Version: FoxPro 7.0
Figures:
File for Subscriber Downloads:

Publishing Your First Web Service – Part III

Whil Hentzen

In the first two articles of this series, we've built a Web Service, published it, and consumed it – all on our development machine. Then we reviewed some of the inevitable questions that come up. Now it's time to shed those short pants and propeller beanie – let's deploy this sucker for REAL!

There are five steps involved in deploying a web service on a live server – otherwise known as "publishing" it. First, you need to create the components that you're going to deploy on your live server – a DLL as well as the WSDL and WSML files. Second, you need to create an installation package that will allow you to properly install those components on your live server. Third, install the SOAP toolkit on your live server. Fourth, install your DLL on your live server. Fifth, install and reconfigure the related components on your live server. Finally, consume that web server from another machine, such as your development box.

1. Create the components

You'll need three sets of components. The first is the DLL itself that does the work. The second are the Web Service files, like the WSDL and WSML files. And the third is the…. Data!

Create the DLL

I'll refer you to my February article for the gory details. Suffice it to say that by this point, you should have a functioning Web Service. Remember, it's just a DLL. For purposes of this article, I'll call this DLL FoxTalkDemo5.DLL, and the class inside it is called NewsService.

Since it's been a while, it's probably a good idea to remind you that it expects a parameter for a date, but in a string format, like "2001/11/1". Secondly, the code in the class is going to look for a table called NEWS.DBF in a directory called database that's located below the location of the DLL. This is the code that grabs the records of interest:

```
select cNews, dNews from database\NEWS ;
  where dNews >= ldDate ;
  order by dNews ;
  into array aNews
```

I'll come back to this in a moment or two.

Create the Web Service WSDL and WSML files

On your development machine, use the Visual FoxPro Web Service wizard to publish the Web Service. (The wizard is located under Tools | Wizards.) You'll end up with WSDL and WSML files. Where are they? The location is displayed in the Advanced Settings dialog of the wizard. See Figure 9 of February's article. On my machine, that's c:\inetpub\webpub.

If you use an ASP listener, you'll also find an ASP file in the same location as the WS?L files. Don't discard the ASP file because you'll need it too.

2. Create an InstallShield package

Now you have to install your Web Service DLL on your live web server. Yes, I said *live* machine. That's the whole point, isn't it? There are a bunch of ways to do this, but the easiest is to use InstallShield Express to create a package on your development box that you can install on your live server. Using InstallShield to do so is actually way past overkill (you only end up needing four of over twenty possible options), but it's also safest and fastest. (For more information on using InstallShield Express, see my article in January's FoxTalk.)

Run the InstallShield Express from the Start menu and click on the General Information node under Step 1. Create an InstallDir and a DatabaseDir as you desire and/or as your requirements specify. For the purposes of this article, I chose the InstallDir of Program Files\Hentzenwerke\FoxTalkDemo5 and the DatabaseDir of InstallDir\Database. (For security purposes, you'd most likely want to have your DatabaseDir set to a different location on your server – perhaps even a different machine.)

Next, under Step 2 of ISE, select the Files node. In the bottom half of the screen, be sure to right-click on the Destination Computer node and select the Show Predefined Folder menu option to display both the InstallDir and DatabaseDir nodes. Then, using the top half of the screen, navigate to your DLL and drag it to the InstallDir node, then to your NEWS.DBF file, and drag it to the DatabaseDir node.

The third step is to identify which runtime files you want to bring along for the ride. Select the Objects/Merge Modules node in Step 2, and select all three VC++ files, the first VFP runtime file, plus any local language merge modules as your needs dictate.

Finally, select the Build Your Release node under Step 6. I created a Single Image because it just creates a single SETUP.EXE file, instead of all those extra files that end up cluttering the drive somewhere. Press F7 or right-click and select Build, and you've got your SETUP.EXE file, ready for sending over to your live server.

