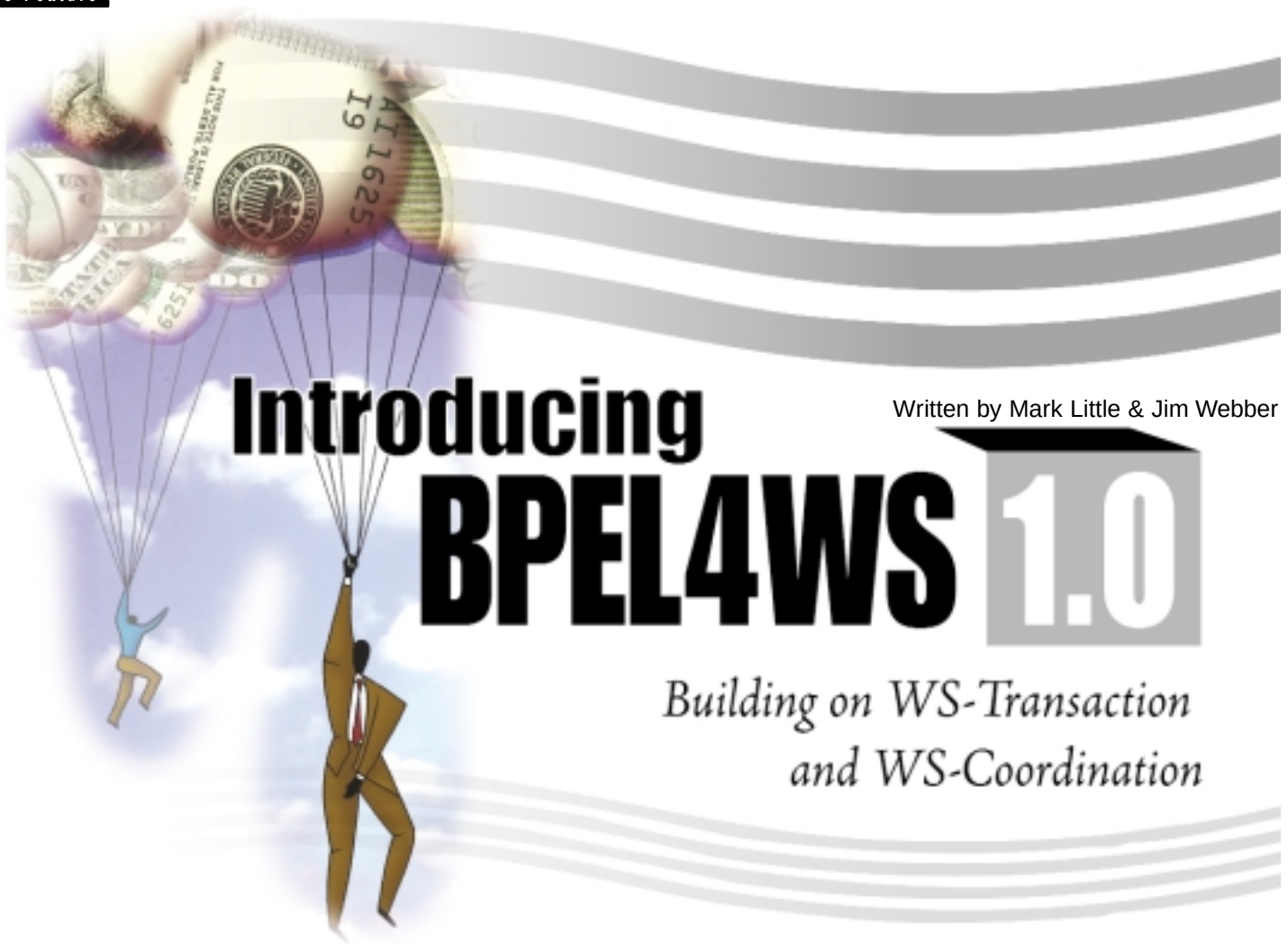
Acknowledgements

We'd like to thank Marty Humphrey (University of Virginia), David Snelling (Fujitsu), the OGSI Working Group for their contribution to establishing open standards for Grid technology, Ian Foster (Argonne National Laboratory), and Carl Kesselman (University of Southern California).

> "Grid services marry important concepts from the Grid computing community with Web services"

References

- Avaki Corporation: www.avaki.com
- Bal, H; Steiner, J., and Tanenbaum, A. "Programming Languages for Distributed Computing Systems." ACM Computing Surveys, pp. 261–322, vol. 21, no. 3, Sept., 1989.
- Chin, R.; and Chanson, S. "Distributed Object-Based Programming Systems," ACM Computing Surveys, pp. 91–127, vol. 23, no. 1, March, 1991.
- Foster, I.; Kesselman, C.; Nick, J.; and Tuecke, S. "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration." Open Grid Services Infrastructure WG, Global Grid Forum, June 22, 2002.
- Foster, I.; Kesselman, C; Nick, J. and Tuecke, S. "Grid Services for Distributed System Integration." *Computer*, 35(6), 2002.
- Grimshaw, A.S. "Enterprise-Wide Computing." *Science*, 256: 892–894, Aug. 12, 1994.
- Grimshaw, A.S., and Wulf, W. A. "The Legion Vision of a Worldwide Virtual Computer." *Communications of the ACM*, pp. 39–45, vol. 40, no. 1, January, 1997.
- S. Mullender, ed. *Distributed Systems*, ACM Press, 1989.
- Bayer, R.; Graham, R.M.; and Seegmuller, G. eds. (1979). *Operating Systems: An Advanced Course*: Springer-Verlag.
- Notkin, D., et al. "Heterogeneous Computing Environments: Report on the ACM SIGOPS Workshop on Accommodating Heterogeneity," *Communications of the ACM*, vol. 30, no. 2, pp. 132–140, February, 1987.
- Notkin, D., et al., "Interconnecting Heterogeneous Computer Systems," *Communications of the ACM*, vol. 31, no. 3, pp. 258–273, March, 1988.
- Stankovic, J.; Ramamritham, K.; and Kohler, W.H. "A Review of Current Research and Critical Issues in Distributed System Software," IEEE Distributed Processing Technical Committee Newsletter, pp. 14–47, vol. 7, no. 1, March, 1985.
- Tanenbaum, A.S.; and van Renesse, R. "Distributed Operating Systems," *ACM Computing Surveys*, pp. 419–470, vol. 17, no. 4, December, 1985.
- Tuecke, S.; Czajkowski, K; Foster, I.; Frey, J.; Graham, S.; Kesselman C.; Maquire, T.; Sandholm, T.; Snelling, D.; Vanderbilt, P.; Eds. "Open Grid Services Infrastructure v1.0", Global Grid Forum.



Introducing BPEL4WS 1.0

Building on WS-Transaction and WS-Coordination

Written by Mark Little & Jim Webber

In July 2002, BEA, IBM, and Microsoft released a trio of specifications designed to support business transactions over Web services. These specifications, BPEL4WS, WS-Transaction, and WS-Coordination (see *WSJ*, Vol. 3, issues 5–7), form the bedrock for reliably choreographing Web services–based applications, providing business process management, transactional integrity, and generic coordination facilities, respectively.

The value of BPEL4WS is that if a business is the sum of its processes, the orchestration and refinement of those processes is critical to an enterprise's continued viability in the marketplace. Those businesses whose processes are agile and flexible will be able to adapt rapidly to and exploit new market conditions. This article introduces the key features of Business Process Execution Language for Web Services, and shows how it builds on the features offered by WS-Coordination and WS-Transaction.

AUTHOR BIOS:

Dr. Jim Webber is an architect and Web services fanatic at Arjuna Technologies, where he works on Web services transactioning and Grid computing technology. Prior to joining Arjuna Technologies, he was the lead developer with Hewlett-Packard working on their BTP-based Web Services Transactions product – the industry's first Web Services Transaction solution. An active speaker and Web Services proponent, Jim is the coauthor of Developing Enterprise Web Services. *JIM.WEBBER@ARJUNA.COM*

Dr. Mark Little is chief architect, transactions, for Arjuna Technologies Limited, a spin-off from Hewlett-Packard that develops transaction technologies for J2EE and Web services. Previously, Mark was a distinguished engineer and architect at HP Middleware, where he led the transactions teams. He is a member of the expert group for JSR 95 and JSR 117, is the specification lead for JSR 156, and is active on the OTS Revision Task Force and the OASIS Business Transactions Protocol specification. Mark is the coauthor of J2EE 1.4 Bible *and* Java Transactions for Architects. *MARK.LITTLE@ARJUNA.COM*

The BPEL4WS Stack

The BPEL4WS model is built on a number of layers, each one building on the facilities of the previous. Figure 1 shows the fundamental components of the BPEL4WS architecture, which consists of the following:

- A means of capturing enterprise interdependencies with partners and associated service links
- Message correlation layer that ties together messages and specific workflow instances
- State management features to maintain, update, and interrogate parts of process state as a workflow progresses
- Scopes where individual activities (workflow stages) are composed to form actual algorithmic workflows

We'll explore the features of this stack, starting with the static aspects of the application – capturing the relationship between the Web services participating in workflows – and on to the creation of workflows using the BPEL4WS activities.

> The value of BPEL4WS is that if a business is the sum of its processes, the orchestration and refinement of those processes is critical to an enterprise's continued viability in the marketplace

Mapping Interenterprise Relations

To create workflows that span enterprises, we must understand how those enterprises are related. BPEL4WS provides a means of capturing the roles played by business partners in a Web services–based workflow through service linking, partners, and service references.

Figure 2 shows the relationship between service links, partners, and service references. *Service links* are the most abstract relationship supported in BPEL4WS, and link two parties by specifying the roles of each party and the (abstract) interface that each provides. serviceLinkType definitions can either be part of a service's WSDL interface, or defined separately and referenced by the WSDL. Embedding this definition directly in a WSDL description leverages WSDL's extensibility mechanism, allowing serviceLinkType elements to become a direct child of the wsdl:definitions element.

The actual content of a serviceLinkType is straightforward. It usually defines a link between two services, qualified by the targetNamespace of the WSDL document; and then exposes that relationship as two roles. In some cases, a serviceLinkType may specify a single role, which indicates that the workflow is willing to bind to any other service, without placing any requirements on that service.

In Listing 1 (the code for this article can be found online at www.sys-con.com/webservices/sourcec.cfm), two sample serviceLinkType elements are defined. The first defines a link between a WidgetSeller and a WidgetBuyer service. When a WidgetBuyerSellerLinkType is used in a workflow, it will implicitly associate a WidgetSellerPortType with a WidgetBuyerPortType, and enforce the appropriate operation and message constraints. The second defines an EnquiryLinkType that is used to model the link between the widget manufacturer and a third party making widget-related enquiries. Note that in this case, there is only one role specified, WidgetAuthority, which indicates that the widget manufacturing service is willing to link to any other service without placing any further constraints on the interface exposed by that service.

A BPEL4WS partner refines a serviceLinkType declaration by defining the roles played by actual partners at the endpoints of the relationship. A partner is declared within the workflow script because it forms part of the behavior of that workflow. Partnerships only make sense within the scope of the workflow where business partners interact. A sample partner declaration for a user authentication system is presented in Listing 2.

Inside the partners element we have individual partner declarations that specify the role of our enterprise and its partners on a per-serviceLinkType basis. Of the two partners defined in Listing 2, a customer specifies roles for both ends of the corresponding serviceLinkType declaration, in preparation for the bilateral exchanges that purchasing widgets necessitates. However, while enquiring about widgets the manufacturer is not fussy about who binds to and uses it, and so the partner declaration is unilateral, specifying only the myRole attribute as WidgetAuthority.

The final step in cementing our business interrelationships is to specify the network locations of our partners so that we can discover and consume their Web services.

Of course, physical network address locations change over time (and indeed sometimes change very rapidly over

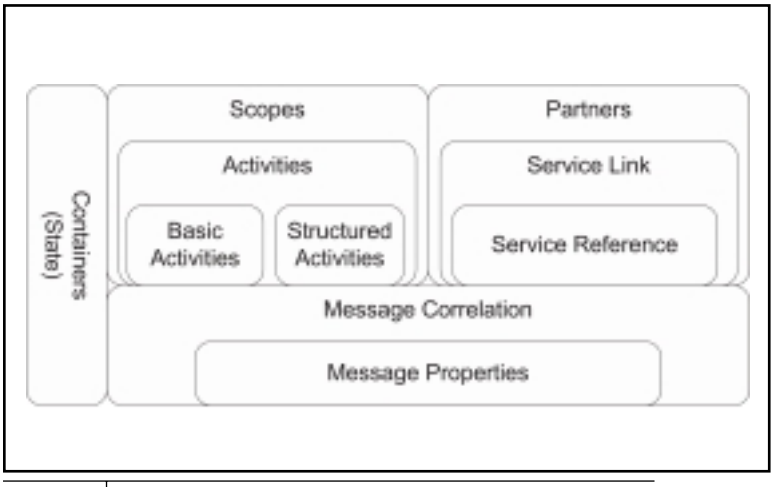


FIGURE 1 | **BPEL4WS Logical View**

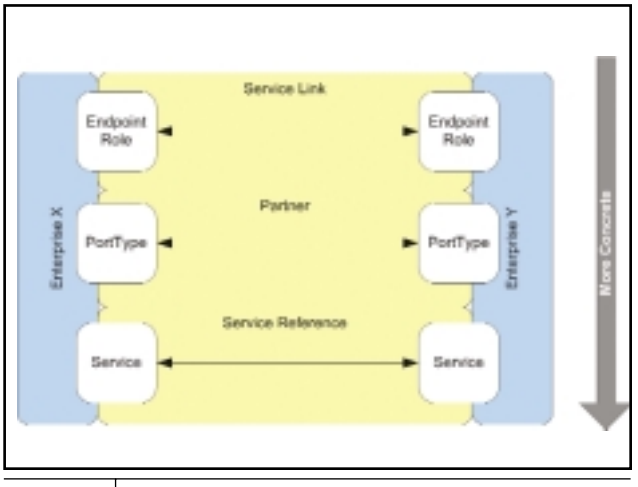


FIGURE 2 | **Abstract and concrete enterprise relationships**

time), and WSDL has a means of supporting this through a separation of portType (abstract network interface) and port (physical, protocol-bound interface on the network), which are mapped through bindings and later exposed as services. The service consumer must understand the portType section of a WSDL document before it can consume a service, though the binding to an actual port can be delayed right up until that client needs to invoke that service at runtime. The information needed to create the runtime binding can be accessed in a number of ways, including out-of-band communication between users and directory services like UDDI. The point is, given the distinction between abstract and concrete in WSDL, BPEL4WS needs a means of bridging the same gap between abstract partner declarations and exchanging messages over the network with real services at runtime. This is addressed by ServiceReference elements, which are part of a workflow that acts as typed references to a specific service. ServiceReferences allow consuming services to bind abstractly defined partners to physical network endpoints, and expose those endpoints (along with other useful data) to workflow activities.

