

# SiteCatalyst Media Tracking

JavaScript and ActionSource

OMNITURE®  
An Adobe company

IMPLEMENTATION GUIDE

**Copyright 1996-2010.** Adobe Systems Incorporated. All rights reserved. Omniture® is a registered trademark of Adobe Systems Incorporated in the United States, Japan, and the European Community.

[Terms of Use](#) | [Privacy Center](#)

Omniture products and services are licensed under the following Netratings patents: 5,675,510, 5,796,952, 6,115,680, 6,108,637, 6,138,155, 6,643,696, and 6,763,386.

A trademark symbol (®, ™, etc.) denotes an Adobe trademark. An asterisk (\*) denotes a third-party trademark. All third-party trademarks are the property of their respective owners.

14.6 02032009

# Table of Contents

<b>PREFACE</b> .....	<b>iv</b>
<b>CHAPTER 1 - VIDEO TRACKING IN SITECATALYST</b> .....	<b>1</b>
CONFIGURING VIDEO TRACKING .....	2
MEDIA.AUTOTRACK PLAYERS .....	4
UPDATE WEB PAGE CODE TO TRACK VIDEO USAGE .....	6
Video Tracking Options .....	6
Export Video Tracking Data .....	7
FLASH VIDEO TRACKING .....	7
VIDEO TRACKING VARIABLES AND METHODS .....	11
<b>CHAPTER 2 - FLASH AND FLEX TRACKING WITH ACTIONSOURCE</b> .....	<b>15</b>
DOWNLOAD ACTIONSOURCE CODE .....	16
INSTALL THE ACTIONSOURCE COMPONENT .....	16
ACTIONSOURCE IMPLEMENTATION WITH FLASH .....	17
ActionSource Data Tracking Options .....	17
Drag-and-Drop Implementation .....	19
ActionScript Implementation .....	21
ACTIONSOURCE IMPLEMENTATION WITH FLEX .....	23
ACTIONSOURCE VARIABLES .....	24
ACTIONSOURCE METHODS .....	31
track .....	31
trackLink .....	31
setInterface .....	31
<b>APPENDIX A - SAMPLE CODE</b> .....	<b>33</b>
VIDEO TRACKING CODE SAMPLES .....	33
JavaScript Code Placement .....	33
ActionSource Code Placement .....	34
Common Code Samples .....	36
Media.monitor Code Sample .....	37
ACTIONSOURCE CODE SAMPLES .....	38
ActionScript Samples .....	38
HTML Object Tag Sample .....	39
configURL XML File Sample .....	40
NON-AUTOMATIC MEDIA PLAYERS .....	40

# Preface

The Omniture® Media Tracking Implementation Guide describes how to monitor the usage of media files (videos) on your Web site, including Adobe\* Flash\* and Flex\* applications, and correlates this data with revenue generated on your Web site. You can track videos using both JavaScript\* and ActionSource™.

The Media Tracking Implementation Guide includes the following sections:

- [Video Tracking in SiteCatalyst](#)
- [Flash and Flex Tracking with ActionSource](#)

## Terms and Conditions of Use

This document and the related software described in this document is proprietary to Omniture, Inc. and is supplied only under license. The document and software can be used or copied only in accordance with the terms of the license agreement ([Omniture Enterprise Terms of Use](https://sc.omniture.com/p/l10n/1.0/terms.html) - <https://sc.omniture.com/p/l10n/1.0/terms.html>).

The information in this document is subject to change without notice and does not represent a commitment on the part of Omniture.

## Intended Audience

This guide is intended for Web site developers and engineers that are familiar with Omniture SiteCatalyst and implementing SiteCatalyst beacons (image requests) to track Web pages. Additionally, those using this guide should be familiar with the terminology, technologies, and tools related to creating and using videos, including Flash and Flex applications, on their Web site. More information about SiteCatalyst is available in the SiteCatalyst Implementation Guide, which is available in the [SiteCatalyst Help System](#).

## Account Support

Omniture's support team is available to help you:

- Answer specific product questions.
- Use SiteCatalyst reports to their maximum benefit.
- Resolve any technical difficulties you might have.
- Configure SiteCatalyst variables (eVars, props, and events).

## Service and Billing Information

Depending on the service level you have purchased, some of the options described in this guide might not be available to you. Additionally, each account has unique billing needs. Please refer to your contract for pricing, due dates, terms and conditions. If you would like to add to or otherwise change your service level, or if you have questions regarding your current service, please contact your Omniture Account Support Manager.

Omniture welcomes suggestions or feedback you have regarding the contents of this guide. Send comments to your Omniture account representative.

# Contact Information

[CORPORATE ADDRESS]	Omniture, an Adobe Company 550 East Timpanogos Circle Orem, UT 84097
[PHONE]	1.801.722.7000
[FAX]	1.801.722.7001
[TOLL FREE]	1.877.722.7088 (support, billing and sales)
[SUPPORT E-MAIL]	clientcare@omniture.com
[SALES E-MAIL]	sales@omniture.com
[INFORMATION E-MAIL]	info@omniture.com
[CORPORATE URL]	<a href="http://www.omniture.com">http://www.omniture.com</a>
[LOG-IN]	<a href="http://my.omniture.com">http://my.omniture.com</a>

# Video Tracking in SiteCatalyst

SiteCatalyst® video tracking lets you monitor the number of times that visitors view video files on your Web site so you can correlate this data with other Web site analytics. Video tracking is supported in JavaScript and ActionSource.

SiteCatalyst tracks videos on your Web site by gathering basic information from the media player. From this information, SiteCatalyst builds a session of events and sends it to Omniture® collection servers for processing. After the collection servers process the video data, it is accessible through a set of SiteCatalyst video reports.

The general tasks involved in configuring and using video tracking on your Web site include:

TASK	DESCRIPTION
1	Enable and configure video tracking in the SiteCatalyst Administration Console. For more information, see <a href="#">"Configuring Video Tracking" on page 2</a> .
2	If necessary, generate Omniture site tracking code that supports video tracking. Use JavaScript tracking code version H.19.3 or later, and ActionSource version 2.7 or later.  You can generate updated site tracking code in SiteCatalyst's Administration Console. For more information, see "Code Manager" in the Administration Console User Guide.
3	Determine the media players and Flash video players being used on your Web site, then determine how best to track player usage. For more information, see <a href="#">"Flash Video Tracking" on page 7</a> , and <a href="#">"Non-Automatic Media Players" on page 40</a> .
4	Add video tracking code to the Web site pages where you want to track video usage. For more information, see <a href="#">"Media.autoTrack Players" on page 4</a> .  For information specific to tracking Flash videos, see <a href="#">"Flash Video Tracking" on page 7</a> .  Configure the appropriate video tracking method and settings. For more information, see <a href="#">"Video Tracking Variables and Methods" on page 11</a> .
5	Generate video tracking reports. For more information about SiteCatalyst video reports, see "Report Descriptions" in the Omniture SiteCatalyst User Guide. Some examples of video tracking report data include the following: <ul style="list-style-type: none"> <li>• The number of times videos were viewed</li> <li>• Whether the videos were watched in their entirety</li> <li>• Which portions of the videos were viewed most frequently</li> </ul>

## 1.1 Configuring Video Tracking

Before using SiteCatalyst video tracking, you must configure it in the Administration Console. The video tracking configuration determines what data you see in SiteCatalyst video reports.

### To configure video tracking

1. Log in to the Omniture Suite.
2. In the Omniture Suite, click **Admin > Report Suites**.
3. In the Report Suite Manager, click **Edit Settings > Video Management > Video Tracking Settings**.

### Video Tracking ?

Video reports are a predefined set of reports used to report on visitor engagement and the effect videos have on conversion. Once enabled, video reports will only show data if JavaScript or ActionSource have been configured to send video data from the web page or video player.

#### Video Settings

Setting	Value
<input type="checkbox"/> Enable Video Reports	<input type="text" value="Disabled"/>
<input type="checkbox"/> Visits Tracked	<input type="text" value="Disabled"/>
<input type="checkbox"/> Daily Visitors Tracked	<input type="text" value="Disabled"/>
<input type="checkbox"/> Paths Tracked	<input type="text" value="Disabled"/>
<input type="checkbox"/> Conversion Level	<input type="text" value="Disabled"/>
<input type="checkbox"/> Segments Tracked	<input type="text" value="Disabled"/>
<input type="checkbox"/> Segment Length	<input type="text" value="0 Seconds"/>
<input type="checkbox"/> Allocate the success events across the last X videos	<input type="text" value=""/>
<input type="checkbox"/> Video History Expiration	<input type="text" value=""/>

4. Enable video tracking, configure the video tracking settings to your needs, then click **Save**.

To enable a video tracking setting, select the setting you want to enable, then change the setting's value to **Enabled**. [Table 1.1](#) describes the video tracking settings.

---

**NOTE:** To prevent data processing delays for the entire report suite, enable only the video tracking features you need.

---

**Table 1.1: Video Report Settings**

SETTING	DESCRIPTION
Enable Video Reports	<p>Enables video reports in the selected report suite. Video reporting must be enabled for video tracking to function. It also enables the following SiteCatalyst video tracking reports.</p> <ul style="list-style-type: none"> <li>• Video Views</li> <li>• Video Visits</li> <li>• Video Daily Unique Visitors</li> <li>• Videos</li> <li>• Time Spent on Video</li> <li>• Media Player</li> </ul>
Visits Tracked	Enables a SiteCatalyst metric that tracks the number of views on a per-video basis.
Daily Visitors Tracked	Enables a SiteCatalyst metric that tracks daily unique visitors on a per-video basis.
Paths Tracked	Enables video pathing, which tracks the order in which visitors view videos on your Web site.
Conversion Level	<p>Determines whether success events are allocated to videos. Enable Conversion Level by selecting one of the following options:</p> <p><b>Enabled with No Subrelations:</b> Enables conversion tracking, but does not permit report breakdowns by other commerce metric reports. Disabling sub-relations improves report performance speed.</p> <p><b>Enabled with Basic Subrelations:</b> Enables conversion tracking, and lets you breakdown the video conversion report by any other report that has full subrelations enabled (for example, campaign or product).</p>
Segments Tracked	Enables reporting of video segments (portions of the video) in SiteCatalyst reports. This lets SiteCatalyst report on video usage based on portions of the video that visitors actually view.
Segment Length	<p>Used when <b>Segments Tracked</b> is enabled.</p> <p>Defines the segment length for videos, in seconds. For example, set the Segment Length to 10 to report a visitor's video viewing in 10-second increments. This number must be a whole number greater or equal to five (5).</p>
Allocate Success Events Across the Last X Videos	<p>Determines how many videos in the visitor's path receive credit for a conversion event. For example, if someone views 5 videos prior to purchasing a \$100 item and this value is set to 5, SiteCatalyst assigns each a video revenue credit of \$20 (using the Revenue model) or \$100 (using Revenue Participation model).</p> <p>The video count can span visits.</p>
Video History Expiration	<p>Determines when to clear a visitor's video history. Select the appropriate expiration trigger from the menu. The default value is <code>visit</code>.</p> <p>You can select a time period or an event. If you use a time period, the time period resets each time the visitor views a video. For example, if you select <code>Week</code> for the Video History Expiration value, the video history clears one week following the last visitor view of the video. SiteCatalyst starts a new video history the next time the visitor views a video.</p>

## 1.2 Media.autoTrack Players

In most cases, you can use `Media.autoTrack` to have the media module track and send information automatically. However, you should be aware of the following issues when using `Media.autoTrack`:

- Omniture's JavaScript beacon identifies all `<embed>` or `<object>` tags in the page HTML. It then searches the data in each tag to determine which media player, if any, is being used. When the type of player is determined, the beacon monitors the player for video tracking data.
- Omniture uses the video name to identify the video in SiteCatalyst reports. You can use classifications to assign a more friendly name to the video.

