Adobe® Marketing Cloud

AppMeasurement for Flash, Flex, and AIR

Contents

AppMeasurement for Flash, Flex, and AIR

AppMeasurement measures Flash and Flex applications with ActionScript*.

The Adobe AppMeasurement for Adobe Flash, Flex, and OSMF implementation guide describes using the AppMeasurement interface to measure the usage of your Adobe Flash, Flex, and AIR applications and websites.

It lets you capture user activity and forward that data to Adobe data collection servers where it is available for use by Marketing Cloud services, including Analytics reports.

This guide is intended for application developers, and assumes that you are familiar with both implementing Marketing Cloud data collection code, and Flash development.

The general tasks involved in implementing AppMeasurement for Flash, Flex, and AIR include:

Task	Description
1	From the Analytics Code Manager, download the AppMeasurement code necessary to collect data from your Flash or Flex application. For more information, see *Downloading the AppMeasurement Library*.
2	Install the AppMeasurement component into your Flash or Flex application. For more information, see *Adding the AppMeasurement Library to a Project*.
3	Implement AppMeasurement in your Flash or Flex application. AppMeasurement supports multiple implementation options for Flash applications. For more information, see *Implementing in Flash using ActionScript* and *Implementing in Flash Using Drag-and-Drop*. For more information about implementing AppMeasurement in a Flex application, see *Implementing in Flex*.
4	Use the AppMeasurement variables and methods to configure AppMeasurement to gather the proper data, and transmit that data to Adobe data collection servers at the appropriate times. For more information, see *AppMeasurement Library Reference*.
5	If you are delivering Flash video on a Web page, or as part of your Flash application, you can use AppMeasurement to track video usage. For more information, see *Measuring Flash and OSMF Video*.

Downloading the AppMeasurement Library

AppMeasurement is a component that you compile with your Flash or Flex application.

The AppMeasurement component acts as a code library for your Flash or Flex application, similar to `s_code.js` for standard JavaScript tracking.

Adobe distributes the AppMeasurement 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 Analytics Admin Tools.

To download AppMeasurement code

1. Log in to*Reports & Analytics*.
2. Click **Admin** > **Code Manager**.

3. Click the download for Flash, Flex, & AIR.
4. Add the `s.tracking` and `s.trackingServerSecure` variables to the configuration script. These values should match what is in your core `s_code.js` file.

Adding the AppMeasurement Library to a Project

Before using AppMeasurement in your media application, you must install the AppMeasurement component into your Flash or Flex project.

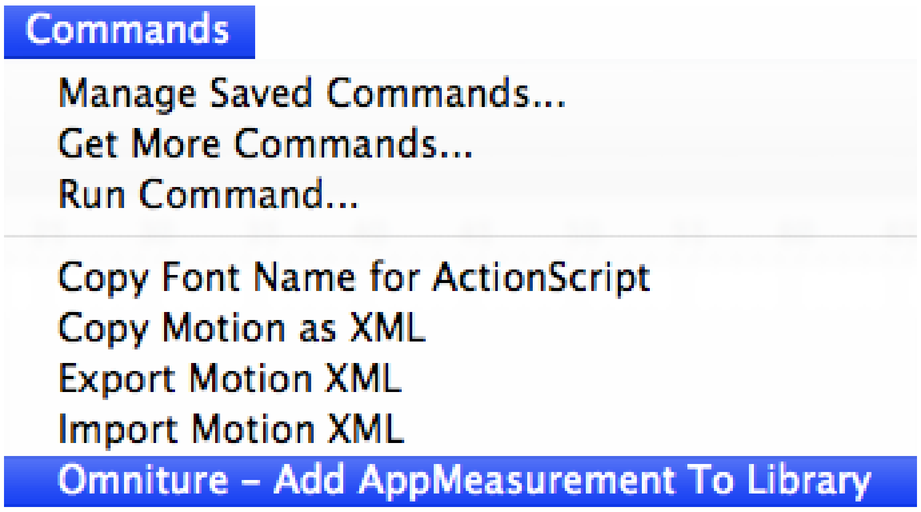
- *To Install the AppMeasurement Component into a Flash Project*
- *To Install the AppMeasurement Component into a Flex Project*

To Install the AppMeasurement Component into a Flash Project

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

   For more information, see *Downloading the AppMeasurement Library*.

