April, 2003

Customize the Task Pane Manager

You can add panes provided by others or create and distribute your own

Tamar E. Granor, Technical Editor

In my last article, I explored the panes provided with the Task Pane Manager. In this article, I'll show you how to extend the tool by incorporating additional panes.

There are two ways to add panes to the Task Pane Manager. The simpler, by far, is to install a pane supplied by someone else. The hard way is to create your own.

Installing third-party panes

As VFP 8 comes into wide use, I expect most of the vendors of tools for VFP will create task panes for their products. While each framework and developer tool now has its own portal, some of them modal, the Task Pane Manager provides an opportunity to unify all the tools we use into a single interface.

Installing a pane supplied by someone else is a breeze. Open the Task Pane Manager and click the Options button. In the Options window, open the Task Pane Manager folder. Then choose Customize. On the page that appears (Figure 1), click Install Pane.

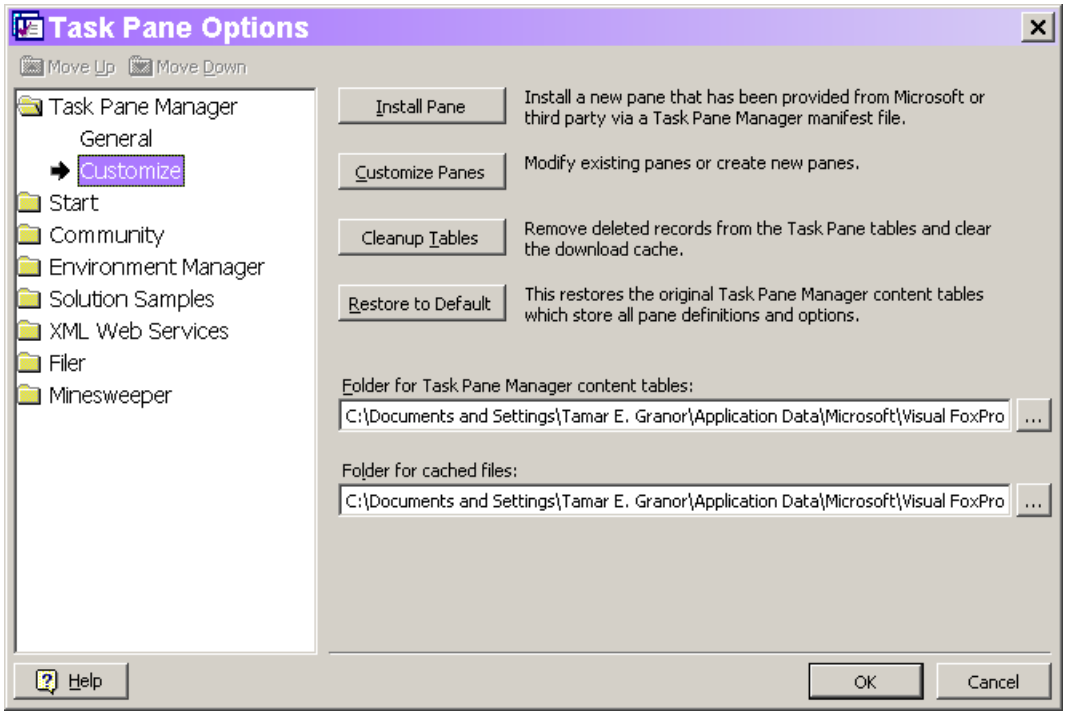

Figure 1: Adding and creating panes—This page of the Task Pane Options window lets you install third-party panes and create your own panes.

Task panes are distributed as XML files, so you're prompted to point to the appropriate XML file. Once you do so, it appears in the Task Pane Options window, and focus lands on that pane, so you can set any options it may have. The pane is also added to the Task Pane Manager, of course.

This month's Professional Resource CD includes Mortgage_Calculator.XML, the installation file for a simple pane that computes mortgage payments (Figure 2).

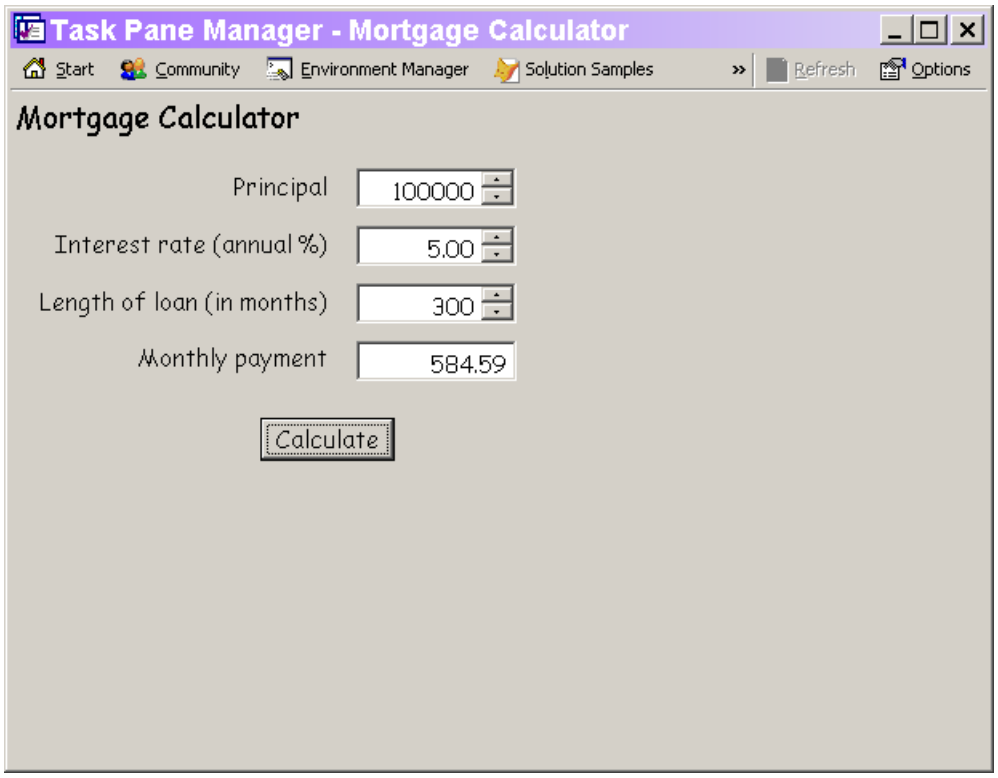

Figure 2: Mortgage Calculator pane—This pane, built with VFP code, computes loan payments.