`Media.autoTrack` supports the following media players. You should be aware of the caveats associated with each type of media player, where applicable. To track other media players, you must manually implement video tracking by writing functions that call the appropriate JavaScript or ActionSource functions at the proper times during player usage. For more information, see ["Non-Automatic Media Players" on page 40](#).

### Windows Media Player (Windows IE only)

- `Media.autoTrack` works with Windows Media Player only in Internet Explorer on the Windows operating system. To track Windows Media Player in Firefox or other Web browsers, treat Windows Media Player as a non-automatic media player (See ["Non-Automatic Media Players" on page 40](#)).
- When using `Media.autoTrack` with Windows Media Player, you must use an `<object>` tag with the correct class ID in addition to an `<embed>` tag.

### QuickTime

- QuickTime does not support event listeners, so `Media.autoTrack` queries the QuickTime player every 500 milliseconds to see if the video is still playing. Because of this, `Media.autoTrack` might miss some visitor events (stops, starts, etc.)

### Real Player

- No issues.

Additionally, `Media.autoTrack` supports the following Flash video players. For more information about Flash video tracking, see ["Flash Video Tracking" on page 7](#).

### FLVPlayback

- No issues.

### MediaDisplay

- No issues.

### MediaPlayback

- No issues.

### Brightcove 2 and 3 Players

- `Media.autoTrack` detects instances of Brightcove 2 and Brightcove 3 video players that exist in the Flash movie that contains the ActionSource instance. For information about configuring Brightcove video players, see ["BrightCove Players" on page 9](#).
- `Media.autoTrack` supports externally loaded video players, such as Brightcove, that might not be instantiated when the Flash creates the ActionSource instance. To recognize an external video player, `Media.autoTrack` periodically rechecks for a video player if it does not initially find one.
- ActionSource tracks Brightcove video playback using the Brightcove video ID, prefixed with either `Brightcove 2:` or `Brightcove 3:` (depending on the Brightcove player version). For example, a Brightcove 3 video ID of `abc123667` gets a video name of `Brightcove 3:abc123667`.

- ActionSource gets the video ID from the video DTO calling `getCurrentVideo` or `getCurrentTitle`.
- To improve video name readability, you can upload classifications that assign the video a friendly name. For example, `Brightcove 3:abc123667` could be classified as `On-line Auto Ad #1`.
- Brightcove recommends using its "analytics SWF" for analytics tracking of Brightcove video players so you can build a tracking player without Flash. You can use the pre-built Flash movie, `ActionSourceExtension.swf`, for this purpose. For more information, see ["External Flash Players" on page 8](#).
- If you are already building a more complex Flash movie and using the Brightcove video player with your movie, you can also implement ActionSource using the Drag-and-Drop or ActionScript method. Doing this lets ActionSource detect the Brightcove video player in your movie when it loads.

---

**NOTE:** For the best control over video, including Flash video, tracking, create a player "skin" that calls the video tracking methods manually each time the user interacts with the video player. This lets you set events or variables at certain milestones, send data to Omniture prior to the end of the video, or otherwise enhance data collection. For more information, see ["Non-Automatic Media Players" on page 40](#).

---

## 1.3 Update Web Page Code to Track Video Usage

To track video file usage on your Web site, make sure that your Web site's Omniture tracking code includes the appropriate media module functionality, in the form of video tracking variables and methods. Regardless of the tracking method used, video tracking variables and method names are the same.

**JavaScript Media Module:** The media module contains all variables and methods used to configure video tracking for your Web pages. Add the video tracking code to your Web site's JavaScript beacon (`s_code.js`), then use `Media.autoTrack` ([See "Video Tracking Variables and Methods" on page 11](#)) to enable video tracking on your Web pages.

**ActionSource Media Module:** Add the ActionSource component code to the Flash application. ActionSource supports both AS2 (ActionScript 2) and AS3 (ActionScript 3) for Flash video tracking. For detailed information about tracking Flash videos, see ["Flash Video Tracking" on page 7](#).

This section includes the following topics:

- [Video Tracking Options](#)
- [Export Video Tracking Data](#)

All video tracking functionality is part of the media module, meaning that you reference video tracking variables and methods using the `Media` prefix. For example, if `s` is the name of your JavaScript object or ActionSource instance, reference media module components using the `s.Media` prefix.

### Video Tracking Options

SiteCatalyst supports the following video tracking options. You can select the tracking option with the `Media.trackWhilePlaying` variable ([See "Video Tracking Variables and Methods" on page 11](#)).

#### Continuous Tracking

Continuous tracking is a relatively new method for tracking video usage that sends data to Omniture collection servers at the beginning of the video, the end of the video, and at selected increments throughout the course of the video. This prevents loss of data in the event that a user closes a Web browser or goes to another Web site while the video is playing.

The media module includes variables that let you define how often the media player sends data to Omniture collection servers. This method provides the most accurate video tracking.

#### Buffered Request Tracking

Buffered Request tracking is the default, and legacy, method for tracking video usage that buffers video usage data in a cookie or Flash shared object, and sends data to Omniture collection servers only at the end of the video, or when the visitor next accesses a page on your Web site (JavaScript tracking only). This method reduces traffic to Omniture collection servers, but risks losing data if the visitor leaves your Web site or closes the Web browser before the end of the video.

When using buffered request tracking with the `Media.autoTrack` variable, the video name is set to the URL of the video for Video reports. Variables specified in `Media.trackVars` and `Media.trackEvents` that have values at the time of the data transfer are sent with the video tracking data. For more information about these variables, see ["Video Tracking Variables and Methods" on page 11](#).

## Export Video Tracking Data

In addition to standard tracking options, you can use Omniture Data Sources or the Data Insertion API to export video tracking data to Omniture collection servers.

**Table 1.2: Attributes for Full Data Sources and the XML Data Insertion API**

XML TAG	DESCRIPTION
<mediaName>	The name of the video.
<mediaLength>	The length, in seconds, of the video.
<mediaPlayer>	The name of the media player used.
<mediaSession>	<p>Pipe-delimited " " list of video segments played, in seconds. For example, 0-15 20-25 describes a video playback session where the visitor:</p> <ul style="list-style-type: none"> <li>Started playing the video at offset 0 (the beginning), and played to second 15.</li> <li>Skipped seconds 16-19 (by jumping forward, scrubbing, fast-forwarding, etc.)</li> <li>Stopped viewing the video at 25 seconds (clicked Stop, closed video player, navigated to a different Web page, etc.)</li> </ul>

For more information about these data export options, see the Omniture [Data Sources User Guide](#) and the [Data Insertion API](#).

## 1.4 Flash Video Tracking

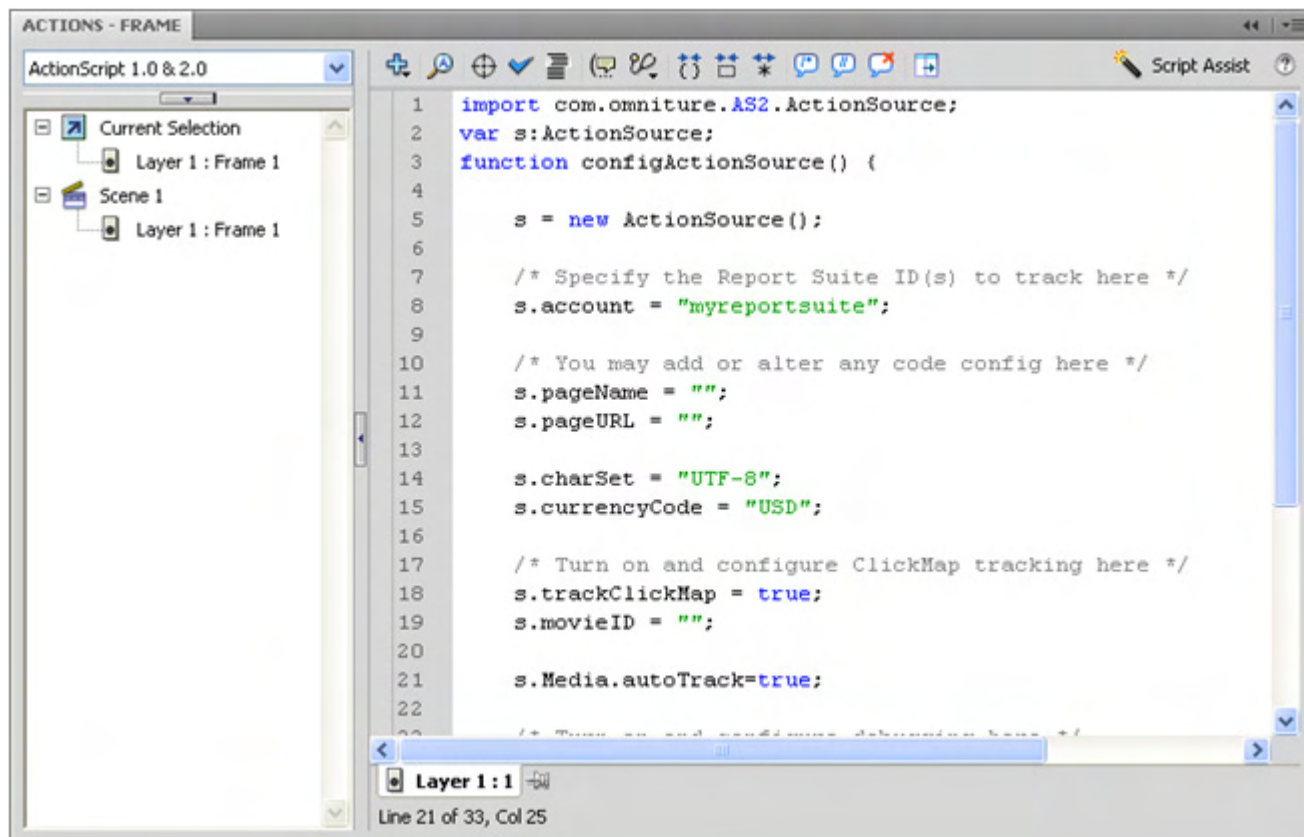
Omniture supports automated Flash video tracking through `Media.autoTrack` and `ActionSource`. Because of this, you should be familiar with general `ActionSource` tracking tools and techniques before working with Flash videos.

**NOTE:** The tasks in this section illustrate Flash video tracking using Adobe Flash CS4. If you use a different version of Flash, your installation and configuration process might vary.

### To implement Flash video tracking with ActionSource

- Download and install the latest `ActionSource` component (version 2.6.4 or later) into your Flash project. For more information, see ["Download ActionSource Code" on page 16](#) and ["Install the ActionSource Component" on page 16](#).
- Launch Flash and open the Flash project where you want to include a Flash video.
- In the Timeline pane, select a frame, available to the entire Flash application, where you want to import the Flash video.
- In the Actions pane (**Window > Actions**), insert the appropriate configuration script.

Make sure the configuration script matches the ActionScript version of your Flash project. For more information, see ["Download ActionSource Code"](#) on page 16.



5. In the configActionSource function, set `Media.autoTrack=true` to automatically track visitor usage data for the Flash video.
6. Open the Output pane (**Window > Output**), then test the Flash video (**Control > Test Video**).

The Output pane displays all image requests sent during the video playback. The video tracking data is transmitted in a pev3 query-string parameter that includes the following information about the video:

- **Name:** The name of the video file.
- **Length:** The length of the video file, in seconds.
- **Player Name:** The video player used to play the video file.
- **Total Seconds Played:** The total number of seconds the video was played.
- **Start Timestamp:** Timestamp that identifies when the video play started.
- **Play Session:** The details of the play session. This field indicates how the user interacted with the video. This might include data such as where they started playing the video, whether they used the video slider to advance the video, and where they stopped playing the video.

For example:

```
pev3=GordonVid.flv--*--138--*--Flash FLVPlayback--*--138--*--1236791711--*--S0E125S125E138
```

## External Flash Players

To support external Flash players, ActionSource provides pre-built Flash movie files that create an ActionSource instance when they play. You can download these pre-built Flash movie in the SiteCatalyst Code Manager. For more information, see "Code Manager" in the Administration Console User Guide.

