Using the ArcIMS .NET Link

By Tim Bricker, ESRI Educational Services

ArcIMS is a multitiered software product for distributing interactive maps using standard Web protocols. A connector is required so that a Web server can access the ArcIMS Application Server and Spatial Server to create maps dynamically. ArcIMS provides various connector options: servlet; Java; ColdFusion; ActiveX; and, with the release of ArcIMS 9—.NET Link. This article will discuss the use of the .NET Link to create custom ArcIMS clients. ArcIMS 9 requires both .NET Framework 1.1 and .NET Framework 1.1 SDK to install the .NET Link and Developer Help feature.

Why Use the .NET Link?

With the .NET Link, code to interact with ArcIMS is written in a .NET language such as VB.NET or C#. Active Server Pages .NET (ASP.NET) is the technology for using .NET to create Web pages. This code, which is executed on the server side rather than on the client side (like JavaScript), gives the developer several advantages. First, only the results of the serverside code are actually sent over the Internet to the browser. This results in a thin client that takes less time to download to a Web site and requires less clientside processing.

The second advantage is that the code running on the server has access to the server's file system and databases that reside on the server. If the developer needs to create a Web site that adds new information to a database based on where the user clicked on a map, serverside processing would be required. The final advantage of ASP.NET over other Web development environments is the event driven abstraction provided by development environments such as Visual Studio .NET. Writing code in Visual Studio .NET is as simple as dragging a button onto a form and writing the serverside code that executes when the user clicks on the button. The details of how that form gets posted to the server so the code can run are handled by the abstraction.

Using .NET Link

Although .NET Link has several objects that allow the developer to send and receive ArcXML, as well as generate a few predefined ArcXML requests, the primary object is the ServerConnection. ServerConnection can be used to define a connection to the Application Server and the ArcIMS Service to send the request to and also to send the request. The code sample in Listing 1 uses the ServerConnection.

To send an ArcXML request, use the ServerConnection's send method:

```
conn.Send(sAXL)
```

The sAXL variable would be a string containing a complete ArcXML request. The AXLRequests object can create several predefined ArcXML requests, such as GET_SERVICE_INFO, GETCLIENTSERVICES, and GET_IMAGE, at full map extent.

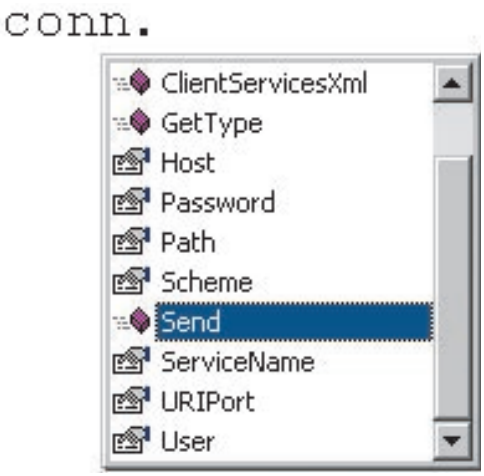
Generating and Parsing ArcXML

To use the .NET Link, the developer must create ArcXML requests and handle ArcXML responses. The *ArcXML Programmer's Reference Guide* provides guidance in parsing ArcXML. With .NET, many objects can help with XML generation and parsing. Strings can be generated using simple string concatenation or by using the StringBuilder object. To parse XML strings, an XMLDocument object can be used to represent the entire response and each element can be represented by a node. A node consists of the element, attributes, and all child elements. In the code sample in Listing 2, a ServerConnection object is used to send an ArcXML request and the response is loaded into an XMLDocument.

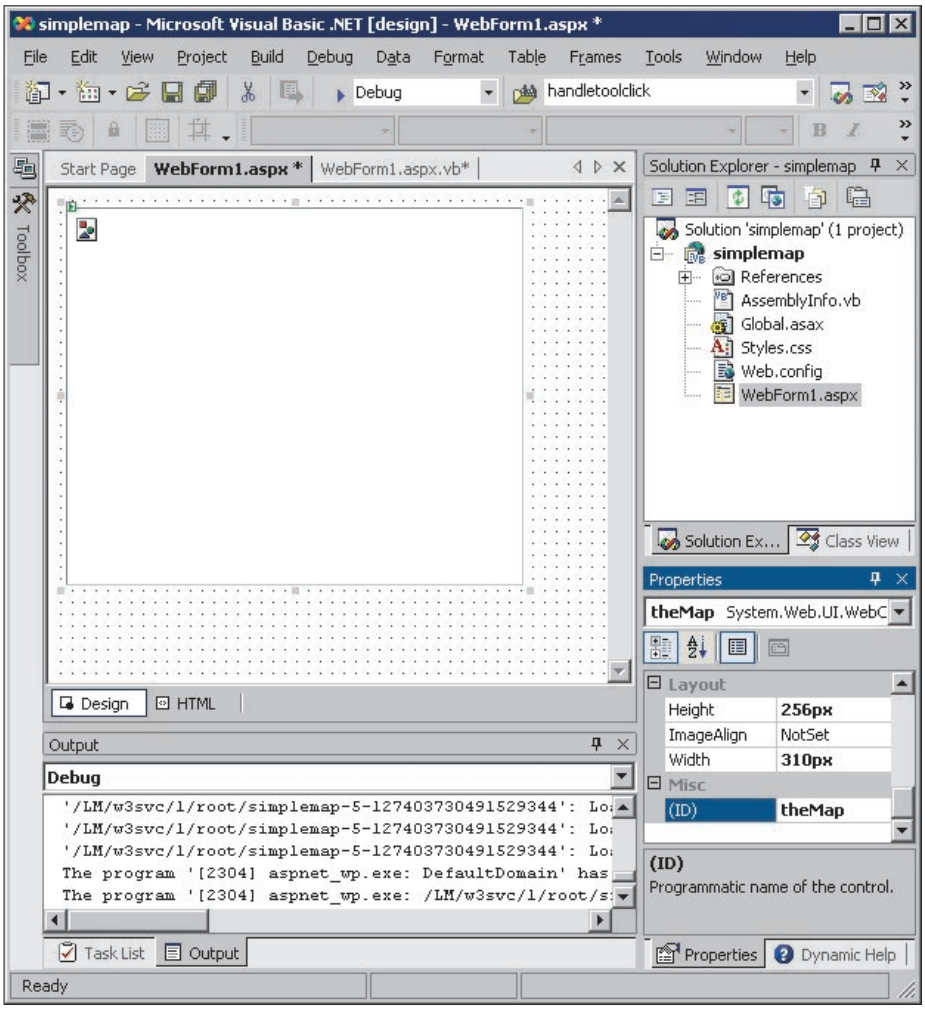
To retrieve a collection of a particular element, XMLNodeList can be used as shown in Listing 3. Once an XMLNodeList has been generated, a particular node can be accessed via the item property (the default property). Each node gives access to all child nodes (child elements) and attributes. If a GET_IMAGE request was sent, the code to retrieve the URL of the new image would look like Listing 4.

