Data Tutorials
(Flash Professional Only)

The following tutorials illustrate several ways to use data binding and the data components in Macromedia Flash MX Professional 2004. Many of the tutorials use public web services and therefore require that you have an Internet connection. In addition, the tutorials won't work in a browser because of sandbox restrictions, but they will work in the Flash authoring environment or Flash Player.

- Web service tutorial: Babel Fish
- Web service tutorial: Stock quotes
- Web service tutorial: Update the stock quotes web service
- XUpdate tutorial: Timesheet

Note: To complete the timesheet tutorial, you must download the file data.xml.

These tutorials are working models that illustrate how to use the data components (XMLConnector, WebServices Connector, RDMBSResolver and XUpdateResolver) with Data binding within Flash MX Professional 2004. They are not intended to be production-ready applications.

Note: The use of public web services in these tutorials in no way implies that you should use them 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 "Applications and Web Services" in the "Data Integration" chapter in *Using Flash* (in Flash, select Help > Using Flash).

If you have trouble downloading or decompressing the files, see TechNote 13686.

Web service tutorial: Babel Fish

In this tutorial, you will use the Web Services panel to connect to Alta Vista's public web service, Babel Fish, which is used to translate text from one language to another. You will then use components to set up a simple user interface.

You will complete the following tasks:

- "Connect to a public web service" on page 2
- "Create a user interface and bind the components with the web service" on page 3

Connect to a public web service

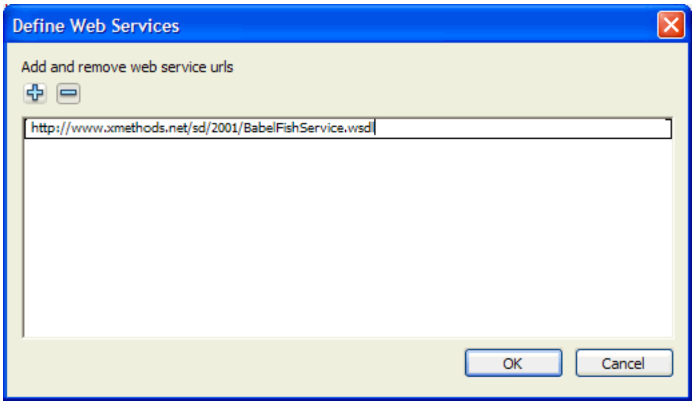
You will begin by defining a web service to connect to a public web service.

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.

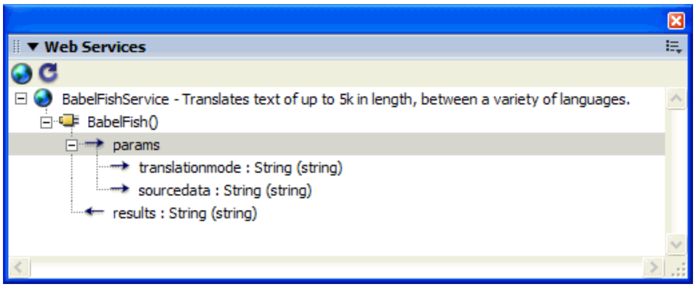
3 In the Define Web Services dialog box that appears, click the Add Web Service (+) button, then click the highlighted line to edit it.

4 Enter the URL **http://www.xmethods.net/sd/2001/BabelFishService.wsdl** and click OK.



http://www.xmethods.net/sd/2001/BabelFishService.wsdl

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

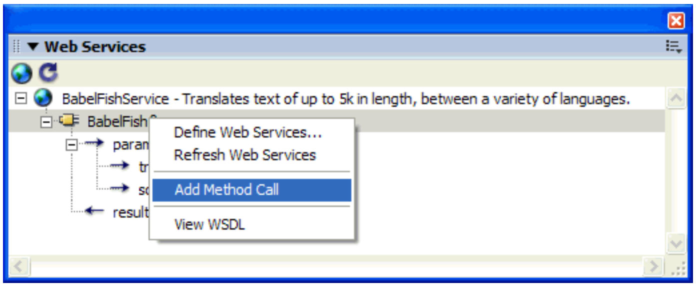


BabelFishService - Translates text of up to 5k in length, between a variety of languages.
BabelFish()
params
translationmode : String (string)
sourcedata : String (string)
results : String (string)

There are two parameters: `translationmode` and `sourcedata`. In the next step, you will connect these parameters with the text input boxes in your application. You will also connect the results with your application.

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

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



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

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 will use components to create a simple user interface in which users can enter a word in one language, click a button, and see the results in another. This is accomplished by binding the user interface components on the Stage to the parameters and results in the Babel Fish web service.

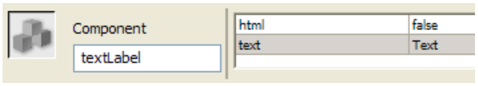
1 In the Components panel, select UI components > TextInput. Drag a TextInput component to the Stage. In the Property inspector, enter the instance name **source**.

2 Drag another TextInput component to the Stage. In the Property inspector, enter the instance name **mode**.

3 In the Components panel, select UI Components > TextArea. Drag the component onto the Stage. In the Property inspector, enter the instance name **result**.

4 In the Components panel, select UI Components > Label and drag a Label component onto the Stage. Place it to the left of the source TextInput component.

5 In the Property inspector, in the Instance name field type **textLabel** and for the text property type **Text**, as follows:

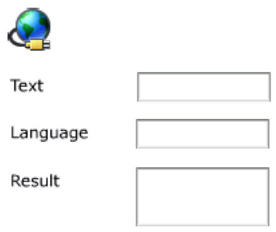

Component
textLabel

html	false
text	Text

The Property inspector showing the instance name textLabel and the text Text.

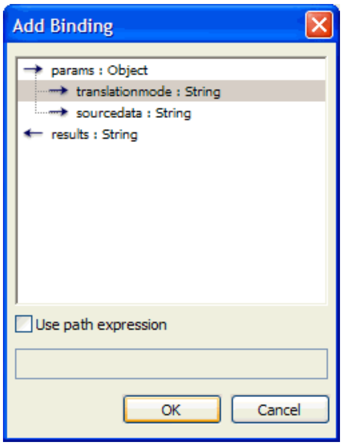
6 In the same way, drag another Label component to the left of the mode TextInput component. In the Property inspector, give it the Instance name **languageLabel** and in the text field type **Language**.

