



OSMF Release Samples

Walkthrough 9: Analytics Integration

Overview:

This walkthrough will demonstrate how to load in dynamic plug-ins, which are plug-ins that are loaded at runtime in the form of a SWF. The walkthrough will illustrate some of the more difficult aspects of working with dynamic plug-ins, such as the need to test on a web server so that the plug-ins can be loaded appropriately. Unlike static plug-ins, dynamic plug-ins, although easier to distribute and load at runtime, have more complex environmental concerns, such as the inability to work locally without a web server, and the possibility requiring a crossdomain.xml file if loading the plug-ins from a different web server or domain from where the player is hosted. One way to handle this complexity issue during the development life cycle is to use static, class-based plug-ins for development, as well as when integration testing begins with web server testing, and then switch it to load as a dynamic plug-in.

The plug-in used for this walkthrough is a custom built plug-in from RealEyes Media that enables Google Analytics tracking from an OSMF player. A key aspect to any complex plug-in is configuration during integration. When working with OSMF plug-ins, in general, configuration is done via meta-data. This walkthrough will demonstrate how to inject custom meta-data for a plug-in.

Objectives:

- Show the proper setup for testing a dynamic plug-in with a web server and Flash Builder
- Load a dynamic (SWF) plug-in at runtime for analytics tracking
- Apply custom meta-data for a plug-in configuration
- Verify that tracking data is being sent

Setup

1. Set the class file as the application file to compile. There are two different ways of doing this depending on which program you are building your application in.
Flash Builder
Right-click the WT09_AnalyticsIntegration.as file and select Set as Default Application from the context menu that appears. This will add the project to the list of compilable applications. A blue dot on the file icon indicates that the file is the default application file.
Flash Professional
Open the OSMF_SampleTemplate.fla and save it as WT09_AnalyticsIntegration.fla.

Then change the document class for the file (in the Properties panel) to WT09_AnalyticsIntegration.

2. Based on your coding environment, there are two ways to output your SWF to a web server.

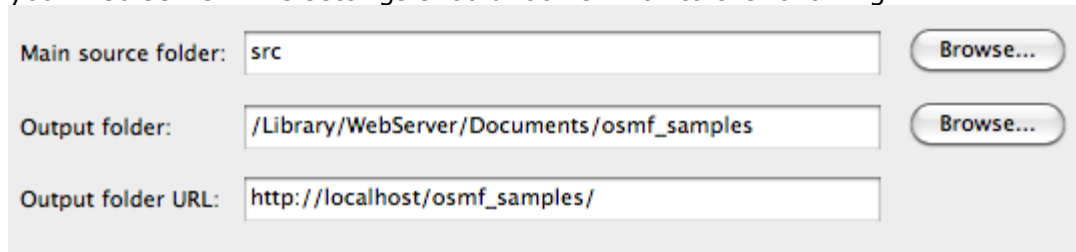
Flash Builder

In the Project Properties (Main menu -> Project -> Properties) set the output folder to your local web server.

Flash Professional

In the Publish Settings (File -> Publish Settings), set the publish directory for your file to a folder on your local web server. Note that you won't be able to do debugging in Flash for projects such as this that use dynamic plug-ins.

3. For Flash Builder, also set the Output folder URL to reflect the same directory on your web server. The settings should look similar to the following:



Main source folder:

Output folder:

Output folder URL:

4. Click OK.
5. Open the file WT09_AnalyticsIntegration.as in the {SAMPLES_PROJECT}/src directory.

NOTE: This file has been provided as a starting point for these walkthroughs.

Loading the plug-in

6. In the initPlayer() method locate the comment that begins "//Marker 1:".
7. Call the loadPlugin() method passing it the static constant GTRACK_PLUGIN.

```
//Marker 1: Load the plugin  
loadPlugin( GTRACK_PLUGIN );
```

8. In the loadPlugin() method, under the "//Marker 2:" comment create a MediaResourceBase variable named pluginResource and set it equal to a new URLResource. Make sure you pass the 'source' argument to the URLResource constructor as the only parameter.

```
//Marker 2: Create the plug-in resource using a static plug-in  
var pluginResource:MediaResourceBase = new URLResource( source );
```

9. Open the AnalyticsSampleData.as class in the com/realeyes/data/config/ directory. The only thing in the class is an XML property that provides the configuration data to the plug-in.