Listing 3 shows a minimal ServiceReference declaration where the service provided by a particular partner is statically embedded. In this case, the wsdl:service element defined in a service's WSDL interface is used to create a "Web pointer" that can be used within the activities of a single workflow and passed amongst collaborating services as part of their message exchanges.

However, the real potency of ServiceReference comes to light when we dynamically compute or discover the endpoint or business process instance that we want to communicate with. We can thus augment the minimal ServiceReference shown in Listing 3 with specific instance information such as the ws:existingCustomer shown in Listing 4.

The ServiceReference shown in Listing 4 has additional information held by property elements within the referenceProperties element that identifies a specific resource hosted by a service. In BPEL4WS, that resource is likely to be an instance of a workflow. However, it may be a process or object identifier, or other identifier that has significance to both ends of the interaction. It is important to understand that while the computational aspects of BPEL4WS provide the ability to be able to obtain and utilize such properties, BPEL4WS does not place any semantics on them.

Message Properties and Property Aliases

Once we've captured the relationships between our enterprise and its partners, we can begin to exchange messages through the conduits defined by those relationships. Whether we are dealing with an invoice or a dispatch note, there is often a field or set of fields within that note that can be used to unambiguously differentiate that note from piles of other similar looking ones. For instance, an invoice number is usually used in correspondence rather than the date and address of the sender of the invoice since it is both simpler and more likely to resolve to a unique result. This notion of "distinguished" data is supported through message properties. Put simply, a message property is a unique name (within the workflow) that has a specific type from XML Schema (e.g., xs:postive Integer) and whose name has significance to the workflow (e.g., invoice number; see Listing 5).

Having a friendly name and type information for our property is akin to having object references in traditional programming languages. However, just like object references need to be bound to objects before we can use them, we need to bind properties to values before workflow scripts can access those values. In BPEL4WS we have a way of binding typed friendly names to values that we can use within our workflows – *property aliases*. A property alias binds the value of a property to the value of an element in a message using an XPath query. For instance, we may be interested in the invoice number from a purchase order and want to expose that value to the workflow.

Listing 6 shows you how to bind properties to values through propertyAlias declarations. The attributes in the element declare the property name that we are binding to (InvoiceNumber), the message (PurchaseOrderMessage), and the specific message part (invoice) where the value that we wish to bind to is located. The final step to complete the binding is to specify the XPath query (specified in the query attribute) that returns the value from the specified message part. In Listing 6 this is calculated by the expression/invoice number, which evaluates the contents of the first invoice-number element from the root context, where context is provided by the preceding messageType and part attributes. Now when PurchaseOrderMessage messages are processed, the property InvoiceNumber will be assigned the value of the corresponding invoice - number in the message, or conversely may be used to assign such a value to the invoice-number element, just like an object reference.

Once properties have been defined, they can be used to correlate messages. Using a property like an invoice number allows the underlying BPEL4WS implementation to route messages to particular workflow instances at the application level without relying on sophisticated conversational transports to manage

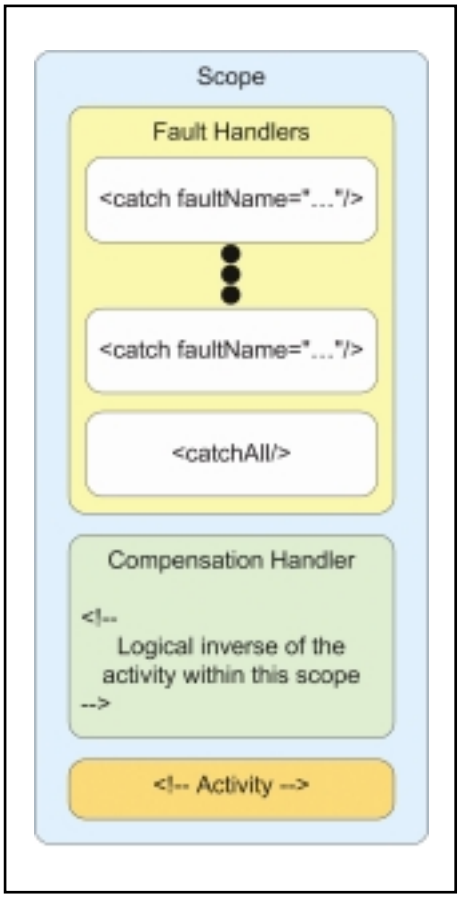


FIGURE 3 **BPEL4WS scope Structure**

that mapping. As we shall see, the BPEL4WS communication activities encapsulate this behavior to further simplify matters

Containers and Data Handling

In dealing with Web services–based workflows we encounter a significant philosophical difference between the two technologies: workflows are inherently stateful applications, whereas Web services are inherently stateless. Of course many Web services do actually maintain state between invocations, but do so in a proprietary manner in databases, files, statically allocated program variables, and so on, all of which requires programmer effort and is likely to be inaccessible to the business analyst. BPEL4WS has abstracted these proprietary approaches and replaced them with a generic state management facility based on containers.

A BPEL4WS container is a typed data structure that stores messages associated with a workflow instance. The underlying notion of containers is that in a workflow the state of the application is simply a function of the messages that have been exchanged. Containers begin their lives uninitialized, and are populated over time by the arrival of messages, or computations being executed that populate them.

Listing 7 shows a simple container declaration that could be used by a cable operator to store requests for package updates. This container is used for holding incoming channel addition requests from customers while our workflow process goes about the business of updating the customer's billing details and set-top box software accordingly.

Declaring a container is straightforward. It consists of a name for the container that is unique within the scope of the workflow process definition, and the type of the message as defined in the corresponding WSDL document. For this example the WSDL interface identified by the coprefix resolves to http://cableoperator.example.org/wsdl which is the address at which our fictitious cable operator resides.

Generally, the messages stored in containers are the same messages that are exchanged with partners. However, there is nothing to prevent a programmer from concocting a message type purely to store local variables during computations. Such messages are never exchanged with partners, and are usually declared in-line with their associated containers (see Listing 8).

Communication Activities

BPEL4WS defines a set of communication activities that deal with the sending and receiving of messages so that a workflow process instance can communicate with partners' Web services. BPEL4WS provides three activities – invoke, receive, and reply – each of which handles a different type of interaction between partners in a workflow.

The invoke activity allows a workflow instance to call a synchronous or asynchronous operation on a remote Web service. An asynchronous one-way operation is the simplest form of invoke since it only requires a single input container to send messages. Look at the example shown in Listing 9 where a request for additional cable TV channels is sent from a set-top box to a cable operator's system. The invoke activity calls the addChannel operation from the ChannelManagementPortType portType exposed by its CableOperator partner, sending a message from the Re-quested Channels container to request additions to the subscriber's package.

Web service operations are exposed to the outside world by a receive activity. The receive activity is the workflow entity that a WSDL operation maps onto. In Listing 10 we show the receive activity that the cable operator exposes as its addChannel operation (the operation invoked by customers in Listing 9). The cable operator uses a container called addChannelOrders to hold incoming AddChannelMessage messages from customers. When the receive activity is activated by the arrival of an AddChannelMessage from a customer, a new instance of the cable operator's channel adding workflow is created and executed.

A receive activity is blocking, which does not allow the workflow graph it precedes to progress until the messages it requires have been received. Such activities may be used to trigger the creation of a new workflow instance in response to an incoming message, or they may deal with the message delivery to an existing instance.

A reply activity sends synchronous responses to messages received through a receive activity. Correlation between a receive and a reply is handled by the underlying BPEL4WS implementation.

Listing 11 shows an example reply activity, where a message from CustomerSubscriptionDetails is sent back to a customer partner in response to a prior incoming message to a receive activity.

Activities

In order to execute a process, we need a means of describing its behavior. We have to understand the features that the workflow language provides to manipulate data, iter-

BPEL4WS 1.1 and OASIS WSBPEL

The original BPEL4WS 1.0 specification that we considered in this article has been superceded as part of the original vendors' efforts to standardize the technology. IBM, Microsoft, BEA, and their partners have submitted a version 1.1 BPEL to OASIS under the WSBPEL (Web Services Business Process Execution Language) Technical Committee. The most obvious changes in BPEL4WS 1.1 are that the term "container" has been replaced with the more traditional term "variable," although its type is still considered in terms of messages. These variables are now supported at arbitrary scope, unlike BPEL4WS, which only supported containers at the global process scope.

In addition to variables, the specification authors have added event handlers into the activity set by introducing the <eventHandlers> activity. An eventHandlers activity is similar to a pick activity insofar as it contains a number of onMessage or onAlarm activities, but it differs from the standard pick activity in that an eventHandler can be executed concurrently with the currently running scope. This allows concurrent processing within a single scope where previously concurrent "threads" of control were not permitted. Of course, there are some standard caveats with the use of an eventHandler, like the fact that one cannot be used to call a compensate activity, but these are minor and will easily be handled by tool support.

ate, call external functions, and so on; and how to compose these primitives into meaningful workflows. To support this, the BPEL4WS specification defines a number of fundamental activities that are the basic building blocks of the workflow. It is beyond the scope of this article to look at every facet of each language construct defined by BPEL4WS, but we will give you an idea of what kinds of things are possible.

BPEL4WS control flow activities are responsible for serializing and parallelizing activities, choosing from alternative paths in a workflow, iterating commands, and so on. The simplest construct is the sequence activity that executes subactivities serially, as shown in Listing 12.

Parallelizing activities that have no dependencies is achieved by enclosing the parallel activities within a flow element. For example, a customer's computing system, which initiated a hotel reservation, may also have been organizing flights and car rentals simultaneously. If we assume these activities are independent, we can execute them in parallel with a flow activity like that shown in Listing 13.

A *scope* is a means of explicitly packaging activities together so that they can share common error handling and compensation routines. The full structure for a scope is shown in Figure 3 and consists of a set of optional fault handlers, a single optional compensation handler, and the primary activity of the scope, which defines its behavior.

In the absence of a scope declaration, each activity is implicitly associated with its own scope with the same name as, and delimited by, the activity. An example scope that captures the booking process for a ticket is shown in Listing 14.

The normal behavior for the scope shown in Listing 14 is for the booking activity near the bottom of the example to be executed and for flight tickets to be reserved. However, this scope declares a number of exception handlers with catch activities that allow a variety of faults that might occur while booking tickets to be rectified before they cause further problems. For instance, these catch activities deal with such matters as a flight number being incorrectly specified, a flight already being fully booked, or a fault in the payment method used to purchase the tickets. We can assume here that these fault handlers are able to correct any problems so that the scope can complete normally. The catchAll handler is a little different in that it handles any faults other than those that are explicitly handled by the preceding catch activities. Since the nature of the fault is unknown, the designer of this scope has decided that the safest thing to do is to compensate the inner scopes by calling the logic held in their compensationHandler activities to restore the system to the same state (or a logically equivalent state) as it was before the top-level scope executed. The skeleton for a compensationHandler is shown in Listing 15.

> "The ultimate goal of business process languages like BPEL4WS is to abstract underlying Web services"

Compensation handlers are a fundamental component of BPEL4WS workflows to support reliable long-lived business processes. During the execution of a workflow, data in the various systems that the workflow encompasses changes. Since we have no knowledge of the underlying computing systems (databases, queues, etc.) that the workflow is utilizing, we must compensate at the application level by performing the logical reverse of each scope that was executed as part of our workflow, from the most recently executed scope back to the earliest executed scope.

Where fault handlers provide alternative forward execution paths through a scope, compensation handlers, when invoked, undo the work performed by a scope. Since a compensationHandler for a specific scope reverses that scope's work, the handler can potentially be as complex and intricate as the scope's normal original activity.

A compensationHandler can also be set to compensate an entire business process after its normal completion (instead of individual scopes).

As we saw in an earlier article in this series, the BPEL4WS specification suggests WS-Transaction as the protocol of choice for coordinating distributed transactions across workflow instances. Thus, when a scope containing invocations on a partner's Web services is compensated, the underlying BPEL4WS engine should ensure that the appropriate WS-Transaction messages are sent to the transaction coordinator so that any partner's systems can be informed of the need to compensate the invoked activities.

Summary

BPEL4WS is at the top of the WS-Transaction stack and utilizes WS-Transaction to ensure reliable execution of business processes over multiple workflows, which BPEL4WS logically divides into two distinct aspects. The first is a process description language with support for performing computation, synchronous and asynchronous operation invocations, control-flow patterns, structured error handling, and saga-based long-running business transactions. The second is an infrastructure layer that builds on WSDL to capture the relationships between enterprises and processes within a Web services–based environment.

