OMNITURE™

WHITE PAPER

OMNITURE ACTIONSOURCE

ActionScript Tracking for Flash

October 23, 2006 Version 1.1

CONTENTS

PREFACE

Omniture ActionSource™ is a new patent-pending method for natively tracking Flash application used with the ActionScript programming language (used by applications like Adobe Flash, Flex, and others). This solution removes dependencies on JavaScript for Flash tracking through the use of native ActionScript analytics.

Product Overview

SiteCatalyst™ delivers a powerful, intuitive, and industry-leading user interface, which enables users to quickly access actionable information. SiteCatalyst™ is the only solution that provides access to Traffic, Path, Campaign, Commerce, Segmentation, and Data Warehousing all in one interface. The easy-to-use approach allows Omniture customers of all skill levels to access actionable, real-time reports and dashboards. SiteCatalyst is an enterprise-grade solution designed for rapid user adoption across the organization.

SiteCatalyst™ comes with a comprehensive set of out-of-the-box reports that can be further tailored to suit special needs. Unique to SiteCatalyst™ is the ability to move between reports in an interactive fashion, allowing the user to keep the analysis momentum. SiteCatalyst is the only web analytics product that provides a comprehensive view with real-time reporting as well as a full data warehouse. SiteCatalyst provides superior scalability and reliability and is designed to handle massive traffic spikes. SiteCatalyst can report web site analysis data for clients from different business models and/or industries, whether the client has a smaller online client base or a billion page views per month.

Document Conventions

To ensure that this document is as easy to read and understand as possible, the following conventions may be used throughout.

NOTE: A Note distinguishes helpful information of which you may not be aware.

TIP: A Tip adds insightful information that may help you use the system better.

CAUTION: A Caution makes you aware of the results of an action. The results may not necessarily be damaging, but they definitely are important.

WARNING!: A Warning highlights an action that may result in damage to your system or an unforeseen loss of data.

Additional References/Resources

In addition to this document, Omniture offers white papers for several other topics. The white papers are available in PDF format. In order to view them, you must first have Adobe® Reader® installed on your system. Reader is a free download from Adobe at http://www.adobe.com. For more information on the available white papers, contact Omniture Live Support.

About Omniture

Omniture, Inc., headquartered in Orem, Utah, is the pioneer of next-generation online analytics technology that delivers the essential intelligence needed by Web commerce leaders and innovators to drive the business decisions that increase ROI. Omniture is the largest, ASP based, online analytics provider by revenue, and Omniture's SiteCatalyst is the most mature and comprehensive technology on the market, offering industry leading scalability and flexibility combined with an intuitive user interface. Omniture is the only company in its market to offer a comprehensive view of activity on a company's Web site that includes historical (data warehouse) and real-time analysis and reporting. In addition, Omniture offers knowledgeable professional service teams, experienced in helping customers determine the questions they must ask to arrive at the answers they require. Proof of its world-class technology and outstanding team, Omniture has the highest level of retained and satisfied customers in the market, including eBay, AOL, Wal-Mart, Gannett, Microsoft, Oracle, Overstock.com, GM and Hewlett-Packard. www.omniture.com.

1 ActionSource

Omniture ActionSource™ is a new patent-pending method for natively tracking Flash applications used with the ActionScript programming language (used by applications like Adobe Flash, Flex, and others). This solution removes dependencies on JavaScript for Flash tracking through the use of native ActionScript analytics.

Until now, tracking Flash use, while maintaining accurate visitor metrics with other web site activity, has been dependent on two programming languages: ActionScript and JavaScript. Communication between these technologies complicated the implementation process and introduced technical barriers. For Flash developers who did not understand the nuances of JavaScript, it has been complicated to put analytics solutions in place. Even for those who could deploy the JavaScript solution, limitations based on language and browser communication have caused performance and data integrity problems including the following.

- Cancellation of analytics when Flash attempted to communicate through the browser to JavaScript for analytics and to link out to another browser page
- Limitations of 508 characters per data transmission from Flash through the browser to JavaScript
- Playback delays for animation while JavaScript is executing
- Localized tracking only – JavaScript must be present on the same page as the Flash file to execute analytics tracking

Omniture's ActionSource™ was developed to simplify the implementation process and to improve performance and accuracy. Its native ActionScript engine removes the dependency on JavaScript and maximizes application portability and accuracy. Some of the key benefits of this solution are listed in the table below.

Table 1-A: Solution Benefits

Solution Benefit	**Description**
Omniture AutoTrack™ for automatic Flash analytics	An industry first. Omniture ActionSource can "listen" for appropriate click activity and automatically transmit appropriate information to Omniture for analysis. For some implementations, this can effectively eliminate the need to add analytics code to each click event in a Flash application, thereby saving development time and money.
ClickMap for Flash	Omniture's visual overlay, ClickMap, is now integrated into Flash tracking through ActionSource. Depending on the state of the application, ClickMap can run and display an overlay of where users have clicked, and how much each click has contributed to other activity within the site.
Familiar Programming Language	Using ActionScript, Flash developers can quickly employ the Omniture analytics engine in a familiar programming language.
Incredible Performance	With one programming language, Flash does not have to channel through browser actions and page level scripting. This allows for immediate code execution and high-speed data transactions.
Transparent Analytics	Some JavaScript commands can cause the browser to render a "click" sound and hesitate animation playback. ActionSource executes independent of JavaScript and is so efficient that tracking is "transparent" to the user experience.

Portability	Using native ActionScript technology, Omniture tracking can accompany Flash applications wherever they are accessed, even across domains and devices.
Accurate Visitor Metrics	Omniture ActionSource maintains unique visitor counts across technologies, even though it is capturing and transmitting metrics data independent of JavaScript. User privacy is upheld, and traditional cookies are used to maintain consistency with privacy advocates. This solution even supports first-party cookies to help users feel comfortable with the user experience and to reduce cookie blocking.
File Size	Flash applications (swf, exe, …) are compiled, and very efficient. The code for this solution is only about 4KB in size.
Larger Data Transmission Sizes	Internet Explorer limited the amount of data that could be passed from Flash to JavaScript. Only 508 characters could be sent through the browser. Omniture's native ActionScript solution is not dependent on communicating with JavaScript through the browser and is not subject to this browser limitation.
Accurate Link Tracking	Flash analytics through JavaScript requires communication through the browser. Executing a link from Flash also requires communication through the browser. When trying to track both at the same time (tracking the click of a button that redirects the browser is a very common Flash activity), the browser will often drop one of the commands. ActionSource solves this problem by eliminating the need to use the browser (JavaScript) for analytics.
Flash Analytics Debugging	Omniture clients now have the ability to run a specialized debugger on Flash files to understand what information is being transmitted from the application.

1.1 When to use Omniture ActionSource vs. JavaScript

Omniture has been supporting Flash analytics through JavaScript for a number of years. This solution is excellent for those with a firm understanding of JavaScript and for those who have already deployed this solution. However, this method of tracking Flash does not provide ClickMap support within the Flash application. As such, to view ClickMap reports for your site, ActionSource should be used.

If a Flash application is using JavaScript effectively, there is no need to switch to the ActionSource solution. However, if any of the above listed issues with the JavaScript solution (currently used by all major analytics vendors) have caused development, performance, or data integrity problems, Omniture recommends the use of Omniture ActionSource.

2 Implementing ActionSource

Omniture ActionSource was designed for rapid implementation. It is currently available in a standard format for “centralized” distribution. This means that the Omniture ActionSource “engine” is centralized and independent of the files it supports. The benefit of this centralized solution is that if Omniture makes updates to this solution, there is only one file that needs to be updated. Each Flash application that needs to be tracked simply includes a reference to this centralized file and the Omniture ActionSource engine is included at runtime as if it were part of the application. This option is ideal for ActionScript applications where files are distributed from a central file location and the application can reference the ActionSource engine.

A customized solution (not recommended for most companies) can directly embed the Omniture ActionSource engine into an application. This solution is ideal for files that may be decentralized for maximum portability. If your ActionScript application will float around, independent of a centralized file system, this option may be the best for you. The downside is versioning. If Omniture releases a new version of this solution, Flash applications with embedded Omniture ActionSource will need to be recompiled to leverage the latest code version.

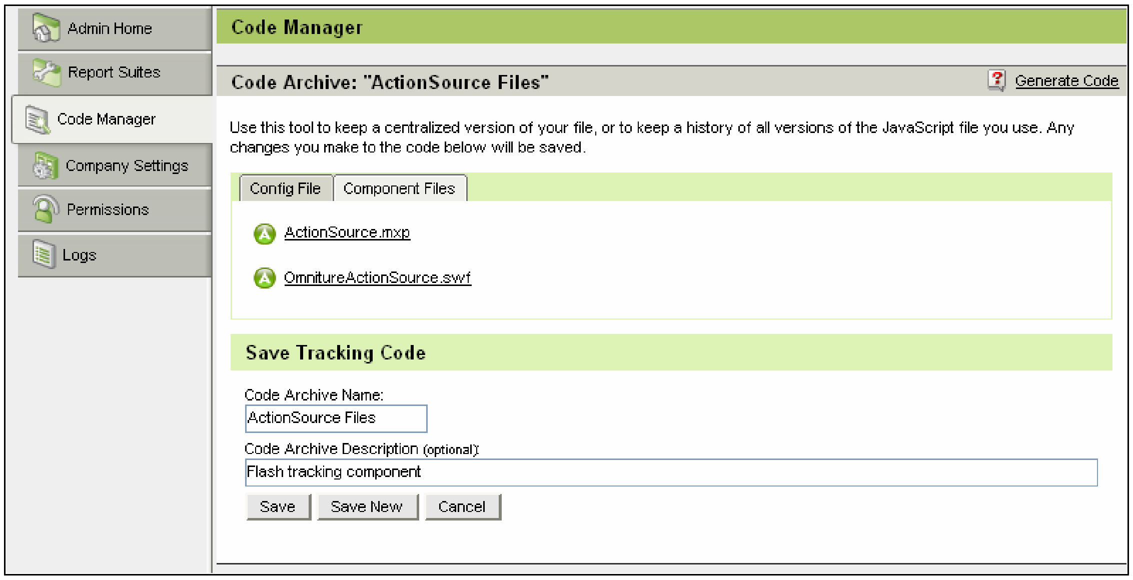
2.1 Centralized Code

The standard and most easily maintained implementation of Omniture ActionSource is through the centralized option. With this approach, the Omniture ActionSource engine is referenced by an ActionScript application, but is not embedded as part of the compiled application file. Instead, it is included at runtime as an independent file and acts as if it were part of the compiled application. The benefit to this approach is that future enhancements to Omniture ActionSource can be provided to previously compiled files through the simple update of the Omniture ActionSource engine. This means that only one file needs to be updated instead of every Flash application that includes the solution.