The GTrackPlugin analytics plug-in supports tracking to multiple accounts at the same time, percent and time based tracking, as well as the tracking of any of the

MediaElement's possible traits, such as 'playStateChange', 'volumeChange' etc. The XML in this class has a configuration example of all of the available tracking items that are provided. In this example, we will use the percent-based tracking, and set up 'page' tracking so we can see when a video was watched, as well as how much of that video was watched.

For additional configuration information go to <http://code.google.com/p/reops/wiki/GTrackPlugin>

10. First, you will need to create a Google Analytics account if you don't already have one.
11. Under the "//Marker 4:" comment replace the account text value with your Google Analytics account id ("Web property ID"). Your id should be similar to 'UA-1782464-4'.

```
<!-- //Marker 4: Set your analytics account ID -->
<account><![CDATA[UA-1782464-4]]></account>
```

12. Under the "//Marker 5:" comment set the <url> node's text value to the URL you set up with your Google Analytics account.

```
<!-- //Marker 5: Set the url that you registered with your GA account -
->
<url><![CDATA[http://osmf.realeyes.com]]></url>
```

13. Under the "//Marker 6:" comment, an initial configuration for the percent-based tracking has been provided. This configuration will track the start of the video, as well as its progress at 25, 50, and 75 percent. Notice that we don't have a 100 percent marker.

```
<!-- //Marker 6: Set up the percent based tracking -->
<event name="percentWatched" category="video" action="percentWatched">
<marker percent="0" label="start" />
<marker percent="25" label="view" />
<marker percent="50" label="view" />
<marker percent="75" label="view" />
</event>
```

14. To track the completion of the video we will use the MediaElement's complete event. Under the "//Marker 7:" comment, the complete event tracking configuration has been provided. The name attribute tells the plug-in that we will be tracking the complete event. The category, action, and value attributes map to the Google Analytics tracking values as described here: <http://code.google.com/apis/analytics/docs/tracking/eventTrackerGuide.html>

```
<!-- //Marker 7: Set up the event tracking for the completed event -->
<event name="complete" category="video" action="complete"
label="trackingTesting" value="1" />
```

15. Now that the configuration is set up, we can add the tracking plug-in configuration XML as metadata to the pluginResource by calling the addMetadataValue() method

on the pluginResource object back in the WT09_AnalyticsIntegration.as file. The first parameter should be GTRACK_NAMESPACE, and the second should be the static gTrackPluginConfigXML property on the AnalyticsSampleData object.

```
//Marker 3: Add the metadata necessary to configure the plugin.  
pluginResource.addMetadataValue( GTRACK_NAMESPACE,  
AnalyticsSampleData.gTrackPluginConfigXML );
```

16. Save and run the application. If you are using Flash Builder, the video should play successfully. If you are using Flash Professional, you will need to load the SWF in a web browser using web address to see it run (e.g. http://localhost/osmf_samples/). Use a tool such as Firebug to see the tracking calls as they are sent to your Google Analytics account. NOTE: It can take up to 24 hours for the actual tracking data to show up in your Google Analytics dashboard.



URL	Status	Domain	Size	Timeline
GET _utm.gif?utmwv=4	200 OK	google-analytics.com	35 B	

1 request 35 B

17. Return to the AnalyticsSampleData.as.
18. Under the "//Marker 8:" comment there is an <event /> node with the name of 'pageView'. If this node exists, pageView tracking will be sent to Google Analytics.

```
<!-- //Marker 8: Set up the event tracking for the completed event -->  
<event name="pageView" />
```

19. By default, the URL of the MediaResource will be sent as the 'page' so you can identify the videos. To specify a value that will be sent, we will add metadata specifying the value to send to the MediaResource. If you run the application now and inspect the tracking data that is sent you, should find that a URL that matches