Taken together, these two aspects support the orchestration of Web services in a business process, where the infrastructure layer exposes Web services to the process layer, which then drives that Web services infrastructure as part of its workflow activities.

The ultimate goal of business process languages like BPEL4WS is to abstract underlying Web services so that the business process language effectively becomes the Web services API. While such an abstract language may not be suitable for every possible Web services–based scenario it will certainly be useful for many, and if tool support evolves it will be able to deliver on its ambition to provide a business analyst–friendly interface to choreographing enterprise systems.

Written by Aravilli Srinivasa Rao

Web Services Made Easy with Ruby

A simple development method

It's easy to develop Web services using Ruby. This article looks at how to develop a Web service client to access the Web services that are hosted in the Internet and how to develop a Web service with simple steps using Ruby.

An Overview

Ruby is the interpreted scripting language invented by Yukihiro Matsumoto for quick and easy object-oriented programming and is more popular in Japan than in other countries. Ruby's open-source nature makes it free for anyone to use. All data structures in Ruby are objects, but you can add methods to a class or instance of a class during runtime. Because of this, any instance class can behave differently from other instances of the same class. Ruby's dynamic typing nature doesn't require explicit declaration of variable types. It features a true mark-and-sweep garbage collector that cleans all Ruby objects without needing to maintain a reference count. (See www.ruby-lang.org for the complete features of Ruby.) The following libraries are developed with Ruby:

- ***Ruby/DBI:*** To access different databases
- ***Ruby/LDAP:*** To search and operate on entries in the LDAP
- ***XSLT4R:*** For XSL transformation
- ***XMLRPC4R:*** For writing and accessing Web services
- ***SOAP4R:*** For writing and accessing Web services

SOAP Programming Using Ruby

SOAP4R is the implementation of Simple Object Access Protocol for Ruby developed by Hiroshi Nakamura. SOAP4R depends on the XML processor, http-access, logging, and date-time packages. (To install SOAP4R and its dependencies refer to its home page at www.jin.gr.jp/~nahi/Ruby/SOAP4R.)

Develop SOAP4R clients

The SOAP::Driver class provides support for writing SOAP client applications using Ruby. I'll explain how to use this class in order to develop the SOAP client. To invoke a SOAP Service the client should know the URL of the SOAP Service, namespace of the service methods, and the names of the service methods and their parameters. With this information let's develop a SOAP client in Ruby.

I've chosen a "SendEmail" Web service that is listed in Xmethods' Web services list. The client we're developing invokes a service named SendEmail to send an e-mail message to any mail address (see Listing 1).

In the first line I load the feature "soap/driver", which implements the SOAP::Driver class. Then I define four constants like NAMESPACE, URL, SOAPACTION, and HTTP_PROXY, which are required to create an instance of SOAP::Driver class by calling its new method.

```
driver  =  SOAP::Driver.new(log, logId,
namespace , endpoint, httpProxy,
soapAction )
Where
namespace = namespace to use for RPC
calls
endpoint = URL of the SOAP service
httpproxy = to specify the proxy server,
instead of direct connection to SOAP
server
soapaction = value for SOAPAction field
of the SOAP Header
```

After creating an instance of the driver class, call the addMethod method passing the name of the SOAP service method and its parameters' names. SOAP method parameters are often not taken into account on the SOAP server side but it is important to know the order of the parameters that were passed to the service.

The SOAP::Driver class has two methods to add service methods:

```
SOAP::Driver.addMethod(name, *paramArg)
SOAP::Driver.addMethodWithSOAPAction(name,
soapaction, *paramArg)
Where name = the name of the remote pro-
cedure
          soapaction = soap action header
to be used instead of the default one
specified on SOAPDrivers new   method
                paramArg = to specify
the names and modes of the remote proce-
dures parameters. The parameter modes are
input parameter, output parameter or in
out parameter.
```

After adding the service method to the Driver class, invoke it by passing the parameters with values.

```
SOAPDriver::ServiceMethod(
*paramvaluesArg)
```

Where ServiceMethod is the name of the remote procedure, that was added using addMethod API.

The Web services client is now ready for testing. Test the client using the following command:

```
ruby
```

Author Bio

Aravilli Srinivasa Rao, a software analyst at Hewlett-Packard, is technical lead for the development of HP's public UDDI Registry. He has worked with HP's e-speak and Total-e-Server (J2EE Application Server) products and is currently involved in the design and development of the projects in the mobile space to implement Web services.
SRINAVASA.RAO.ARAVILLI@HP.COM

introductory subscription offer!

A TRULY INDEPENDENT VOICE IN THE WORLD OF .NET

.NET Developer's Journal is the leading independent monthly publication targeted at .NET developers, particularly advanced developers. It brings .NET developers everything they need to know in order to create great software.

Published monthly, ***.NET Developer's Journal*** covers everything of interest to developers working with Microsoft .NET technologies – all from a completely independent and nonbiased perspective. Articles are carefully selected for their prime technical content – technical details aren't watered down with lots of needless opinion and commentary. Apart from the technical content, expert analysts and software industry commentators keep developers and their managers abreast of the business forces influencing .NET's rapid development.

Wholly independent of both Microsoft Corporation and the other main players now shaping the course of .NET and Web services, ***.NET Developer's Journal*** represents a **constant, neutral, expert voice** on the state of .NET today – the good, the bad, and the ugly…no exceptions.

SYS-CON MEDIA

SUBSCRIBE ONLINE!

www.sys-con.com/dotnet/

or Call

1 888 303-5282

Here's what you'll find in every issue of .netdj:

- Security Watch
- Mobile .NET
- .NET Trends
- Tech Tips
- Standards Watch
- Business Alerts
- .NET News
- Book and Software Announcements

.NET Developer's Journal is for .NET developers of all levels, especially those "in the trenches" creating .NET code on a daily basis:

- For beginners:
  Each issue contains step-by-step tutorials.
- For intermediate developers:
  There are more advanced articles.
- For advanced .NET developers:
  In-depth technical articles and columns written by acknowledged .NET experts.

Regardless of their experience level, *.NET Developer's Journal* assumes that everyone reading it shares a common desire to understand as much about .NET – and the business forces shaping it – as possible. Our aim is to help bring our reader-developers closer and closer to that goal with each and every new issue!

SAVE 16% OFF

THE ANNUAL COVER PRICE

Get 12 issues of .NETDJ for only $69.99!

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

ANNUAL COVER PRICE:

~~$83.88~~

YOU PAY

$69.99

YOU SAVE

$13.89

OFF THE ANNUAL COVER PRICE

SendEmail.rb

Just three steps are needed to develop a Web services client using SOAP4R.

Develop SOAP4R Services

SOAP4R supports the following servers:

- CGI/Fast CGI-based servers (SOAP::CGI Stub)
- Stand-alone server (SOAP::Standalone Server)

Now let's look at how to develop services using the stand-alone server. I'll develop a temperature conversion service that converts the temperature from Celsius to Fahrenheit (see Listing 2).

Develop a class called Temperature_Conversion_Server and inherit from SOAP::StandaloneServer, which inherits from SOAP::Server.

```
class Temperature_Conversion_Server  <
SOAP::StandaloneServer
# declare methods
# define methods
end
```

Declare your service methods by overriding the methodDef method in the StandaloneServer. For example, convert_to_ Fahrenheit, which takes degrees as input parameters:

```
  def methodDef
addMethod(self, 'convert_to_Fahrenheit',
'degrees' )
end
```

The methodDef method is implicitly called from SOAP::Server's initialize method, which exposes one of the following two service methods.

```
Class Server < Devel::Application
def addMethod( receiver, methodName,
*paramArg )
end

def addMethodWithNS( namespace, receiver,
methodName, *paramArg )
end
end
Where
Receiver = the object which contains the
methodName method.  Use "self" if the
service method is in the same class
methodName =  the name of the method that
is called through an RPC request
paramArg = to specify the parameter names
and parameter modes
namespace = in order to specify the name-
space for the method. Use this if you
want different namespace for each method.
```

After you declare the service method, define it as:

```
def convert_to_Fahrenheit( degrees )
 temp = (9*degrees + 160)/5
  return temp
end
```

The StandaloneServer constructor looks like:

```
class StandaloneServer < Server
def initialize( appName, namespace, host
= "127.0.0.1", port = 8080 )
end
appName – to specify the applications log
Id paramters
namespace - Namespace parameters apply to
default namespace for all the service
methods. app
host   to specify the server host by
default it is local host i.e "127.0.0.1",
port   to specify the port number for the
server , by default it is 8080.
```

Create an instance of the StandaloneServer class and start the server:

```
myServer =
Temperature_Conversion_Server.new('Convers
ionServer', 'urn:ruby:conversionservice',
'localhost', 8080)
myServer.start
```

The temperature conversion service is now ready for testing. Start the server using the following command:

```
ruby  ConversionServer.rb
```

Develop a Web service client application to test the Temperature conversion service (see Listing 3) and test the service using the command:

```
ruby ConversionClient.rb
```

In order to monitor the Web service traffic (i.e, SOAP Request and SOAP Response), run the monitor application, which is included with the XML-RPC samples.

Conclusion

It's easy to develop Web services with Ruby. SOAP4R supports logging for the client side and server side as well and supports the user-defined data types. SOAP4R provides the ability to customize the mapping between the Ruby and the SOAP Types. It has good support for SOAP Parameters like IN, OUT, INOUT, and return, and supports WSDL as well. Refer to the samples that come with SOAP4R for more information.

References

- Ruby: www.ruby-lang.org/en/
- SOAP4R: www.jin.gr.jp/~nahi/Ruby/SOAP4R

Listing 1: SendEmail.rb

```
# SendEmail.rb
# Email Client
require 'soap/driver'

NAMESPACE  =
'http://www.abysal.com/Abysal-webDTP'
URL     =
'http://www.abysal.com/soap/abysal_we
bdtp'
SOAPACTION =
'http://www.abysal.com/soap#abysal_pr
ogram=soapmail'
HTTP_PROXY =  nil

#create a SOAP Driver Object
driver =SOAP::Driver.new(nil, nil,
NAMESPACE, URL, HTTP_PROXY, SOA-
PACTION)

# Add the SOAP Method "SendEmail"
that takes 8 arguments
# From, FromAddress, To, ToAddress,
Subject, MsgBody, Acknowledgement,
Priority

driver.addMethod('SendEmail','From','
FromAddress', To', 'ToAddress',
'Subject', 'MsgBody',
'Acknowledgement', 'Priority' )
```

```
# call the SOAP Service

result = driver.SendEmail( 'srinivas','aravilli@yahoo.com',
'Gail Schultz',
'gail@sys-con.com','SOAP programming using Ruby',
'Ruby SOAP Client and testing the email web service Listed
in Xmethods',
'1', '1' )
```

Listing 2: ConversionServer.rb

```
# ConversionServer.rb
# Temperature conversion services from  Celsius to
Fahrenheit

require "soap/standaloneServer"

class Temparature_Conversion_Server  <
SOAP::StandaloneServer

def methodDef
          addMethod(self, 'convert_to_Fahrenheit', 'degrees'
)
end

def convert_to_Fahrenheit( degrees )
  temp = (9*degrees + 160)/5
   return temp
end

end  #class end

server    =
Temperature_Conversion_Server.new('ConversionServer',
'urn:ruby:conversionservice', 'localhost',8080)

puts     "Now SOAP Server is starting"

server.start
```

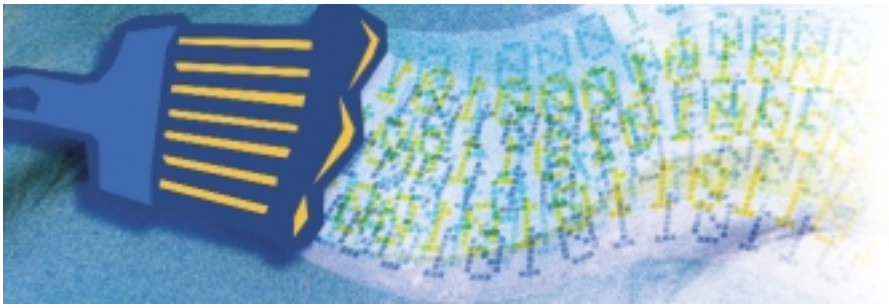
Listing 3: ConversionClient.rb

```
# ConversionClient.rb
# Temperature conversion client
require "soap/driver"

driver    =  SOAP::Driver.new(nil,nil,'urn:ruby:conversion
        service,'http://localhost:8080/')

driver.addMethod('convert_to_Fahrenheit','degrees')

puts driver.convert_to_Fahrenheit(27.3)
```

Download the code at
sys-con.com/webservices

Creating Web Services Using GLUE

An easy development framework



GLUE, by The Mind Electric, is a framework for developing and publishing Web services. It is simple, easy, and fast to create Web services. GLUE supports SOAP1.2, WSDL1.1, and UDDI v2. It comes in two editions: GLUE standard is free, and GLUE Professional has more advanced features. The Mind Electric hosts a Yahoo group for developers to post questions and share knowledge.

This article provides an introduction to how to use GLUE, including publishing and invoking Web services, working on SOAP messages, using SOAP over JMS, publishing EJBs as Web services, and publishing and inquiry using UDDI.

A Simple Web Service

Let's start by creating a simple Web service. Assume that we have an online bookstore service that provides searching for books by keywords. Here is the Java interface, which has one method that takes a string of keywords and returns a list of books matching the keywords.

```
public interface OnlineBookStore
{
  public List keywordSearch(String key-
words);
}
```

Suppose we have a class, OnlineBook StoreImpl, that implements the interface. For simplicity, the implementation will return three books if the keywords are "java web service" (see Listing 1; the code for this article is online at www.sys-con.com/web-services/sourcec.cfm).

