

HBX FlashTrack  
A Guide to Flash ActionScript Based Implementations

January 4th, 2006

Mike Anderson  
Manager, Enterprise Technology Group

Professional Services Guide



<u>INTRODUCTION.....</u>	<u>3</u>
<u>LIBRARY FUNCTIONS: HBXFLASHTRACK2 / HBXFLASHTRACK3.....</u>	<u>5</u>
<u>HOW IT WORKS: HBXFLASHTRACK2.....</u>	<u>8</u>
<u>HOW IT WORKS - HBXFLASHTRACK3.....</u>	<u>9</u>
<u>EXAMPLES: HBXFLASHTRACK2.....</u>	<u>11</u>
<u>EXAMPLES: HBXFLASHTRACK3.....</u>	<u>14</u>
<u>APPENDIX A - HBXFLASHTRACK2.AS.....</u>	<u>17</u>
<u>APPENDIX B - HBXFLASHTRACK3.AS.....</u>	<u>19</u>
<u>APPENDIX C - PERSISTED REQUEST DATA.....</u>	<u>22</u>
<u>APPENDIX D - GATEWAY VARIABLES.....</u>	<u>23</u>

## Introduction

ActionScript is an ECMAScript-based programming language used for scripting Adobe Flash movies and applications. Since both ActionScript and JavaScript are based on the same ECMAScript syntax, fluency in one theoretically translates easily to the other. However, while JavaScript's DOM is browser window-, document- and form-centric, the ActionScript DOM is movie-centric, which may include animations, audio, text and event handling.

ActionScript first appeared in its current syntax with the release of Flash 5, which was the first thoroughly programmable version of Flash. This ActionScript release was named ActionScript 1.0. Flash 6 (MX) then further broadened the utility of the programming environment by adding a number of built-in functions and allowing better programmatic control of movie elements. Flash 7 (MX 2004) introduced ActionScript 2.0, which added strong typing and class-based programming features such as explicit class declarations, inheritance, interfaces, and Strict Data Typing. ActionScript 1.0, 2.0 and 3.0 share the same compiled form within Flash SWFs.

With the increase in RIA (Rich Internet Applications), Flash has evolved as one of the most popular and preferred delivery platforms. As analysis of Web Analytic behavior continues to thrive, utilization of the internal functionality made available in a Flash SWF file via ActionScript can provide a very convenient and logical solution for tracking this analytic behavior.

Traditional tagged based web analytics solutions rely heavily on the functionality provided by the embedded browser language, Javascript. In the past web behavior within a Flash SWF has been tracked by dispatching requests to the Javascript from within the Flash object. This approach requires the availability of the Javascript functionality in the containing HTML page, but also requires dual knowledge of ActionScript as well as Javascript and how the two communicate.

With the maturity of ActionScript, the capability to mimic the behaviors typically only provided from within Javascript has dramatically increased. As a result a standalone HBX ActionScript ("FlashTrack") library has been created to facilitate the flexibility of the HBX Implementation engine from within the FlashTrack library. This enables flash developers to stay within their realm without depending on knowledge or functionality of the containing HTML page. This also allows for Flash objects to be distributed on remote sites without the need to have the standard HBX Javascript distribution.

Included in this document is support for ActionScript 1.0-2.0 (hbxFlashTrack2) based Flash objects and ActionScript 3.0 (hbxFlashTrack3) based Flash objects. The hbxFlashTrack2 methodology utilizes the hbxFlashTrack2.as library provided in

Appendix A and is utilized via a #include directive. The hbxFlashTrack3 methodology utilizes the hbxFlashTrack3.as class provided in Appendix B and is utilized via an import directive.

## Library Functions: hbxFashTrack2 / hbxFashTrack3

- **\_hbPageView / object.hbPageView**  
A utility function for tracking a pageview request. This function takes in two parameters: the pagename value and the content category value. The default value for pagename is "flash default" and the default content category is "/". Once the first pageview call or set call is made the default pagename and content category values are overwritten.  
*(this function includes an automatic \_hbSend call.)*
- **\_hbLink / object.hbLink**  
A utility function for tracking a link click request. This function takes in two parameters: the linkId value and the linkPosition value. The linkPosition value is optional.  
*(this function includes an automatic \_hbSend call.)*
- **\_hbDownload / object.hbDownload**  
A utility function for tracking a download request. This function takes in one parameter: the download file name.  
*(this function includes an automatic \_hbSend call.)*
- **\_hbExitLink / object.hbExitLink**  
A utility function for tracking an exitlink request. This function takes in one parameter: the exitlink URL.  
*(this function includes an automatic \_hbSend call.)*
- **\_hbVisitorSeg / object.hbVisitorSeg**  
A utility function for tracking a population group. This function takes in one parameter: the population group ID.  
*(this function includes an automatic \_hbSend call.)*
- **\_hbCampaign / object.hbCampaign**  
A utility function for tracking a campaign response. This function takes in one parameter: the campaign ID.  
*(this function includes an automatic \_hbSend call.)*
- **\_hbFunnel / object.hbFunnel**  
A utility function for tracking a funnel level. This function takes in one parameter: the funnel data in the format "(ID,LEVEL[c])". The parenthesis are required and the "c" value in brackets is optional and is only set on the conversion level.  
*(this function includes an automatic \_hbSend call.)*

