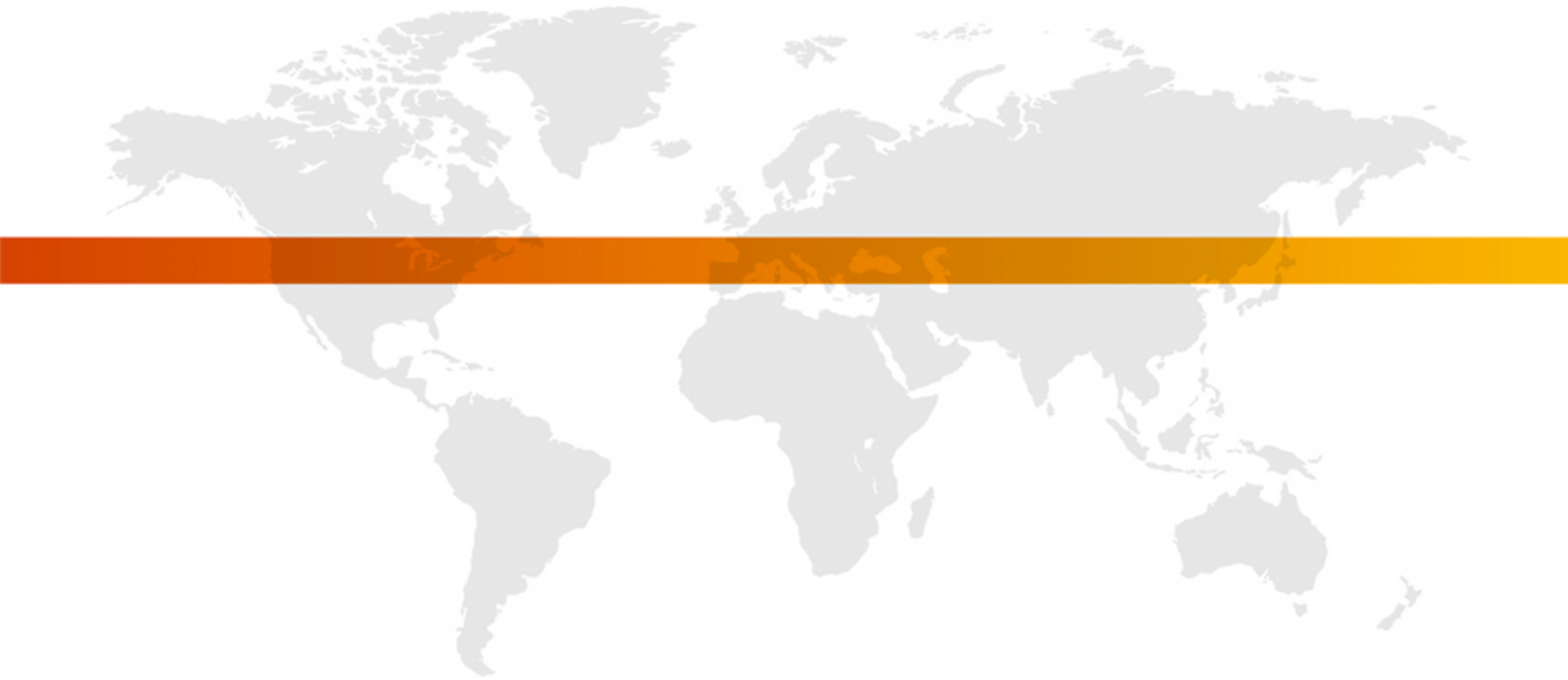
DIGITAL ANALYTIX™

Flash/Flex(AIR) Application Measurement

Tag Specification Document

May 2012



FOR FURTHER INFORMATION, PLEASE CONTACT:

comScore, Inc.
+1 866 276 6972
sdksupport@comscore.com

OVERVIEW

comScore Flash/Flex(AIR) Application measurement solutions are designed to provide easy usage reporting about the application start and duration.