Now we want to publish this class as a Web service. Using GLUE, we only need two lines. First, we start up an HTTP Web server on local host using port 8000, and give an application name "book". Second, we publish an instance of the class as a web service with the name "Online BookStore".

```
  electric.server.http.HTTP.startup(
"http://localhost:8000/book" );
  electric.registry.Registry.publish(
"OnlineBookStore", new
OnlineBookStoreImpl( ));
```

GLUE will generate the WSDL (Web Service Description Language) file automatically and you can access it from http://localhost:8000/book/OnlineBookStore.wsdl. The WSDL file for our service is in Listing 2.

Invoking a Web service is equally simple. First, get the URL of the WSDL of the service and bind to the service by passing the WSDL and the interface class. Then, invoke the method just like a normal method invocation.

```
 String wsdl =
"http://localhost:8000/book/OnlineBookSt
ore.wsdl";

 OnlineBookStore service =
(OnlineBookStore)
electric.registry.Registry.bind( wsdl,
OnlineBookStore.class );

 List list = service.keywordSearch("java
web services");
```

Working with SOAP Messages

The underlying structure for transporting XML documents is SOAP (Simple Object Access Protocol), a standard packaging mechanism. In our last example, the SOAP message is encapsulated from GLUE. Sometimes you need to work on low-level SOAP messages directly. GLUE provides SOAP-level APIs for you to create, send, and parse SOAP messages. After going through the following exercise, you'll have some idea of how to use SOAP messages.

This time we'll perform a keyword search on books using Amazon.com's Web services. Amazon.com Web services offer applications

Author Bio

Shannon Ma is a senior software developer specialized in J2EE, Web Services and BPM. He is currently working at AVAO Corporation, a software company that provides a robust, comprehensive and portable business-process management system (BPMS).

SHANNON.MA@AVAO.COM

Launching at LinuxWorld Conference & Expo

www.LinuxWorld.com

The Leading Magazine for Corporate IT and Business Leaders

LinuxWorld Magazine

There is no escaping the penetration of Linux into the corporate world. Traditional models are being turned on their head as the open-for-everyone Linux bandwagon rolls forward.

Linux is an operating system that is traditionally held in the highest esteem by the hardcore or geek developers of the world. With its roots firmly seeded in the open-source model, Linux is very much born from the "if it's broke, then fix it yourself" attitude.

Major corporations including IBM, Oracle, Sun, and Dell have all committed significant resources and money to ensure their strategy for the future involves Linux. Linux has arrived at the boardroom.

Yet until now, no title has existed that explicitly addresses this new hunger for information from the corporate arena. *LinuxWorld Magazine* is aimed squarely at providing this group with the knowledge and background that will allow them to make decisions to utilize the Linux operating system.

Look for all the strategic information required to better inform the community on how powerful an alternative Linux can be. *LinuxWorld Magazine* will not feature low-level code snippets but will focus instead on the higher logistical level, providing advice on hardware, to software, through to the recruiting of trained personnel required to successfully deploy a Linux-based solution. Each month will see a different focus, allowing a detailed analysis of all the components that make up the greater Linux landscape.

Regular features will include:

- Advice on Linux Infrastructure
- Detailed Software Reviews
- Migration Advice
- Hardware Advice
- CEO Guest Editorials
- Recruiting/Certification Advice
- Latest News That Matters
- Case Studies

SAVE 30% OFF!

REGULAR ANNUAL COVER PRICE $71.76

YOU PAY ONLY

$49.99

12 ISSUES/YR

*OFFER SUBJECT TO CHANGE WITHOUT NOTICE

SUBSCRIBE TODAY!

WWW.SYS-CON.COM

OR CALL

1-888-303-5282

FOR ADVERTISING INFORMATION:

CALL 201 802.3020 OR

VISIT WWW.SYS-CON.COM

that range from retrieving information about a set of products to adding an item to a shopping cart. The WSDL file of the Amazon.com Web services can be found at http://soap.amazon.com/schemas2/AmazonWebServices.wsdl.

1. ***Make a SOAP connection:*** We can get the endpoint of the Web services from the WSDL file and create a logic connection to the endpoint. In our example, the endpoint is http://soap.amazon. com/onca/soap.

```
String endpoint = "http://soap.amazon.com/onca/soap";
electric.soap.SOAPConnection connection = new electric.soap.SOAPConnection(endpoint);
```

2. ***Create a SOAP message, and add a "SOAPAction" to the HTTP header:*** The value of the SOAP action is also read from WSDL.

```
electric.soap.SOAPMessage request = new electric.soap.SOAPMessage();
request.addMIMEHeader("SOAPAction", "http://soap.amazon.com");
```

3. ***Create a SOAP envelope and set some general namespaces.***

```
electric.xml.Element envelope = request.addEnvelope();
envelope.setNamespace("soapenc", "http://schemas.xmlsoap.org/soap/encoding/");
envelope.setNamespace("soap", "http://schemas.xmlsoap.org/soap/envelope/");
envelope.setNamespace("xsi", "http://www.w3.org/2001/XMLSchema-instance");
envelope.setNamespace("xsd", "http://www.w3.org/2001/XMLSchema");
```

4. ***Create a SOAP body:*** Set the message name (KeywordSerarchRequest), SOAP encoding style, and namespace.

```
electric.xml.Element body = request.addBody();
electric.xml.Element method = body.addElement();
method.setAttribute("soap:encodingStyle",
"http://schemas.xmlsoap.org/soap/encoding/");

method.setName("namesp1:KeywordSearchRequest");
method.setNamespace("namesp1", "urn:PI/DevCentral/SoapService");
```

5. ***Put the keyword search parameters into the message:*** Under the "KeywordRequest" element, specify the following:
   - Keyword is "java web services".
   - Page number is "1", which means return the first page of the result.
   - Product mode is "books".
   - Tag is "webservices-20" (Amazon Web services 2.0)
   - Document type is "lite"
   - Developer tag is "D11EFJ3ULGUTHN". (you need to obtain a tag from Amazon.com before you can use the Web services.)
   - Version is "1.0" (see Listing 3).

6. ***Invoke the Web service by calling invoke on the SOAPConnection:*** Pass the SOAP message we just created and get a list of books matching the keyword "java web services".

```
electric.soap.SOAPMessage response = connection.invoke(request);
```

Listing 4 is the generated "KeywordSearchRequest" SOAP request message. The complete code for the SOAP message is in Listing 5.

SOAP over JMS

JMS (Java Message Service) provides reliable and guaranteed delivery messaging. GLUE provides a mechanism for sending and receiving SOAP messages over JMS. Built-in adapters are included for MQSeries (IBM), SonicMQ (Sonic Software), TIBCO JMS (TIBCO), and SwiftMQ (IIT Software). For other JMS products, you can write your own adapter. We will use the same example, OnlineBookStore, to demonstrate SOAP over JMS using the SonicMQ JMS Server.

Publishing a Web service using JMS is similar to publishing one using HTTP. Start the SonicMQ Broker, do the following, and the Web service will be ready over JMS:

```
electric.server.jms.JMS.startup( "jms:///book" );
electric.registry.Registry.publish( "OnLineBookService", new OnlineBookServiceImpl() );
```

It creates a queue named "GlueQueue/book" in SonicMQ. To invoke the service, just bind to the service using JMS and call the method.

```
String wsdl = "jms:///book/OnlineBookStore.wsdl";
OnlineBookService service = (OnlineBookService) electric.registry.Registry.bind( wsdl, OnlineBookService.class );
List list = service.keywordSearch("java web services");
```

The example above uses synchronous messaging. JMS also provides asynchronous messaging that loosely couples the sending and receiving of the messages. To receive SOAP messages asynchronously, we need to add another method with a different signature to our interface OnlineBookStore .

```
public void keywordSearch(String keywords, electric.util.async.Async async);
```

You may have noticed that the new method is different from the previous method in that it has an extra parameter of type Async and the return type is void. Now we add the implementation of the method in our class OnlineBookStoreImpl.

Publishing the Web service is the same for both synchronous and asynchronous messaging. The difference is the invoking. Contrary to the invoking of a synchronous method, when you invoke an asynchronous method there is no response returned as the result of the method call. Instead, after binding to the Web service we need to create an instance of Async and pass it to the asynchronous method. Now we can do something else, then call getResponse on Async to get the result (see Listing 7).

Publishing EJBs as Web Services

GLUE can publish any stateless session

bean as a Web service. It uses a generic IService interface that allows different kinds of objects to be exposed. GLUE includes an implementation of IService called StatelessSessionBean Service that acts as a proxy to forward a SOAP request to the stateless session bean, and return the result as a SOAP response.

We will show, in the example, how to publish a stateless session bean that runs on WebLogic App Server. First we'll create a GLUE application to run in WebLogic using the 'newapp' command-line utility.

```
> newapp -h ejb2ws
```

Now we have a Web application called ejb2ws. Next we'll create an XML file called ejb2ws.xml and save it under ejb2ws\WEB-INF\services. The file will look like Listing 8, assuming that we have an EJB with the JNDI name OnlineBookStore and a home interface named OnlineBookStoreHome.

The ejb2ws.xml file provides the StatelessSessionBeanService with information on both the BEA WebLogic Server and the stateless session bean. We can add the following to instruct WebLogic to load the GLUE application to config.xml of the domain of the application server, assuming that our ejb2ws application is under d:\myapplication.

```
<Application Deployed="true"
Name="ejb2ws" Path=" d:\myapplication">
 <WebAppComponent Name=" ejb2ws "
Targets="myserver" URI=" ejb2ws "/>
</Application>
```

Starting the WebLogic Server will expose the stateless session bean as a Web service.

Using UDDI

UDDI (Universal Description, Discovery and Integration) provides a registry of Web services for advertisement, discovery, and integration purposes. You can use GLUE to publish your business to a registry, or to search for a business using UDDI.

The following code snippet demonstrates publishing a service through UDDI. For simplicity, we use GLUE's UDDI server.

```
electric.server.http.HTTP.startup(
"http://localhost:8004/glue/inquiry",
"/inquiry" );
electric.server.http.HTTP.startup(
"http://localhost:8005/glue/publication",
"/publication" );
electric.uddi.server.UDDIServer server =
new electric.uddi.server.UDDIServer(
"inquiry/uddi", "publication/uddi", "pub-
lication/admin", "/uddi", true );
server.setOperator( "me" );
```

Now we are ready to publish our online book service. Remember that a valid user is required when publishing a service (Listing 9 is the code to create a user).

1. Create a UDDI client connecting to the UDDI server.

```
String inquiryURL = "http://local-
host:8004/glue/inquiry/uddi";
String publicationURL = "http://local-
host:8005/glue/publication/uddi";
String user = "me";
String password = "ok";
electric.uddi.client.UDDIClient uddi =
new electric.uddi.client.UDDIClient(
inquiryURL, publicationURL , user , pass-
word);
```

2. Define the business and add some contact information.

```
electric.uddi.Business business = new
electric.uddi.Business( new
electric.uddi.Name( "Online Book Store,
Inc.", "en" ) );
 electric.uddi.Contact contact = new
electric.uddi.Contact( "support" );
 contact.setUseType( "Support" );
 electric.uddi.Email email = new elec-
tric.uddi.Email( "support@onlinebook.com"
);
 contact.addEmail( email );
 electric.uddi.Phone phone = new elec-
tric.uddi.Phone( "1-800-SUPPORT" );
 contact.addPhone( phone );
 business.addContact( contact );
 business.addDescription( new
electric.uddi.Description( "An online
book store" ) );
```

3. Categorize the business as bookstore (code 451211) using the NAICS 2002 code (North American Industry Classification System) taxonomy.

```
electric.uddi.Category cat = new elec-
tric.uddi.Category( "ntis-gov:naics:2002",
"451211" );
cat.setTModelKey(
electric.uddi.IUDDIConstants.UDDI_NAICS_UU
ID );
business.addCategory( cat );
electric.uddi.Business savedBusiness =
uddi.saveBusiness( business );
```

4. TModel provides information about a Web service specification, categorization specification, or identifier specification. Here we create a TModel categorized as "wsdlSpec" using the "uddi-org:types" taxonomy, which means that it points to the relevant WSDL file.

```
electric.uddi.TModel tModel = new elec-
tric.uddi.TModel( "Online Book Store" );
 electric.uddi.Category category = new
electric.uddi.Category( "uddi-org:types",
"wsdlSpec" );
 category.setTModelKey(
electric.uddi.IUDDIConstants.UDDI_TYPE_TAX
ONOMY_NAME_UUID );
 tModel.addCategory( category );
 String url =
"http://localhost:8000/book/OnlineBookStor
e.wsdl";
 electric.uddi.Overview overview = new
electric.uddi.Overview( new
electric.uddi.Description( "wsdl link" ),
url );
 tModel.setOverview( overview );
 electric.uddi.TModel savedTModel =
uddi.saveTModel( tModel );
```

The complete code is shown in Listing 10.

As I mentioned earlier, another big use for UDDI is to perform inquiries. The example here demonstrates an inquiry using GLUE by connecting to the X-Methods' UDDI server. XMethods is a "virtual laboratory" for developers, listing publicly available Web services and showcasing new ways of using this particular technology.