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



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

To Install the AppMeasurement Component into a Flex Project

1. In FlexBuilder, select **Project** > **Properties** > **Build Path**.
2. Specify the path to the `AppMeasurement.swc` file that you downloaded from the Analytics Code Manager.

   For more information, see *Downloading the AppMeasurement Library*.

Implementing in Flash using ActionScript

Using ActionScript requires you to add, configure, and trigger AppMeasurement events programmatically through Flash ActionScript.

It gives you more complete control over how and when the Flash application communicates with Adobe data collection servers.

To implement in Flash, select either this method or *Implementing in Flash Using Drag-and-Drop*.

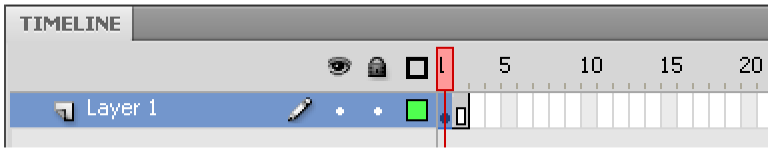
Depending on your Flash application structure and your data collection needs, you can add the configuration script to your application in different ways:

- If your Flash application is a single frame, or if you want to collect data only in one frame of a multi-frame application, add the configuration script to the frame directly.
- If you want to collect 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 AppMeasurement component that you can use to collect data from any frame in the Flash application.

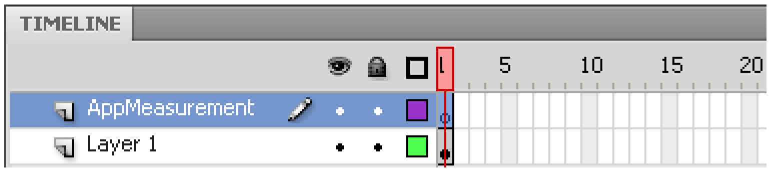
Note: *To avoid creating multiple instances of the AppMeasurement component, do not add configuration script to multiple frames in a Flash application.*

To implement AppMeasurement using the ActionScript method

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



    2. To add the AppMeasurement 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 AppMeasurement.



2. In the **Actions** pane, paste and update the following configuration code:

```
/* Import line for ActionScript 3 */
import com.omniture.AppMeasurement;

var s:AppMeasurement = new AppMeasurement();
addChild(s);

/* Specify the Report Suite ID(s) to track here */
s.account = "myrsid"; // CHANGE THIS!

/* Turn on and configure debugging here - turn this off for production deployment */
s.debugTracking = true;
s.trackLocal = true;
```

```
/* You may add or alter any code config here */
s.pageName = "";
s.pageURL = "";
s.charSet = "UTF-8";
s.currencyCode = "USD";

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

/* 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.*/

/* These values can be copied from your s_code.js file. Your trackign server varies based on
1st or 3rd party cookies, and if you are using SSL. If using first party cookies,
trackingServer will be on your domain, for example metrics.mysite.com */

s.visitorNamespace = "yourNamespace";
s.trackingServer="mycompany.112.2o7.net"; // CHANGE THIS!
s.trackingServerSecure=""; //might not be needed
```

