macromedia®

FLASH™ MX

2004

Data Tutorials

Acknowledgments

Director: Erick Vera

Project Management: Julee Burdekin, Erick Vera

Writing: Jay Armstrong, Jody Bleyle, Mary Burger, Francis Cheng, Jen deHaan, Stephanie Gowin, Phillip Heinz, Shimul Rahim, Samuel R. Neff

Managing Editor: Rosana Francescato

Editing: Mary Ferguson, Mary Kraemer, Noreen Maher, Antonio Padial, Lisa Stanziano, Anne Szabla

Production Management: Patrice O'Neill

Media Design and Production: Adam Barnett, Christopher Basmajian, Aaron Begley, John Francis

Second Edition: June 2004

Macromedia, Inc.
600 Townsend St.
San Francisco, CA 94103

CONTENTS

Web Service Tutorial: Macromedia Tips (Flash Professional Only) 5

Connect to a public web service 5
Create a user interface and bind the components with the web service 7

XML Tutorial: Timesheet (Flash Professional Only) 11

Create the user interface 12
Edit the data 16

XUpdate Tutorial: Update the Timesheet (Flash Professional Only) 17

Update the timesheet. 18

Web Service Tutorial: Macromedia Tips (Flash Professional Only)

In this tutorial, you use the Web Services panel to connect to a web service, which you use to return a random tip about Macromedia software. You then use components to set up a simple user interface.

In this tutorial, you will complete the following tasks:

- "Connect to a public web service" on page 5
- "Create a user interface and bind the components with the web service" on page 7

This tutorial uses a public web service and therefore requires that you have an Internet connection.

If you have trouble downloading or decompressing the files, see TechNote 13686 at www.macromedia.com/support/general/ts/documents/downfiles.htm.

Note: The use of a public web service in this tutorial in no way implies that you should use one for real-world applications. In fact, Macromedia does not recommend using public web services directly from within any client-side application. For more information, see "About data connectivity and security in Flash Player" in the "Data Integration" chapter in *Using Flash* (in Flash, select Help > Using Flash). In a production environment, you should use web services that are placed on your own web server.

The finished FLA file for this tutorial installs with Flash. The following list provides typical paths to this directory.

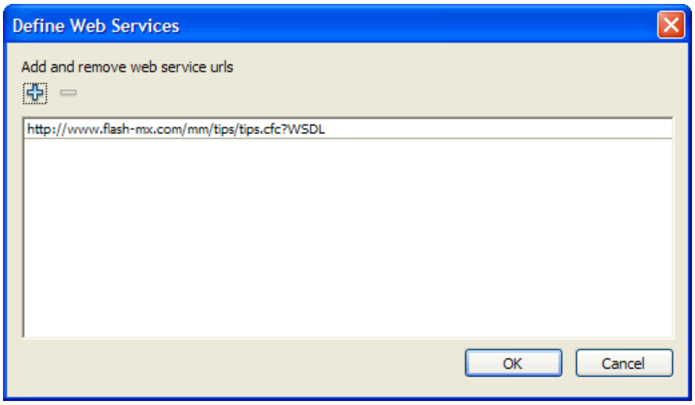
- Windows: \Program Files\Macromedia\Flash MX 2004\Samples\HelpExamples\tips
- Macintosh: HD/Applications/Macromedia Flash MX 2004/Samples/HelpExamples/tips

Connect to a public web service

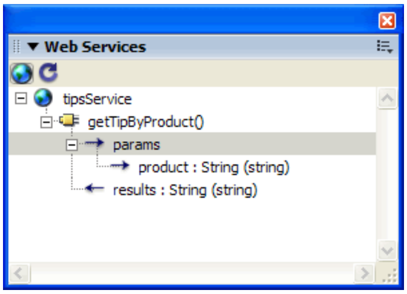
Define a web service in Flash that will connect to a public web service.

1. Create a new Flash document using Flash MX Professional 2004. Make sure your computer is connected to the Internet.
2. Open the Web Services panel (Window > Development Panels > Web Services), and click the Define Web Services button.
3. In the Define Web Services dialog box that appears, click the Add Web Service (+) button, then click the highlighted line to edit it.

