Actions Tech. Note 9

Draft 1.0

File Parameters and File Objects

(Using Files in Freeway Actions - lifting the bonnet)

The JavaScript API facilitates the reading and writing of files. This document goes some way to explaining the interrelation of files, file parameters, uploading and related matters.

Files in Freeway are handled by two distinct entities.

- **File Objects** - these allow the actions writer to read and write files from within JavaScript.
- **File Parameters** - these allow the user to specify files in the Actions UI.

File Objects

The file object `FWFile` allows you to read and write files.

Reading Files

If you want to read a file you will have to do the following:

- Create a new `FWFile`.
- Open this file for reading, and in doing so specify which physical file you want to read
- Read whatever you want from the file.
- Close the file.

The following Action will open and the first line of a text file when you click on the button.

```
<item-action name="Test">
<action-button name="read file" onclick = "DoRead()"/>
<action-javascript>

  function DoRead()
  {
    // create a new file object

    myFile = new FWFile;

    // choose a file of type text with the prompt 'Locate File'
    // and open it

    if (myFile.fwLocateRead('Locate File','TEXT'))
    {
      // read a line
      alert(myFile.fwReadln());

      // close the file
      myFile.fwClose();
    }
  }

</action-javascript>
</item-action>
```

There are two ways to open a text file for reading:

- `fwLocateRead` - this prompts the user to locate a file on their disk for reading.
- `fwOpenRead` - this allows you to specify directly which file you want to read from.

Writing Files

If you want to write to a file you will have to do the following:

- Create a new `FWFile`.
- Open this file for writing, and in doing so specify which physical file you want to write to.
- Write whatever you want to the file.
- Close the file.

This creates a writes "test" into it and then closes it.

```
<item-action name="Test">
<action-button name="locate" onclick = "DoLocate()"/>
<action-javascript>

  function DoLocate()
  {
    // create a new file object
    myFile = new FWFile;

    // get the user to create the file
    var wasChosen = myFile.fwLocateWrite('Locate a NEW File',"My File",'TEXT');
    if (wasChosen)
    {
      myFile.fwWrite('test');
      myFile.fwClose();
    }
  }

</action-javascript>
```

Freeway is quite restrictive about which files it is possible to write to. It is, for example, not possible to open an arbitrary file and just write to it. This restriction has been put in place to prevent malicious actions from destroying files on your system. It is only possible to write to files that are Temporary Files, or files that have been specified by the user using `fwLocateWrite`.

Reading Files

There are two ways to open a text file for reading:

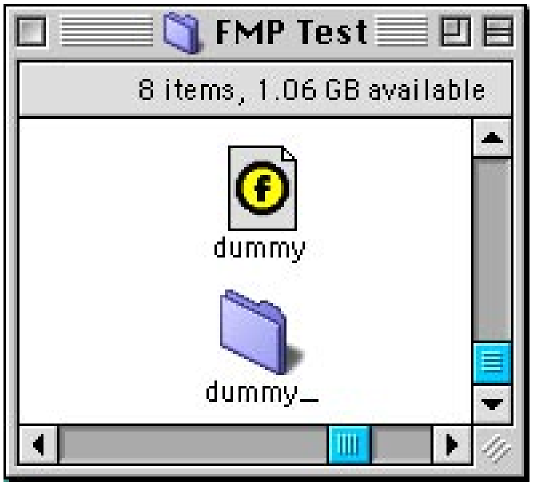
- `fwLocateWrite`- this prompts the user to specify a new file to write into.
- `fwOpenWrite`- this allows you to specify directly which file you want to write from, you can also choose to create a new one.

Temporary Files

The JavaScript API facilitates the use of temporary files. They are files that are generated by Actions, for the use of actions and are intended to be local to your document. If you want to create a file within an action that has some information that you want to use on later pages in the publishing process, as temporary storage, or to make available for upload you will need to use temporary files.

Location of the Files

Temporary Files live in a folder next to the Freeway Document. The folder has a name that is the same as the name of the document but with an underscore appended onto the end. This folder will be created the first time that write to a temporary file.



You should note that if the document that you write into has never been saved then Freeway will create a folder next to your copy of Freeway and place the temporary file in there. Your action will then loose access to this file once the document is saved (and gets a proper filename).

Writing to a Temporary File

The `fwOpenWrite` method will create a temporary file of the name that you pass as a parameter.

