Dasher Web Service
ADOBE FLASH USER/DEVELOPER DOCUMENTATION
July 2010
Version 1.0

This documentation is to be used in conjunction with the
Dasher Web Service User/Developer Documentation. Further,
this document assumes some familiarity with both Web
Services and Adobe Flash™. All code examples are in
ActionScript 2.0.

Credit for the instructional design theory upon which Dasher is based goes to Jim Pusack
and Sue Otto of The University of Iowa
Table of Contents

1 USING A WEB SERVICE IN FLASH ......................................................... 3
  1.1 Two Basic Options ........................................................................ 3

2 SETTING UP A WEB SERVICE ................................................................. 4
  2.1 Adding the Component to Your Project's Library ............................. 4
  2.2 Setup Using the WebServiceConnector Component ....................... 6
  2.3 Setup Using ActionScript Code .................................................... 12

3 CALLING A WEB SERVICE ...................................................................... 13
  3.1 About Callback Functions ................................................................ 13
  3.2 Creating a Web Service Listener ................................................... 14
  3.3 Making a Web Service Listener Listen .......................................... 14
  3.4 Triggering a Web Service ............................................................. 14

4 PUTTING IT ALL TOGETHER ................................................................. 15

APPENDIX A: ACTIONSCRIPT SYNTAX/EXAMPLES .................................. 16

  Syntax Conventions and Explanation .................................................. 16
  DasherInit ......................................................................................... 17
  DasherDispose .................................................................................. 18
  DasherMessage ................................................................................ 18
  DasherMarkup .................................................................................. 19
  DasherGetMarkedResponse ................................................................ 19
  DasherGetObject .............................................................................. 20
  DasherGetResponseInfo .................................................................... 20
  DasherGetFeedback .......................................................................... 21
  DasherSetCaseMatters ....................................................................... 21
  DasherSetJudgeWholeWords ............................................................. 22
  DasherSetCriticalLength .................................................................. 22
  DasherSetAuthorMarks ..................................................................... 23
  DasherGetAuthorMarks .................................................................... 23
  DasherSetMarkupCharacters ............................................................. 24
  DasherGetMarkupCharacters ............................................................. 24
  DasherSetNonJudgedCharacters ....................................................... 25
  DasherGetNonJudgedCharacters ....................................................... 25
  DasherSetJudgedCharacters .............................................................. 26
  DasherGetJudgedCharacters .............................................................. 26
  DasherSetFillInBlank ....................................................................... 27
  DasherGetFillInBlank ...................................................................... 27
  DasherCheckItem .............................................................................. 28
  DasherMakeFillIn ............................................................................. 28
  DasherGetFillInPrompt ..................................................................... 29
  DasherGetFillInObj ......................................................................... 29
  DasherGetFillInResponse .................................................................. 30

APPENDIX B: DASHERFIELD OBJECT ................................................... 31
1 Using a Web Service in Flash

1.1 Two Basic Options

There are two ways that Web Services are commonly used from within Flash. One method utilizes Flash’s WebServiceConnector component for almost all the work, and the other does much of the work using ActionScript code.

Both methods require the WebServiceConnector component, which is included with Flash Professional. But only the first method uses an instance of the component on the stage. The difference lies in how the parameters which the Web Service expects are conveyed to it.

In the first method, each call made requires an instance of the WebServiceConnector component on the stage. All parameters that you pass to the component are specified in the Component Inspector. These are usually bound to Flash text fields on the stage. Note that the Component is not visible at run-time, and the parameter text fields can be made invisible as well if desired.

In the second method, the information specific to each call is specified in Actionscript code. This requires that the WebServiceConnector component exist in the Flash project's Library, but no instance of the component is put on the stage. Instead, Actionscript code is used to set all of the values that are required (which, in the first method, you would type directly into the component's fields using the Component Inspector). This allows the developer greater flexibility in setting the values and bindings than does the first method, at the expense of having to write some code.

The rest of this document will show you how to use both of these methods to incorporate calls to Web Services into your Flash project.
2 Setting Up a Web Service

2.1 Adding the Component to Your Project's Library

1. Make sure that the components panel and the library window are open. If they are not, open those using the window pull down menu in Flash.
2. Now put the WebServiceConnector component into your project's Library by dragging and dropping from the Components panel to the Library window.

If you are using the second method (Actionscript coding) to specify the details of the Web Service, then you can skip to the section entitled "Setup Using ActionScript Code"

Otherwise continue to Step 3.
2.2 Setup Using the WebServiceConnector component

3. From the Library window, drag and drop the WebServiceConnector component onto the stage. It doesn't have to be on the "active" part of the stage; it can be in the grey area "off-stage" as well. In either case, however, the component will not be visible at run-time.

It is important, however, that the component be on the first frame of the Flash timeline; the WebServiceConnector component will not function unless it is on the first frame. Note that this does not mean that code that calls the Web Service has to reside on the first frame. The component must still be on the stage, however, at the time of the call. This can be accomplished by using a separate layer for WebServiceConnector components, and keeping that keyframe active across the timeline, wherever it can be called, as shown below:
4. Now you are ready to fill in necessary values and create associations used by the Web Service.

