ADOBE® DIGITAL MARKETING SUITE

AppMeasurement for Android

IMPLEMENTATION GUIDE

1.0 09092010

Table of Contents

Preface

The Adobe® *AppMeasurement for Android Integration Guide* describes using the AppMeasurement interface to measure the usage of your Google* Android* application. It lets you capture certain types of activity in the application, and forward that data to Adobe data collection servers where it is available for use in SiteCatalyst® reports.

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

Terms and Conditions of Use

Account Support

ClientCare is available to:

- Answer specific product questions.
- Help you utilize SiteCatalyst reports to the greatest effect.
- Resolve any technical difficulties you might have.
- Help you configure SiteCatalyst variables.

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 Account Manager.

Adobe welcomes any suggestions or feedback you might have regarding AppMeasurement for Android or the contents of this guide. Send comments to your Account Manager.

Contact Information

[CORPORATE ADDRESS]	Adobe Systems, Inc. 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]	http://www.omniture.com
[LOG-IN URL]	http://my.omniture.com

AppMeasurement for Android

Adobe® developed AppMeasurement for Android to provide a mechanism to capture usage of native Google Android applications in SiteCatalyst®. AppMeasurement for Android lets you capture application events so you can monitor and evaluate the use of your Android applications.

AppMeasurement for Android leverages the standard Android development tools to let you integrate Android application tracking into both new and existing Android applications.

This section includes the following topics:

- Requirements
- Install AppMeasurement for Android Library
- Implement AppMeasurement for Android
- Supported SiteCatalyst Reports

1.1 Requirements

- AppMeasurement for Android consists of the following software components. You can download the AppMeasurement software from SiteCatalyst's Code Manager. For information about using Code Manager, see "Code Manager" in the *Admin Console User Guide*, which is available in the Digital Marketing Suite help system.

  AppMeasurement_Android.jar: Library designed for use with Android devices and simulators.

- Adobe recommends the following for developing Android applications. Consult the Android Web Site for the latest Android development information.
  - Eclipse IDE for Java Developers (ganymede or later)
  - Android 1.0 SDK, Release 1 or later

NOTE: AppMeasurement for Android is not thread-safe. You can use it in a multi-threaded application, but you cannot access it from multiple threads.

NOTE: When installing an Android application that is enabled for AppMeasurement, end users might see system notifications that the application is accessing certain services related to phone and network privileges.

1.2 Install AppMeasurement for Android Library

Once you have AppMeasurement for Android, complete the following steps to install the AppMeasurement for Android library into an Android application project.

1. In the Eclipse IDE, right-click on the project name.
2. Select **Build Path** > **Add External Archives**.
3. Select `AppMeasurement_Android.jar`.
4. Click **Open**.

Your application can import the classes/interfaces from the `AppMeasurement_Android.jar` library by using `import com.omniture.*`.