APPLICATION MEASUREMENTS

On the Flash/Flex(AIR) platform, applications may use the *comscore.swc* in order to tag the following events:

- `Start`
- `View`
- `Hidden`
- `Aggregate` (no network transmission)

Start

One very important event is the *Application Start* event, which collects information about the current application version, the last time it was executed, and the last version used if there was an update – among others.

View

You may report a *View* event whenever your application users enter a new context, launch an action or the screen changes sufficiently.

Hidden

Hidden events are meant to report any other action, process or data than *Views* or any other event.

Aggregate

Aggregated events don't generate any network request. Instead, any custom label you send with this event will be taken and stored in memory instead of being transmitted immediately. When more than one aggregated event is fired consecutively, its values are appended or added to those previously stored in patterns we'll later describe.

When the first non-aggregated event occurs, the whole of the aggregated data will be appended and transmitted out.

Aggregation patterns

Label values are always `Strings` but their aggregation pattern may vary depending on their value format, defined as:

Numbers

If a label's value can be casted to number then this label will be summed consecutively, to report the total sum as value.

Lists

We'll define Lists as *"comma-separated lists of strings not containing spaces and with at least one comma even for single-item lists"*. Aggregation here will record the number of times each single value appeared, reporting a concatenation with semicolons of each value plus a colon, plus the total number of times it was received.
Example:

Event 1 `"gems=blue,"`
Event 2 `"gems=blue,red,green"`
Event 3 `"gems=blue,green"`
Result: `gems=blue:3;red:1;green:2` → `gems%3Dblue%3A3%3Bred%3A1%3Bgreen%3A2`

Strings

Disregarding the above types, strings will be defined as *"any group of characters not containing commas, or containing commas and spaces"*. The aggregation method will store each unique value received, reporting a comma-separated list of values.
Example:

Event 1 `"gems=blue"`
Event 2 `"gems=red"`
Event 3 `"gems=green"`
Event 4 `"gems=red"`
Result: `gems=blue,red,green` → `gems%3Dblue%2Cred%2Cgreen`

