# Parsing XML in Flash Using the XMLConnector

The following section only applies to people using Flash MX Professional 2004 as it uses the new XMLConnector component as well as bindings and schemas. Users without Flash Pro can still send and receive XML using the various techniques above although a lot more code is required. Next you will see how you can actually syndicate a person's blog (assuming they have a public XML file available) and display the results in a Flash form to create a crude news system. This example uses the XMLConnector component to connect to a remote XML file and display the headlines in a List component. When a user clicks on a headline a brief description will be displayed in a TextArea component. Since we're using the XMLConnector it is only possible to grab news from a single XML file. If you were using the WebServiceConnector component it would be possible to syndicate a remote Web Service at Full As A Goog which would allow you to syndicate several of their available blogs feeds and return the combined results.

**Step 1: Adding the XMLConnector component to the Stage**
Create a new Flash document and add a XMLConnector component on to the Stage. Since the XMLConnector component instance will not be visible once the Flash document is published it doesn't really matter where it is is placed on the Stage, or if it is even off the Stage altogether. Give the XMLConnector an instance name of xmlcon.

**Step 2: Adding the List and TextArea components to the Stage**
Drag an instance of the List and TextArea components on to the Stage. Give the List component an instance name of headlines_ls and the TextArea component an instance name of description_txt. Position them in roughly the same position as the image below, using your artistic talents as a guide.

Figure 1: Position the elements on the Stage.

**Step 3: Configuring the XMLConnector compoent instance**
Enter the URL to an XML file in the XMLConnector component, for our example we're using Jen's blog XML file from TrainingFromTheSource.com which is powered by Movable Type. With the XMLConnector instance selected on the Stage, enter the following URL into the URL property using either the Property Inspector or Component Inspector panel: http://www.trainingfromthesource.com/blog/index.rdf. You could use any other blog URL or XML file you wanted, but the details may differ slightly if the XML structures are different. In the Property Inspector set the direction to receive instead of send/receive as we're only fetching the remote file and not passing any values to it.

**Step 4: Importing an XML Schema**
The next part is where it begins to get a bit confusing. The XMLConnector is very useful as it allows you to build complex applications without having to write much ActionScript, if any at all. The only way the XMLConnector component knows the

structure of the XML file is by making you manually define the XML schema using the Schema tab in the Component Inspector panel or by making you import the XML schema from a file of sample data. Being as though we're lazy, we'll choose the path of least resistance/effort and choose to import a file and have Flash do the dirty work for us.

Before we can import the sample data, we need, well, sample data. The easiest way to do this is to download a copy of the live data, and use that as our sample data. It is important to note that even though you're downloading a copy of the most recent blog entries Flash will always fetch the most recent XML file from the URL you specified earlier, it won't use the sample data being imported. So, sample data, click *here* to open a new browser and view the remote XML file of blog entries. Save this file using File > Save As to somewhere easy to locate (such as your Desktop) and give it a filename of sampledata.xml or something easy to recognize. After the schema has been imported into Flash you can safely delete this file.

After the XML file has been saved on to your local computer, make sure the Component Inspector panel is expanded and click on the Schema tab. Near the left of this tab you'll see the Import a schema from a sample XML file button. Make sure you have the results entry selected in the pane and click on the Import schema button to bring up a file browser where you can locate the XML file you saved in the previous paragraph. Once you've selected the file click the Open button to close the Open window and import the schema. If all went well you should now see a whole bunch of new entries in the Schema tab with fun double headed arrows and happy @ symbols. With the results : XML entry still selected in the schema pane change the resd only property to true to change all the double headed arrows into single headed arrows. Congratulations, you have imported an XML schema.

**Step 5: Binding the XML schema to the List and TextArea components**
Now that the schema has been imported and the villagers rejoice at your beheading of the double headed arrows, you are able to bind the values from the XML file to the components on the Stage. With the XMLConnector component still selected on the Stage switch to the Bindings tab in the Component Inspector panel. Click the Add binding button near the upper left corner of the tab (looks like a + sign) to open the Add Binding window. In the Add Binding window scroll down near the bottom of the schema and select the item : Array entry and click the OK button to add the first half of a binding. You should now see results.RDF.item in the pane in the Bindins tab. With the results.RDF.item entry selected i the pane double-click on the empty cell beside the bound to row. Flash will now open the Bound To window where you can complete the binding. In the Component path pane select your List component then select dataProvider : Array entry from the Schema Locationpane. Click the OK button to complete the binding and close the window.

Add another binding for the blog description. Click the Add binding button again and this time select the description entry from the item array. Click OK to close the window and you should now see results.RDF.item.[n].description in the pane. Double-click the bound to column to open the Bound To window again and this time select the TextArea

component from the Compoent path pane and select text : String from under Schema location.

**Step 6: Adding a behavior to trigger loading the XML file**
If you were to test the Flash movie right now you would notice that nothing happens, which makes for a fairly uninteresting tutorial. The problem is that the XMLConnector component doesn't do anything until you tell it to actually do something. There are two ways to do this, by typing a single line of ActionScript to trigger the XMLConnector component or you can add a behavior which adds the single line of code for you. It is probably faster to type the code yourself, but since we are lazy, automation rules.

Click on the main Stage so the behavior is added to Frame 1 instead of to our XMLConnector component. Make sure the Behaviors panel is open and click the Add Behavior button (again, looks like a + sign) to open the menu and add our behavior. From the menu select Data > Trigger Data Source to open the Trigger Data Source window. Select your XMLConnector component from the pane, make sure the Relative radio button is selected (it will work either way, but Relative addressing is always better than using Absolute referencing). Click OK to close the window and return to your Flash document. You should now see three (3) lines of code in the Actions panel, two of which are comments generously supplied by Macromedia and the final line of code triggers the XMLConnector.

**Step 7: Testing the Flash movie and complaining that it looks "ugly"**
At this point you should be able to test the Flash movie in the authoring environment by pressing Ctrl+Enter (for PC users) and see that the List component contains a bunch of entries and everything looks ugly. Clicking on the entries in the List component brings up a description in the TextArea component, but they're likely to be the same for each entry. Yeah.

A couple minor tweaks are still needed to make the crude blog interface usable. The first change is to only display the blog's title in the List component instead of a combination of the URL and every other child node in the XML packet. This is actually fairly simple, click again on the XMLConnector component and make sure the Binding tab is selected in the Component Inspector panel. Click on the results.RDF.item entry. Below the bound to row in the grid below the pane is the formatter row. Set the formatter to Rearrange Fields, and double the empty cell in the formatter options row and type label=title in the Fields Definitions text field. Click OK to close the window and return to Flash.

The last step is to modify the binding for results.RDF.item.[n].description. The description should currently be "broken" and always only display the description for the first blog entry. To fix this we can tell the description that it should show the text for the currently selected item in the List component, which again is an easy fix. With the XMLConnector component still selected on the Stage, select the results.RDF.item.[n].description entry in the Bindings tab of the Compoent Inspector panel. The problem lies in the entry for Index for 'item' row, which is set to 0 by default. Double click in the empty cell beside the Index for 'item' row to open the Bound Index

window. In this window, select the List component from the left pane and select selectedIndex : Number from the right pane. Click OK to close the window and return to the Stage.

**Step 8: Testing the Flash movie again and complain that it still looks "ugly"**
Now if you test the movie again it should display the title in the List component and the blog entry description in the TextArea below. Congratulations, you're finished. You successfully (hopefully) managed to syndicate an external XML news feed and display the contents on the Stage without having to write a single line of ActionScript.