4. Enter the URL **http://www.flash-mx.com/mm/tips/tips.cfc?WSDL** and click OK.

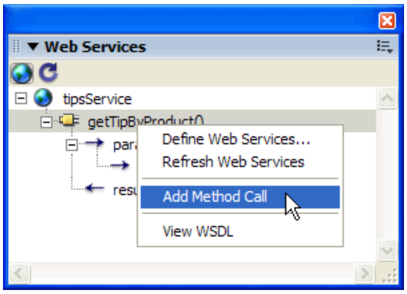

Define Web Services
Add and remove web service urls
http://www.flash-mx.com/mm/tips/tips.cfc?WSDL
OK
Cancel

5. In the Web Services panel, inspect the methods, parameters, and results of the Macromedia Tips web service.



The web service has one method, called `getTipByProduct`. This method accepts a single parameter called `product`. The parameter is a string that tells the web service what Macromedia product you want to see a tip for. In the next step, you bind this parameter with a ComboBox instance in your application.

6. Right-click the `getTipByProduct` method, and select Add Method Call from the context menu.



An instance of the WebServiceConnector component is added to the Stage.

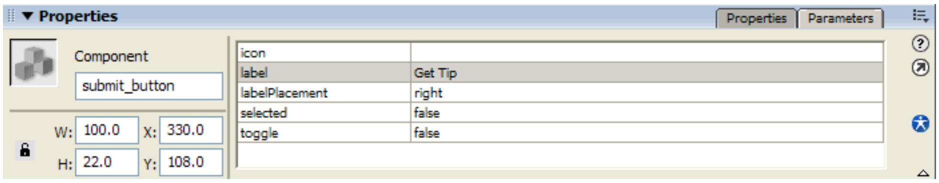
7. In the Property inspector, enter the instance name **tips_wsc**.

   The component is now configured and on the Stage. You can place the component anywhere on or off the Stage—it is invisible when you run the application.

Create a user interface and bind the components with the web service

Next, you use components to create a simple user interface that you can use to select a product, click a button, and see a random tip about the product. You create this application by binding the user interface components on the Stage to the parameter and results in the Macromedia Tips web service.

1. In the Components panel, select UI components > ComboBox. Drag a ComboBox component to the Stage. In the Property inspector, enter the instance name **products_cb**.
2. In the Components panel, select UI components > Button. Drag a Button component to the Stage. In the Property inspector, enter the instance name **submit_button** and for the label property type **Get Tip**, as follows:

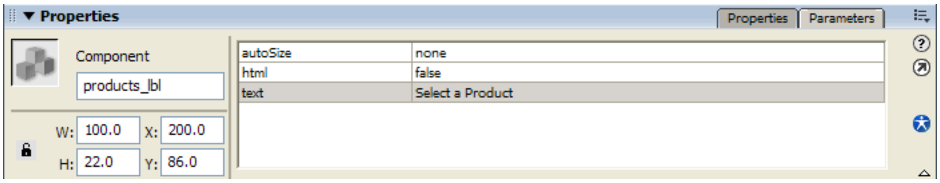


submit_button

W: 100.0 X: 330.0
H: 22.0 Y: 108.0

icon	
label	Get Tip
labelPlacement	right
selected	false
toggle	false

3. In the Components panel, select UI Components > TextArea. Drag the component onto the Stage. In the Property inspector, enter the instance name **tip_ta**.
4. In the Components panel, select UI Components > Label and drag a Label component onto the Stage. Place it above the ComboBox component.
5. In the Property inspector, in the Instance name field type **products_lbl** and for the text property type **Select a Product**, as follows:



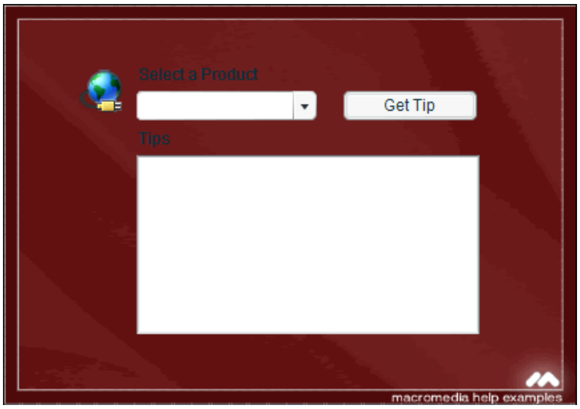
products_lbl

W: 100.0 X: 200.0
H: 22.0 Y: 86.0

autoSize	none
html	false
text	Select a Product

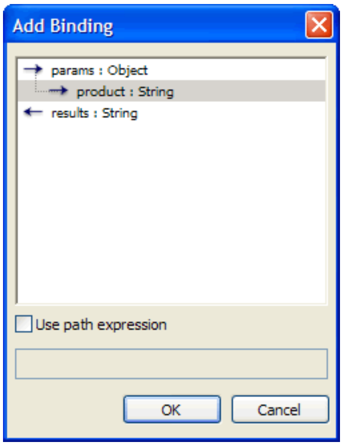
The Property inspector showing the instance name products_lbl and the text Select a Product.

6. Drag another Label component above the `tip_ta` TextArea component. In the Property inspector, give it the Instance name **tip_lbl** and in the text field type **Tips**.

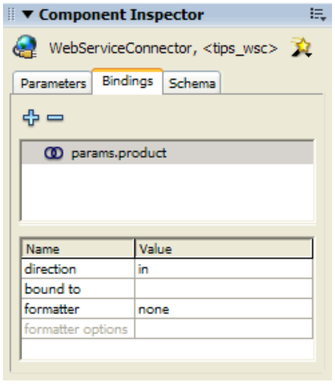


Now add a binding for the WebService connector component from the Macromedia Tip service to ComboBox component that allows the user to choose a product and return a tip about the product.

7. Select the WebServiceConnector component on the Stage. Open the Component inspector, and click the Bindings tab. Click the Add Binding (+) button. In the Add Binding dialog box, select `product:String` (under params:Object), and then click OK.

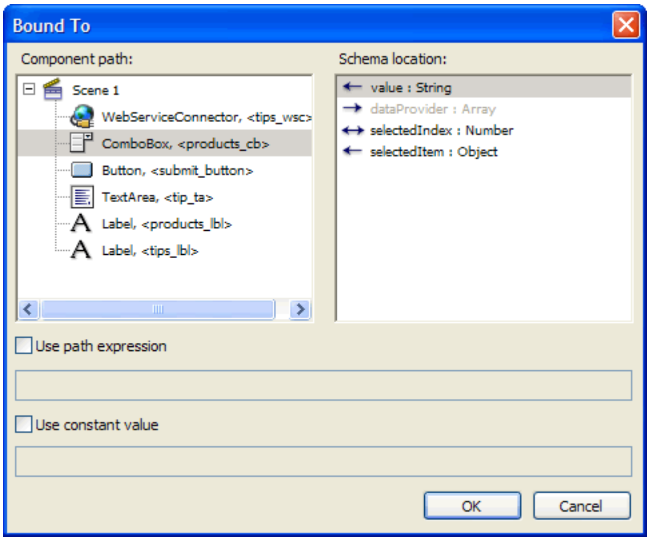


8. In the Component inspector, double-click the empty value in the Bound To field. In the Bound To dialog box, select `ComboBox, <products_cb>` for the component path and `value:String` for the schema location. Then click OK.



Name	Value
direction	in
bound to	
formatter	none
formatter options	

Bound To field in the Component inspector



Selecting the component path and schema location in the Bound To dialog box

Next, you will bind the `results` parameter in the web service connector to the TextArea component on the Stage.

9. In the Component inspector, click the Add Binding (+) button again. In the Add Binding dialog box, select `results:String`, then click OK. In the Component inspector, double-click the empty value in the Bound To field, and in the Bound To dialog box, select `TextArea, <tip_ta>` as the component path and `text:String` as the schema location. Then click OK.

Finally, you will use a Button component and the `trigger()` method to trigger the service. You use the trigger method to attempt to retrieve a tip whenever you click the button.