- **\_hbGoalPage / object.hbGoalPage**  
A utility function for tracking a campaign conversion. This function takes in one parameter: the campaign ID, or the literal "LAST".  
*(this function includes an automatic \_hbSend call.)*
- **\_hbForm / object.hbForm**  
A utility function for tracking a form completion or a form abandonment. For a form completion the function takes in one parameter: the value 1. For a form abandonment the function takes in two parameters: the value 0 and the last field focused prior to abandonment.  
*(this function includes an automatic \_hbSend call.)*
- **\_hbMediaEvent / object.hbMediaEvent**  
A utility function for tracking a streaming media event. This function takes in six arguments: the media name, the current position of the video in milliseconds, the end position of the video in milliseconds, the play state (play, pause, stop or playp\*), the video client, the video client version.  
\* playp is the play progress state. This is sent at 120 second intervals of continuous play to keep a session from timing out.  
*(this function includes an automatic \_hbSend call.)*
- **\_hbSet / object.hbSet**  
A utility function for setting any gateway variable. This function is used in conjunction with \_hbSend to transmit the data. You can set many variables before sending them. This function takes in two parameters: the gateway variable and the corresponding value. For a list of available gateway variables see the HBX help documentation under the topic "gateway variables".  
*(this set function does not include an automatic \_hbSend call.)*
- **\_hbSetAccount / object.hbSetAccount**  
A utility function for setting the HBX account value. This function takes in one parameter: the HBX account.  
*(this set function does not include an automatic \_hbSend call.)*
- **\_hbSetCustomerId / object.hbSetCustomerId**  
A utility function for setting a customer ID value. This function takes in one parameter: the customer ID.  
*(this set function does not include an automatic \_hbSend call.)*
- **\_hbSetGateway / object.hbSetGateway**  
A utility function for setting the HBX gateway value. This function takes in one parameter: the HBX gateway.  
*(this set function does not include an automatic \_hbSend call.)*
- **\_hbSetMulti / object.hbSetMulti**

A utility function for setting multiple variables at once. This function takes in one parameter: an object value that contains object attributes. Each value is set in the relationship attribute -> value.

*(this set function does not include an automatic \_hbSend call.)*

- **\_hbSend / object.hbSend**

A utility function for sending data set through the \_hbSet functions. This function takes no parameters.

- **\_hbGetJSVal / object.hbGetJSVal**

A utility function for retrieving elements from the browser DOM.

## How It Works: hbxFashTrack2

To enable hbxFashTrack2, you first need to be using a Flash version that supports ActionScript 2.0. To add the functionality include the following line at the beginning of your flash movie:

```
#include "hbxFashTrack2.as"
```

This assumes the hbxFashTrack2.as file is in the same directory as the flash movie when the movie is compiled.

Edit the hbxFashTrack2.as file to set the appropriate HBX account and gateway values:

```
hbz.acct = "<!-- HBX ACCOUNT -->";  
hbz.gn = "<!-- HBX GATEWAY -->";
```

If you are unable to modify this file, or wish to change accounts before sending a request, you can use the `_hbSetAccount` or `_hbSetGateway` utility functions.

You can use any of the functions listed in the function section to track analytic behavior. The hbxFashTrack2 library will poll the HTML webpage through an external interface call to determine the following information:

- `document.referrer`
- `navigator.javaEnabled`
- `navigator.userAgent`
- `screen.width`
- `screen.height`
- `screen.colorDepth` or `screen.pixelDepth` depending on browser
- `document.URL`

These external interface calls do not require any supporting javascript functions, and are only called one time when the hbxFashTrack2 library is loaded.

To enable the external interface communication, the following parameter must be set within the flash object tag.

```
<param name="allowScriptAccess" value="always" />
```

The hbxFashTrack2 library has two sets of internal collection stores to handle request information. The first is `_hbBaseRequest` which is a LoadVars object and stores persistent data related to the HBX analytics request. This stores data such as the account, pagename, content category, customerID, system demographics (screen size, color, java status), referrer, pageUrl, and hbxFashTrack library version. The second internal collection, `_hbRequestData`, is also a LoadVars object and contains all data set through the `_hbSet` methodology or the convenience functions. This secondary store is re-initialized once the request is sent.

## How It Works - hbxFashTrack3

To enable hbxFashTrack3, you first need to be using a Flash version that supports ActionScript 3.0. To add the hbxFashTrack3 functionality, include the following line at the beginning of your flash movie:

```
import hbxFashTrack3;
var hbx:hbxFashTrack3 = new hbxFashTrack3();
```

This assumes the hbxFashTrack3.as file is in the same directory as the flash movie when the movie is compiled. The hbxFashTrack3 object will be created with the namespace "hbxF" in this sample.

Edit the hbxFashTrack3.as file to set the appropriate HBX account and gateway values:

```
private var acct = "<!-- INSERT HBX ACCOUNT -->";
private var gn = "<!-- INSERT HBX GATEWAY -->";
```

If you are unable to modify this file, or wish to change accounts before sending a request, you can use the hbxF.hbSetAccount or hbxF.hbSetGateway utility functions.

You can use any of the functions listed in the function section to track behavior. The hbxFashTrack3 library will poll the HTML webpage through an external interface call to determine the following information:

- document.referrer
- navigator.javaEnabled
- navigator.userAgent
- screen.width
- screen.height
- screen.colorDepth or screen.pixelDepth depending on browser
- document.URL

These external interface calls do not require any supporting javascript functions, and are only called one time when the hbxFashTrack3 library is loaded.

To enable the external interface communication, the following parameter must be set within the flash object tag.