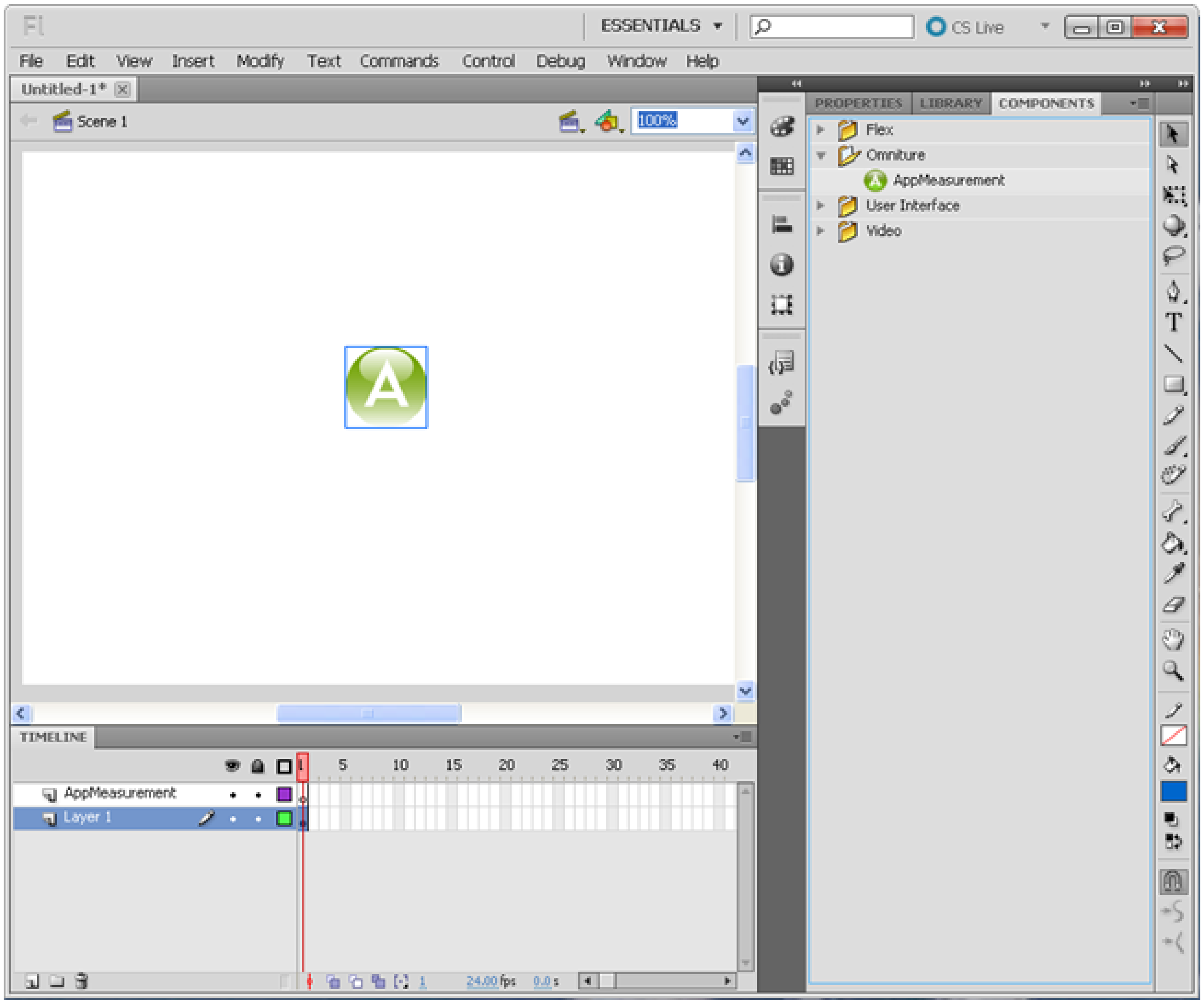
Flash AppMeasurement uses ActionScript, and is configured identically to the JavaScript implementation on your website. The standard *Analytics Variables* are all available in Flash. At a minimum, update `s.account` and `s.trackingServer` with the settings for your company.

Implementing in Flash Using Drag-and-Drop

The Drag-and-Drop implementation lets you leverage the drag-and drop features of the Flash environment to implement AppMeasurement without any additional programming.

To implement in Flash, select either this method or *Implementing in Flash using ActionScript*.

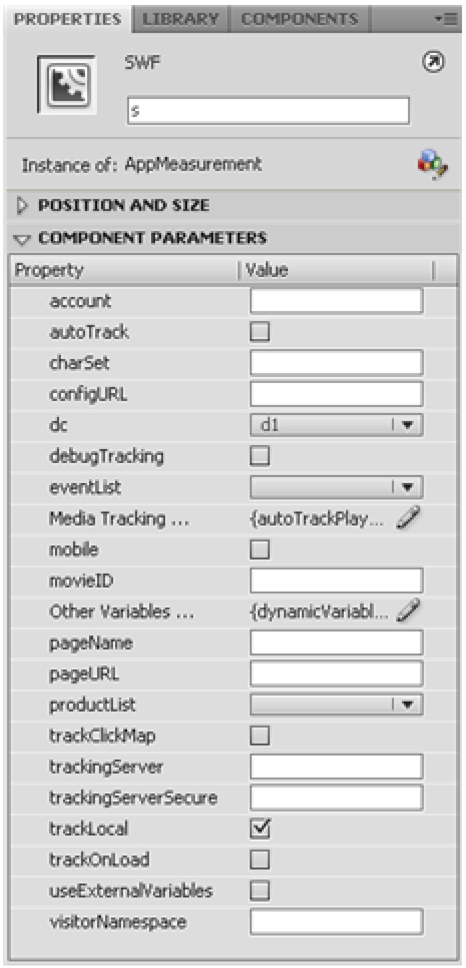
1. Launch Adobe Flash, then open the project where you want to implement AppMeasurement.
2. In the **Timeline** pane, select the frame where you want to send measurement data to Adobe data collection servers.
3. From the **Component** pane, or the **Library** pane, drag the AppMeasurement component onto the application stage.