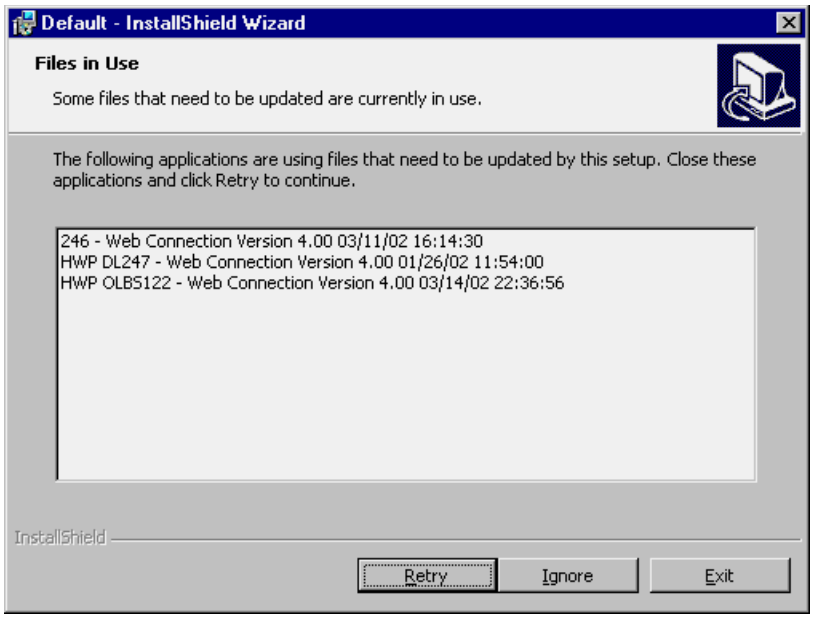
3. Install the SOAP toolkit on your live Web Server

As of this writing, the latest version of the SOAP Toolkit is still SP2, so wander on over to msdn.Microsoft.com (search on "SOAP") and grab it if you don't already have it. The toolkit itself (sans the demos and whatnot) is only about 1.5 MB.

You'll also want to make sure that IIS is configured for the proper type of listener, depending on if you're using ISAPI or ASP. The details for doing this on your live server are in the second article in this series.

4. Install the InstallShield files on your live Web Server

Copy the SETUP.EXE file to your live server – where exactly it doesn't matter – hopefully you've got hygienic habits and thus have a default location for files like this. If you've got any Visual FoxPro applications running on your live server (like a Web Connection app), you'll need to shut them down before running the installation routine. If you don't, you'll get an error message as shown in Figure 1.



WS3A_01.TIF

Figure 1. Be sure to shut down applications that are using files that will be updated by your Web Service installation.

Then run SETUP.EXE. This process will put your DLL into the proper location – in my setup, that's Program Files\Hentzenwerke\FoxTalkDemo5 - as well as register it like all good COM servers should be.

Now is also the time to create a directory under Program Files\Hentzenwerke called "database", and then to copy NEWS.DBF into it.

5. Install the related components on your live Web Server

The Web Service, if you recall, is merely a DLL sitting on your Web server that can be called by other computers on the Internet (or your Intranet.) The methods in the DLL can do things just like COM servers can now. But the DLL can't be accessed by itself; it needs a WSDL file that describes what the interface will be, what types of data can be read, and so on. So you'll need to get the WSDL, WSML and ASP files over to the live Web Server as well.

When users of your Web Service attempt to consume it, they'll put together a little snippet of code that references a URL like so:

```
http://www.yourwebsite.com/SomeDir/NewsService.WSDL
```

Thus, you'll need to create a virtual directory that SomeDir points to. Then place the WSDL, WSML (and ASP) files in that virtual so that they can be executed by your users.

But that's not all, and this next step trips up a lot of people. The WSDL was generated on your development machine, and likely contains a reference to your development box. When you move your WSDL file over to your live box, it's still going to contain that development box reference. For example, my development box WSDL file contains lines (near the bottom) like this:

```
<service name='NewsService' >
  <port name='NewsServiceSoapPort'
     binding='wsdlns:NewsServiceSoapBinding' >
    <soap:address location=
      'http://HERMAN/Webpub/NewsService.ASP' />
  </port>
</service>
```

The name of my development machine is HERMAN, and the Webpub directory is on HERMAN's C drive. This information won't be useful to someone trying to consume the NewsService Web Service on my live Web Server.