To use a pre-built Flash movie file, predefine the `ActionSource` instance with `useExternalVariables`, then configure it using a query-string, `FlashVars`, and/or an external XML configuration file (with the `configURL` variable). For more information about these variables, see ["ActionSource Variables" on page 24](#).

---

**NOTE:** This feature supports the Brightcove "analytics SWF".

---

Omniure provides a prebuilt Flash movie file for each of the following combinations of ActionScript/Flash. Use the file that supports the video player for which your Flash movie is built:

- `ActionSourceExtension.swf`: ActionScript 3 (AS3), Flash 9+
- `ActionSourceExtension_AS2.swf`: ActionScript 2 (AS2), Flash 8+
- `ActionSourceExtension_AS2_FPL.swf`: AS2, Flash 6+ or Flash Lite

When a movie file that points to the pre-built Flash movie loads, you must provide a mechanism that `ActionSource` can use to find the root/stage of your movie file or `ClickMap` and `Media.autoTrack` cannot function. To do this, call `setInterface` (See ["setInterface" on page 31](#)) on the `MovieClip`, passing in the root/stage of your Flash movie, or a `Display` object on the stage of your Flash movie.

### BrightCove Players

`Media.autoTrack` can automatically detect instances of Brightcove 2 and Brightcove 3 players that exist in the Flash movie that contains the `ActionSource` component instance.

Integrating a Brightcove video player with `ActionSource` includes:

- [Installing the Pre-Built Flash Movie](#)
- [Configuring a Brightcove 3 Player](#)
- [Configuring a Brightcove 2 Player](#)

### Installing the Pre-Built Flash Movie

To quickly enable SiteCatalyst tracking for a Brightcove player, leverage the `ActionSource 2.7` pre-built Flash movie and use an XML file to define the SiteCatalyst data capture.

### To install the pre-built Flash movie

1. Upload the appropriate pre-built Flash movie to your Web site.

Use the pre-built movie appropriate to the type of player you want to support. For more information, see ["External Flash Players" on page 8](#).

2. Create an XML configuration file.

The pre-built Flash movie uses the XML configuration to define the data capture for SiteCatalyst. For example:

```
<config>
  <account>ElectronicsPlus</account>
  <debugTracking>>true</debugTracking>
  <visitorNamespace>mycompany</visitorNamespace>
  <dc>112</dc>
  <pageName>Brightcove Testing</pageName>
  <Media>
    <autoTrack>true</autoTrack>
    <trackWhilePlaying>true</trackWhilePlaying>
    <trackSeconds>10</trackSeconds>
  </Media>
</config>
```


For more information, see the `configURL` variable description in ["ActionSource Variables" on page 24](#).

3. Upload the XML configuration to the same URL where you uploaded the pre-built Flash movie.

### Configuring a Brightcove 3 Player

After installing the pre-built Flash movie and XML configuration file, you can integrate a Brightcove 3 player with ActionSource.

#### To configure a Brightcove 3 Player

1. Upload the pre-built Flash movie and XML configuration file to your Web site.  
For more information, see ["Installing the Pre-Built Flash Movie" on page 9](#).
2. Browse to [studio3.brightcove.com](http://studio3.brightcove.com).  
Sign in using your Brightcove credentials.
3. In the Header, select the Publishing Module icon .
4. Double-click the player that you want to configure for ActionSource.  
You can create a new player, if necessary.
5. In the Global tab, select **Enable ActionScript/JavaScript APIs**.
6. In the Plug-Ins tab, specify the URL where you uploaded the pre-built Flash movie and XML file with the `configURL` query-string parameter.

For example: `www.mycompany.com/video/ActionSourceExtension.swf?s.configURL=www.mycompany.com/video/config.xml`.

---

**NOTE:** For more information about `config.xml`, see ["Installing the Pre-Built Flash Movie" on page 9](#). For more information about `configURL`, including information about cross-domain considerations, see ["configURL" on page 25](#).

---

7. Click **Save Changes**.
8. Confirm the player is communicating with ActionSource by previewing the player with FireBug open to view the player communications. When configured properly, Firebug displays the player reference to the pre-built Flash movie URL for the ActionSource configuration information.

### Configuring a Brightcove 2 Player

After installing the pre-built Flash movie and XML configuration file, you can integrate a Brightcove 2 player with ActionSource.

---

**NOTE:** Before attempting to configure a Brightcove 2 player, contact the Brightcove support team to make sure that "Utility SWF" is enabled for your account.

---

1. Upload the pre-built Flash movie and XML configuration file to your Web site.  
For more information, see ["Installing the Pre-Built Flash Movie" on page 9](#).
2. Browse to [studio.brightcove.com](http://studio.brightcove.com).  
Sign in using your Brightcove credentials.
3. Select the Players tab, then click **Continue to Section**.
4. Select the player you want to configure for ActionSource, then click **Edit**.  
You can create a new player, if necessary.
5. Select the Security / Filtering tab and deselect **Disable ActionScript/JavaScript API**, if necessary.
6. In the **Paths to utility SWFs** field, specify the URL where you uploaded the pre-built Flash movie and XML configuration file with the `s.configURL` query-string parameter. For example:

`www.mycompany.com/video/ActionSourceExtension_AS2.swf?s.configURL=www.mycompany.com/video/config.xml.`

For more information, see ["Installing the Pre-Built Flash Movie" on page 9.](#)

7. Click **Save**.
8. Confirm the player is communicating with ActionSource by opening Firebug, then clicking **Preview Video Player**. When configured properly, Firebug displays the player reference the pre-built Flash movie URL for the ActionSource configuration information.

## 1.5 Video Tracking Variables and Methods

The media module provides several video tracking variables ([Table 1.3](#)) and methods ([Table 1.4](#)) that let you configure either JavaScript or ActionSource tracking code to support the level of video tracking you need. Variable and method names are the same in both JavaScript and ActionSource.

**Table 1.3: Video Tracking Variables**

VARIABLE	DESCRIPTION
<code>Media.autoTrack</code>	<p><b>Syntax:</b> <code>s.Media.autoTrack = true</code></p> <p>Enables basic video tracking that includes: video name, video length in seconds (both length watched, and total length), and media player name.</p> <p>For more information about the media players supported by <code>Media.autoTrack</code>, see <a href="#">"Media.autoTrack Players" on page 4.</a></p>
<code>Media.playerName</code>	<p><b>Syntax:</b> <code>s.Media.playerName = "Custom Player Name"</code></p> <p>Use when <code>Media.autoTrack=true</code>.</p> <p>Specifies a custom media player name when using automatic tracking.</p>
<code>Media.trackVars</code>	<p><b>Syntax:</b> <code>s.Media.trackVars = "events,prop1,prop5"</code></p> <p>Specifies a comma-separated list of variables to send with the video tracking data.</p> <p><b>NOTE:</b> To use <code>Media.trackEvents</code>, you must include <code>events</code> as one of the variables in <code>Media.trackVars</code>.</p> <p>For more information, see the SiteCatalyst Implementation Guide, available in the SiteCatalyst Help System.</p>
<code>Media.trackEvents</code>	<p><b>Syntax:</b> <code>s.Media.trackEvents = "event2,event3"</code></p> <p>Use when <code>Media.trackVars</code> includes the <code>events</code> variable.</p> <p>Specifies a comma separated list of events to send with the video tracking data.</p> <p>For more information, see the SiteCatalyst Implementation Guide, available in the SiteCatalyst Help System.</p>

VARIABLE	DESCRIPTION
Media.trackWhilePlaying	<p><b>Syntax:</b> <code>s.Media.trackWhilePlaying = true</code></p> <p>When <code>true</code>, video tracking uses Continuous Tracking for sending data to Omniture collection servers. Otherwise, video tracking uses Buffered Request Tracking (the default and legacy tracking method) for sending data to Omniture collection servers. For more information, see <a href="#">"Video Tracking Options" on page 6</a>.</p> <p>Use <code>Media.trackSeconds</code>, <code>Media.trackMilestones</code>, <code>Media.track</code>, and <code>Media.monitor</code> to define how often to send video tracking data to Omniture collection servers.</p>
Media.trackSeconds	<p><b>Syntax:</b> <code>s.Media.trackSeconds = 15</code></p> <p>Used when <code>s.Media.trackWhilePlaying = true</code>.</p> <p>Defines the interval, in seconds, for sending video tracking data to Omniture collection servers while the video is playing. The value must be set in increments of 5 seconds.</p> <p>The variable sends video tracking data at the start (when <code>Media.open</code> is called) and end (when <code>Media.close</code> is called) of the video, and on multiples of the value specified for <code>Media.trackSeconds</code>. For example, every 15 seconds that the video is playing.</p> <p><code>Media.trackSeconds</code> tracks time spent viewing a video. It does not track how much of the video a visitor views. It does not distinguish between viewing the file from beginning to end, and replaying a portion of the video multiple times.</p> <p><b>NOTE:</b> Do not use this variable simultaneously with <code>s.Media.trackMilestones</code>.</p>
Media.trackMilestones	<p><b>Syntax:</b> <code>s.Media.trackMilestones = "10,50,90"</code></p> <p>Used when <code>s.Media.trackWhilePlaying = true</code>.</p> <p>Defines the interval, as a percentage of the video watched, for sending video tracking data to Omniture collection servers. Specify the milestones as a comma-separated list of whole numbers. For example: 10 = 10%, 23 = 23%).</p> <p>The variable sends video tracking data at the start (when <code>s.Media.open</code> is called) and end (when <code>s.Media.close</code> is called) of the video, and on video percentages specified in <code>Media.trackMilestones</code>, based on the length of the video.</p> <p>Because these milestones are fixed points in the video, if a visitor views past the 10% milestone, then rewinds and passes the 10% milestone again, Omniture sends the tracking data multiple times. Similarly, if a visitor fast forwards past a milestone, Omniture does not send the tracking data for that milestone.</p> <p><b>NOTE:</b> Do not use this variable simultaneously with <code>s.Media.trackSeconds</code>.</p>

**Table 1.4: Media Tracking Methods**

METHOD	DESCRIPTION
Media.open	<p><b>Syntax:</b> <code>s.Media.open(mediaName,mediaLength,mediaPlayerName)</code></p> <p>Use when <code>Media.autoTrack=false</code>.</p> <p>Prepares the media module to collect the following video tracking data. This method takes the following parameters:</p> <p><b>mediaName:</b> The name of the video as you want it to appear in SiteCatalyst video reports.</p> <p><b>mediaLength:</b> The length of the video in seconds.</p> <p><b>mediaPlayerName:</b> The name of the media player used to view the video, as you want it to appear in SiteCatalyst video reports.</p>
Media.close	<p><b>Syntax:</b> <code>s.Media.close(mediaName)</code></p> <p>Use when <code>Media.autoTrack=false</code>.</p> <p>Ends video data collection and sends information to Omniture collection servers. Call this method at the end of the video. This method takes the following parameters:</p> <p><b>mediaName:</b> The name of the video as you want it to appear in SiteCatalyst video reports.</p> <p>When using Buffered Request Tracking, this function is called automatically the next time <code>t()</code> is called (typically on the next page) to clear the <code>s_br</code> cookie. For more information, see <a href="#">"Video Tracking Options" on page 6</a></p>
Media.play	<p><b>Syntax:</b> <code>s.Media.play(mediaName, mediaOffset)</code></p> <p>Use when <code>Media.autoTrack=false</code>.</p> <p>Call this method anytime a video starts playing. This method takes the following parameters:</p> <p><b>mediaName:</b> This must match the name used in <code>Media.open</code>.</p> <p><b>mediaOffset:</b> The number of seconds into the video that play begins. Specify the offset based on the video starting at second zero.</p>
Media.stop	<p><b>Syntax:</b> <code>s.Media.stop(mediaName, mediaOffset)</code></p> <p>Use when <code>Media.autoTrack=false</code>.</p> <p>Tracks a stop event (stop, pause, etc.) for the specified video. This method takes the following parameters:</p> <p><b>mediaName:</b> The name of the video as you want it to appear in SiteCatalyst video reports.</p> <p><b>mediaOffset:</b> The number of seconds into the video that the stop or pause event occurs. Specify the offset based on the video starting at second zero.</p>