2.2 Installing the Omniture ActionSource Component

The first step to implementing this solution is to obtain the ActionSource engine (a compiled Flash file “OmnitureActionSource.swf”) and the appropriate Omniture ActionSource extension to the Flash authoring environment. These files are available in the SiteCatalyst Administration Console Code Manager.

Figure 2-A: Action Source Extension and Engine in the Administration Console



Code Archive: "ActionSource Files"

Use this tool to keep a centralized version of your file, or to keep a history of all versions of the JavaScript file you use. Any changes you make to the code below will be saved.

ActionSource.mxp

OmnitureActionSource.swf

Save Tracking Code

Code Archive Name: ActionSource Files

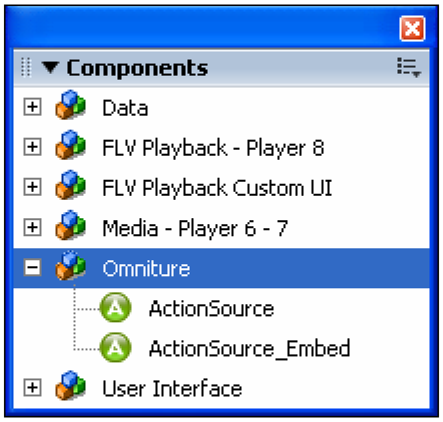
Code Archive Description (optional): Flash tracking component

To add the ActionSource component to your Flash development environment, install the Omniture ActionSource extension “ActionSource.mxp” by double-clicking it. The components are Flash version MX 2004 (version 7) specific for the authoring environment, but support player compiled versions of Flash MX (version 6). This means that the components can be used when developing in Flash MX 2004 (version 7) and newer, and can be published for use with Flash MX (version 6) and newer. Any instances of the Flash authoring environment must be closed and reopened before the components will be available.

The ActionSource.mxp extension contains two components: 1) the standard (and recommended) ActionSource component, and 2) the non-standard embedded component. Both components will be installed with the extension. It is only necessary to use one component or the other. Do not use both components to track one application.

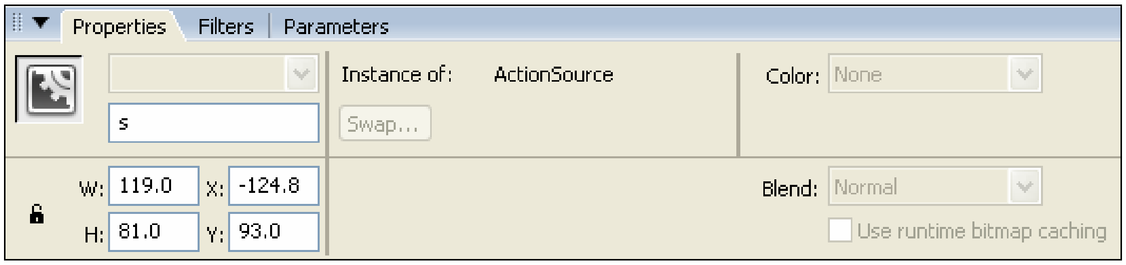
To select the appropriate component, consider future updates versus ultimate portability. The standard component will easily accommodate future updates by simply updating the "OmnitureActionSource.swf" file. The embedded component will compile an embedded copy of the Omniture ActionSource engine in your Flash file. This option creates a file that can be tracked no matter where the file is served from. With the standard ActionSource file, the "OmnitureActionSource.swf" file and the Flash application it supports need to be served from the same domain.

Figure 2-B: Components



2.3 Adding a Component Instance

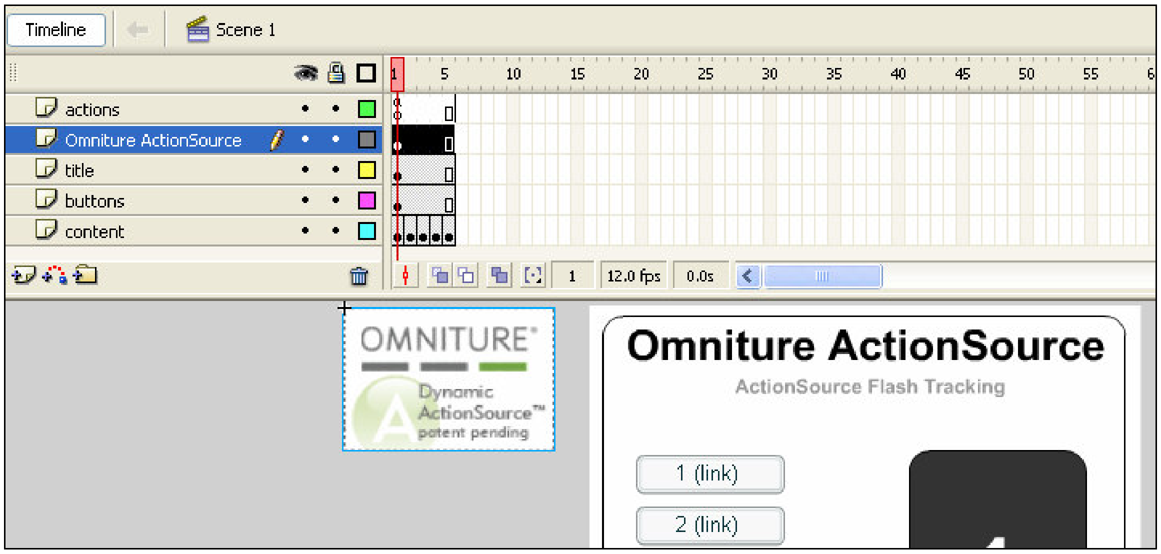
With the Omniture ActionSource component installed, (see above), simply open the component window and drag an instance of the ActionSource component onto the application's root timeline. Then give the component instance a name. The component can be named through the property tab when the component instance is selected (see below). It is recommended that the component instance be named "s", however it can have any name as long as it is referenced properly in the application code.



Instance name: s
Instance of: ActionSource
Color: None
W: 119.0 X: -124.8
H: 81.0 Y: 93.0
Blend: Normal

NOTE: Ensure that the component is on its own layer and that the layer and its frames span the entire application. If there are no layers, or is just one frame, just ensure that the component can be globally referenced.

Figure 2-C: Adding a Component Instance



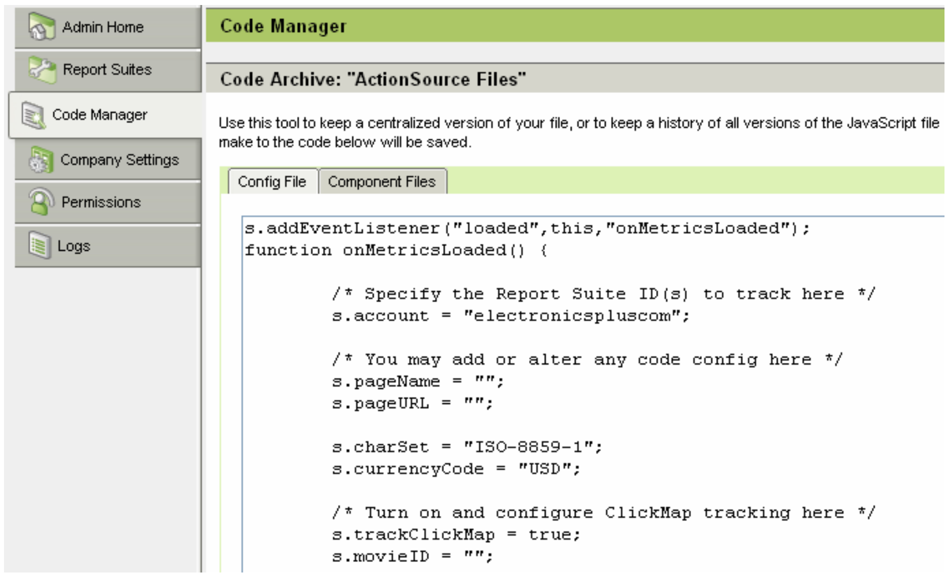
Flash includes some security controls that prohibit inclusion of files outside the domain where your Flash application is served from. This means that your Flash application and the Omniture ActionSource engine must be on the same domain in order to function under default settings. This does not limit where the Flash file can be served to; it just requires that files be served from the same location. It is possible to alter permissions for inclusion across domains, but is not recommended. Contact an Omniture Implementation Consultant if housing the Omniture ActionSource file on a separate domain from the Flash Application is required.