Creating a Simple Map With .NET Link

This section shows how to create a simple Web page that displays a map. This example assumes that ArcIMS is installed with the .NET Link and both an Image Service named sample and Visual Studio .NET are running. The sample service should reference data that covers the entire world (countries)



Use the ServerConnection's send method to send an ArcXML request.



To create an area to display the map, drag an image from the Web Forms category in the toolbox onto the form. Change the ID property of the map to theMap.

and is in geographic coordinates.

1. In Visual Studio .NET, create a new ASP.NET Web application. For this example, the code will be written in VB.NET.

2. This will allow the use of .NET Link in code.

3. To create an area to display the map, drag an image from the Web Forms category in the toolbox onto the form. Change the ID property of the map to theMap.

4. In the code behind page, create a function that will return the URL to a new map image using the code shown in Listing 5.

5. To create the new image and update the URL of theMap control, add the lines of code shown in Listing 6 to the Page_Load procedure. Test this new Web application. Clicking the Start button in Visual Studio .NET should display a map at the full extent of the world.

State Maintenance

The previous example did not include any tools for the user to interact with the map. When tools are added to a Web application, state maintenance becomes an issue. For example, adding a tool that changes the extent of the map by zooming will require that the code "remember" the previous extent of the map in order to zoom in from there. There are several ways to maintain the state of the map. These methods include hidden inputs, cookies, session variables, and ViewState. The ViewState records state information in the HTML returned to the client, so the server does not need to use resources to maintain state for each Web client as session variables do.

Using the BlueViewer

A sample that is included on the ArcIMS installation CD–ROM can be used as a starting point for developing .NET Link Web applications more quickly. This sample, the BlueViewer, has built-in basic functionality: fixed and dynamic zoom (i.e., zoom with a rubber band box), legend, large/small map toggle, recenter, print, and *x,y* coordinate display. The BlueViewer uses hidden inputs to store state information.

For More Information

For more information on ArcIMS 9 and .Net Link, see the white paper *ArcIMS 9 Architecture and Functionality,* available online at www.esri.com/arcims and *Customizing ArcIMS—Using .NET Link,* which is included in the ArcIMS ESRI Software Documentation Library

LISTINGS

```
Dim conn As New ESRI.ArcIMS.Server.ServerConnection("localhost", 5300)
connArcIMS.ServiceName = "sample"
```

Listing 1

```
Dim axlResponse As New System.Xml.XmlDocument()
axlResponse.LoadXml(conn.Send(sAXL))
```

Listing 2

```
Dim nodeOutput As System.Xml.XmlNodeList
nodeOutput = axlResponse.GetElementsByTagName("OUTPUT")
```

Listing 3

```
Dim url as string
url = nodeOutput(0).Attributes("url")
```

Listing 4

```
Private Function createMap( _
ByVal minx As Long, _
ByVal miny As Long, _
ByVal maxx As Long, _
ByVal maxy As Long) _
As String

Dim sAXL As String
sAXL = "<?xml version=""1.0"" encoding=""UTF-8""?>"
sAXL &= "<ARCXML version=""1.1"">"
sAXL &= "<REQUEST><GET_IMAGE><PROPERTIES>"
sAXL &= "<IMAGESIZE width="""" & theMap.Width.Value
sAXL &= """" height="""" & theMap.Height.Value & """/>"
sAXL &= "<ENVELOPE minx="""
sAXL &= minx & """ miny="""" & miny & """" maxx="""
sAXL &= maxx & """ maxy="""" & maxy & """" />"
sAXL &= "<LEGEND display=""false"" />"
sAXL &= "</PROPERTIES></GET_IMAGE></REQUEST></ARCXML>"
Dim conn As New ESRI.ArcIMS.Server.ServerConnection("localhost", 5300)
conn.ServiceName = "sample"
Dim axlResponse As New System.Xml.XmlDocument()
axlResponse.LoadXml(conn.Send(sAXL))
Dim imageURL As String
If axlResponse.GetElementsByTagName("OUTPUT").Count = 1 Then
  Dim nodeOutput As System.Xml.XmlNodeList = _
  axlResponse.GetElementsByTagName("OUTPUT")
  imageURL = nodeOutput(0).Attributes("url").Value
End If
Return imageURL
End Function
```

Listing 5

```
Dim url As String
url = createMap(-180, -90, 180, 90)
theMap.ImageUrl = url
```

Listing 6