1. Create a UDDI client connecting to the XMmethods UDDI server.

```
String inquiryURL =
"http://uddi.xmethods.net/inqu
ire";

electric.uddi.client.UDDIClien
t uddi = new
electric.uddi.client.UDDIClien
t( inquiryURL );
```

2. Search a list of businesses containing the word "bookservice", and get the TModel of the first one found.

```
electric.uddi.FindTModels
findTModels = new
electric.uddi.FindTModels();
 findTModels.setName( "book-
service" );
 electric.uddi.TModelInfos
tModelInfos =
uddi.findTModels( findTModels
);
 String tModelKey =
tModelInfos.list[ 0
].getTModelKey();
 electric.uddi.TModel tModel =
uddi.getTModel( tModelKey );
```

3. Get information for the services with bindings to implement the TModel, and get the service of the first match.

```
electric.uddi.FindServices
findServices = new
electric.uddi.FindServices();
 findServices.setTModelKeys(
new String[]{ tModelKey } );
 electric.uddi.ServiceInfos
serviceInfos =
uddi.findServices(
findServices );
 String serviceKey =
serviceInfos.list[ 0
].getServiceKey();
 electric.uddi.Service service
= uddi.getService( serviceKey
);
```

4. Get the first binding of the service and obtain the endpoint address and URL of the WSDL.

```
electric.uddi.Binding binding
= service.getBindings()[ 0 ];
 String address =
binding.getAccessPoint().getAd
dress();
 String wsdlURL =
tModel.getOverview().getOvervi
ewURL();
```

The results we get are:

```
wsdlURL =
http://hosting.msugs.ch/cheeso
9/books/books.asmx?WSDL#LookyB
ookServiceSoap
address =
http://hosting.msugs.ch/cheeso
9/books/books.asmx
```

The complete code is shown in Listing 11.

Summary

This article is an introduction to GLUE, a framework for developing Web services. Hopefully it has provided you with a good understanding of the Web services features GLUE provides.

References

- *The Mind Electric GLUE*: www.themindelectric.com/glue/index.html
- *The Mind Electric Yahoo Group*: http://groups.yahoo.com/group/MindElectricTech nology/
- *Amazon.com Web services*: http://associates.amazon.com/exec/panama/associates/ntg/browse/-/1067662/104-6667851-1953517
- *SonicMQ from Sonic Software*: www.sonicsoftware. com/products/sonicmq/index .ssp
- *BEA WebLogic Application Server*: www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/server
- *XMethods*: www.xmethods.com

Identifying and Brokering Mathematical Web Services

A formal language can move service discovery forward faster

An important part of the Web service vision being promoted by the Worldwide Web Consortium (W3C) and others is that of automated service discovery, the idea being that when we need a particular kind of service we will no longer have to go out and search for it manually; our computer will do it for us.

AUTHOR BIO:

Mike Dewar is a senior technical consultant with the Numerical Algorithms Group (NAG, www.nag.com), a worldwide organization dedicated to developing quality mathematical, statistical, and 3D visualization software components.

MIKE.DEWAR@NAG.CO.UK

Nowhere is this more necessary than in scientific computation. International collaboration here is already the norm and is increasingly being supported by the Grid, where specialized resources are connected by high-speed, high-bandwidth networks. To realize this vision requires mechanisms for describing what services actually do, and for reasoning about those descriptions in the presence of a user's problem.

MONET: Mathematics on the NET

The MONET project is a two-year investigation into how service discovery can be performed for mathematical Web services, funded by the European Union under its Fifth Framework program. The project focuses on mathematical Web services for two reasons: first, mathematics underpins almost all areas of science, engineering, and increasingly, commerce. Therefore, a suite of sophisticated mathematical Web services will be useful across a broad range of fields and activities. Second, the language of mathematics is fairly well formalized already, and in principle it ought to be easier to work in this field than in some other, less well-specified areas.

According to the Worldwide Web Consortium (W3C), the key to service discovery will be provided by the semantic Web, where information is encoded using formal languages or ontologies whose meaning is well defined and unambiguous. Given a set of service descriptions and a job specification written using a suitable collection of ontologies, a software agent ought to be able to select the appropriate service for the job and generate any necessary proxies to enable the interaction between the client and the service.

This is fine in theory but making it work in practice is rather difficult. It is actually quite hard to describe what a service does, particularly when it is based on an existing piece of mature software whose semantics are not fully documented. Moreover, the software agent needs to be able to understand the relationships between the high-level specifications described by the ontologies and the low-level service interfaces that are currently likely to be written in WSDL.

The MONET consortium includes commercial companies, national research laboratories, and universities; and aims to develop mechanisms that can be freely used by scientists worldwide. Details of the technology it is developing can be downloaded from the project Web site at http://monet.nag.co.uk.

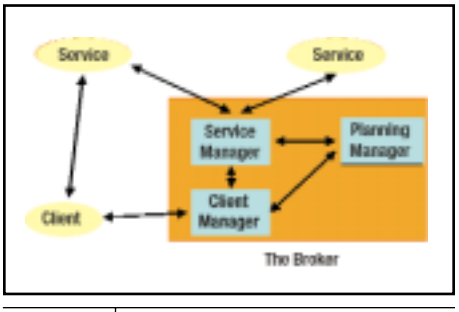


FIGURE 1 | **Interaction example**

The Problem

While for many areas of mathematics there are generic algorithms that can, in theory, solve many problems in that class, in practice most of the algorithms scientists use are designed to address a relatively narrow range of problems. Specialized algorithms are generally much more efficient and predictable in terms of resource usage and can yield more accurate results. This leads to an interesting question about mathematical Web services: Will they be fairly monolithic and address a whole class of problem, or will each algorithm be a separate service?

The advantage of deploying Web services that address whole classes of problems is that it makes the process of matching problem to service much easier. The software doing the matching does not need to understand the more subtle features that are used to distinguish the subclasses; this kind of specialized knowledge is part and parcel of the service. On the other hand, specialized services are more lightweight from a developer's point of view, and reflect the reality of how mathematical software is developed and used in practice. The MONET project takes the view that you need to be able to capture arbitrarily detailed information about the applicability of a service.

The MONET Architecture

A simple example of the kind of interaction MONET aims to facilitate is shown in Figure 1.

There are three "actors" in this process. The first is a group of services and the second is a client. The third is a piece of software called the Broker, but for convenience we distinguish between several of its components that undertake distinct, well-defined tasks.

The process of the client discovering an appropriate service and then invoking it can be broken down into five steps:

1. ***Registration:*** The services register their capabilities, access policies, etc., with the broker's service manager.
2. ***Inquiry:*** The client sends a description of the kind of service it is looking for to the broker. This description may be generic (e.g,. something like "find me a service that performs definite integration", or specific (e.g., "find me a service to solve the problem $\int_a^b \sin(x)\,dx$".
3. ***Analysis:*** The planning manager inside the broker analyzes the problem and extracts the criteria on which to select a service, which it then matches against the registry maintained by the service manager. If it finds one or more possible matches their details are returned to the client along with an indication of how closely they fit its requirements.
4. ***Selection:*** The client selects a suitable service and requests access to it via the broker's service manager. What this entails depends on the access policies of the service and the particular service infrastructure being used; in the case of grid services, for example, where every abstract service is actually a factory, a new service instance would be generated and a handle returned to the client.
5. ***Connection:*** If access is granted, then the client initiates a connection to the service.

This is a simple scenario and MONET has been designed to work with more complex and less easily defined problems. In practice, the broker might employ other, more sophisticated planning services to help it with the matching process, and the result might not be a list of candidate services but a list of execution plans based on the composition of multiple services using a suitable choreography language such as BPEL4WS. However, in all cases the key to making the process work is a sophisticated mechanism for describing problems and services that allows for the effective matching of one to the other.

Mathematical Service Description Language (MSDL)

MSDL is the collective name for the language we use to describe problems and services. Strictly speaking, it is not itself an ontology but it is a framework in which information described using suitable ontologies can be embedded. One of the main languages we use is OpenMath, which is an XML format for describing the semantics of mathematical objects. Another is the Resource Description Framework Schema (RDFS), which is a well-known mechanism for describing the relationship between objects. The idea is to allow a certain amount of flexibility and redundancy so that somebody deploying a service will not need to do too much work

to describe it.

An MSDL description comes in four parts:

1. A ***functional description*** of what the service does
2. An ***implementation description*** of how it does it
3. An ***annotated description*** of the interface it exposes
4. A ***collection of metadata*** describing the author, access policies, etc.

MSDL Functional Description

There are two main ways in which it is possible to describe the functionality exposed by a service. The first is by reference to a suitable taxonomy such as the "Guide to Available Mathematical Software (GAMS)" produced by NIST, a tree-based system where each child in the tree is a more specialized instance of its parent. This has the twin advantages of being easy to do and of providing a hook into other taxonomy-based classification systems such as UDDI. The disadvantages are that fixed taxonomies fail to capture the evolving nature of mathematical algorithms, and a particular taxonomy may not be rich enough in certain areas (for example, GAMS makes detailed distinctions between software for numerical analysis while lumping all software for symbolic computation into one category). Moreover, while it is easy enough to grow the taxonomy from the leaves, adding internal nodes disrupts the inheritance structure.

The second way to describe the functionality exposed by a service is by reference to a Mathematical Problem Library, which describes problems in terms of their inputs, outputs, preconditions (relationships between the inputs), and post-conditions (relationships between the inputs and outputs).

For example, the problem of finding the minimum value of an expression that is subject to simple bounds on its parameters where both the expression and its derivatives are present might be expressed as follows.

Inputs

1. $F(v_1, \ldots, v_n) : \mathcal{R}^n \rightarrow \mathcal{R}$
2. $A \subseteq \mathcal{R}^n$
3. $\{D_i(v_1, \ldots, v_n) : \mathcal{R}^n \rightarrow \mathcal{R}, i = 1 \ldots n\}$

Outputs

1. $x \in A$
2. $f \in \mathcal{R}$

Pre-conditions

1. $D_i = \frac{\partial F}{\partial v_i}, i = 1 \ldots n$

Post-conditions

1. $F(x) = f$
2. $\nexists y \in A : F(y) < f$

One important use of the Mathematical Problem Library is to provide names for the various objects that form parts of the problem (in this case F, A, Di, x, and f). This is used in both formulating a problem (i.e., one can say, etc.) and also, as we shall see later, in understanding how to construct the messages defined in the WSDL file.

There are a number of other pieces of information that can be included in the functional description. A directive is used to indicate something about the approach taken by the service to tackle a user's problem. In the above case the directive would usually be *solve*, but an alternative might be *prove* – i.e., given particular inputs and outputs plus the preconditions *prove* that the post-conditions hold. It is also possible to include statements in other formalisms such as RDFS or the emerging Web Ontology Language (OWL) being developed by the W3C.

While this may seem to involve a certain amount of redundancy, the various parts of the functional description can, in practice, be highly complementary. For example, the entry in the problem description library might indicate that the problem involves the solution of a particular kind of differential equation while the taxonomy reference would add the fact that the equation is stiff.

MSDL Implementation Description

The MSDL Implementation Description provides information about the specific implementation that is independent of the particular task the service performs. This can include the specific algorithm used (by reference to an Algorithm Library), the type of hardware the service runs on, and so on.

In addition, it provides details of how the service is used. This includes the ability to control the way the algorithm works (for example, by limiting the number of iterations it can perform or request a specific accuracy for the solution), and also the abstract actions that the service supports. While in the MONET model a service described in MSDL solves only one problem, it may do so in several steps. For example, there may be an initialization phase, then an execution phase that can be repeated several times, and finally a termination phase. Each phase is regarded as a separate action supported by the service.

MSDL Interface Description

While WSDL does a good job in describing the syntactic interface exposed by a service, it does nothing to explain the semantics of ports, operations, and messages. MSDL has facilities that relate WSDL operations to actions in the implementation description, and message parts to the components of the problem description. In fact the mechanism is not WSDL-specific and could be used with other interface description schemes such as IDL.

Conclusions

There are many other aspects of Web services – not least the ability to negotiate terms and conditions of access, measure the quality of the actual service provided, and choreograph multiple services to solve a single problem – that are still being worked out. The partners in the project's ultimate goal is to develop products and services based on the MONET architecture but the viability of this depends to a large extent on solutions to the other emerging issues. While we are confident that this will happen, it is not yet clear what the timescale will be.

The MONET project is currently building prototype brokers that can reason about available services using MSDL descriptions encoded in the W3C's OWL. We are also investigating the applicability of this technology to describing services deployed in the Open Grid Service Architecture (OGSA).

Useful Resources

- *MONET*: http://monet.nag.co.uk
- *OpenMath*: www.openmath.org
- *OGSA*: www.ggf.org/ogsa-wg/
- *BPEL4WS*: www.ibm.com/developerworks/Web services/library/ws-bpel/
- *Worldwide Web Consortium*: www.w3c.org
- *OWL*: www.w3.org/2001/sw/WebOnt/
- *GAMS*: http://gams.nist.gov
- *UDDI*: www.uddi.org

International Web Services Conference & Expo

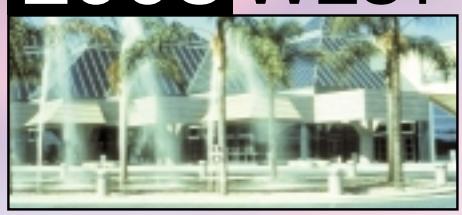
Web Services Edge

3rd Annual

web services conference EDGE &expo

Delivering .NET, JAVA, Mac OS X, & XML Technologies

2003 WEST



SEPT. 30 - OCT. 2, 2003

Santa Clara, CA

web services conference EDGE &expo

REGISTER TODAY!

CALL 201-802-3058

www.sys-con.com

Register before August 1st and

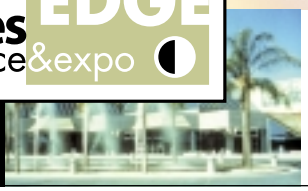
SAVE Up To $300

Owned and Produced by: SYS-CON EVENTS

Media Sponsors: WebServices Journal, Java Developer's Journal, WebSphere Developer's Journal, XML Journal, .NET Developer's Journal, WebLogic Developer's Journal, LinuxWorld Magazine, wireless Business Technology, Linux Business Week, CE Advisor, ColdFusion Developer's Journal, PowerBuilder Developer's Journal, market WIRE, ASP Today, OASIS, SD Times, LinuxWorld.com, SAMS, Java Industry

WEB SERVICES EDGE CONFERENCE & EXPO

web services conference&expo EDGE



SEPT. 30 - OCT. 2, 2003

Santa Clara Convention Center

It is our pleasure to bring the latest edition of the highly successful Web Services Edge Conference & Expo to the Santa Clara Convention Center, September 30 – October 2, 2003. The Third Annual Web Services Edge West Conference & Expo will continue to build on our past success to make available the most current and relevant information to you, our valued attendee.