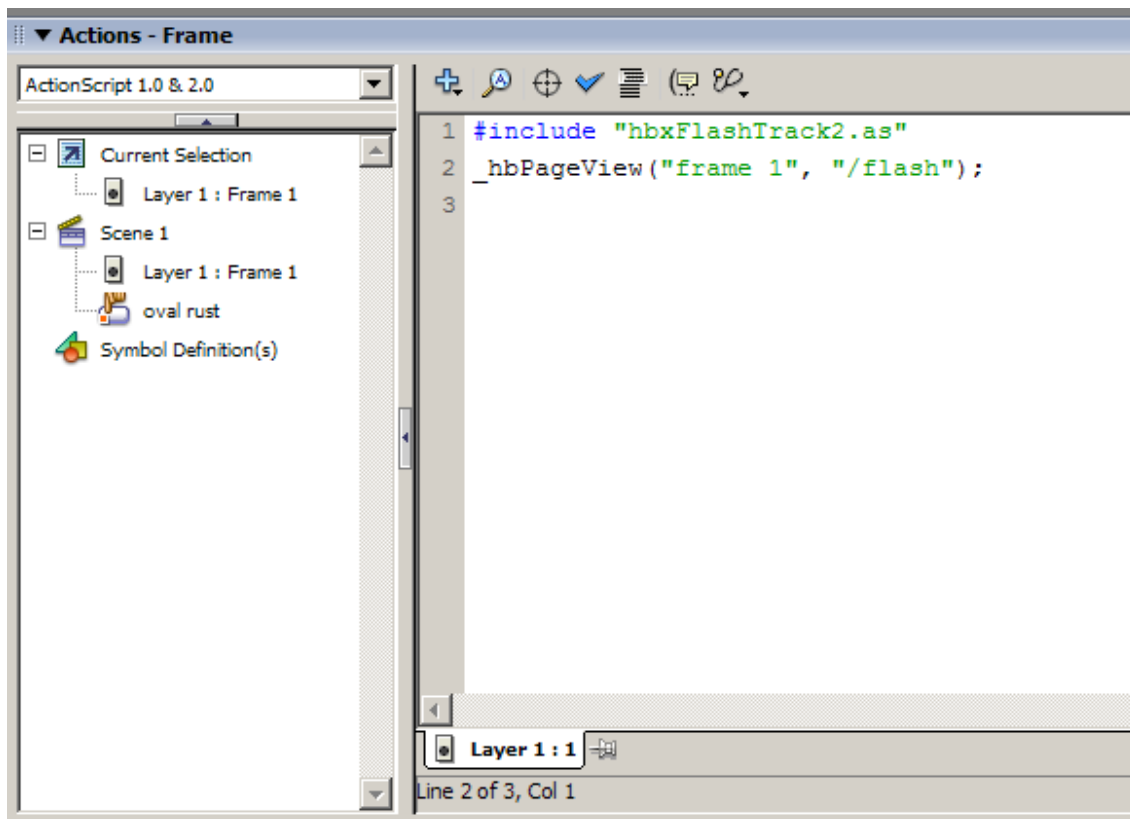
```
<param name="allowScriptAccess" value="always" />
```

The hbxFashTrack3 library has two sets of internal collection stores to handle request information. The first is hbxF.req which is a LoadVars object and stores persistent data related to the HBX analytics request. This stores data such as the account, pagename, content category, customerID, system demographics (screen size, color, java status), referrer, pageUrl, and hbxFashTrack library version. The second internal collection, hbxF.reqData, is also a LoadVars object and contains all data set through the hbxF.hbSet

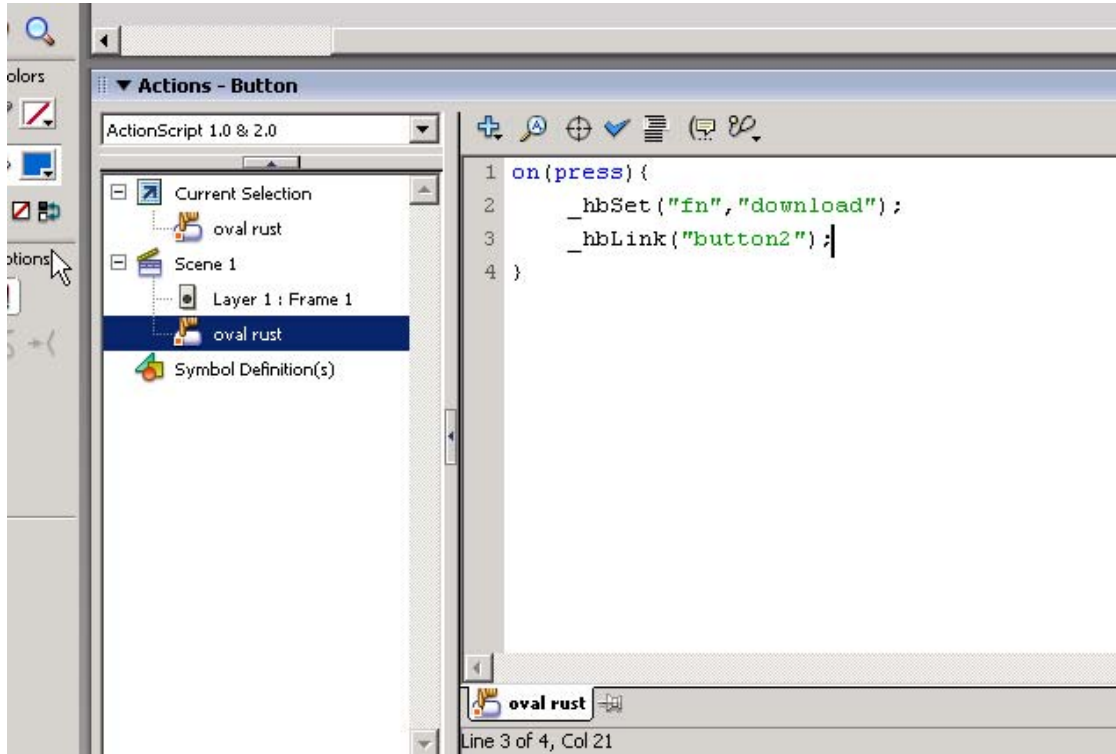
methodology or the convenience functions. This secondary store is re-initialized once the request is sent.

