ADVISOR EXPERT ON
LOTUS NOTES & DOMINO R5

November, 2001
Volume 3, Issue 6

www.Advisor.com

Lotus Domino ??versions??
Simple Object Access Protocol

WEB SERVICES

Develop and Implement a Web Service for Apache SOAP and Domino

By John Duggan

Web services might be the "next big thing" as far as the IT industry is concerned; but it won't be a reality for you (the one who gets stuff done) until you can get it to work. There are plenty of articles about Web services, but this one will tell you how to create and implement a Web service for Domino.

What is a Web service?

In a nutshell, a Web service is an application that can interact with other applications using open Web standards.

In other words, it is a collection of open standards, such as:

- **Simple Object Application Protocol (SOAP)**—A protocol that lets you call methods on remote objects. (Although some would argue that SOAP has outgrown its origins and is better defined as Services-Oriented Architecture Protocol.)
- **Universal Discovery, Description, and Integration (UDDI)**—Directory of available services using SOAP.
- **Web Services Description Language (WSDL)**—Description language based on XML for services.

This article concentrates on SOAP, which lets an application call a method on a remote object. This sounds similar to Common Object Request Broker Architecture (CORBA), Distributed Component Object Model (DCOM), or Remote Method Invocation (RMI). SOAP differs from these in that it has the following advantages:

- The underlying technology is based on existing standards endorsed by the World Wide Web Consortium, such as XML and HTTP (http://www.w3.org/TR/SOAP/).
- It isn't binary, meaning it is human readable, so it's easier to debug. If your application requires it, you can encrypt it.
- SOAP is platform-independent, thus endian and byte ordering issues become irrelevant. A Windows client can happily talk to a Linux service.
- SOAP uses HTTP for its protocol, so it is "firewall friendlier" compared to CORBA and RMI.
- It has backing from the big players, such as IBM, Sun, Lotus, and Microsoft.

If you use Windows XP, you're probably already using Web services without realizing it! Figure 1 shows what happens when you click

Subscriber Download

- The Apache SOAP code used in this article
- Filename: http://Advisor.com/Article/DUGGJ02

John Duggan has been involved with IT development/management technologies since 1990. He has worked for a variety of large corporations including Coca-Cola, SAP, and Psion. He now works for Osprey Consultancy. He has a bachelor's degree in computer science from the University of Leeds and a master's degree in advanced software technology from the University of Wolverhampton. http://www.ospreyconsultancy.co.uk, John_duggan@ospreyconsultancy.co.uk.

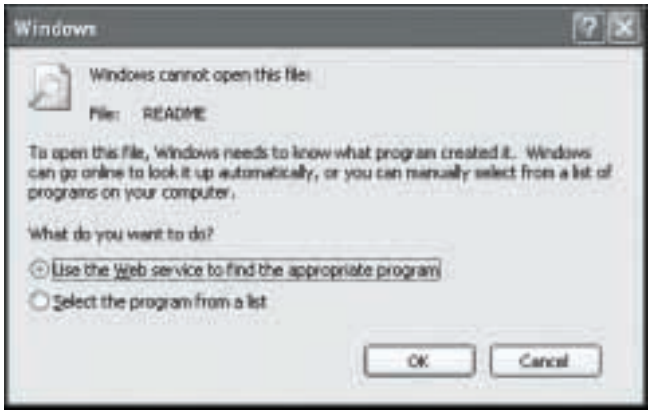

***Figure 1:* Unknown file**—Windows XP asks whether to use a Web service when it doesn't recognize the file type.

on a file Windows XP doesn't recognize. It asks you if you want to find a Web service to view the file.

OAP can replace RMI, but there are situations where it is still better to use RMI. In fact, you can use both, but that is beyond the scope of this article.

There is plenty of documentation available about SOAP, and a good place to look for further information is the SOAP FAQ at http://www.SoapRPC.com/faqs/.

What can you do with a Web service?

We have yet to realize the full potential of Web services, but you can use them for news feeds, translations, stock quotes, price checks, and many others.

SOAP introduces a performance overhead that other standards don't, but SOAP is evolving. As with most new technologies, there is room for improvement and the research on performance should eventually ripple through the SOAP implementations over the next year or two.

