OMNITURE™

WHITE PAPER

OMNITURE IMPLEMENTATION

Common Code Mistakes

September 2, 2009 Version 1.0

1 JavaScript Code Mistakes

A successful implementation means drawing useful data from your website. An unsuccessful implementation can mean spending time looking through your JavaScript code to find errors. In an attempt to save you time and prevent frustration, the purpose of this white paper is to show you some of the most common code mistakes users make when implementing SiteCatalyst code.

1.1 Putting SiteCatalyst Code in the <head> Tag

Previously, a common implementation practice was to place the SiteCatalyst JavaScript code between the <head> and </head> tags. The attraction was that by placing the code between these tags prevented the 1×1 pixel image that was returned by the request that sent data into Omniture servers from affecting page layout in any way. However, this is no longer a concern, as the SiteCatalyst code now creates an image object—a non-visible image that doesn't show up on your page at all. Putting the code in the document head also means that the code appears earlier in the code, allowing it to execute sooner, which is significant inasmuch as it allows you to count page views for partial page loads more effectively.

However, certain elements of the code require the existence of the body object. Since Web browsers execute code in the order they receive it, if the SiteCatalyst JavaScript code is in the document head, then it executes before the body object exists. As a result, your implementation will not collect ClickMap data, and automatic tracking of file downloads or exit links will not be available. You also will not receive connection type data or visitor home page data. Technically, putting the code in the document head will work, but the result will be a very limited version of SiteCatalyst, and users may wonder why certain reports and tools, including ClickMap, aren't recording data.

The SiteCatalyst code can be placed anywhere inside BODY tags (`<BODY></BODY>`) of a well-formed HTML page. Omniture recommends placing the code in a global include file at the top of the page (inside the HTML body tag), however the code can be place anywhere on the page, except as noted below.

- If placed within a table, post the code only within the `<td></td>` tags. For example, do not place the code between an opening <tr> tag and an opening <td> tag.
- The code that sets the variables must occur after the reference to the s_code.js file.
- Make certain that the report suite IDs in the s_account variable in the s_code.js file are set correctly. This variable will typically be set correctly when downloading code from the Code Manager for a particular report suite, or as supplied by an Omniture Technical Consultant.

The following example shows correct placement of the SiteCatalyst code.

```
<html>
<head></head>
<body>
<!-- SiteCatalyst code version: H.20.3.
Copyright 1997-2009 Omniture, Inc. More info available at
http://www.omniture.com -->
<script language="JavaScript" type="text/javascript"
         src="http://www.yourdomain.com/js/s_code.js"></script>
<script language="JavaScript" type="text/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=""
s.prop4=""
s.prop5=""
/* Conversion Variables */
s.campaign=""
s.state=""
s.zip=""
s.events=""
s.products=""
s.purchaseID=""
s.eVar1=""
s.eVar2=""
s.eVar3=""
s.eVar4=""
s.eVar5=""
/************* DO NOT ALTER ANYTHING BELOW THIS LINE ! **************/
var s_code=s.t();if(s_code)document.write(s_code)//--></script>
<!-- End SiteCatalyst code version: H.20.3. -->
</body>
</html>
```

1.2 Using s.linkTrackVars and s.linkTrackEvents

The key to a successful link tracking implementation is an understanding of the s.linkTrackVars and s.linkTrackEvents variables, which allow you to pass custom variable values on these user actions. If you are implementing custom link tracking and will be setting custom variables and events, make sure that your s.linkTrackVars variable contains a comma-separated list of all variables that you will be passing, *including the events variable*. Make sure that s.linkTrackEvents includes a comma-separated list of all events that you will be passing.

Note that setting s.linkTrackVars and s.linkTrackEvents does not actually set these variables/events; it only prepares the SiteCatalyst code to do so. You still need to set the variables manually, as shown in the example below.

```
<script language="javascript">
function customLinks () {
         var s=s_gi('myreportsuite');
         s.linkTrackVars="prop1,eVar7,products,events";
         s.linkTrackEvents="event5,event9";
         s.prop1="Link Click";
         s.eVar7="my_page.html";
         s.events="event5";
         s.tl(true,'o','Custom Link Click');
}
</script>
<a href="my_page.html" onclick="customLinks();">My Page</a>
```

Notice that “events” is listed in the s.linkTrackVars variable; the individual events that may be passed are included in the s.linkTrackEvents variable and are also included within s.events later in the function. Each of the variables that are passed are listed in s.linkTrackVars before they are populated later in the function. Also, as described above, “event9″ has been included in s.linkTrackEvents, but has *not* been included it in s.events. It will not be recorded, but could be recorded if the user had chosen to set s.events=”event5,event9.”

Note that automatic file download and exit link tracking work differently. that the s.linkTrackVars and s.linkTrackEvents are included in the standard s_code.js file, and both are set to “none.” The reason these variables are typically left set to “none” in the s_code.js file is that automatic link tracking, unlike custom link tracking, will use the s.linkTrackVars and s.linkTrackEvents values that you set in the global JavaScript file and will pass whatever the existing values of those variables are whenever a file download or exit link is recorded.