Note: Because this imports the AppMeasurement component into the current frame, you must drag the AppMeasurement component onto each frame where you want to measure the application.

4. Select the AppMeasurement component, then select **Properties**.

   The **Properties** pane displays the settings for the AppMeasurement component.



s

Instance of: AppMeasurement

Property	Value
account	
autoTrack	☐
charSet	
configURL	
dc	d1
debugTracking	☐
eventList	
Media Tracking ...	{autoTrackPlay...
mobile	☐
movieID	
Other Variables ...	{dynamicVariabl...
pageName	
pageURL	
productList	
trackClickMap	☐
trackingServer	
trackingServerSecure	
trackLocal	☑
trackOnLoad	☐
useExternalVariables	☐
visitorNamespace	

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

   For more information about each of these settings, see *AppMeasurement Library Reference*. Contact your 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 Adobe's object naming practice.*

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

   trackOnLoad = **true:** Automatically sends all configured values to Adobe data collection servers when the component loads. For more information, see *AppMeasurement Library Reference*.

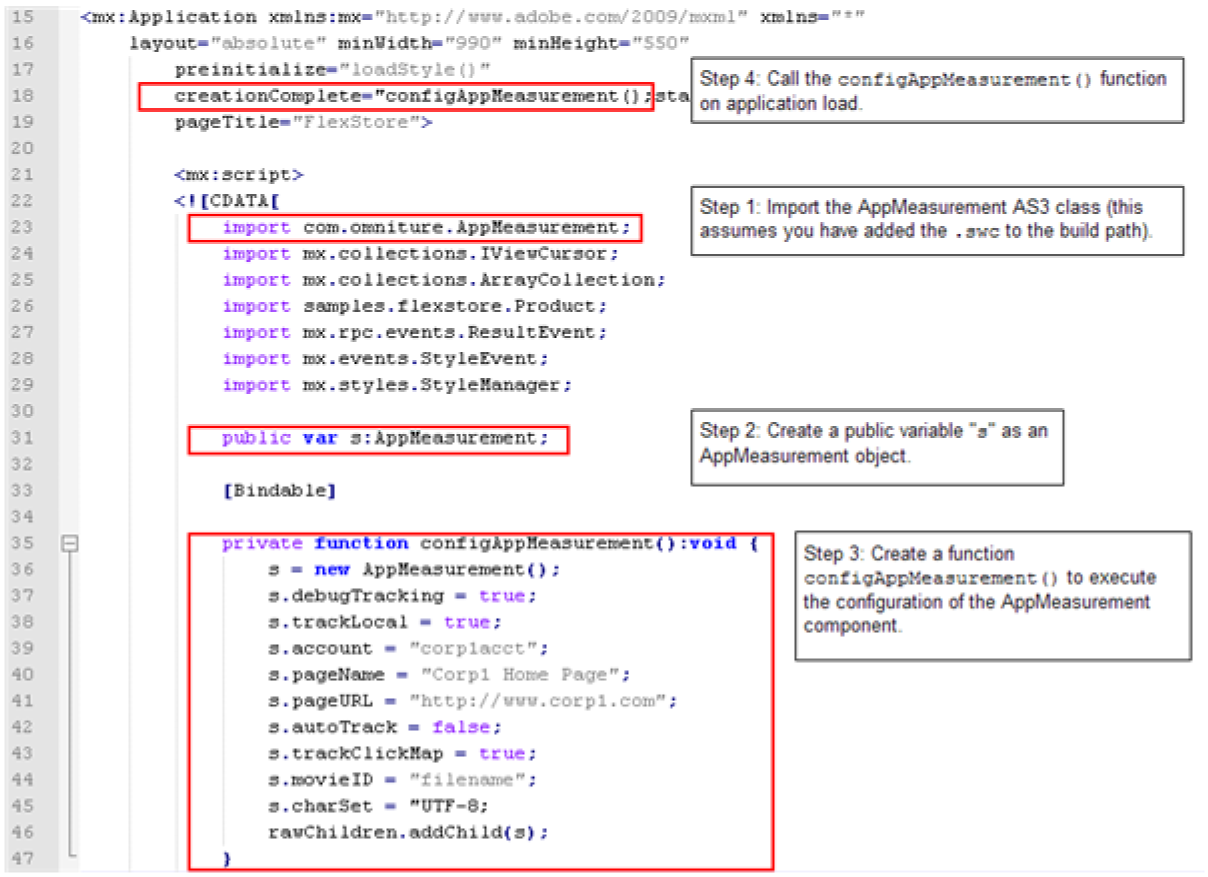
   autoTrack = **true:** Automatically tracks every visitor click on an interactive object in the Flash application. A visitor click sends data to Adobe as a custom link along with any variables set in the AppMeasurement component. For more information, see *AppMeasurement Library Reference*.