## Examples: hbxFlashTrack2

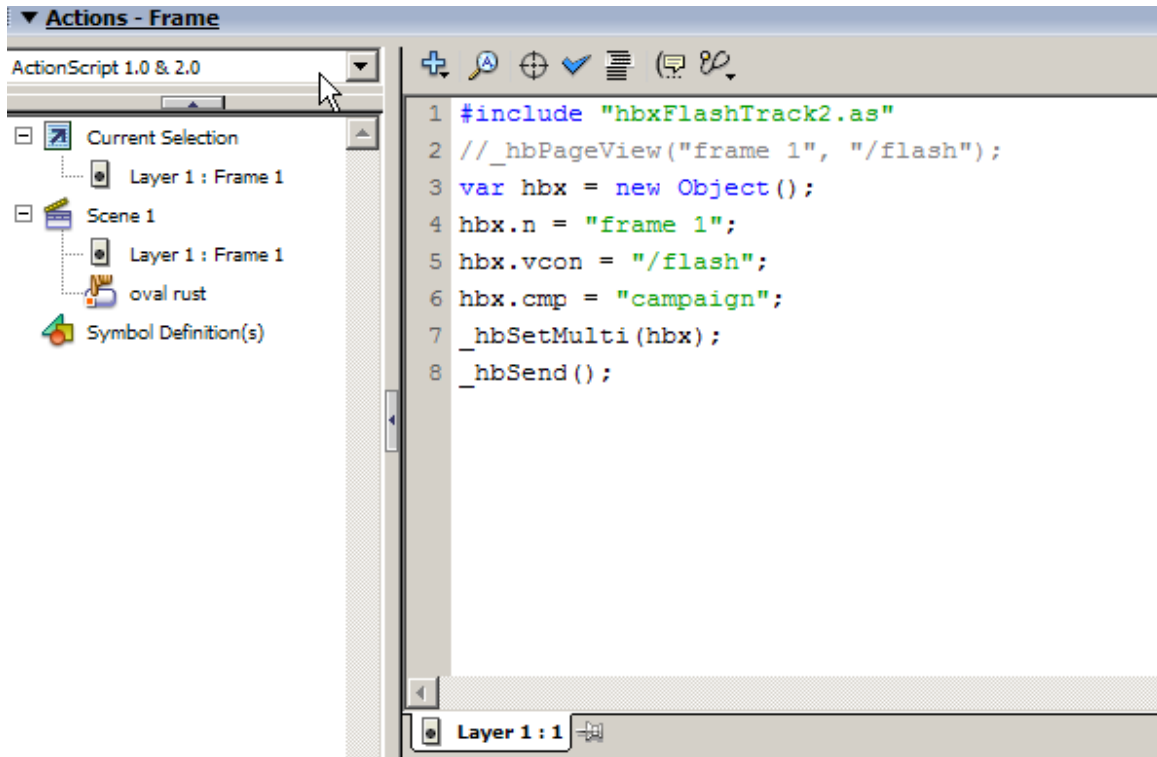
The following example shows how to track a pageview request in the main scene frame using the hbxFlashTrack2 methodology.



The following example shows how to track a download file with a link click request using the hbxFlashTrack2 methodology.



The following example shows how to use the `_hbSetMulti` function. This example sets the page name "n", content category "mlc" and a campaign response "cmp" using the `hbxFashTrack2` methodology.



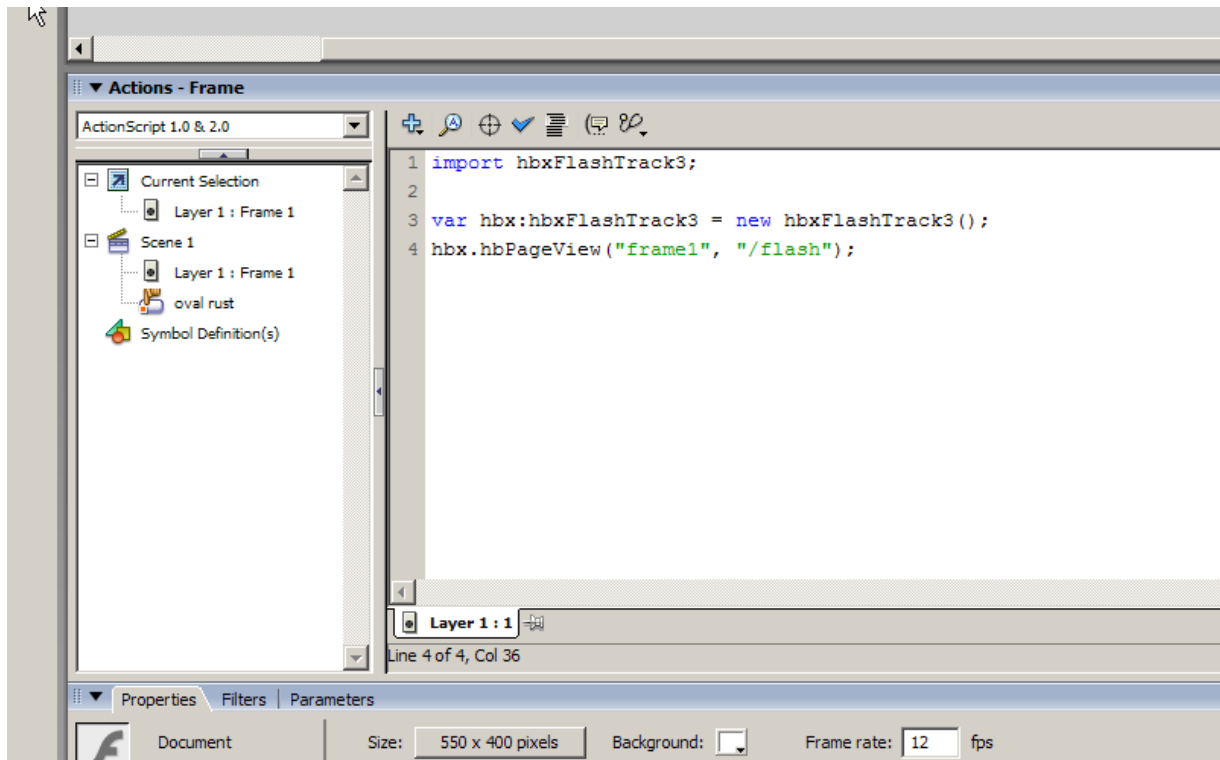
The screenshot shows an IDE window titled "Actions - Frame" for "ActionScript 1.0 & 2.0". The left sidebar displays a hierarchy: "Current Selection" (selected), "Layer 1 : Frame 1", "Scene 1", "Layer 1 : Frame 1", "oval rust", and "Symbol Definition(s)". The main editor area contains the following code:

```
1 #include "hbxFashTrack2.as"
2 // _hbPageView("frame 1", "/flash");
3 var hbx = new Object();
4 hbx.n = "frame 1";
5 hbx.vcon = "/flash";
6 hbx.cmp = "campaign";
7 _hbSetMulti(hbx);
8 _hbSend();
```

