



JavaScript Integration in Flash 8

By: [Danny Patterson](#)

Flash's ability to integrate with JavaScript just took a huge leap forward with the release of Flash 8. Introducing the External Interface! This is a new feature in Flash 8 that allows for better communication between Flash and its host. Most of us will use this for integration with JavaScript. This article will show you how easy this new feature is to use. It will whet your appetite and get you thinking about how you want to use this in your applications.

The Flash 8 authoring tool has not been released at the time of this writing. It is set to be released sometime in September 2005. To view the examples shown in this article, you will need to install the [Flash Player 8 public beta](http://www.macromedia.com/software/flashplayer/public_beta/) (http://www.macromedia.com/software/flashplayer/public_beta/).

Out With the Old

In previous versions of Flash, JavaScript communication was done via `fscommand`. This was a buggy and complex technology. In fact, it was so bad that most developers actually preferred to put their JavaScript function calls inside the `getURL` function. This got the job done, but was far from ideal.

Mike Chambers and Christian Cantrell released the [Flash/JavaScript Integration Kit](http://www.communitymx.com/abstract.cfm?cid=D7491) (<http://www.communitymx.com/abstract.cfm?cid=D7491>) as a better way to handle communication between the two technologies. This made the integration much easier to implement, but was still built on the same old technology. What we really needed was a new technology.

In With the New

Flash 8 introduces a ground-breaking new way to integrate Flash with its host called the **External Interface**. This allows Flash/JavaScript integration to be more powerful and stable. It is also very easy to use. External Interface offers Flash 8 Developers the following advantages: it's easier to implement, it allows for synchronous communication, and it supports sending a wider selection of data types, including objects.

External Interface is extremely easy to implement. As you will see in the examples below, only a few lines of code are needed on either the Flash or JavaScript side. `fscommand` required a large amount of JavaScript/VBScript on the host page just to call a JavaScript function. The only code required on the host page for Flash to call a JavaScript function is the function itself. That's right, Flash can call any JavaScript function on the SWF's host page without any additional JavaScript code. Going from

JavaScript to Flash is also very simple. All you need to do in JavaScript is get a reference to your HTML object or embed tag (depending on the browser) and call the ActionScript function. Inside the Flash application, you simply need to make that function available to JavaScript by calling the `addCallback` function of the External Interface class. More on this below.

`Fscommand` didn't allow for synchronous communication. Therefore, to make a round trip, Flash would need to call a JavaScript function, the JavaScript would need to turn around and use `SetVariable` to create a variable in the Flash movie. All this time Flash was checking to see if that variable was set. This communication is far from ideal. The External Interface allows you to call a JavaScript function and receive a return value. This simplifies the required logic and should open the door for more complex applications.

`Fscommand` allowed you to send primitive data types as arguments in the function calls. Now, with External Interface, you can send complex objects as arguments. In example 2 below, we are passing the entire `system.capabilities` object back to JavaScript.

I know what you're thinking: "External Interface is great, but does it work in all the browsers?" The following is a list of supported browser versions and platforms (and yes, it is Flash 8-only):

- Internet Explorer 5.0+ (Windows)
- Netscape 8.0+ (Windows & Macintosh)
- Mozilla 1.7.5+ (Windows & Macintosh)
- Firefox 1.0+ (Windows & Macintosh)
- Safari 1.3+ (Macintosh)

Third-party projector tools use **fscommand** to allow Flash to communicate with the container application. This will also be replaced by the new External Interface. Since External Interface allows for sending objects and returning values, we can expect a huge increase in the stability and functionality of these projector tools.

Flash to JavaScript Communication Example

In the Flash to JavaScript communication example, we will ask JavaScript to tell us the host page's URL. (Within Flash we can access the SWF's URL, but not the host page.) This is a very simple example that will quickly return a string back to the Flash application.

| | |
|---|---|
| <input type="text" value="http://www.communitymx.com/content/source/0922A/Example1.htm"/> | <input type="button" value="Get Location"/> |
|---|---|

Example 1 *Example of Flash to JavaScript communication where a JavaScript function returns browser information.*

You don't need to do anything special to make a JavaScript function available to ActionScript. In the following example, you can see that we just create a function called `getLocation` that returns the URL of the page:

```
<script language="JavaScript">
function getLocation() {
    return window.location.toString();
}
</script>
```