REQUIREMENTS

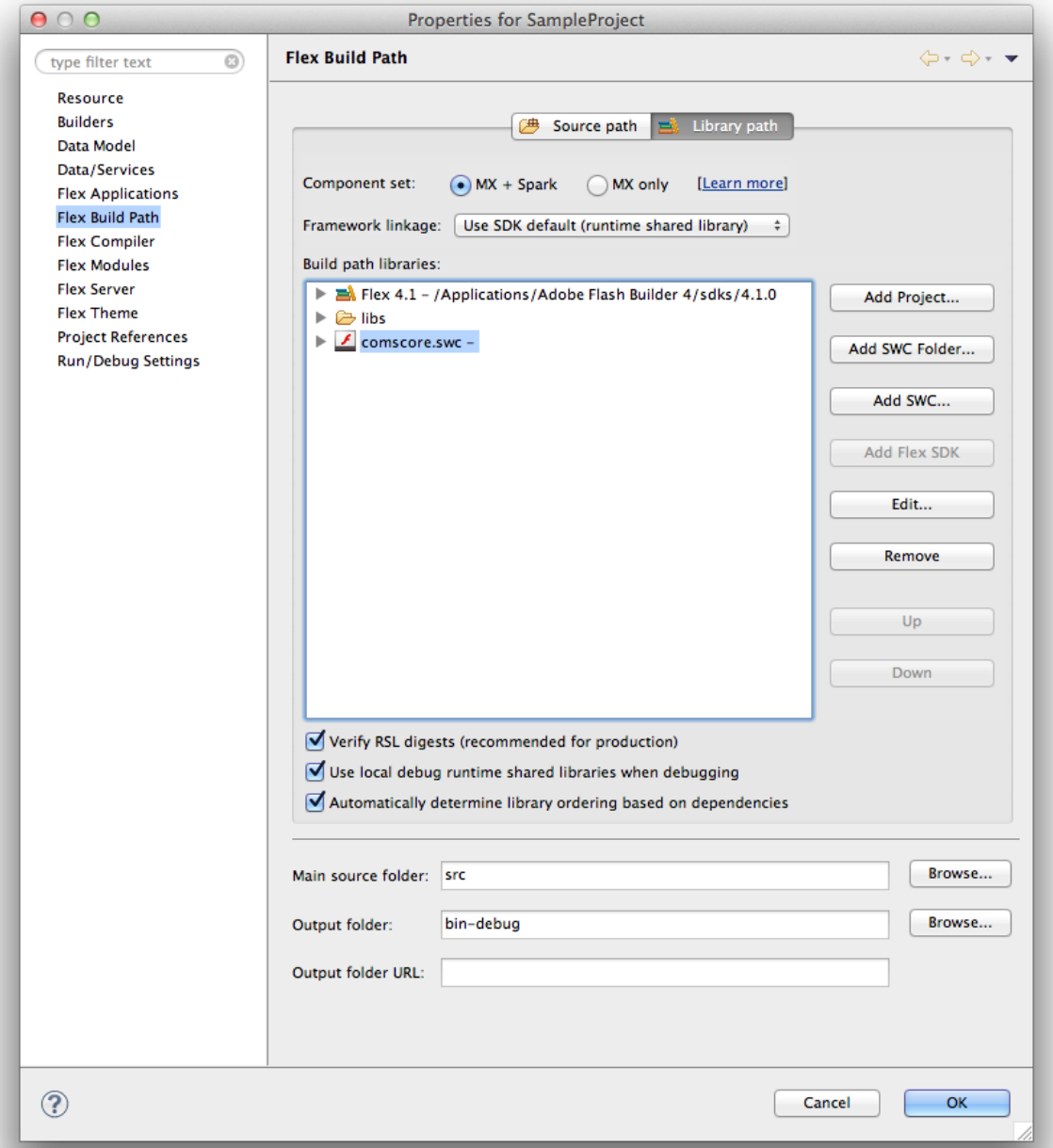
- Your comScore Client ID, a 7-digit number. This parameter is the C2 value.
  - Available in the comScore Direct tag, found at direct.comscore.com
  - Log in to comScore Direct with your client Login ID and password
  - Click "Get Tag"
  - The seven-digit number after `c2:` is your Client ID
  - Contact your client service representative or sdksupport@comscore.com with questions
- A *Publisher Secret*, supplied by comScore
  - The *Publisher Secret* is a text string used to obfuscate your application users' UDID when measurements are sent to comScore servers
  - It is the same for all of your mobile applications, also the *iOS* ones, but unique to you
  - This is required for security, and to protect the privacy of your application's users
  - This will be sent by your comScore Client Service representative
- *comScore SDK*'s `comscore.swc`
  - This is the comScore Flash/Flex(AIR) app tagging SDK. Please request it to your comScore Client Service representative.