7 Finally, drag another Label component to the left of the result TextArea component. In the Property inspector, give it the Instance name **resultLabel** and in the text field type **Result**.

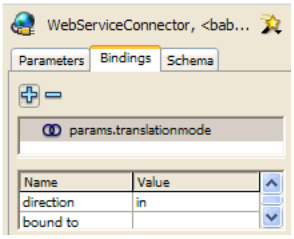


Now add a binding for the WebService connection component from the BabelFish translation mode to the mode TextInput component that will allow the user to enter the translation language.

8 On the Stage, select the WebServiceConnector component. Open the Component Inspector panel, and click the Bindings tab. Click the Add Binding (+) button. In the Add Binding dialog box, select `translationMode:String` (under params:Object), and then click OK.

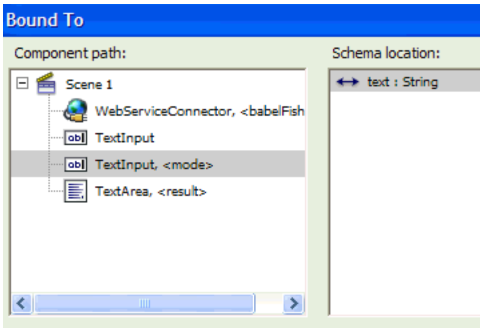


9 In the Component Inspector panel, double-click the empty value in the Bound To field, and in the Bound To dialog box, select `TextInput, <mode>` for the component path and `text:String` for the schema location. Then click OK.



Name	Value
direction	in
bound to	

Bound To field in the Component Inspector panel



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

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

10 In the Component Inspector panel, click the Add Binding (+) button again. In the Add Binding dialog box, select `sourcedata:string`, then click OK. In the Component Inspector panel, double-click the empty value in the Bound To field, and in the Bound To dialog box, select `TextInput, <source>` as the component path and `text:String` as the schema location. Then click OK.

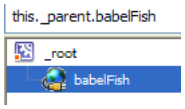
Next, you will bind the `results` in the web service connector to the result component on the Stage.

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

Finally, you will create a Submit button. To do this, you will use a Button component and the trigger data source behavior. The trigger data source behavior allows you to target any Connector component within your application. If this behavior is added to a button, the connector will attempt to retrieve data whenever the button is pressed.

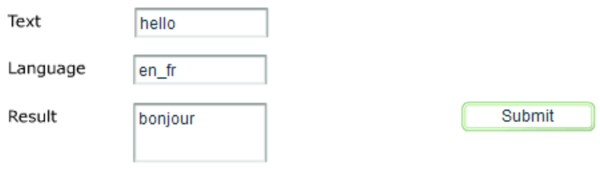
12 Drag a Button component to the Stage, and give it the instance name **submit_btn** in the Property inspector. In the Component Inspector panel, click the Parameters tab. In the Label field, type **Submit**.

13 With the button still selected on the Stage, open the Behaviors Panel (Window > Development Panels > Behaviors). Click the Add Behavior (+) button, and select Data > Trigger Data Source. In the Trigger Data Source dialog box, select the babelFish component. Click OK.



14 Save your file.

15 Test the application (Control > Test Movie). Enter **hello** in the Text (source input) box. Enter **en_fr** in the Langauge (mode input) box. Click the Submit button. The results should look something like the following graphic:



Text	hello
Language	en_fr
Result	bonjour

Try entering other English words or phrases and clicking Submit.

Web service tutorial: Stock quotes

In this tutorial, you will use the Web Services panel to connect to a public stock quotes site. Then you will create and modify a user interface.

You will complete the following tasks:

- "Connect to a public stock quotes site" on page 7
- "Create a user interface that integrates with the stock quotes site" on page 8
- "Change the display of stock information" on page 10
- "Display the stock volume" on page 11
- "Display additional stock information" on page 13
- "Access the data through the DataSet component" on page 15
- "Add some navigation" on page 19
- "Edit the data" on page 20
- "Use a resolver to update the data" on page 20

Connect to a public stock quotes site

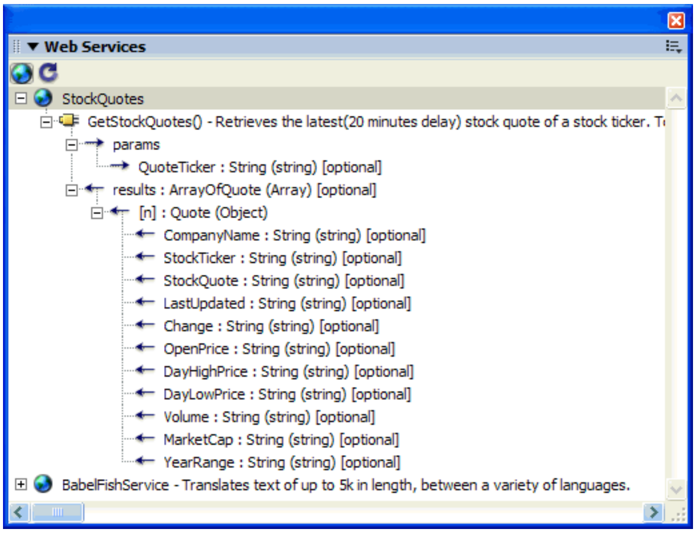
You will begin by building a user interface, then creating a connection to a public stock quotes site.

1 Using Flash MX Professional 2004, create a new Flash document. Make sure your computer is connected to the Internet.

2 Open the Web Services panel (Window > Development Panels > Web Services), and then click the Define Web Services button. The web service you previously set up for Babel Fish is still available.

3 In the Define Web Services dialog box, click the Add Web Service (+) button, and then click on the highlighted line to edit it. Enter the URL **http://www.swanandmokashi.com/HomePage/WebServices/StockQuotes.asmx?WSDL**. When you are finished, click OK.

