I'd like to thank the following people for their help with software that I've used in this tutorial, and for helping me with some questions I had getting started. If you're interested in using PHP and Flex or Flash, then you should check out their blogs.
Patrick Mineault - http://www.5etdemi.com/blog/
Jesse Warden – http://www.jessewarden.com/
http://www.tweenpix.net/blog - A blog in French where I got the ActionScript code for RemotingConnection from (http://www.tweenpix.net/blog/index.php?2006/01/03/543-hello-world-en-amfphp-avec-flex20)

In the last article, I showed you how to develop a simple Flex application that connected to a PHP backend. As you were going through it, you were probably thinking "Is there anyway to pass variables directly from PHP, rather than having to encode them into XML?" The answer is yes, and today I'm going to show you how to do that.

If you're looking to build a medium to large scale enterprise application, you should check out Adobe Flex Enterprise Services at http://labs.macromedia.com/technologies/flexenterprise_services2/. That will simplify the data interaction by adding high performance data transfer, message based publish and subscribe and more. AMFPHP only provides a small subset of what is possible with Flex Enterprise Services. If you're a large company, then you'll want to look at FES for your data interaction layer.

The key to doing this is a small project called AMFPHP. As stated on their webpage, the project was originally started by Wolfgang Hamann. The team has grown lately, and now has about 5 or 6 developers. Thanks to their hard work, we can now have a Flex based front end to our PHP backend applications.

If you haven't done so already, download the Flex Beta from the Adobe Labs website at http://labs.adobe.com.
Then, download AMFPHP v. 1.1 from http://sourceforge.net/project/showfiles.php?group_id=72483 Follow the instructions to deploy it.

Now would probably be a good time to read over the first article that I wrote, if you haven't done so already. Check it out at my blog at http://blogs.adobe.com/mikepotter/2006/02/flex_and_php_a_1.html

The sample that we're creating today will show us how to use Flex to display records from a database. We will not be inserting or updating any records, only showing them to the user. We'll use a similar database to the first example, so if you've got a database and table created from that example, then use that. Otherwise, create the following MySQL table:

```
CREATE TABLE `users` (
  `userid` int(10) unsigned NOT NULL auto_increment,
  `username` varchar(255) collate latin1_general_ci NOT NULL,
  `emailaddress` varchar(255) collate latin1_general_ci NOT NULL,
  PRIMARY KEY  (`userid`)
```

```
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci
AUTO_INCREMENT=3 ;
```

You should populate that database with some rows of data. The field names should be self explanatory. Remember, we're only showing the data from the database, so if you don't populate it, then you'll have nothing to show. Use PHPMyAdmin if you need something to populate the table.

Here is the new PHP code. Place this in a file called sample.php in the "services" folder of your AMFPHP installation.

```
<?php
// Create new service for PHP Remoting as Class
class sample
{
   function sample ()
         {
      // Define the methodTable for this class in the constructor
      $this->methodTable = array(
         "getUsers" => array(
            "description" => "Return a list of users",
            "access" => "remote"
         )
      );
   }

   function getUsers () {
         $mysql = mysql_connect(localhost, "username", "password");

         mysql_select_db( "sample" );

         //return a list of all the users
                  $Query = "SELECT * from users";
                  $Result = mysql_query( $Query );
                  while ($row = mysql_fetch_object($Result)) {
                           $ArrayOfUsers[] = $row;
                  }
                  return( $ArrayOfUsers );
         }
}
?>
```

If you're familiar with AMFPHP, then this is likely familiar to you.

The class name needs to match the filename. In our case, the filename is sample.php, so the class is called sample. When the class is loaded, then it runs the function sample() (classes always initialize themselves by calling functions which match the class name), which defines the methods available to AMFPHP. In our case, there's only one:

getUsers, which as the description says, returns a list of users. In the function itself, we can see that its relatively simple: connect to a MySQL database, get all the users, and return them in an array of objects.

Now, let's check out the front end of our application. Amazingly, its only about 50 lines of code in total!
Sample.mxml:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.macromedia.com/2005/mxml" xmlns="*"
creationComplete="initApplication()">
	<mx:DataGrid dataProvider="{dataProvider}">
		<mx:columns>
			<mx:DataGridColumn headerText="Userid"
columnName="userid"/>
			<mx:DataGridColumn headerText="User Name"
columnName="username"/>
			<mx:DataGridColumn headerText="User Name"
columnName="emailaddress"/>
		</mx:columns>
	</mx:DataGrid>
	<mx:Script>
		<![CDATA[
			[Bindable]
			public var dataProvider:Array;

			import flash.net.Responder;

			public var gateway : RemotingConnection;

			public function initApplication()
			{
				gateway = new RemotingConnection(
"http://localhost/flex/php/gateway.php" );
				gateway.call( "sample.getUsers", new Responder(onResult,
onFault));
			}

			public function onResult( result : Array ) : void
			{
				dataProvider = result;
			}


			public function onFault( fault : String ) : void
			{
				trace( fault );
```

```
                }
            ]]>
        </mx:Script>
</mx:Application>
```

The first line, <?xml version="1.0" encoding="utf-8"?>, says that this is an XML document. That goes at the top of all Flex applications.

The second line, <mx:Application xmlns:mx="http://www.macromedia.com/2005/mxml" xmlns="*" creationComplete="initApplication()">, says that this is an Application, and on creationComplete (ie. once the Flash has loaded), run the function initApplication();

The next section:

