OMNITURE™

FLASH TRACKING (H CODE)

Tracking Events in Flash

June 15, 2006   Version 1.1

1   Tracking Flash Objects

Omniture's data collection process is to accept an image request and then to report on the data contained in the string of the request. The traditional method of creating an image request is through browser-based HTML pages using JavaScript. Macromedia's Flash has the ability to either execute image requests through the traditional method of executing JavaScript on an HTML page, or it can create image requests independent of the browser and JavaScript. This section describes the implementation of Flash for Omniture tracking, and will provide options for various implementation methods.

1.1 Functional Description

As noted in the overview, there are two main methods of implementing Flash.

- JavaScript (standard)
- Non-JavaScript - Independent of JavaScript (executable files, etc.)

The decision to implement either method needs to be made carefully. It may seem logical to jump right to the option of using Flash independent of JavaScript, but there are Pros and Cons to consider. The following matrix outlines some of the functional differences between the methods.

Table 1-A: Functional Differences of JavaScript vs. No JavaScript

Functionality	JavaScript	No JavaScript
Track usage outside of browser	no	yes
Use ClickMap	yes	no
Track unique visitors	yes	no
Track visits	yes	no
Track pathing	yes	no
Uses cookies	yes	no

The matrix above shows that browser based, JavaScript implementations with Flash can provide the most advanced web metrics for SiteCatalyst. The majority of this documentation will be related to implementing with JavaScript, because this is Omniture's recommended approach. Non-JavaScript implementations will also be touched on, but will be limited to syntax and brief explanations.

1.1.1   JavaScript Implementation

The reason the JavaScript Implementation is named as such is because while the Flash application triggers whether the image request should be made, and what information the request should contain, it is the JavaScript on the HTML page that creates and executes the image request. This means that Flash is simply using getURL("javascript:…"); to set some Omniture variables on the HTML page, and triggering the included Omniture JavaScript file to execute an image request.

The following items are required for JavaScript implementation.

- Omniture's "Code-to-Paste" (which sets some JavaScript variables and includes the Omniture JavaScript file "s_code.js")

- Omniture's JavaScript file (to be included by the "Code-to-Paste" on the HTML page ("s_code.js")
- Flash ActionScript to execute JavaScript on the HTML page when a "track-worthy" user action is performed

The *Technical Description* in the following section describes the syntax of executing the Omniture JavaScript in order to create an image request. With an understanding of the syntax, you can determine the best method of implementing for your unique case by referring to the *Use Case/Example* Section for sample implementation strategies.

1.1.2 Non-JavaScript Implementation

Flash can bypass the use of JavaScript and create its own image request to Omniture servers. It is important to understand that anytime Flash creates the image request with this approach it is considered "outside the browser", even if the file is used in a browser. (See Note Below) Making an image request "outside the browser" means that a uniqueID cannot be created for users through the browser's third or first-party cookies. Without a uniqueID for users, each image request will appear to Omniture's systems as though it is coming from a unique visitor even if the same user has submitted the images. For this reason, it is important to understand that unique visitor counts in your SiteCatalyst reports will be inaccurate without a means of identifying images to unique users with this Flash non-JavaScript approach. It is therefore recommended that traffic reported through this approach be sent to a separate report suite where it is understood that visits, unique visitors, pathing, cookies, and any persistent variables (eVars, campaign variable etc.) will not be accurate.

The following items are required for a non-JavaScript implementation.

- Flash ActionScript to create the image request when a "track-worthy" user action is performed

NOTE: Non-JavaScript implementations using HTML or ServerSide code in an HTML page are different from Flash non-JavaScript implementations because the HTML versions can set and refer to a third-party cookies through the browser to identify unique users, whereas implementations "outside the browser" cannot. The implications of not accurately identifying unique visitors are listed in the functional description above.

1.2 Technical Description

The following information is useful for both JavaScript and non-JavaScript implementations.

1.2.1 Javascript Implementation

In a standard implementation, Flash uses JavaScript to execute an image request on the HTML page. This has many benefits. When implementing with this method, the Omniture JavaScript also collects information on browser technologies and has the ability, if implemented in the Flash files, to allow for ClickMap reporting. Additionally, variables defined on the HTML page, and potentially unavailable to Flash, can be sent in the generated image request.