IMPLEMENTING DIGITAL ANALYTIX™

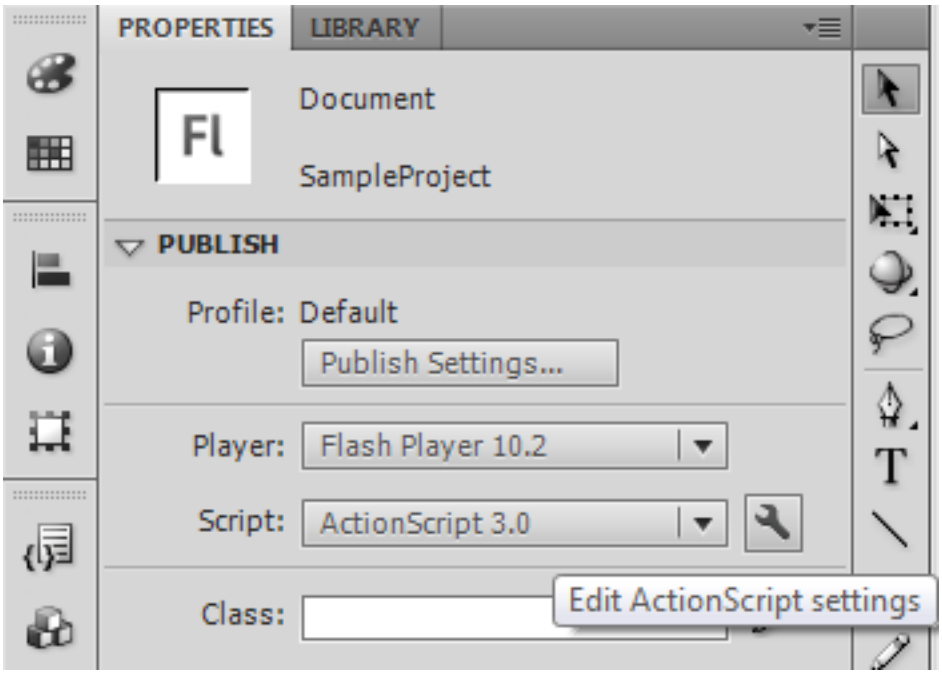
INCLUDE THE LIBRARY

To begin, `comscore.swc` must be included in your Build Path.

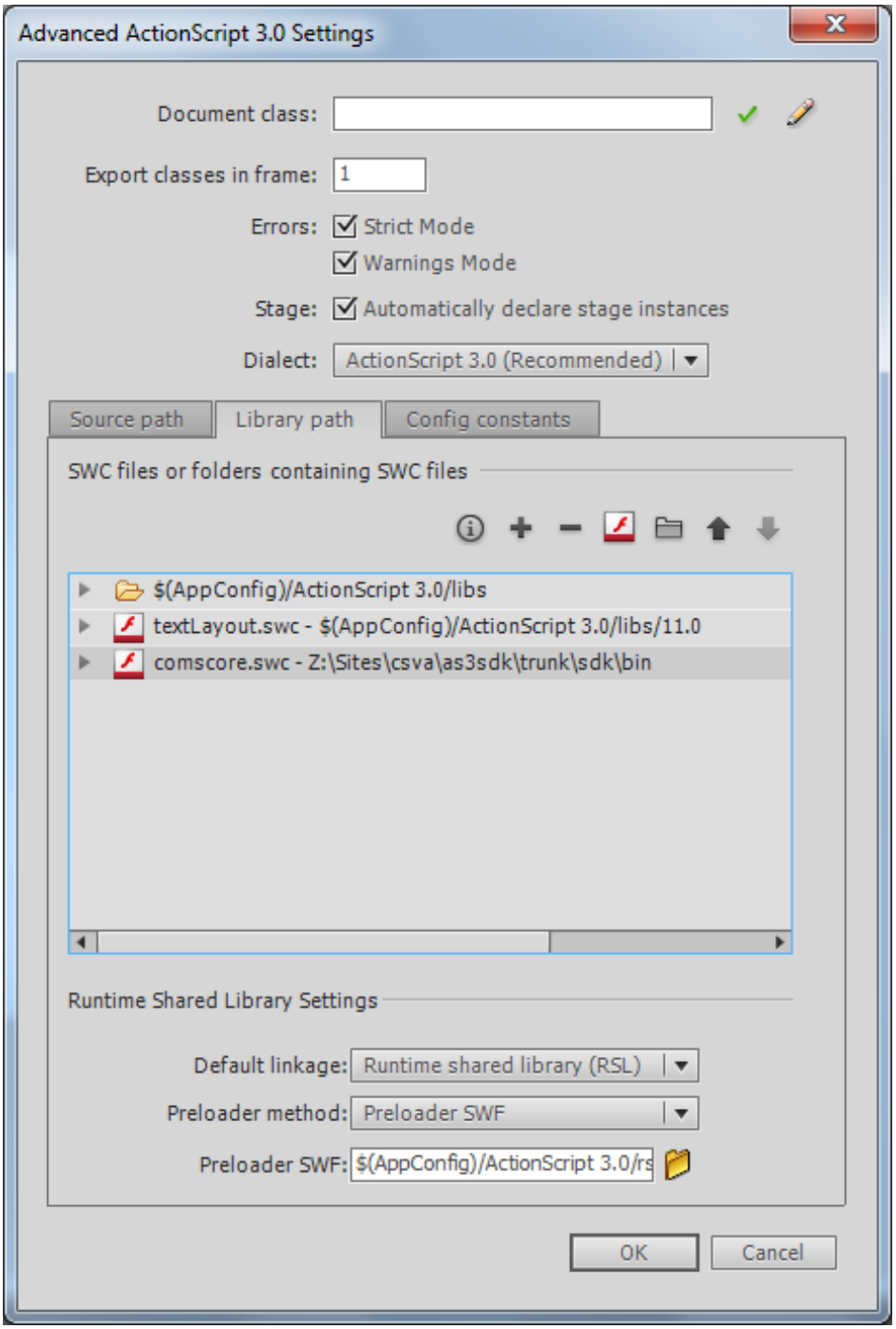
- In Flash Builder:
  - Right-click your project and select *Properties*.
  - Go to the *Flex Build Path* panel.
  - Click *Add SWC…* and navigate to the `comscore.swc` file you were provided.