With the widespread adoption of Web services across the industry, developers are facing new challenges. In this year's conference program, we will address these challenges with our most comprehensive program to date. Web Services Edge 2003 West will provide practical approaches and solutions to overcome the hurdles in developing and deploying Web services in today's competitive markets. Once again Web Services Edge will feature dedicated tracks covering Java, Web Services, .NET, and XML - along with the newly added Mac OS X Track. Sessions in this track will highlight the use of the Mac OS X platform, which combines the ease of use of a Mac with the power of Unix, in applications and Web services development, deployment, and management.

Your three days will include highly informative keynotes, conference sessions, tutorials, industry-leading university certification programs, case studies, and demo presentations. The Expo Hall will be open on September 30 and October 1, featuring the largest grouping of quality exhibitors prepared to field your questions and solve your development needs.

All the best,
SYS-CON Events

FEATURES & ATTRACTIONS

TO ALL CONFERENCE & EXPO REGISTRANTS

- 3 Days Packed with Education and Training
- Keynotes & Panel Discussions from Industry Leaders
- 60 Hard-hitting and Informative Seminars
- FREE .NET Tutorial with Russ' Tool Shed
- Java University Certification Training
- Industry-Leading Certification Programs
- "Birds of a Feather" Discussions
- Round Table Discussions
- Opening Day Welcome Reception
- SAMS Meet the Authors Hot Topics Lounge
- Compelling Case Studies & Best Practices
- Hands-On Labs
- Featured Product Demonstrations
- Exhibit Floor featuring more than 40 companies and hundreds of products
- Real-time SYS-CON Radio Interviews

WHO SHOULD ATTEND

- Software Developer
- Software Engineer
- Development Manager
- Application Developer
- Technical Director
- Analyst/Programmer
- IT Manager
- Technical Architect
- Team Leader
- Software Consultant

HIGHLIGHTED SPEAKERS

Allan Vermeulen

CTO, Amazon.com

CTO and vice president at Amazon.com directly oversees the Platform Technologies group. This group is responsible for guiding Amazon.com's technology architecture, including building and acquiring foundational components. Prior to his move to Amazon.com, Vermeulen was CTO and vice president of development at Rogue Wave Software. He holds a PhD in Systems Design Engineering from the University of Waterloo.

John Schmidt

Leader of Systems Integration and Middleware, Best Buy Co.

John Schmidt is the chairman of the Methodology Committee for the EAI Industry Consortium and leader of systems integration and middleware at Best Buy Co., a leading specialty retailer of consumer electronics, personal computers, entertainment software, and appliances.

Dave Chappell

VP, Chief Technology Evangelist, Sonic Software

Dave Chappell is the vice president and chief technology evangelist for Sonic Software. He has more than 18 years of industry experience building software tools and infrastructure for application developers, spanning all aspects of R&D, sales, marketing, and support services. Dave has also been published in numerous technical journals, and is currently writing a series of contributed articles for *Java Developer's Journal*.

Anne Thomas Manes

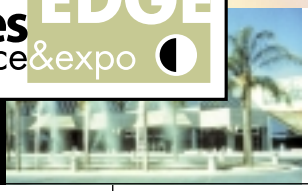
Research Director, Burton Group

Anne Thomas Manes is a research director at Burton Group, a research, consulting, and advisory firm. Anne leads research for the Application Platform Strategies service. Named one of NetworkWorld's "50 Most Powerful People in Networking" in 2002, and one of Enterprise Systems Journal's "Power 100 IT Leaders" in 2001, Anne is a renowned technologist in the Web services space. Anne participates in standards development at W3C and OASIS. She is a frequent speaker at trade shows and author of numerous articles and the book *Web Services: A Manager's Guide*.

Web Services Edge WEST 2003

WEB SERVICES EDGE CONFERENCE & EXPO

web services conference&expo EDGE



SEPT. 30 - OCT. 2, 2003

Santa Clara Convention Center

WEB SERVICES TECHNOLOGY

Presentations will include discussions of security, interoperability, the role of UDDI, progress of the standards-making bodies, SOAP, and BPM. Case studies cover the design and deployment of Web services in the marketplace.

Sessions will focus on:

- Interoperability
- Enterprise Networks
- Web Services Management
- Web Services Standards
- Web Services Orchestration
- Security (WS-Security, SAML)
- BPEL4WS
- UDDI: Dead or Alive?
- ebXML & Web Services
- EAI & Web Services
- RPC vs Documents: Uses and Differences
- User Interfaces for Web Services
- Web Services Best Practices
- Service Oriented Architecture

JAVA TECHNOLOGY

The Java Track features presentations aimed at the beginner, as well as the seasoned Java developer. Sessions will explore the whole spectrum of Java, focusing on J2EE, application architecture, EJB, and J2ME. In addition the track will cover the latest in SWT, Ant, JUnit, open source frameworks, as well as an in-depth look into the vital role that Java is playing in building and deploying Web services.

Sessions will focus on:

- Enterprise Java 1.4
- Ant Applied in "Real World" Web Services
- Developing Application Frameworks w/SWT
- Empowering Java and RSS for Blogging
- JUnit: Testing Your Java with JUnit
- JDK1.5: The Tiger
- Simplifying J2EE Applications
- Using IBM's Emerging Technologies Toolkit (ETTK)
- Apache Axis
- Meeting the Challenges of J2ME Development
- Integrating Java + .NET
- Squeezing Java

XML TECHNOLOGY

Presentations will focus on the various facets of XML technologies as they are applied to solving business computing problems. Sessions will include emerging standards in XML Schemas, XML repositories, industry applications of XML, applying XML for building Web services applications, XML/XSLT/XQuery-based programming using Java/.NET, XML databases, XML tools and servers, XML-based messaging, and the issues related to applying XML in B2B/EAI applications. The XML Track is geared for audiences ranging from beginners to system architects and advanced developers.

Sessions will focus on:

- XML Standards & Vocabularies
- Introduction to XForms
- Securing Your XML and Web Services Infrastructure
- XQuery Fundamentals: Key Ingredient to Enterprise Information Integration
- XML and Enterprise Architecture: Technology Trends
- Standards-Based Enterprise Middleware Using XML/Web Services
- XML and Financial Services
- Canonical Documents for Your Business: Design Strategies
- XPath/XSLT 2.0: What's New?
- XML Schema Best Practices
- XML in EAI, Enterprise Portals, Content Management

.NET TECHNOLOGY

Presentations will explore the Microsoft .NET platform for Web services. To the average developer, it represents an entirely new approach to creating software for the Microsoft platform. What's more, .NET development products - such as Visual Studio .NET - now bring the power of drag-and-drop, GUI-based programming to such diverse platforms as the Web and mobile devices.

Sessions will focus on:

- ASP.NET
- Security
- VB.NET
- .NET and XML
- Smart Device Extensions for VS.NET
- Best Practices
- Shared Source CLI
- .NET Remoting
- Smart Devices in Health Care Settings
- Mobile Internet Toolkit
- ROTOR
- Portable .NET
- ASP.NET Using Mono
- Using WSE with IBM's WSTK
- GUI applications Using Mono
- Portals – Windows SharePoint Services/SharePoint Portal Server
- Windows Server 2003 and IIS 6
- .NET and Java Interoperability
- Distributed .NET for Financial Applications
- Developing C# with Eclipse

MAC OS X

OS X represents a new wave of operating systems. It combines the ease of use of a Mac with the power of Unix. Sessions in this track will highlight the use of the Mac OS X platform in applications and Web services development, deployment and management.

Sessions will focus on:

- Introducing OS X (Panther): What's New?
- Quick Applications using AppleScript
- Enterprise Java and OS X
- Developing Web Services Using WebObjects
- Xserve: Ease of OS X and Power of Unix
- Introducing Quartz: 2D Graphics for Apple
- OS X for the Unix Developer
- Securing OS X Applications
- Java and OS X: A Perfect Marriage
- Programming Rich User Interfaces Using Cocoa

SPECIAL DISCOUNTS AVAILABLE

Take advantage of the Early Bird and Pre-registration values available right now, or save even more with a group of 5 or more. For special group discounts contact Michael Lynch at mike@sys-con.com, or by phone at (201) 802-3058.

For more information visit
www.sys-con.com
or call
201 802-3069

PRODUCED BY SYS-CON EVENTS

Conference at-a-Glance

		JAVA	.NET	WEB SERVICES	XML
TUESDAY, SEPTEMBER 30	DAY 1				
	8:00AM – 4:00PM	**REGISTRATION**			
	9:00AM – 9:50AM	Enterprise Java 1.4	Using WSE 2.0	Web Services Management	Introduction to Xforms
	10:00AM – 10:50AM	Opening Keynote - Allen Vermeulen, CTO, Amazon.com			
	11:00AM – 6:00PM	**EXPO OPEN**			
	2:00PM – 2:50PM	Keynote Panel Discussion - Enterprise Application Integration			
	3:00PM – 3:50PM	Ant Applied in "Real World" Web Services	Smart Devices in Health Care Settings	Service Oriented Architecture	Securing Your XML and Web Services Infrastructure
	4:00PM – 4:50PM	Developing Application Frameworks with SWT	Using the Mobile Internet Toolkit	Web Services Orchestration	XQuery Fundamentals: Key Ingredient to Enterprise Information Integration
	5:00PM	**OPENING NIGHT RECEPTION**			
WEDNESDAY, OCTOBER 1	DAY 2				
	8:00AM – 4:00PM	**REGISTRATION**			
	9:00AM – 9:50AM	Integrating Java and .NET	Introduction to ROTOR	Security (WS-Security, SAML)	Standards-Based Enterprise Middleware Using XML/Web Services
	10:00AM – 10:50AM	Morning Keynote			
	11:00AM – 4:00PM	**EXPO OPEN**			
	2:00PM – 2:50PM	Keynote Panel Discussion - Interoperability: Is Web Services Delivering?			
	3:00PM – 3:50PM	JUnit: Testing Your Java with JUnit	Using Portable .NET	WS-BPEL	XML and Enterprise Architecture: Technology Trends
	4:00PM – 4:50PM	JDK1.5: The Tiger	ASP.NET with Mono	UDDI: Dead or Alive?	Using XML Schemas Effectively in WSDL Design
	5:00PM – 6:00PM	Squeezing Java	Using WSE with IBM's WSTK	Web Services Choreography, Management, and Security - Can They Dance Together?	Canonical Documents for Your Business: Design Strategies
THURSDAY, OCTOBER 2	DAY 3				
	8:00AM – 4:00PM	**REGISTRATION**			
	9:00AM – 9:50AM	Using IBM's Emerging Technologies Toolkit (ETTK)	Distributed .NET for Financial Applications	eAI & Web Services	XML and the Fortune 500
	10:00AM – 10:50AM	Morning Technical Keynote			
	11:00AM – 11:50AM	Apache Axis	Developing C# with Eclipse	RPC vs Documents: Uses and Differences	XPath/XSLT 2.0: What's New?
	12:00PM	**BREAK**			
	1:00PM – 1:50PM	Meeting the Challenges of J2ME Development	Windows SharePoint Services	The Seven Habits of Highly Effective Enterprise Service Buses (ESBs)	ebXML & Web Services
	2:00PM – 2:50PM	Keynote Panel Discussion - Summit on Web Services Standards			
	3:00PM – 3:50PM	Empowering Java and RSS for Blogging	BizTalk 2003	See www.sys-con.com for more information	See www.sys-con.com for more information
	4:00PM – 5:00PM	See www.sys-con.com for more information	See www.sys-con.com for more information	See www.sys-con.com for more information	See www.sys-con.com for more information

MAC OS X

	Introducing OS X (Panther) What's New?
	Programming Rich User Interfaces Using Cocoa
	Quick Applications using AppleScript
	Java and OS X: A Perfect Marriage
	Enterprise Java and OS X
	Developing Web Services Using WebObjects
	Cocoa, Carbon, Java: Application Frameworks for OS X (When to use what)
	Securing OS X Applications
	Xserve: Ease of OS X and Power of Unix
	OS X for the Unix Developer
	Introducing Quartz: 2D Graphics for Apple
	See www.sys-con.com for more information

PROGRAM SUBJECT TO CHANGE

Microsoft .net

FREE Microsoft® Tutorial

RUSS' TOOLSHED

TINKERING WITH VISUAL STUDIO

RUSS' TOOL SHED

Join Russ as he shows you how to use Visual Studio .NET

INTRO TO WEB SERVICES USINGVS.NET

One of the key ideas behind the .NET strategy is the concept of software as a service, or in short, Web services. This session will explain what a Web service is and provide an overview of its related technologies like XML, SOAP, and UDDI. We will demonstrate how the .NET Framework makes it easy to implement them for new and existing applications. This session will also provide concrete best practices for building XML Web services using Visual Studio .NET. We'll answer many common questions like: How will my Web service scale? How can my XML Web services enable interoperability with Web services from other vendors as well as within my own organization? We'll delve into building highly reliable and secure Web services. Also, we will discuss issues such as dealing with complex data types using WSDL (Web Services Description Language), as well as securing SOAP messages using encryption. We'll see how developers can use enterprise-level XML Web services to simplify customer solutions.

ADVANCED WEB SERVICES USING ASP .NET

This session will explore some of the more advanced areas of SOAP in ASP.NET's support for Web services. ASP.NET Web services are the preferred way for Web developers to expose Web services on the Internet. The goal is quick, easy, and high-performing SOAP services. We will look at how to use the SOAP extension classes to create some very interesting applications on top of the core SOAP architecture found within the .NET Framework. For instance, you can implement an encryption algorithm or screen scraping on top of the Web service call. We'll dig into more advanced topics, explore the SOAP headers, and see ways to ensure security in our Web services.