4 In the Web Services panel, inspect the methods, parameters, and results of the stock quotes web service.



```
StockQuotes
  GetStockQuotes() - Retrieves the latest(20 minutes delay) stock quote of a stock ticker. T
    params
      QuoteTicker : String (string) [optional]
    results : ArrayOfQuote (Array) [optional]
      [n] : Quote (Object)
        CompanyName : String (string) [optional]
        StockTicker : String (string) [optional]
        StockQuote : String (string) [optional]
        LastUpdated : String (string) [optional]
        Change : String (string) [optional]
        OpenPrice : String (string) [optional]
        DayHighPrice : String (string) [optional]
        DayLowPrice : String (string) [optional]
        Volume : String (string) [optional]
        MarketCap : String (string) [optional]
        YearRange : String (string) [optional]
BabelFishService - Translates text of up to 5k in length, between a variety of languages.
```

There is one parameter: QuoteTicker. In addition, there are several results that you can display in the user interface you create.

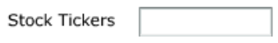
5 In the Components panel, select Data Components > WebServiceConnector. Drag the component to the Stage. In the Property inspector, enter the instance name **getquote**.

Note: In the previous tutorial, you created the component by selecting Add Method Call from the Web Services context menu. However, the panel is the only place you can go to force a web service to refresh. This re-reads the parameters and results. If a change has been made on the server a refresh will get it.

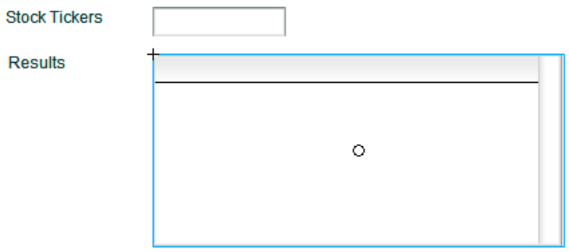
Create a user interface that integrates with the stock quotes site

Next, you will create a user interface in which you can enter stock tickers and get information from the stock quotes web service.

1 In the Components panel, select UI Components > TextInput. Drag the component to the Stage. In the Property inspector, enter the instance name **input**.
2 In the Components panel, select UI Components > Label and drag a Label component onto the Stage. Place it to the left of the source TextInput component.
3 In the Property inspector, in the Instance name field type **stockLabel** and for the text property type **Stock Tickers**, as follows:



4 In the Components panel, select UI Components > DataGrid. Drag the component to the Stage. In the Property inspector, enter the instance name **resultsGrid**. Select the Free Tranform tool from the Tools panel, and drag the data grid on the Stage to make it wide.
5 Drag a Label component to the left of the data grid and in the Property inspector, give it the instance name **resultsLabel** and in the text field type **Results**.

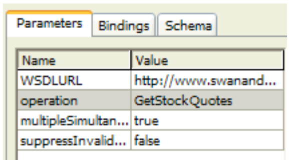


Next, you will configure the getquote WebServiceConnector component.

6 Select the getquote WebServiceConnector component on the Stage. In the Component Inspector panel (Window > Development Panels >Component Inspector), click the Parameters tab.

Note: Alternately, you also enter component parameters using the Parameters tab of the Property inspector.

7 In the Value field for the `WSDLURL` parameter, select http://www.swanandmokashi.com/HomePage/WebServices/StockQuotes.asmx?WSDL from the pop-up menu.
8 In the Value field for the `operation` parameter, select GetStockQuote from the pop-up menu.

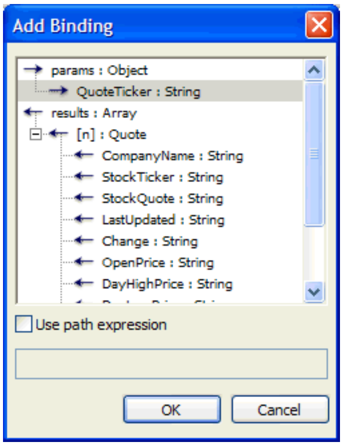


Name	Value
WSDLURL	http://www.swanand...
operation	GetStockQuotes
multipleSimultan...	true
suppressInvalid...	false

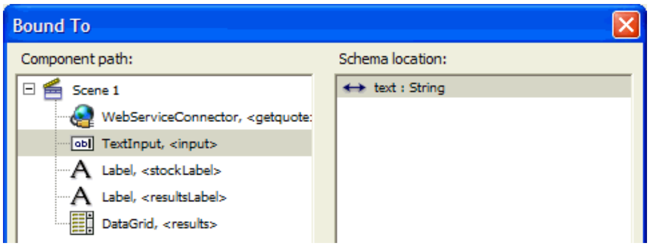
Next, you will create a binding from the web service `QuoteTicker` parameter to the input TextInput component on the Stage.

9 With the getquote WebServiceConnector component still selected on the Stage, in the Component Inspector panel, click the Bindings tab and click the Add Binding (+) button.

10 In the Add Binding dialog box, select `QuoteTicker:String`, and click OK.



11 In the Component Inspector panel, double-click the empty value in the Bound To field. In the Bound To dialog box that appears, select `Text Input, <input>` as the component path and `Text:String` as the schema location. Then click OK.



Now you will bind the `results:Array` parameter to the results DataGrid component on the Stage. This provides a connection between the DataGrid component on the Stage and the results in the stock quotes web service.

12 With the getquote WebServiceConnector component still selected on the Stage, in the Component Inspector panel, click the Add Binding (+) button again. In the Add Binding dialog box, select `results:Array`, then click OK. In the Component Inspector panel, double-click the empty value in the Bound To field. In the Bound To dialog box, select `DataGrid, <resultsGrid>` for Component Path and `dataProvider:Array` for Schema Location, then click OK.

Finally, you will create a button to submit the data entered to the stock quotes service and return the results.

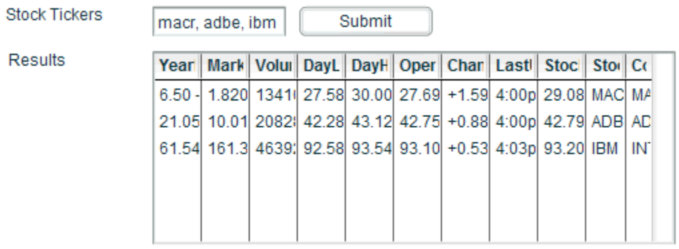
13 Drag a Button component to the Stage, and give it the instance name **submit_btn** in the Property inspector. In the Component Inspector panel, click the Parameters tab. In the Label field, type **Submit**.