2.3.1 Configuring the Component

The following steps will configure the Omniture ActionSource component.

1. Add a layer to the root timeline with a keyframe that spans the application and makes the code globally available, or add it through an included ActionScript file (as with the component instance above).
2. Paste the Config File code from the Admin Console Code Manager into this keyframe's ActionScript. This code creates an event listener and sets required configuration variables.

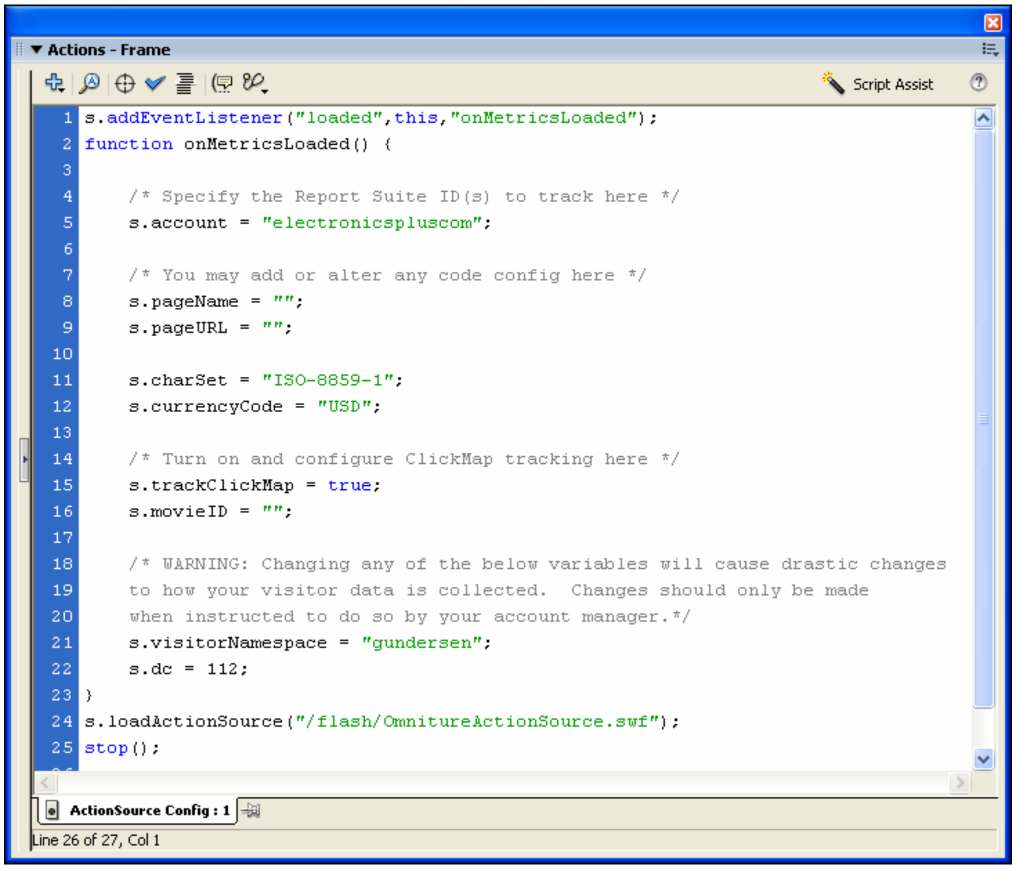
Figure 2-D: Config File Code Manager



Use this tool to keep a centralized version of your file, or to keep a history of all versions of the JavaScript file make to the code below will be saved.

```
s.addEventListener("loaded",this,"onMetricsLoaded");
function onMetricsLoaded() {

        /* Specify the Report Suite ID(s) to track here */
        s.account = "electronicspluscom";

        /* You may add or alter any code config here */
        s.pageName = "";
        s.pageURL = "";

        s.charSet = "ISO-8859-1";
        s.currencyCode = "USD";

        /* Turn on and configure ClickMap tracking here */
        s.trackClickMap = true;
        s.movieID = "";
```

Figure 2-E: Event Listener



```
s.addEventListener("loaded",this,"onMetricsLoaded");
function onMetricsLoaded() {

    /* Specify the Report Suite ID(s) to track here */
    s.account = "electronicspluscom";

    /* You may add or alter any code config here */
    s.pageName = "";
    s.pageURL = "";

    s.charSet = "ISO-8859-1";
    s.currencyCode = "USD";

    /* Turn on and configure ClickMap tracking here */
    s.trackClickMap = true;
    s.movieID = "";

    /* WARNING: Changing any of the below variables will cause drastic changes
    to how your visitor data is collected.  Changes should only be made
    when instructed to do so by your account manager.*/
    s.visitorNamespace = "gundersen";
    s.dc = 112;
}
s.loadActionSource("/flash/OmnitureActionSource.swf");
stop();
```

3. Check that the loadActionSource function call references the OmnitureActionSource.swf file appropriately on your servers.

4. Within the onLoad event, insure that configuration and tracking variables are given the appropriate values.

Table 2-A: Configuration Variables

Variable	Description
Required	
s.account = "mycompanycom";	The report suite or report suites (multi-suite tagging) that you wish to track.
s.dc = 112; or 122;	Used to identify collection servers. Do not change this value from the one provided by the Code Manager in the Administration Console.
s.visitorNameSpace = "mycompany";	**Must match the visitorNamespace used in your JavaScript tracking file (s_code.js).** If your JavaScript file does not have visitorNamespace defined, do not define it in the ActionSource component. Using the wrong value for visitorNamespace will cause incorrect

	reporting.
Optional	
s.autoTrack = true;	Set to true to turn on automatic ClickMap tracking with each click.
s.charSet = "UTF-8";	Set the Character Encoding used in the Flash component (refer to the *SiteCatalyst Implementation Manual*).
s.cookieDomainPeriods = 2; or 3;	Number of sections or periods to use when setting Omniture's visitor ID cookie. For example, if s.trackingServer contains ".co.uk" it is 3, if it contains ".com" it is 2. Refer to the *Omniture Implementation Manual* for more details.
s.cookieLifetime = "SESSION";	The number of seconds Omniture's visitor ID cookie should persist or "SESSION" to have the cookie expire with a browser session (very rare).
s.currencyCode = "USD";	Set the Currency Code - use purchases or currency events are collected in the Flash file.
s.delayTracking = 500;	Set to the number of milliseconds to delay the initial image request sent from Flash, based on the load time of the Flash movie. When the web page housing the Flash file also uses Omniture's JavaScript file, this should be set to avoid the image requests overwriting cookies. Recommended delay for pages with JavaScript tracking is 500.
s.linkLeaveQueryString = true;	Set to true to leave the query-string on the URLs tracked for links. If query strings contain session IDs or other unique information, talk to an Omniture representative to have those query string parameters removed by SiteCatalyst.
s.movieID = "my_movie_id"	This is used for ClickMap to uniquely identify clickable objects. Click data sent to Omniture consists of the movieID:instance_name of the clickable object. If movieID does not exist, ActionSource will use the filename from the embed (Firefox) or object (IE) tag. This is typically the same value as the ID attribute of the object tag.
s.trackClickMap = true;	Set to true to send ClickMap data when using the track and tracklink functions.
s.trackingServer = "metrics.mycompany.com";	If you are using a first-party cookie implementation, these

s.trackingServerSecure = “smetrics.mycompany.com”;	variables are used to identify the collection domain. They should match the collection domain used in your JavaScript file, if one exists.
s.vmk = ”[VALUE_PROVIDED_BY_OMNITURE]”	Migration key used to migrate visitor cookies from third to first party. An Omniture representative must provide this value.
Debugging	
s.debugTracking = true;	Set to true to view debug in the flash editor trace window.
s.trackLocal = true;	Set to true to try and track movies on your local machine instead of a webserver in flash 8.

The following tracking variables are supported in the ActionSource component. The functions of these variables are described in detail in the *Omniture Implementation Manual*.