Make sure that the *component inspector* panel is open. If it is not, open it using the *window* pull down menu.
5. Click on the new WebServiceConnector component on stage. Use the Properties inspector to name the instance of the WebServicesComponent on stage. This will be necessary for later steps, so it's best to do it now. You can change the name later if you want.

6. Now go to the parameters tab in the component inspector.

Under WSDLURL, paste the URL to the WSDL of the web service you want to use. Typically, this URL ends in “?wsdl”.

To obtain the URL for the Dasher Web Service, register a developer ID at:

http://ddra.its.uiowa.edu

7. Under operation, select the method that you wish to call using the now available drop-down menu.

In the example above the user is specifying that, for this particular instance of the WebServiceConnector component, the dasherInit method of the Dasher Web Service will be called.

At this point we have defined the call to make, and the Web Service which contains that call, but have yet to define how to pass input variables.
8. Now we will bind the input variables to web services. Binding a value to a field in the web service will tell Flash what to pass for that field every time the web service is called. Click on the bindings tab. Make sure that the WebServiceConnector component (the blue globe) icon is still selected on stage.

![Component Inspector](image)

9. Click on the ‘+’ icon to add a binding. In the box that pops up, select the field in the web service that you want to bind to.

![Add Binding](image)
10. Now you need to supply the value to this field of the web service. To do this, select the newly added binding in the component inspector and click twice in the *bound to* field. A window will popup showing the list of containers (normally text fields) to which you can bind the web service field.

![Component Inspector](image1.png)

11. In the dialog box that pops up, you can either:
   a. Hardcode the value using a constant,
b. or, tell Flash to read the value from a field on the screen. For instance, if you create a textbox on the stage and name it drillId using the properties tab, you will see it in the dialog box that pops up when you click the bound to field twice.

12. In a similar manner, bind all the remaining fields of the web service to either constant values or to text fields or other fields on the screen. Note that you can hide fields on the screen by setting the visible attribute to false in the parameters tab of the component inspector pane.

Note that any text fields referenced by the WebServiceConnector component must exist on the timeline at the time calls to the Web Service are made. For example, if a text field is bound to a parameter passed to the Web Service, and a text field with that name is not present on frame 11 (as an intervening keyframe has changed that layer's contents), then any Web Service calls made on frame 11 will fail.

If a text field with the same name exists on frame 11—even if it's a different instance—the binding will occur. This allows you to change the parameters passed on a frame by frame basis while still using hardcoded text fields mapped to the input variables. Another option is to keep the same text field on stage the whole time, and using Actionscript to change the text in that field if necessary.
2.3 Setup Using ActionScript Code

You can also use ActionScript code to set up a call to a web service.

Sample ActionScript code for setting up a web service is as follows:

```actionscript
0  import mx.data.components.WebServiceConnector;
1  var wsConn:WebServiceConnector;
2  wsConn = new WebServiceConnector();
3  wsConn.addEventListener("result", wscListener);
4  wsConn.WSDLURL = string_containing_WSDL_URL;
5  wsConn.operation = "dasherMarkup";
6  wsConn.params = [sessionId, "c'est la vie", "say la vee"];```

Explanation of code:

**Line 0:** Tells Flash you are using the WebServiceConnector component.

**Line 1:** Declares a new variable of type `WebServiceConnector` with the name `wsConn`.

**Line 2:** Creates a new object and assigns it to the variable that was created in line 1.

**Line 3:** `wscListener` is a listener object that is executed after the Web Service sends a `result` event. The `result` event is sent when the Web Service has finished execution. See the section, "Creating a Web Service Listener" for more information about listeners and callback functions.

**Line 4:** Sets the URL of the web service you want to call.

To obtain the URL for the Dasher web service, register a developer ID at:

http://ddra.its.uiowa.edu

**Line 5:** Sets the name of the method that you want to call.

**Line 6:** Sets the parameters you want to pass to the chosen method. Enumerations are passed as strings. You must supply values for all parameters, or the call to the Web Service may not work.

If you receive an error message in Flash's output window when testing your Web Service call, chances are it will look something like this:

```actionscript
error opening URL "http://...[Dasher url]...
```

This error message is returned almost exclusively, no matter what is wrong with the your Web Service code. First check that the URL and method name are correct. If they are, the problem is likely in the parameter code. See the Flash samples for examples of how to set the parameters for Dasher calls.

Remember that you still must follow steps 1 and 2 in the previous section, "Setting Up A Web Service," to place the `WebServiceConnector` component in your Flash project's Library.
3 Calling a Web Service

Whichever method above that you used to set up the Web Service call, you use ActionScript to call or "trigger" the Web Service. To use Web Services in Flash, you must include this code at the top of your page of code:

```actionscript
import mx.data.components.WebServiceConnector;
```

3.1 About Callback Functions

Flash uses callback functions to execute code when a web service call returns. The way callback functions execute is as follows:

1. Before calling the web service, we indicate that when the call is complete, a specific method should be called.
2. The web service is triggered, and the flow of execution immediately continues to the next line.
3. When the web service call is complete and the result is available, a new thread of execution is started which calls the method defined in step 1.
3.2 Creating a Web Service Listener
A web service listener will wait until the result of the web service is available. It will then execute the code under the function assigned to `result`. The value returned by the web service (if any) can be accessed using `evt.target.results`.