10. Open the Actions panel and add the following ActionScript on Frame 1 of the Timeline:

```
submit_button.onRelease = function(){
  tips_wsc.trigger();
};
```

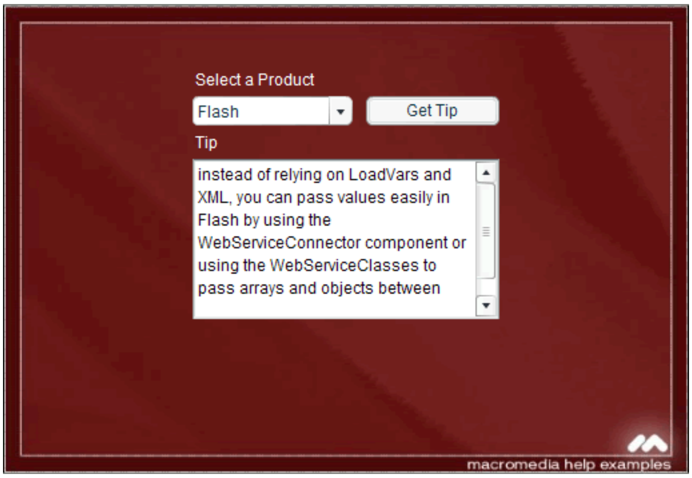
11. Next, add the following ActionScript after the code from step 10. The code uses the `dataProvider` property to set the items in the ComboBox instance to the contents of the array.

```
products_cb.dataProvider = ["Flash", "Dreamweaver"];
```

Note: If necessary, you can use the `setStyle` method to change the color of the Label instance text to white using `products_lbl.setStyle("color", 0xFFFFFF);`

12. Save your file.

13. Test the application (Control > Test Movie). Select **Flash** from the ComboBox instance and click the Get Tip button. The results should look similar to the following graphic:



instead of relying on LoadVars and XML, you can pass values easily in Flash by using the WebServiceConnector component or using the WebServiceClasses to pass arrays and objects between

Select **Dreamweaver** and click the Get Tip button again to view another tip.

XML Tutorial: Timesheet (Flash Professional Only)

In this tutorial, you will create an application for editing timesheet data. The timesheet data is stored as XML within a native XML database. The XUpdateResolver component is the best choice for this type of application, because it generates XUpdate statements that can be sent to the server to update the data.

You will complete the following tasks:

- "Create the user interface" on page 12
- "Edit the data" on page 16

This tutorial uses a public web service and therefore requires that you have an Internet connection. In addition, the tutorial won't work in a browser because of sandbox restrictions, but will work in the Flash authoring environment or Flash Player.

Note: The use of a public web service in this tutorial does not imply that you should use one for real-world applications. In fact, Macromedia does not recommend using public web services directly from within any client-side application. For more information, see "About data connectivity and security in Flash Player" in the "Data Integration" chapter in *Using Flash* (in Flash, select Help > Using Flash).

For this tutorial, you will need to download the Data Tutorials ZIP file from www.macromedia.com/go/flmx2004_data_tutorials. The ZIP file contains the data.xml file you use in the tutorial.

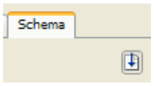
If you have trouble downloading or decompressing the files, see TechNote 13686 at www.macromedia.com/support/general/ts/documents/downfiles.htm.

Note: For demonstration purposes, you will access the XML data from your hard disk and display the DeltaPacket within your screen. In the real world, the XUpdate would be sent to the server for processing.

Create the user interface

You will begin by creating a user interface, which displays the information in the XML file.

1. Create a new Flash document using Flash MX Professional 2004. Make sure your computer is connected to the Internet.
2. From the Components panel, drag an XMLConnector component to the Stage. In the Property inspector, enter the instance name **timeInfo_con**.
3. In the Component inspector or the Property inspector, click the Parameters tab. For the URL parameter, enter **data.xml**, and for the Direction parameter, select Receive from the pop-up menu.
4. From the Components panel, drag a DataSet component to the Stage. In the Property inspector, enter the instance name **timeInfo_ds**.
5. On the Stage, select the XMLConnector component. In the Component inspector, click the Schema tab. Select the `results:XML` property, then click the Import a Schema from a Sample XML File button, at the upper right of the Schema tab.