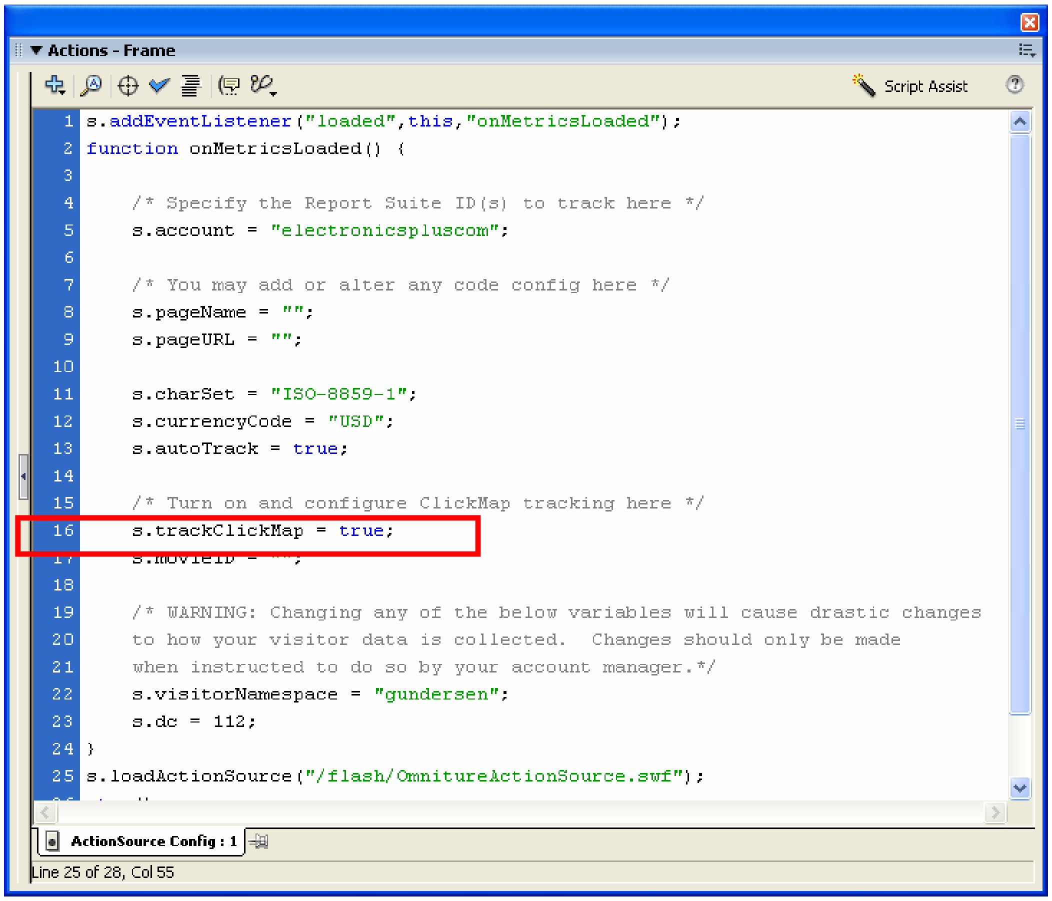
- s.pageName
- s.pageURL
- s.referrer
- s.purchaseID
- s.channel
- s.server
- s.pageType
- s.campaign
- s.state
- s.zip
- s.events
- s.products
- s.prop1-50
- s.eVar1-50
- s.hier1-5

2.4 AutoTrack **OR** Event Scripting

No analytics will be recorded without Flash events triggering the Omniture ActionSource engine to compile and send data to Omniture. For some users, just getting a ClickMap overlay and understanding which buttons were clicked is enough. In this case, autoTrack may be used to listen for click activity and immediately send data for every qualifying click. If the click is related to a button or qualifying movie clip, the click will be recorded as a custom link with the name of the link being ActionSource.AutoTrack:movieID. You can enable this feature by setting s.autotrack to the Boolean value true.

When autoTrack is enabled, any tracking variables set will be sent with every click. This is useful when you want to identify which Flash movies or Flash sections incite the most user interaction. By populating prop1, for example, you can report on Flash interaction by prop1 value. Remember that pageName is not sent with autoTrack.

Figure 2-F: AutoTrack



```
s.addEventListener("loaded",this,"onMetricsLoaded");
function onMetricsLoaded() {

    /* Specify the Report Suite ID(s) to track here */
    s.account = "electronicspluscom";

    /* You may add or alter any code config here */
    s.pageName = "";
    s.pageURL = "";

    s.charSet = "ISO-8859-1";
    s.currencyCode = "USD";
    s.autoTrack = true;

    /* Turn on and configure ClickMap tracking here */
    s.trackClickMap = true;
    s.movieID = "";

    /* WARNING: Changing any of the below variables will cause drastic changes
    to how your visitor data is collected.  Changes should only be made
    when instructed to do so by your account manager.*/
    s.visitorNamespace = "gundersen";
    s.dc = 112;
}
s.loadActionSource("/flash/OmnitureActionSource.swf");
```

WARNING: Using AutoTrack may cause a significant increase in server call volume because data is sent to Omniture anytime a user clicks on a button.

NOTE: Any tracking variables with values will be sent with every autoTrack call, with the exception of pageName. Every click will increase the number of times those variables are sent to SiteCatalyst.

2.4.1 Event Scripting

In most cases, specific features or functionality should be tracked, but not every button click. If this is the case, or if the data needs to be customized beyond classifications, it is necessary to assign a small script to each event to be tracked. This small script will transmit the included information about the event to Omniture for reporting.

The format of this customized script is to execute one of two functions within the Omniture ActionSource engine. track() will send a page view to Omniture, and trackLink() will send a custom link to Omniture. By setting tracking variables prior to calling these functions, you can send specific data to Omniture servers. The following sections offer an explanation how these methods should be used.

2.4.2 Functions Available to the ActionSource Component

Table 2-B: ActionSource Methods

Method	Description
s.track()	Used to send a standard page view to SiteCatalyst servers. Any tracking variables with values will be sent.
s.trackLink(url,type,name)	Used to send custom, download or exit link data to SiteCatalyst servers. The pageName variable is not recorded when trackLink is used, but all other variables are passed in. The parameters are defined as follows: **url**: The URL that identifies the link clicked. If no URL is specified, the name is used. It is recommended you use _root._url to identify the link URL. **type**: A letter identifying which link report the URL or name will be displayed in. Possible values are “o” (Custom Links), “d” (File Downloads) and “e” (Exit Links) **name**: The name that will appear in the link report. If no name is specified, the URL is used. name *or* url are required for data to be collected. An empty string should be used when one should be empty.

2.4.3 Page View using “track()”

```
on (click) {
   gotoAndPlay(1);
   // set variables
   s.track();
}
```

Example 1: Send a Page View Only

```
on (click) {
   gotoAndPlay(1);
   s.pageName="My Flash Media Player";
   s.channel="";
   s.prop1="";
   s.track();
}
```

Example 2: Send a Page View with Channel Information

```
on (click) {
   gotoAndPlay(1);
   s.channel="Hip-Hop";
   s.pageName="My Flash Media Player";
   s.track();
}
```

Example 3: Send a Page View with Channel and Custom Insight (prop) Information

```
on (click){
   gotoAndPlay(1);
   s.channel="Hip-Hop";
   s.prop1="Snoop";
   s.pageName="My Flash Media Player";
   s.track();
}
```

2.4.4 Custom Link (Application Activity) using “trackLink()”

Although the word “link” is used with this option, it may be used to denote “activity”. All micro level activity, which should not capture a page view, should be tracked as a custom link using the trackLink() function.

```
on (click) {
   gotoAndPlay(1);
   s.trackLink(url,type,name);
}
```

Example 1: Send a Custom Link

```
on (click) {
   gotoAndPlay(1);
   s.trackLink(_root._url,"o","Hip-Hop link");
}
```

Example 2: Send a Custom Link with Channel Information

```
on (click) {
   gotoAndPlay(1);
   s.channel="Hip-Hop";
   s.trackLink(_root._url,"o","Buy Hip-Hop link");
}
```

Example 3: Send a Download Link with Channel and Custom Insight (prop) Information

```
on (click) {
   gotoAndPlay(1);
   s.channel="Hip-Hop";
   s.prop1="Snoop";
   s.trackLink("http:www.mysite.com/downloads/filea.mp3","d","");
}
```

Example 4: Send an Exit Link with no variables

```
on (click) {
   gotoAndPlay(1);
   // clear out previously used variables
   s.channel="";
   s.prop1="";
   s.trackLink("http://www.somesite.com","e","Affiliate Exit Link");
}
```

2.5 Implementing the Non-Standard Embedded Option

To use the embedded version of the Omniture ActionSource component, simply open the component window and drag an instance of the Omniture ActionSource_Embed component onto the application's root timeline. Please ensure that the component is on its own layer and that the layer and its frames span the entire application.

2.6 ClickMap Support

2.6.1 Aggregating or Separating ClickMap Data

ClickMap uses the movie ID and instance name of objects to uniquely identify them. Once identified, each object is compared with data stored in SiteCatalyst reports. Objects are then overlaid based on the SiteCatalyst database.

In order to aggregate ClickMap data for objects in separate movies, you can make the movieID and instance name of the objects identical. For example, if you'd like to see all clicks on a "download now" button aggregated from all movies on your site, make sure the "download now" button has the same instance ID in every movie, and the movie ID is identical. Make sure that instance names only match for objects whose clicks should be aggregated in ClickMap reports.

Separating data from different versions of a page works similarly. For example, if one button appears on two separate frames, create separate buttons, one for each frame. Because the buttons will now have unique instance names, those will be reported separately in ClickMap.

2.6.2 HTML Object Tag <Object>

ClickMap uses the object tag to determine which part of the HTML page to look for ClickMap data for Flash. It identifies the appropriate object through the "id"/"name" parameter. Please be sure that the ID is the same in the HTML object tag as it is within the Flash configuration. This can be done dynamically as described in *Dynamic Object Tag* section below. The basic format of the object tag, along with its required parameters, is listed as follows.