Implementing in Flex

After installing the AppMeasurement component into your Flex project, you can configure it as needed to support your specific data collection needs.

See *To Install the AppMeasurement Component into a Flex Project*

Flex Implementation of AppMeasurement



```
<mx:Application xmlns:mx="http://www.adobe.com/2009/mxml" xmlns="*"
    layout="absolute" minWidth="990" minHeight="550"
        preinitialize="loadStyle()"
        creationComplete="configAppMeasurement();sta
        pageTitle="FlexStore">

        <mx:script>
        <![CDATA[
            import com.omniture.AppMeasurement;
            import mx.collections.IViewCursor;
            import mx.collections.ArrayCollection;
            import samples.flexstore.Product;
            import mx.rpc.events.ResultEvent;
            import mx.events.StyleEvent;
            import mx.styles.StyleManager;

            public var s:AppMeasurement;

            [Bindable]

            private function configAppMeasurement():void {
                s = new AppMeasurement();
                s.debugTracking = true;
                s.trackLocal = true;
                s.account = "corp1acct";
                s.pageName = "Corp1 Home Page";
                s.pageURL = "http://www.corp1.com";
                s.autoTrack = false;
                s.trackClickMap = true;
                s.movieID = "filename";
                s.charSet = "UTF-8;
                rawChildren.addChild(s);
            }
```

Step 4: Call the `configAppMeasurement()` function on application load.

Step 1: Import the AppMeasurement AS3 class (this assumes you have added the `.swc` to the build path).

Step 2: Create a public variable "`s`" as an AppMeasurement object.

Step 3: Create a function `configAppMeasurement()` to execute the configuration of the AppMeasurement component.

To configure the Flex component

The example above provides a Flex code sample that illustrates the steps for adding AppMeasurement to your Flex application.

1. In the Flex application's main program file, import the AppMeasurement component.
2. Create a public variable `s` as a new AppMeasurement object.
3. Create a function `configAppMeasurement` to execute at application load time, and add it to the main code for the project.
4. Add the function call to execute `configAppMeasurement` when you load the application, or as advised by your Adobe Implementation Consultant. The standard *Analytics Variables* are all available in Flex.

AppMeasurement Library Reference

AppMeasurement lets you capture events so you can monitor and evaluate the use of your applications.

AppMeasurement provides a common set of variables and methods that you can use to configure application and website measurement.

AppMeasurement Variables Reference

The AppMeasurement component supports a variety of variables that let you configure the data to send to Adobe data collection servers.

The following table describes the configuration variables available in the AppMeasurement component. The next table lists the measurement variables supported by the AppMeasurement component.

The following topics are discussed in this section:

- *Measurement Variables*
- *Context Data Variables*

AppMeasurement Configuration Variables

Variable	Required	Description
account	Yes	Syntax: `s.account="mycompanycom";` The report suite or report suites (multi-suite tagging) that you want to track.
dc	Yes	Syntax: `s.dc="d1";` or `"d2";` Identifies the Adobe data center. Do not change this value unless so instructed by your Account Manager. d1 = San Jose data center d2 = Dallas data center For reference, the dc variable is also specified in your website's core JavaScript file (`s_code.js`).
visitorNamespace	Yes	Syntax: `s.visitorNamespace="mycompany";` Must match the visitorNamespace used in your JavaScript Web beacon (`s_code.js`). If your Web beacon does not have a visitorNamespace variable defined, do not define it in the AppMeasurement component. Using the wrong value for visitorNamespace causes incorrect reporting.
pageName or pageURL	Yes	Syntax: `s.pageName="Some Page Name";` `s.pageURL="http://www.myurl.com";` Specify pageName so that ClickMap can properly overlay data. If users do not access the application through a Web browser, specify a pageURL as well. If users access the application through a Web browser, leave

Variable	Required	Description
		pageURL blank and AppMeasurement automatically sets pageURL to the Web browser's URL.

autoBindVariablesByValue	No	Syntax: `s.autoBindVariablesByValue=true;`
		
		Provides backward compatibility with the pre-version 3.5 variable binding behavior. Default is false.
		