So I changed that line to

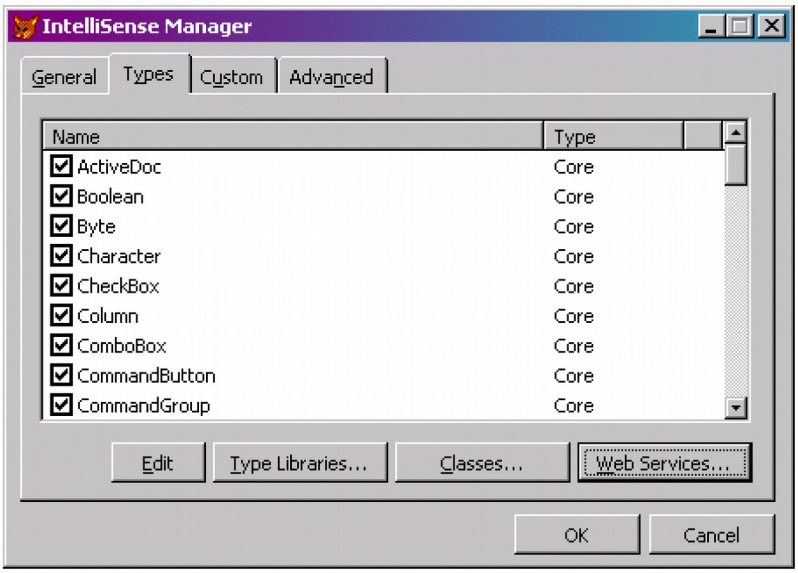
```
<soap:address location=
  'http://www.yourwebsite.com/sns/NewsService.ASP' />
```

where "sns" is the name for my virtual directory.

6. Test

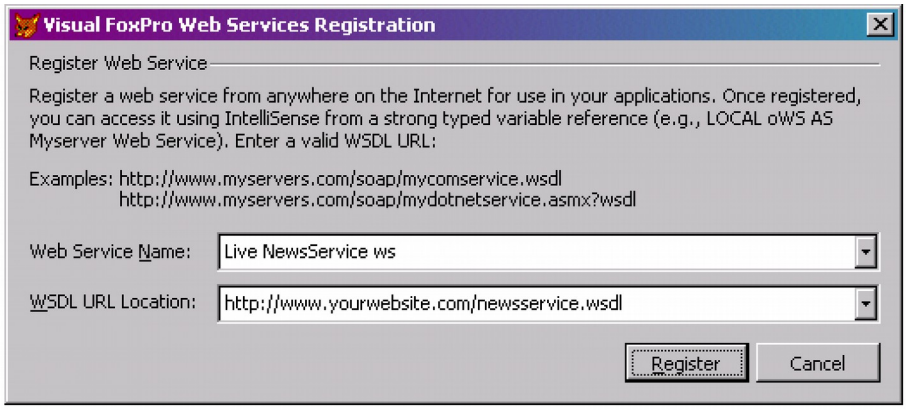
OK, so the NewsService Web Service is installed and ready to go on your live server. White knuckle time. Let's see if it really works.

Pop back over to your development box and fire up Visual FoxPro 7.0. Time to register the Web Service so you can consume it. Bring up the IntelliSense Manager (under the Tools menu) and click on the Types tab, as shown in Figure 2.



WS3A_02a.TIF
Figure 2. The Types tab of the IntelliSense Manager.