```javascript
var wscListener:Object = new Object();
wscListener.result = function (evt:Object) {
    var result_of_web_service = evt.target.results;
    // Do something
};
```

3.3 Making a Web Service Listener Listen
Once the web service listener has been created, you need to instruct it to listen to the web service for the occurrence of the event `result`. This is done by adding the web service listener using the `addEventListener` method of the web service connector, as shown here.

```javascript
dasherWS_dasherInit.addEventListener("result", wscListener);
```

3.4 Triggering a Web Service
Triggering (or calling) a web service is simply done by calling the trigger method of the web service connector. The name of this connector can be found in the properties window.

```javascript
dasherWS_dasherInit.trigger();
```
This calls the Web Service; your Flash actionscript code, however, continues to execute. It does NOT wait for the Web Service call to finish. Therefore, it is standard practice to make Web Service calls the last line of a function, after which nothing is executing. The next action to be taken is put in the callback function as detailed above. The `listener` detects when the Web Service is done and the `result` function is called. At this point you can get the value returned by the Web Service.
4 Putting It All Together

In short, in order to call a web service from Flash and process its results, the following steps need to be followed:

1. Set up the web service connector object. You must have the WebServiceConnector component in your library [see Setting Up A Web Service, steps 1 & 2].

2. From there, you can drag a copy of the WebServiceConnector to the stage, once for each method you want to use, and bind the parameters, either to constants or to fields on the screen. [see Setting Up A Web Service, steps 3 to 12] ; or, you can programmatically set up the calls [see Setting Up A Web Service Using ActionScript], assigning all the parameters to the webServiceConnector.params.

3. Whichever method above that you used to set up the Web Service call, you use ActionScript to call or “trigger” the Web Service. Create a listener object and encode into its function (the one assigned to result) whatever you want done when the web service call is complete [see Creating A Web Service Listener].

4. Add the listener object as an event listener [Making the Listener Listen].

5. Trigger (call) the web service! [Triggering a Web Service]
Appendix A: ActionScript Syntax/Examples

Syntax Conventions and Explanation

For a generic web service call named "WebServiceCall", the following conventions will be used.

The variable for the WebServiceConnector object will use the name of the web service call itself. In practice, this variable can be called anything.

```actionscript
var webServiceCall:WebServiceConnector = new WebServiceConnector();
```

The variable "dasherURL" must hold the address of the Dasher web service. This is obtained during registration along with your developer ID.

```actionscript
WebServiceCall.WSDLURL = dasherURL;
```

The operation parameter is set to the exact name of the actual call to be executed.

```actionscript
UserServiceCall.operation = "UserServiceCall";
```

The two parameters below are optional, and can be true or false depending on the needs of the developer. If unsure, use the values as listed.

```actionscript
UserServiceCall.multipleSimultaneousAllowed = false;
UserServiceCall.suppressInvalidCalls = true;
```

The listener object for retrieving results will be named resultListener, but can be named anything the developer chooses.

```actionscript
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
    var resultType:ResultType = evt.target.results;
}
UserServiceCall.addEventListener("result", webServiceCallListener);
```

All of the parameters needed for the call, and their types, will be declared immediately prior to the web service call. The exceptions is sessionID, which is a Number returned by dasherInit, and must be kept by the developer for use in subsequent calls. The key lines—setting the parameters and making the web service call—are highlighted in bold.