		Prior to version 3.5, variable values that started and ended with curly braces ({ }) were bound to a matching variable. For example:
		
		`var value:int;`
		`value = 20;`
		`s.eVar1 = ({value});`
		
		Previous to version 3.5, or if `autoBindVariablesByValue=true` in later versions, s.eVar1 is 20. If set to false (default), s.eVar1 = {value}.
		
		Starting in version 3.5, we recommend using the bindVariable method to bind variables:
		
		`s.bindVariable("eVar1", "source");`

autoTrack	No	Syntax: `s.autoTrack=true;` Enables automatic ClickMap and CustomLink measurement with each click. The autoTrack variable records the default instance name of the selected object as part of the data it collects. When autoTrack is enabled, AppMeasurement sends all measurement variables with every click event. However, Adobe does not count the pageName variable as a unique page view. This lets you identify the Flash element that causes the most user interaction. For example, by populating prop1 you can report on Flash interaction using the prop1 value. Note that this value is not related to the media.AutoTrack option that is used to enable automatic video tracking.
charSet	No	Syntax: `s.charSet="UTF-8";` Sets the character encoding used in the Flash component. For more information about character encoding, see the *Analytics Implementation Guide*, which is available in the Marketing Cloud help system.
configURL	No	Syntax: `s.configURL="www.mycorp.com/XMLcode";` Lets you specify the URL of an XML file, external to the Flash application, that contains the AppMeasurement configuration variables. This variable exists primarily to support external Flash players. See External Flash Players. • The XML file does not need to be URL-encoded.

Variable	Required	Description
		• You can use configURL with useExternalVariables by including the configURL variable in a query-string or FlashVar. AppMeasurement then gets the rest of the configuration variables from the specified XML file. • Before using configURL, 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: Allowing Cross-Domain Data Loading on the Adobe Flash 8 Documentation site. *Cross-Domain Policy File Specification* on the *Adobe Developer Connection*. • XML tags are case sensitive. • Do not enclose strings in quotes. • Booleans can be set to true or false. • Use entities for special characters. For example: &lt; for "<", &gt; for ">", and &amp; for "&".
cookieDomainPeriods	No	Syntax: `s.cookieDomainPeriods=2;` Specifies the number of elements in your domain. Analytics uses this variable when setting the visitorID cookie. For example, `s.trackingServer="www.domain.co.uk"` has 3 domain periods, and `s.trackingServer="www.mycorp.com"` has 2 domain periods. For more information, see the *Analytics Implementation Guide*, which is available in the Marketing Cloud help system.
cookieLifetime	No	Syntax: `s.cookieLifetime="600";` Specifies the amount of time, in seconds, to store the visitorID cookie. Set `cookieLifetime="session"` to have the cookie expire with the Web browser session.
currencyCode	No	Syntax: `s.currencyCode="USD";` Specifies the currency code that Adobe uses to denominate purchases or currency events collected in the Flash application.
delayTracking	No	Syntax: `s.delayTracking = 500;` Specifies a delay, in milliseconds, for the initial Flash image request, following the load of the Flash application. When a Web page contains both Flash applications and an Adobe Web beacon, delayTracking prevents the Flash image request from overwriting the JavaScript cookies, which causes inflated measurement of unique visitors.

Variable	Required	Description
		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. In most cases, a delay of 500 ms is sufficient.

doPlugins	No	Syntax: `s.doPlugins=function(s) {`
		`s.campaign=...;`
		`};`
		
		Calls the specified function as part of the Flash image request. For example, you can set an eVar if certain conditions are met.