Code 1 *JavaScript function that returns the URL of the page.*

On the Flash side, all we need to do is call that function by using the static `call` method of the `ExternalInterface` class. We do that when the button is clicked, therefore we add a listener to the button and then set the text input's display to the result of the JavaScript function call. This would not be possible in earlier versions of Flash because synchronous communication wasn't supported. The following code shows you how simple it is to call the JavaScript function from within Flash:

```
import flash.external.ExternalInterface;

function displayPageLocation():Void {
    locationDisplay.text = ExternalInterface.call("getLocation");
}

locationButton.addEventListener("click", mx.utils.Delegate.create(this,
displayPageLocation));
```

Code 2 *ActionScript that calls the JavaScript `getLocation` function when the button is clicked, and displays it in the text input.*

JavaScript to Flash Communication Example

In our JavaScript to Flash communication example, we will be stepping it up a notch by returning an object instead of just a string. We return the `system.capabilities` object from Flash, then loop through the object and output it in a textarea form element.

```
hasIME: false
language: en
os: Windows 2000
manufacturer: Macromedia Windows
windowlessDisable: false
localFileReadDisable: false
avHardwareDisable: false
playerType: PlugIn
isDebugger: false
hasScreenBroadcast: false
hasScreenPlayback: true
hasPrinting: true
hasEmbeddedVideo: true
hasStreamingVideo: true
hasStreamingAudio: true
version: WIN 8,0,15,0
serverString:
A=t&SA=t&SV=t&EV=t&MP3=t&AE=t&VE=t&ACC=f&PR=t&SP=t
hasAudio: true
hasMP3: true
hasAudioEncoder: true
hasVideoEncoder: true
```

Get Flash Info

Example 2 *Example of JavaScript to Flash communication where an ActionScript function returns Flash Player information.*

All you need to do to make your Flash function available to JavaScript is call the static `addCallback` method of the `ExternalInterface` class. The `addCallback` method has three parameters: the first is the name you want JavaScript to use when it calls your function; the second is the object this function can be found at; and the third is a reference to the function itself. As you can see in the following example, our function is very simple and just returns the `system.capabilities` object to JavaScript.

```
import flash.external.ExternalInterface;

function getFlashInfo():Object {
    return System.capabilities;
}

ExternalInterface.addCallback("getFlashInfo", this, getFlashInfo);
```

Code 3 *ActionScript code that creates the `getFlashInfo` function and makes it available to JavaScript.*

On the JavaScript side, the first thing you must do is get a reference to the object or embed tags in your page. Since these elements aren't available until the page is done loading, we'll put them inside the `window.onload` function:

```

<script language="JavaScript">
var flash;

window.onload = function() {
    if(navigator.appName.indexOf("Microsoft") != -1) {
        flash = window.flashObject;
    }else {
        flash = window.document.flashObject;
    }
}
</script>

```

Code 4 *JavaScript code that creates the reference to the Flash object.*

When the HTML form button is clicked, we call the `displayFlashInfo` function shown below. This function makes a call to the `getFlashInfo` function in the Flash application and sets its return value to the `flashInfo` variable. Then it loops through that object and outputs the property values to the textarea form element. The following code shows just how easy it is to call an ActionScript function from JavaScript. It also proves that the External Interface can transfer object data types:

```

<script language="JavaScript">
function displayFlashInfo() {
    var flashInfo = flash.getFlashInfo();
    for(var key in flashInfo) {
        document.flashForm.flashInfoDisplay.value += key + ": " +
flashInfo[key] + "\n";
    }
}
</script>

```

Code 5 *JavaScript code that calls the `getFlashInfo` function in the SWF and outputs it to the textarea form element.*

Along with your code to embed the Flash application, you will also need the following `<form>` code. It includes a textarea and a button. When the button is clicked, it invokes the method shown in the Code 5.

```

<form name="flashForm">
<textarea name="flashInfoDisplay"
style="width:390px;height:300px;"></textarea>
<input type="Button" value="Get Flash Info" name="flashInfoButton"
onclick="javascript:displayFlashInfo();" style="width:100px;" />
</form>

```

Code 6 *Clicking the button calls the `getFlashInfo` function and the textarea displays the result.*

Conclusion

As you can see, the new External Interface is a huge leap forward in Flash/JavaScript integration. This

article shows that this new feature is easy to use and has some impressive new features.

NOTE: To download the example files used in this article, click **Download Support Files** below. Please note that you will not be able to open the FLA files until you have a valid copy of Flash 8. Flash 8 is set to be released sometime in September 2005.

Approximate download size: 266k

Keywords

ExternalInterface, External, Interface, sneak peek, Flash 8, fscommand, javascript, integration, communication

All content ©CommunityMX 2002-2004. All rights reserved.