The linear execution of a Flash triggered JavaScript image request is as follows.

The HTML page loads with Omniture code.

1. As the Omniture code on the page executes, the Omniture JavaScript file ("s_code.js") is loaded and JavaScript variables are set with values like the following.

   - `s.pageName='home page'`
   - `s.channel='media'`

2. An image request is created by the JavaScript, which sends the data set on the page to Omniture's servers for reporting.

3. Flash application, included in the HTML page, executes some command, user initiated or not, which executes the getURL command in Flash and sends through a JavaScript command to the HTML page to create an image request using Omniture's JavaScript file.

4. Omniture's JavaScript file creates the image request with all the declared variables on the page and sends this to Omniture's data collection servers.

NOTE: It may be important to your business requirements on some pages not to capture an image request when the page loads. In this case, the Omniture code on the HTML page, which makes the image request, needs to be altered for this unique application. Please see appendix A for more information.

1.2.2 JavaScript functions for initiating an image request with Flash "t()" and "tl()"

The standard Omniture JavaScript code contains two functions used to track Flash events. Both the t() and tl() functions have different properties and uses.

Function Name	Use
t()	This function is used to track a "virtual" page load in Flash. Calling this function results in a page view.
tl()	This function is used to track links. Calling this function results in a Custom Link (Custom, Download, or Exit)

Using t()

Each time the t() function is called, SiteCatalyst increments the total page views for the site. Additionally, any variables (Omniture JavaScript variables) that have a value (other than an empty string: "") at the time the function is called, are sent to SiteCatalyst. Therefore, it is essential to reset all variables that shouldn't be resent to Omniture from within the Flash file in the getURL command before calling t() and creating the image request. This will prevent the values of those variables from being sent to Omniture with each Flash image request. The following outlines how and when t() should be used.

- Use t() when an event should be considered a page view, for example, when moving from one "page" to another in a mulit-"page" Flash application

- Make sure to set the pageName variable each time you call the t() function. If the pageName variable is not changed to an appropriate value for the Flash application, the page containing the Flash file will get credit when the image request is made, causing an inflated number of page views for that HTML page / URL.

- All variables which shouldn't be sent through to SiteCatalyst from the JavaScript on the HTML page, including variables previously set by the Flash application, should be reset to "". This is to ensure that they are not recorded in SiteCatalyst each time the t() function is called.
    - For example, prop1 was previously set to s.prop1="somevalue" in a call to the t() function. On a current image request, if your business rule requires that the value should not be sent again; prop1 should be reset to s.prop1='' in the Flash call to getURL. Changing the value to '' (empty string) will ensure that SiteCatalyst does not record it.

        Example: getURL("javascript: var s=s_gi('your_report_suite-here'); s.pageName='my Flash File';s.prop1='';void(s.t()));");

NOTE: The function is lowercase.

The previous section showed that t() sends data in the same way that a page load does. The tl() function, however, sends data in the same way that custom link tracking does. In other words, the total page views to the site are not incremented when calling tl(), but non-page view data is recorded.

Variables and events that you want to include with tl() must be identified in the s.linkTrackVars and s.linkTrackEvents variables. For example, if you would like to send data for prop1, eVar4, and event3 using the tl() function, set linkTrackVars and linkTrackEvents before calling the function, as shown below.

Example: getURL("javascript: var s=s_gi('your_report_suite_here'); s.prop1='some Traffic value'; s.eVar4='some Commerce value'; s.events='event3'; s.linkTrackVars='prop1,eVar4,events'; s.linkTrackEvents='event3'; void(s.tl(true, 'o', 'my custom link'));");

NOTE: Any variables or events identified in linkTrackVars or linkTrackEvents will only be sent if the variable has a value other than '' (empty string). This includes variables already set in the JavaScript on the HTML page.

The following list outlines how and when the tl() function should be used.

- tl() should be used when a Flash code execution should NOT be considered an additional page view, i.e. to send information about a button clicked or a text field updated in the Flash application.
- The pageName variable is not recorded when you use tl().
- All variables and events that are used in conjunction with tl() must be identified in linkTrackVars and linkTrackEvents.
- All variables and events identified in the linkTrackVars and linkTrackEvents which should not be sent with the image request, should be reset to '' (empty string) before calling tl().

