Lasso Fusebox 3.0 Specification

Original by Hal Helms
Adapted for Lasso by Rich Tretola

December 12, 2002

Fusebox

Introduction

Fusebox is a development methodology for web applications. Its name derives from its basic premise: a well-designed application should be similar to the fusebox in a house. In a fusebox, if one fuse is blown, the rest of the fuses continue to work. Each fuse has its own defined job, and Fuse A does its job without any help from Fuses B, C, or D. Similarly, in a Fusebox application, if one section of the application "breaks", the rest of the application should continue to work.

Fuseactions

In a Fusebox application, every action that an application must handle is referred to as a Fuseaction. For example, an e-commerce application would include the following possible Fuseactions:

- viewCart
- viewCatalog
- addItem
- checkOut

Fuses

To perform each Fuseaction, a Fusebox application uses a series of lasso pages. Each page is referred to as a Fuse. To perform the Fuseaction 'viewCart' mentioned above, the user's cart must first be queried and then the results must be displayed. Therefore, the Fuses called for the Fuseaction viewCart would be:

- qry_getCart.lasso
- dsp_Cart.lasso

This brings us to a discussion of the possible types of fuses.

Types of fuses

Fuses can be grouped into a finite number of types. The four most common are:

- Display fuses, prefixed with dsp_, which are used to show information to the user.
- Select Query fuses, prefixed with qry_, which contain SELECT, INSERT, UPDATE, DELETE or queries.
- Action fuses, prefixed with act_, which contain any other data manipulation.
- Location fuses, prefixed with url_, which redirect the application to a new URL.
- Form fuses, prefixed with frm_, which contain forms. (frm_ is not an officially recognised fuse prefix but it is a way to further distinguish your display files.)

Back to the Fusebox analogy

Fuseactions are defined, with their Fuses, in one file, fbx_Switch.lasso. A sample fbx_Switch file is given below. At the end of each Fuseaction, the application is sent back to this file with the next Fuseaction.

Sample fbx_Switch file

We have used many new terms, which can be confusing. Before looking at the files used in a Fusebox application, here is a short sample of code from a fbx_Switch file, showing Fuseactions and Fuses in use. After reviewing, you should have a better idea of what Fuseactions and Fuses are.

```
[Encode_Set: -EncodeNone]
[Select:$fusebox->(Find:'fuseaction')]
        [Case: showInputForm]
                [Var: 'xfa_add' = 'admin.add']
                [Var: 'xfa_edit' = 'admin.edit']
                [Var: 'xfa_delete' = 'admin.delete']
                [FBX_Include: 'frm_InputForm.lasso']
        [Case: showEmployeeList]
                [FBX_Include: ' qry_EmployeeList.lasso']
                [FBX_Include: ' dsp_EmployeeList.lasso']
        [Case]
                <p>This is the default case tag. I received a fuseaction called
                "[$fuseaction]" and I don't know what to do with it.</p>
[/Select]
[/Encode_Set]
```

Explanation of sample fbx_Switch file

In the above file, the Fuseactions are "showInputForm" and "showEmployeeList". (Don't worry about the [Var: 'xfa_add' = 'admin.add'] statements in the "showInputForm" fuseaction, they will be explained later.) The **include** statements are the Fuses for the Fuseactions. The Fuseaction that is required can be passed to the application through a URL variable.

Core Files

Fusebox 3 contains several core files, prefixed with FBX_. These include:

- FBX_Fusebox30_LP6.lasso: Non editable file which contains the main Fusebox code.
- FBX_Circuits.lasso: This file provides circuit-to-directory mappings.
- FBX_Layouts.lasso: This file determines the layout file to be used for a particular circuit.
- FBX_Switch.lasso: This file processes the fuseaction sent to the application.
- FBX_Settings.lasso: This file allows variables to be set at each circuit. Variables set by parent circuits are inherited by their descendants and may be overridden.
- FBX_Library.lasso: Adds additional functionality that is needed to run the Fusebox core file on the Lasso 6 server. It is included within the core file. This file is unique to the Lasso port of Fusebox 3.0.

A. FBX_Fusebox30_LP*6*.lasso

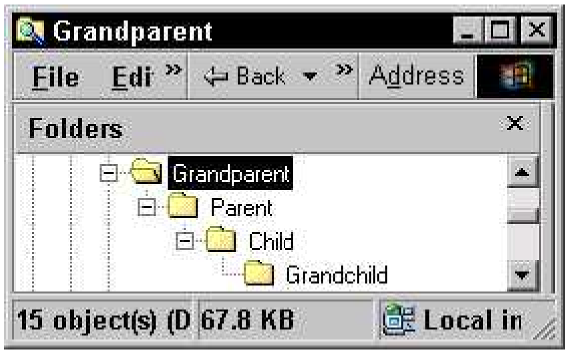
This file should be called by your home circuit's default file (index.lasso, default.lasso, etc).

```
[Include:'fbx_fusebox30_LP6.lasso']
```

B. FBX_Circuits.lasso

This file provides directory to circuits mapping. The reserved structure Fusebox.Circuits contains key/value pairs where the key is the circuit name and the value is the directory path beginning with the home circuit.

