Using MultiSpeak 3.0 Web Services
With Microsoft Excel

Applications of MultiSpeak
Web Services Web Conference

August 16th, 2007
Presented by Sean Dennis, Lead Developer
Powel Inc, Eagan, MN
What is Web Services?
Web Services is a mechanism for software applications to communicate with one another over the network. It uses pretty much the same architecture that a web browser uses when surfing the web and getting web pages delivered to your web browser. One significant difference is that the format of the data that travels between the server and the client is not easily readable by humans like a web page is as it only contains data; not appearance properties such as color, font size, layout, alignment, etc. Web services calls are based on SOAP on top XML on top of HTTP on top of TCP/IP.

- IP is the Internet Protocol that routes packets of data from one computer to another.
- TCP is the Transmission Control Protocol which ensures that the packets were sent successfully to the destination and in the right order.
- HTTP is the Hyper Text Transmission Protocol which is how web browsers ask web servers for certain web pages. This protocol specifies whether we are getting or putting data and the type of data that is being transmitted such as a file, image, html document or xml document.
- XML stands for eXtensible Markup Language which is a text file that is made up of organized pieces of data called elements. Each element begins with a start tag like `<tagname>` and ends with a closing tag like `</tagname>`.
  For example: `<firstName>Billy</firstName>`.
- SOAP stands for Simple Object Access Protocol. This is a set of special xml elements that define what methods are called on the server, what data is passed from the client to the server as arguments to the method calls, and what data is return from the server.

What is MultiSpeak Web Services?
Multispeak is an organization comprised of utilities, software vendors that server utilities and NRECA. This organization works together to define standard mechanisms for the exchange of data between the most common software applications in use at today’s utilities. The Multispeak organization has produced three versions: Multispeak 1.1, Multispeak 2.2 and Multispeak 3.0. Versions 1.1 and 2.2 were primarily batch interfaces that defined xml files that could be exported in one application and then imported in another application. Version 2.2 did define some real-time exchange mechanisms but they were not widely adopted.

Multispeak version 3.0 still supports the file-based batch transfer architecture, but it also introduces a web services architecture that allows two systems to communicate real-time with each other. This version defines about 30 function to function interfaces. Some of them are unidirectional, some of them are bi-directional, some of them are real-time using web services, and some of them are not real-time implemented with a file-based batch mechanism. All of the web-services interfaces can be inspected at http://www.multispeak.org/interface/30j/. When looking at the interfaces defined on this web page, the server side is always listed first. For example, “FA-Staking” means that the Finance and Accounting system is listening as a server and the
Staking system can make web services calls to it. “Staking-FA” means that the staking system is listening as a server and the Finance and Accounting system can make web services calls to it.

On any interface, there are potentially two sets of methods. In any set of methods (say CB-LM for Interface 1) the function listed first hosts all of the methods listed on the referenced web page. For instance, on Interface 1, the CB-LM methods are hosted by the CB function.
Building a Sample Application using Microsoft Excel

This exercise will build a sample web services client using Microsoft Excel to pull data from StakeOut Server real-time. We will consume one of the Multispeak WSDLs, implement some basic calls and publish the results onto a spreadsheet. Our spreadsheet will basically consist of a button that gets all the work order numbers and descriptions that belong to a particular job status and bring those work order numbers and descriptions into the spreadsheet.

1. Your computer must have the Microsoft Office Web Services Toolkit installed. Once it is installed, you will be able to use it within the VBA environment inside of Excel. If you don’t have it installed, it can be downloaded at http://www.microsoft.com/downloads/details.aspx?familyid=fa36018a-e1cf-48a3-9b35-169d819eef18&displaylang=en

2. In Excel open the VBA editor and set a reference to the Microsoft Soap Type Library (MSSOAP3.DLL) from the References dialog box (located in the Tools menu). You can open the VBA editor from the Tools -> Macro -> Visual Basic Editor menu.

3. Then, on the Tools menu, choose the Web Service References option. Here we need to tell the web services toolkit how to find the WSDL that describes the web service that StakeOut Server supports. Type in the location to your web service WSDL. It can be served from your StakeOutServer or it can be served from the multispeak web site at http://www.multispeak.org/interface/30j/21_Staking_GIS.asmx?WSDL
4. Once the wizard consumes the WSDL, select the interfaces that you want to consume. In this example, select the Staking_GIS example. Each of the methods defined by the Staking_GIS interface shows a description for that method when you click on the method name. Now, click the Add button. This will generate a bunch of VB source code files for you automatically.

5. Now, we want to put a button on an excel spreadsheet that will call one of our web services. To do that, turn on the Control Toolbox by right-clicking on the toolbar and turning on the Control ToolBox toolbar. Then, using that toolbar, insert a button and set the caption to “Get Work Order List” in the button’s properties dialog.