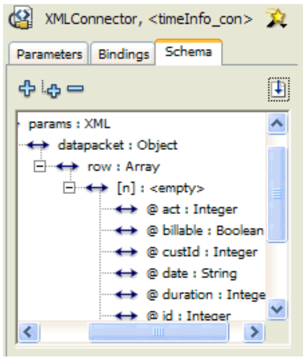
   

   Note: Alternatively, you can select Import XML Schema from the Component inspector title bar menu.

6. Browse to where you saved the data.xml file, and select the file.

   The Schema tab now shows the structure of the data in the file. The `row` node is mapped to an ActionScript array of anonymous objects, because it repeats several times within the XML file. Any subnodes or attributes directly under the row node are considered properties of the anonymous objects contained within the array.

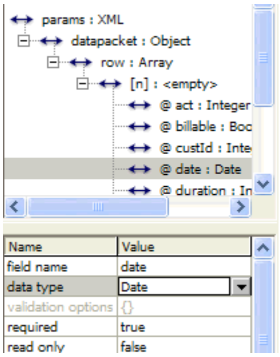
   For more information about how Flash translates XML documents into an internal schema representation, see the "Data Integration" chapter in *Using Flash* (in Flash, select Help > Using Flash).

```
params : XML
  datapacket : Object
    row : Array
      [n] : <empty>
        @ act : Integer
        @ billable : Boolean
        @ custId : Integer
        @ date : String
        @ duration : Intege
        @ id : Integer
```

Note: The XMLConnector component stores information internally as strings. When a request is made for the data through a DataBinding component, you can define how the string data is converted into the correct ActionScript types. This is accomplished by selecting an item within the Schema Tree pane and modifying its settings.

7. Select the Date schema field. Its type is set to String. This is because the Flash authoring tool cannot determine that it is a date type based on its value. You need to give it some additional information to encode this value correctly.
8. Select the Data Type parameter for the Date schema field and change it to Date. This tells the DataBinding component to try to work with this value as a date.

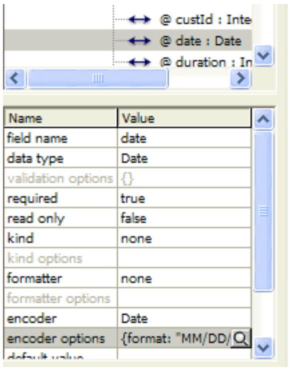


```
params : XML
  datapacket : Object
    row : Array
      [n] : <empty>
        @ act : Integer
        @ billable : Boo
        @ custId : Inte
        @ date : Date
        @ duration : In
```

Name	Value
field name	date
data type	Date
validation options	{}
required	true
read only	false

For more information on data binding and data types, see “Schema data types” in the “Data binding” section of the “Data Integration” chapter in *Using Flash* (in Flash, select Help > Using Flash).

9. Select the encoder parameter for the Date schema field and change it to Date. Select the encoder options parameter and select the value “MM/DD/YYYY”. This tells the DataBinding component how the string value is represented in the XML file. With this information, the DataBinding component can successfully take any string in this format and convert it into an ActionScript date object.



```
@ custId : Inte
@ date : Date
@ duration : In
```

Name	Value
field name	date
data type	Date
validation options	{}
required	true
read only	false
kind	none
kind options	
formatter	none
formatter options	
encoder	Date
encoder options	{format: "MM/DD/

For more information on data binding and encoders, see “Schema encoders” in the “Data binding” section of the “Data Integration” chapter in *Using Flash* (in Flash, select Help > Using Flash).

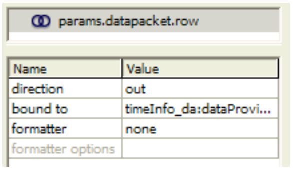
10. Select the `@billable` schema field.

 Notice that the field's data type was automatically set to Boolean by the authoring tool, which looks for certain patterns to guess the type of an XML element. However, you need to modify the Encoder Options for the field. For Boolean data types, the encoder options specify strings that indicate true and false values.

11. With the `@billable` schema field still selected, double-click the Encoder Options field.
12. In the Boolean Encoder dialog that appears, enter **true** in the String That Mean True text box and enter **false** in the Strings That Mean False text box.
13. Select the `@duration` schema field.