14 With the button still selected on the Stage, open the Behaviors Panel (Window > Development Panels > Behaviors). Click the Add Behavior (+) button, and select Data > Trigger Data Source. In the Trigger Data Source dialog box, select the getquote component, and click OK.

15 Save your file.

16 Test your application (Control > Test Movie). Type the following stock symbols, separated by commas, in the Stock Tickers text box: **macr,adbe,ibm**. Then click the Submit button.

The Results data grid shows the raw array of data that arrived back from the server, which in this case is stock quotes for the stock symbols you typed in the text box, as shown in the following graphic:



Year	Mark	Volu	DayL	DayH	Oper	Char	Last	Stoc	Sto	C
6.50 -	1.820	1341	27.58	30.00	27.69	+1.59	4:00p	29.08	MAC	MA
21.05	10.01	2082	42.28	43.12	42.75	+0.88	4:00p	42.79	ADB	AD
61.54	161.3	4639	92.58	93.54	93.10	+0.53	4:03p	93.20	IBM	IN

You see that each piece of information about each stock (company name, price, volume, and so forth) is displayed in its own column in the grid. Because there are so many columns, this presentation makes it difficult to read the information contained by the grid. Also, the column names in the grid assume the original name of each field returned by the web service (for example, "CompanyName" and "StockQuote"). In the next section, you'll apply a formatter to just display certain pieces of information returned by the web service, using column names that you specify.

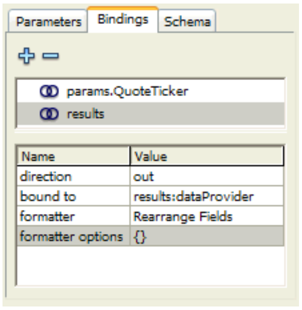
Change the display of stock information

In this section, you will modify the user interface to display just two columns in the grid: the company name and its current stock price.

1 On the Stage, select the getquote component. In the Component Inspector panel, click the Bindings tab. Select the results binding.

2 In the Formatter field, double-click None. In the pop-up menu that appears, select Rearrange Fields.

The Name field `formatter options` is added to the bottom of the list of component parameters, as shown in the following:



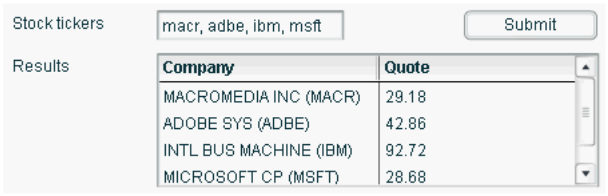
Name	Value
direction	out
bound to	results:dataProvider
formatter	Rearrange Fields
formatter options	{}

3 Double-click the {} in the `formatter options` field. In the Rearrange Fields dialog box that appears, type **Company='<CompanyName> (<StockTicker>)';Quote=StockQuote**, then click OK.

This formatter will result in just two columns being displayed in the grid, Company and Quote. Each row in the grid will be formatted as illustrated below:

Company	Quote
Company Name (*TickerSymbol*)	*Stock quote price*

4 Save your file, then test the application again. Type several stock symbols separated by commas, then click the Submit button. The grid now displays two columns in the predefined format, as shown below:



Company	Quote
MACROMEDIA INC (MACR)	29.18
ADOBE SYS (ADBE)	42.86
INTL BUS MACHINE (IBM)	92.72
MICROSOFT CP (MSFT)	28.68

Note: For more information on formatters that are included with Flash MX Professional 2004, see the "Schema formatters" section in the "Data Integration" chapter in *Using Flash* (in Flash, select Help › Using Flash).

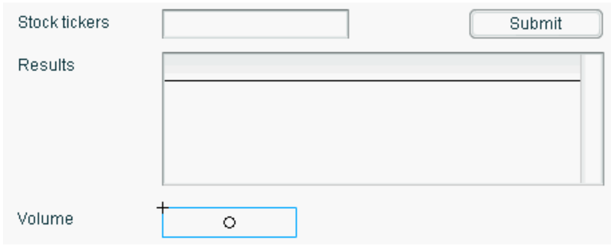
Display the stock volume

Now you will add a feature so that you can select a stock in the Results list and show the volume for that stock in a separate text field.

1 In the Components panel, select UI Components > Label. Drag the component to the Stage. In the Property inspector, type the instance name **volumeValue** and delete the current value ("Label") assigned to the text property.

2 Drag another Label component onto the Stage. Place it to the left of the volumeValue Label component.

3 In the Property inspector, in the Instance name field type **volumeLabel** and for the text property type **Volume**.



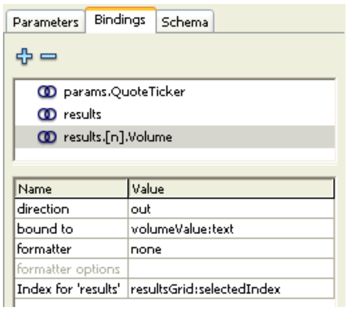
4 On the Stage, select the getquote component. In the Component Inspector panel Bindings tab, click the Add Binding (+) button.

5 In the Add Binding dialog box, select Volume:String, then click OK. In the Component Inspector panel, double-click the empty value in the Bound To field. In the Bound To dialog box that appears, select `Label, <volumeValue>` as the component path and text:String as the schema location, then click OK.

In the Component Inspector panel, a new field is added, `Index for 'results.'` In this field you can select which item in the results array to display in volumeValue. Instead of displaying a constant value, the default, you will select an index.

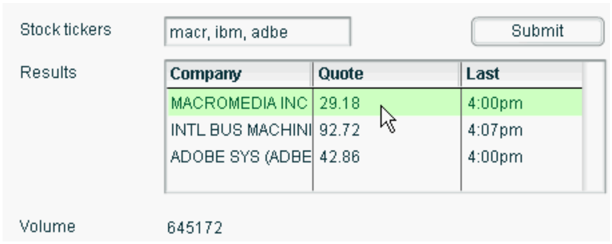
6 Double-click on the `Index for 'results'` field. In the Bound Index dialog box that is displayed, deselect Use Constant Value.