METHOD	DESCRIPTION
<code>Media.monitor</code>	<p><b>Syntax:</b> <code>s.Media.monitor(s,media)</code></p> <p>Lets you monitor the status of each video that is currently playing. With it, you can setup other variables (Props, eVars, Events) and call <code>Media.track</code> based on the current state of the video as it is playing.</p> <p>This is useful if you have code that tracks events, such as resizing and volume adjustments, that you want to send when the video finishes playing. This method takes the following parameters:</p> <p><b>s:</b> The tracking object name (the JavaScript object or ActionSource instance).</p> <p><b>media:</b> An object with members providing the state of the video. These members include:</p> <ul style="list-style-type: none"> <li>• <b>media.name:</b> The name of the video given in the call to <code>Media.open()</code></li> <li>• <b>media.length:</b> The length of the video in seconds given in the call to <code>Media.open()</code>.</li> <li>• <b>media.playerName:</b> The name of the media player given in the call to <code>Media.open()</code>.</li> <li>• <b>media.event:</b> A string containing the event name that caused the <code>Media.monitor</code> call. These events are: <ul style="list-style-type: none"> <li>• <b>OPEN:</b> The first call to <code>Media.open</code>.</li> <li>• <b>CLOSE:</b> When <code>Media.close</code> is called.</li> <li>• <b>PLAY:</b> When <code>Media.play</code> is called.</li> <li>• <b>STOP:</b> When <code>Media.stop</code> is called.</li> <li>• <b>MONITOR:</b> When our automatic monitoring checks the state of the video while it's playing.</li> </ul> </li> <li>• <b>media.openTime:</b> A date object containing data about when <code>s.Media.open()</code> was called.</li> <li>• <b>media.offset:</b> The current offset, in seconds, (actual point in the video) into the video. The offset starts at zero (the first second of the video is second 0).</li> <li>• <b>media.percent:</b> The current percentage of the video that has played, based on the video length and the current offset.</li> <li>• <b>media.timePlayed:</b> The total number of seconds played so far.</li> </ul>
<code>Media.track</code>	<p><b>Syntax:</b> <code>s.Media.track(mediaName)</code></p> <p>Immediately sends the current video state, along with any <code>Media.trackVars</code> and <code>Media.trackEvents</code>. Call <code>Media.open</code> and <code>Media.play</code> on the video before calling this method. This method takes the following parameter:</p> <p><b>mediaName:</b> The name of the video as you want it to appear in SiteCatalyst video reports. Make sure you pass the same <code>mediaName</code> used to call <code>Media.open</code>.</p> <p>This method is the only way to send in other variables while the video is playing.</p> <p>This method resets the seconds interval and percent milestone counters to zero to prevent multiple tracking hits.</p>

# Flash and Flex Tracking with ActionSource

Omniure® ActionSource™ tracks Flash applications with ActionScript\*. With ActionSource, you can track, record, and measure the bandwidth and effectiveness of Flash applications on your Web site.

Tracking Flash applications is dependent on two programming languages: ActionScript and JavaScript. However, communication between these technologies can complicate the implementation process and introduce technical barriers.

Omniure ActionSource simplifies the implementation process for tracking Flash applications, and improves performance and accuracy. Its native ActionScript engine eliminates the JavaScript dependency and maximizes application portability and accuracy.

The general tasks involved in configuring and using ActionSource on your Web site include:

TASK	DESCRIPTION
1	From SiteCatalyst's Code Manager, download the ActionSource code necessary to collect tracking data from your Flash or Flex application. For more information, see <a href="#">"Download ActionSource Code" on page 16</a> .
2	Install the ActionSource component into your Flash or Flex application. For more information, see <a href="#">"Install the ActionSource Component" on page 16</a> .
3	<p>Implement ActionSource tracking in your Flash or Flex application. To do this, you must be familiar with ActionSource variables and methods so you can configure ActionSource to gather the proper data, and transmit that data to Omniure collection servers at the appropriate times. For more information, see <a href="#">"ActionSource Variables" on page 24</a> and <a href="#">"ActionSource Methods" on page 31</a>.</p> <p>ActionSource supports multiple implementation options for Flash applications. For more information, see <a href="#">"ActionSource Implementation with Flash" on page 17</a>.</p> <p>For more information about implementing ActionSource tracking in a Flex application, see <a href="#">"ActionSource Implementation with Flex" on page 23</a>.</p>
4	If you are delivering Flash video on a Web page, or as part of your Flash application, you can use ActionSource to track video usage. For more information, see <a href="#">"Flash Video Tracking" on page 7</a> .

## 2.1 Download ActionSource Code

ActionSource is a component that you compile with your Flash or Flex application. The ActionSource component acts as a code library for your Flash or Flex application, similar to `s_code.js` for standard JavaScript tracking.

Omniture distributes the ActionSource component and configuration files as both a `.mxp` (Flash) and a `.swc` (Flex) file. Both distributions use the same core `.swc` file, but the `.mxp` package simplifies the Flash installation. Flex references the `.swc` directly from the code.

You can access the component and configuration script from the SiteCatalyst Administration Console.

---

**NOTE:** Omniture does not recommend using an external `.swf` file (`OmnitureActionSource.swf`) at runtime. Instead, embed the component and all supporting files directly in your application. This has the following benefits:

- Eliminates the possibility of broken references to an external file.
  - Lets you use the application across domains without a cross domain security file.
  - Simplifies application implementation.
- 

### To download ActionSource code

1. Log in to the Omniture Suite.
2. In the Omniture Suite, click **Admin > Admin Console**.
3. In the left-side navigation, select **Admin Console > Code Manager**.
4. In the **Select the type of code to generate** field, select **ActionScript (Flash/Flex)**.
5. In the Options pane, specify the following information, then click **Generate Code**.
  - Report Suite:** Select your report suite.
  - Character Encoding:** Select your Web site's character-encoding format.
  - Currency:** Specify your desired currency for tracking conversion statistics.
  - Number of periods in domain name:** Specify the number of periods in your domain name.
6. In the Warning window, click **OK**.
 

The Code Manager generates the ActionSource code for your Web site in multiple versions: ActionScript 2, ActionScript 3, and Flex.
7. Select the **Component Files** tab, then save one or both component files to your local system.
8. Copy the appropriate configuration script.
 

Select the tab appropriate to the type of application you are creating: ActionScript 2 (AS2), ActionScript 3 (AS3), or Flex. Copy this code onto your computer's clipboard, or into a text file where you can access it later.

## 2.2 Install the ActionSource Component

Before using ActionSource in your Flash applications, you must install the ActionSource component into your Flash or Flex application.

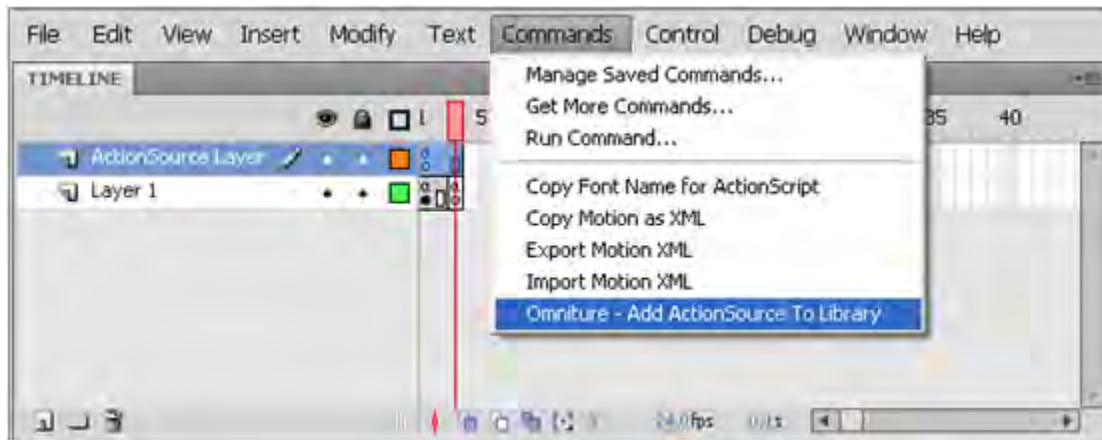
- [To install the ActionSource component into a Flash application](#)
- [To install the ActionSource Component into a Flex project](#)

### To install the ActionSource component into a Flash application

1. Close any open instances of Adobe Flash.
2. Launch the `ActionSource.mxp` file that you downloaded from the SiteCatalyst Code Manager.

For more information, see ["Download ActionSource Code" on page 16](#).

3. The Adobe Extension Manager installs the ActionSource component into your Flash environment.  
You must have Adobe Extension Manager installed to install `ActionSource.mxp`. Without the Extension Manager, your Flash development environment cannot recognize the file correctly.
4. Launch Adobe Flash, then open the application where you want to install the ActionSource component.
5. Click **Commands > Omniture - Add ActionSource To Library**.



Alternatively, you can open the Component pane and drag an instance of the ActionSource component into the application's library.

### To install the ActionSource Component into a Flex project

1. In FlexBuilder, select **Project > Properties > Build Path**.
2. Specify the path to the `ActionSource.swc` file that you downloaded from the SiteCatalyst Code Manager.  
For more information, see ["Download ActionSource Code" on page 16](#).

## 2.3 ActionSource Implementation with Flash

Depending on your needs, you can implement ActionSource in the following ways. This section includes the following topics:

- [ActionSource Data Tracking Options](#)
- [Drag-and-Drop Implementation](#)
- [ActionScript Implementation](#)

---

**NOTE:** The following sections illustrate the use of ActionSource with Adobe Flash CS4. If you use a different version of Flash, your installation and configuration process might vary.

---

### ActionSource Data Tracking Options

ActionSource offers both automated and script-based options for configuring data collection for your Flash applications. Each option has its benefits, and how you choose to implement ActionSource depends largely upon your data collection needs, the complexity of your Flash application, and your level of technical expertise.

---

**NOTE:** To avoid sending multiple transactions for each visitor click that can skew your report data, do not use automated tracking and event scripting simultaneously.

---

## Automated Tracking

Automated tracking lets ActionSource manage the data collection for your Flash application. It is most appropriate for relatively simple Flash applications where you want to collect a static set of data elements. ActionSource provides two variables that enable automated tracking: `autoTrack` and `trackOnLoad`.

ActionSource collects tracking data only when a Flash event triggers an Omniture image request. In some cases, configuring a ClickMap overlay and collecting data about the buttons that visitors click is sufficient. In this case, you can use `autoTrack` to immediately send data for every qualifying click.

If the click is related to a button or qualifying movie clip (Interaction Object), Omniture collection servers record the click as a custom link, with a link name of `ActionSource.AutoTrack:<movieID>`.

If you plan on using a [Drag-and-Drop Implementation](#), Omniture recommends using automated tracking because it eliminates the need for ActionScript programming as part of the implementation process.

However, if you plan on using an [ActionScript Implementation](#), evaluate the use of automated tracking based on your data collection needs. It can simplify your implementation, but might not provide the data collection granularity that you want for more complex Flash applications.

---

**WARNING:** With `s.autoTrack` enabled, every visitor click on a button or qualifying object increases the number of times those variables are sent to SiteCatalyst. Because of this, using AutoTrack can cause an increase in server call volume, and the associated SiteCatalyst fees.

---

## Event Scripting

Event scripting gives you full control over data collection for your Flash application. It is appropriate for more complex Flash applications where you want to explicitly define where and when to collect data in your Flash application's ActionScript. Typically, event scripting does not make sense unless you are using an [ActionScript Implementation](#) of ActionSource.