```actionscript
var param1:varType = sampleValue1;
var param2:varType = sampleValue2;
var param3:varType = sampleValue3;
UserServiceCall.params = [sessionID, param1, param2, param3...];
UserServiceCall.trigger();
```
dasherInit

// The 5 lines below create the web service connector object
// for the dasherInit call. These lines are needed once,
// and go near beginning of the program
var dasherInit:WebServiceConnector = new WebServiceConnector();
dasherInit.WSDLURL = dasherURL;  // the Dasher web service URL
dasherInit.operation = "dasherInit";
dasherInit.multipleSimultaneousAllowed = false;
dasherInit.suppressInvalidCalls = true;

// the lines below create the listener object
// and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
    var theSessionID:Number = evt.target.results;
    // note: MUST retrieve this and put into global variable
    _root.sessionID = theSessionID;
}
dasherInit.addEventListener("result", resultListener);

// the code below creates variables and assigns values
// note that developerID is obtained via registration
var developerID:String = "your developer ID";
var interactionID:String = "Presidents Drill";
var studentID:String = "Joe Student";
var language:String = "English";

// the line below assigns parameters just before calling
dasherInit.params = [developerID, interactionID, studentID, language];

// the line below actually calls the Web Service
dasherInit.trigger();
dasherDispose

// The 5 lines below create the web service connector object
// for the dasherDispose call. These lines are needed once,
// and go near beginning of the program
var dasherDispose:WebServiceConnector = new WebServiceConnector();
dasherDispose.WSDLURL = dasherURL;  // the Dasher web service URL
dasherDispose.operation = "dasherDispose";
dasherDispose.multipleSimultaneousAllowed = false;
dasherDispose.suppressInvalidCalls = true;

// the lines below create the listener object
// and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
   var errCode:Number = evt.target.results;
}
dasherDispose.addEventListener("result", resultListener);

// the line below assigns parameters just before calling
dasherDispose.params = [sessionID];

// the line below actually calls the Web Service
dasherDispose.trigger();

dasherMessage

// The 5 lines below create the web service connector object
// for the dasherMessage call. These lines are needed once,
// and go near beginning of the program
var dasherMessage:WebServiceConnector = new WebServiceConnector();
dasherMessage.WSDLURL = dasherURL;  //the Dasher web service URL
dasherMessage.operation = "dasherMessage";
dasherMessage.multipleSimultaneousAllowed = false;
dasherMessage.suppressInvalidCalls = true;

// the lines below create the listener object
// and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
   var msgString:String = evt.target.results;
}
dasherMessage.addEventListener("result", resultListener);

// the line below assigns parameters just before calling
dasherMessage.params = [sessionID];

// the line below actually calls the Web Service
dasherMessage.trigger();
dasherMarkup

// The 5 lines below create the web service connector object
// for the dasherMarkup call. These lines are needed once,
// and go near beginning of the program
var dasherMarkup:WebServiceConnector = new WebServiceConnector();
dasherMarkup.WSDLURL = dasherURL; // the Dasher web service URL
dasherMarkup.operation = "dasherMarkup";
dasherMarkup.multipleSimultaneousAllowed = false;
dasherMarkup.suppressInvalidCalls = true;

// the lines below create the listener object
// and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
    var correct:Number = evt.target.results;
}
dasherMarkup.addEventListener("result", resultListener);
// the lines below set the object and response values to pass
var obj:String = "fourscore and seven years ago";
var resp:String = "foursquare & 7 ears ago";

// the line below assigns parameters just before calling
dasherMarkup.params = [sessionID, obj, resp];
// the line below actually calls the Web Service
dasherMarkup.trigger();

dasherGetMarkedResponse

// The 5 lines below create the web service connector object
// for the dasherGetMarkedResponse call. These lines are needed once,
// and go near beginning of the program
var dasherGetMarkedResponse:WebServiceConnector = new WebServiceConnector();
dasherGetMarkedResponse.WSDLURL = dasherURL; // the Dasher web service URL
dasherGetMarkedResponse.operation = "dasherGetMarkedResponse";
dasherGetMarkedResponse.multipleSimultaneousAllowed = false;
dasherGetMarkedResponse.suppressInvalidCalls = true;

// the lines below create the listener object
// and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
    var markedResponse:String = evt.target.results;
}
dasherGetMarkedResponse.addEventListener("result", resultListener);
// the line below assigns parameters just before calling
dasherGetMarkedResponse.params = [sessionID];
// the line below actually calls the Web Service
dasherGetMarkedResponse.trigger();
**dasherGetObject**

// The 5 lines below create the web service connector object // for the dasherGetObject call. These lines are needed once, // and go near beginning of the program
var dasherGetObject:WebServiceConnector = new WebServiceConnector();
dasherGetObject.WSDLURL = dasherURL; // the Dasher web service URL
dasherGetObject.operation = "dasherGetObject";
dasherGetObject.multipleSimultaneousAllowed = false;
dasherGetObject.suppressInvalidCalls = true;

// the lines below create the listener object // and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
  var obj:Number = evt.target.results;
}  
dasherGetObject.addEventListener("result", resultListener);

// the line below assigns parameters just before calling
dasherGetObject.params = [sessionID];

// the line below actually calls the Web Service
dasherGetObject.trigger();

**dasherGetResponseInfo**

// The 5 lines below create the web service connector object // for the dasherGetResponseInfo call. These lines are needed once, // and go near beginning of the program