```
<mx:DataGrid dataProvider="{PHPData}">
        <mx:columns>
            <mx:DataGridColumn headerText="Userid" columnName="userid"/>
            <mx:DataGridColumn headerText="User Name"
columnName="username"/>
            <mx:DataGridColumn headerText="User Name"
columnName="emailaddress"/>
        </mx:columns>
</mx:DataGrid>
```

Is a datagrid that will display our list of users. Unlike the first article, this shows all three columns from the database. Note that the columnName must match the field name in the MySQL table.

The next section:

```
<mx:Script>
        <![CDATA[
            [Bindable]
            public var PHPData:Array;

            import flash.net.Responder;

            public var gateway : RemotingConnection;

            public function initApplication()
            {
                gateway = new RemotingConnection(
"http://localhost/flex/php/gateway.php" );
                gateway.call( "sample.getUsers", new Responder(onResult,
onFault));
            }

            public function onResult( result : Array ) : void
            {
```

```
                PHPData = result;
        }


        public function onFault( fault : String ) : void
        {
                trace( fault );
        }
        ]]>
    </mx:Script>
```

Is a set of ActionScript. Here's what it does:

```
[Bindable]
public var PHPData:Array;
```

declares an array, PHPData, that can be bound ([Bindable]) to MXML objects. In this case, the PHPData array is being bound to the <mx:DataGrid as its dataProvider (meaning that the DataGrid should get its data from this variable, in our case this is going to be an array of PHP objects).

The next code is required to get AMFPHP working in Flex:

```
import flash.net.Responder;

public var gateway : RemotingConnection;

public function initApplication()
{
        gateway = new RemotingConnection( "http://localhost/flex/php/gateway.php" );
        gateway.call( "sample.getUsers", new Responder(onResult, onFault));
}

public function onResult( result : Array ) : void
{
        PHPData = result;
}


public function onFault( fault : String ) : void
{
        trace( fault );
}
```

First, we import the flash.net.Responder package, which we need for remoting. Actually, Flex will include this automatically when we declare that we need a new Responser later on, but we'll include it here just to be clear what's being included.

Then, we set a variable, gateway, to be a RemotingConnection datatype. The RemotingConnection datatype is known because we have a file in the root of our Flex

application called RemotingConnection.as, which has the following inside it:

```
package
{
	import flash.net.NetConnection;
	import flash.net.ObjectEncoding;

	public class RemotingConnection extends NetConnection
	{
		public function RemotingConnection( sURL:String )
		{
			objectEncoding = ObjectEncoding.AMF0;
			if (sURL) connect( sURL );
		}

		public function AppendToGatewayUrl( s : String ) : void
		{
			//
		}
	}
}
```

Note that the class name is the same as the file name… In our case, we are adding 1 method to the NetConnection package, AppendToGatewayUrl.

Next, we see a function, initApplication

```
public function initApplication()
{
	gateway = new RemotingConnection( "http://localhost/flex/php/gateway.php" );
	gateway.call( "sample.getUsers", new Responder(onResult, onFault));
}
```

This gets run when the Flash application gets loaded. We set the variable gateway to be a new connection to the PHP script “gateway.php” that came with AMFPHP. Be sure to set the proper path to that file here. Then, in the next line we call the function getUsers on the sample class (sample.getUsers). On response, we run one of two functions: onResult if things go well, or onFault if we get an error.

Next:

```
public function onResult( result : Array ) : void
{
	PHPData = result;
}
```

Simply sets the variable PHPData to the variable result, which was passed back to ActionScript via AMFPHP. This is the array of PHP objects that we got when we ran the MySQL query ($ArrayOfUsers). AMFPHP has translated that PHP array of objects into an ActionScript array of objects for us. Pretty cool eh!

Finally:

```
public function onFault( fault : String ) : void
{
        trace( fault );
}
```
In this case, if we've got an error, we should handle it here, likely by displaying a message to the user. In our case, we simply trace the fault variable, which is useful if you're running the Flex application in debug mode.

Things are still a bit complicated, so let's be clear about what files go where. There are 3 files that you've created, and a bunch of files that you got from AMFPHP.

Flex Project Files. Put these in the same directory.
sample.mxml
RemotingConnection.as

PHP Files:
AMFPHP directories, like actions, adapters, app, browser etc…
In the services directory, you have a file called sample.php.

So, there you go, a very simple application that shows how to connect a PHP backend to the front end, built in Flex. Contact me to let me know what you come up with!