  For example, prop1 was previously set to prop1="somevalue" in a call to the tl() function. On a current image request, if your business rule requires that the value should not be sent again; prop1 should be reset to s.prop1='' in the Flash call to getURL. Changing the value to '' (empty string) will ensure that SiteCatalyst does not record it.

- All parameters are required when calling the function. Each parameter is described in the table below.
  function reference : tl(**Object**, **Type**, **Linkname**);

Table 1-B: Parameters

Parameter	Description
Object	This is the link object for custom link tracking. For Flash purposes, the Boolean value should equal "true."
Linkname	Linkname is the friendly name for the link that displays in the Custom, Download, or Exit links report. If this parameter is left out, then the default value of linkName is used.
Type	Type can be defined by one of three values, including the default 'o' (Custom), 'e' (Exit), or 'd' (Download), respectively. These values identify which of the three Custom Link reports in which the Linkname displays.

Using getURL

The following steps outline the process used to track user actions in Flash applications. The steps displayed and syntax used may vary depending on the version of Flash you are using.

5. Open the Flash (.fla) file you will be using to track with SiteCatalyst.

6. Using ActionScript, use the getURL function to send a JavaScript command to the HTML page and execute the t() function. A call to getURL is made on a user action which you would like to track, such as a user clicking a button. The following is sample syntax for the getURL function call using the "on (press)" event (the example only shows a few variables used, but other SiteCatalyst variables could be used in addition to these).

```
on (press) {
getURL("javascript: var s=s_gi('your_report_suite_here'); s.pageName='Flash Order';
s.channel='Music'; s.prop1='Download MP3'; void(s.t());");
}
```

NOTE: "var s=s_gi('your_report_suite_here');" declares the object the JavaScript variables belong to on the HTML page. In this case, it is the object "s".

If you wish to treat the user action (clicking a button in this case) as a custom link, use a call to the tl() function, which displays the information in the Custom Links report in SiteCatalyst, but does not increase page views.

```
on (press) {
getURL("javascript: var s=s_gi('your_report_suite_here');
s.linkTrackVars='prop1,prop2'; s.prop1='Download MP3'; s.prop2='Rock'; void(s.tl(true,
'o', 'Link to Rock Music'));");
}
```

'Link to Rock Music' is the link name that displays in the Custom Links report in SiteCatalyst.

NOTE: The base account ('your_reportsuite_here') is set on the HTML page either through the JavaScript "Code to Paste," or through the JavaScript file (s_code.js), and does not need to be reset in the t() or tl() function call from Flash. If left blank "var s=s_gi()", the s_account variable declared on the HTML page will be used. **However, the "var s=s_gi()" command is required.** Any other variables that consecutively remain the same can also be set on the HTML page.

1.2.3 Non-JavaScript Implementation

As previously noted, Flash can create its own image request. The steps displayed and syntax used may vary depending on the version of Flash you are using.

7. Open the Flash (.fla) file you will be using to track with SiteCatalyst.

8. Add a similar function to the one shown below to a globally accessible frame in your root timeline:

```
var s_account = 'your_report_suite_here';
var s_dataCenter = '112';
var nextDepth = this.getNextHighestDepth();

function s_vp_hci(pn, p1)
{
//random number used to avoid retrieving a "cached" version of the image for this
user
var rn = int(Math.random()*1000000);
//image request string
s_image='http://'+s_account+'.'+s_dataCenter+'.2o7.net/b/ss/'+s_account+'/1/NS/'+rn
+'?pageName='+pn+'&c1='+p1;
loadMovieNum(s_image, nextDepth);
}
```

Table 1-C: Functions Described

Function	Description
var s_account	var s_account is the report suite ID for the report suite where the information should be reported
dataCenter	dataCenter is the prefix for the data center where your company information is stored, (112 for San Jose, and 122 for Dallas – ask your Implementation Consultant if you are unsure)
nextDepth	nextDepth gets the next highest depth on the current "root" timeline to ensure that the returned 1x1 transparent pixel is not replacing any critical content in the application.
rn (random number)	rn is used in the image request to ensure that if the application were ever used in a browser, that it wouldn't reference a cached image, but instead would request a new one from Omniture.
image	image is the URL-encoded image request string composed of some of the other variables, including the passed parameters. Change this and the parameters as needed to accommodate your reporting needs. Notice that the variables in the image request are different than in JavaScript. This is to reduce the size of the image request. A variable mapping is shown in the table below.