This action creates a file "Test File".

```
<item-action name="Test">
<action-button name="Make File" onclick = "doit()"/>
<action-javascript>

  function doit()
  {
    myFile = new FWFile;

    myFile.fwOpenWrite('Test File',false,'TEXT');
    myFile.fwWrite('test');
    myFile.fwClose();
  }

</action-javascript>
</item-action>
```

Reading from a Temporary File

If you pass a file name to the `fwOpenRead` method it open a temporary file of the name that you pass to it.

```
<item-action name="Test">
<action-button name="read" onclick = "Test()"/>
<action-javascript>

  function Test()
```

```
    {
      // create a new file object

      myFile = new FWFile;

      // open a file
      if (myFile.fwOpenRead('Test File'))
      {
        // read a line
        alert(myFile.fwReadln());

        // close the file
        myFile.fwClose();
      }
    }

  </action-javascript>
  </item-action>
```

Finding a Temporary File

The global method `fwFindFile` will allow you to test for the existence of a particular temporary file. The following has a "Find It" button that when pressed will look for a file whose name appears as a parameter.

```
  <page-action name="Find File">
  <action-text name ="File Name" default="test">
  <action-button name ="Find It" onClick="FindIt()">
  <action-javascript>

    function FindIt()
    {
      var fileName = fwParameters["File Name"];
      if (fwFindFile(fileName))
        alert("Temporary file '"+fileName+"' exists.");
      else
        alert("Temporary file does not '"+fileName+"' exist.");
    }

  </action-javascript>
  </page-action>
```

Searching for Temporary Files

The global method `fwFindAllFiles` will allow search for a number of temporary files. It will return all the temporary files that the document can access. There are optional parameters that will allow you to specify a file-type or creator and so you can do things like list all the temporary files of a give type.

The following action will list all the temporary files. If you set a type it will list al of the files of that type.

```
  <page-action name="Find All Files">
  <action-text name ="Type" default="TEXT">
  <action-button name ="Find All" onClick="FindThem()">
  <action-javascript>

    function FindThem()
    {
      var files;
      var fileName = fwParameters["File Name"];

      // find the files
      if (fwParameters["Type"]== "")
        files = fwFindAllFiles();
      else
        files = fwFindAllFiles(fwParameters["Type"]);

      alert("Temporary files are: "+files.join(", "));
    }

  </action-javascript>
  </page-action>
```

Deleting Temporary Files

The API contains a global method `fwDeleteFile` that will allow you to delete a temporary file.

The following action will delete a temporary file of a given name.

```
  <page-action name="Delete File">
  <action-text name ="File Name" default="test">
  <action-button name ="Create It" onClick="CreateIt()">
  <action-button name ="Delete It" onClick="DeleteIt()">
  <action-javascript>

    function DeleteIt()
    {
      var files;
      var fileName = fwParameters["File Name"];

      // find the files
      if (fwFindFile(fileName))
      {
        if (confirm("Do you want to delete '",fwParameters["File Name"],"'?"))
          fwDeleteFile(fwParameters["File Name"]);
      }
      else
        alert("Could not find the file '",fwParameters["File Name"],"'.");
    }

    function CreateIt()
    {
      var fileName = fwParameters["File Name"];
      myFile = new FWFile;

      myFile.fwOpenWrite(fileName,false,'TEXT');
      myFile.fwWrite('foobar');
      myFile.fwClose();
    }

  </action-javascript>
  </page-action>
```

File Parameters

In Freeway Actions you can specify File Parameters using the`<action-file>` tag. File Parameters allow the user to select files in the Actions UI. These files are generally destined to be uploaded but that does not have to be the case.

The simplest way of specifying a File parameter is in the form:

```
<action-file name="File">
</action-file>
```

If you wish to restrict the files that can be chosen in the Actions interface you can do so. Just add a list of types. The following definition will restrict the user to importing GIF and JPeg files.

```
<action-file name="File">
  <value type="GIFf"><value type="JPEG">
</action-file>
```

File Parameters in JavaScript

It is Important to recognise that file parameters in JavaScript have essentially two representations.

a) **Files that are chosen**. This is the file as it has been chosen by the user. These are available at any time.
b) **Files that are Uploaded**. This is the file path as it will be uploaded. the name may bear no similarity to the one chosen by the user, but it probably will do.