A more comprehensive example of SOAP in action is to use it to translate this article to Spanish. You could click on a button to activate some client code that searches the Internet (or intranet) for a service available for translation, find what methods are available, call the remote method, and return the results of a translated document. If you would like to see some currently available SOAP-based services, check http://www.xmethods.net.

Apache SOAP is the Apache Software Foundation's implementation of the SOAP specification. According to TechMetrix, it is a popular implementation (http://www.techmetrix.com/trendmarkers/techmetrixwsd.php) and it makes developing Web services surprisingly easy. It currently stands at v2.2 based on the v1.1 SOAP specification.

Installing Apache SOAP on Domino

There are several steps to get Apache SOAP working in Domino.

Download the relevant files

These files are required for implementing Apache SOAP:

- Apache SOAP v2.2 (http://xml.apache.org/soap/)
- JavaMail v1.2 (http://www.javasoft.com/products/javamail/index.html)
- JavaBeans Activation Framework v1.0.1 (http://java.sun.com/products/javabeans/glasgow/jaf.html)
- Xerces v1.4.3 (http://xml.apache.org/dist/xerces-j)

ADVISOR EXPERT ON LOTUS NOTES & DOMINO R5

ADVISOR.com
Get comprehensive technical and business resources at Advisor.com. You'll find news, articles, tips, jobs, user groups, product directories, events, forums, training, and more.

ADVISOR MEDIA
Printed in the USA

Editorial Director John L. Hawkins
Executive Managing Editor Jeanne Banfield Hawkins
Senior Managing Editor Christa L. Coleman
Managing Editor Elizabeth Olsen
Technical Editors Steve Caudill, Terrance A. Crow
Production Services Manager Jose S. Martinez
Production Manager Tina L. Bennett
Graphic Designer Terry Slusher
Circulation Manager Kristen Murphy

ADVISOR MEDIA, Inc.
Founded 1983
Chairman & Chief Executive Officer John L. Hawkins
President Jeanne Banfield Hawkins
Vice President Geri Beaty
Group Publisher Michael Callan

Contact ADVISOR Via the Web: The fastest way to reach editors, customer service, conferences and seminars, and other departments, is by going to www.Advisor.com/ContactAdvisor.

858-278-5600
Fax: 858-278-0300
5675 Ruffin Road, Suite 200
San Diego, CA 92123

Advertising Sales To speak to a sales representative about advertising opportunities, call 800-336-6060 or 858-278-5600, e-mail AdSales@Advisor.com.

Reprints and Bulk Sales: To purchase reprints or bulk magazines, contact the customer service department.

Back Issues: If a back issue is still in stock it can be purchased for US$20, and US$25 for Canada and other countries, includes postage. Contact the customer service department.

Permissions: To quote from an article, contact our Permissions Editor. Include the publication date, the title of the article, the portion you want to reprint, and the purpose of your request.

Photocopy rights: Permission to photocopy articles for use in educational institutions is granted by the Copyright Clearance Center. Contact Academic Permissions at: 222 Rosewood Dr.
Danvers, MA 01923
978-750-8400, fax 978-750-4470

Advisor Expert on Lotus Notes & Domino R5 (ISSN 1524-6388) is published by ADVISOR MEDIA, Inc., 5675 Ruffin Road, Suite 200, San Diego, CA 92123, 858-278-5600. POSTMASTER: Send address changes to Advisor Expert on Lotus Notes & Domino R5, 5675 Ruffin Road, Suite 200, San Diego, CA 92123. Subscription: 12 issues US$199, Canada US$219 (includes GST R127384618), other countries US$239.

If you use Windows XP, you're probably already using Web services without realizing it!

Why download another XML parser when there is already a parser included with R5 and Rnext? XML4j doesn't support Document Object Model (DOM) level 2, which provides XML namespaces. Until Lotus upgrades XML4j, Xerces is a good parser to use for Apache SOAP. **??edits okay??**

The implementation of Apache SOAP works on any Java-enabled platform that supports JSP and servlets. Rnext includes Tomcat Web Server/3.2.1 which supports JSP 1.1 and Servlets 2.2. This article outlines the installation using the Windows platform, but it should work on any other OS running Rnext. (I haven't tested the other platforms, but as long as Tomcat is installed with Domino, it will work.) If you're using R5, the servlet engine doesn't support JSPs, so you must install an application that supports JSPs such as Tomcat.