var dasherGetResponseInfo:WebServiceConnector = new WebServiceConnector();
dasherGetResponseInfo.WSDLURL = dasherURL; // the Dasher web service URL
dasherGetResponseInfo.operation = "dasherGetResponseInfo";
dasherGetResponseInfo.multipleSimultaneousAllowed = false;
dasherGetResponseInfo.suppressInvalidCalls = true;

// the lines below create the listener object // and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
  var responseInfo:String = evt.target.results;
}  
dasherGetResponseInfo.addEventListener("result", resultListener);

// the line below assigns parameters just before calling
dasherGetResponseInfo.params = [sessionID];

// the line below actually calls the Web Service
dasherGetResponseInfo.trigger();
dasherGetFeedback

    // The 5 lines below create the web service connector object
    // for the dasherGetFeedback call. These lines are needed once,
    // and go near beginning of the program
    var dasherGetFeedback:WebServiceConnector = new WebServiceConnector();
    dasherGetFeedback.WSDLURL = dasherURL; // the Dasher web service URL
    dasherGetFeedback.operation = "dasherGetFeedback";
    dasherGetFeedback.multipleSimultaneousAllowed = false;
    dasherGetFeedback.suppressInvalidCalls = true;
    // the lines below create the listener object
    // and define the callback function; done once
    var resultListener:Object = new Object();
    resultListener.result = function(evt:Object)
    {
        var feedbackMsg:String = evt.target.results;
    }
    dasherGetFeedback.addEventListener("result", resultListener);
    
    // the line below assigns parameters just before calling
    dasherGetFeedback.params = [sessionID];
    // the line below actually calls the Web Service
    dasherGetFeedback.trigger();

dasherSetCaseMatters

    // The lines below create the web service connector object
    // for the dasherGetFeedback call. These lines are needed
    // once, and go near beginning of the program
    var dasherSetCaseMatters:WebServiceConnector = new WebServiceConnector();
    dasherSetCaseMatters.WSDLURL = dasherURL; // the dasher web service URL
    dasherSetCaseMatters.operation = "dasherSetCaseMatters";
    dasherSetCaseMatters.multipleSimultaneousAllowed = false;
    dasherSetCaseMatters.suppressInvalidCalls = true;

    // the lines below create the listener object
    // and define the callback function; done once
    var resultListener:Object = new Object();
    resultListener.result = function(evt:Object)
    {
        var error:Number = evt.target.results;
    }
    dasherSetCaseMatters.addEventListener("result", resultListener);

    // the code below creates a variable to pass;
    // reassign as needed
    var caseMatters:Boolean = true;
    // the line below assigns parameters just before calling
    dasherSetCaseMatters.params = [sessionID, caseMatters];

    // the line below actually calls the Web Service
    dasherSetCaseMatters.trigger();
dasherSetJudgeWholeWords

// The lines below create the web service connector object
// for the dasherSetJudgeWholeWords call. These lines are needed
// once, and go near beginning of the program
var dasherSetJudgeWholeWords:WebServiceConnector = new WebServiceConnector();
dasherSetJudgeWholeWords.WSDLURL = dasherURL; // the dasher web service URL
dasherSetJudgeWholeWords.operation = "dasherSetJudgeWholeWords";
dasherSetJudgeWholeWords.multipleSimultaneousAllowed = false;
dasherSetJudgeWholeWords.suppressInvalidCalls = true;

// the lines below create the listener object
// and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
    var errCode:Number = evt.target.results;
}
dasherSetJudgeWholeWords.addEventListener("result", resultListener);

// the code below creates a variable to pass;
// set to true or false depending on your needs
var judgeWholeWords:Boolean = true;

// the line below assigns parameters just before calling
dasherSetJudgeWholeWords.params = [sessionID, judgeWholeWords];
// the line below actually calls the Web Service
dasherSetJudgeWholeWords.trigger();

dasherSetCriticalLength

// The lines below create the web service connector object
// for the dasherSetCriticalLength call. These lines are needed
// once, and go near beginning of the program
var dasherSetCriticalLength:WebServiceConnector = new WebServiceConnector();
dasherSetCriticalLength.WSDLURL = dasherURL; // the dasher web service URL
dasherSetCriticalLength.operation = "dasherSetCriticalLength";
dasherSetCriticalLength.multipleSimultaneousAllowed = false;
dasherSetCriticalLength.suppressInvalidCalls = true;

// the lines below create the listener object
// and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
    var errorCode:String = evt.target.results;
}
dasherSetCriticalLength.addEventListener("result", resultListener);

// the code below creates a variable to pass;
// reassign to set a different value
var criticalLength:Number = 4;

// the line below assigns parameters just before calling
dasherSetCriticalLength.params = [sessionID, criticalLength];
// the line below actually calls the Web Service
dasherSetCriticalLength.trigger();
dasherSetAuthorMarks
   // The 5 lines below create the web service connector object
   // for the dasherSetAuthorMarks call. These lines are needed once,
   // and go near beginning of the program
   var dasherSetAuthorMarks:WebServiceConnector = new WebServiceConnector();
   dasherSetAuthorMarks.WSDLURL = dasherURL; // the Dasher web service URL
   dasherSetAuthorMarks.operation = "dasherSetAuthorMarks";
   dasherSetAuthorMarks.multipleSimultaneousAllowed = false;
   dasherSetAuthorMarks.suppressInvalidCalls = true;
       // the lines below create the listener object
       // and define the callback function; done once
   var resultListener:Object = new Object();
   resultListener.result = function(evt:Object)
   {
       var errorCode:Number = evt.target.results;
   }
   dasherSetAuthorMarks.addEventListener("result", resultListener);