File Parameters are represented by the class `FWFileParameter` in JavaScript. For a given action you reference them in the normal way. So for our earlier example you would get this object as

```
fwParameters.File
```

or

```
fwParameters["File"]
```

The class supports the following properties that you may find useful

- `fwParameters["File"].fwHasFile` - has a file been selected for this parameter.
- `fwParameters["File"].fwFullFileName`- the full name of the file (including the path).
- `fwParameters["File"].fwFileName`- the (short) name of the file (not including the path).

You should note that `fwFullFileName` and `fwFileName` return the name of the source file. That file that has been chosen by the user not the file that will be uploaded.

Uploading

A file that is selected by the user will not be uploaded unless something happens that forces it to be uploaded. Files are uploaded when somebody interrogates their upload path. In Classic Actions Language this happens whenever someone includes a reference to the parameter. So when Freeway encounters a statement like:

```
&File;
```

It will have to find an upload path for the file (in order to substitute it into the mark-up) and the file will be uploaded.

In JavaScript the upload path is discovered when you call the `toString` method of the parameter. So you can upload a file with the following code:

```
// upload the "File" parameter
fwParameters["File"].toString();j
```

You should remember that the `toString` method can be called implicitly. For example code such as:

```
alert(fwParameters["File"].toString());
```

will have to coerce the `FWFileParameter` to a string. To do this it will call the `toString` method of the parameter. Also code like:

```
var myString = "test"+fwParameters["File"]);
```

Will result in the parameter being automatically coerced to a string.

Location of Uploaded Files

Files specified in the form

```
<action-file name="File">
```

Will be uploaded as were they any normal freeway resource. So, depending on the settings that you have for your Freeway document, they will be uploaded into a "Resources Folder". Freeway will pull it's normal "trick" of looking for any file that has the same contents as the file you have specified so were you to have two files that are the same then Freeway will include just the one and have common references to it.

If you are dealing with Java Applets this is not desirable as you can not guarantee the location of the file. Additionally almost all JavaApplets require the applet to be in the same location as the HTML file. Freeway supports this by having an additional attribute `keepwithhtml`. Including this will ensure that a copy of the file will be uploaded to the same location as the HTML file on which the action lives.

```
<action-file name="File" keepwithhtml>
```

File Parameters and File Objects

It is possible to use File Parameters with file objects. This means that you can get your user to specify files in the Actions interface and then to read those files. It also means that it is possible to set files that are parameters in the File Parameters interface. Setting file parameters means that you can create files and arrange for them to be uploaded.

Reading files Specified by File Parameters

The method `FWFile` method `fwOpenRead` can be passed a parameter of type `FWFileParameter` and it will open that file in the normal way. The following action demonstrates this:

```
<page-action name="Read File">
<action-file name ="File">
  <value type="TEXT">
</action-file>
<action-button name ="Read It" onClick="ReadIt()">
```

```
<action-javascript>

  function fwInterface()
  {
    // enable the read button only if you have selected a file
    fwParameters["Read It"].fwEnable
      = fwParameters["File"].fwHasFile;
  }

  function ReadIt()
  {
    myFile = new FWFile;

    // open the file specified in the parameter "File"
    if (myFile.fwOpenRead(fwParameters["File"]))
    {
      // read a line
      alert(myFile.fwReadln());
      myFile.fwClose();
    }
  }

</action-javascript>
</page-action>
```

Setting a File Parameter

The `FWFileParameter` has a method `fwSpecify` that allows you to specify a file that it is to use. This is how you set this parameter. You will want to use this if you want to generate a file that you want to be uploaded.

The following example will create a text file with the contents "foobar" and uploaded it to the same location as the graphics and other resources:

```
<page-action name="Upload A File">
<action-file name ="File"/>
<action-javascript>

  function fwBeforeEndHead()
  {
    myFile = new FWFile;

    // create the file
    if (myFile.fwOpenWrite("foobar file",true,'TEXT'))
    {
      myFile.fwWrite('foobar');
      myFile.fwClose();

      // set the file parameter
      fwParameters["File"].fwSpecify(myFile);

      // upload it
      fwParameters["File"].toString();
    }
  }

</action-javascript>
</page-action>
```

If you do not want the parameter to be visible you can do so my baking it a "var" parameter

```
<action-file name ="File" var/>
```

Or by setting the `fwVisible` attribute to false in a `fwInterface` function.