The status bar at the bottom indicates "Layer 1 : 1".

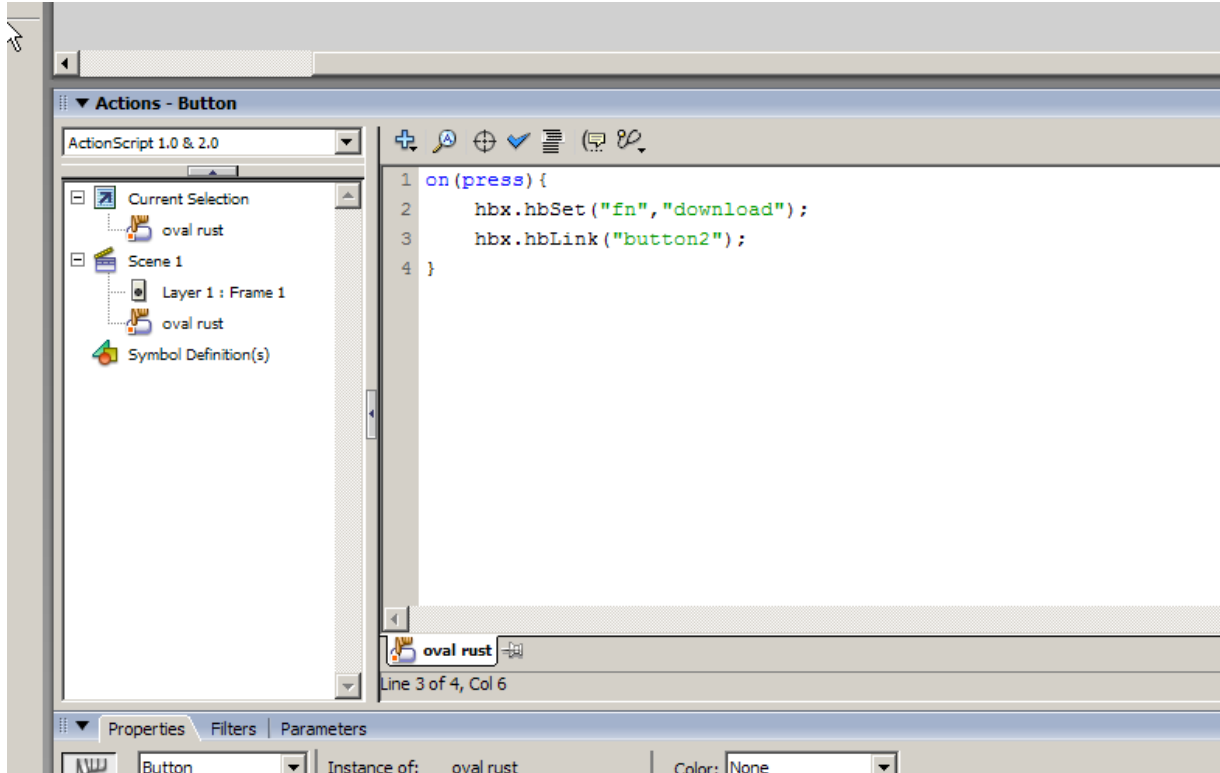
## Examples: hbxFashTrack3

The following example shows how to track a pageview request in the main scene frame using the hbxFashTrack3 methodology.