Table 1-D: JavaScript Variable vs. Query String Parameters

JavaScript Variable	Query String Parameter
pageName	pageName
server	server
pageType	pageType
channel	ch
prop1 – s.prop50	c1 - c50
hier1 - s.hier5	h1 - h5
campaign	v0
state	state
zip	zip
events	events

products	products
purchaseID	purchaseID
eVar1 – s.eVar50	v1 - v50
charSet	ce
currencyCode	cc
Link Type	pe (lnk_d, lnk_e, lnk_o)
Link URL	pev1
Link Name	pev2
Referring URL	r
Current URL	g

With the function which creates the image request in a global location, any user action in the Flash application can call that function and execute an image request. Below is an example:

```
_root.one_btn.onRelease = function() {
 s_vp_hci("home page", "some value for Prop1");
};
```

Be cautious of rapid sequential image requests from the same Flash application. Triggering rapid image request right after another will not allow Flash to fully execute image requests, causing some requests to cancel others. It is not recommended to execute two or more image requests within a second of each other as the second may cancel the first depending on computer processing speed and Internet connection et cetera.

1.2.4 Use Case/Example

The following sections contain use cases that explain tracking click-through Flash files and tracking a Flash application.

Click-Through Flash File

There are two effective methods of tracking click through Flash files. The choice of which method to go with depends on the landing page.

9. If the landing page is coded for SiteCatalyst tracking, in a report suite you want the data to appear in, using a query string parameter in the link from the click through is a great option.

Table 1-E: Example 1

Parameter	Description
Required Code	▪ Flash file adjustment to pass query parameter for landing page ▪ Omniture SiteCatalyst JavaScript code on the landing page with getQueryParam() set in the JavaScript file to capture the value of the

	parameter and populate a SiteCatalyst variable.

Flash ActionScript	On the click:
	
	``
	on (press) {
	getURL("http://www.somedomain.com/somepage.html?pid=f12345");
	}
	``

Omniture JavaScript	In the doPlugins() function of the s_code.js file –
	
	``
	/* Plugin Config */
	s.usePlugins=true
	function s_doPlugins(s) {
	 /* Add calls to plugins here */
	 s.eVar1=s.getQueryParam('pid');
	}
	s.doPlugins=s_doPlugins
	``
	
	🖉 **NOTE:** This assumes "pid" is the query string parameter and that eVar1 will be used. Please replace eVar1 and 'pid' with the variable and parameter for your custom application */

10. If the landing page is not coded for SiteCatalyst tracking, or if the click should be recorded in a different report suite than the landing page, use Flash to trigger a custom image request before the page changes. This example will use the tl() function and not create a page view in SiteCatalyst, but instead, record data in other SiteCatalyst variables of your choice and create an instance of a custom link.

Table 1-F: Example 2

Parameter	Description
Required Code	▪ Flash ActionScript to trigger the JavaScript on the current html page to create a new image request before changing the page in the browser ▪ Standard Omniture SiteCatalyst JavaScript (H) code on the current html page

Flash ActionScript	On the click:
	
	``
	on(press){
	getURL("javascript: var s=s_gi('your_reportsuite');
	s.prop1='Download MP3 Music'; void(s.tl(true, 'o', 'Flash Link
	to Downloads')); window.open('http://www.somedomain.com',
	'popup')");
	}
	``
	
	This code allows the window containing the Flash object to remain open so that tl() has time to fire and request an image from SiteCatalyst.

Flash Application: Tracking a Flash Application Similar to a Web Site

Flash is becoming more and more prevalent as a choice for interactive applications, and even as a substitute for complex html pages. If you have a Flash application which has "pages" or other important metrics to capture, you can use both the t() function for page view metrics, and tl() for custom link type metrics.

11. Page View equivalent in a Flash application. Moving from a main menu to a "page" with specific details, or moving from one detail page to another, etc. may be a good time to send a page view to SiteCatalyst for path analysis or traffic information. In this case, it is appropriate to use the t() function.