ActionSource provides two methods that let you initiate an Omniture image request from ActionScript: `track` for page view tracking, and `trackLink` for custom link tracking. To see ActionScript code samples that call ActionSource tracking methods in different scenarios, see "[ActionSource Code Samples](#)" on page 38.

When using event scripting, you typically want to track specific features or functionality, but not necessarily every button click; or you need to customize the data beyond just creating a classification. To address these situations, assign small scripts to each event that you want to track. The script can set specific tracking variables before calling the appropriate tracking method that transmits the data to Omniture collection servers.

---

**NOTE:** In both page view tracking and custom link tracking, ActionSource sends all variables that have values. Be sure to reset or empty all variables that you do not want to include in the data collection prior to calling `track` or `trackLink`.

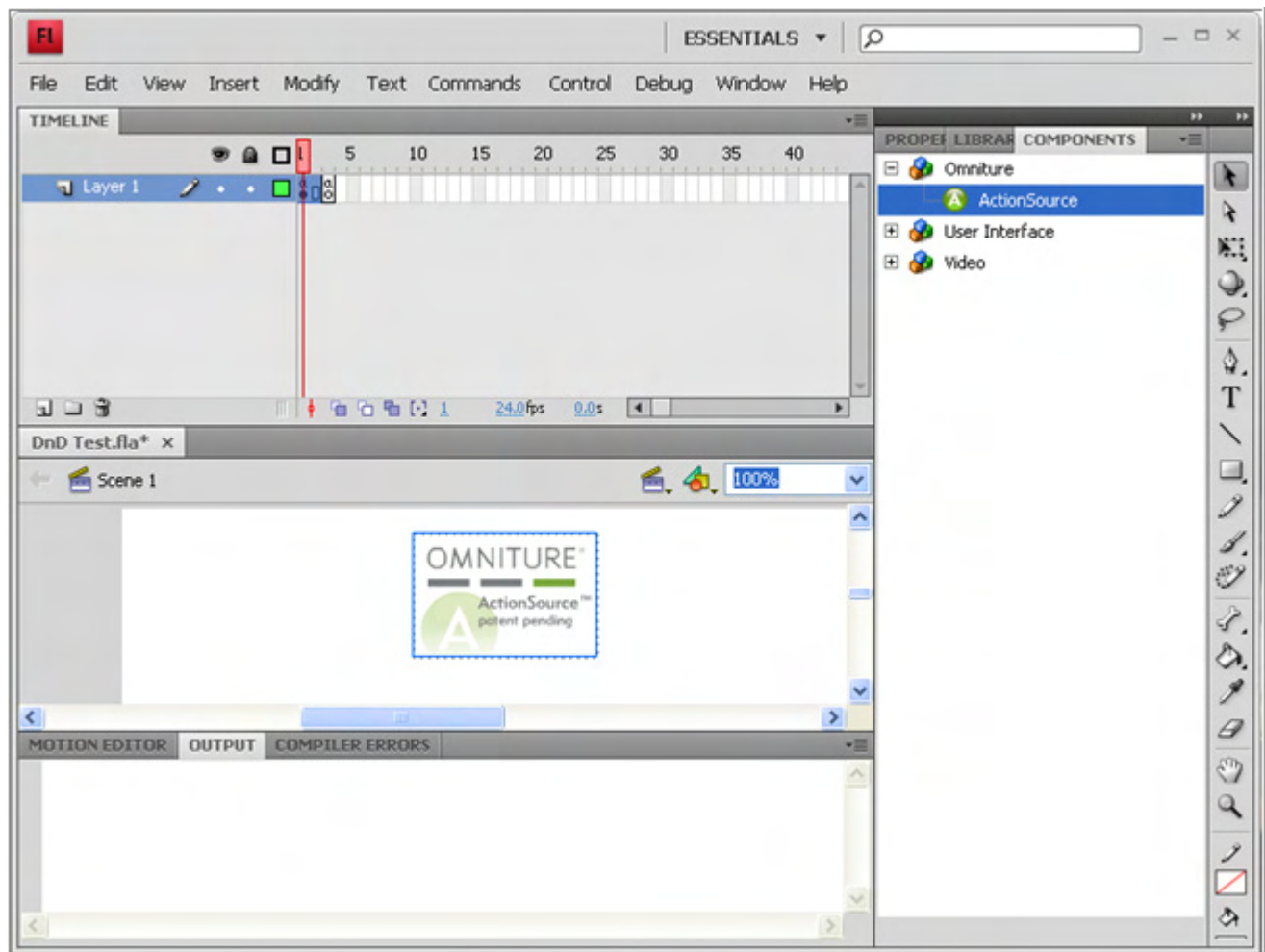
---

## Drag-and-Drop Implementation

The Drag-and-Drop implementation lets you leverage the drag-and drop features of the Flash environment to implement a basic ActionSource implementation with automated ActionSource tracking. This option lets you implement ActionSource without any additional programming.

### To implement ActionSource using the drag-and-drop method

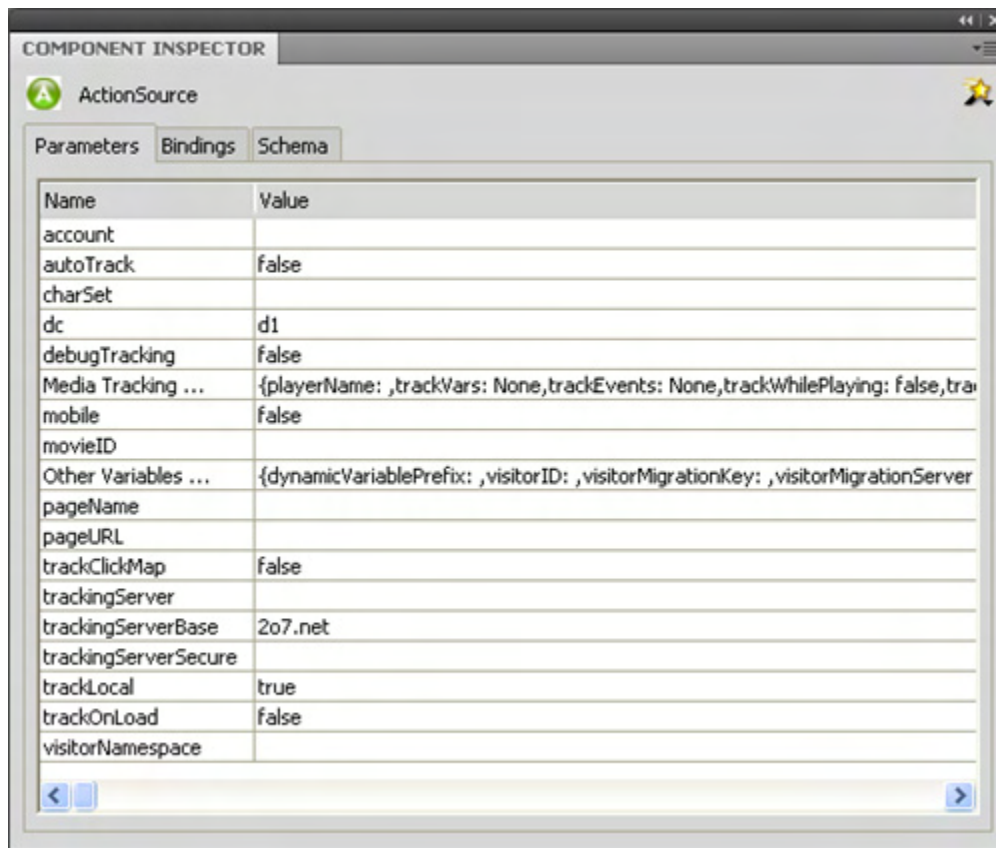
1. Launch Adobe Flash CS4 and open the project where you want to implement ActionSource.
2. In the Timeline pane, select the frame where you want to send tracking data to Omniture collection servers.
3. From the Component pane, or the Library pane, drag the ActionSource component onto the application stage.



**NOTE:** Because this imports the ActionSource component into the current frame, you must drag the ActionSource component onto each frame where you want to have Omniture tracking.

4. Select the ActionSource component, then select **Window > Component Inspector**.

The Component Inspector displays the settings for the ActionSource component.



5. Configure the component settings as needed for your Flash application.

For more information about each of these settings, see ["ActionSource Variables" on page 24](#). Contact your Omniture Account Manager for details about your configuration requirements, if necessary.

**NOTE:** You do not need to name the component instance as part of the drag-and-drop implementation, but doing so lets you later extend the implementation through ActionScript programming, if necessary. When naming the component instance, use "s", for consistency with Omniture's object naming practice.

6. To enable automated Flash tracking, select one of the following component settings:

**trackOnLoad = true:** Automatically sends all configured values to Omniture collection servers when the component loads. For more information, see ["ActionSource Variables" on page 24](#).

**autoTrack = true:** Automatically tracks every visitor click on an interactive object in the Flash application. A visitor click sends data to Omniture as a custom link along with any variables set in the ActionSource component. For more information, see ["ActionSource Variables" on page 24](#).

## ActionScript Implementation

The ActionScript implementation is Omniture's recommended ActionSource implementation method. It gives you more complete control over how and when the Flash application communicates with Omniture collection servers. This options requires you to add, configure, and trigger ActionSource events programmatically through Flash ActionScript.

The ActionScript implementation uses a configuration script (available in both AS2 and AS3 versions) that includes the basic ActionScript necessary to use the ActionSource component in your Flash application. The configuration script automatically adds the ActionSource component to your application, and names it `s`.

Depending on your Flash application structure and your data collection needs, you can add the configuration script to your application in different ways:

- If your Flash application is a single frame, or if you want to collect tracking data only in one frame of a multi-frame application, add the configuration script to the frame directly.
- If you want to collect tracking data in multiple frames, create a layer that exists for the duration of the Flash application, then add the configuration script to this layer. This creates a single instance of the ActionSource component that you can use to collect tracking data from any frame in the Flash application.

**NOTE:** To avoid creating multiple instances of the ActionSource component, do not add configuration script to multiple frames in a Flash application.

### To implement ActionSource using the ActionScript method

1. Determine how you want to add the ActionSource component to your Flash application.
  - a. To add the ActionSource component to a single frame, select the frame (in the Timeline pane) where you want to add ActionSource.

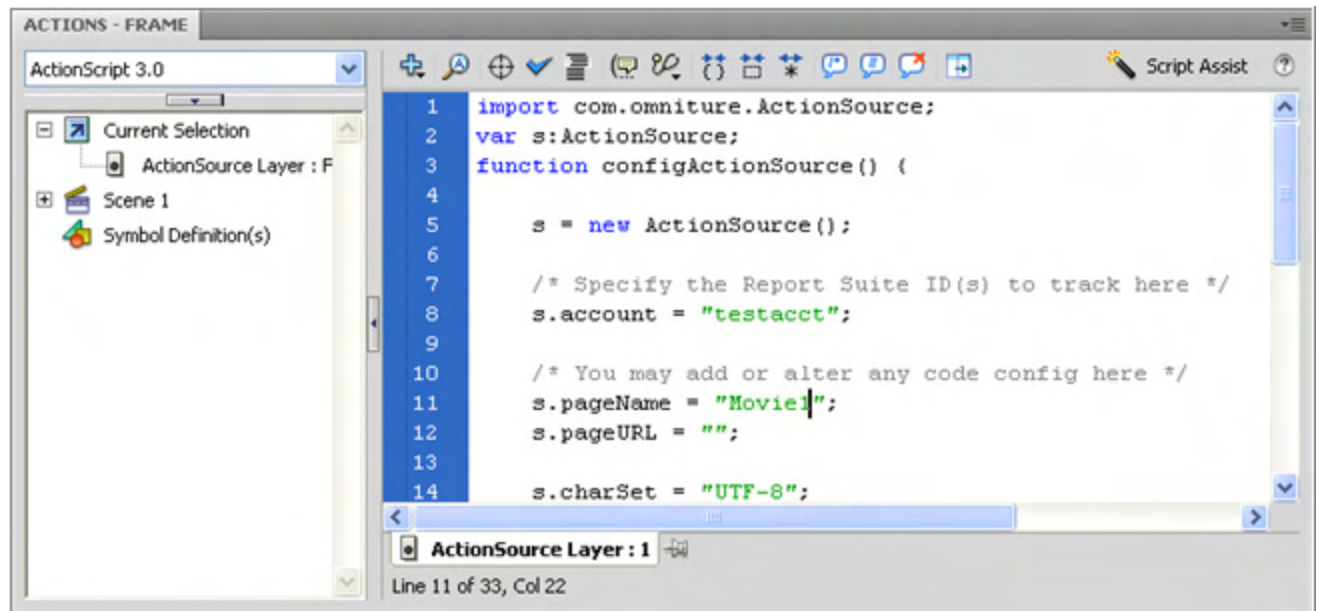


- b. To add the ActionSource component to a layer where it is accessible by all frames in the application, select the layer (in the Timeline pane) where you want to add ActionSource.