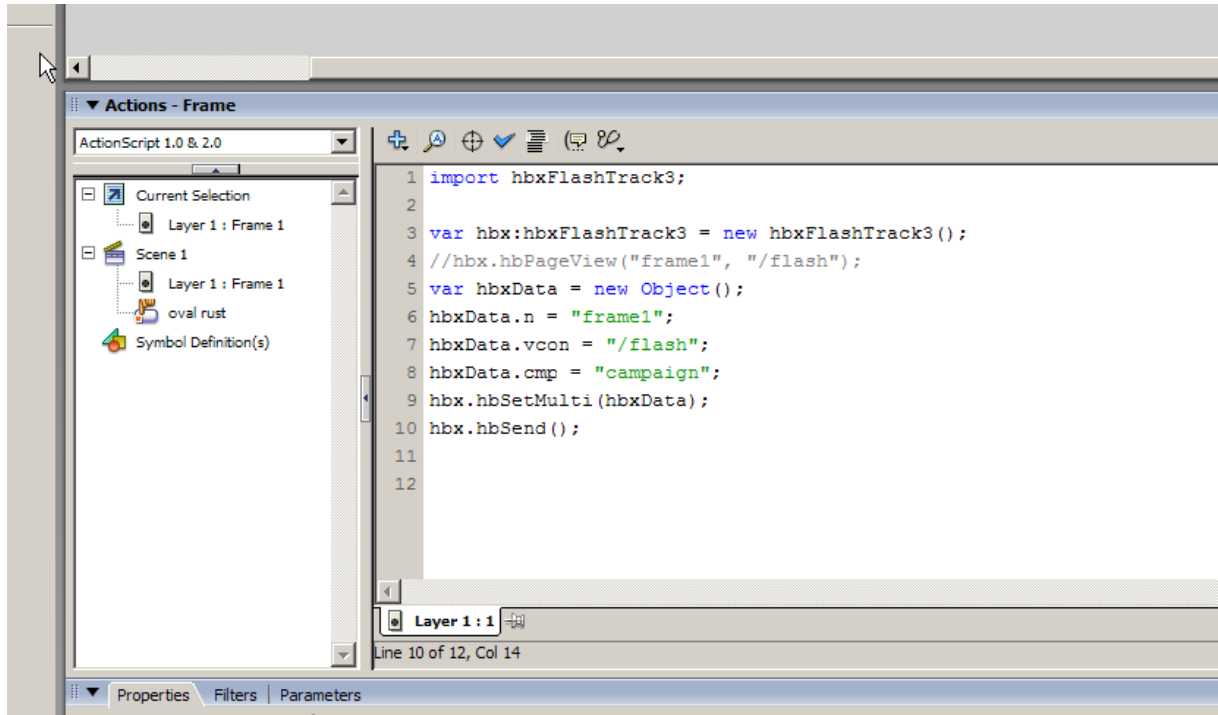


The following example shows how to track a download file with a link click request using the hbxFlashTrack3 methodology.

*(this sample assumes the hbx object is created in frame 1 of scene 1)*



The following example shows how to use the `_hbSetMulti` function. This example sets the page name "n", content category "mlc" and a campaign response "cmp" using the `hbxFashTrack3` methodology.



## Appendix A – hbxFashTrack2.as

```
//hbxFashTrack2.as,HBX2.5 ActionScript,COPYRIGHT 1997-2006 WEBSIDESTORY,INC. ALL RIGHTS
RESERVED. U.S.PATENT No.6,393,479B1 & 6,766,370. INFO:http://websidestory.com/privacy
import flash.external.ExternalInterface;

var _hbx = new Object();
_hbx.acct = "<!-- INSERT HBX ACCOUNT -->";
_hbx.gn = "<!-- INSERT HBX GATEWAY -->";
_hbx.protocol = _hbGetJSVal("window.location.protocol","http");

var _hbxBaseRequest = _hbInitBaseRequest();
var _hbxRequestData:LoadVars = new LoadVars();

function _hbInitBaseRequest(){
    var a:LoadVars = new LoadVars();
    a.hb=_hbx.acct;
    a.hec="-1";
    a.n="flash default";
    a.vcon="/";
    a.vpc="HBX0100ft2";
    a.vjs="HBX0250.1ft2";
    a.cd="1";
    a.hv="6";
    a.rf=_hbGetJSVal("window.document.referrer","bookmark");
    a.ja=_hbGetJSVal("window.navigator.javaEnabled()?'y':'n'");
    a.ln=System.capabilities.language;
    a.ss=System.capabilities.screenResolutionX+"*"+System.capabilities.screenResolutionY;
    a.sc=_hbGetJSVal("window.screen.colorDepth",_hbGetJSVal("window.screen.pixelDepth"));
    a.pu=_hbGetJSVal("window.document.URL");
    return a;
}

function _hbSend(){
    var _hbxResponse:LoadVars = new LoadVars();
    _hbxBaseRequest.hec++;
    _hbxRequestData.hid=(new Date()).getTime();
    for(var i in _hbxBaseRequest){_hbxRequestData[i]=_hbxBaseRequest[i]}
    _hbxRequestData.sendAndLoad(_hbx.protocol+"//"+_hbx.gn+"/cgibin/hg.pl",_hbxResponse,"GET"
);
    _hbxRequestData = new LoadVars();
}

function _hbSet(varName,
varValue){if(_hbxBaseRequest[varName]!=""){_hbxBaseRequest[varName]=varValue}else{_hbxRequestData
[varName]=varValue}}
function _hbSetAccount(varValue){ _hbxBaseRequest['hb']=varValue }
function _hbSetGateway(varValue){ _hbxBaseRequest['gn']=varValue }
function _hbSetCustomerId(varValue){ _hbxBaseRequest['customerid']=varValue }
function _hbSetMulti(obj){for(var i in
obj){if(_hbxBaseRequest[i]!=""){_hbxBaseRequest[i]=obj[i]}else{_hbxRequestData[i]=obj[i]}}}
function _hbPageView(pn,mlc){_hbxBaseRequest.hec=-1;_hbSet('n',pn);_hbSet('vcon',mlc);_hbSend()}
function _hbExitLink(el){_hbSet('el',el);_hbSend()}
function _hbDownload(fn){_hbSet('fn',fn);_hbSend()}
function _hbVisitorSeg(seg){_hbSet('seg',seg);_hbSend()}
function _hbCampaign(cmp){_hbSet('cmp',cmp);_hbSend()}
function _hbFunnel(fn1){_hbSet('fn1',fn1);_hbSend()}
function _hbGoalPage(gp){_hbSet('gp',gp);_hbSend()}
function _hbLink(lid,lpos){_hbSet('lid',lid);if(lpos!=undefined)_hbSet('lpos',lpos);_hbSend()}
function _hbForm(formState,focusField){if(formstate ==
0){_hbSet('lf',focusField)}else{_hbSet('sf',1)}_hbSend()}
function _hbMediaEvent(mName,mCurPos,mEndPos,mState,mClient,mVersion){var mObj=new
Object();mObj['m.f']=mName;
```



```
mObj[ 'm.cp' ]=mCurPos;mObj[ 'm.ep' ]=mEndPos;mObj[ 'm.s' ]=mState;mObj[ 'm.cl' ]=mClient;mObj[ 'm.cv' ]=mVersion;_hbSetMulti(mObj);_hbSend()
function _hbGetJSVal(val,dval){dval=(dval!="null")?dval:"";var
v=String(ExternalInterface.call("eval",val));
if(v!=undefined&&v!="null")return v;return dval}
```



