Adobe® Marketing Cloud

Java

Contents

AppMeasurement for Java

Adobe Measurement Library (Java) lets you use Java to monitor and evaluate visitor activity in your native Java application or service.

Once captured, data is available for analysis using Adobe Analytics.

The Adobe Measurement Library (Java) Integration Guide describes using the Measurement Library (Java) to measure the usage of your website. It lets you capture certain types of activity on your website, and forward that data to Adobe data collection servers.

This guide is intended for Web developers, and assumes that you are familiar with both implementing Analytics data collection code, and Java Web development.

Overview

The Measurement Library provides the following key benefits:

- **Streamlined Implementation for Non-JavaScript Tagging:** Provides an alternative to hard-coding JavaScript Web beacons (image requests) on your Web pages using tools similar to `track` and `trackLink`.
- **Integrated Data Tracking:** Lets you incorporate external data into your data collection (external applications, off-line data, and so forth).
- **Tagless Implementation:** Supports server-to-server (Web server to Adobe data collection server) data transfers that can greatly simplify implementation and maintenance of Java application tracking. Server-side measurement also provides a global implementation option independent of the front-end application (mobile application, website, off-line application, and so forth). Server-to-server is enabled using the sendFromServer flag.
- **Visitor ID Management:** Provides additional visitor identification options that improve visitor tracking, including limiting third-party cookie usage, using the carrier's subscriber ID, and using server-side tracking of visitor IDs.
- **Asynchronous Operation:** AppMeasurement for Java sends requests in a separate thread.

Before implementing Measurement Library (Java), you should understand the following concepts and limitations:

- Measurement Library (Java) consists of two JAR files. You can download Measurement Library software from Reports & analytics code manager.
- Adobe provides two versions of the Measurement Library JAR. Which one you use depends on your needs. Where necessary, this document points out differences between the two versions.
  - **AppMeasurement_SE.jar:** A J2SE version that you can customize to your measurement needs. It contains all functionality necessary to track any client environment.
  - **AppMeasurement_EE.jar:** A J2EE version that includes all functionality from the J2SE version, but is primarily for use with Web applications. The J2EE version uses the `HttpServletRequest` object to automatically set default values for certain attributes, such as `referrer` and `user agent`. It also has some additional support methods that simplifies tasks such as managing visitor IDs and using first party cookies.
- To use the functionality of the Measurement Library JAR, simply import it into your Java application.
- To view debugging information, use either the JavaScript debugger or Standard Out for server-side debugging. For more information, see *debugTracking*. If you do not have the JavaScript debugger, contact your Account Manager.
- When using Measurement Library outside of a Web context, data specific to an HTTP request is not available (for example, pageURL, userAgent, and so forth).

Requirements

- **J2SE version:** Requires J2SE 5.0 or later.
- **J2EE version:** Requires J2SE 5.0 or later and the `javax.servlet` package (available in the J2EE APIs).
- Measurement Library was developed and tested using Java 1.5. Earlier versions might work, but are not formally supported.
- Measurement Library assumes that all strings are in UTF-8 format. You must convert any non-UTF-8 strings before using them in Measurement Library object.

Implement Measurement Library (Java)

Once installed, Measurement Library (Java) provides several variables and methods for configuring event tracking on your application.

Note: *When you set a variable directly on the object, the value acts as the default value until you change it on the object. However, you can use the variable overrides parameter to set a one-time-use value that persists only for the current track call, then resets to the default value. You cannot unset variables with variable overrides. For information about using variable overrides, see Implementation Examples.*

Track Config Variables

Variables are typically set before calling one of the track methods.

For more information about these tracking variables (except where noted) see the *Analytics Implementation Guide*, which is available in the Analytics Help System.

pageName	server	zip
pageURL*	pageType	events
referrer*	dynamicVariablePrefix	products
purchaseID	variableProvider	prop (1-75)
transactionID	campaign	eVar (1-75)
channel	state	hier (1-5)
timestamp**		list (1-3)

* (J2EE only) If not explicitly set, Management Library automatically fills in these values.

Methods

Tracking methods provide functionality to configure tracking data and send it to Adobe data collection servers.

Note: *When using variable overrides, use the variable name as the key in a key-value pair.*

Method	Description
`track()`	If `sendFromServer` is enabled, this method makes a direct HTTP request to Adobe data collection servers. This sends any *Track Config Variables* that have assigned values. If `sendFromServer` is not enabled, this method returns an HTML IMG tag web beacon.