2. In the Actions pane, insert the appropriate configuration script.

If necessary, open the Actions pane by selecting **Window > Actions**. You can copy the ActionSource config script from the SiteCatalyst Code Manager. For more information, see ["Download ActionSource Code" on page 16](#).



3. Modify the ActionSource config script as needed for your Flash application.

The SiteCatalyst Code Manager sets certain configuration variables based on your selections when generating the code, but you can add or change configuration variables as needed. Contact your Omniture Account Manager for details about your report suite's configuration requirements, if necessary.

4. In the Timeline pane, select the frame where you want to collect tracking data, then add the appropriate ActionSource method call (`track` or `trackLink`). For more information, see ["Event Scripting" on page 18](#).

Alternatively, you can use automated tracking to let ActionSource manage the collection of tracking data for you. For more information, see ["Automated Tracking" on page 18](#).

This enables ActionScript tracking in the currently selected frame. After ActionSource tracking is enabled, press **Ctrl+Enter** to run the application. Use the Output pane to view the image request that sends data to Omniture collection servers.

## 2.4 ActionSource Implementation with Flex

After installing the ActionSource component into your Flex application (See ["To install the ActionSource Component into a Flex project" on page 17](#)), you can configure it as needed to support your specific data collection needs.

**Figure 2.1: Flex Implementation of ActionSource**

```

15 <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns=""
16     layout="absolute" minWidth="990" minHeight="550"
17     preinitialize="loadStyle()"
18     creationComplete="configActionSource();" styleName="main"
19     pageTitle="FlexStore">
20
21 <mx:Script>
22     <![CDATA[
23         import com.omniture.ActionSource;
24         import mx.collections.IViewCursor;
25         import mx.collections.ArrayCollection;
26         import samples.flexstore.Product;
27         import mx.rpc.events.ResultEvent;
28         import mx.events.StyleEvent;
29         import mx.styles.StyleManager;
30
31         public var s:ActionSource;
32
33         [Bindable]
34
35         private function configActionSource():void {
36             s = new ActionSource();
37             s.debugTracking = true;
38             s.trackLocal = true;
39             s.account = "someaccount";
40             s.pageName = "Some Page Name";
41             s.pageURL = "http://www.somedomain.com";
42             s.autoTrack = false;
43             s.trackClickMap = true;
44             s.movieID = "NameOfFile";
45             s.charSet = "UTF-8";
46             rawChildren.addChild(s);
47         }

```

Step 4 - Call the `configActionSource()` function on application load

Step 1 - Import the ActionSource AS3 Class (Assumes `.swc` was added to the Build Path)

Step 2 - Create a public variable "s" as an ActionSource object

Step 3 - Create a function `configActionSource()` to execute the configuration of the Omniture ActionSource component

### To configure the Flex component

[Figure 2.1](#) provides a Flex code sample that illustrates the steps for adding ActionSource to your Flex application.

1. In the Flex application's main program file, import the ActionSource component.
2. Create a public variable `s` as a new ActionSource object.
3. Create a function `configActionSource` to execute at application load time, and add it to the main code for the project.
4. Add the function call to execute `configActionSource` when you load the application, or as advised by your Omniture implementation consultant.

## 2.5 ActionSource Variables

The ActionSource component supports a variety of variables that let you configure the tracking data to send to Omniture collection servers. [Table 2.1](#) describes the configuration variables available in the ActionSource component. [Table 2.2](#) lists the tracking variables supported by the ActionSource component.

**Table 2.1: ActionSource Configuration Variables**

VARIABLE	REQUIRED	DESCRIPTION
account	Yes	<p><b>Syntax:</b> <code>s.account="mycompanycom";</code></p> <p>The report suite or report suites (multi-suite tagging) that you want to track.</p>
dc	Yes	<p><b>Syntax:</b> <code>s.dc="d1";</code> or <code>"d2";</code></p> <p>Identifies the Omniture data center. Do not change this value unless so instructed by your Omniture representative.</p> <p>d1 = San Jose data center</p> <p>d2 = Dallas data center</p> <p>For reference, the <code>dc</code> variable is also specified in your Web site's core JavaScript file (<code>s_code.js</code>).</p>
visitorNamespace	Yes	<p><b>Syntax:</b> <code>s.visitorNamespace="mycompany";</code></p> <p>Must match the <code>visitorNamespace</code> used in your JavaScript tracking file (<code>s_code.js</code>). If your JavaScript file does not have a <code>visitorNamespace</code> variable defined, do not define it in the ActionSource component. Using the wrong value for <code>visitorNamespace</code> causes incorrect reporting.</p>
pageName or pageURL	Yes	<p><b>Syntax:</b> <code>s.pageName="Some Page Name";</code> <code>s.pageURL="http://www.myurl.com";</code></p> <p>Specify <code>pageName</code> so that ClickMap can properly overlay data.</p> <p>If users do not access the application through a Web browser, specify a <code>pageURL</code> as well. If users access the application through a Web browser, leave <code>pageURL</code> blank and ActionSource automatically sets <code>pageURL</code> to the Web browser's URL.</p>
autoTrack	No	<p><b>Syntax:</b> <code>s.autoTrack=true;</code></p> <p>Enables automatic ClickMap and CustomLink tracking with each click. The <code>autoTrack</code> variable records the default instance name of the selected object as part of the data it collects.</p> <p>When <code>autoTrack</code> is enabled, ActionSource sends all tracking variables with every click event. However, Omniture does not count the <code>pageName</code> variable as a unique page view. This lets you identify the Flash element that causes the most user interaction. For example, by populating <code>prop1</code> you can report on Flash interaction using the <code>prop1</code> value.</p>

VARIABLE	REQUIRED	DESCRIPTION
charSet	No	<p><b>Syntax:</b> <code>s.charSet="UTF-8";</code></p> <p>Sets the character encoding used in the Flash component. For more information about character encoding, see the Omniture SiteCatalyst Implementation Guide, which is available in the SiteCatalyst Help System.</p>
configURL	No	<p><b>Syntax:</b> <code>s.configURL="www.mycorp.com/XMLcode";</code></p> <p>Lets you specify the URL of an XML file, external to the Flash application, that contains the ActionSource configuration variables. This variable exists primarily to support external Flash players (See <a href="#">"External Flash Players" on page 8</a>).</p> <p>For information about the XML structure and tags, see <a href="#">"configURL XML File Sample" on page 40</a>.</p> <ul style="list-style-type: none"> <li>• The XML file does not need to be URL-encoded.</li> <li>• You can use <code>configURL</code> with <code>useExternalVariables</code> by including the <code>configURL</code> variable in a query-string or FlashVar. ActionSource then gets the rest of the configuration variables from the specified XML file.</li> <li>• Before using <code>configURL</code>, you must configure Flash security to support using a configuration file from an external domain. For more information about Flash security, see the following links, which were accurate as of the release of this guide: <ul style="list-style-type: none"> <li><a href="#">Allowing Cross-Domain Data Loading</a> on the Adobe <a href="#">Flash 8 Documentation</a> site.</li> <li><a href="#">Cross-Domain Policy File Specification</a> on the <a href="#">Adobe Developer Connection</a>.</li> </ul> </li> <li>• XML tags are case sensitive.</li> <li>• Do not enclose strings in quotes.</li> <li>• Booleans can be set to <code>true</code> or <code>false</code>.</li> <li>• Use entities for special characters. For example: <code>&amp;lt;</code> for "&lt;", <code>&amp;gt;</code> for "&gt;", and <code>&amp;amp;</code> for "&amp;"</li> </ul>
cookieDomainPeriods	No	<p><b>Syntax:</b> <code>s.cookieDomainPeriods=2;</code></p> <p>Specifies the number of elements in your domain. SiteCatalyst uses this variable when setting the <code>visitorID</code> cookie. For example, <code>s.trackingServer="www.domain.co.uk"</code> has 3 domain periods, and <code>s.trackingServer="www.mycorp.com"</code> has 2 domain periods.</p> <p>For more information, see the Omniture SiteCatalyst Implementation Guide, which is available in the SiteCatalyst Help System.</p>

VARIABLE	REQUIRED	DESCRIPTION
cookieLifetime	No	<p><b>Syntax:</b> <code>s.cookieLifetime="600";</code></p> <p>Specifies the amount of time, in seconds, to store the visitorID cookie.</p> <p>Set <code>cookieLifetime="session"</code> to have the cookie expire with the Web browser session.</p>
currencyCode	No	<p><b>Syntax:</b> <code>s.currencyCode="USD";</code></p> <p>Specifies the currency code that Omniture uses to denominate purchases or currency events collected in the Flash application.</p>
delayTracking	No	<p><b>Syntax:</b> <code>s.delayTracking = 500;</code></p> <p>Specifies a delay, in milliseconds, for the initial Flash image request, following the load of the Flash application.</p> <p>When a Web page contains both Flash applications and Omniture's standard JavaScript tracking, <code>delayTracking</code> prevents the Flash image request from overwriting the JavaScript cookies, which causes inflated tracking of unique visitors.</p> <p>After the cookie is set, transactions can occur without any conflicts, and unique visitor counts remain accurate. However, in the short time the cookie is being set, subsequent transactions result in a new cookie overriding the previously set cookie. Each time a cookie is set, a user is identified as a new visitor.</p> <p>In most cases, a delay of 500 ms is sufficient.</p>
doPlugins	No	<p><b>Syntax:</b> <code>s.doPlugins=function(s) {</code>  <code>    s.campaign=...;</code>  <code>};</code></p> <p>Calls the specified function as part of the Flash image request. For example, you can set an eVar if certain conditions are met.</p>
linkLeaveQueryString	No	<p><b>Syntax:</b> <code>s.linkLeaveQueryString = true;</code></p> <p>Determines whether the query string should be included in the Exit Links and File Download reports. By default, SiteCatalyst excludes query strings from all reports. However, for some Exit Links and File Download reports, the important portion of the URL might be in the query string.</p> <p>For example, the download file name might be defined in the query string, making the query string necessary for an accurate File Downloads report.</p> <p><b>NOTE:</b> If query strings contain session IDs or other personally identifiable information, contact your Omniture representative to have SiteCatalyst strip out those query string elements.</p>