Creating custom panes

You use the same page of the Task Pane Options window to start the process of creating your own panes, as well as to modify existing panes. In this case, click Customize Panes, which opens the Pane Customization window (Figure 3).

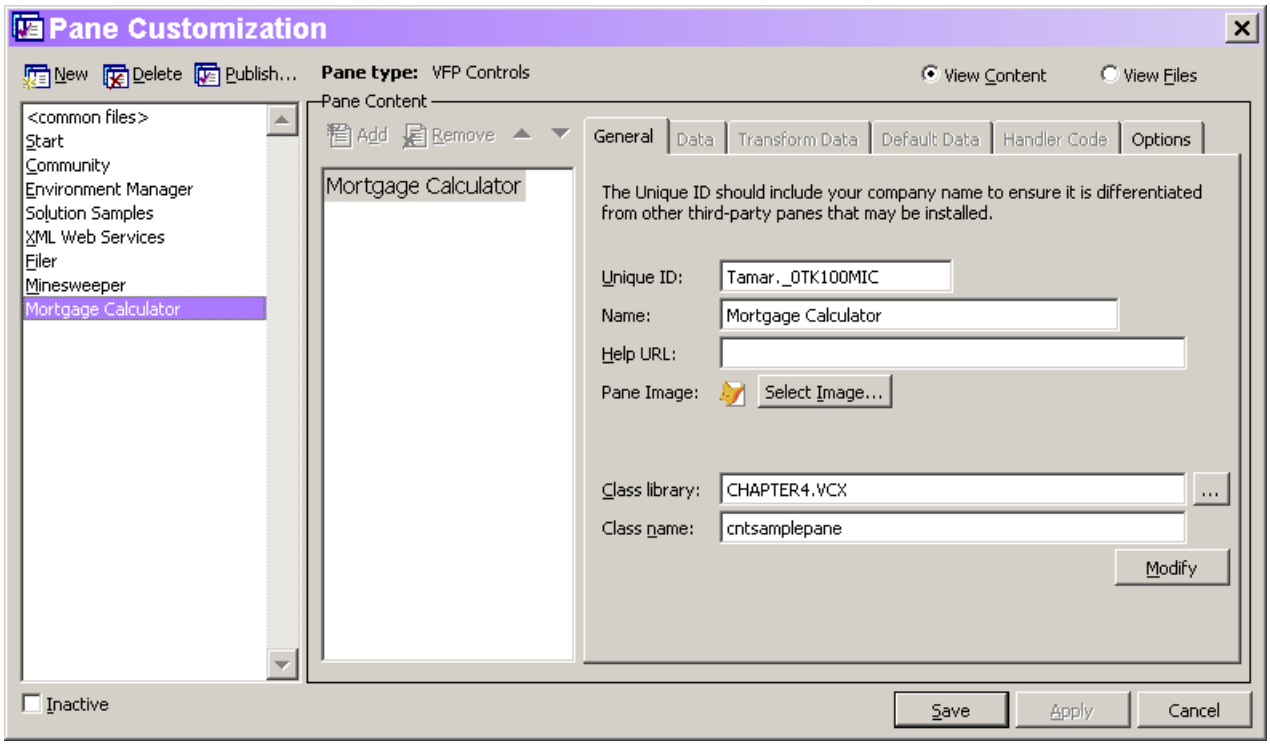

Figure 3: Creating and modifying panes—The Pane Customization window lets you modify existing panes and create new ones.

To create a new page, click the New button on the button bar. The Pane dialog (Figure 4) appears. In this dialog, you specify a "vendor" (your name or your company name), the name for the new pane, and the type of pane you're building.

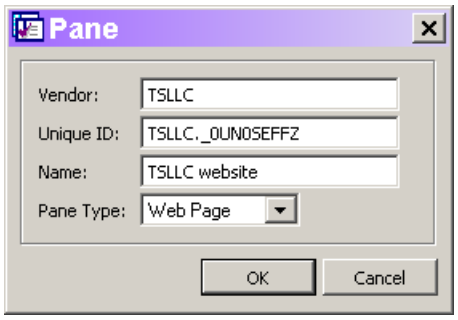

Figure 4: Creating a new pane—This dialog lets you specify basic information for a new task pane.

By default, the name you specify for Vendor is concatenated with the Unique ID to form the complete identifier for the pane. This combination increases the chances that every pane a user installs has a completely unique identifier. (The default value for Unique ID is generated using SYS(2015), but you can specify a different value if you want.)

The string you specify for Name is the pane's "friendly name." It appears in the Task Pane's button bar, as well as in the Task Pane Options window.

The tool supports four pane types: Web Page, HTML, XML, and VFP Controls. The type of a pane appears in the button bar of the Pane Customization window. (In Figure 3, the Mortgage Calculator pane is a VFP Controls pane.) We'll look at each type.

Web Page panes

The simplest type of pane to create is a Web Page. Web Page panes simply display a specified website in the Task Pane Manager. The only thing you need to specify for a Web Page pane is the URL of the website. You do so on the Data page of the Pane Customization dialog (Figure 5). Be aware that the appearance of this page changes depending on the type of pane you're creating and on choices you make on the page.