.NET REMOTING ESSENTIALS

Microsoft .NET Remoting is the .NET technology that allows you to easily and quickly build distributed applications. All of the application components can be on one computer or they can be on multiple computers around the world. .NET Remoting allows client applications to use objects in other processes on the same computer or on any other computer to which it can connect over its network. During this presentation we will discuss what you will need to know to get started with .NET Remoting. We will talk about how .NET Remoting compares with DCOM, how to host remoted objects in a variety of applications, how to call remoted objects from a client application, how to control the lifetime of remoted objects, and how to secure remoting applications.

REGISTRATION FORM

CONFERENCE: Sept. 30 – Oct. 2, 2003 EXPO: Sept. 30 – Oct. 1, 2003

Santa Clara Convention Center • Santa Clara, CA

THREE WAYS TO REGISTER FOR CONFERENCE

1) On the Web: Credit Cards or "Bill Me." Please make checks payable to SYS-CON Events.
2) By Fax: Credit Cards or "Bill Me" 201-782-9651
3) By Mail: 135 Chestnut Ridge Road, Montvale, New Jersey 07645, Attention: Registration

Please note: Registrations are not confirmed until payment is received.

Please complete sections 1, 2, 3 and 4

1 YOUR INFORMATION (Please Print) ☐ Mr. ☐ Ms.

First Name _____ Last Name _____

Title _____

Company _____

Street _____

Mail Stop _____

City _____

State _____ Zip _____ Country _____

Phone _____

Fax _____ E-Mail _____

2 PAYMENT METHOD: (Payment in full due with registration)

☐ Check or Money Order Enclosed (Registration confirmed upon receipt of payment)

Check #_____ Amount of Check $ _____

Charge my ☐ Visa ☐ MasterCard ☐ American Express ☐ Discover

Name on card_____

Card #_____ Exp. Date_____

Signature_____

Billing Address (if different from mailing address)

_____

3 PLEASE INDICATE YOUR CONFERENCE CHOICE

Total Registration fee $_____

	Before 8/1/03	Before 9/26/03	Onsite
❑ **GP Gold Passport** Good for all three days of the .NET, Web Services, XML, Java, and Mac OS X Tracks, including Keynotes, Panel Discussions, preferred seating for Microsoft .NET's Russ'"Tool Shed" Tutorial, and your choice of one Sun Microsystems Java™ University Class	$1,195.00	$1,395.00	$1,495.00
❑ **3D Three Day Conference** (Does not include Sun Java™ Education)	$1,095.00	$1,295.00	$1,395.00
❑ **2D Two Day Conference** (Does not include Sun Java™ Education) (select any two days: ❑ Tue. ❑ Wed. ❑ Thurs.)	$995.00	$1,195.00	$1,295.00
❑ **1D One Day Conference** (Does not include Sun Java™ Education) (select any one day: ❑ Tue. ❑ Wed. ❑ Thurs.)	$495.00	$595.00	$695.00
❑ **JU1 Sun Java™ University Class** Select one: ❑ Web Services Programming Using Java™ Technology and XML (Sept. 30) ❑ Java™ Fast Path: Programmer (Oct. 1) ❑ Java™ Fast Path: Architect (Oct. 2)	$595.00	$695.00	$795.00
❑ **JU2 Sun Java™ University Class** Select two: ❑ Web Services Programming Using Java™ Technology and XML (Sept. 30) ❑ Java™ Fast Path: Programmer (Oct.1) ❑ Java™ Fast Path: Architect (Oct. 2)	$1,095.00	$1,295.00	$1,395.00
❑ **JU3 Sun Java™ University Class** Select three: ❑ Web Services Programming Using Java™ Technology and XML (Sept. 30) ❑ Java™ Fast Path: Programmer (Oct. 1) ❑ Java™ Fast Path: Architect (Oct. 2)	$1,195.00	$1,395.00	$1,495.00
❑ **EO Expo Only**	FREE	FREE	$50.00

4

A. Your Job Title

❑ CTO, CIO, VP, Chief Architect
❑ Software Development Director/Manager/Evangelist
❑ IT Director/Manager
❑ Project Manager/Project Leader/Group Leader
❑ Software Architect/Systems Analyst
❑ Application Programmer/Evangelist
❑ Database Administrator/Programmer
❑ Software Developer/Systems Integrator/Consultant
❑ Web Programmer
❑ CEO/COO/President/Chairman/Owner/Partner
❑ VP/Director/Manager Marketing, Sales
❑ VP/Director/Manager of Product Development
❑ General Division Manager/Department Manager
❑ Other (please specify)_____

B. Business/Industry

❑ Computer Software
❑ Computer Hardware and Electronics
❑ Computer Networking & Telecommunications
❑ Internet/Web/E-commerce
❑ Consulting & Systems Integrator
❑ Financial Services
❑ Manufacturing
❑ Wholesale/Retail/Distribution
❑ Transportation
❑ Travel/Hospitality
❑ Government/Military/Aerospace
❑ Health Care/Medical
❑ Insurance/Legal
❑ Education
❑ Utilities
❑ Architecture/Construction/Real Estate
❑ Agriculture
❑ Nonprofit/Religious
❑ Other (please specify) _____

C. Total Number of Employees at Your Location and Entire Organization (check all that apply):

	Location	Company
10,000 or more	01 ❑	01 ❑
5,000 - 9,999	02 ❑	02 ❑
1,000 - 4,999	03 ❑	03 ❑
500 - 999	04 ❑	04 ❑
100-499	05 ❑	05 ❑
100 or less	06 ❑	06 ❑

D. Please indicate the value of communications and computer products and services that you recommend, buy, specify, or approve over the course of one year:

❑ $10 million or more
❑ $1 million - $9.9 million
❑ $500,000 - $999,999
❑ $100,000 - $499,999
❑ $10,000 - $99,999
❑ Less than $10,000
❑ Don't know

E. What is your company's gross annual revenue?

❑ $10 billion or more
❑ $1 billion - $9.9 billion
❑ $100 million - $999 million
❑ $10 million - $99.9 million
❑ $1 million - $9.9 million
❑ Less than $1 million
❑ Don't know

F. Do you recommend, specify, evaluate, approve or purchase wireless products or services for your organization?
01 ❑ Yes 02 ❑ No

G. Which of the following products, services, and/or technologies do you currently approve, specify or recommend the purchase of?

❑ Application Servers
❑ Web Servers
❑ Server Side Hardware
❑ Client Side Hardware
❑ Wireless Device Hardware
❑ Databases
❑ Java IDEs
❑ Class Libraries
❑ Software Testing Tools
❑ Web Testing Tools
❑ Modeling Tools
❑ Team Development Tools
❑ Installation Tools
❑ Frameworks
❑ Database Access Tools / JDBC Devices
❑ Application Integration Tools
❑ Enterprise Development Tool Suites
❑ Messaging Tools
❑ Reporting Tools
❑ Debugging Tools
❑ Virtual Machines
❑ Wireless Development Tools
❑ XML Tools
❑ Web Services Development Toolkits
❑ Professional Training Services
❑ Other [Please Specify]

Web Services Edge 2003 WEST

SEPT. 30 - OCT. 2, 2003

Santa Clara Convention Center

web services conference&expo EDGE

WEB SERVICES EDGE CONFERENCE & EXPO

If you require special assistance covered under the Americans with Disabilities Act, please call 201-802-3058 by September 16, 2003.

SYS-CON EVENTS

CANCELLATIONS, SUBSTITUTIONS, REFUNDS

Fax written request to SYS-CON Registration 201-782-9651. Requests for refunds received prior to August 29, 2003 will be honored, less a 10% handling charge; requests received after August 29, 2003, and before September 12, 2003, will be honored less a 20% handling charge. No requests for refunds will be honored after September 12, 2003. Requests for substitutions must be made in writing prior to September 26, 2003. No one under 18 is permitted to attend. No warranties are made regarding the content of sessions or materials.

Speakers, sessions, and schedule are subject to change without prior notice.

No solicitation by anyone other than official exhibitors, sponsors or marketing partners is permitted. Such behavior is cause for expulsion without refund.

Reviewed by Joseph A. Mitchko

SoftArtisans FileUpEE from SoftArtisans

Industrial-grade uploading/downloading for Web services

Author Bio:
Joe Mitchko is senior technical specialist for CSC Consulting in New York and is product review editor and a contributing writer for Web Services Journal.
JOE@SYS-CON.COM

When we think of all of the various complexities that can surface when implementing Web services at the enterprise level, we often think of integration issues, transactional integrity, security, and so on. We wouldn't necessarily rank file transfers as one of the more difficult technological hurdles. Just stick that big old file somewhere in the SOAP message payload area and you should be fine – unless that file happens to be several hundred megabytes or more in size or contains sensitive client financial information.

Given that the Internet is a fickle and unreliable network backbone to begin with, the formula changes a bit, and we must rely on a more industrial-strength solution to do it right. FileUp from SoftArtisans can help you overcome these issues and more when it comes to adding file transfer capabilities for your Web service and Web application needs.

Features

FileUp comes in three versions; standard, professional, and enterprise. The differences between the versions are documented on the SoftArtisans Web site, but it is important to note that SOAP-based file transfer requests are only available with the Enterprise edition. Since we are interested in the Web service features, we will cover the Enterprise Edition here.

Before we dive into the Web service–specific stuff, one important feature common to all versions is the ability of FileUp to overcome certain ASP.NET IIS limitations regarding document-processing size. I recently ran into this problem on a financial services Web service integration project where IIS could not reliably process SOAP requests greater than 64K in size under load, and that's considered small compared to having SOAP response messages include large documents such as PDF files. This is where FileUpEE comes in by providing a special ISAPI filter designed to throttle large files through the server.

FileUpEE also provides support for the development of SOAP-based Web services and has methods available for uploading and downloading content and managing the file store (copy, rename, delete, etc.).

In addition to basic file upload and download functionality, FileUpEE does a number of other things to help guarantee that a file gets there in one piece and intact. For instance, FileUpEE includes an MD5 hashing algorithm that verifies the integrity of the file. FileUpEE also allows you to continue or resume an upload that was interrupted by a network or system failure – by persisting or archiving the file transfer status to a database. If the process is interrupted, you can design scripts to pick the transfer up where it left off and guide it to completion. This is a very important feature, especially for large files. FileUpEE also allows you to monitor the progress of an upload in number of bytes transferred by using the Progress object included in the object model library.

SOFTARTISANS

COMPANY INFO
SoftArtisans
1330 Beacon Street
Suite 400
Brookline, MA 02446
Tel: 1-877-SOFTART
Web: www.softartisans.com
E-mail: info@softartisans.com

DOWNLOAD INFORMATION
http://support.softartisans.com/eval.aspx

LICENSING INFORMATION
Server License:
http//:fileup.softartisans.com/default.asapx?pageid=132

TESTING ENVIRONMENT
OS: Windows-XP
Hardware: 1GHZ Athlon, 1G RAM

Functional Overview

FileUpEE is designed to allow users on a browser application to upload or download files from their hard drives to a separate file server on the network. This process involves several mechanisms. First, getting the file from the browser to the Web server requires a browser that can implement "RFC 1867", the standard for form-based file uploads. Microsoft Internet Explorer 4.0 (or later) and Netscape Navigator/Communicator 4.5 (or later, Windows only) are compatible with the required standards.

The second half of the upload process involves physically transferring the file contents to a file server situated behind the firewall (see Figure 1). FileUpEE uses

SOAP for this transfer, hence it falls into the Web service space. But there is more to this than meets the eye. The product flexibility is such that you can initiate SOAP-based file transfers from Web server to file server across the Internet without involving the browser. FileUpEE provides a rich set of object and method calls that allow you to customize the transfer process.

Support for .NET

FileUpEE .NET assembly objects are available for Visual Basic and C# uploading scripts and take up the SoftArtisans.Net namespace.

> "FileUpEE provides a rich set of features that can allow you to integrate large-size file download and upload operations in a number of different application scenarios"

In order to perform uploads with .NET, you'll need to do some additional configuration settings in your environment, such as setting NTFS permissions on temporary and destination directories. The documentation does a great job of guiding you through any configuration settings that you may need to make, so you shouldn't have much of a problem.

Web Browser Support

FileUpEE comes bundled with full licenses for two client-side controls, XFile and JFile, which help support browsers that are not fully compliant with the RFC 1867 file transfer standard. Client-side controls can also be used to increase upload performance. XFile provides both visual and nonvisual ActiveX controls for use in Internet Explorer Web pages or Visual Basic applications, and supports SSL file transfers, handling of multiple files, and other advanced features). JFile provides similar advanced client-side features but comes packaged as a Java applet that is fully scriptable using JavaScript.

Installation

FileUpEE is supported only on the Microsoft platform and requires either Windows 2000 or Windows XP Professional, along with Microsoft Internet Information Server (IIS) version 5.x. Installation is fairly straightforward for both Web server and file server modes of installation. After installation, be prepared to make some additional configuration changes depending on the type of server-side scripting you are implementing. For instance, you may need to set permission settings on the various file directory spaces, so make sure you are logged in as an administrator. In cases where you may need to repeat the system configuration and registration steps in your environment and don't feel like reinstalling, the documentation provides step-by-step directions on setting up your server.