For example, consider an example where s.channel="Home" when the page loads, and where s.linkTrackVars="channel" in your s_code.js file. If a user clicks to download a file, automatic file download tracking will pass data into SiteCatalyst, including the value of s.channel that was set on page load. "Home" will be passed a second time, leading to inflation in page view data for this value in the Site Sections report.

Thus, Omniture strongly recommends leaving the s.linkTrackVars and s.linkTrackEvents set to "none" in the global JavaScript file and setting them explicitly as necessary with your custom link tracking implementation.

1.3 Common Mistakes in the Products Variable

The s.products variable may be the most syntactically complex variable that SiteCatalyst offers. Commas, semi-colons, pipes, and equals signs all play specific roles in the variable. It has no overall maximum length, but each individual product entry cannot be longer than 100 bytes (including multi-byte characters). Mistakes in implementation of this variable are understandable, but unfortunately for developers, s.products is often a site's most important variable because it makes possible the tracking of revenue, units, product names, etc.

This variable is explained in detail in the *SiteCatalyst Implementation Manual* and the online Knowledge Base, but here are a few extremely easy-to-make mistakes that can wreak havoc on any implementation.

Make sure that your category, product name, and revenue totals are devoid of commas and semi-colons. The comma is used to separate entries in the s.products string, as happens when you have two products in the same transaction; the semi-colon is used to delimit fields within an entry. If you use a comma or semi-colon in any other way, SiteCatalyst will assume that you are separating product entries. Consider the following example.

```
s.products="widgets;large widget, 40'x40';1;19.99,wugs;tiny wug;2;1,999.98";
```

In this implementation, the developer probably intended for SiteCatalyst to read this as shown below.

Category 1: widgets
Product 1: large widget, 40'x40'
Units 1: 1
Revenue 1: 19.99
Category 2: wugs
Product 2: tiny wug
Units 2: 2
Revenue 2: 1,999.98

However, note the commas in the "Product 1" and "Revenue 2" entries. These indicate a new product entry. SiteCatalyst would actually interpret the above as:

Category 1: widgets
Product 1: large widget
Category 2: 40'x40'
Product 2: 1
Units 2: 19.99
Category 3: wugs
Product 3: tiny wug
Units 3: 2
Revenue 3: 1
Category 4: 999.98

The result of a mistake such as this is often that unexpected numerical values display in the Products report, as a result of the units field being recorded as the product name. If you see invalid product names in your Products report, review your s.products variable implementation for misuse of reserved characters such as the comma.

Additionally, note that that your product and category names should not contain unsupported characters. This can be especially difficult in the s.products string because product names are often likely to contain characters such as ™, ©, and ®. These characters will need to be stripped out of the product and category values before they are placed into s.products. You will also need to ensure that currency symbols are not included in your revenue values.

NOTE: Supported characters are numbers 1-127 from the ASCII table.

Finally, if you are not passing a product category in the product string, make sure to include a semi-colon (;) where the product category is normally displayed, as shown below.

```
s.products=";product name"
```

In this case, the semi-colon represents a placeholder for the product category. If the semi-colon is left out of the product string, then "product name" would be counted as the category, the number of units to be counted as the product name, the revenue to be counted as the units, as so on.

1.4 Setting the pageType Variable Correctly

The pageType variable is used only to designate a 404 (Page Not Found) error page. It has only one possible value, which is "errorPage."

```
pageType="errorPage"
```

On a 404 error page, the pageName variable should not be populated. The pageType variable should be set ONLY on a designated 404 error page. In other words, any page containing content should never have a value in the pageType variable. For pages containing content, you can set the variable as shown below.

```
pageType=""
```

However, the Omniture best practice is to delete the variable completely from pages containing content. This practice is recommended to avoid confusion regarding the purpose of the variable.

1.5 Using White Space in Variable Values

In HTML there are several characters that create whitespace, including a space, a tab, and a carriage return (or linefeed).Consider the following example.

```
<head>
         <title>
           Home Page
         </title>
</head>
<body>
<script language="javascript";
         s.pageName=document.title
</script>
```

In this case, document.title populates s.pageName, which should receive a value of "Home Page." However, you'll notice the space before "Home Page." Not all browsers will interpret this white space in the same way. The result may be either:

```
s.pageName="Home Page"
```

or

```
s.pageName="        Home Page"
```

Tthe first value displays correctly, but the second one displays white space before the text. SiteCatalyst will treat these as distinct values for the s.pageName variable, but the SiteCatalyst interface will strip the leading white space from the second value.The result is a report that displays as shown below.