Table 1-G: Example 3

Parameter	Description
Required Code	▪ Flash ActionScript to trigger the JavaScript on the current html page to create a new image request ▪ Standard Omniture SiteCatalyst JavaScript (H) code on the current html page

Flash ActionScript	On the click –
	
	``
	on (press) {
	getURL("javascript: var s=s_gi('your_report_suite_here');
	s.pageName='Flash App. : Home'; s.channel='Finance';
	s.prop1='Registered User'; void(s.t()); ");
	}
	``

12. Custom link equivalent in a Flash application. If you are interested in tracking links clicked without incurring a page view, using a custom link is appropriate. With this approach, other SiteCatalyst variables can still be sent to your reports, but a page view is not recorded.

Table 1-H: Example 4

Parameter	Description
Required Code	▪ Flash ActionScript to trigger the JavaScript on the current html page to create a new image request ▪ Standard Omniture SiteCatalyst JavaScript (H) code on the current html page

Flash ActionScript	On the click:
	
	``
	on (press) {
	getURL("javascript: var s=s_gi('your_report_suite_here');
	s.linkTrackVars='prop1,prop2'; s.prop1='Download MP3';
	s.prop2='Rock'; void(s.tl(true, 'o', 'Link to Rock
	Music')); ");
	}
	``

1.2.5 Reference Tables (Parameters, Max Values, chartypes, etc.)

All SiteCatalyst variables set through Flash applications are subject to the same restrictions as with an html implementation.

1.2.6 Technical Considerations and Implementation Issues

The following information is important for technical consideration during implementation.

- When SiteCatalyst variables are set from the Flash application the html page holds the value of those variables in memory as long as the html page isn't refreshed. This means that when an image request is created by the JavaScript, whether initiated by Flash or the HTML page, all variables with values other than "" (empty string) will be sent with the image request when using the t() function.

  Example: if prop1 was previously set to "MP3 Downloads", but on the current image request should not be set.

  ```
  getURL("javascript: var s=s_gi('your_report_suite_here'); s.pageName='Flash
  App. : Home'; s.channel='Finance'; s.prop1=''; void(s.t());");
  ```

- The tl() function will only send through the variables declared in the linkTrackVars variable and in the case of events, only the events in the linkTrackEvents variable.

  Example: If sending prop1 and evar1:

  Flash ActionScript –

  ```
  getURL("javascript: var s=s_gi('your_report_suite_here');
  s.linkTrackVars='prop1,eVar1'; s.prop1='some value'; s.eVar1='some value';
  void(s.tl(true, 'o', 'some link'));");
  ```

  Example: if sending prop1, eVar1 and event1 –

  ```
  getURL("javascript: var s=s_gi('your_report_suite_here');
  s.linkTrackVars='prop1,eVar1'; s.linkTrackEvents='event1'; s.prop1='some
  value'; s.eVar1='some value'; s.events='event1'; void(s.tl(true, 'o', 'some
  link'));");
  ```

- The "getURL" command can only be called once during a movie "frame." This is not an issue with some implementations that primarily track mouse clicks and similar events instead of movie frames.

- If "getURL" must be called more than one time in the same "frame," it may be necessary to use a queuing function in Flash to ensure that secondary calls to "getURL" don't override previous calls. Macromedia has built an analytics component to address this. The component must be configured for your custom application. Omniture has customized this component slightly to comply with the latest omniture JavaScript code. If your application requires this type of a component to queue calls to getURL, please contact your Omniture Implementation Consultant for a copy of this component. The component requires Flash MX or newer.

- Using "fscommand" is not as compatible a method of calling JavaScript as "getURL." Particularly, "fscommand" is not compatible with Mac IE. Omniture recommends using the "getURL" command instead.

- Calling getURL from the Flash file causes the browser to cancel any pending HTTP: requests. Anything that causes a load just after the Omniture call may prevent the image request from correctly setting any cookies, etc. (although it may still register the page view or click event).

1.2.7 Additional Information

The following code can be used when using the s_code.js.file An initial image request will not be sent when the page loads, and data will not be sent until the Flash object specifically calls the t() or tl() functions.