       // the code below creates a variables and assigns it;
       // to change the set of characters used, reassign
       // the string as needed before the .param assignment
   var authorMarks:String = "[]/*{}";
       // the line below assigns parameters just before calling
   dasherSetAuthorMarks.params = [sessionID, authorMarks];
       // the line below actually calls the Web Service
   dasherSetAuthorMarks.trigger();


dasherGetAuthorMarks
   // The 5 lines below create the web service connector object
   // for the dasherGetAuthorMarks call. These lines are needed once,
   // and go near beginning of the program
   var dasherGetAuthorMarks:WebServiceConnector = new WebServiceConnector();
   dasherGetAuthorMarks.WSDLURL = dasherURL; // the Dasher web service URL
   dasherGetAuthorMarks.operation = "dasherGetAuthorMarks";
   dasherGetAuthorMarks.multipleSimultaneousAllowed = false;
   dasherGetAuthorMarks.suppressInvalidCalls = true;
       // the lines below create the listener object
       // and define the callback function; done once
   var resultListener:Object = new Object();
   resultListener.result = function(evt:Object)
   {
       var authorMarks:String = evt.target.results;
   }
   dasherGetAuthorMarks.addEventListener("result", resultListener);
       // the line below assigns parameters just before calling
   dasherGetAuthorMarks.params = [sessionID];
       // the line below actually calls the Web Service
   dasherGetAuthorMarks.trigger();
**dasherSetMarkupCharacters**

// The lines below create the web service connector object
// for the ddGenerateHardestItemList call. These lines are needed
// once, and go near beginning of the program
var dasherSetMarkupCharacters:WebServiceConnector
    = new WebServiceConnector();

dasherSetMarkupCharacters.WSDLURL = dasherURL;
// the dasher web service URL

dasherSetMarkupCharacters.operation = "dasherSetMarkupCharacters";
dasherSetMarkupCharacters.multipleSimultaneousAllowed = false;
dasherSetMarkupCharacters.suppressInvalidCalls = true;
// the lines below create the listener object
// and define the callback function; done once
var resultListListener:Object = new Object();
resultListListener.result = function(evt:Object)
{
    var errorCode:Number = evt.target.results;
}
dasherSetMarkupCharacters.addEventListener("result", resultListListener);
// the code below creates a variable and sets it to the
// characters to pass; reassign as necessary
var markupCharacters:String = "<>!~$";
// the line below assigns parameters just before calling
dasherSetMarkupCharacters.params
    = [sessionID, markupCharacters];
// the line below actually calls the Web Service
dasherSetMarkupCharacters.trigger();

**dasherGetMarkupCharacters**

// The 5 lines below create the web service connector object
// for the dasherGetMarkupCharacters call. These lines are needed
// once, and go near beginning of the program
var dasherGetMarkupCharacters:WebServiceConnector =
    new WebServiceConnector();


dasherGetMarkupCharacters.WSDLURL = dasherURL;
// the Dashr web service URL

dasherGetMarkupCharacters.operation = "dasherGetMarkupCharacters";
dasherGetMarkupCharacters.multipleSimultaneousAllowed = false;
dasherGetMarkupCharacters.suppressInvalidCalls = true;
// the lines below create the listener object
// and define the callback function; done once
var resultListListener:Object = new Object();
resultListListener.result = function(evt:Object)
{
    var markupCharacters:String = evt.target.results;
}
dasherGetMarkupCharacters.addEventListener("result", resultListListener);
// the line below assigns parameters just before calling
dasherGetMarkupCharacters.params = [sessionID];
// the line below actually calls the Web Service
dasherGetMarkupCharacters.trigger();
dasherSetNonJudgedCharacters

// The lines below create the web service connector object
// for the ddGenerateHardestItemList call. These lines are needed
// once, and go near beginning of the program
var dasherSetNonJudgedCharacters:WebServiceConnector
  = new WebServiceConnector();
dasherSetNonJudgedCharacters.WSDLURL = dasherURL;
  // the dasher web service URL
dasherSetNonJudgedCharacters.operation = "dasherSetNonJudgedCharacters";
dasherSetNonJudgedCharacters.multipleSimultaneousAllowed = false;
dasherSetNonJudgedCharacters.suppressInvalidCalls = true;
  // the lines below create the listener object
  // and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
  var errorCode:Number = evt.target.results;
}
dasherSetNonJudgedCharacters.addEventListener("result", resultListener);
  // the code below creates a variable and sets it to the
  // characters to pass; reassign as necessary
var nonJudgedCharacters:String = "<>!~$";
  // the line below assigns parameters just before calling
dasherSetNonJudgedCharacters.params
  = [sessionID, nonJudgedCharacters];
  // the line below actually calls the Web Service
dasherSetNonJudgedCharacters.trigger();

dasherGetNonJudgedCharacters

// The 5 lines below create the web service connector object
// for the dasherGetNonJudgedCharacters call. These lines are needed
// once, and go near beginning of the program
var dasherGetNonJudgedCharacters:WebServiceConnector
  = new WebServiceConnector();
dasherGetNonJudgedCharacters.WSDLURL = dasherURL;
  // the Dasher web service URL
dasherGetNonJudgedCharacters.operation = "dasherGetNonJudgedCharacters";
dasherGetNonJudgedCharacters.multipleSimultaneousAllowed = false;
dasherGetNonJudgedCharacters.suppressInvalidCalls = true;
  // the lines below create the listener object
  // and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
  var nonJudgedCharacters:String = evt.target.results;
}
dasherGetNonJudgedCharacters.addEventListener("result", resultListener);
  // the line below assigns parameters just before calling
dasherGetNonJudgedCharacters.params = [sessionID];
  // the line below actually calls the Web Service
dasherGetNonJudgedCharacters.trigger();
dasherSetJudgedCharacters
   // The lines below create the web service connector object