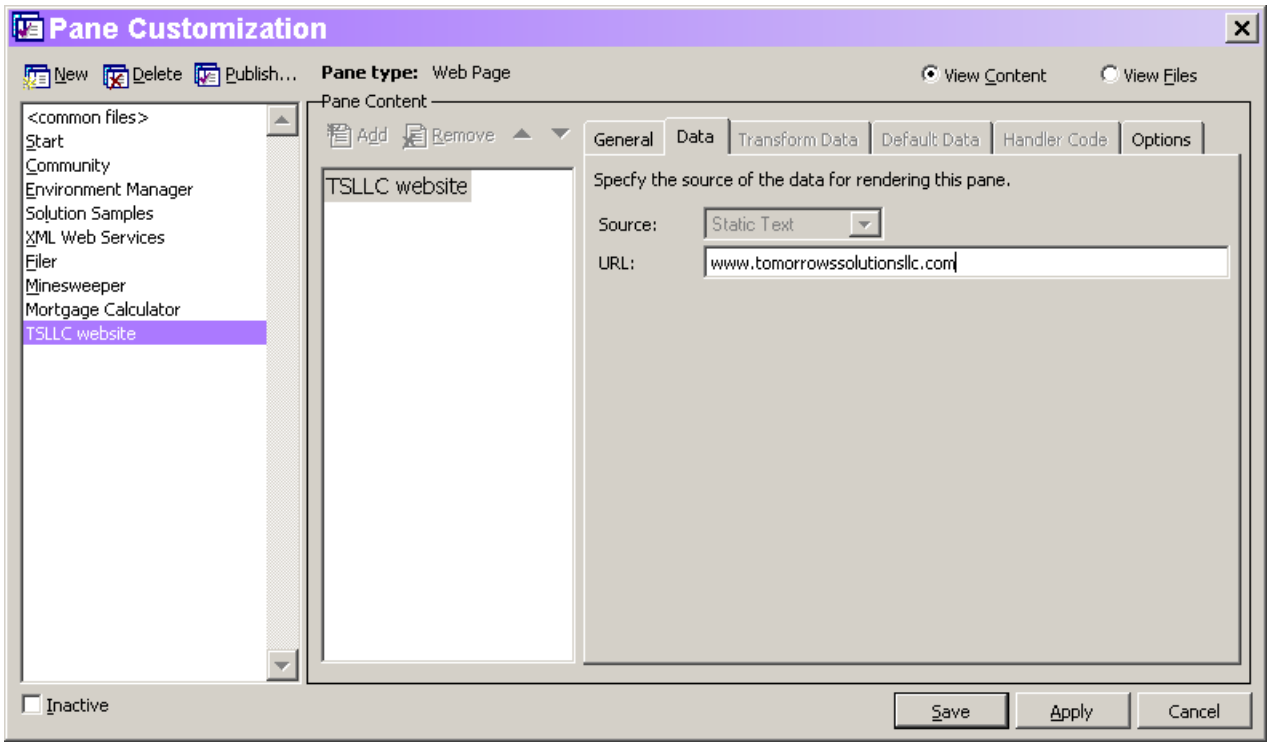

Figure 5: Specifying a Web Page pane—For this type of pane, the only required information is the URL of the website to display.

There's one other item you may want to specify for this type of pane, as well as for other panes—that's the image to associate with the pane. The image appears next to the pane name in the button bar.

You specify it on the General page of the Pane Customization window (see Figure 3).

The pane defined in Figure 5 is included on this month's PRD as TSLLC_website.XML.

VFP Controls panes

For VFP developers, panes based on VFP code are clearly the next easiest to create. The Mortgage Calculator on this month's PRD is a VFP Controls pane—we'll look at its code to see how to build this sort of pane.

For a VFP Controls pane, you specify a class and class library. The class should be subclassed from the PaneContainer class in the FoxPane.VCX class library that's part of the TaskPane project. (Unzip XSource.ZIP in the Tools folder, and then look in the VFPSource\TaskPane folder to find that .VCX.) While you probably can create a pane based on a different class, using PaneContainer ensures that all the necessary methods are present and that default behavior is provided.

When you subclass PaneContainer, add the controls and code necessary for the functionality you want the pane to have. The Mortgage Calculator pane has a few labels and spinners to represent the principal, interest and length of the loan, plus a textbox to contain the payment amount and a button to perform the calculation. There's one custom method, ComputePayment, which does the calculation (using the VFP's Payment() function). The Init method of the pane calls ComputePayment, as does the Click method of the Calculate button.

Defining Options

Panes can have options, items that the user specifies that change the behavior of the pane. For example, the Minesweeper pane has an option for the size of the matrix. (To see it, open the Task Pane Options window, and click on Minesweeper.) The Mortgage Calculator pane has an option to indicate whether the interest rate specified is annual or monthly. Figure 6 shows this option in the Task Pane Options window.

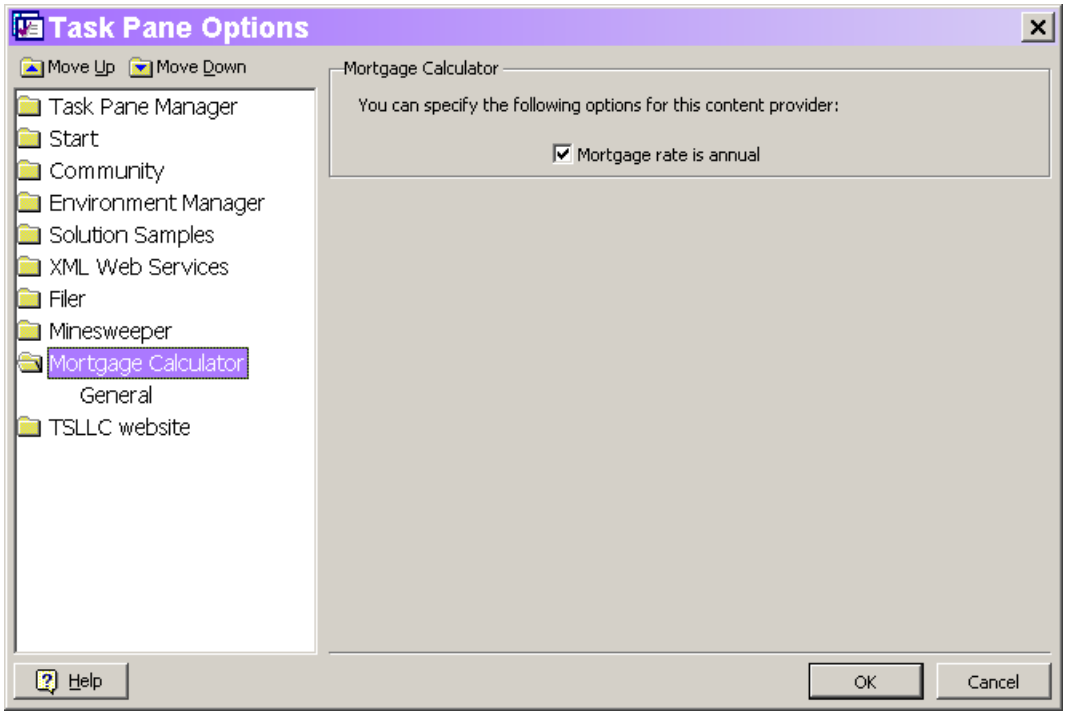

Figure 6: Pane option—When creating a custom pane, you can define options that the user can specify when using the pane.