 Notice that the field's data type was automatically set to Integer. This is because the sample XML field contained only whole number values for this attribute.

14. Select the Data Type setting for the `@duration` schema field and change it to Number so that it is not limited to integer values.
15. In the Component inspector, click the Bindings tab and create a binding between the `row` array and the DataSet component's `dataProvider:Array` property. Select the Direction property and set it to Out.



params.datapacket.row

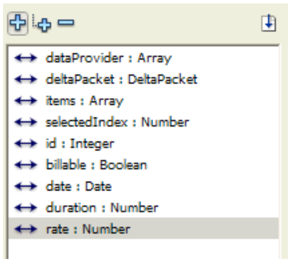
Name	Value
direction	out
bound to	timeInfo_da:dataProvi...
formatter	none
formatter options	

 The DataBinding component copies each object within the `row` array into a new record (transfer object) within the DataSet component. It applies the settings you selected as the data is copied, so that the DataSet component receives ActionScript Date, Boolean, and Number fields for the `@date`, `@billable` and `@duration` attributes.

 Next, you will create fields for the DataSet component that match those in the XMLConnector component.

16. On the Stage, select the DataSet component. In the Component inspector, click the Schema tab.
17. Click the Add a Component Property (+) button and enter **id** for Field Name and **Integer** for Data Type.
18. Using the same method, create the following new fields:
    - Field Name = **billable**, Data Type = **Boolean**
    - Field Name = **date**, Data Type = **Date**
    - Field Name = **duration**, Data Type = **Number**
    - Field Name = **rate**, Data Type = **Number**

Note: The field names must exactly match the names of their corresponding properties within the XMLConnector component (@date = date, @billable = billable, @duration = duration), including capitalization.


dataProvider : Array
deltaPacket : DeltaPacket
items : Array
selectedIndex : Number
id : Integer
billable : Boolean
date : Date
duration : Number
rate : Number

19. Select the Date field that you just created. Select the encoder setting and change it to **DateToNumber**.

    Note: The DataSet component needs to store date values in their numeric equivalents so that they can be sorted correctly. This encoder converts a Date into a Number whenever the value is set. It converts a Number into a Date whenever the value is accessed.

20. With the Date field still selected, double-click the Formatter field in the Component Inspector and choose Date from the pop-up menu.
21. Double-click the Formatter Options field in the Component Inspector.
22. In the Date Formatter Settings dialog that appears, enter **MM-DD-YYYY** in the Format text box.
23. Drag a DataGrid component to the Stage, and in the Property inspector enter the instance name **timeInfo_grd**.
24. In the Component inspector, click the Bindings tab. Create a binding between the DataGrid component's `dataProvider` property and the DataSet component's `dataProvider` property. Set the direction to **In**.
25. Add another binding between the DataGrid component's `selectedIndex` property and the DataSet component's `selectedIndex` property.
26. Drag a Button component to the Stage, and give it the instance name **loadData_btn** in the Property inspector.
27. In the Component inspector, click the Parameters tab. In the Label field, type **Load Data**.
28. With the button still selected on the Stage, open the Behaviors Panel (Window > Development Panels > Behaviors).
29. Click the Add Behavior (+) button, and select Data > Trigger Data Source. In the Trigger Data Source dialog box, select the timeInfo_con component, and click OK.
30. Save the file in the same folder where the data.xml file resides.
31. Run the application, and click the Load Data button.

    The XML data is retrieved, converted, and loaded into the DataSet component. The binding between the DataSet and the DataGrid copies the data into the Grid for display.

Edit the data

Now you will modify the application so that you can edit data through the DataGrid component.

1. On the Stage, select the DataGrid component. Then click the Parameters tab in the Component inspector.
2. Set the `editable` property to `true`.
3. Run the application.

   You can now edit the data within the grid.

XUpdate Tutorial: Update the Timesheet (Flash Professional Only)

Prerequisite: "XML Tutorial: Timesheet (Flash Professional Only)"

This tutorial starts where the XML tutorial: Timesheet left off. Now that the DataSet component is managing the Data, it is tracking changes that are made to the data into the DeltaPacket. A resolver is needed to send the changes back to the server in an optimized way. The XUpdateResolver component is the best choice for updating an XML source, because it generates XUpdate statements that can be sent to the server to update the data.