6. Double click on the button to allow you to enter code behind the button. This code will get executed each time you click your new button. Enter the following code

```vbnet
Private Sub CommandButton1_Click()
    Dim StakeOutWS As New clsws_StakingGIS
    Dim workOrder As Variant

    On Error GoTo ButtonErrorHandler

    ActiveSheet.Range("A10:D999").ClearContents

    StakeOutWorkOrders = StakeOutWS.wsm_GetWorkOrderSelectionByStatus(ActiveSheet.Cells(6, 2), ")"

    For i = 0 To UBound(StakeOutWorkOrders, 1)
        Set workOrder = StakeOutWorkOrders(i)
        ActiveSheet.Cells(i + 10, 1).Value = workOrder.woNumber
        ActiveSheet.Cells(i + 10, 2).Value = workOrder.jobNumber
        ActiveSheet.Cells(i + 10, 3).Value = workOrder.jobDescr
        ActiveSheet.Cells(i + 10, 4).Value = workOrder.statusCode
    Next

Exit Sub
```

7. Now, the above code creates a cls廿s_StakingGIS class which provides us the methods we need to call the StakingGIS interface on the server application (StakeOut Server). The WSDL file that we imported tells the cls廿s_StakingGIS class where and how to communicate over the network to StakeOut Server. If you want to change your connection to a different location than what is defined in the WSDL, edit the cls廿s_StakingGIS file as follows:

At the bottom of the Private Sub Class_Initialize() method, insert the following code after the sc_StakingGIS.MSSoapInit2 call in the cls廿s_StakingGIS module.

```
sc_StakingGIS.ConnectorProperty("EndPointURL") = "http://URLToYourServer"
```

8. Before we test our web services code, we need to setup our spreadsheet a little bit. Fill in the cells as shown in the following image using the exact same cells as shown in the image.

9: Now let’s test our sample web services call by clicking on the “Get Work Order List” button. This will run our VBA code that will in turn call the GetWorkOrderSelectionByStatus web services method on StakeOut Server with the text in cell B6 being passed in as the job status
code. StakeOut will return a list of workOrderSelection items that are then displayed in the cells starting in row 10.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get Work Order List</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Job Status:</strong></td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Work Order Number</strong></td>
<td><strong>Extension Description</strong></td>
<td><strong>Status</strong></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1137569 @</td>
<td>REVAMP</td>
<td>A</td>
</tr>
<tr>
<td>11</td>
<td>1166028 @</td>
<td>DIS/RETIRED S/L HOLD FOR 3</td>
<td>A</td>
</tr>
<tr>
<td>12</td>
<td>1157421 @</td>
<td>DIS/RETIRED S/L # 9686, HOL</td>
<td>A</td>
</tr>
<tr>
<td>13</td>
<td>1159057 @</td>
<td>2003 CONST. WORK PLAN</td>
<td>A</td>
</tr>
<tr>
<td>14</td>
<td>1169052 @</td>
<td>2003 CONST. WORK PLAN</td>
<td>A</td>
</tr>
<tr>
<td>15</td>
<td>1159506 @</td>
<td>REVAMP</td>
<td>A</td>
</tr>
<tr>
<td>16</td>
<td>1159821 @</td>
<td>REVAMP OH SERVICE</td>
<td>A</td>
</tr>
<tr>
<td>17</td>
<td>1160655 @</td>
<td>DIS/RETIRED R11# 11605</td>
<td>A</td>
</tr>
<tr>
<td>18</td>
<td>1162640 @</td>
<td>INSTALL SECURITY LIGHT</td>
<td>A</td>
</tr>
<tr>
<td>19</td>
<td>1163288 @</td>
<td>BROKEN POLE</td>
<td>A</td>
</tr>
<tr>
<td>20</td>
<td>1163931 @</td>
<td>BURTONWOOD BUI/L BURGER</td>
<td>A</td>
</tr>
<tr>
<td>21</td>
<td>1164305 @</td>
<td>NEW CONNECT</td>
<td>A</td>
</tr>
<tr>
<td>22</td>
<td>1165180 @</td>
<td>REPL BAD UG WIRE KEENWICK</td>
<td>A</td>
</tr>
<tr>
<td>23</td>
<td>1165745 @</td>
<td>DISC AND RETIRE S/L AND R</td>
<td>A</td>
</tr>
<tr>
<td>24</td>
<td>1165795 @</td>
<td>REVAMP</td>
<td>A</td>
</tr>
<tr>
<td>25</td>
<td>1166556 @</td>
<td>NEW CONNECT</td>
<td>A</td>
</tr>
<tr>
<td>26</td>
<td>1166642 @</td>
<td>DISC RETIRE SEC. LIGHT AND</td>
<td>A</td>
</tr>
<tr>
<td>27</td>
<td>1166757 @</td>
<td>DISC RETIREMENT</td>
<td>A</td>
</tr>
</tbody>
</table>

End of Document.

Authored September, 2006
Sean Solberg, Powel-MiniMax
930 Blue Gentian Road
Eagan, MN 55121
651-251-3005