   // for the ddGenerateHardestItemList call. These lines are needed
   // once, and go near beginning of the program
var dasherSetJudgedCharacters:WebServiceConnector
 = new WebServiceConnector();
dasherSetJudgedCharacters.WSDLURL = dasherURL;
   // the dasher web service URL
dasherSetJudgedCharacters.operation = "dasherSetJudgedCharacters";
dasherSetJudgedCharacters.multipleSimultaneousAllowed = false;
dasherSetJudgedCharacters.suppressInvalidCalls = true;
   // the lines below create the listener object
   // and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
   var errorCode:Number = evt.target.results;
}
dasherSetJudgedCharacters.addEventListener("result", resultListener);
   // the code below creates a variable and sets it to the
   // characters to pass; reassign as necessary
var judgedCharacters:String = "AEIOU";
   // the line below assigns parameters just before calling
dasherSetJudgedCharacters.params
 = [sessionID, judgedCharacters];
   // the line below actually calls the Web Service
dasherSetJudgedCharacters.trigger();


dasherGetJudgedCharacters
   // The 5 lines below create the web service connector object
   // for the dasherGetJudgedCharacters call. These lines are needed
   // once, and go near beginning of the program
var dasherGetJudgedCharacters:WebServiceConnector
 = new WebServiceConnector();
dasherGetJudgedCharacters.WSDLURL = dasherURL;
   // the Dasher web service URL
dasherGetJudgedCharacters.operation = "dasherGetJudgedCharacters";
dasherGetJudgedCharacters.multipleSimultaneousAllowed = false;
dasherGetJudgedCharacters.suppressInvalidCalls = true;
   // the lines below create the listener object
   // and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
   var judgedCharacters:String = evt.target.results;
}
dasherGetJudgedCharacters.addEventListener("result", resultListener);
   // the line below assigns parameters just before calling
dasherGetJudgedCharacters.params = [sessionID];
   // the line below actually calls the Web Service
dasherGetJudgedCharacters.trigger();
**dasherSetFillInBlank**  
// The lines below create the web service connector object  
// for the ddGenerateHardestItemList call. These lines are needed  
// once, and go near beginning of the program  
var dasherSetFillInBlank:WebServiceConnector  
    = new WebServiceConnector();  
dasherSetFillInBlank.WSDLURL = dasherURL;  
// the dasher web service URL  
dasherSetFillInBlank.operation = "dasherSetFillInBlank";  
dasherSetFillInBlank.multipleSimultaneousAllowed = false;  
dasherSetFillInBlank.suppressInvalidCalls = true;  
// the lines below create the listener object  
// and define the callback function; done once  
var resultListener:Object = new Object();  
resultListener.result = function(evt:Object)  
{  
    var errorCode:Number = evt.target.results;  
}  
dasherSetFillInBlank.addEventListener("result", resultListener);  
// the code below creates a variable and sets it to the  
// characters to pass; reassign as necessary  
var fillInBlank:String = "<>!~$";  
// the line below assigns parameters just before calling  
dasherSetFillInBlank.params  
    = [sessionID, fillInBlank];  
// the line below actually calls the Web Service  
dasherSetFillInBlank.trigger();

**dasherGetFillInBlank**  
// The 5 lines below create the web service connector object  
// for the dasherGetFillInBlank call. These lines are needed  
// once, and go near beginning of the program  
var dasherGetFillInBlank:WebServiceConnector  
    = new WebServiceConnector();  
dasherGetFillInBlank.WSDLURL = dasherURL;  
// the Dasher web service URL  
dasherGetFillInBlank.operation =  
    "dasherGetFillInBlank";  
dasherGetFillInBlank.multipleSimultaneousAllowed = false;  
dasherGetFillInBlank.suppressInvalidCalls = true;  
// the lines below create the listener object  
// and define the callback function; done once  
var resultListener:Object = new Object();  
resultListener.result = function(evt:Object)  
{  
    var fillInBlank:String = evt.target.results;  
}  
dasherGetFillInBlank.addEventListener("result", resultListener);  
// the line below assigns parameters just before calling  
dasherGetFillInBlank.params  
    = [sessionID];  
// the line below actually calls the Web Service  
dasherGetFillInBlank.trigger();
dasherCheckItem

// The 5 lines below create the web service connector object
// for the dasherCheckItem call. These lines are needed once,
// and go near beginning of the program
var dasherCheckItem:WebServiceConnector = new WebServiceConnector();
dasherCheckItem.WSDLURL = dasherURL;  // the Dasher web service URL
dasherCheckItem.operation = "dasherCheckItem";
dasherCheckItem.multipleSimultaneousAllowed = false;
dasherCheckItem.suppressInvalidCalls = true;
  // the lines below create the listener object
  // and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
    var errCode:Number = evt.target.results;
}
dasherCheckItem.addEventListener("result", resultListener);
  // the code below creates variables and assigns values;
  // reassign as needed for further calls
var prompt:String = "What was Lincoln's first name?";
var obj:String = "Abraham";
  // the line below assigns parameters just before calling
dasherCheckItem.params = [sessionID, prompt, obj];
  // the line below actually calls the Web Service
dasherCheckItem.trigger();

dasherMakeFillIn

// The 5 lines below create the web service connector object
// for the dasherMakeFillIn call. These lines are needed once,
// and go near beginning of the program
var dasherMakeFillIn:WebServiceConnector = new WebServiceConnector();
dasherMakeFillIn.WSDLURL = dasherURL;  // the Dasher web service URL
dasherMakeFillIn.operation = "dasherMakeFillIn";
dasherMakeFillIn.multipleSimultaneousAllowed = false;
dasherMakeFillIn.suppressInvalidCalls = true;
  // the lines below create the listener object
  // and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
    var errCode:Number = evt.target.results;
}
dasherMakeFillIn.addEventListener("result", resultListener);