VARIABLE	REQUIRED	DESCRIPTION
linkTrackEvents	No	<p><b>Syntax:</b> <code>s.linkTrackEvents="purchase,event1";</code></p> <p>A comma-separated list of events to send with a custom, exit, or download links.</p> <p>Populate <code>linkTrackEvents</code> in the <code>onClick</code> event. If an event is not specified in <code>linkTrackEvents</code>, ActionSource does not send it to SiteCatalyst, even if it is populated in the link's <code>onClick</code> event.</p>
linkTrackVars	No	<p><b>Syntax:</b> <code>s.linkTrackVars= "events,prop1,eVar49";</code></p> <p>A comma-separated list of variables to send with custom, exit, and download links.</p> <p>Set <code>linkTrackVars=""</code>; to send all variables that have values.</p> <p>To avoid inflation of instances or page views associated with other variables, populate <code>linkTrackVars</code> in the link's <code>onClick</code> event that is used for link tracking.</p>
movieID	No	<p><b>Syntax:</b> <code>s.movieID = "my_movie_id";</code></p> <p>Specifies a unique identifier that ClickMap uses to identify clickable objects. Click data sent to Omniture consists of the <code>movieID:instance_name</code> of the clickable object.</p> <p>If <code>movieID</code> is not set, ActionSource uses the filename from the <code>&lt;embed&gt;</code> (Firefox) or <code>&lt;object&gt;</code> (IE) tag. This is typically the same value as the ID attribute of the <code>&lt;object&gt;</code> tag. For more information about <code>&lt;object&gt;</code> tag structure, see <a href="#">"HTML Object Tag Sample" on page 39</a>.</p> <p><b>NOTE:</b> To aggregate ClickMap tracking for similar objects in separate Flash applications, use the same <code>movieID</code> and <code>instance_name</code> for each object. For example, to aggregate data about all Download Now button clicks across all Flash applications on your Web site, make sure every "Download Now" button uses the same <code>movieID</code> and <code>instance_name</code>.</p>
trackClickMap	No	<p><b>Syntax:</b> <code>s.trackClickMap = true;</code></p> <p>When enabled, sends ClickMap data with <code>track</code> and <code>trackLink</code> functions. For more information about ClickMap, see "Configuring SiteCatalyst ClickMap" in the SiteCatalyst User Guide.</p>
trackDownloadLinks	No	<p><b>Syntax:</b> <code>s.trackDownloadLinks = true;</code></p> <p>Lets you track links to downloadable files in your Flash application. When enabled, ActionSource uses <code>linkDownloadFileTypes</code> to determine which links are downloadable files.</p> <p>Set <code>trackDownloadLinks</code> to <code>false</code> only if there are no downloadable file links your Web site, or if you don't want to track file downloads.</p>

VARIABLE	REQUIRED	DESCRIPTION
trackExternalLinks	No	<p><b>Syntax:</b> <code>s.trackExternalLinks = true;</code></p> <p>Lets you track clicks on links that lead to Web pages external to your Web site. When enabled, ActionSource uses <code>linkInternalFilters</code> and <code>linkExternalFilters</code> to determine if a clicked link is an exit link.</p> <p>Set <code>trackExternalLinks=false</code> only if there are no exit links on your Web site, or if you don't want to track clicks on exit links.</p>
trackingServer trackingServerSecure	No	<p><b>Syntax:</b> <code>s.trackingServer = "metrics.mycompany.com";</code> <code>s.trackingServerSecure = "smetrics.mycompany.com";</code></p> <p>Identifies the collection domain when you are using a first-party cookie implementation. These values should match the collection domain in your <code>s_code.js</code> file, if one exists.</p>
trackingServerBase	No	<p><b>Syntax:</b> <code>s.trackingServerBase=true;</code></p> <p>Enables using <code>omtrdc.net</code> rather than <code>2o7.net</code>.</p>

VARIABLE	REQUIRED	DESCRIPTION
<code>useExternalVariables</code>	No	<p><b>Syntax:</b> <code>s.useExternalVariables=true;</code></p> <p>Lets you dynamically configure ActionSource by pulling variables from a query-string and FlashVars. When enabled, ActionSource extracts variable settings from the query-string or FlashVars automatically. No further configuration is necessary. This variable exists primarily to support the external Flash players (<a href="#">See "External Flash Players" on page 8</a>).</p> <p>Be aware of the following when using <code>useExternalVariables</code>:</p> <ul style="list-style-type: none"> <li><code>useExternalVariables</code> can extract variables from query-strings on the following types of URLs. These URLs are listed in the order of precedence; meaning that an ActionSource Instance URL query-string overrides an ActionSource Parent Instance URL query-string, which overrides a Root Flash movie URL query-string. <ul style="list-style-type: none"> <li><b>The Root Flash Movie URL:</b> The URL referenced by the <code>object</code> or <code>embed</code> HTML tag on the Web page.</li> <li><b>The ActionSource Parent Instance URL:</b> The URL of the parent movie that loads the Flash movie with the embedded ActionSource component. Typically, this is the same as the Root Flash movie URL. This query-string support is useful when using the <code>ActionSourceExtension.swf</code> implementation option.</li> <li><b>The ActionSource Instance URL:</b> The URL of an external movie, being used as a document class, that includes the the ActionSource component, or a class that extends the ActionSource component.</li> </ul> </li> <li>The query-string or Flash Var variable names must be identical to the ActionScript variable names. Use the <code>s</code> object name in all query-strings and FlashVar references (for example, <code>s.account</code>, <code>s.visitorNamespace</code>, <code>s.pageName</code>).</li> <li>URL-encode the variable values.</li> <li>Strings do not need quotes around them.</li> <li>Booleans can be set to <code>true</code> or <code>false</code>.</li> </ul>
<code>visitorMigrationKey</code>	No	<p><b>Syntax:</b> <code>s.visitorMigrationKey=[omniture_provided_value];</code></p> <p>Specifies the migration key used to migrate visitor cookies from a third-party to a first-party cookie implementation.</p> <p>Contact your Omniture representative to get this value.</p>
<code>debugTracking</code>	No	<p><b>Syntax:</b> <code>s.debugTracking=true;</code></p> <p>Lets you view debug information in the Flash editor trace window, or in the Firebug console (Firefox), when available.</p>

VARIABLE	REQUIRED	DESCRIPTION
trackLocal	No	<b>Syntax:</b> <code>s.trackLocal=true;</code> Lets you track Flash applications accessed without a Web browser. When <code>trackLocal=false</code> , ActionSource does not track Flash applications that are not served from an HTTP location.
trackOnLoad	No	<b>Syntax:</b> <code>s.trackOnLoad=true;</code> Automatically sends all configured values to Omniture collection servers when the component loads. If the component is placed on a frame, it sends the data each time the frame loads. If your application uses frames to represent pages, drag a copy of the component onto each keyframe (representing a page view) to track the keyframes as page views.
usePlugins	No	<b>Syntax:</b> <code>s.usePlugins=true;</code> Lets you use <code>s.doPlugins</code> .

The ActionSource component supports the following tracking variables. For more information about each of these variables, see the Omniture SiteCatalyst Implementation Guide, which is available in the SiteCatalyst Help System.

**Table 2.2: Tracking Variables Supported by the ActionSource Component**

pageName	server	zip
pageURL	pageType	events
referrer	visitorID	products
purchaseID	variableProvider	prop1 - prop50 (For example, prop1, prop27)
transactionID	campaign	eVar1 - eVar50 (For example, eVar8, eVar43)
channel	state	hier1 - hier5 (For example, hier1, hier3)

**WARNING:** When capturing variables such as `campaign` that should be set only one time per visit, make sure you reset the variable to NULL (or empty string "") immediately after calling `track` or `trackLink`. Otherwise, these variables continue to increment with each subsequent transaction, which corrupts your tracking data. For example:

```
someLoadFunction():void {s.campaign = "campaign13499"};
  s.pageName = "Application Load";
  s.track();
  s.campaign = "";
}
```

This example assumes that you explicitly set `pageName` on each transaction, so it does not need to be reset.

## 2.6 ActionSource Methods

The ActionSource component supports the following methods for tracking Flash applications:

- [track](#)
- [trackLink](#)
- [setInterface](#)

### track

Sends a standard page view to Omniture collection servers, including any tracking variables that have values. The track method takes no parameters and returns no response.

SYNTAX	PARAMETERS	RESPONSE
<code>s.track();</code>	None	None

### trackLink

Sends custom, download, or exit link data to Omniture collection servers. Track all micro-level activity that should not capture a page view with `trackLink`.

---

**NOTE:** When using `trackLink`, ActionSource sends the `pageName` variable, but SiteCatalyst does not record it as a page view.

---

SYNTAX	PARAMETERS	RESPONSE
<code>s.trackLink(url, type, name);</code>	<p><b>url:</b> (Required) The URL that identifies the clicked link. If you do not want to specify a URL, use an empty string ("" ) for the <code>url</code> parameter. If no URL is specified, <code>trackLink</code> uses the <code>name</code> parameter to identify the clicked link.</p> <p><b>type:</b> (Required) A letter code that specifies the link report that should display the URL or name. Supported values include:</p> <ul style="list-style-type: none"> <li>o: Custom Links report</li> <li>d: File Downloads report</li> <li>e: Exit Links report</li> </ul> <p><b>name:</b> (Required) The name that appears in the link report. If you do not want to specify a name, use an empty string ("" ) for the <code>name</code> parameter. If no name is specified, <code>trackLink</code> uses the <code>url</code> parameter to identify the clicked link.</p> <p>You must provide a value for either <code>name</code> or <code>url</code>.</p>	None

### setInterface

Specifies a display object (typically, the root/stage) on the stage that you can use to find the root of the Flash movie. You can use `s.setInterface` instead of adding an instance of ActionSource to the stage.

**NOTE:** Use `s.setInterface` only when you cannot add ActionSource to the stage.

---

SYNTAX	PARAMETERS	RESPONSE
<code>s.setInterface(interface);</code>	<b>interface:</b> (Required) The stage, or display, object on the stage that you can use to find the root of the Flash movie.	None

# Sample Code

The following code samples help illustrate the implementation concepts related to video tracking and ActionScript tracking.

- [Video Tracking Code Samples](#)
- [ActionSource Code Samples](#)
- [Non-Automatic Media Players](#)

## A.1 Video Tracking Code Samples

The following section provides information and code samples for the various implementation methods available for video tracking:

- [JavaScript Code Placement](#)
- [ActionSource Code Placement](#)
- [Common Code Samples](#)
- [Media.monitor Code Sample](#)

### JavaScript Code Placement

The following code shows an sample s\_code.js with media module code inserted:

---

**NOTE:** Events that you set in video tracking count towards your total event limit.

---

```
s.usePlugins=true
function s_doPlugins(s) {

    /* Add manual calls to modules and plugins here */
}
s.doPlugins=s_doPlugins

/*****Media Module Calls*****/
s.loadModule("Media")
s.Media.autoTrack=true
s.Media.trackWhilePlaying=true;
s.Media.trackVars="" //Enter the variables to be sent withOmniture image requests
s.Media.trackEvents="" //Enter the events to be sent withOmniture image requests
s.Media.trackMilestones="25,50,75" //could also use s.Media.trackSeconds, if desired

s.Media.monitor = function (s,media){ //If Needed
}

/* 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 = "mysitenamespace";
s.trackingServer="metrics.mysite.com" //Use only if using first party cookies
s.trackingServerSecure="smetrics.mysite.com" //Use only if using first party cookies
in conjunction with SSL
```

## Appendix A - Sample Code

---

```
s.dc=122

/***** PLUGINS SECTION *****/
/* Insert any plugins code you want to use here. */

/***** MODULES *****/
/* Module: Media */
//Media module tracking code
s.m_Media_c=""=new
Function(~(`SWhile^sing~;`H~m.ae(mn,1,\"'+p+'\",~='s_media_'+m._in+'_o.'+f~=function(~
){var m=this~}^9 p');p=tcf(o)~o.Get~o;w.percent=(w.offset/
w`J)*100;w.time^sed=i.t~.addEventLis"
+tener~=new Object~'^b_c_il['+m._in+']~x,^v2?p:-1,o)}~set`lout(~'o',

/***** DO NOT ALTER ANYTHING BELOW THIS LINE ! *****/
var s_code='',s_objectID;function s_gi(un,pg,ss){var c=""=fun`j(~){`Ks=^T~.substring(~$v
~.indexOf(~;@u~`d@u~=new Fun`j(~.toLowerCase()~.length~s_c_il['+s^wn+']~=new
Object~};s.~`XMigrationServer~.toU"
+"pperCase~){@u~', '~s.wd~);s.~')q='~var
~ookieDomainPeriods~.location~^KingServer~dynamicAccount~link~s.apv~=new
Array~BufferedRequests~== '~)@ux^Y!Object#OObject.prototype#OObject.prototype[x]~s.m_~"
+"Element~visitor~$o@W(~referrer~s.pt(~.get#8()~)c#A(e){~else ~.la

//Remaining s_code
```

## ActionSource Code Placement

The following AS2 and AS3 code samples demonstrate an [ActionScript Implementation](#) of video tracking that uses a custom media player name, and sends an Omniture image requests when a visitor reaches 25%, 50%, or 75% milestones in the video.