NOTE: The code shown below differs from standard code (version H.0 and above) in that there is no script tag that writes the image to the page.

```
<!-- SiteCatalyst code version: H.0.
Copyright 1997-2005 Omniture, Inc. More info available at
http://www.omniture.com -->
<script language="JavaScript" src="http://INSERT-DOMAIN-AND-PATH-TO-CODE-
HERE/s_code.js"></script>
<script language="JavaScript"><!--
/* You may give each page an identifying name, server, and channel on
the next lines. */
s.pageName=""
s.server=""
s.channel=""
s.pageType=""
s.prop1=""
s.prop2=""
s.prop3=""
/* E-commerce Variables */
s.campaign=""
s.state=""
s.zip=""
s.events=""
s.products=""
s.purchaseID=""
s.eVar1=""
s.eVar2=""
s.eVar3=""
/************* DO NOT ALTER ANYTHING BELOW THIS LINE ! **************/
/* NO IMAGE REQUEST SENT */
//var s_code=s.t();if(s_code)document.write(s_code)//--></script>
<!-- End SiteCatalyst code version: H.0. -->
```

2 Flash and ClickMap

ClickMap supports the reporting of clicks within Flash movies or menus. The Flash movie must call the s.t() or s.tl() function in order for the data collection about clicks to be sent to Omniture. Each click or "virtual page load" within the Flash movie must trigger one of those functions. The only "automatic" feature is that the location of the clicks is detected and reported through an overlay.

2.1 Assumptions

The Flash movie has implemented a s.t() function at the beginning of each "virtual page" that is displayed to the user. The page is uniquely identified by a pageName that is different for each page of the movie (and actually it should be unique across the entire site). The developer is required to implement a "page view" within the Flash as described earlier in this document.

NOTE: Links that take the visitor away from the site must use the s.tl(), and designate the link click as an "Exit Link" in order to collect ClickMap data. Otherwise, it is assumed that the data will be sent in on the next "page" within the site.

2.2 Step 1: Modify the Flash Movie File (.fla)

The first step is to insert the SiteCatalyst ClickMap tracking component into the first frame of the Flash movie file (.fla).

1. Open the s_clickmap.fla movie file (supplied by Omniture).
2. View the Library contents of the s_clickmap.fla (Press F11).
3. Open the Flash movie file (.fla) in which ClickMap is to be implemented.
4. Create a new layer (Insert > Layer) and select the first frame of the new layer.
5. Drag the "SiteCatalyst ClickMap" component from the Library onto the stage.

NOTE: Be sure that the component persists to the end of the timeline as shown below.

Figure 2-A: Flash Timeline
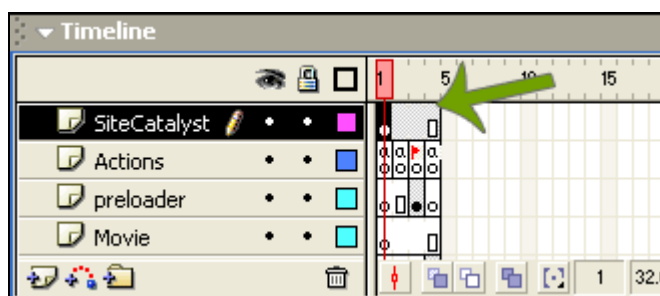
6. Publish the movie and save your changes (File > Publish).

2.3 Step 2: Add Supporting JavaScript to HTML

Flash ClickMap tracking is dependent on the following JavaScript components which reside in the HTML page.

▪ (This should already be completed as a part of a standard implementation). Standard SiteCatalyst JavaScript tracking file (s_code.js) must be present on the page. Note that if this file is not available on the HTML page containing the Flash movie, some JavaScript errors may occur if the Flash movie is implemented as described.

- ▪ (This should already be completed as a part of a standard Flash implementation, as described earlier in this document). The standard Flash tracking functions (s.t() and s.tl()) must be defined within the HTML page.

NOTE: These functions must be called in order to send data to SiteCatalyst, as described above. Exit links must be identified as such in order to correctly collect data.