7 In the Component Path field, select `DataGrid, <resultsGrid>`, and in the Schema Location field, select `selectedIndex:Number`. Then click OK.



Name	Value
direction	out
bound to	volumeValue:text
formatter	none
formatter options	
Index for 'results'	resultsGrid:selectedIndex

8 Save your file, then test the application (Control > Test Movie). Type several stock symbols separated by commas, then click the Submit button. Select a line in the Results list. The text of `volumeValue` is the `Volume` field of whatever record you selected in the Results list.



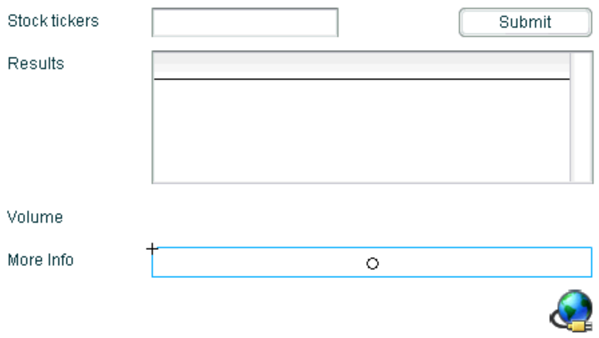
Company	Quote	Last
MACROMEDIA INC	29.18	4:00pm
INTL BUS MACHINI	92.72	4:07pm
ADOBE SYS (ADBE	42.86	4:00pm

Display additional stock information

In the same way, you will add another item to display even more details about the selected stock.

1 In the Components panel, select UI Components > Label. Drag the component to the Stage beneath the `volumeValue` component. In the Property inspector, type the instance name **rangeValue**. Use the Free Transform tool to make it at least twice as wide.

2 Drag another Label component onto the Stage. Place it to the left of the volumeValue Label component. In the Property inspector, in the Instance name field type **rangeLabel** and for the text property type **More Info**.



Next, you will add a binding. In this case you will bind the `[n]quote` parameter from the stock web service to the `rangeValue` component.

3 On the Stage, select the getquote component. Then in the Component Inspector panel Bindings tab, click the Add Binding (+) button.

4 In the Add Binding dialog box, select `[n]quote`, then click OK. In the Component Inspector panel, double-click the empty value in the Bound To field. In the Bound To dialog box that appears, select `Label, <rangeValue>` as the component path and `text:String` as the schema location, then click OK.

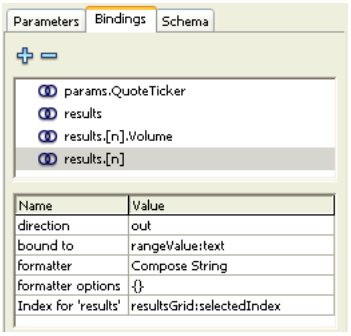
Now you will configure what to display.

5 Double-click the value in the `Index for 'results'` field. In the Bound Index dialog box that appears, deselect Use Constant Value.

6 In the Component Path field, select `DataGrid, <resultsGrid>`, and in the Schema Location field, select `selectedIndex:Number`. Then click OK.

7 Double-click the `formatter` field, and from the pop-up menu, select Compose String.

In the Component Inspector panel a new field is added, `formatter options`.



params.QuoteTicker
results
results.[n].Volume
results.[n]

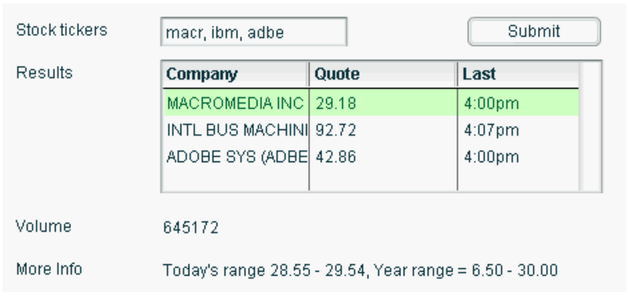
Name	Value
direction	out
bound to	rangeValue:text
formatter	Compose String
formatter options	{}
Index for 'results'	resultsGrid:selectedIndex

8 In the `formatter options` field, click on the magnifying glass. In the Compose String dialog box, type **Today's range <DayLowPrice> - <DayHighPrice>, Year range = <YearRange>**, then click OK.

This string displays the text Today's range, then gets the value from the web service for DayLowPrice, displays -, gets the value from the web service for DayHighPrice, displays Year range =, and gets the value for the YearRange from the web service.

9 Save the file, then test the application (Control > Test Movie). Type several stock symbols separated by commas, then click the Submit button. Select a line in the Results list.

The text in the rangeValue field has a formatted string containing several items from the currently selected data record.



Stock tickers: macr, ibm, adbe

Company	Quote	Last
MACROMEDIA INC	29.18	4:00pm
INTL BUS MACHINI	92.72	4:07pm
ADOBE SYS (ADBE	42.86	4:00pm

Volume 645172

More Info Today's range 28.55 - 29.54, Year range = 6.50 - 30.00

Web service tutorial: Update the stock quotes web service

In this tutorial, you will use a DataSet component and an RDBMSResolver component to generate an update packet to be sent back to the server for updating the data that was retrieved from the stock quotes web service.

You will complete the following tasks:

- "Access the data through the DataSet component" on page 15
- "Add some navigation" on page 19
- "Edit the data" on page 20
- "Use a resolver to update the data" on page 20

Note: In a real-world example, you would add an additional WebServiceConnector component on the Stage to call a method on the server that accepts the update packet, parses the instructions, and updates the data appropriately.

Access the data through the DataSet component

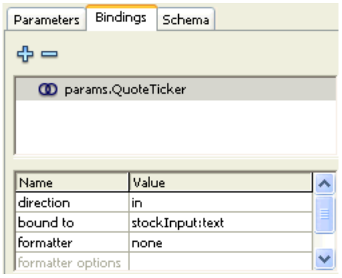
In this section you'll introduce the DataSet component into the stock quote application. First, we bind the data from the WebServiceConnector component to the DataSet. The data in the DataSet is then bound to the DataGrid and other user interface components. The DataSet component is used to manage and track changes made to data.