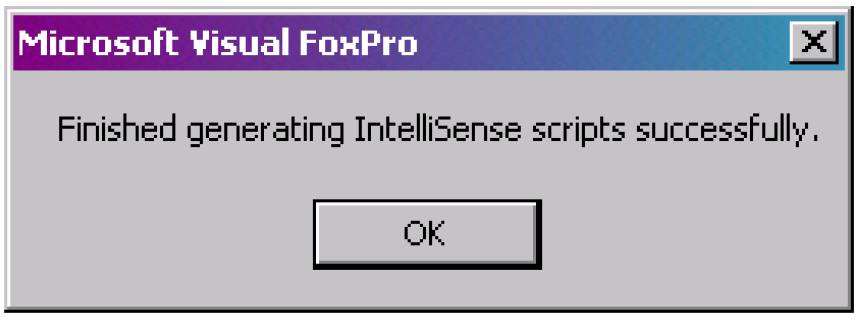
Click on the Web Services button to bring forward the Visual FoxPro Web Services Registration dialog. Enter a Web Service Name and specify the URL for the WSDL on your live server, as shown in Figure 3.



WS3A_03a.TIF
Figure 3. Entering the URL for your live Web Service.

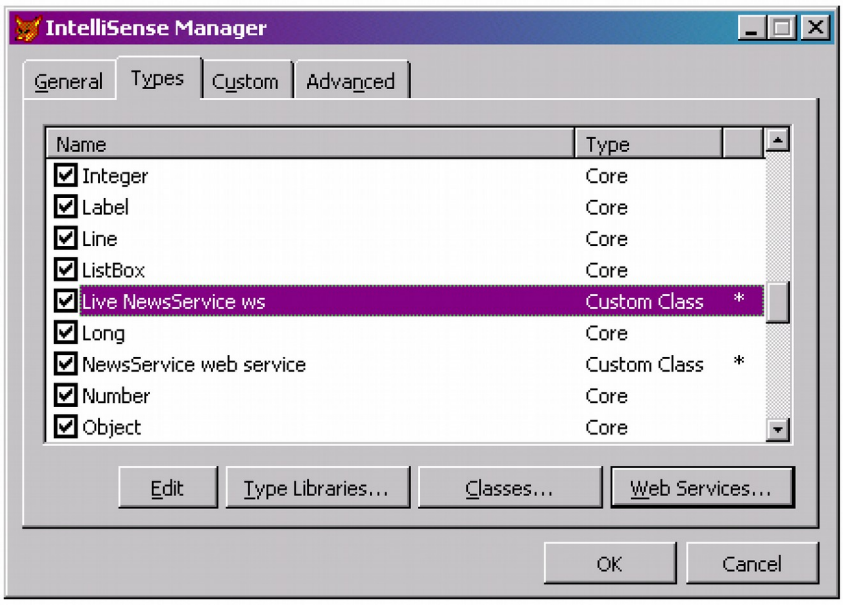
If you're using your development box to test your live Web Service, you may run into a problem. You've probably given your live Web Service a name when you published it on your development box to create the WS?L files – and being a logical kind of person, that name made sense at the time. Now that you want to consume your Web Service, it may be tempting to use a logical sounding name to name it – the same name that you used earlier. Fortunately, the VFP Web Services Registration wizard will warn you if you're about to reuse the same name.

Once you've registered the service successfully, you'll get a dialog like that shown in Figure 4.



WS3A_05a.TIF
Figure 4. Dialog indicating that your Web Service has been registered properly.

After you click "OK" in the confirmation dialog of Figure 4, you'll be returned to the IntelliSense Manager with a new type available, as shown in Figure 5.