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.c
ab#version=6,0,0,0" width="310" height="240" id="SOME_MOVIE" align="middle">

     <param name="movie" value="SOME_MOVIE.swf" />
     <param name="quality" value="high" />
```

```
    <param name="bgcolor" value="#ffffff" />
    <param name="wmode" value="transparent" />
    <param name="allowScriptAccess" value="sameDomain" />

    <embed src="SOME_MOVIE.swf" quality="high" bgcolor="#ffffff" width="310"
    height="240" name="SOME_MOVIE" swLiveConnect="true" wmode="transparent"
    align="middle" allowScriptAccess="sameDomain" type="application/x-
    shockwave-flash"
    pluginspage="http://www.macromedia.com/go/getflashplayer" />

</object>
```

NOTE: <param name="wmode" value="transparent" /> and wmode="transparent" are only required for ClickMap. If this parameter interferes with normal production, it can be removed. However, for those who need to run ClickMap on Flash, they will need to have this parameter included on their page version. This is often accomplished by altering the HTML based on query string parameter or IP address.

2.7 Tracking Flash Application Loads (Caution!)

ActionSource can track application loads very effectively. This can be an excellent means of understanding ad/app impressions, even on other sites.

CAUTION - When tracking loads, or successive actions for the first analytics transmission for users, it is **important** to be cautious of accidentally inflating unique visitor counts.

When can this happen?

Unique visitor counts can be inflated when a visitor comes to a page with Omniture's JavaScript sending data as the page loads, and ActionSource also sending data as the page loads.

Why does this happen?

The first data transmission for a user to an Omniture report suite sets a cookie on the user's machine. The cookie is used to uniquely identify the user as a "Unique Visitor". This transmission and cookie setting process can take up to 1 second for users with slow Internet connections. In most cases it happens in less than 500 milliseconds. After the cookie is set, transactions can fire as quickly as necessary without any conflicts, and unique visitor counts remain accurate. However, in the short time the cookie is being set, any subsequent transactions (while the cookie is still not yet set), will result in a new cookie overriding the previously set cookie. Each time a cookie is set, a user is identified as a new visitor. If the initial transmissions are so close to each other that each transmits before the visitor cookie is set, this will result in inflated unique visitor counts. Again, once the cookie has been set, on the initial data transmission, an application can send through sub-second data transmissions and unique visitor metrics will remain accurate.

How to test if a conflict has occurred and one user is being counted as two-

In a controlled environment, where no other users can send data to a report suite in the same day, clear browser cookies and run through the application ONE time, or if multiple tests are required, keep an accurate count of tests. Then run the daily unique visitor's report for that calendar day and check the unique visitor count against tests.

I need to track loads, and it is possible an event could be triggered immediately, what can I do?

The delayTracking configuration variable identifies the amount of time that should elapse between the Flash object load time and the first image request sent. By setting delayTracking to 500, you will prevent any cookie overwriting. Note that this only applies to cases where both the JavaScript and ActionSource files are sending data as the page/Flash object loads.

2.8 Dynamic Object Tag <object>

All configuration variables can be set dynamically. This option provides the greatest flexibility. The following code should be included in a JavaScript file “s_flashObj.js” and included in the HTML page.

```
// JavaScript Document
var s_movieName = "";
function Flash_embedSWF(srcURL,srcID,swfbgColor,sW,sH,sfv) {

     s_movieName = srcID;
     var defaultColor = (document.bgColor != null) ? document.bgColor :
     "#ffffff";
     var bgcolor = (swfbgColor != null) ? swfbgColor : defaultColor;

     var objectStr = '<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-
     444553540000"' +
     'codebase="http://active.macromedia.com/flash2/cabs/swflash.cab#version=4
     ,0,0,0"' +
     'ID="' + srcID + '" WIDTH="' + sW + '" HEIGHT="' + sH + '">' +
     '<PARAM NAME="movie" VALUE="' + srcURL + '">' +
     '<PARAM NAME="allowScriptAccess" VALUE="sameDomain">' +
     '<PARAM NAME="quality" VALUE="high">' +
     '<PARAM NAME="wmode" VALUE="transparent">'+
     '<PARAM NAME=FlashVars VALUE="'+sfv+'&s_movieID='+srcID+'">'+
     '<PARAM NAME="autostart" VALUE="false">'+
     '<PARAM NAME="bgcolor" VALUE=' + bgcolor + '>' +
     '<embed src="' + srcURL + '" quality="high"' + 'bgcolor="' +
     bgcolor + '"' + 'width="' + sW + '" height="' + sH + '"
     wmode="transparent" swLiveConnect="true"
     FLASHVARS="'+sfv+'&s_movieID='+srcID+'"' +
     'type="application/x-shockwave-flash" NAME="' + srcID + '"' +
     'pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?P1_P
     rod_Version=ShockwaveFlash"></embed></OBJECT>';

     document.writeln(objectStr);
}
```

Then in an HTML page, include the s_FlashObj.js file and dynamically create the object by placing a command call to Flash_embedSWF() where the object should go. The Flash_embedSWF() function parameters are listed below.

```
Flash_embedSWF(srcURL,srcID,swfbgColor,sW,sH,sfv)
```

Table 2-C: Function Parameters

Function Parameter	Description
srcURL	The location of the .swf file
srcID	The object id for the .swf file (required for ClickMap)
swfbgColor	color of the .swf object background (default is #ffffff)
sW	Width
sH	Height

sfy	Any FlashVariables that need to be available to the Flash movie. This is where the dynamic setting of all configuration variables can take place. The inclusion of variables in this section "&" delimited, makes the variables and their values immediately available to the Flash application.

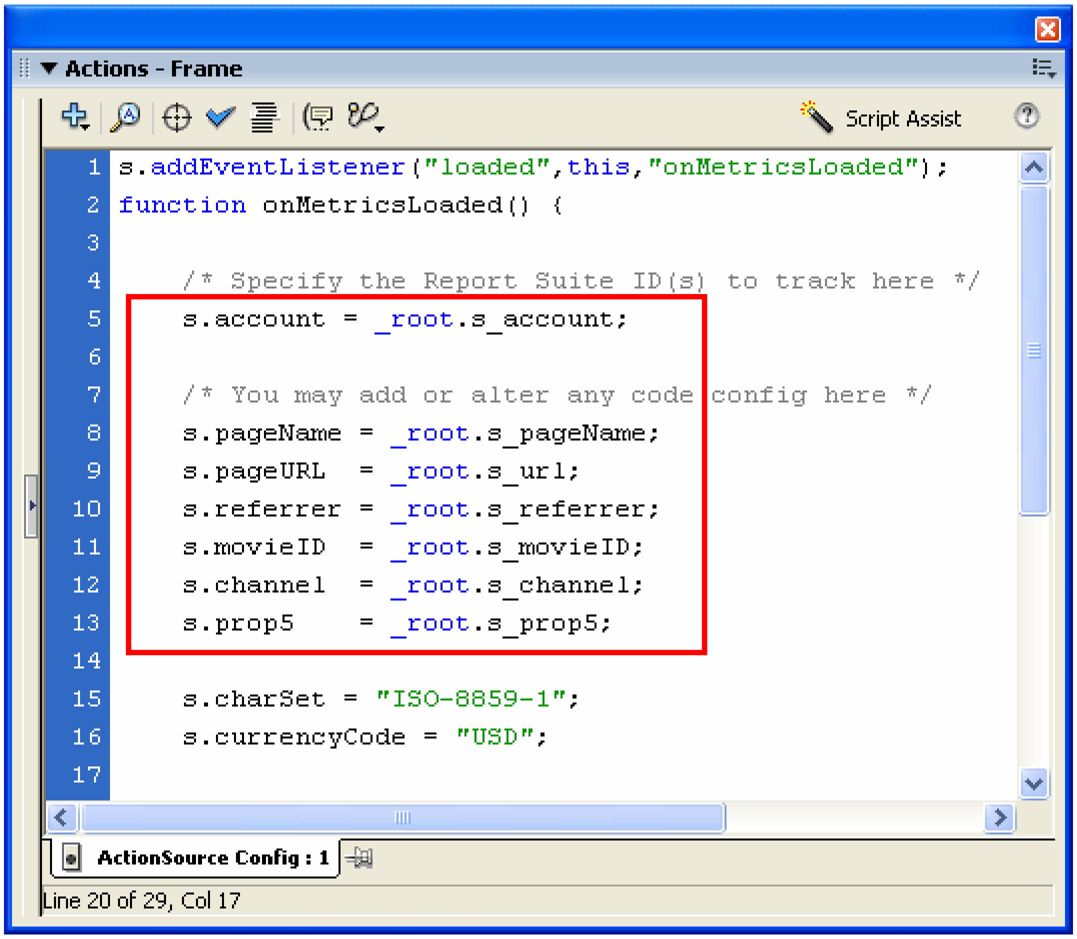
For example, notice the "sfv" parameter in the following code.

```
Flash_embedSWF("sample_dynamic.swf","sample_dynamic","#ffffff",310,240,"s_account=myex
amplerscom&s_url="+document.URL+"&s_referrer="+document.referrer+"&s_pageName=Sample
Dynamic&s_CName=112.2o7.net&s_nameSpace=devstephenh");
```

This will make "s_account" available to the Flash file with "myexamplerscom" as the report suite, "s_url" with is value available as the current browser URL, "s_referrer" with its value available as the current browser referrer, "s_pageName" with its value available as the current browser page name (for ClickMap), and "s_CName" and "s_nameSpace" with their values available for direction of the cookie settings.

Within the Flash application, it is just a matter of referencing these dynamic variables as shown below.

Figure 2-G: Dynamic Variables



```
s.addEventListener("loaded",this,"onMetricsLoaded");
function onMetricsLoaded() {

    /* Specify the Report Suite ID(s) to track here */
    s.account = _root.s_account;