Modifying the Project Manifest File

To use AppMeasurement for Android, you must enable both `INTERNET` and `READ_PHONE_STATE` permissions within your `AndroidManifest.xml` file (in the application project directory). For example:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.android.Hello"
      android:versionCode="1"
      android:versionName="1.0.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".HelloActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
</manifest>
```

Viewing AppMeasurement Debugging Information

To view AppMeasurement for Android debugging information, enable the LogCat view provided by the Android SDK.

To enable the LogCat view

1. In the Eclipse IDE, select **Window** > **Show View**.
2. Click **Other**.
3. Expand the Android folder.
4. Select **LogCat**.
5. Click **OK**.
6. In the LogCat view, select the **Create Filter** icon.
7. In the **by Log Tag** field, enter "Omniture AppMeasurement Debug".
8. Click **OK**.

1.3 Implement AppMeasurement for Android

Once installed, AppMeasurement for Android provides the following variables and methods for configuring event tracking on your Android application.

- Application Config Variables
- Track Config Variables
- Methods
- Offline AppMeasurement
- Plug-In Architecture (Optional)
- Implementation Examples

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" on page 9.

Application Config Variables

The following variables are set when the application launches, and typically do not need to change. However, you can change them at any time, if necessary.

VARIABLE	REQUIRED	DESCRIPTION
`account`	Yes	The report suite or report suites (multi-suite tagging) that you wish to track. Separate multiple report suites with commas. You cannot override this variable using variable overrides. For example: `s.account = "myrsid";`
`dc`	No	The Adobe data collection servers that receive the event tracking data. Consult your Account Manager to find out how you should set this value. 112 specifies the San Jose, CA collection servers, while 122 specifies the Dallas, TX collection servers. By default, this variable is 112 (San Jose). For example: `s.dc = "112";`
`linkTrackEvents`	No	A comma-separated list of events that restricts the current set of events for link tracking. If `s.linkTrackEventsEvents` is not defined, AppMeasurement for Android sends all events. For example: `s.linkTrackEvents = "purchase,event1";`
`linkTrackVars`	No	A comma-separated list of variable names that restricts the current set of variables for link tracking. If `s.linkTrackVars` is not defined, AppMeasurement for Android sends all variables with values. Even though `pageName` is sent, its Page View count is not incremented. For example: `s.linkTrackVars = "events,prop1,eVar49";`

`trackingServer`
 	
`trackingServerSecure`
	No	Identifies the collection domain.
		
		If you are using a first-party cookie in other SiteCatalyst implementations, set these variables to the same value as your other implementations.
		
		If you want to use these variables, you must set them both. You cannot set only one tracking server variable.
		
		For example:
		`s.trackingServer = "metrics.corp1.com";`
		`s.trackingServerSecure = "smetrics.corp1.com";`

VARIABLE	REQUIRED	DESCRIPTION
`userAgent`	No	The HTTP User-Agent field that is sent with each track call to Adobe data collection servers. By default, AppMeasurement for Android sets a default value for this variable that includes Android device information and application name and version. If necessary, you can append additional information to this default value. For example: `s.userAgent += "more data";`
`visitorNamespace`	No	The domain used to set cookies. For more information about namespace, see the **SiteCatalyst Implementation Guide**. For example: `s.visitorNamespace = "mycompany";`
`currencyCode`	No	The Currency Code used for purchases or currency events that are tracked in the Android application. For example: `s.currencyCode = "USD";`
`visitorID`	No	Unique identifier for each visitor. Defaults to the Android device ID. For example: `s.visitorID = "12345";`
`linkLeaveQueryString`	No	Enables (`true`) or disables (`false`) preserving the query-string on the URL for links to Web pages from within the Android application. By default, this variable is `false`. You cannot override this variable using variable overrides. For example: `s.linkLeaveQueryString = true;`
`ssl`	No	Enables (`true`) or disables (`false`) sending tracking data via SSL (HTTPS). By default, this variable is `false`. You cannot override this variable using variable overrides. For example: `s.ssl = true;`
`debugTracking`	No	Enables (`true`) or disables (`false`) viewing debug information in the Eclipse console. By default, this variable is `false`. You cannot override this variable using variable overrides. For example: `s.debugTracking = true;`
`usePlugins`	No	Enables (`true`) or disables (`false`) using plug-ins as part of a delegate implementation. By default, this variable is `false`. You cannot override this variable using variable overrides. For example: `s.usePlugins = true;`

Track Config Variables

The following variables are typically set before calling one of the track methods. For more information about these tracking variables, see the *SiteCatalyst Implementation Guide*.

`pageName`	`server`	`zip`
`pageURL`	`pageType`	`events`
`referrer`	`dynamicVariablePrefix`	`products`