1 Using the previous application, delete the **companyLabel**, **companyValue**, **stockLabel**, and **stockValue** components.

2 On the Stage, select the WebServiceConnector (`getquote`) component.

3 In the Component Inspector panel's Bindings tab, delete the `results` binding by selecting it and the clicking the Remove Binding (-) button. Repeat this step for the `results.[n].Volume` and `results.[n]` bindings.

The only binding that should remain is the `params.QuoteTicker` binding, as shown below.



Name	Value
direction	in
bound to	stockInput:text
formatter	none
formatter options	

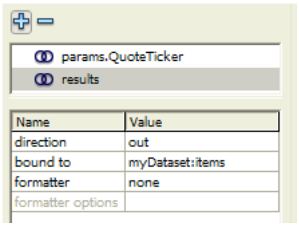
4 Drag a DataSet component from the Components panel to the Stage and name it **myDataSet**.

Now you will create a new binding from the results in the web service to the DataSet component's `items` array.

5 Select the `getquote` component, and click the Add Binding button on the Component Inspector panel's Binding tab.

6 In the Add Binding dialog box, select `results: Array`, then click OK.

7 In the Component Inspector panel, double-click the empty value in the Bound To field. In the Bound To dialog box that appears, select `DataSet, <myDataSet>` for Component Path and `items:Array` for Schema Location, then click OK.



params.QuoteTicker
results

Name	Value
direction	out
bound to	myDataset:items
formatter	none
formatter options	

Next you will create field properties for the DataSet component to define which fields to bind to the UI controls. Any updates made to these fields are tracked by the DataSet component and stored in the DeltaPacket component.

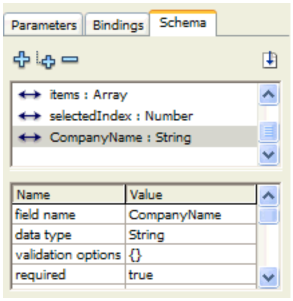
Note: The field names you create must match the schema names (parameters) defined within the WebServiceConnector component. The names are case-sensitive.

8 On the Stage, select the DataSet component.

9 In the Component Inspector panel, click the Schema tab. Click the Add a Component Property (+) button at the upper left of the tab.

A new component property is added to the list of properties.

10 In Component Inspector panel, type **CompanyName** for the Field Name value, then make sure that String is selected for the Data Type value.



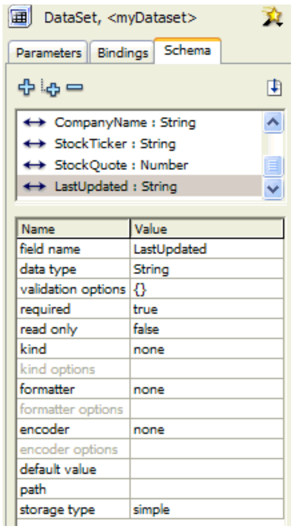
items : Array
selectedIndex : Number
CompanyName : String

Name	Value
field name	CompanyName
data type	String
validation options	{}
required	true

11 Click the Add a Component Property (+) button again. In Component Inspector panel, type **StockTicker** for the Field Name value, and make sure that String is selected for the Data Type value.

12 Click the Add a Component Property (+) button again. In Component Inspector panel, type **StockQuote** for the Field Name value, then select Number for the Data Type value.

13 Click the Add a Component Property (+) button again. In Component Inspector panel, type **LastUpdated** for the Field Name value, then select String for the Data Type value.



CompanyName : String
StockTicker : String
StockQuote : Number
LastUpdated : String

Name	Value
field name	LastUpdated
data type	String
validation options	{}
required	true
read only	false
kind	none
kind options	
formatter	none
formatter options	
encoder	none
encoder options	
default value	
path	
storage type	simple

Next, you will add a binding from the DataSet component's `dataProvider` property to the DataGrid component's `dataProvider` property.

14 On the Stage, select the DataSet component. In the Component Inspector panel, click the Bindings tab, then click the Add Binding (+) button.

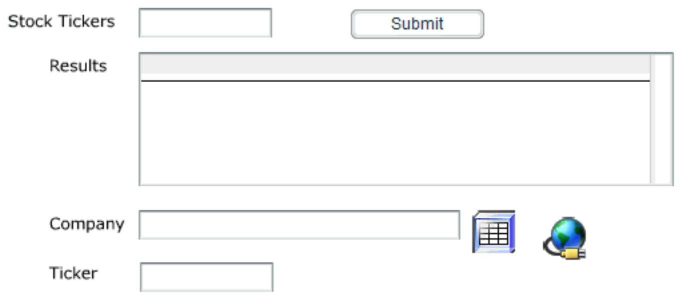
15 In the Add Binding dialog box, select **dataProvider:Array**, then click OK.

16 In the Component Inspector panel, select **Out** for the direction, then Double-click the value in the Bound To field and select **DataGrid, <resultsGrid>** for Component Path and **dataProvider:Array** for Schema Location.

The DataGrid component uses the DataSet component as a DataProvider object for reading and writing data. Therefore, the binding is not bidirectional.

Next, you will add two TextInput components to the Stage. You will use those to display the company name and stock ticker of the selected stock after the results are displayed.

17 Drag two TextInput components to the Stage, and in the Property inspector, type the instance name **company_txt** for one and **ticker_txt** for the other. Use the Text tool to label them **Company** and **Ticker**, respectively. Use the Free Transform tool to widen the `company_txt` component so it is wide enough to display a company name.



Next, you will add a binding from the `company_txt` TextInput component's `text` property to the DataSet component's `CompanyName` property.

18 On the Stage, select the company_txt TextInput component. Then click the Bindings tab in the Component Inspector panel, and click the Add Binding (+) button.

19 In the Add Binding dialog box, select **Text:string**, then click OK.

20 In the Component Inspector panel, double-click the value in the Bound To field and select **DataSet, <myDataset>** for Component Path and **CompanyName:String** for Schema Location.