    /* You may add or alter any code config here */
    s.pageName  = _root.s_pageName;
    s.pageURL   = _root.s_url;
    s.referrer  = _root.s_referrer;
    s.movieID   = _root.s_movieID;
    s.channel   = _root.s_channel;
    s.prop5     = _root.s_prop5;

    s.charSet = "ISO-8859-1";
    s.currencyCode = "USD";

```

In most cases, it is recommended to follow this dynamic object population approach. Whether through JavaScript, another script, or through server-side programming, dynamically configuring the Flash file through the object tag is a very powerful way to control your Flash implementation.

In the Appendix, some reports are only available with some customization. These reports can all be made available by using FlashVars and by dynamically setting the values in the Flash application.

CAUTION: s.referrer should be populated dynamically if a page is tracked with ActionSource only (no JavaScript tracking file). Flash does not have access to the page referrer, which is used to populate search engine, referrer and referring domains reports. You will not have search engine data if s.referrer is not populated. However, s.referrer should be emptied (set to “”) once it has been sent in once. Sending it in multiple times will cause reports based on the referrer to be inflated.

3 Quick Wins

3.1 Rich Internet Applications (RIA)

Rich Internet Applications (RIAs) are changing the face of the web. They bridge the gap between the promise of on-demand technologies for the masses and realistic user experiences. RIAs have been maturing for years. The biggest leaps forward have occurred with wide-scale adoption of browsers which support the underlying technologies that power these applications. While RIA has many forms and supporting technologies, the most common, and perhaps most widely adopted, are AJAX and Flash. It is important to understand that the technology is not what defines an RIA, but instead, its usability and application.

RIA looks cool, and is changing the web, but, should you use it on your site? This is an area where caution is important. Rich Internet Applications are expensive to develop. Don't jump in without using SiteCatalyst to test the water.

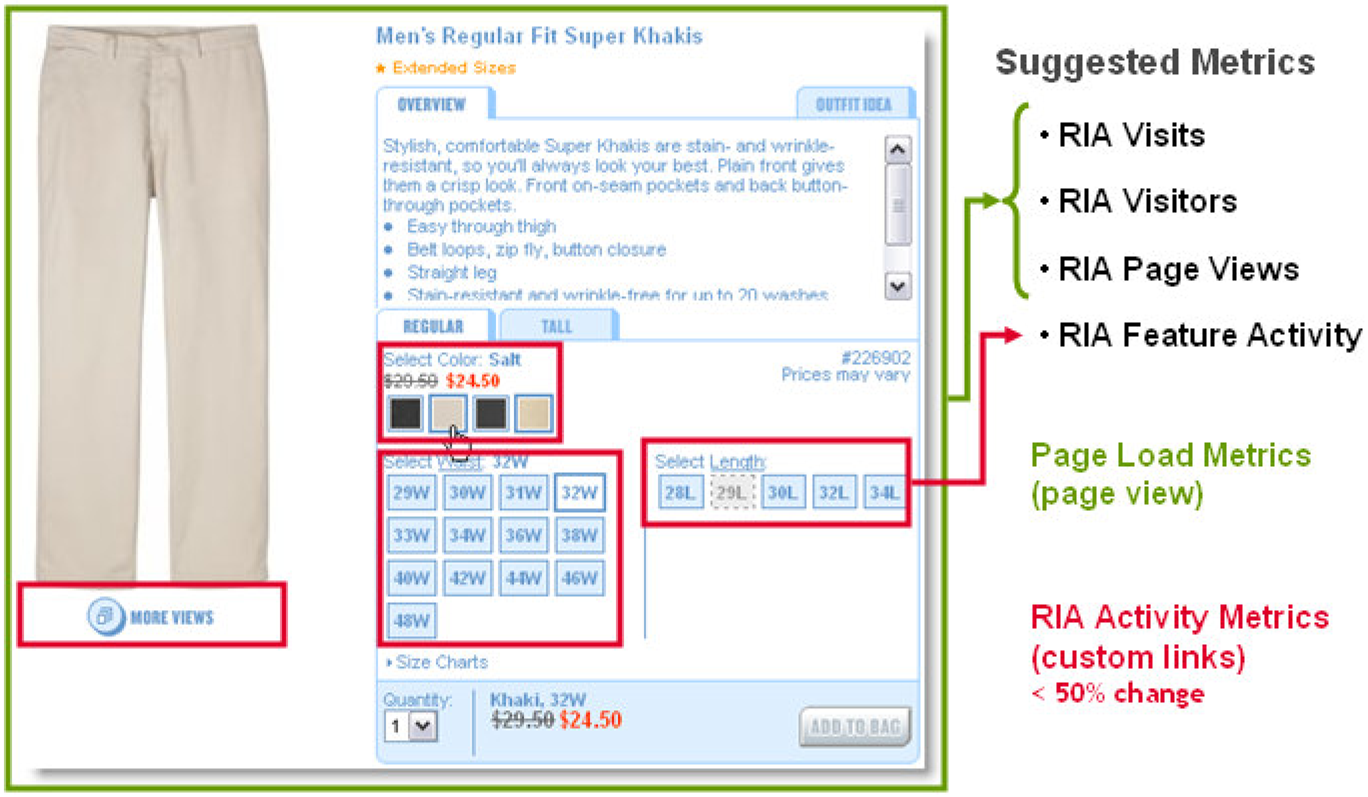
3.1.1 A/B Testing

Omniture strongly recommends that if possible, RIA is run in parallel to a standard and similar functionality. If this is not possible or simply doesn't make sense for the application, consider running a parallel version of the application with slightly different features. A/B testing provides valuable insight into efficiencies and application effectiveness. It also creates a fallback option if the RIA isn't working for you.

3.1.2 What to Measure

One of the most commonly asked questions with RIA is how to track micro-level activity separate from macro-level activity, and when it is appropriate to do either. For example, say you have an application that allows customers to uniquely configure a product. The application may have significant steps to which users are exposed. Are these steps considered page views? In addition, there are micro-level activities within each step. Should these activities be tracked as page views? What if you want to understand the flow between activities, or which features get the most activity? The trouble with RIA is that each application is unique. They are designed to mimic realistic actions, and since actions are relative to the situation, the possibilities are endless. However, most applications have a few major components: mile stone steps, features, and micro-level actions within features. So, while each application requires some consideration as to what specifically should be measured, there are some generalizations that can be applied to RIA tracking.

Figure 3-A: Example



Stylish, comfortable Super Khakis are stain- and wrinkle-resistant, so you'll always look your best. Plain front gives them a crisp look. Front on-seam pockets and back button-through pockets.

- Easy through thigh
- Belt loops, zip fly, button closure
- Straight leg
- Stain-resistant and wrinkle-free for up to 20 washes

3.1.3 Macro-Level Activity

Macro-level activity usually constitutes the loading of the application, which provides information on visits, visitors, instances, value to future actions, etc., but it can and should also represent major steps in the process. A good rule of thumb is that if an RIA action changes the application more than 50% (or whatever is considered significantly changing the user experience or content), then it is macro-level, and should be tracked as a page view.

3.1.4 Micro-Level Activity

Micro-level activity includes any changes less than 50% (or not considered as significantly changing the user experience or content). Toggling between color selections, for instance, would be considered micro-level activity. Omniture recommends that micro-level tracking be related to features. For example, in the case of toggling between colors, is it really important to understand which colors were considered? Or is it more important to know that the color selection feature was used? Perhaps both are important, and if so, capture both; but, when measuring the effectiveness of RIA, consider the feature level activity as being more valuable. All micro-level activity should be tracked as “custom links” with specifics measured through associated “traffic” variables (props and even eVars if the use needs to be measured against success events). This will ensure that page views are not inflated by micro-level activity, and allows for path analysis through the traffic variable.

3.1.5 What to Analyze

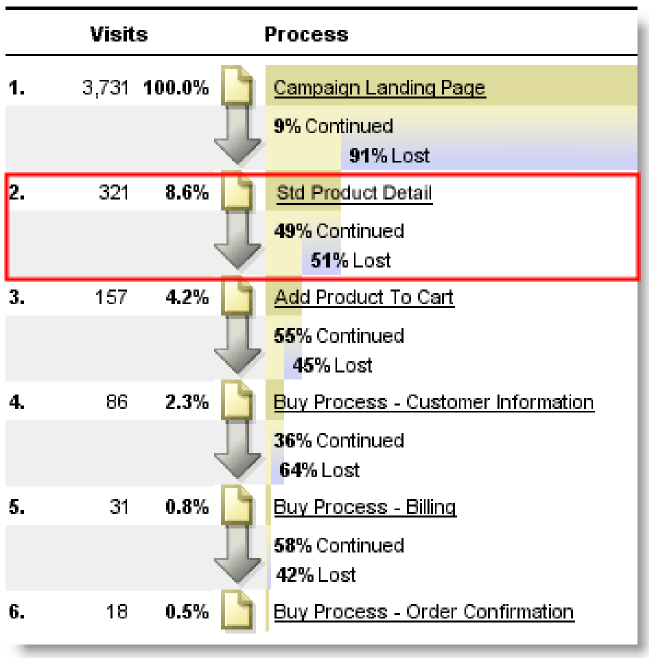
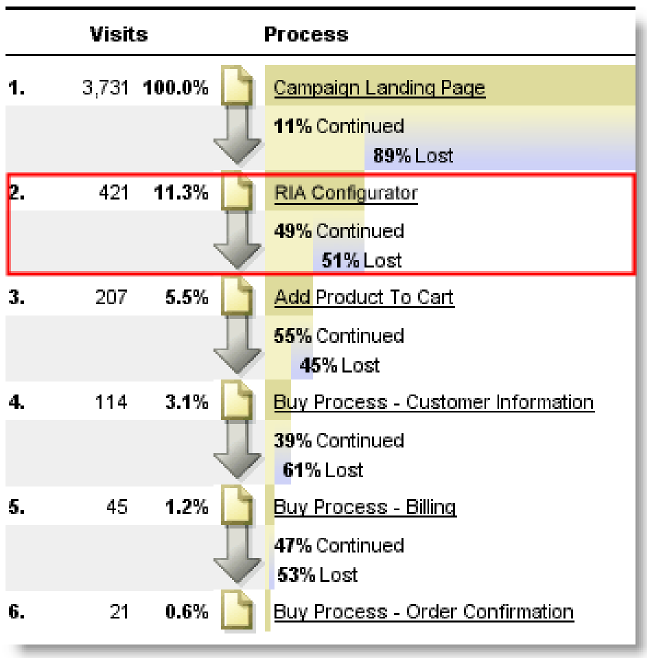
It is important to understand how effectively your RIA is at driving success. This is measured through conversions. A/B comparison becomes critical at this point. Run conversion analysis from the macro-level metrics to success metrics and compare this between the control and the test. A macro-level analysis will provide insight into effectiveness as a whole. Micro-level analysis may provide insight into which features help drive conversion.

Omniture customers who have performed this analysis (macro and micro) have been shocked by the differences between control and test. What will you find?

Next, you can measure efficiencies. This is an analysis of micro-level activity relative to the RIA macro metrics. Do users have to run through more steps to arrive at the same goal? Alone, this analysis points to user actions for the application relative to a control, but, coupled with effectiveness measurements (conversions), it provides insight into whether users like the activity or not. Analysis metrics include visits/features activity; page views/feature activity, visitors/feature activity, etc.

Finally, Omniture recommends that analysis be conducted on path flow and fall out. Are users avoiding the RIA and finding another path to the goal? This analysis can point to incentives, barriers, and usability. Run SiteCatalyst fallout reports built around the architected site flow. Compare the control to the test to gain insight into potential problems. Run path analysis from landing pages to gauge the true traffic patterns. Look into barriers and incentives to channel users toward the goal.

Figure 3-B: Example

Standard Control – Not RIA (8.6%)	RIA Test (11.3%)
	

Standard Control – Not RIA (8.6%)

	Visits		Process