Options are specified on the Options page of the Pane Customization window (Figure 7). To add an option item, click New; the New Option dialog (Figure 8) appears. Specify a name for the option, the type of control it should use (textbox, checkbox, spinner, or password textbox) and a caption for that control.

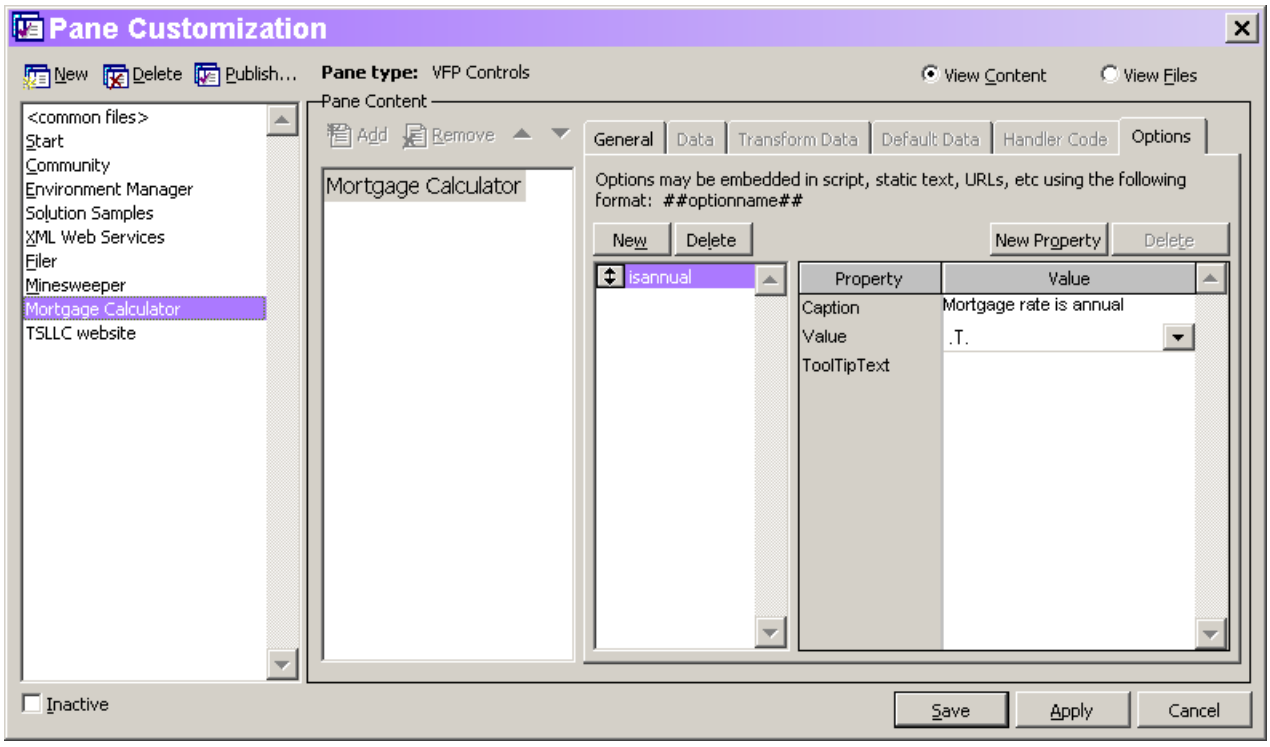

Figure 7: Pane options—The Options page of the Pane Customization dialog defines options for the pane. The options specified here appear on the Options page for the specified pane.

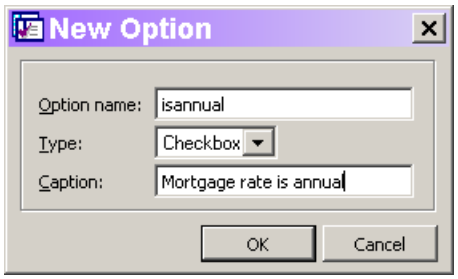

Figure 8: Adding options—The New Option dialog lets you specify the name of an option and the control used for it.

Once an option has been defined, you can modify the control used to display it by setting properties of the control. The properties grid on the right-hand side of the Options page is initially populated with a few properties you're likely to want to change. You can add other properties to the list by clicking the New Property button above the grid. Then specify the name of the property and it's added to the grid. The Value column of the grid is editable, so you can set the properties.

Using Options

Of course, defining options wouldn't be very useful if there weren't a way to use them in figuring out what appears in a pane. There are actually two ways to access the value of a property; which one you use depends on the situation.

VFP code that you write for a pane (whether in a subclass of PaneContainer or in one of several other places used for HTML and XML panes) generally receives one or more parameters. The one you need for accessing option values is oContent, an object reference to an object based on the Content class defined in FoxPaneContent.PRG. The Content class has a GetOption method—pass it the name of the option and it returns the current value. (Option values are stored between VFP sessions.)

In the Mortgage Calculator pane, the OnRender method (inherited from PaneContainer) has this code to get and apply the value of the IsAnnual option:

```
LPARAMETERS oPane, oContent
LOCAL cResult
cResult = oContent.GetOption("isannual")
This.lIsAnnual = IIF(cResult = ".T.", .T., .F.)
IF This.lIsAnnual
  This.lblInterest.Caption = "Interest rate (annual %)"
ELSE
  This.lblInterest.Caption = "Interest rate (monthly %)"
ENDIF
```

The second way to use an option's value applies when you're using the option in the specification for the pane. For example, you might define a generalized Web Page pane that lets you specify a URL and then displays that page. To do so, you'd add an option called url to the pane. Then, on the Data page, in the URL textbox, specify:

```
##url##
```

You can see an example of this approach in the Community pane. Look at the Data page for the Wiki section of the page. The call to the Wiki's web service passes the parameter ##daysold##.

HTML and XML Panes

The last two types of panes, HTML and XML, have a lot in common. Both display their content in an Internet Explorer ActiveX control. The

principal difference between the two is that XML panes must include an XSL style sheet to convert the content to HTML.

The biggest difference between HTML and XML panes and the other types is that HTML and XML panes generally have most of their content in sections beneath the main pane. Web Page and VFP Controls panes don't support sections. You can tell whether a pane has sections by looking at the Task Pane Options window. Each section appears as a checkbox on the General options page for the pane, so you can control each separately. For example, the Solution Samples pane contains two sections: Solution Samples and Add-in Links.