  // the code below creates variables and assigns values;
  // reassign as needed for further calls
var prompt:String = "What was Lincoln's first name?";
var obj:String = "Abraham";
  // the line below assigns parameters just before calling
dasherMakeFillIn.params = [sessionID, prompt, obj];
  // the line below actually calls the Web Service
dasherMakeFillIn.trigger();
dasherGetFillInPrompt

// The 5 lines below create the web service connector object
// for the dasherGetFillInPrompt call. These lines are needed
// once, and go near beginning of the program
var dasherGetFillInPrompt:WebServiceConnector = new WebServiceConnector();
dasherGetFillInPrompt.WSDLURL = dasherURL;// the Dasher web service URL
dasherGetFillInPrompt.operation = "dasherGetFillInPrompt";
dasherGetFillInPrompt.multipleSimultaneousAllowed = false;
dasherGetFillInPrompt.suppressInvalidCalls = true;
    // the lines below create the listener object
    // and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
    var prompt:String = evt.target.results;
}
dasherGetFillInPrompt.addEventListener("result", resultListener);

    // the line below assigns parameters just before calling
dasherGetFillInPrompt.params = [sessionID];
    // the line below actually calls the Web Service
dasherGetFillInPrompt.trigger();

dasherGetFillInObj

// The 5 lines below create the web service connector object
// for the dasherGetFillInObj call. These lines are needed
// once, and go near beginning of the program
var dasherGetFillInObj:WebServiceConnector = new WebServiceConnector();
dasherGetFillInObj.WSDLURL = dasherURL;// the Dasher web service URL
dasherGetFillInObj.operation = "dasherGetFillInObj";
dasherGetFillInObj.multipleSimultaneousAllowed = false;
dasherGetFillInObj.suppressInvalidCalls = true;
    // the lines below create the listener object
    // and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
    var obj:String = evt.target.results;
}
dasherGetFillInObj.addEventListener("result", resultListener);

    // the line below assigns parameters just before calling
dasherGetFillInObj.params = [sessionID];
    // the line below actually calls the Web Service
dasherGetFillInObj.trigger();
dasherGetFillInResponse

// The 5 lines below create the web service connector object
// for the dasherGetFillInResponse call. These lines are needed
// once, and go near beginning of the program
var dasherGetFillInResponse:WebServiceConnector = new WebServiceConnector();
dasherGetFillInResponse.WSDLURL = dasherURL; // the Dasher web service URL
dasherGetFillInResponse.operation = "dasherGetFillInResponse";
dasherGetFillInResponse.multipleSimultaneousAllowed = false;
dasherGetFillInResponse.suppressInvalidCalls = true;

// the lines below create the listener object
// and define the callback function; done once
var resultListener:Object = new Object();
resultListener.result = function(evt:Object)
{
    var response:String = evt.target.results;
}
dasherGetFillInResponse.addEventListener("result", resultListener);

// the line below assigns parameters just before calling
dasherGetFillInResponse.params = [sessionID];
// the line below actually calls the Web Service
dasherGetFillInResponse.trigger();
APPENDIX B: DasherField Object

The DasherField ActionScript object is a custom object created to work in synch with Flash's TextField object to more closely behave the way input fields should work when using Dasher.

DOS Origins and editing issues

Dasher was originally created in the days when DOS-type input was more common. In DOS systems (and terminal input systems, etc) each character occupied its own defined space, and the input cursor was "over" a character rather than in-between characters. Selection of multiple characters via mouse was not standard. This worked well for Dasher-type input/editing, where a user could only edit certain characters in a string (the characters s/he hadn't gotten correct yet).

In contrast, free-form editing and mouse-selection of the post-DOS world has made it more difficult to restrict user editing in a meaningful way for Dasher. When the selection caret is "in-between" characters (as in Microsoft Word), typing a character results in adding a new character at that spot. However for Dasher's purposes, it's best to have an editing system where a character is replaced when a key is pressed.

DasherField

DasherField is an ActionScript object that helps facilitate this kind of editing. The DasherField object maintains a 1-character selection range, so that the "cursor" is "over" a letter in the input field. Pressing a key replaces that character. If the user tries to select characters using the mouse, or clicks in another place in the field, the 1-character selection is reapplied at that point.

DasherField selects only characters that haven't been confirmed as correct by Dasher calls. If the user moves the cursor with the arrow keys, the cursor moves over only editable characters. Users can't type over the characters they already have correct.

Using DasherField

DasherField is used in the PhrasesQuiz example contained on the U of Iowa Dasher site (http://ddra.its.uiowa.edu). The code is downloadable as a .zip file.

To use DasherField, simply create a new instance and associate it with an existing TextField object, as in this code:

```javascript
var dasherField = new DasherField (responseField);  
// tells it that responseField is the TextField object to trap events for.
```

The above call links responseField (an existing Flash TextField) to the instance dasherField, which listens for events associated with responseField. That's all there is to it. Be sure to have a copy of the DasherField.as file somewhere in your classpath (having it in the same folder as your .fla file will do it).

DasherField does not completely solve the problems with input formatting, but it greatly improves input for Dasher applications. Compare text input in the PhrasesQuiz to DasherTestConsole, which allows regular-style input.