1.	3,731	100.0%	Campaign Landing Page
			9% Continued 91% Lost
2.	321	8.6%	Std Product Detail
			49% Continued 51% Lost
3.	157	4.2%	Add Product To Cart
			55% Continued 45% Lost
4.	86	2.3%	Buy Process - Customer Information
			36% Continued 64% Lost
5.	31	0.8%	Buy Process - Billing
			58% Continued 42% Lost
6.	18	0.5%	Buy Process - Order Confirmation

RIA Test (11.3%)

	Visits		Process
1.	3,731	100.0%	Campaign Landing Page
			11% Continued 89% Lost
2.	421	11.3%	RIA Configurator
			49% Continued 51% Lost
3.	207	5.5%	Add Product To Cart
			55% Continued 45% Lost
4.	114	3.1%	Buy Process - Customer Information
			39% Continued 61% Lost
5.	45	1.2%	Buy Process - Billing
			47% Continued 53% Lost
6.	21	0.6%	Buy Process - Order Confirmation

3.1.6 Suggested Metrics

- RIA Visits
- RIA Visitors
- RIA Page Views
- RIA Feature Activity (Custom Links) measure click activity by feature

3.1.7 Suggested Analysis

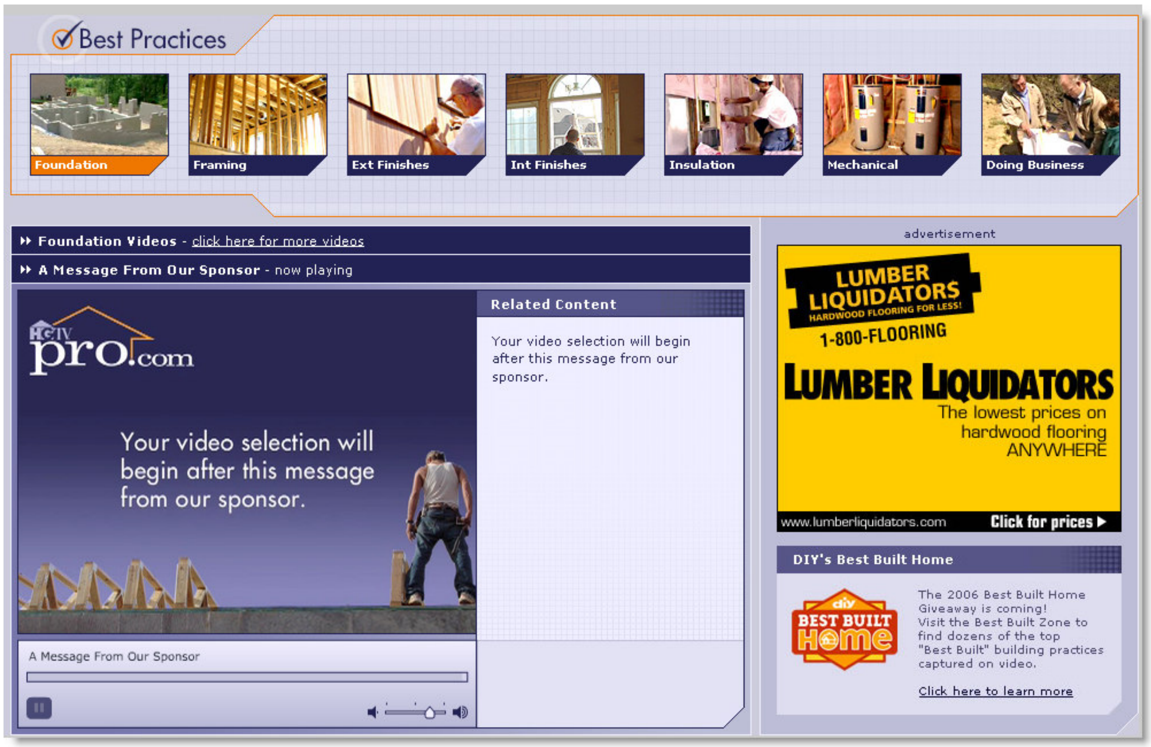
- RIA Feature Activity / RIA Page Views

- RIA Feature Activity / RIA Visits -
- RIA Page Views / Success Metric – Conversion Ratio: measures application effectiveness
- Total RIA Activity / Success Metric – Conversion Ratio: measures application efficiency
- Feature RIA Activity / Success Metric – Conversion Ratio: measures application feature efficiency
- Path Flow to and from RIA
- Fallout Rates through RIA conversion process

3.2 Multimedia Tracking – Video and Audio

Audio and video are booming in popularity. It is estimated that rich media advertising will spike to nearly three times its current level by 2010 *(eMarketer 2006).* With this growth and adoption of broadband connections, it is very likely that you have considered using audio or video on your site.

Audio and video have many playback options. Content can be played through popular media players outside the browser, or through embedded browser controls. Third-party media players, outside the browser, cannot be tracked effectively. For this reason, the following suggestions are based on the assumption that your media is either being played back through browser controls or through a proprietary media player where custom scripting is available, like Flash architected players.


Best Practices
Foundation
Framing
Ext Finishes
Int Finishes
Insulation
Mechanical
Doing Business
Foundation Videos - click here for more videos
A Message From Our Sponsor - now playing
HGTV pro.com
Your video selection will begin after this message from our sponsor.
A Message From Our Sponsor
Related Content
Your video selection will begin after this message from our sponsor.
advertisement
LUMBER LIQUIDATORS HARDWOOD FLOORING FOR LESS!
1-800-FLOORING
LUMBER LIQUIDATORS
The lowest prices on hardwood flooring ANYWHERE
www.lumberliquidators.com
Click for prices
DIY's Best Built Home
diy BEST BUILT Home
The 2006 Best Built Home Giveaway is coming! Visit the Best Built Zone to find dozens of the top "Best Built" building practices captured on video.
Click here to learn more

3.2.1 Measurement

Like RIA, video and audio can be measured based on levels of detail. Most of the measurement is at the page view level, including visits, visitors, and impressions, but detailed analysis provides critical insight into how users consume specific media and media types.

3.2.2 Analysis

Page level metrics provide insight into how effective marketing efforts are at getting people to access the content. These metrics point to how many visitors are finding the content and how compelling collections of content are for the target market. Suggested metrics include Visits, Visitors, Page Views, and Content Categories. Effectiveness is measured by comparing marketing campaigns to these page level metrics.

Detailed metrics offer information about specific content and how effective individual media clips are at retaining customer attention and driving further content consumption. Suggested metrics include clip impressions, ad impressions, and % completion. Effectiveness is measured by % completion.

3.2.3 Suggested Metrics

- Clip Visits: Visits to content
- Clip Visitors: Unique visitors to content
- Clip Impressions: # of times content accessed
- Clip Page Views: Views to page containing clip(s)
- Ad Impressions: # of times ads accessed
- Playback %Completion: % of content played back

3.2.4 Suggested Analysis

- Clip (And Ad) Impressions/Visit
- Clip (And Ad) Impressions/Visitor
- Clip Path Analysis
- Clip Fallout Report
- Avg. Playback % Completion: Content Starts/Percent Complete
- Clip % Completion Rank: Content report broken down by Avg. Playback Completion
- Clip Conversion Rank: Content report broken down by success metric
- Time on page: time on page containing clip(s)
- Demographic trends: Breakdown clip by demographics captured (requires correlation between “Clip Name” traffic variable and related demographic variable
- Geographic trends: Breakdown clip by geographies captured (requires correlation between “Clip Name” traffic variable and geo-segmentation variables

4 Appendix

4.1 Supported SiteCatalyst Reports

The following table displays the SiteCatalyst reports that are available with server-side image tags. For more information on supported SiteCatalyst reports, contact an Omniture representative.

Table 4-A: Key for Appendix Tables

Key Option	Symbol
Fully supported with standard implementation	●
Supported with minor customization to Implementation	◒
Relatively difficult to implement, unavailable, or not applicable	○

NOTE: This list is broken down into major sections, including traffic, pathing, and commerce.

Table 4-B: Supported SiteCatalyst Reports

TRAFFIC	
Site Traffic	
Page Views	●
Hourly Unique Visitors	●
Daily Unique Visitors	●
Monthly Unique Visitors	●
Yearly Unique Visitors	●
Visits	●
File Downloads	◒
Finding Methods	
Referring Domains	◒
Referrers	◒
Search Engines	◒
Search Keywords	◒
Return Frequency	●
Daily Return Visits	●
Return Visits	●
Visit Number	●
Visitor Profile	
Domains	●
Full Domains	●
Top Level Domain	●
Languages	●
Time Zones	●
Visitor Detail	●
Last 100 Visitors	●
User Home Page	○

GeoSegmentation	
Country	●
State	●
City	●
Technology	
Browser Types	●
Browsers	●
Mobile Devices	●
Browser Width	○
Browser Height	○
Operating Systems	●
Monitor Color Depths	○
Monitor Resolutions	●
Netscape Plug-Ins	○
Java	○
JavaScript	○
JavaScript Version	○
Cookies	○
Connection Types	○
Segmentation	
Most Popular Pages	●
Most Popular Channels	●
Most Popular Servers	●
Key Visitors	●
Pages Viewed by Key Visitors	●
Custom Insight	●
Custom Links	●
Custom Insight Properties 1-20	●
PATHS	
Pages	
Page Summary	●
Most Popular Pages	●
Reloads	●
Clicks to Page	●
Time Spent on Page	●
Pages Not Found	●
Entries & Exits	
Entry Pages	●
Exit Pages	●
Exit Links	◒
Complete Paths	
Full Paths	●
Longest Paths	●
Path Length	●
Time Spent per Visit	●
Single-page Visits	●
ClickMap	●

Advanced Analysis	
Previous Page	●
Next Page	●
Previous Page Flow	●
Next Page Flow	●
PathFinder	●
Fall-out	●
COMMERCE	
Purchases	
Conversions & Averages	●
Revenue	●
Orders	●
Units	●
Shopping Cart	
Conversions & Averages	●
Carts	●
Cart Views	●
Cart Additions	●
Cart Removals	●
Checkouts	●
Products	
Conversions & Averages	●
Products	●
Cross-Sell	●
Categories	●
Campaigns	
Conversions & Averages	●
Campaigns	●
Customer Loyalty	
Customer Loyalty	●
Sales Cycle	
Days Before First Purchase	●
Days Since Last Purchase	●
Visit Number	●
Daily Unique Customers	●
Monthly Unique Customers	●
Yearly Unique Customers	●
Finding Methods	
Referring Domains	●