As an example, we'll build a pane that displays the weather forecast for a number of different cities. The forecast information will be retrieved using a web service whose definition is found at http://hosting001.vs.k2unisys.net/Weather/PDCWebService/WeatherServices.asmx?WSDL . (You need to register this web service before creating or using the pane. See the sidebar for details.) The Weather Report pane is included on this month's PRD for you to install as described earlier in the article.

Once you've created the pane, you add sections by clicking the Add button in the Pane Content listing of the Pane Customization window. The new section shows up as a leaf node under the pane; you can change its ID and name on the General page, as in Figure 9. In the Weather Report pane, we'll use a section for each city we're interested in.

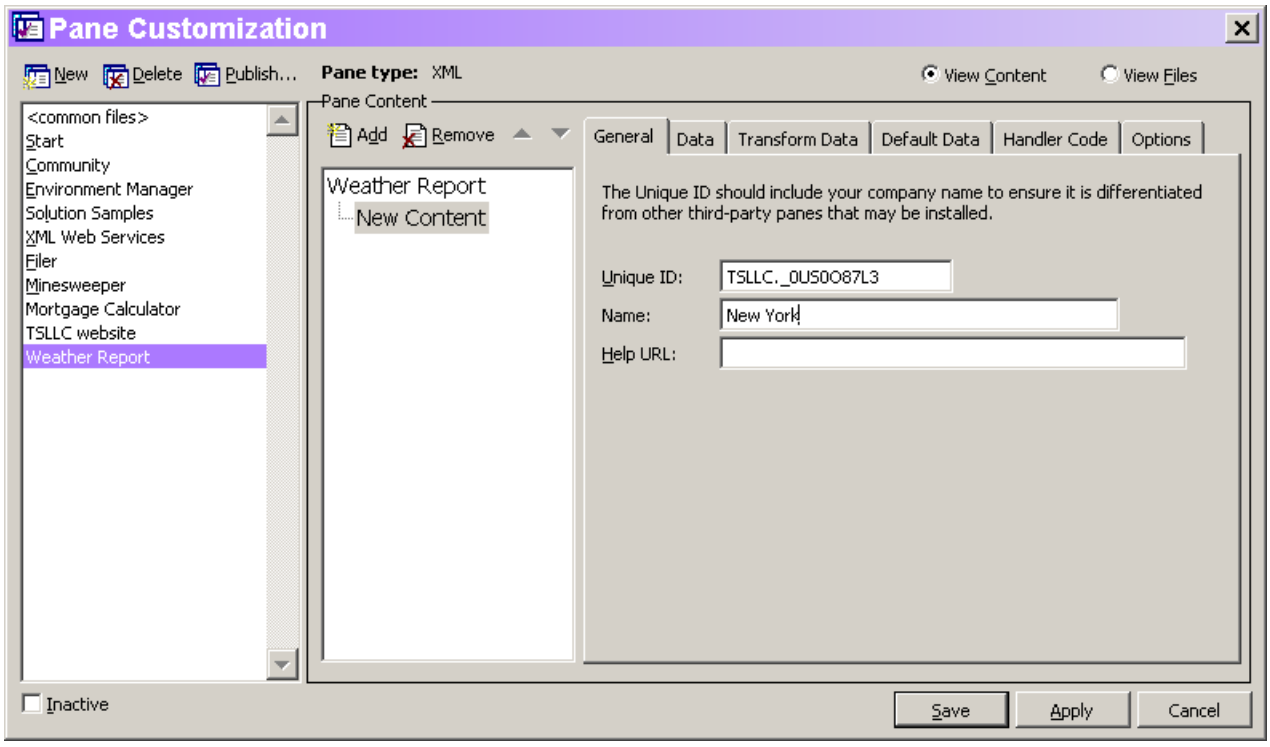

Figure 9: Adding subsections—The Add button adds a section to the pane you're editing. Each section has its own unique ID. Display of sections can be controlled individually through the Task Pane Options window.

For an XML or HTML pane, the Data page specifies the raw source of the pane. It may be HTML or XML. When a pane has sections, the Data page for the pane itself ("Weather Report" in Figure 9) contains information for the pane as a whole. Include the string "<!-- CONTENT -->" to indicate where the content from the sections should be placed. The data for the sections is concatenated and substituted for the Content comment.

The data for the pane as a whole and for each section can come from a variety of sources, shown in Table 1. The Help topic "Data Tab, Pane Customization Dialog Box" has more information on these choices.

Table 1. Sources for XML and HTML data—The data for a pane or its sections can be specified in a variety of ways. The same choices apply for specifying transformation of data.

Source Type	Description
Static Text	The content or transformation is specified as XML, HTML (or, for transformations, XSL) in an edit box on the page. A Modify button appears to let you open an editing window.

Source Type	Description
URL	A URL to the content or transformation is specified in a text box on the page.
Script	The content or transformation is returned from VFP code. The VFP code is specified in an edit box on the page. A Modify button appears to let you open an editing window.
File	The content or transformation is contained in a file, which you specify in a text box on the page.
Web Service	The content or transformation is supplied by calling a web service. You specify the details for the call on the page.

For the Weather Report pane itself, the default data for an XML pane works with just a small addition. Here's the Static Text specified:

```
<?xml version='1.0' encoding='utf-8' standalone='no'?>
<?xml:stylesheet type="text/xsl" href="weather.xsl"?>
<VFPData>
<!-- CONTENT -->
</VFPData>
```

The key change is the addition of a stylesheet reference. The specified stylesheet (weather.xsl) is used to convert the XML weather data to HTML for display. We'll take a look at it a little later on.