## Appendix B – hbxFashTrack3.as

//hbxFashTrack3.as,HBX2.5 ActionScript,COPYRIGHT 1997-2007 WEBSIDESTORY,INC. ALL RIGHTS RESERVED. U.S.PATENT No.6,393,479B1 & 6,766,370. INFO:http://websidestory.com/privacy

```
import flash.external.ExternalInterface;

class hbxFashTrack3 {

    private var acct = "<!-- HBX ACCOUNT -->";
    private var gn = "<!-- HBX GATEWAY -->";
    private var protocol = "";
    private var vpc = 'HBX0100ft3';
    private var vjs = 'HBX0250.1ft3';
    private var req:LoadVars;
    private var reqData:LoadVars;

    function hbxFashTrack3(){
        this.req = new LoadVars();
        this.reqData = new LoadVars();
        this.req['hb']=this.acct;
        this.req['hec']='-1';
        this.req['n']='flash default';
        this.req['vcon']='/' ;
        this.req['vpc']=this.vpc;
        this.req['vjs']=this.vjs;
        this.req['cd']='1';
        this.req['hv']='6';
        this.req['ln']=System.capabilities.language;
        this.req["ss"]=System.capabilities.screenResolutionX+"*"+System.capabilities.screenResolu
tionY;
        this.req['rf']=hbGetJSVal("window.document.referrer","bookmark");
        this.req['ja']=hbGetJSVal("window.navigator.javaEnabled()?'y':'n'");

        this.req['sc']=hbGetJSVal("window.screen.colorDepth",hbGetJSVal("window.screen.pixelDepth
"));
        this.req['pu']=hbGetJSVal("window.document.URL");
        this.protocol=hbGetJSVal("window.location.protocol","http");
    }

    public function hbSend(){
        var _response:LoadVars = new LoadVars();
        this.req['hec']++;
        this.reqData['hid']=(new Date()).getTime();
        for(var i in this.req){
            this.reqData[i]=this.req[i];
        }
        this.reqData.sendAndLoad(this.protocol+"//"+this.gn+"/HG",_response,"GET");
        this.reqData=new LoadVars();
    }

    public function hbSet(varName, varValue){
        if(this.req[varName]!=""){
            this.req[varName]=varValue;
        }else{
            this.reqData[varName]=varValue;
        }
    }

    public function hbSetAccount(varValue){
        this.req['hb']=varValue ;
    }

    public function hbSetGateway(varValue){
```



```
        this.req['gn']=varValue;
    }

    public function hbSetCustomerId(varValue){
        this.req['customerid']=varValue;
    }

    public function hbSetMulti(obj){
        for(var i in obj){
            if(this.req[i]!="")this.req[i]=obj[i];
            else this.reqData[i]=obj[i];
        }
    }

    public function hbPageView(pn,mlc){
        this.req['hec']=-1;
        hbSet('n',pn);
        hbSet('vcon',mlc);
        hbSend();
    }

    public function hbExitLink(el){
        hbSet('el',el);
        hbSend();
    }

    public function hbDownload(fn){
        hbSet('fn',fn);
        hbSend();
    }

    public function hbVisitorSeg(seg){
        hbSet('seg',seg);
        hbSend();
    }

    public function hbCampaign(cmp){
        hbSet('cmp',cmp);
        hbSend();
    }

    public function hbFunnel(fnl){
        hbSet('fnl',fnl);
        hbSend();
    }

    public function hbGoalPage(gp){
        hbSet('gp',gp);
        hbSend();
    }

    public function hbLink(lid,lpos){
        hbSet('lid',lid);
        if(lpos!=undefined)hbSet('lpos',lpos);
        hbSend();
    }

    public function hbForm(formState,focusField){
        if(formState==0)hbSet('lf',focusField)
        else hbSet('sf',1);
        hbSend();
    }

    public function hbMediaEvent(mName,mCurPos,mEndPos,mState,mClient,mVersion){
        var mObj=new Object();
        mObj['m.f']=mName;
        mObj['m.cp']=mCurPos;
        mObj['m.ep']=mEndPos;
        mObj['m.s']=mState;
        mObj['m.cl']=mClient;
```





```
        mObj['m.cv']=mVersion;
        hbSetMulti(mObj);
        hbSend();
    }

    public function hbGetJSVal(val,dval){
        dval=(dval!="null"?dval:"");
        var v=String(ExternalInterface.call("eval",val));
        if(v!=undefined&&v!="null")return v;
        return dval;
    }
}
```

## Appendix C – Persisted Request Data

This appendix contains the collection of HBX request variables that are persisted between requests. These values can be set and reset throughout the implementation, but after the initial setting, they will persist through all subsequent tracking events. For example, sending a pageview event with the pagename “flash page” and a content category of “/” would cache the pagename and content category data. On a subsequent link click or event type call, the pagename and content category data would be sent in addition to the link click data which is the correct behavior.