	Page	Page Views	
1.	Home Page	10	76.9%
2.	Home Page	3	23.1%
	Total	13	

This implementation error will cause your variable values to be fragmented across multiple line items. Furthermore, SAINT does not allow leading white space in a key value, which means that it cannot be used to "group" multiple line items as a workaround if this issue is affecting your site. The only way to fix the problem is to pre-process the desired variable value (in this case, the document.title property) to remove any leading (or trailing) white space.

NOTE: The example above uses the s.pageName variable with the document.title property. Omniture does not recommend using document.title as the page name, nor does this issue only affect the s.pageName variable. Any variable that may have leading/trailing white space in its value can be affected.

1.6 Using Quotes

When you input values into a variable, there are a few best practices to follow. First, you can use either single quotes or double quotes. Just be sure you are consistent. If you are using single quotes, always use single quotes. If you are using double quotes, always use double quotes. Both of the following examples are correct.

```
s.prop2='test' (single quotes)
```

and

```
s.prop2="test" (double quotes)
```

NOTE: Make sure that you have smart quotes turned OFF.

Finally, if you are using single quotes, and you have an apostrophe in the variable value, JavaScript interprets the apostrophe as a single quote, which means it is the end of the string. For example, consider the following example.

```
s.pageName='John's Home Page'
```

In this case, JavaScript would interpret the apostrophe (which is also a single quote) in "John's" to be the end of the string. Since "s Home Page" has no meaning in JavaScript, it will cause an error that will prevent the image request from occurring. Instead, either of the following examples would work.

```
s.pageName='John\'s Home Page'
```

or

```
s.pageName="John's Home Page"
```

In the first example, you can *escape* the apostrophe by inserting a backslash (\) immediately before it, which tells JavaScript that the apostrophe is not ending the string, but rather is part of it. The string then ends as intended, at the closing single-quote. The second example uses double-quotes around the entire string, in which case a single-quote cannot indicate the end of the string.

NOTE: The same is true if a double-quote is part of the string. A value of s.pageName=”Team Says “We Win!”” would be incorrect because of the use of double-quotes within the string as well as surrounding it. This would be correct if entered as s.pageName=”Team Says \“We Win!\””.

1.7 Replacing your SiteCatalyst Code

Omniture offers some best practices for replacing your SiteCatalyst code. Frequently, customers will use the Omniture Code Manager to replace their code with the most recent version. This practice is good to follow as long as you keep certain things in mind, as described in the following sections.

1.7.1 Using the Incorrect Data Collection Server ID

Occasionally, generic s_code.js files that have not been generated from the Code Manager in the Omniture Suite are sent to those implementing the code on your site. As a result, the incorrect data collection server ID (as shown in the s.dc variable) for the account is used. The best practice in this case is to generate new code directly from the Code Manager for a specific report suite, rather than reusing code from a different report suite. This practice is the best way to make sure the s.dc variable is populated correctly.

1.7.2 Plug-ins

Some customers implement plug-ins to enhance their Omniture experience. When you replace your code, make sure you do not forget that you have plug-ins as part of that code. The code created in the Code Manager does not have any plug-in code with it, so all plug-ins will need to migrated manually to the newer code base. The best practice is to make a copy of your existing code just in case something happens, and you need to revert to your previous code.

1.8 Table of Common Syntax Mistakes

The following table shows the quick difference between correct and incorrect code mistakes.

Incorrect	Correct
prop1	s.prop1 (uses “s.”)
s.evar1	s.eVar1 (uses correct capitalization)
s.pagetype='errorpage';	s.pageType='errorPage'; (uses correct capitalization)
s.tl(this,o,pageA)	s.tl(this,'o','pageA'); (correct usage of single quotes)
s.events='event1'; s.events='event2';	s.events='event1,event2'; (correct format)
s.pageName=”John” (smart quotes on)	s.pageName="John" (smart quotes off)
s.products=”shoes,UX4879,1,19.99”	s.products=”shoes;UX4879;1;19.99” (uses commas, not semicolons)
s.products=”;MacBook Air;1;$1999.99”	s.products=”;MacBook Air;1;1999.99” (does not use dollar signs)
s.products=";Nikon SB-600 Speedlight Flash for Nikon Digital SLR Cameras;1;229.9489183”	s.products=";Nikon SB-600 Speedlight Flash for Nikon Digital SLR Cameras;1;229.95” (round or truncate long prices)

var s_account=”rsid1, rsid2”	var s_account=”rsid1,rsid2” (no space between report suite IDs when doing multi-suite tagging)
s.events=”event1, event2”	s.events=”event1,event2” (no space between event IDs when doing multi-suite tagging)
s.products="product name"	s.products=”;product name” (semicolon used when no product category is listed)

1.9 Additional Notes

In addition to the potential problems explained in the previous section, be sure to watch for the following items.

- Keep the closing HTML comment at the very end of the Omniture code (even if you are using the NOSCRIPT part of the script).

.

OMNITURE™

CALL 1.877.722.7088
1.801.722.0139

www.omniture.com
info@omniture.com

550 East Timpanogos Circle
Orem, Utah 84097

OMNITURE™