Method	Description
`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. 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`.
`clearVars()`	Clears the following track config variables on the object: `channel` `events` `purchaseID` `transactionID` `products` `state` `zip` `campaign` `prop (1-75)` `eVar (1-75)` `hier (1-5)` `list (1-3)`
(J2EE only) `manageVisitorID(boolean persistVisitorIDInLinks, boolean useFirstPartyCookies)` For example: `s.manageVisitorID(true, true);`	Provides advanced visitor ID management capabilities. Because this method might cause an HTTP redirect, call it before any output is rendered to the page. `manageVisitorID` provides the following functionality: • If `setFirstPartyCookie = true, manageVisitorID` attempts to set a first party cookie (using `cookielifetime` and `cookiedomainperiods`), then redirects to the same page to verify that the cookie was set. The redirect occurs only on the first page unless the user has cookies disabled. • Used in conjunction with the rewriteLinks method. If `persistVisitorIdInLinks = true, manageVisitorID` ensures that `visitorId` is on the current URL, by redirecting if necessary. This redirect happens once per visit.

Method	Description
	• If a visitor ID is not already set and not available in the cookie, the mobile subscriber (`mobile = true`), or from the query string, `manageVisitorID` generates a unique `visitorId` for use in subsequent page views.
(J2EE only) `rewriteLinks(String contents)` For example: `output = s.rewriteLinks(output);` `output = s.rewriteLinks(output);`	Modifies link and form content so you can track visitors that have cookies disabled. To do this, `rewriteLinks` rewrites all HTML hyperlinks, appending the `visitorId` to the end of the href. It also adds a hidden `visitorId` form element to all forms. If you are not manually setting `visitorId`, you must call `manageVisitorId` with `persistVisitorIdInLinks=true` before calling `rewriteLinks`. ***Note:*** *Make sure that you close all form tags and that you do not use '>' in the href attributes of your links.*

Plug-In Architecture (Optional)

Measurement Library supports a plug-in architecture that lets you build custom plug-ins that extend Measurement Library (Java) functionality.

Use this functionality to modify Measurement Library (Java) object to manipulate data before sending it to Adobe data collection servers.

To do this, extend the simple class `AppMeasurement.DoPlugins` and overload the method `doPlugins()`. The following example demonstrates how this is done:

```
s = new OmnitureMeasurement();
s.account = "myreportsuiteid";
s.usePlugins = true;.
s.doPlugins = new AppMeasurement.DoPlugins() {
 public void doPlugins(AppMeasurement s) {
        s.prop6 = "plugin prop";
 }
};
s.pageName = "Main View";
s.track();
```

Implementation Examples

The following examples demonstrate some common use cases for App Measurement for Java:

- *(Instance Instantiation) Import the AppMeasurement Library*
- *(Instance Instantiation) Instantiate Instance and Setup Account Configuration Variables*
- *Send a Page View Only*
- *(trackLink) Send a Custom Link*
- *(trackLink) Send a Custom Link with Channel Information*
- *(trackLink) Send an Exit Link with no Variables*

(Instance Instantiation) Import the AppMeasurement Library

J2SE

```
import com.omniture.*;
```

or

```
<%@ page import="com.omniture.*" %>
```

J2EE

```
import com.omniture.*;
```

or

```
<%@ page import="com.omniture.*" %>
```

(Instance Instantiation) Instantiate Instance and Setup Account Configuration Variables

J2SE

```
//Instantiate instance
AppMeasurement s = new AppMeasurement();
//Setup application config variables
s.account = "myaccount";
s.mobile = true;
```

J2EE

```
//Instantiate instance
AppMeasurement s = new AppMeasurement(request, response);
//Setup application config variables
s.account = "myaccount";
s.mobile = true;
```

Send a Page View Only

```
//Set Variables
s.pageName = "Some Page Name";
s.channel = null; //clears any previously set value for channel
s.prop1 = null; //clears any previously set value for prop1
s.track();
```

(trackLink) Send a Custom Link

```
s.trackLink(null, "o", "Some Action Name");
//you can pass in null for linkURL it if doesn't apply
```

(trackLink) Send a Custom Link with Channel Information

```
//Set Variables

s.channel = "Some Site Section";

s.prop1 = "Snoop";

s.trackLink("http://www.somedownloadURL.com", "d", "Some Download");
```

(trackLink) Send an Exit Link with no Variables

```
//Set Variables

s.channel = null;//clears any previously set value for channel

s.prop1 = null;//clears any previously set value for prop1

s.trackLink("http://www.someexitdomain.com", "e", "Some Link Name");
```