The sections each need to call the web service's GetWeather method, passing an appropriate zip code. Although many web service methods return an XML string as the result, the GetWeather method has a more complex result, which is returned as an object. This requires additional processing, so using a data source of Web Service isn't possible. Instead, VFP code calls the web service and processes the result. Since each section needs the same code, it's parameterized:

```
#define WEBSERVICE_URL ;  "http://hosting001.vs.k2unisys.net/Weather/;
PDCWebService/WeatherServices.asmx?WSDL"
#define ERROR_NORETRIEVE_LOC     ;
   "Unable to retrieve content at this time."
#define ERROR_WSCONNECT_LOC ;
   "Unable to connect to Web Service: "
LPARAMETERS oContent, nZipCode
LOCAL cXML, oXML
LOCAL oProxy
LOCAL oException
cXML = .NULL.
```

```
oProxy =Createobject("mssoap.soapclient30")
TRY
   oProxy.mssoapinit(WEBSERVICE_URL)
   TRY
      oXML = oProxy.GetWeather(nZipCode)
      * Rudimentary checking for right return type
      IF oXML.Length>0 AND ;
         NOT ISNULL(oXML.Item(0).ParentNode)
         cXML = oXML.item(0).ParentNode.xml
      ENDIF
      * Remove XMLNS info
      cRemove = STREXTRACT(cXML,"<GetWeatherResult",'>')
      cXML = STRTRAN(cXML,cRemove,"")
   CATCH TO oException
      m.oContent.LogError(.NULL., ERROR_NORETRIEVE_LOC, ;
         .NULL., .NULL., oException.Message)
   ENDTRY
CATCH TO oException
   m.oContent.LogError(.NULL., ;
      ERROR_WSCONNECT_LOC + WEBSERVICE_URL, .NULL., ;
      .NULL., oException.Message)
ENDTRY
RETURN cXML
```

If only one section needed this code, we could specify a Data Source type of Script and paste the code into the edit box, substituting the appropriate zip code. Since every section of the pane uses it, it makes more sense to store the code as a PRG and call it from each pane. To add the PRG to the pane, choose the View Files option button in the Pane Customization window, which switches to the view shown in Figure 10. Use the Add button to add the PRG file. In Figure 10, the program has already been added; it's called GetWeather.PRG.

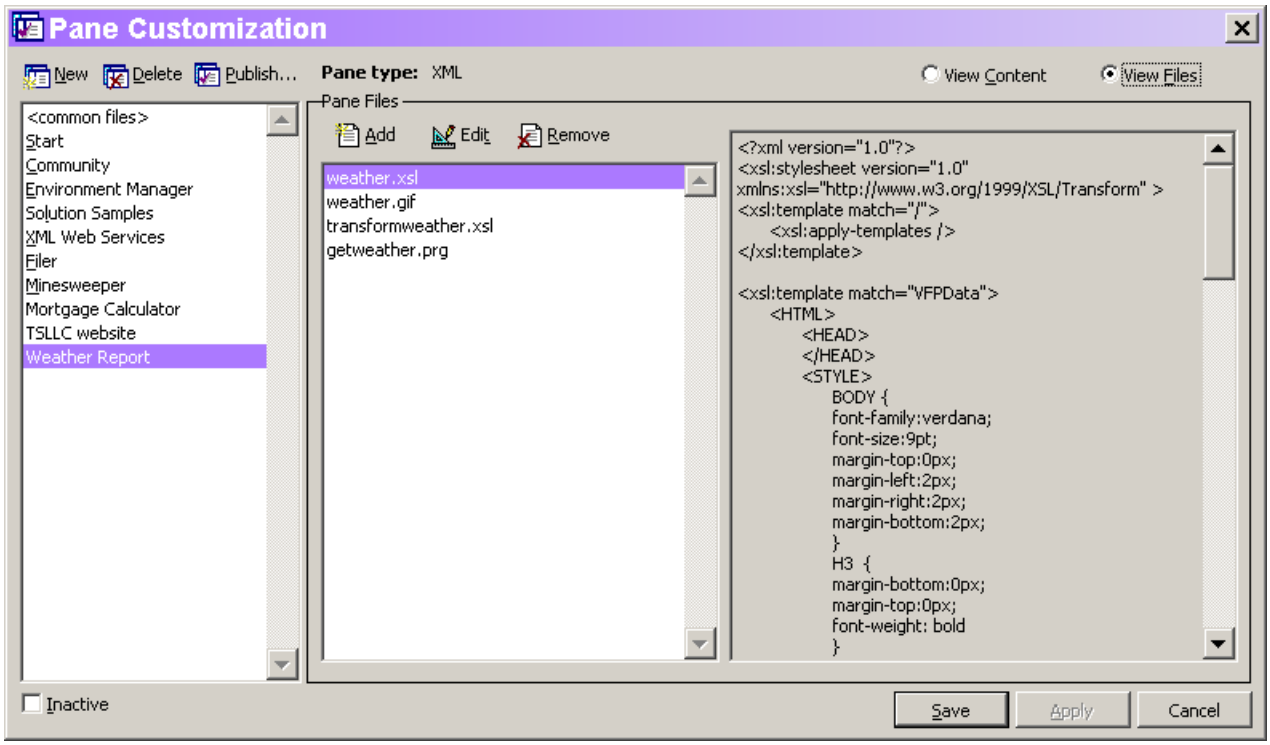

Figure 10: Pane Files—Choosing the View Files option button lets you see the files associated with (and stored with) your pane.

Each section of the Weather Report pane needs to make a call to the GetWeather program. To do this, specify the Data source as Script, and then use code like this (for New York) as the script:

```
LPARAMETERS oContent
RETURN GetWeather(oContent, "10001")
```

Note the oContent parameter. Scripts used on the Data page must be prepared to receive a single parameter, an object reference to the pane's content. In this case, we pass that parameter along to GetWeather, which uses it in case of a problem with the web service.

In addition to any manipulation you perform in the code that creates the XML data, you have another opportunity to process the XML, using the Transform Data page. You can specify no transformation, an XSL transformation, or a VFP script. You have the same choices of source for either XSL or a VFP script as for the data itself, those listed in Table 1.

For the Weather Report pane, the XML returned from the GetWeather method needs to be tweaked a little before it's combined with the results of the other sections. The main change is to eliminate a level of nesting. The GetWeather method wraps the whole set of weather

forecasts in a <DayForecast> tag, then puts the individual forecasts within <DailyForecast> tags within that tag. In addition, the method result includes some information we're not interested in, such as an abbreviation for the overall forecast.