- ▪ **New:** The Implementation Consultant should install the ClickMap Flash Plug-in. For more information on the ClickMap Flash Plug-in, contact your Omniture Implementation Consultant.
- ▪ **New:** Set `s.inlineStatsHandleMovie('flash_id')` once for each Flash movie on the page, as shown below.

```
<script language="javascript">
<!--
s.inlineStatsHandleMovie('my_movie_id');
s.inlineStatsHandleMovie('my_movie_id2');
...etc...
//-->
</script>
```

NOTE: The movie ID can be found in the <object> and <embed> tags used to create an instance of the Flash movie on the HTML page.

```
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=
6,0,0,0" WIDTH="470" HEIGHT="300" id="my_movie_id" ALIGN="">
<EMBED src="curb_styles.swf" quality=high bgcolor=#FFFFFF  WIDTH="470" HEIGHT="300"
NAME="my_movie_id" ALIGN="" TYPE="application/x-shockwave-flash"
PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer"></EMBED>
</OBJECT>
```

2.4 Step 3: Modify Flash <object> Tags

Each Flash object must include the following required parameters:

1. Add the "wmode" <param> tag (see example below)

   <param name="wmode" value="transparent">

2. Add the "allowScriptAccess" <param> tag (see example below)

   <param name="allowScriptAccess" value="sameDomain" />

2.4.1 Sample Flash <object> tag:

```
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab
#version=6,0,0,0" WIDTH="470" HEIGHT="300" id="my_movie_id" ALIGN="">

<param name="wmode" value="transparent">

<param name="allowScriptAccess" value="sameDomain" />

<PARAM NAME=movie VALUE="my_movie.swf">

<PARAM NAME=quality VALUE=high>

<PARAM NAME=bgcolor VALUE=#FFFFFF>
```

```
<EMBED src="my_movie.swf" quality=high bgcolor=#FFFFFF  WIDTH="470"
HEIGHT="300" NAME="my_movie_id" ALIGN="" TYPE="application/x-shockwave-flash"
PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer"></EMBED>

</OBJECT>
```

2.5 Important Points to Remember

- The pageName variable is required for tracking links within Flash. Otherwise, ClickMap overlays will not be displayed.

- s.t() or s.tl() must be called specifically from within the Flash movie in order to collect link data.

- ClickMap data is sent to Omniture on the next "virtual" page (the next time that s.t() or s.tl() is called) — not on the when the click actually occurs.

- Links that exit the site must be tagged as Exit Links using s.tl() - otherwise, exit link ClickMap data will not be captured.

OMNITURE™

→
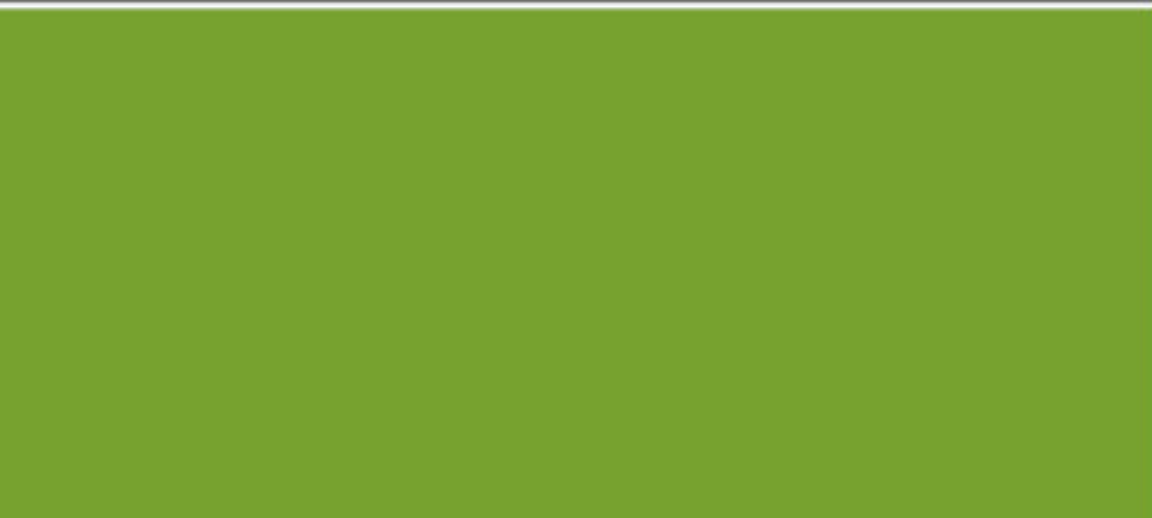
CALL 1.877.722.7088
1.801.722.0139

www.omniture.com
info@omniture.com

OMNITURE™

550 East Timpanogos Circle
Orem, Utah 84097