Install the files onto the server

A new feature of Rnext is support for Web Application Archive (WAR) files. This means you'll be working with a new database type called Web Application. This is useful, because it contains a package of files and is where I recommend putting the Apache SOAP files.

When the HTTP task loads, the files within this database are written to the local hard drive and you can access them via the HTTP task directly rather than delivering them from Domino via the HTTP task. The WAR servlet checks the settings within the Domino directory and re-deploys the files as specified to keep the files updated. All the Apache SOAP files have been attached to this article's sample database SOAP.NSF and are located in the Shared Resources\Files directory, available on ADVISOR.COM.

The file DeployedServices.ds keeps the contents of the services currently being deployed. (It is a serialized hash table.) When the HTTP task shuts down, you have to update the contents of this file within the .NSF database. Fortunately, this happens automatically if you check the Need Refresh file property.

The files activation.jar (JavaBeans Activation Framework), mail.jar (JavaMail), xerces.jar (Xerces), and soap.jar (Apache SOAP) must be located within the server's classpath. The easiest option is to copy these files to the domino\jvm\lib\ext directory where they're automatically added to the classpath on server startup.

As mentioned earlier, XML4j doesn't support XML namespaces, Xerces must appear before XML4j within the classpath. You must explicitly state the Xerces class in the JavaUserClasses within the notes.ini like this:

```
JavaUserClasses=c:\notes\jvm\lib\ext\xerces.jar;
```

Alternatively, remove XML4j.jar.

Figure 2: **SOAP administration page**—Apache SOAP has been successfully installed on a Domino server.

Test it

Finally, test the Apache SOAP by accessing http://localhost/soap/. It opens the SOAP administration page as shown in figure 2.

If you have problems with installation, check the troubleshooting guide within the Apache SOAP documentation (http://xml.apache.org/soap/docs/trouble/index.html).

Writing the Web service

Now let's write a simple Web service that looks up a share price for a specific company stored in the sample database SOAP.NSF, available for download from ADVISOR.COM.

The service is passed the company name and returns the company's share price.

The functional definition is:

```
Method: getSharePrice(companyname)
Returns: shareprice
```

It's simple to write the Web service for Apache SOAP:

```
package DominoSharePrice;
import lotus.domino.*;

public class SharePrice
{
 public static String getSharePrice(String name)
 {
   try
   {
NotesThread.sinitThread();
Session s = NotesFactory.createSession();
    Database db = s.getDatabase("", "soap.nsf");
    View view = db.getView("SharePrice");
Document doc =  view.getDocumentByKey(name, false);
if (doc != null)
      String price =
          doc.getItemValueString("SharePrice");
     else
       String price = "Not Found.";
     return (price);
   }
   catch(Exception e)
   {
     e.printStackTrace();
   }
   finally
```

```
    {
      NotesThread.stermThread();
    }
  }
}
```

Looking at the code, there is nothing special, because it is a standard Java program. There is no evidence of anything SOAP-related. Thus, you can make almost any Java program into a Web service.

Compile this code, package it, and put it into the server's CLASSPATH (e.g., jvm\lib\ext).

You should put the code into a package, because Tomcat is particular about standard classes. Create a directory "DominoSharePrice" with the class file and run from the command line:

```
jar cvf DominoSharePrice.jar DominoSharePrice
```

The package is already included within this article's sample database.

How to deploy it within Apache SOAP?

Browse to http://localhost/apache-soap/admin and click on the Deploy button. A form with a number of fields appears (figure 3). There are a lot of fields, but for this example, you're only interested in a few of them.