### AS2

```
import com.omniture.AS2.ActionSource;
var s;
function configActionSource() {

    s = new ActionSource();

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

    s.charset = "UTF-8";
    s.currencyCode = "USD";

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

    /*Configure Media Module Functions */
    s.Media.autoTrack= true;
    s.Media.trackWhilePlaying = true;
    s.Media.trackMilestones="25,50,75";//could use s.Media.trackSeconds, if desired
    s.Media.playerName="My Media Player";

    /* Turn on and configure debugging here */
```

## Appendix A - Sample Code

---

```
s.debugTracking = true;
s.trackLocal = true;

/* 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 = "mysitenamespace";
s.trackingServer="metrics.mysite.com" //Use only if using first party cookies
s.trackingServerSecure="smetrics.mysite.com" //Use only if using first party
cookies in conjunction with SSL
s.dc = '122';
}
configActionSource();

s.Media.monitor = function (s,media) {//If Needed
}
```

### AS3

```
import com.omniture.ActionSource;
var s;
function configActionSource() {

    s = new ActionSource();

    /* Specify the Report Suite ID(s) to track here */
    s.account = "myreportsuteID";
    s.charSet = "UTF-8";
    s.currencyCode = "USD";

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

    /*Configure Media Module Functions */
s.Media.autoTrack= true;
s.Media.trackWhilePlaying = true;
s.Media.trackMilestones="25,50,75";//or s.Media.trackSeconds if desired
s.Media.playerName="My Media Player";

    /* Turn on and configure debugging here */
    s.debugTracking = true;
    s.trackLocal = true;

    /* 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 = "yourNamespace";
    s.trackingServer="metrics.mysite.com" //Use only if using first party cookies
    s.trackingServerSecure="smetrics.mysite.com" //Use only if using first party
cookies in conjunction with SSL
    s.dc = '122';

    addChild(s);
}
configActionSource();
s.Media.monitor = function (s,media) {//If Needed
```

---

```
}
```

## Common Code Samples

The following code examples demonstrate the use of video variables and methods that you can use in both JavaScript and ActionSource implementations:

### Auto Track Alone

```
s.Media.autoTrack = true
s.Media.playerName = "Custom Player Name"
```

### Auto Track with Milestones at the 25%, 50%, 75%

```
/* Do not use Media.trackMilestones with Media.trackSeconds*/
s.Media.autoTrack = true
s.Media.playerName = "Custom Player Name"
s.Media.trackWhilePlaying = true
s.Media.trackMilestones = "25,50,75"
```

### Auto Track with Track Seconds Set to Every 60 Seconds

```
/* Do not use Media.trackSeconds with Media.trackMilestones*/
s.Media.autoTrack = true
s.Media.playerName = "Custom Player Name"
s.Media.trackWhilePlaying = true
s.Media.trackSeconds = 60
```

### Manual Tracking with trackMilestones

```
//This code is in the s_code.js, or configActionSource function (ActionSource implementation)
```

```
s.Media.trackWhilePlaying = true
s.Media.trackMilestones="15,25,35";
```

```
//This code is in the s_code.js, in an independent .js file, or outside of the configActionSource function (ActionSource implementation).
```

```
function startMovie(){
    //s.Media.open(mediaName,mediaLength,mediaPlayerName)
    s.Media.open("Movie Title",100, "Custom Player Name");
    // s.Media.play(mediaName,mediaOffset)
    s.Media.play("Movie Title", 0);
}
```

```
function endMovie(){
    // s.Media.stop(mediaName,mediaOffset)
    s.Media.stop("Movie Title",100);
    // s.Media.close(mediaName)
    s.Media.close("Movie Title");
}
```

### Manual Tracking with trackSeconds

```
//This code is in the s_code.js, or configActionSource function (ActionSource implementation)
```

```
s.Media.trackWhilePlaying = true
s.Media.trackSeconds = 60
```

```
//This code is in the s_code.js, in an independent .js file, or outside of the
configActionSource function (ActionSource implementation).
function startMovie(){
    //s.Media.open(mediaName,mediaLength,mediaPlayerName)
    s.Media.open("Movie Title",100, "Custom Player Name");
    // s.Media.play(mediaName,mediaOffset)
    s.Media.play("Movie Title", 0);
}

function endMovie(){
    // s.Media.stop(mediaName,mediaOffset)
    s.Media.stop("Movie Title",100);
    // s.Media.close(mediaName)
    s.Media.close("Movie Title");
}
```

### Media.monitor Code Sample

The following code sample demonstrates using `Media.monitor` to send custom variables to Omniture during video play. To do this you must set `trackWhilePlaying=true`. In this sample code, `Media.monitor` runs every 5 seconds during video play.

```
var tracked10=false; //Variables used as "flags" to prevent the same code from running
var tracked90=false; //twice in the same video play.
```

```
s.Media.monitor = function (s,media) { //Use this code with either JavaScript or Flash.
    // eVar1 = Media Name
    // event1 = Video Begins
    // event2 = Reached 10%
    // event3 = Reached 90%
    // event4 = Reached 100%

    if (media.event == "Open") { //Executes when the video opens.
        s.Media.trackVars = "eVar1,events";
        s.Media.trackEvents = "event1";
        s.events="event1";
        s.eVar1 = media.name;
        s.Media.track(media.name);
    }

    if ((!tracked10) && (media.percent >= 10) { //Executes at 10% complete.
        s.Media.trackVars = "eVar1,events";
        s.Media.trackEvents = "event2";
        s.events="event2"
        s.eVar1 = media.name;
        s.Media.track(media.name);
        tracked10 = true;
    }

    if ((!tracked90) && (media.percent >= 90)) { //Executes at 90% complete.
        s.Media.trackVars = "eVar1,events";
        s.Media.trackEvents = "event3";
        s.events="event3"
        s.eVar1 = media.name;
    }
}
```

```
s.Media.track(media.name);
tracked90 = true;
}

if (media.event == "CLOSE") { //Executes when the video completes.
s.Media.trackVars = "eVar1,events";
s.Media.trackEvents = "event4";
s.events="event4"
s.eVar1 = media.name;
s.Media.track(media.name);
var tracked10=false; //Reset flags values at Media.close if visitors can play
var tracked90=false; //additional videos without reloading the page.
}
};
```

## A.2 ActionSource Code Samples

The following code samples provide information about object structure and code usage.

- [ActionScript Samples](#)
- [HTML Object Tag Sample](#)

### ActionScript Samples

The following ActionScript samples demonstrate setting ActionScript variables in different scenarios. The code samples are organized as follows:

- [Page View Tracking Samples](#)
- [Custom Link Tracking Samples](#)

---

**NOTE:** In both page view tracking and custom link tracking, ActionSource sends all variables that have values. Be sure to reset or empty all variables that you do not want to include in the data collection prior to calling `track` or `trackLink`.

---

### Page View Tracking Samples

#### Send a Page View Only

```
//Set Variables
s.pageName = "Some Page Name";
s.track();
```

#### Send a Page View with Variable Information

```
//Set variable values as needed
s.pageName = "Some Page Name";
s.channel="Some Site Section";
s.prop1="prop_value";
s.eVar12="evar_value";
s.event4="event_value";
s.track();
s.channel="";
s.prop1="";
s.eVar12="";
s.event4="";
```

## Custom Link Tracking Samples

### Send a Custom Link

```
//Set Variables
s.trackLink(s.pageURL,"o","Some Action Name");
```

### Send a Download Link with Variable Information

```
//Set Variables
s.prop1 = "prop val1";
s.prop2 = "prop val2";
s.eVar5 = "eVar val5";
s.events = "event2,event3";

// If linkTrackVars = "", Omniture sends all variables that have a value.
However, you can reset linkTrackVars to send only the variables you want to send.
s.linkTrackVars = "prop1,eVar5,events"; //Prop2 is not sent.
s.trackLink("http://www.downloadURL.com","d","Download File");
```

### Send an Exit Link with no variables

```
//Set Variables
s.channel="";
s.prop1="";
s.trackLink("http://www.someexitdomain.com","e","Some Action Name");
```

## HTML Object Tag Sample

ClickMap uses The basic format of the <object> tag and its required parameters, is as follows.

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/Flash/
swflash.cab#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 required only for ClickMap. Remove these parameters if they interfere with normal production. However, to use ClickMap with a Flash application, you must include this parameter in the page HTML. You can do this by altering the HTML based on a query-string parameter or IP address.

---

## configURL XML File Sample

Use the following structure and tags when creating an external XML file that contains an ActionSource variable configuration:

```
<config> //root tag
  <account>myrsid</account> //Report Suite ID
  <visitorNamespace>mycompany</visitorNamespace> //Used to build the data capture
domain.
  <dc>112</dc> //Data center code (112 or 122). Used to build the data capture domain.
  <pageName>Some Page Name</pageName> //a friendly name for the URL
  <trackOnLoad>true</trackOnLoad> //send config to Omniture when component loads
  <Media> //Sets ActionSource variables. Use the variable name as the XML tag.
    <autoTrack>true</autoTrack>
    <trackWhilePlaying>true</trackWhilePlaying>
    <trackSeconds>15</trackSeconds>
  </Media>
</config>
```

## A.3 Non-Automatic Media Players

`Media.autoTrack` supports several Media players. Using these players limits the amount of additional code you must write to effectively track media player usage. However, you can still track media players that are not supported by `Media.autoTrack`, or if you choose not to use `Media.autoTrack`, by creating functions attached to a player's event handlers that call `Media.open`, `Media.play`, `Media.stop`, and `Media.close` at the appropriate times. For example:

**Video Load:** Call `Media.open` and `Media.play`

**Video Pause:** Call `Media.stop`. For example, if a user pauses a video after 15 seconds, call `s.Media.stop("Video1", 15)`

**Video Resume:** Call `Media.play`. For example, when a user resumes a video after initially playing 15 seconds of the video, call `s.Media.play("Video1", 15)`.

**Video Scrub (slider):** When the user drags the video slider, call `Media.stop`. When the user releases the video slider, call `Media.play`.

**Video End:** Call `Media.stop`, then `Media.close`. For example, at the end of a 100-second video, call `s.Media.stop("Video1", 300)`, then `s.Media.close("Video1")`.

To accomplish this, it is common practice to define four custom functions that you can call from the media player's event handlers. The various parameters passed into `Media.open`, `Media.play`, `Media.stop`, and `Media.close` can come from the player or from custom variables created by the developer. The following pseudocode demonstrates how this might be done:

```
/*Call on video load*/
function startMovie(){
  s.Media.open(mediaName,mediaLength,mediaPlayerName);
  playMovie();
}

/*Call on video resume from pause and slider release*/
function playMovie(){
  s.Media.play(mediaName,mediaOffset);
}
```

## Appendix A - Sample Code

---

```
/*Call on video pause and slider grab*/  
function stopMovie(){  
    s.Media.stop(mediaName,mediaOffset);  
}
```

```
/*Call on video end*/  
function endMovie(){  
    stopMovie();  
    s.Media.close(mediaName);  
}
```