The values persisted are:

- protocol
- gateway name
- account (gateway var: hb)
- location referrer (gateway var: rf)
- page URL (gateway var: pu)
- pagename (gateway var: n)
- content category (gateway var: vcon)
- customerId (gateway var: customerid)
- screen size (gateway var: ss)
- screen color depth (gateway var: sc)
- java enabled status (gateway var: ja)
- user language (gateway var: ln)

The protocol can be changed by setting the internal attribute of the hbxFashTrack object. (Example: `hb.protocol = “https”;`)

The gateway name should not normally be changed, but if required there is an `hbSetGateway` convenience function to address this.

The account can be changed via the `hbSetAccount` call.

The `customerId` can be changed via the `hbSetCustomerId` call.

The remaining variables can all be altered using the `hbSet` call. The input parameters for the `hbSet` call are the gateway variable, and the corresponding value.

## Appendix D – Gateway Variables

The following table shows all gateway variables currently in HBX and HBX commerce. It also indicates HBX page code variables with no gateway variable equivalents and gateway variables with no HBX page code equivalents.

page code variable	gateway variable	definition
hbz.acct	hb	ACCOUNT NUMBER
hbz.pn	n	PAGE NAME
hbz.mlc	vcon	MULTI-LEVEL CONTENT
hbz.seg	seg	POPULATION GROUP
hbz.cmp	cmp	CAMPAIGN ID
hbz.gp	gp	GOAL PAGE
hbz.fnl	fnl	FUNNEL ID
hbz.pec	pec	PAGE ERROR CODE
hbz.dcmp	dcmp	DYNAMIC CAMPAIGN ID
hbz.dcmpre	dcmpre	DYNAMIC CAMPAIGN RESPONSE EXPIRATION
hbz.dcmpe	dcmpe	DYNAMIC CAMPAIGN EXPIRATION
hbz.hra	ra	CAMPAIGN RESPONSE ATTRIBUTE
hbz.hcn	gn	CAMPAIGN CONVERSION ATTRIBUTE
hbz.hlt	ld	CAMPAIGN LEAD TRACKING
hbz.hla	la	CAMPAIGN LEAD ATTRIBUTE

hbx.hcv	cv	CAMPAIGN CONVERSION VALUE
hbx.cp	cp	legacy campaign Id for HBX Commerce
hbx.hc1	c1 or cv.c1	CUSTOM METRIC 1
hbx.hc2	c2 or cv.c2	CUSTOM METRIC 2
hbx.hc3	c3 or cv.c3	CUSTOM METRIC 3
hbx.hc4	c4 or cv.c4	CUSTOM METRIC 4
	cv.cN	CUSTOM METRIC N
hbx.ci	customerid	CUSTOMER ID
hbx.rf	rf	indicates custom referrer
hbx.lt	tt	link analysis mode
cnv.value	cnv.value	custom conversion value
cnv.id	cnv.id	custom conversion ID
Gateway variables with no HBX page code equivalents	gateway variable	definition
	lid	LINK ID
	lpos	LINK TRACKING POSITION
	lv.id	Link view and link id list
	lv.lpos	Link view and link position list
	el	EXIT LINK
	sf	INDICATES FORM COMPLETION
	lf	FORM ABANDONMENT LAST FIELD FOCUSED

fn	INDICATES DOWNLOAD
search.keyword	INTERNAL SEARCH KEYWORD
search.results	SUCCESSFUL RESULTS BOOLEAN (1-SUCCESSFUL SEARCH, 0-FAILED SEARCH)
search.attr1	
through	INTERNAL SEARCH ATTRIBUTES
search.attr4	
m.f	name of tracked media file
m.cp	current position of tracked media file in milliseconds
m.ep	End position of tracked media file in milliseconds
m.s	media state (play, pause, stop, playp)
m.cl	media player type
m.cv	media client version
m.tt	media tracking type
pu	page Url

Draft	Author	Description	Date
1	Mike Anderson	Initial draft	Dec 19, 2006
2	Mike Anderson	Updated draft with ActionScript 2 and 3 support	Jan 4 <sup>th</sup> , 2006

**Document prepared by**

Mike Anderson  
Manager  
*Enterprise Technology Group*  
WebSideStory, Inc.  
10182 Telesis Court, 6th Floor  
San Diego, CA 92121  
[p] 760.716.5776  
[f] 858.546.0480  
[e]  
[manderson@websidestory.com](mailto:manderson@websidestory.com)

**United States**

10182 Telesis Court, 6th Floor  
San Diego, CA 92121  
[P] 858.546.0040  
[F] 858.546.0480

**EMEA**

Neptunusstraat 23  
2132 JA Hoofddorp, NL  
[P] +31 (0) 23-5677900  
[F] +31 (0) 23-5541011

**United Kingdom**

212 Piccadilly  
London, W1J 9HG  
[P] +44 (0) 207 917 6280  
[F] +44 (0) 207 917 6281

Founded in 1996, WebSideStory is a leading provider of on-demand Web analytics services. WebSideStory's enterprise solutions provide actionable, real-time information about online visitor and customer behavior. More than 600 enterprises rely on WebSideStory's solutions to sell more products and services, increase online lead generation and enhance online customer support.

**For more information, visit [www.WebSideStory.com](http://www.WebSideStory.com) or call 877.2BUY.HBX**