Original Referring Domains	●
Search Engines	●
Search Keywords	●
Visitor Profile	
Countries	●
Languages	●
Time Zones	●

States	●
ZIP/Postal Codes	●
Domains	●
Technology	
Browsers	●
Operating Systems	●
Monitor Resolutions	●
Site Path	
Entry Pages	●
Original Entry Pages	●
Pages per Visit	●
Time Spent on Site	●
Content Groups	●
Custom Variables	
Custom ECommerce Variables 1-20	●

4.2 Variables

SiteCatalyst variables are usually populated in the HTML code on each page or template of the site, as mentioned above. Additionally, plug-ins and global code may be added to the JavaScript file that also populates the variables, or overrides the values set in the HTML pages. The variables that are populated include the following.

Table 4-C: Variables

Supported	Variable	Param	Description
●	pageName	pageName	The name of the page. Uniquely identifies the page and URL in plain English (i.e., "homepage").
●	channel	ch	The section of the site, or "channel" (i.e., "electronics", "news")
●	server	server	Server name or vanity domain to be tracked
●	prop1 - prop50	c[1-50]	Custom values. The specific meaning of each variable may be defined uniquely for each web site.
●	hier1 - hier5	h[1-5]	Hierarchy variables used to record visits and visitors for a hierarchically structured site.
●	campaign*	v0	The campaign variable tracks advertising or email clickthroughs to the site. Campaigns are also correlated to many values throughout the system, such as to commerce and custom events, referring domains, and search engines.
●	eVar1 - eVar50*	v[1-50]	Custom variables that will be tracked and correlated to any events. The specific meaning of each variable may be defined uniquely for each web site. s.eVar values are stored, and correlated to events that happen afterward.

●	products*	products	List of products used in conjunction with the "events" variable
●	events*	events	The list of events that occurred on the current page. Examples include: scOpen, scAdd, scCheckout, prodView, purchase, or any custom event from "event1" to "event20."
●	purchaseID*	purchaseID	Up to 20 character code to uniquely identify the purchase, to be used on the order confirmation ("thank you") page, in conjunction with the "purchase" event.
●	state*	state	State name or ID, to be used on the order confirmation ("thank you") page, in conjunction with the "purchase" event.
●	zip*	zip	Zip code, to be used on the order confirmation ("thank you") page, in conjunction with the "purchase" or other event
NA	pageType	pageType	Used on 404-Page Not Found error pages
◒	referrer	r	Optionally used to override a page's referrer as recorded in SiteCatalyst

NOTE: * Commerce variable. These variables require that the Commerce module be enabled within SiteCatalyst.

4.2.1 Control Variables

Control variables, which control data collection, are contained in the .js file, but they are not considered data collection elements. Control variables are not included in any SiteCatalyst report, but they may affect the data that SiteCatalyst collects, or the appearance of the reports. Control variables include the following.

Table 4-D: Control Variables

Supported	Control Variable	Param	Description
●	s_account		The SiteCatalyst account(s) that the pageview will be reported in. This is referred to as the "Report Suite ID(s)."
●	charSet	ce	The character set used on the page (default is assumed to be ISO-8859-1). A list of available character sets is available from your Omniture Implementation Consultant.
●	currencyCode	cc	The currency code used in the s.products and s.events variables (default is assumed to be "USD" – U.S. Dollars). A list of supported currency codes is available from your Omniture Implementation Consultant.
●	cookieLifetime	cl	Establishes the lifetime of the SiteCatalyst visitor cookie (s.vixxxx). Configuration of this variable is completed by the Omniture Implementation Consultant and should not be

			modified by client personnel.
○	objectID	oid	Manually define the ClickMap object ID for links or buttons which may have ClickMap overlays.

4.2.2 Automatic Variables

Automatic variables are obtained by the .JS code either automatically, or they are populated in the .JS file as a result of the control variables explained above. The names of these variables are defined only within the query string of the image request. There is no equivalent HTML-based variable. The automatic variables include the following.

Support	Automatic Variable	Description
◒	r (Referring URL)	The referring URL as defined by the browser. This value includes all query string parameters for the referring URL, including search strings and other information.
◒	g (Current URL)	The current page's URL. This value may include all query string parameters, depending upon the account settings and configuration.
●	t	Browser time information "Day/Month/Year Hour:Min:Sec Weekdsay Timezoneoffset"
○	v	Whether Java is enabled (Y/N)
○	j	The version of JavaScript supported by the browser
○	bw, bh	The width and height of the browser window
●	s	The width and height of the screen (the physical monitor)
○	c	Screen color depth (8, 16, 32, etc.)
○	ct	Connection type, including LAN, modem, etc.
○	p	Netscape plug-ins (if running Netscape, the list of plug-ins installed in the browser ';' separated list)
○	k	Cookies enabled (whether or not cookies are enabled in the browser based on a test cookie)
○	hp	Whether the current page is set as the browser's Home page (Y,N)
●	pid	Page identifier for ClickMap – previous page

●	pidt	Page identifier type for ClickMap – URL or Page Name of Previous
●	oid	Object identifier for ClickMap – Link URL or User Defined Object ID
●	oidt	Type of link clicked on ("0 = HREF URL, 1 = Custom ID, 2 = Javascript onClick Event, 3 = Form Elements")
●	ot	Object tag name for ClickMap - "Clickmap information, type of link/ form element that was clicked on" "Set by custom code in JS file. More specific than click_context_type, gives type of form element. Example: A = anchor tag (<a href=xxx>xxx</a>), td = table cell (<td>xxx</td>)"

4.2.3 Direct Variables

Direct variables are those set directly by the browser in the HTTP header of the image request sent to SiteCatalyst. These variables are set in each request that is made for any content on the Internet and include some of the most basic user information. The variables are reported directly or indirectly in a number of SiteCatalyst reports. Direct variables include the following.

Support	**Automatic Variable**	**Description**
●	IP address	The IP address is the Internet Protocol address of the user's browser or machine. Note that IP addresses may be "pooled" among multiple users, and that a single user may also use more than one IP address from page to page. The user's IP address is resolved to geographic location based on data provided by Digital Envoy through a partnership with Omniture.
●	domain	The domain from which the user is requesting data. In many cases, this is a company or an Internet Service Provider (ISP) such as AOL.
●	user agent string	The user agent string indicates the browser and version, and the Operating System used. In some cases, it may also contain major plug-in or customization features (i.e., .NET). The Browser, Browser Version, and Operating System reports are based on the user agent string.
●	language	The preferred language setting of the browser

Index

CALL 1.877.722.7088
1.801.722.0139

www.omniture.com
info@omniture.com

550 East Timpanogos Circle
Orem, Utah 84097