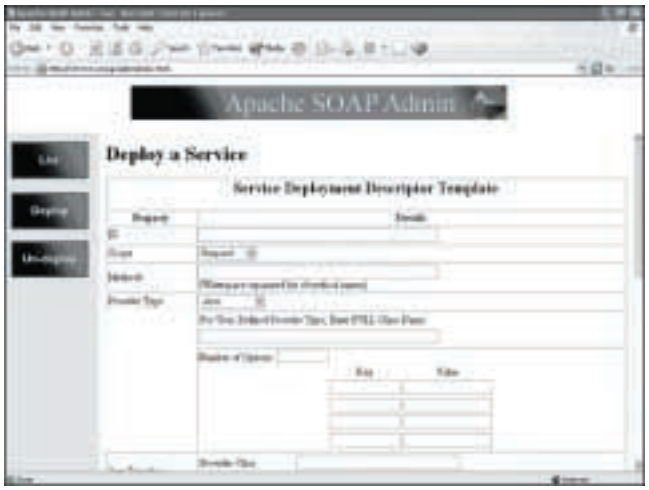

Figure 3: **Deploying a service within Apache SOAP**—At this point, you're most interested in the ID, scope, methods, provider type, and user-defined provider type fields.

The first field, ID, sets the ID of the particular object. This must be unique because this is the name that identifies the service. For this example, call this urn:DominoSharePrice to comply with uniform resource naming convention defined at http://www.ietf.org/rfc/rfc2141.txt

The Scope field defines the lifetime of the service instance. Leave this to the default Request, so the service is only available for the duration of the request. Although if you're rolling out a proper Web service, read the Apache SOAP documentation because there can be security implications with this setting.

Because there is only one method in this service, enter getSharePrice in the methods field.

Leave Provider Type as Java because you've written the program in Java.

For the user-defined provider, enter the full package name, which in this example is DominoSharePrice.SharePrice.

Leave the static field set to No because you aren't using a static method. At the bottom of the page, click on Deploy. You can check that the service has been deployed by clicking on List within the SOAP Admin page. Doing this shows the full list of deployed services on that server. Clicking on urn:DominoSharePrice shows the screen in figure 4.

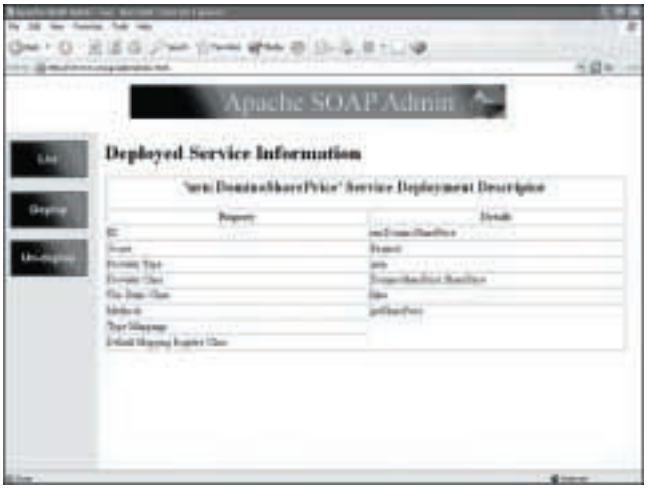

Figure 4: **The deployed Domino share price**—Here you can check the deployed services to make sure you deployed your Web service.

Writing an agent to call the Web service

You now have a Web service sitting on a Domino server waiting to be used. How do you call it from Notes?

Unfortunately, the client code is a little more complex than the server code, but it is still straightforward. You can find the complete code in the agent SOAPClient in this article's sample database.

The first step is to make sure the SOAP packages are included in the agent:

```
import org.apache.soap.SOAPException;
import org.apache.soap.Constants;
import org.apache.soap.Fault;
import org.apache.soap.rpc.Call;
import org.apache.soap.rpc.Parameter;
import org.apache.soap.rpc.Response;
```

The next step is to create a URL object with the Apache SOAP's address; for this example, it is http://localhost/soap/servlet/rpcrouter.

```
URL url = new
URL("http://localhost/soap/servlet/rpcrouter");
```

Create the Call object passed within the XML. It requires the object ID of the service, which for this example is urn:DominoSharePrice.

```
Call call = new Call();
call.setTargetObjectURI("urn:DominoSharePrice");
```