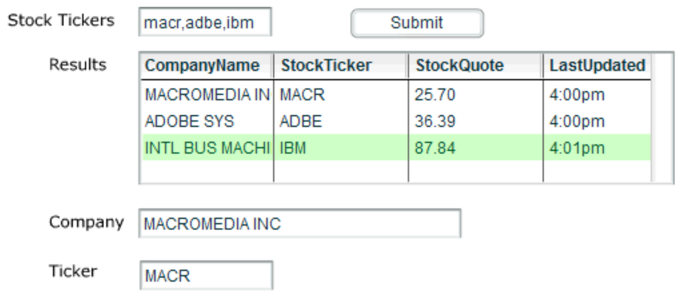
Finally, you will add a binding from the `ticker_txt` TextInput component's `text` property to the DataSet component's `StockTicker` property.

21 On the Stage, select the `ticker_txt` TextInput component. Then click the Bindings tab in the Component Inspector panel and click the Add Binding (+) button.

22 In the Add Binding dialog box, select **Text:string**, then click OK.

23 In the Component Inspector panel, double-click the value in the Bound To field and select **DataSet, <myDataSet>** for Component Path and **StockTicker:String** for Schema Location.

24 Save the file, then test the application. Type several stock symbols separated by commas, then click the Submit button.



CompanyName	StockTicker	StockQuote	LastUpdated
MACROMEDIA IN	MACR	25.70	4:00pm
ADOBE SYS	ADBE	36.39	4:00pm
INTL BUS MACHI	IBM	87.84	4:01pm

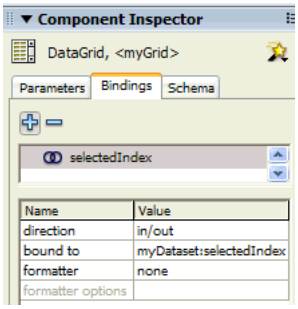
Data is now being accessed through the DataSet component. However, if you select a new record in the DataGrid component, the company name and ticker in the Label components are not updated. You will remedy that in the next section.

Note: You The names of the schema properties you added to the DataSet appear as the column headers in the grid. For information on specifying custom column headers, see "Modify the column headers" on page 21.

Add some navigation

Next, you will add some navigation so you can select a stock in the results data grid and display its company name and stock ticker.

1 Select the DataGrid component on the Stage, then click the Bindings tab in the Component Inspector panel. Add a new binding from the DataGrid component's `selectedIndex` property to the DataSet component's `selectedIndex` property.



Name	Value
direction	in/out
bound to	myDataset:selectedIndex
formatter	none
formatter options	

This step makes sure the Label components display the company name and stock quote for the selected item in the DataGrid.

Next you'll add two button components to the Stage and write ActionScript to make them move to the previous and next records within the DataSet. Because the selected item in the DataSet is bound to the corresponding property in the DataGrid, navigating through the DataSet records will, in turn, navigate through the items in the DataGrid.

2 Drag two Button components onto the lower portion of the Stage and place them next to each other. Select the button on the right. In the Property inspector, type **btn_next** in the Instance Name text box. In the Component Inspector panel, click the Parameters tab. Type **Next** in the Label field. With the button selected on the Stage, open the Actions panel (F9) and enter the following code:

```
on (click) {
  _parent.myDataSet.next();
}
```

3 Select the second button. In the Property inspector, type **btn_previous** in the Instance Name text box. In the Component Inspector, click the Parameters tab. Type **Previous** in the Label field. With the button selected on the Stage, open the Actions panel (F9) and type the following code:

```
on (click) {
  _parent.myDataSet.previous();
}
```

4 Save your file. Test the application (Control > Test Movie). Type several stock symbols separated by commas, then click the Submit button.

You can now navigate through the data and display the company name and stock ticker by using the Next and Previous buttons or by selecting a new record in the DataGrid component. Verify that the text input controls are displaying the correct data.

Edit the data

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

1 Select the DataGrid component. In the Component Inspector panel, click the Parameters tab, then set the `editable` property to `true`.

2 Save your file. Test the application (Control > Test Movie). Type several stock symbols separated by commas, then click the Submit button.

  You can now edit the data within the DataGrid or the TextInput controls. If you modify the data in the TextInput component, the DataGrid component is automatically updated, and vice versa.

Use a resolver to update the data

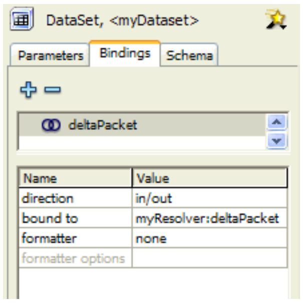
Now that the DataSet component is managing the data, it is tracking changes that are made to the data into the DeltaPacket. A resolver is needed to send the changes back to the server in an optimized way. The RDBMSResolver component is used to update a generic database. In this example, assume that the web service is pulling its data from a table called stock_info.

1 Using the previous application, drag an RDBMSResolver component onto the Stage. In the Property inspector, give it the instance name **myResolver**.

2 In the Component Inspector panel, click the Parameters tab and change the `tableName` property to **stock_info**.

  Next you'll add a binding between the DeltaPacket properties in the DataSet component and the RDBMSResolver component. This binding copies the DeltaPacket to the resolver so that it can be manipulated before it is sent to the server.

3 On the Stage, select the DataSet component. In the Component Inspector panel, click the Bindings tab, and add a binding from the DataSet component's `DeltaPacket` property to the RDBMSResolver component's `DeltaPacket` property.



Name	Value
direction	in/out
bound to	myResolver:deltaPacket
formatter	none
formatter options	

Note: the data is copied after the DataSet component's `applyUpdates()` method is called.

4 Drag a TextArea component to the Stage, and in the Property Inspector give it the instance name **deltaText**.

5 With the component selected, click the Bindings tab in the Component Inspector panel. Add a binding from the TextArea component's `Text` property to the RDBMS Resolver component's `UpdatePacket:XML` property. The update packet contains the modified version of the DeltaPacket that will be sent to the server.

6 Drag a Button component to the Stage, and in the Property Inspector give it the name **btn_show**. In the Component Inspector panel, click the Parameters tab, then change the label to **Show Updates**.