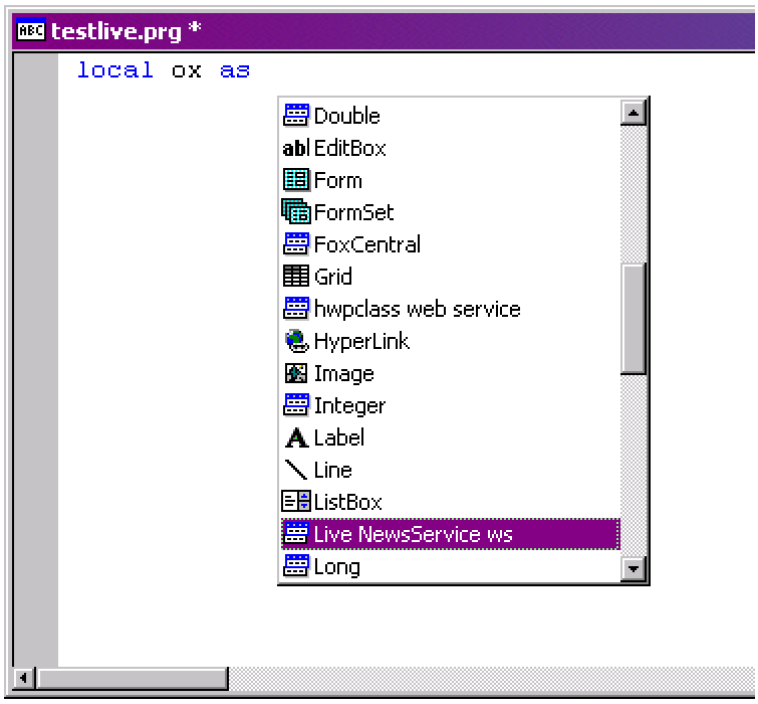


WS3A_06a.TIF
Figure 5. After registering, your live Web Service will have an IntelliSense entry.

Now it's time to test. Create a test program file in VFP on your development box, and enter

```
local ox as
```

The IntelliSense manager will display available types, as shown in Figure 6. Note that "Live NewsService ws" is listed as one available choice. That's why you want to go through the business of using the IntelliSense Manager.



WS3A_08a.TIF
Figure 6. Registering your live Web Service on your development box will allow it to display as an IntelliSense type.

Selecting Live NewsService ws in the combo box will result in a whole slew of code being entered into your test program file, like so:

```
local ox as Live NewsService ws
LOCAL loWS
loWS = NEWOBJECT("Wsclient",HOME() + ;
  "ffc\_webservices.vcx")
loWS.cWSName = "Live NewsService ws"
ox = loWS.SetupClient( ;
  "http://www.yourwebsite/sns/newsservice.wsdl", ;
  "NewsService", "NewsServiceSoapPort")
```

Now enter

```
m.lcNews = ox.GetNews("2001/11/1")
messagebox(m.lcNews)
```

and run your program. You'll get a dialog displaying the records from NEWS.DBF as in prior articles.

Reprise: Sure, it works for you, but…

Well, as with previous articles, it doesn't work for me either. Not the first time, at least. Or the second time, either. Why do you think this article is four months late? Fortunately, this time most of the mistakes had to do with the process itself, not dopey little syntax gotchas. Once you learn what files go where and what steps to follow, things work pretty smooth. But there are still a few possible problems. Let's look at them.

First, you might run into any host of errors as described in my first two articles – about not being able to find NEWS.DBF, or OLE errors like the following:

```
OLE error code 0x800a12be: Unknown COM status code
OLE error code ox80070057: The parameter is incorrect
```

The solutions to these are the same as described in my earlier articles – no need to repeat those answers again. The only difference is that you'll have to fix your errors and then replace your DLL (and maybe your WS?L files) on your live server.

Another problem you might run into is unexplained or nonexistent responses from your live server – where your server just doesn't answer a request. The only advice is to uninstall SOAP and then ensure that your listeners are configured correctly. Remember the joke about the four software engineers with a flat tire, and one of them suggests that if they all get out of the car, run around for a while, and then get back in, maybe the problem will have fixed itself. Same thing here. As with many things software, this stuff is still far too immature to expect to work reliably. Sometimes you'll just have to get out of the car, wait, and get back in again.

Conclusion

That about wraps up this series on your first Web Service. It's been a long, arduous haul, and it would have been even longer without the help of the Fox team, Steven Black, Gary DeWitt, Kevin McNeish and Ted Roche. I don't know how mere mortals are supposed to do this on their own, but I'll reserve my caustic comments about how this stuff is too immature to be touted as the next big thing for another time. For the time being, be prepared to spend some extra time – lots of extra time – as you struggle through your first few Web Service deployments. Hopefully the applications you're working will make the effort worth it!

Whil Hentzen is editor of FoxTalk.