As with the call to the method, every section needs to perform the same XSL transformation, so a source type of File is used and the file TransformWeather.XSL is specified. Here's the contents of that file:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
>
<xsl:output method="xml" indent="yes"
omit-xml-declaration="yes" standalone="no"/>
<xsl:template match="/">
   <xsl:apply-templates />
</xsl:template>
<xsl:template match="GetWeatherResult">
   <WeatherResult>
   <ZipCode><xsl:value-of select="ZipCode"/></ZipCode>
   <CityShortName><xsl:value-of select="CityShortName"/></CityShortName>
   <Time><xsl:value-of select="Time"/></Time>
   <Sunrise><xsl:value-of select="Sunrise"/></Sunrise>
   <Sunset><xsl:value-of select="Sunset"/></Sunset>
   <CurrentTemp><xsl:value-of select="CurrentTemp"/></CurrentTemp>
   <xsl:apply-templates select="DayForecast"/>
   </WeatherResult>
</xsl:template>
<xsl:template match="DayForecast">
   <xsl:apply-templates select="DailyForecast"/>
</xsl:template>
<xsl:template match="DailyForecast">
   <DailyForecast>
   <Day><xsl:value-of select="Day"/></Day>
   <Forecast><xsl:value-of select="Forecast"/></Forecast>
   <Abbrev><xsl:value-of select="Abbrev"/></Abbrev>
   <HighTemp><xsl:value-of select="HighTemp"/></HighTemp>
   <LowTemp><xsl:value-of select="LowTemp"/></LowTemp>
   </DailyForecast>
</xsl:template>
</xsl:stylesheet>
```

The pane itself (Weather Report) has no transformation specified. However, we do need to convert the complete XML result to HTML. That's the role of the Weather.XSL stylesheet specified in the Data section of the pane:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
>
<xsl:template match="/">
   <xsl:apply-templates />
```

```
</xsl:template>
<xsl:template match="VFPData">
   <HTML>
      <HEAD>
      </HEAD>
      <STYLE>
         BODY {
         font-family:verdana;
         font-size:9pt;
         margin-top:0px;
         margin-left:2px;
         margin-right:2px;
         margin-bottom:2px;
         }
         H3  {
         margin-bottom:0px;
         margin-top:0px;
         font-weight: bold
         }
         TD   {font-size:9pt}
         a:link {  color: #0033CC;text-decoration: none}
         a:visited {   text-decoration: none}
         a:hover {  color: #CC0000;
                    text-decoration: underline}
         A {  text-decoration: underline; font-family:
              Verdana, Arial, Helvetica, sans-serif;
              color: #0066FF
         }
         TD.TableTitle
         {
         padding:2px;
         background-color:#8080C0;
         color:#FFFFFF;
         }
         A.toggle:link {  color: #000000;
                          text-decoration: none}
         A.toggle:visited {   text-decoration: none}
         A.toggle:hover {  color: #000000;
                           text-decoration: none}
         A.toggle {  text-decoration: none;
                     color: #000000}

      </STYLE>
      <BODY leftmargin="0" topmargin="0">
         <TABLE width="100%" cellspacing="0"
                cellpadding="0">
            <TR>
               <TD class="TableTitle" width="100%"
                   nowrap="nowrap">
               <h3>Weather Report for Selected Cities
               </h3></TD>
               <TD  align="left" valign="center">
                 <img src="weather.gif" alt="" /></TD>
            </TR>
            <TR>
```

```
               <TD colspan="2" height="10"></TD>
            </TR>
         </TABLE>
         <br/>
         <xsl:apply-templates select="WeatherResult"/>
      </BODY>
   </HTML>
</xsl:template>
<xsl:template match="WeatherResult">
   <table width="100%">
   <tr>
   <td class="TableTitle">
   <font size="3">Forecast for
   <xsl:value-of select="CityShortName"/>
    as of <xsl:value-of select="Time"/></font>
   </td></tr></table>
   <p>Current temperature:
      <xsl:value-of select="CurrentTemp"/></p>
   <table width="100%">
   <xsl:apply-templates select="DailyForecast"/>
   </table>
   <br/>
</xsl:template>
<xsl:template match="DailyForecast">
   <tr>
   <td><xsl:value-of select="Day"/></td>
   <td><xsl:value-of select="Forecast"/></td>
   </tr>
</xsl:template>
</xsl:stylesheet>
```

These settings give you everything you need for the Weather Report pane. In the version on this month's PRD, I've set up four cities (New York, Philadelphia, Redmond, and San Diego), but you can set up whatever list interests you. Figure 11 shows the Weather Report pane.

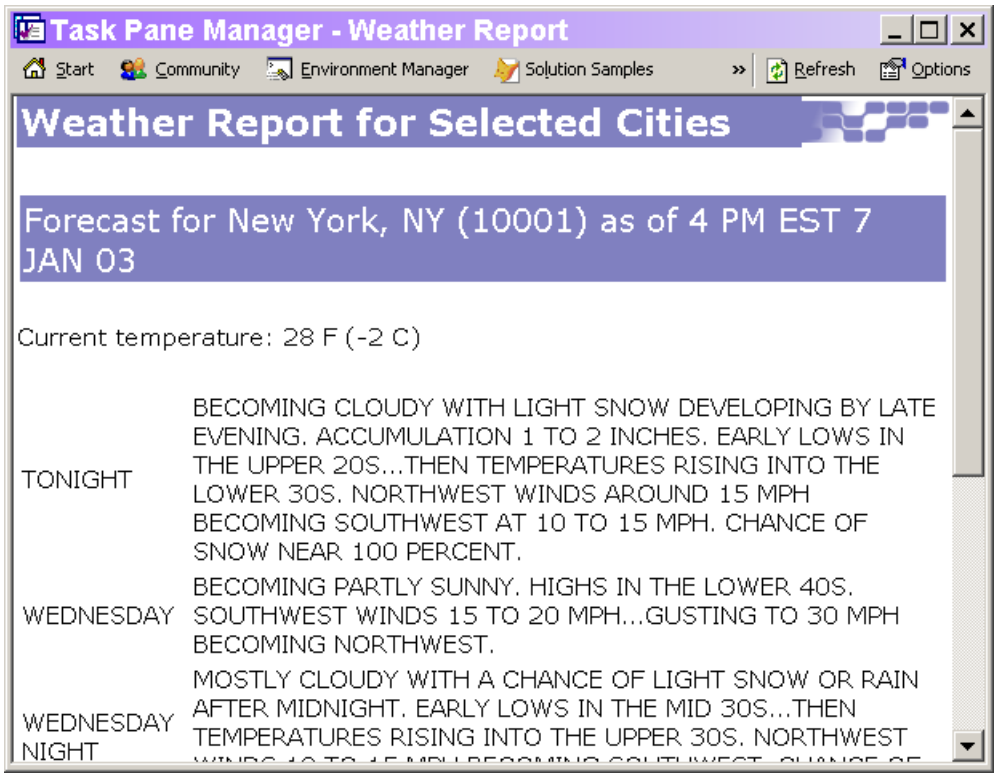

Figure 11: A custom XML pane—This pane shows the weather forecast for selected cities, using a web service that provides the information.