A sample FBX_Circuits.lasso file is provided for the following application:



```
[$_Circuits->(Insert:'Front'='Grandparent')]
[$_Circuits->(Insert:'Front'='Grandparent/Parent')]
[$_Circuits->(Insert:'Front'='Grandparent/Parent/Child')]
[$_Circuits->(Insert:'Front'='Grandparent/Parent/Child/Grandchild')]
```

The FBX_Fusebox30_LP6.lasso file uses the FBX_Circuits.lasso file in resolving compound fuseaction names.

Note that the circuit names do not need to be the same as the directory names.

C. FBX_Layouts.lasso

This file is responsible for setting two variables, Fusebox.layoutDir and Fusebox.layoutFile.

Fusebox.layoutDir points to a directory (if it exists) in which layout files are kept. Fusebox.layoutFile points to the file to be used for layout. This file can be safely omitted, in which case, no layout will be applied. If you create a layout file, it must, at minimum, dereference the variable, Fusebox.layout.

Here is an example layout file that applies a copyright at the bottom of the page which is part of the dsp_footer.lasso file:

```
[FBX_Include: 'dsp_header.lasso']

[Output:$fusebox->(Find:'layout'),-EncodeNone]

[FBX_Include: 'dsp_footer.lasso']
```

D. FBX_Switch.lasso

The fbx_Switch file is probably the simplest file of the core files. It decides what files to include on the page, based on the Fuseaction.

Here is a sample of an FBX_Switch.lasso file:

```
[Encode_Set: -EncodeNone]
[Select:$fusebox->(Find:'fuseaction')]
		[Case:'Welcome']
				[FBX_Include: 'qry_getContacts.lasso']
				[FBX_Include: 'dsp_Contacts.lasso']
		[Case]
				<p>This is the default case tag. I received a fuseaction called
"[$fuseaction]" and I don't know what to do with it.</p>
[/Select]
[/Encode_Set]
```

E. FBX_Settings.lasso

The fbx_Settings file is used by the Fusebox application to set default values. There must be one fbx_Settings file in the root folder of the application. Individual circuits can have their own fbx_Settings files if default values need to be set only for the files in that circuit, or if a child circuit needs to overwrite a default value in a parent circuit.

F: FBX_Library.lasso

The fbx_Library.lasso file adds additional functionality that is needed to run the Fusebox core file on the Lasso 6 server. It is included within the core file. This file is unique to the Lasso port of Fusebox 3.0.

Exit Fuseactions (XFAs)

Fusebox 3 contains the concept of exit fuseactions. Rather than having fuseactions hardcoded into exit points, like this...

```
<form name="contact" action="index.lasso?fuseaction=home.submit" method="post">
```

...Fusebox 3 replaces these hardcoded references with XFAs:

```
<form name="contact" action="[$self]?fuseaction=[$xfa_Submit]" method="post">
```

-([$self] will be explained later)

The value of these fuseactions is then set in the FBX_Switch file:

snippet from FBX_Switc.lasso file

```
[Encode_Set: -EncodeNone]
[Select:$fusebox->(Find:'fuseaction')]
	[Case:'Welcome']
		[Var: 'xfa_Edit' = 'home.edit']
		[Var: 'xfa_Delete' = 'home.delete']
		[Var: 'xfa_New' = 'home.new']
		[FBX_Include: 'qry_getContacts.lasso']
		[FBX_Include: 'dsp_Contacts.lasso']
	[Case]
		<p>This is the default case tag. I received a fuseaction called "[$fuseaction]" and
I don't know what to do with it.</p>
[/Select]
[/Encode_Set]
```

The references to XFAs in the fuse are resolved at run time, allowing for ease of reuse of both individual fuses and entire circuits.