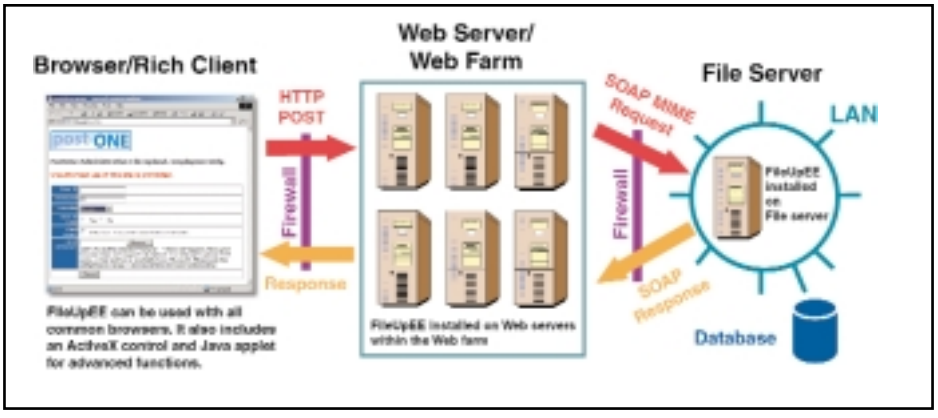


FIGURE 1 | **SoftArtisans FileUp Transfer Flow**

Documentation and Support

FileUp comes with a good deal of on-line help information that includes a number of coding samples to get you started. The documentation also supplies a programmer's reference guide with the API calls explained in detail. Anyone with VB-Script experience should be able to put together server-side scripts for file transfer ,and support for .NET means that the FileUp object model can be referenced and integrated into .NET Web services for SOAP-based server-to-server file transfers.

The online help comes in both a Windows help file and browser-based help hosted on your local IIS server. I found that the browser-based version froze up from time to time, making it impossible to use. I must note that it isn't the first time I've seen Windows Explorer take forever to render HTML content served up from IIS located on the same machine. It's not like the HTTP protocol has far to travel.

Conclusion

What is not readily supported by FileUpEE is certificate-based Web service delivery of files over the Internet using SOAP with attachments. The current SOAP-based file transfers seem intended only to get you from outside the firewall (more so in the DMZ zone) to a file server inside the firewall. You can use the .NET-based API libraries to do Web server–to–file server delivery of files through a SOAP-based delivery mechanism, but I wouldn't venture outside the enterprise without the necessary security mechanisms in place. FileUpEE does allow you to "decouple" your file servers from your Web servers so that access to a secured file server is allowed only through FileUpEE SOAP traffic over HTTP.

In all, FileUpEE provides a rich set of features that allow you to integrate large-size file download and upload operations in a number of different application scenarios, including Web service–based applications. The fact that FileUpEE is not strictly a Web service based product, is actually more beneficial for real-life integration of file upload and download operations originating from the browser.

WSJ News

Actional Announces Actional Looking Glass SOA Planner

(Mountainview, CA) – Actional Corporation, has announced the availability of Actional Looking Glass SOA Planner, their analysis and planning tool to help organizations understand the pattern and volume of activity across their Web service network, and accurately perform service network capacity planning. This new product provides end users with tools that extend the capabilities of Actional's Web services management platform.
www.actional.com

Parasoft Chosen to Improve Software Quality and Reliability

(Monrovia, CA) – Parasoft, provider of automated error prevention software solutions, has announced that Lehman Brothers, Siebel, and Countrywide have selected Parasoft tools to help them prevent software errors.

Lehman Brothers and Countrywide chose Jtest, Parasoft's automated unit testing and static analysis tool for Java. Jtest reduces time spent chasing and fixing software errors by automatically generating test cases to test code construction and functionality.

Siebel selected CodeWizard, an advanced C/C++ source code analysis tool that uses over 300 industry-respected coding guidelines to automatically identify dangerous coding constructs that compilers do not detect.
www.parasoft.com

The Mind Electric Releases GLUE 4.1

(Dallas, TX) – The Mind Electric (TME) has released version 4.1 of TME GLUE, its fast, comprehensive, easy-to-use Web services platform for Java developers creating, deploying, and managing applications with Web services, JSPs, and servlets.

TME GLUE 4.1 provides an easy-to-use, high-performance, compact implementation of all the core Web services standards, including XML, SOAP, WSDL, and UDDI. It allows any Java object to be instantly published as a Web service, and third-party Web services to be consumed as if they were local Java objects. TME GLUE can be plugged into any servlet engine and ships with its own high-performance servlet/JSP engine for out-of-the-box stand-alone use.
www.themindelectric.com

Epicor Ships Enterprise Service Automation Solution

(Irvine, CA) – Epicor Software Corporation, a provider of end-to-end, industry-specific enterprise software solutions for midmarket companies, has announced the immediate availability of Epicor for Service Enterprises, the first XML Web service enterprise service automation solution for mid-size, project-based businesses.

Epicor for Service Enterprises enables technology companies, internal IS departments, embedded service organizations, and professional service firms to carry out complete project portfolio management. Due to its pure Web service architecture, it responds in real time to an organization's specific processes and procedures.
www.epicor.com

Flashline 4 Accelerates Enterprise IT Initiatives

(Cleveland, OH) – Flashline, a provider of enterprise software asset management and reuse solutions, has announced Flashline 4, the latest release of its Asset-based Software Engineering (ASE) product family. Flashline 4 includes new cross-project collaborative capabilities that help organizations discover opportunities to eliminate redundant development and increase the value of their software asset portfolio. Also new to Flashline 4 are five FlashPacks, focused solutions to help companies accelerate the adoption of strategic enterprise-spanning software initiatives. Flashline 4 is the first complete software asset management solution embracing the range of initiatives that development organizations are undertaking to increase IT value to the enterprise.
www.flashline.com

Blue TItan Releases Network Director 2.0

(San Francisco) – Blue Titan Software, Inc., has released Network Director 2.0, the latest version of its Web services networking solution. The new release provides adaptive control of your services through advances in event-driven messaging, distributed policy execution, and shared infrastructure services.

Unlike Web services management solutions that manage services for a particular platform, or merely provide visibility applications, Network Director enforces policies throughout a distributed enterprise by reacting to real-time operational events and metrics. This makes it a pragmatic solution for actually controlling enterprise service-oriented architectures.
www.bluetitan.com

VirtualBank Selects OpenNetwork

(Clearwater, FL) – OpenNetwork, a provider of non-intrusive identity management security solutions, has been selected to implement its product platform to deliver secure internal and subscription-based .NET Web services for VirtualBank. This platform will enable VirtualBank to manage user identities, business resources and administration policies with Microsoft Active Directory as the identity repository. The joint solution provides a transparent and flexible architecture that integrates with and extends the current network environment at VirtualBank.
www.virtualbank.com, opennetwork.com

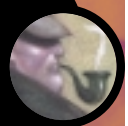
WSJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
.NET Developer's Journal	www.sys-con.com/dotnet	888-303-5282	35
Altova	http://xmlj.altova.com/xslt		27
Choreology	www.choreology.com		8, 9
Ektron	www.ektron.com/ws		11
IBM	www.ibm.com/websphere/seeit		60
Isavix	www.isavix.net	866-472-8849	31
JDJ Store	www.jdjstore.com	888-303-5282	37
LinuxWorld Magazine	www.sys-con.com	201-802-3020	39
Macromedia	www.macromedia.com/go/certification1/		25
Merant	www.merant.com	800-547-7827	23
Mind Reef	www.mindreef.com	603-465-2204	5
Mindreef News	www.mindreef.com/nl0308	603-465-2204	4
Parasoft	www.parasoft.com/wsj08	888-305-0041	2
Quest Software	http://java.quest.com/jcsc/ws	800-663-4723	59
SYS-CON Media	www.sys-con.com/suboffer.cfm	888-303-5282	53
SYS-CON Media	www.sys-con.com	888-303-5282	41
WebLogic Developer's Journal	www.weblogicdevelopersjournal.com	888-303-5282	43
Web Services Edge West 2003	www.sys-con.com	201-802-3069	47-52
WebSphere Developer's Journal	www.websphereedevelopersjournal.com	888-303-5282	45

Advertiser is fully responsible for all financial liability and terms of the contract executed by their agents or agencies who are acting on behalf of the advertiser. This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.

IN THE NEXT ISSUE OF WSJ...

Costco Electronic Hardware Services: A Case Study

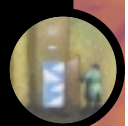


The entrepreneurial spirit that Costco is recognized for is founded on taking risk, changing and adapting to change to facilitate business. Costco EHS was looking for a powerful solution secure enough to handle their needs to add new "functionality to the solution, as their business needs changed.

Web Services: Moving Towards the Zero Latency Enterprise

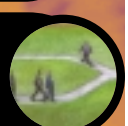
The costs of not becoming a ZLE organization are high; they translate to frustrated customers, disappointed partners, and missed opportunities. The challenges are in understanding critical business processes and developing an architecture that removes the problems. Web services standards lay out the framework you need for implementing this architecture.

Future-Proofing Solutions with Coarse-Grain Service-Oriented Architectures



The success of service-oriented architectures depends on a rich universe of available services that are easily located, understood, and utilized by a diverse community of users. These interfaces must have a life span beyond the first implementation. By taking an "outside-in" approach to modeling the service component interfaces they can become truly ubiquitous in the Web services world.

Integrating CICS Applications as Web Services



Web services are not a trend, but an industry-wide movement that can provide a long-term solution for companies that want to integrate legacy applications and data with new e-business processes.

Be Creative Implementing Web Services

Creative thinking can produce solutions that fill the void while the technology industry agrees on and implements standards.

Facing the Challenges of Web Services BPM

Identifying and overcoming challenges is the first leg on the Web services BPM journey.

WebServices JOURNAL .NET J2EE XML

Infrastructure-Level Web Services

Anne Thomas Manes

Anne Thomas Manes is a research director at Burton Group, a research, consulting, and advisory firm, where she leads research for the Application Platform Strategies service. Named one of NetworkWorld's "50 Most Powerful People in Networking" in 2002, and one of Enterprise Systems Journal's "Power 100 IT Leaders," in 2001, Anne is a renowned technologist in the Web services space. She participates in standards development at W3C and OASIS and is a member of the Web Services Journal *editorial board, a frequent speaker, and author of numerous articles and the book,* Web Services: A Manager's Guide *(Addison Wesley). Prior to joining Burton Group, Anne was CTO of Systinet. anne@manes.net*

When discussing Web services, most people tend to focus on the core Web services framework (the standards and protocols) and the applications that you can build with the framework. Although I have no trouble waxing profusely on these topics, I get even more jazzed when I start to think about infrastructure-level Web services. (I know. I need to get a life.)

Infrastructure-level Web services are Web services that implement part of the distributed computing infrastructure. They help other Web services communicate. In particular, these services make the Web services framework more robust. They provide such functionality as:

- Security and provisioning
- Performance management
- Operational management
- Metering, billing, and payments
- Routing and orchestration
- Advertisement and discovery
- Caching and queuing
- State management and persistence

John Hagel and John Seely Brown refer to infrastructure-level Web services as a service grid. (I recommend their article, "Service Grids: The Missing Link in Web Services" [www.johnhagel.com/paper_servicegrid.pdf].) I'm uneasy with the name "service grid" because it inevitably causes confusion with grid computing, which focuses on making efficient use of available resources by distributing workload across a distributed computing environment. A *service grid* is a set of managed utility services that applications can tap into in the same way that electrical appliances tap into a power grid. The key feature of the service grid is that it makes the infrastructure-level Web service pervasive and available to everyone.

Consider security – a big, gnarly issue. For it to properly protect your environment, security can't be an afterthought. It has to be integral to your entire IT infrastructure. More important, you have to get everyone to use it. Users and applications need a simple, effortless, preferably transparent mechanism to establish and propagate their identity. Runtime frameworks (such as application servers) need a simple, effortless, preferably transparent mechanism to determine a user's identity and to evaluate access rights. Applications and runtime frameworks need a simple, effortless, preferably transparent mechanism to sign and/or encrypt information and to verify signatures and decrypt information. And administrators need a simple, reasonably effortless mechanism to provision and manage identities, access rights, and encryption keys.

Right now, most companies implement security on an application-by-application basis, with a different directory or user store managing user information for each application. There's no common mechanism available to secure all your application systems. You use metadirectories to try to propagate changes across the different directories, but it still turns into an administrative nightmare. The lack of a common, cross-enterprise security infrastructure leads to errors, inconsistency, and security holes. And of course the problem only gets worse when you try to extend your application systems across corporate boundaries to connect with your customers, partners, and suppliers.

Now consider how much simpler it would be if you were using a distributed security infrastructure based on a service-oriented architecture. Imagine a set of Web services that provides simple, easy-to-use security functionality – available to all users and all applications regardless of language, platform, application, or location. These trust services include single sign-on, entitlement, signature generation and verification, and key management. From an administrative point of view, you also have policy management and provisioning services. Once you've defined the standard formats and APIs for these services, the functionality can be built into the runtime frameworks so that you no longer need to rely on developers to implement security properly. Security becomes automatic.

This isn't just a glossy-eyed dream of the future. The standards community is hard at work making this stuff happen. OASIS Security Assertions Markup Language (SAML) defines standard XML formats for security tokens (authentication and authorization assertions). It also defines standard protocols for single sign-on and entitlement Web services. OASIS Service Provisioning Markup Language (SPML) defines provisioning Web services and is a framework for exchanging user, resource, and service provisioning information. OASIS Digital Signature Services (DSS) defines standard Web services for signature generation and verification. And W3C XML Key Management Specification (XKMS) defines standard Web services for key management and distribution.

Security is not the only infrastructure area moving toward Web services. OASIS UDDI defines a standard advertising and discovery service, and OASIS Web Services Distributed Management (WSDM) will use Web services to manage distributed systems.

Peer-to-peer and grid computing systems can also capitalize on a pervasive, distributed set of infrastructure-level services to manage issues such as routing, scheduling, caching, presence, localization, security, state management, and persistence. A Web services–based infrastructure solves a huge number of problems for these systems. I get chills just thinking about it.