You will complete the following task:

- "Update the timesheet" on page 18

This tutorial uses a public web service and therefore requires that you have an Internet connection. In addition, the tutorial won't work in a browser because of sandbox restrictions, but will work in the Flash authoring environment or Flash Player.

Note: The use of a public web service in this tutorial in no way implies that you should use one for real-world applications. In fact, Macromedia does not recommend using public web services directly from within any client-side application. For more information, see "About data connectivity and security in Flash Player" in the "Data Integration" chapter in *Using Flash* (in Flash, select Help > Using Flash).

For this tutorial, you will need to download the Data Tutorials ZIP file from www.macromedia.com/go/flmx2004_data_tutorials. The ZIP file contains the data.xml file that you use in the tutorial.

If you have trouble downloading or decompressing the files, see TechNote 13686 at www.macromedia.com/support/general/ts/documents/downfiles.htm.

Note: For demonstration purposes, you will access the XML data from your hard disk and display the DeltaPacket within your screen. In the real world, the XUpdate would be sent to the server for processing.

Update the timesheet

1. Begin with the file you created in the “XML Tutorial: Timesheet (Flash Professional Only)”
2. Drag an XUpdateResolver component to the Stage, and in the Property inspector enter the instance name **timeInfo_rs**.
3. Click the Schema tab in the Component inspector, and select the `deltaPacket` component property within the Schema Tree pane.
4. Change the DeltaPacket component’s encoder setting to DataSetDeltaToXUpdateDelta.

   This encoder converts data within the DeltaPacket into XPath statements that are supplied to the XUpdateResolver component, but it needs additional information from you to do its job.
5. Double-click the `encoder options` property. When prompted for a value for the `rowNodeKey` property, type **datapacket/row**[**@id=**'**?id**'] .

   This property identifies which node within the XML file will be treated as a record within the data set. It also defines which element or attribute combination makes the row node unique, as well as the schema field within the DataSet component that will represent it. See “DataSetDeltaToXUpdateDelta encoder” in the “Data Integration” chapter of *Using Flash* (in Flash, select Help > Using Flash).

   In the sample XML file, the `id` attribute of the datapacket/row node is the unique identifier, and it will be mapped to the DataSet component’s ID schema field. This is defined with the following expression:

   ```
   datapacket/row[@id=’?id’]
   ```
6. In the Component inspector, click the Bindings tab. Add a binding from the XUpdateResolver component’s `deltaPacket` property to the DataSet component’s `deltaPacket` property. This binding will copy the DeltaPacket component to the XUpdateResolver component so that it can be manipulated before it is sent to the server.

   Note: The data is copied after the DataSet component’s `applyUpdates()` method is called.
7. Drag a TextArea component to the Stage, and in the Property inspector enter the instance name **deltaText**. Select the component, and then in the Component inspector, click the Bindings tab. Add a binding from the TextArea component’s `text` property to the XUpdateResolver component’s `xupdatePacket` property. The update packet contains the modified version of the DeltaPacket that will be sent to the server.
8. Drag a Button component onto the Stage, and in the Property inspector enter the instance name **btn_show**. In the Component inspector, click the Parameters tab and change the label to Show Updates.
9. With the button selected, open the Actions panel (F9) and enter the following code:

   ```
   on (click) {
     _parent.timeInfo_ds.applyUpdates();
   }
   ```
10. Test the application (Control > Test Movie). Load the data and make a change to one or more fields in multiple records.

11. Click the Show Updates button. Review the XML packet in the TextArea component.

    Tip: You can copy the XML data into your favorite XML editor to make it easier to read.

12. Try setting the `includeDeltaPacketInfo` parameter of the XUpdateResolver component to `true` using the Component inspector.

    Note: Additional information is added to the update packet. This information can be used by the server to uniquely identify this update operation. With this information, the server can generate a result packet that can be used by the XUpdateResolver component and the DataSet component to update the client data with changes from the server.