`purchaseID`	`variableProvider`	`prop (1-50)`
`transactionID`	`campaign`	`eVar (1-50)`
`channel`	`state`	`hier (1-5)`

Methods

The following tracking methods send data to Adobe data collection servers.

NOTE: When using variable overrides, use the variable name as the key in a key-value pair. Also, use only strings as values in the key-value pairs.

METHOD	DESCRIPTION
`track()`	Sends a standard page view to Adobe data collection servers, along with any Track Config Variables that have values.
`track(Map<String,String> variableOverrides)`	Same as track, except you can pass in a list of key-value pairs that indicates temporary variable overrides for that track call.
`trackLink(String linkURL, String linkType, String linkName)`	Sends custom, download or exit link data to Adobe data collection servers, along with any track config variables that have values. Use `trackLink` to track all activity that should not capture a page view. `trackLink` has the following parameters: **url**: Identifies the clicked URL. If no URL is specified, the name is used. Use this only when linking to a Web page from within your Android application. Otherwise, pass in null for this parameter. **type**: Identifies the link report that will display the URL or name. Supported values include: • “o” (Custom Links) • “d” (File Downloads) • “e” (Exit Links) **name**: The name that appears in the link report. If no name is specified, the report uses the URL. **NOTE:** To collect data, you must specify either the `name` *or* `url` parameter. When not using one of these parameters, pass in null as the value.

`trackLink(String linkURL, String linkType, String linkName, Map<String,String> variableOverrides)`	Same as trackLink, except you can pass in a list of key-value pairs that indicates temporary variable overrides for that trackLink call.

`clearVars()`	Clears the following track config variables on the object:
	`channel`
	`events`
	`purchaseID`
	`transactionID`
	`products`
	`state`
	`zip`
	`campaign`
	`props (1-50)`
	`eVars (1-50)`
	`hiers (1-5)`

TIP: Page view tracking (track) and link tracking (trackLink) sends all variables that have values (non-null, non-empty). You should reset or empty all variables, as needed, before calling track or trackLink.

Offline AppMeasurement

AppMeasurement lets you measure application usage even when the Android device is offline. When enabled, Offline AppMeasurement behaves in the following way:

- The application generates a "hit" (a `track()` or `tracklink()` method call), but the data transmission fails.
- AppMeasurement generates a timestamp for the current hit.
- AppMeasurement buffers the hit data, and backs up buffered hit data to persistent storage to prevent data loss.
- At each subsequent hit, or at 500-millisecond intervals (whichever is shorter), AppMeasurement attempts to send the buffered hit data; maintaining the original hit order. If the data transmission fails, it continues to buffer the hit data. (This continues even while the device is offline.)

To enable offline AppMeasurement, your report suite must be timestamp-enabled. After you make this change, all hits must be time stamped or they will be dropped. If you are currently reporting AppMeasurement data to a report suite that also collects data from JavaScript, you might need to set up a separate report suite for Offline Tracking to avoid data loss.

Offline measurement uses the following methods and variables:

Table 1.1: Offline Measurement Variables

NAME	TYPE / DEFAULT VALUE	DESCRIPTION
`trackOffline`	Boolean / false	Enables/Disables offline AppMeasurement.

NAME	TYPE / DEFAULT VALUE	DESCRIPTION
`offlineLimit`	Integer / 10	Sets the maximum number of hits to buffer. When the number of buffered hits reaches `offlineLimit`, AppMeasurement drops the oldest buffered hit, then buffers the newest hit. Tune this value based on the amount of data (the number of variables) sent with each hit.
`offlineThrottleDelay`	Integer / 0	Specifies a cadence (or delay), in milliseconds, for sending buffered hit data when AppMeasurement detects an active network connection. Doing so mitigates the performance impact of sending multiple hits on the application. For example, if offlineThrottleDelay=1000, and it takes 300ms to send the hit data, AppMeasurement waits 700ms before sending the next buffered hit.
`timestamp`	Integer / 0	Overrides the default timestamp mechanism in AppMeasurement so you can integrate the application's offline measurement with an existing queue system. Adobe recommends letting AppMeasurement manage timestamps, if possible.

Table 1.2: Offline Measurement Methods

NAME	DESCRIPTION
`forceOffline()`	Forces AppMeasurement to behave as if it is offline. This lets you delay sending hit data during a hardware-intensive operation, such as video playback. Rather, you can collect the data offline, then send it once the operation completes.
`forceOnline()`	Returns AppMeasurement to normal operating mode after using the `forceOffline()` method.

Plug-In Architecture (Optional)

AppMeasurement for Android supports a plug-in architecture that lets you build custom plug-ins that extend AppMeasurement for Android functionality. Use this functionality to modify the AppMeasurement for Android object to manipulate data before sending it to Adobe data collection servers.

The following example demonstrates how this is done:

```
AppMeasurement s = new AppMeasurement(getApplication());
s.doPlugins = new AppMeasurement.DoPlugins() {
   public void doPlugins(AppMeasurement other) {
      // custom code
      s.prop6 = "plugin";
   }
};
s.usePlugins = true;
s.ssl = true;
```

```
s.account = "myreportsuiteid";
s.pageName = "Main View";
s.track();
```

Implementation Examples

The following examples demonstrate some common use cases for AppMeasurement for Android:

(Instance instantiation) Import the AppMeasurement Library

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

(Instance instantiation) Declare instance

Declare instance as a class variable.

```
private AppMeasurement s;
```

(Instance instantiation) Instantiate instance and setup account configuration variables

The most likely place to do this is in the onCreate method of your Activity class, but can be done elsewhere. You must pass in the Application instance from your Activity object to the AppMeasurement constructor.

```
public void onCreate(Bundle savedInstanceState) {
   //Instantiate instance
   s = new AppMeasurement(getApplication());
   //Setup application config variables
   s.account = "myaccount";
   s.ssl = true;
}
```

(track) Send a Page View Only

```
public void onPageLoad() {
   //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();
}
```

(track) Send a Page View with Channel Information

```
public void onPageLoad() {
   //Set Variables
   s.pageName = "Some Page Name";
   s.channel = "Some Site Section";
   s.track();
}
```

(track) Send a Page View with Channel and Custom Insight (prop) Information

```
public void onPageLoad() {
   //Set Variables
   s.pageName = "Some Page Name";
   s.channel = "Some Site Section";
   s.prop1 = "Some Value";
   s.track();
}
```

(track) Send pageView data using variable overrides

Specifies a temporary variable value used only for track call.

```
public void onPageLoad() {
   //Create a Map, set values on it corresponding to tracking variable keys, pass it
into track call
   Map<String, String> trackData = new HashMap<String, String>();
   trackData.put("pageName", "Some page name");
   trackData.put("prop1", "Some prop val");
   s.track(trackData);
}
```

(trackLink) Send a Custom Link

```
public void someNonPageViewAction() {
   s.trackLink(null, "o", "Some Action Name");//may pass in 'null' for linkURL if it
doesn't apply
}
```

(trackLink) Send a Custom Link with Channel Information

```
public void someNonPageViewAction() {
   //Set Variables
   s.channel = "Some Application Section";
   s.trackLink(null, "o", "Some Action Name");//may pass in 'null' for linkURL if it
doesn't apply
}
```

(trackLink) Send a Download Link with Channel and Custom Insight (prop) Information

```
public void someNonPageViewAction() {
   //Set Variables
   s.channel = "Some Application Section";
   s.prop1 = "Snoop";
   s.trackLink("http://www.somedownloadURL.com", "d", "Some Action Name");
}
```

(trackLink) Send an Exit Link with no variables

```
public void someNonPageViewAction() {
   //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 Action Name");
}
```

1.4 Supported SiteCatalyst Reports

The following SiteCatalyst reports include data submitted by AppMeasurement for Android. For more information about these reports, see the *SiteCatalyst User Guide*.

Site Metrics Report Support

- Page Views
- Hourly Unique Visitors
- Daily Unique Visitors
- Monthly Unique Visitors
- Yearly Unique Visitors
- Visits
- File Downloads

Finding Methods Report Support

NOTE: These reports are available only if the developer has access to the referrer information when developing the application, which might not be feasible in some circumstances.

- Referring Domains
- Referrers
- Return Frequency
- Daily Return Visits
- Return Visits
- Visit Number

Visitor Profile Report Support

- Domains
- Full Domains
- Top Level Domain
- Languages
- Time Zones
- Visitor Detail
- Last 100 Visitors

GeoSegmentation Report Support

- Country
- State
- City

Mobile Technology Report Support

- Devices
- Manufacturer
- Screen Size
- Screen Height
- Screen Width
- Cookie Support
- Image Support
- Color Depth
- Audio Support
- Video Support

Segmentation Report Support

- Most Popular Pages
- Most Popular Channels
- Most Popular Servers
- Key Visitors
- Pages Viewed by Key Visitors
- Custom Insight
- Custom Links
- Custom Insight Properties 1-50

Pages Report Support

- Page Summary
- Most Popular Pages
- Clicks to Page
- Time Spent on Page
- Pages Not Found

Entries & Exits

- Entry Pages
- Exit Pages
- Exit Links

Complete Paths Report Support

- Full Paths
- Longest Paths
- Path Length
- Time Spent per Visit
- Single-page Visits

Advanced Analysis Report Support

- Previous Page
- Next Page
- Previous Page Flow
- Next Page Flow
- PathFinder
- Fall-out

Purchases

- Conversions & Averages
- Revenue
- Orders
- Units

Shopping Cart

- Conversions & Averages
- Carts
- Cart Views
- Cart Additions
- Cart Removals
- Checkouts

Products Report Support

- Conversions & Averages
- Products
- Cross-Sell
- Categories

Campaigns

- Conversions & Averages
- Campaigns

Customer Loyalty

- Customer Loyalty

Sales Cycle

- Days Before First Purchase
- Days Since Last Purchase
- Visit Number
- Daily Unique Customers
- Monthly Unique Customers
- Yearly Unique Customers

Finding Methods

- Referring Domains
- Original Referring Domains

Visitor Profile

- Countries
- Languages
- Time Zones
- States
- ZIP/Postal Codes
- Domains

Site Path

- Entry Pages
- Original Entry Pages
- Pages per Visit
- Time Spent on Site
- Content Groups

Custom Variables

- Custom E-Commerce Variables 1-50