Specify the name of the method to call within the service, which is getSharePrice:

```
call.setMethodName("getSharePrice");
```

State the parameters to be passed to the method. Note that setParams requires a vector:

```
Vector params = new Vector();
params.addElement(new Parameter("IBM", String.class, name,
                                    null));
call.setParams(params);
```

Finally, activate the call to the object, being careful to catch any exceptions:

```
try
{
  Response response = call.invoke(url, "");
}
catch( SOAPException e )
{
  System.err.println("SOAPException:" + e.getMessage() +
                  "(" +  e.getFaultCode() + ") ");
  System.exit(-1);
 }
```

The details of the call are packaged into a SOAP request and sent via HTTP. After the service is found, it calls the method and sends the result back to the client. (As usual, make sure there is error handling.)

```
// Check the response.
If (!response.generatedFault() )
{
  Parameter retval = resp.getReturnValue();
  Object value = retval.getValue();
  System.out.println(value);
 }
 else
 {
   Fault fault = response.getFault();
  System.err.println("SOAP Error:  Fault:" +
       fault.getFaultString() + "(" +
fault.getFaultCode() + ")");
}
```

That's it. Using this code, you can call any method for a SOAP object anywhere on the Internet. Or, you can call anything that supports the HTTP protocol. (Note that there are plans to provide SOAP over the SMTP protocol.)

Run the SOAPClient agent within the sample database, and if everything is installed correctly, the Java debug console shows the result in figure 5.

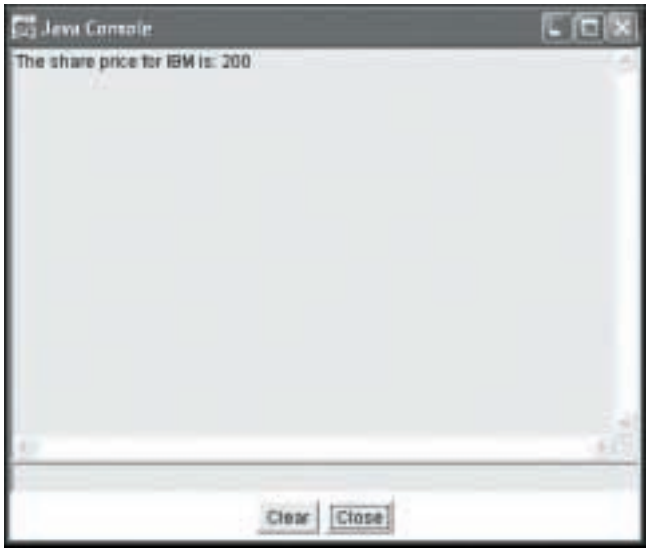

Figure 5: **Finished**—The deployed Domino share price shows the result of this article's exercise.

Figure 6: **SOAP sniffer in action**—With this handy tool, you can see communication between the client and the remote object.

What happens if it doesn't work?

As with most things in life, not everything runs that smoothly. As mentioned earlier in the article, XML isn't binary, thus it is human readable. This means you can see the communication between the client and the service. How? Part of the SOAP toolkit is a useful tool called TCP Tunnel GUI, also known as SOAP sniffer. You can call it from the command line as:

```
java org.apache.soap.util.net.TcpTunnelGui 8080 127.0.0.1
80
```

The tool redirects any requests at port 8080 to port 80. By changing the URL in the agent to read from port 8080, you can see the communication between the client and the remote object (figure 6).

The left side shows the outgoing SOAP request and the right side shows the incoming SOAP request. It is relatively straightforward, so without any knowledge of SOAP, you can get an idea of what is going on. See the SOAP FAQ for more information.

Within the agent you knew about the service to call and its details. But what if you don't know those details?

Fortunately, UDDI and WSDL can help. They let a client determine the appropriate service required for a given application, and to determine the appropriate call. Sounds impressive, doesn't it? That's a topic for another article.

Wrap up

Hopefully, you now have the technical know-how of a Web service within Domino. This article has barely scratched the surface and the potential of this interesting technology, but it's enough to get you started. You're now part of the "next big thing" ■