Fusebox 3 API

Fusebox 3 exposes a series of variables to help in writing code:

Public API variable	Type	May be set?	Description
Fusebox.isCustomTag	BOOLEAN	NO	Is this fusebox being called as a custom tag?
Fusebox.isHomeCircuit	BOOLEAN	NO	Is the directory of the file currently being operated on by FBX_Fusebox30_LP6.lasso the same as the home circuit of the app?
Fusebox.isTargetCircuit	BOOLEAN	NO	Is the directory of the file currently being operated on by FBX_Fusebox30_LP6.lasso the same as the target circuit of the app?
Fusebox.fuseaction	STRING	NO	The simple fuseaction extracted from the attributes.fuseaction of form circuit.fuseaction passed in
Fusebox.circuit	STRING	NO	The simple circuit extracted from the attributes.fuseaction of form circuit.fuseaction passed in
Fusebox.homeCircuit	STRING	NO	The circuit alias of the home circuit of the app
Fusebox.targetCircuit	STRING	NO	The circuit alias of the target circuit of the app. Usually this is the same as fusebox.circuit unless you've made a circuits definition error
Fusebox.thisCircuit	STRING	NO	The circuit alias of the directory of the file currently being operated on by FBX_Fusebox30_LP6.lasso
Fusebox.thisLayoutFile	STRING	YES	The layout file to be applied to this circuit after the fuseaction and its fuse(s) are done creating their content
Fusebox.thisLayoutDir	STRING	YES	The relative path from the target circuit where the layout file to be applied to this circuit is located. If no special layout directory has been created, this should be set to an empty string.
Fusebox.CurrentPath	STRING	NO	The relative directory path of the file currently being operated on by FBX_Fusebox30_LP6.lasso, relative from the root of the application (ie the home circuit). Example: dir1/dir2/dir3/
Fusebox.RootPath	STRING	NO	The relative directory path of the file currently being operated on by the core frozen fusebox code, relative to the root of the application (ie the home circuit). Example: ../../../
Fusebox.layout	STRING	NO	This is the variable used by FBX_SaveContent custom tag that captures the generated content to that point in preparation for wrapping a layout file around it (as defined by fusebox.layoutfile). This variable must be inside each layout file in order for content to be passed up to the next level of nested layouts
Fusebox.SuppressErrors	BOOLEAN	YES	FBX_Fusebox30_LP6.lasso has some abilities built in to suppress native Lasso errors and instead give its best guess at what it wrong with your application (as it relates to Fusebox). Default value is always FALSE, which therefore generates native Lasso errors. You may want to set it to TRUE while you set up your application in Fusebox style and let it default to FALSE when you're confident that your Fusebox is set up correctly but are testing for coding errors in your fuses unrelated to Fusebox.

Directory to Circuit Mapping

Directory to Circuit mapping is provided through FBX_Circuits.lasso. The reserved structure Fusebox.Circuits contains key/value pairs where the key is the circuit name and the value is the directory path beginning with the home circuit.

The FBX_Fusebox30_LP6.lasso file uses FBX_Circuiuts.lasso file in resolving compound fuseaction names.

Note that the circuit names do not need to be the same as the directory names.

Naming Fuseactions

In Fusebox 3, an exit fuseaction for the fuse is defined in FBX_Switch.lasso. The value of that XFA is a compound fuseaction, consisting of the circuit name, a dot separator, and the circuit request.

snippet from FBX_Switch.lasso
```
[Case:'xxx']
        [Var: ' XFA_buyItem ' = ' ShoppingCart.addItem ']
        [FBX_Include: 'xxx.lasso']
```

snippet from xxx.lasso
```
<a href ="[$self]?fuseaction=[$xfa_buyItem]">
```

The circuit name refers to the directory alias defined in FBX_Circuits.lasso.

NOTE: The variable, self, is not part of the Fusebox 3 specification, but is used by many developers to refer to the home circuit's default page and is set in the FBX_Settings.lasso file.

Fusedoc

Fusedoc is a documentation system/program definition language for documenting fuses. The Fusedoc uses XML syntax wrapped in a Lasso comment and is normally place on top of the fuse file itself. Here is a sample Fusedoc for a display login file.

```
<fusedoc fuse="frm_login.lasso" language="Lasso" version="3.0">
   <responsibilities>I am a form that lets users log in.</responsibilities>
   <properties>
      <history author="Rich Tretola" date="12/10/2002" role="Architect" type="Create" />
   </properties>
   <io>
      <in>
         <string name="self" />
         <string name="XFA_submitForm" />
      </in>
      <out>
         <string name="fuseaction" scope="formOrUrl" />
         <string name="userName" scope="formOrUrl" />
         <string name="password" scope="formOrUrl" comments="password field" />
      </out>
   </io>
</fusedoc>
```

The Fusedoc XML root element has three sub-elements: *responsibilities*, *properties*, and *io*. Of these, the only one required is responsibilities.