7 With the button selected on the Stage, display the Actions panel (F9) and type the following code:

```
on (click) {
  _parent.myDataSet.applyUpdates();
}
```

8 Test the application (Control > Test Movie). Type several stock symbols separated by commas, then click the Submit button. Load the data and make a change to one or more fields in multiple records.

9 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.

Try changing the different parameters of the RDBMSResolver component using the Component Inspector panel. Each change affects the formatting of the update packet that is displayed in the TextArea component. Using these settings, you can include just the information that you need to send to the server. For more information about RDBMSResolver component parameters, see the "Data Integration" chapter in *Using Flash* (in Flash, select Help > Using Flash).

Note: In addition to modifying the component parameters, you can also capture the `beforeApplyUpdates` event of the RDBMSResolver component to modify the update packet using ActionScript.

Modify the column headers

You probably noticed that the names of the column headers in the DataGrid are the same as the schema items you added to the DataSet (CompanyName, StockTicker, and so forth). In the previous example where you weren't using the DataSet component you used the Rearrange Fields formatter to specify the column names (see "Change the display of stock information" on page 10). This same technique isn't available when you're using the DataSet component to populate the DataGrid with data. However, you can write custom ActionScript to specify the text for each column header.

To specify custom column headers for the DataGrid:

1 Create a new ActionScript file by choosing File > New, select ActionScript File as the type, and click OK.

2 Add the following code to the file:

```
import mx.controls.gridclasses.DataGridColumn;
resultsGrid.columnNames = ["CompanyName", "StockQuote", "StockTicker",
  "LastUpdated"];
//customize column headers
var companyCol:DataGridColumn = resultsGrid.getColumnAt(0);
companyCol.headerText = "Company";
//
var quoteCol:DataGridColumn = resultsGrid.getColumnAt(1);
quoteCol.headerText = "Quote";
//
var tickerCol:DataGridColumn = resultsGrid.getColumnAt(2);
tickerCol.headerText = "Ticker";
//
var updatedCol:DataGridColumn = resultsGrid.getColumnAt(3);
updatedCol.headerText = "Last updated";
```

3 Save the file as setUpHeaders.as to the same folder that contains the example FLA file you're working on.

4 In the FLA file, add the following code to frame 1 of the Timeline:

```
#include "setUpHeaders.as"
```

5 Test your example file (Control > Test Movie). Notice that the column headers contain the text you specified in the ActionScript file.

XUpdate tutorial: Timesheet

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 22
- "Edit the data" on page 26
- "Update the data" on page 26

For this tutorial, you will need the data.xml file you previously downloaded.

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.

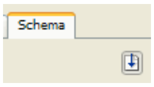
1 Create a new Flash document using Flash MX Professional 2004. Make sure your computer is connected to the Internet.

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 panel 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 panel, 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.

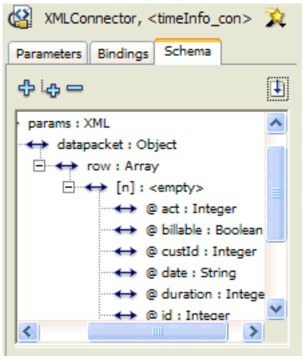


Note: Alternatively, you can select Import XML Schema from the Component Inspector panel title bar menu.

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).

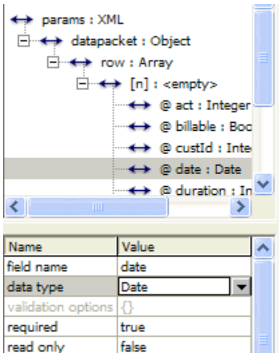


```
params : XML
  datapacket : Object
    row : Array
      [n] : <empty>
        @ act : Integer
        @ billable : Boolean
        @ custId : Integer
        @ date : String
        @ duration : Intege
        @ id : Integer
```

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.

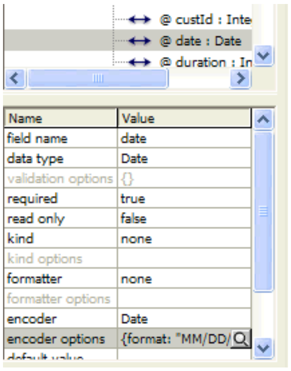


```
params : XML
  datapacket : Object
    row : Array
      [n] : <empty>
        @ act : Integer
        @ billable : Boo
        @ custId : Inte
        @ date : Date
        @ duration : In
```

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.

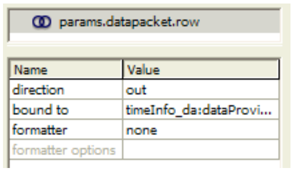
12 In the Boolean Encoder dialog that appears, enter **true** in the String That Mean True text box and enter **false** in the Strings That Mean False text box.

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 panel, 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.

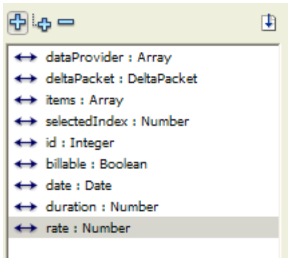
16 On the Stage, select the DataSet component. In the Component Inspector panel, click the Schema tab.

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).



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 panel, 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 panel, 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 panel.

2 Set the `editable` property to `true`.

3 Run the application.

You can now edit the data within the grid.

Update the data

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.

1 Drag an XUpdateResolver component to the Stage, and in the Property inspector enter the instance name **timeInfo_rs**.

2 Click the Schema tab in the Component Inspector panel, and select the `deltaPacket` component property within the Schema Tree pane.

3 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.

4 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']
```

5 In the Component Inspector panel, 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.

6 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 panel, 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.

7 Drag a Button component onto the Stage, and in the Property inspector enter the instance name **btn_show**. In the Component Inspector panel, click the Parameters tab and change the label to Show Updates.

8 With the button selected, open the Actions panel (F9) and enter the following code:

```
on (click) {
   _parent.timeInfo_ds.applyUpdates();
}
```

9 Test the application (Control > Test Movie). Load the data and make a change to one or more fields in multiple records.

10 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.

11 Try setting the `includeDeltaPacketInfo` parameter of the XUpdateResolver component to `true` using the Component Inspector panel.

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.