linkLeaveQueryString	No	Syntax: `s.linkLeaveQueryString = true;` Determines whether the query string should be included in the Exit Links and File Download reports. By default, Analytics 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. For example, the download file name might be defined in the query string, making the query string necessary for an accurate File Downloads report. If query strings contain session IDs or other personally identifiable information, contact your Account Manager to have Analytics strip out those query string elements.
linkTrackEvents	No	Syntax: `s.linkTrackEvents="purchase,event1";` A comma-separated list of events to send with a custom, exit, or download links. Populate linkTrackEvents in the onClick event. If an event is not specified in linkTrackEvents, AppMeasurement does not send it to Analytics, even if it is populated in the link's onClick event.
linkTrackVars	No	Syntax: `s.linkTrackVars= "events,prop1,eVar49";` A comma-separated list of variables to send with custom, exit, and download links. Set `linkTrackVars="";` to send all variables that have values. To avoid inflation of instances or page views associated with other variables, populate linkTrackVars in the link's onClick event that is used for link tracking.
movieID	No	Syntax: `s.movieID = "my_movie_id";`

Variable	Required	Description
		Specifies a unique identifier that ClickMap uses to identify clickable objects. Click data sent to Adobe consists of the `movieID:instance_name` of the clickable object. If movieID is not set, AppMeasurement uses the filename from the <embed> (Firefox) or <object> (IE) tag. This is typically the same value as the ID attribute of the <object> tag. To aggregate ClickMap tracking for similar objects in separate Flash applications, use the same movieID and instance_name for each object. For example, to aggregate data about all Download Now button clicks across all Flash applications on your website, make sure every Download Now button uses the same movieID and instance_name.
trackClickMap	No	Syntax: `s.trackClickMap = true;` When enabled, sends ClickMap data with track and trackLink functions. For more information about ClickMap, see Configuring Analytics ClickMap in the *Analytics User Guide.*
trackDownloadLinks	No	Syntax: `s.trackDownloadLinks = true;` Lets you track links to downloadable files in your Flash application. When enabled, AppMeasurement uses linkDownloadFileTypes to determine which links are downloadable files. Set trackDownloadLinks to false only if there are no downloadable file links on your website, or if you don't want to track file downloads.
trackExternalLinks	No	Syntax: `s.trackExternalLinks = true;` Lets you track clicks on links that lead to Web pages external to your website. When enabled, AppMeasurement uses linkInternalFilters and linkExternalFilters to determine if a clicked link is an exit link. Set `trackExternalLinks=false` only if there are no exit links on your website, or if you don't want to track clicks on exit links.
trackingServer trackingServerSecure	No	Syntax: `s.trackingServer = "metrics.mycompany.com";` `s.trackingServerSecure = "smetrics.mycompany.com";` Identifies the collection domain when you are using a first-party cookie implementation. These values should match the collection domain in your `s_code.js` file, if one exists.
useExternalVariables	No	Syntax: `s.useExternalVariables=true;` Lets you dynamically configure AppMeasurement by pulling variables from a query-string and FlashVars. When enabled, AppMeasurement extracts

Variable	Required	Description
		variable settings from the query-string or FlashVars automatically. No further configuration is necessary. This variable exists primarily to support the external Flash players. See External Flash Players.. Be aware of the following when using `useExternalVariables`: • `useExternalVariables` can extract variables from query-strings on the following types of URLs. These URLs are listed in the order of precedence; meaning that an AppMeasurement Instance URL query-string overrides an AppMeasurement Parent Instance URL query-string, which overrides a Root Flash movie URL query-string. **The Root Flash Movie URL:** The URL referenced by the object or embed HTML tag on the Web page. **The AppMeasurement Parent Instance URL:** The URL of the parent movie that loads the Flash movie with the embedded AppMeasurement component. Typically, this is the same as the Root Flash movie URL. This query-string support is useful when using the `AppMeasurementExtension.swf` implementation option. **The AppMeasurement Instance URL:** The URL of an external movie, being used as a document class, that includes the AppMeasurement component, or a class that extends the AppMeasurement component. • The query-string or Flash Var variable names must be identical to the ActionScript variable names. Use the `s` object name in all query-strings and FlashVar references (for example, `s.account,s.visitorNamespace`, `s.pageName`). • URL-encode the variable values. • Strings do not need quotes around them. • Booleans can be set to true or false.
visitorMigrationKey	No	Syntax: `s.visitorMigrationKey= [adobe_provided_value];` Specifies the migration key used to migrate visitor cookies from a third-party to a first-party cookie implementation. Contact your Account Manager to get this value.
debugTracking	No	Syntax: `s.debugTracking=true;` Lets you view debug information in the Flash editor trace window, or in the Firebug console (Firefox), when available.
trackLocal	No	Syntax: `s.trackLocal=true;` Lets you track Flash applications accessed without a Web browser. When `trackLocal=false`, AppMeasurement does not track Flash applications that are not served from an HTTP location.