Responsibilities use the first person to have the fuse describe what it is responsible for. Properties is a more generic, catch-all section that has three possible sub-elements: history, property, and note. IO (short for input/output) contains in and out sub-elements.

Fusedoc: responsibilities

Each Fusedoc will have a plain english explanation as to what the Fuse is meant to do. This tag does not have any attributes, the english explanation goes in between the opening and closing tags

Fusedoc: properties

1. Fusedoc: properties: history

   The history element has the following attributes:
   - **author**: free text
   - **date**: free text
   - **email**: free text
   - **role**: free text
   - **type**: create | update

   The history tag can be used as a tagset with free text between the tags:

   ```
   <history author="Rich Tretola">
   This is just a sample valid history element.
   </history>
   ```

2. Fusedoc: properties: property

   The property element has the following attributes:
   - **name**: free text
   - **value**: free text

   The property tag is an empty tag set:

3. Fusedoc: properties: note

   The note element has the following attributes:
   - **author**: free text
   - **date**: free text

   The note tag can be used as a tagset with free text between the tags:

   ```
   <note author="Rich Tretola">
   This is just a sample valid note element.
   </note>
   ```

Fusedoc: io

4. Fusedoc: io: in/out

Both in and out elements have no attributes. Both elements accept the following sub-elements:

- string
- number
- boolean
- list
- structure
- array
- recordset
- cookie
- datetime
- file

a) *Fusedoc: io: in/out: string*

The string element has the following attributes:

- **name**: free text
- **scope**: application | attributes | caller | cgi | client | form | formorurl | request | server | session | url | **variables**
- **comments**: free text
- **default**: free text
- **mask**: free text
- **oncondition**: free text
- **optional**: true | **false**

b) *Fusedoc: io: in/out: number*

The number element has the following attributes:

- **name**: free text
- **scope**: application | attributes | caller | cgi | client | form | formorurl | request | server | session | url | **variables**
- **comments**: free text
- **default**: free text
- **precision**: integer | decimal
- **oncondition**: free text
- **optional**: true | **false**

c) *Fusedoc: io: in/out: boolean*

The boolean element has the following attributes:

- **name**: free text
- **scope**: application | attributes | caller | cgi | client | form | formorurl | request | server | session | url | **variables**
- **comments**: free text
- **default**: free text
- **oncondition**: free text
- **optional**: true | **false**

d) *Fusedoc: io: in/out: list*

The list element has the following attributes:

- **name**: free text

- **scope**: application | attributes | caller | cgi | client | form | formorurl | request | server | session | url | **variables**
- **delims**: **comma** | free text
- **comments**: free text
- **oncondition**: free text
- **optional**: true | **false**
- **default**: free text

e) *Fusedoc: io: in/out: structure*

The structure element has the following attributes:

- **name**: free text
- **scope**: application | attributes | caller | cgi | client | form | formorurl | request | server | session | url | **variables**
- **comments**: free text
- **oncondition**: free text
- **optional**: true | **false**

The structure element contains one or more of the following sub-elements:

- string
- number
- boolean
- datetime
- array
- structure
- recordset
- list

f) *Fusedoc: io: in/out: array*

The array element has the following attributes:

- **name**: free text
- **scope**: application | attributes | caller | cgi | client | form | formorurl | request | server | session | url | **variables**
- **comments**: free text
- **oncondition**: free text
- **optional**: true | **false**

The array element contains one or more of the following sub-elements:

- string
- number
- boolean
- datetime
- array
- structure
- recordset
- list

g) *Fusedoc: io: in/out: recordset*

The recordset element has the following attributes:

- **name**: free text
- **scope**: application | attributes | caller | cgi | client | form | formorurl | request | server | session | url | **variables**
- **primarykeys**: free text
- **mask**: free text |
- **comments**: free text
- **oncondition**: free text
- **optional**: true | **false**

h) *Fusedoc: io: in/out: cookie*

The cookie element has the following attributes:

- **name**: free text
- **domain**: free text
- **expires**: free text
- **path**: free text
- **secure**: true | **false**
- **value**: free text
- **comments**: free text
- **oncondition**: free text
- **optional**: true | **false**

i) *Fusedoc: io: in/out: datetime*

The datetime element has the following attributes:

- **name**: free text
- **scope**: application | attributes | caller | cgi | client | form | formorurl | request | server | session | url | **variables**
- **mask**: free text |
- **comments**: free text
- **default**: free text
- **oncondition**: free text
- **optional**: true | **false**

j) *Fusedoc: io: in/out: file*

The file element has the following attributes:

- **path**: free text
- **action**: read | write | append | overwrite | delete | **exists** | module | include
- **comments**: free text
- **oncondition**: free text
- **optional**: true | **false**