This example doesn't demonstrate all the features an XML or HTML pane can have. The Default Data page of the Pane Customization window lets you specify initial contents for a web-based pane, in case no web access is available the first time the pane is displayed. Once a pane has been used once, the contents are cached and those cached contents displayed if no web access is available.

Far more interesting is the Handler Code page. Not only can pane content be dynamic, but it can run code, navigate to other panes, open a browser, and pretty much anything else you can write code for. By convention, any URL in a pane that begins with the string "vfps:" (presumably, for "Visual FoxPro script") executes the handler code. The rest of the URL is parsed to indicate what action to take and to provide parameters for that action.

The Task Pane engine provides a number of script actions; they're shown in Table 2. For example, clicking on a hyperlink that has this URL opens a browser to the Advisor web site:

vfps:linkto?url=http://www.advisor.com/

Table 2. Predefined script actions—You can use these actions in your pane without writing any code.

Action	Purpose	Parameters received
gotopane	Switch to the specified pane	uniqueid=cPaneUniqueID
help	Display the specified help topic	id=cTopicID or topic=cTopicName
linkto	Open a browser displaying the specified web page	url=cUrl
message	Display a message box	msg=cMessage
options	Display the Task Pane Options dialog with a specified pane chosen	uniqueid=cPaneUniqueID
refresh	Reload the current pane	None

In addition, you can write custom code and define your own actions. Handler code is VFP code. It receives four parameters, shown in Table 3. When you click the Modify button on the Handler Code page to begin writing handler code, the window that opens contains the necessary parameter declarations, along with explanatory comments.

Table 3. Parameters to handler code—Code called by a "vfps:" link receives these parameters.

Parameter	Meaning
cAction	The action specified in the link, such as "gotopane."
oParameters	A collection listing the parameters passed in the link. The collection has a GetParam method to return the value of a specified parameter.
oBrowser	An object reference to the Browser ActiveX object in which the pane is displayed.
oContent	An object reference to the content of the pane.

A number of the panes provided with VFP 8 use handler code, using both the built-in actions and custom actions. Exploring those panes in the Pane Customization window is a good way to get a feeling for what's possible.

Publishing Panes

While you can create panes just for your own use, it's easy to distribute your panes to others. The tool allows you to create an XML file that others can install in their own Task Pane Manager. To publish a pane, highlight the appropriate pane in the Pane Customization window and click Publish. The Publish Pane dialog (Figure 12) appears to let you indicate exactly what portion of the pane and its associated files should be included in the XML file.

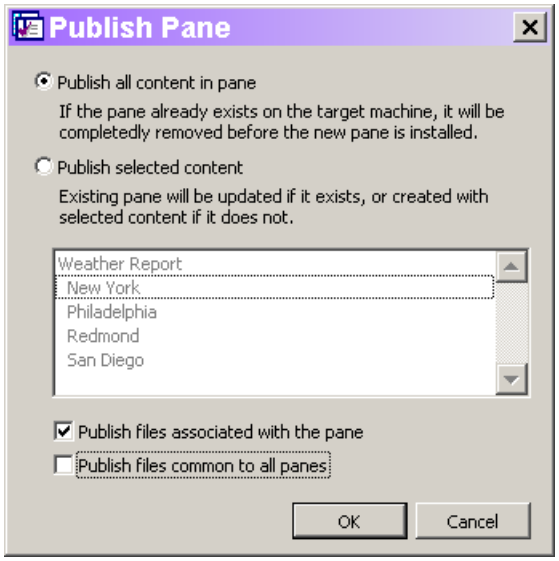

Figure 12: Publishing task panes—The Publish Pane dialog lets you specify which portions of the pane and which associated files are included in the XML file.

You have two choices regarding pane content. You can publish the entire pane, or only selected sections. For files, you have several options. Check Publish files associated with the pane to ensure that all the files used by the pane (such as TransformWeather.XSL in the Weather Report example) are included. Check Publish Files common to all panes to distribute files used by all panes, such as those involved in error reporting.

When you choose OK, you're prompted to specify a file name for the XML file. The default is the name of the pane with spaces replaced by underscores. For example, for the Weather Forecast pane, the default file name is weather_report.XML.

Under the Hood

The strategy used for task pane data is a little unusual, so it's worth taking a brief look at it. By default, task pane data is stored in a directory tree beginning in the TaskPane subdirectory of the user application data directory (specified by HOME(7)). You can change the data storage location in the Task Pane Options window. The discussion here assumes you're using the default location.

The TaskPane directory contains two tables and a subdirectory called PaneCache. The TaskPane table contains one record for each pane. The PaneContent table contains one record for each section of each pane, using the pane's unique id to link the records. Both tables store a fair amount of the pane's data in memo fields. For example, the TaskPane table has ClassLib and ClassName memos to store the class information for VFP Controls panes. Similarly, the Data memo of PaneContent stores the actual data for a pane or section, as specified on the Data page of the Pane Customization window.

The PaneCache directory is where things get interesting. It has a subdirectory for each pane, named using the pane's unique ID. The files that belong to that pane (such as GetWeather.PRG and Weather.XSL in the Weather Report pane) are stored in that directory. It's also used as a working directory for code executed by the pane. The most important consequence of this structure is that once you've added a file to a pane, modifying the original file doesn't affect the pane and modifying the code in the pane doesn't affect your original file on disk.

The Bottom Line

The Task Pane Manager is an incredibly powerful portal that lets you combine a wide variety of functionality into one container. Expect to see task panes for many of the third-party tools for VFP. Consider defining your own task panes for things you need all the time, or to share functionality with coworkers and others.

Sidebar: Registering Web Services

Registering a web service with VFP provides you with IntelliSense for the web service and makes it possible to create code to use the web service via drag-and-drop. In VFP 8, registering a web service is straightforward.

You can access the Web Services Registration dialog from the Web Services pane of the Task Pane Manager (click "Register an XML Web Service"), the My XML Web Services section of the Toolbox (click "Register"), or, as in VFP 7, the Types page of the IntelliSense Manager (click "Web Services"). Once the dialog opens, you can simply type or paste the WSDL URL for the web service into the combo box and click Register.

The dialog also provides search functionality if you're looking for a web service to perform a particular task.