Variable	Required	Description
trackOnLoad	No	Syntax: `s.trackOnLoad=true;` Automatically sends all configured values to Adobe data 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	Syntax: `s.usePlugins=true;` Lets you use `s.doPlugins`.

Measurement Variables

The AppMeasurement component supports the following measurement variables. For more information about each of these variables, see the *Analytics Implementation Guide*, which is available in the Marketing Cloud help system.

Measurement Variables Supported by the AppMeasurement Component

pageName	server	zip
pageURL	pageType	events
referrer	visitorID	products
purchaseID	variableProvider	prop1 - prop75 (For example, prop1, prop27)
transactionID	campaign	eVar1 - eVar75 (For example, eVar8, eVar43)
channel	state	hier1 - hier5 (For example, hier1, hier3)
contextData		

Note: *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 data. For example:*

```
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.

Context Data Variables

The AppMeasurement Libraries for Flash support collecting data using context data variables. Context Data variables let you define variables on each page that can be read by Processing Rules.

For example, you can define the following Context Data variable:

```
s.contextData['myco.rsid'] = 'value'
```

Using Processing Rules, you can add a condition that checks for a `myco.rsid` context data variable. When this variable is found, you can add an action to copy it to a prop or eVar.

When using trackLink, context data variables are added to linkTrackVars list similar to the following:

```
s.linkTrackVars= "events,contextData.myco.rsid"
```

For example, you might configure linkTrackVars to track events, eVar1, and multiple context data variables similar to the following:

```
s.linkTrackVars= "events,eVar1,contextData.myco.rsid,contextData.myco.pageid"
```

For additional information, see *Context Data Variables* in the *Admin Tools User Guide*.

AppMeasurement Methods Reference

The AppMeasurement component supports the following methods for measuring Flash applications:

- *track*
- *trackLink*
- *setInterface*
- *bindVariable*

track

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

Syntax	Parameters	Response
`s.track();`	None	None

trackLink

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

Note: *When using trackLink, AppMeasurement sends the pageName variable, but Analytics does not record it as a page view.*

Syntax	Parameters	Response
`s.trackLink(url,type,name);`	**url:** (Required) The URL that identifies the clicked link. If you do not want to specify a URL, use an empty string ("") for the url parameter.	None

Syntax	Parameters	Response
	If no URL is specified, trackLink uses the name parameter to identify the clicked link. **type:** (Required) A letter code that specifies the link report that should display the URL or name. Supported values include: **o:** Custom Links report **d:** File Downloads report **e:** Exit Links report **name:** (Required) The name that appears in the link report. If you do not want to specify a name, use an empty string ("") for the name parameter. If no name is specified, trackLink uses the url parameter to identify the clicked link. You must provide a value for either name or url.	

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 AppMeasurement to the stage.

Note: *Use* `s.setInterface` *only when you cannot add AppMeasurement to the stage.*

Syntax	Parameters	Response
`s.setInterface(interface);`	**interface:** (Required) The stage, or display, object on the stage that you can use to find the root of the Flash movie.	None

bindVariable

In version 3.5, this method was added to replace the previous method of variable binding using curly braces { }.

Syntax	Parameters	Response
`s.bindVariable(AppMeasurement_variable, variabletobind);`	**AppMeasurement_Variable:** (Required) The target measurement variable that you want to populate using the value from another variable. For example, eVar1. **variabletobind:** (Required) The variable that you want to use to populate the provided AppMeasurement variable. For example, myvar. `s.bindVariable("eVar1", "source");`	None

Measuring Flash and OSMF Video

Analytics video measurement lets you monitor the number of times that visitors view video files on your website so you can correlate this data with other website analytics.

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

- *Measuring Video in Adobe Analytics using Video Heartbeat (latest version)*
- *Measuring Video in Adobe Analytics using Milestones*

AppMeasurement Code 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*
- *configURL XML File Sample*
- *configURL XML File Sample*

Note: *In both page view tracking and custom link tracking, AppMeasurement 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

```
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="";
   //these variables must be cleared after
   //sending the call or they persist
```

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 = "", Adobe 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, 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 AppMeasurement 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 Adobe when component loads
 <Media>  //Sets AppMeasurement variables. Use the variable name as the XML tag.
  <autoTrack>true</autoTrack>
  <trackWhilePlaying>true</trackWhilePlaying>
  <trackSeconds>15</trackSeconds>
 </Media>
</config>
```