- In Flash Professional:
  - Click on "Edit ActionScript settings" of your project.



- Go to the *Library Path* panel.
- Click "+" and navigate to the `comscore.swc` file you were provided.



CODE INITIALIZATION

Import statement

For your convenience, import the classes you are going to use afterwards:

```
import com.comscore.analytics.DAx;
import com.comscore.applications.EventType;
```

Set the Application Context

The *Application Context* needs to be set **just once** before any analytic request is made. It is available by calling `this`, which is reference to the current display object.

```
DAx.getInstance().setAppContext(this);
```

Set the Publisher Secret

The *Publisher Secret* is a comScore-supplied `String` used to obfuscate your application users' UDID when measurements are sent to comScore servers. It is the same for all of your applications, but unique to you, and is required to protect the privacy of your application's users.

```
DAx.getInstance().setSalt("PublisherSecret");
```

Set the Pixel URL

The *Pixel URL* is the combination of comScore's base URL plus your *comScore account ID*.
E.g: "`http://b.scorecardresearch.com/p2?c2=1000001`"

```
DAx.getInstance().setPixelURL("http://b.scorecardresearch.com/p2?c2=1000001");
```

DISPATCHING EVENTS

Fire the Application Start

Fire the Application Start event as soon as you are notified of a successful application launch.

```
DAx.getInstance().notifyEventWithLabels(new com.comscore.applications.EventType(
  com.comscore.applications.EventType.START), new Array());
```

Firing Page Views

On web analytics, Page Views usually have a name to identify them. You can set it by passing a custom label *name*.

```
var labels:Array = new Array();
labels["name"] = "homepage";
DAx.getInstance().notifyEventWithLabels(new com.comscore.applications.EventType(
  com.comscore.applications.EventType.VIEW), labels);
```

Firing Hidden Events

Any particular event may be reported with a Hidden Event. Also usually carries a name.

```
var labels:Array = new Array();
labels["name"] = "hiddenEvent";
DAx.getInstance().notifyEventWithLabels(new com.comscore.applications.EventType(
  com.comscore.applications.EventType.HIDDEN), labels);
```

SAMPLES

Setup and Start event in Flash

```
import com.comscore.analytics.DAx;
import com.comscore.applications.EventType;

var dax:DAx = DAx.getInstance().setAppContext(this);
dax.setPixelURL("http://b.scorecardresearch.com/p2?c2=1000001");
dax.setSalt("PublisherSecret");
dax.notifyEventWithLabels(new EventType(EventType.START), new Array());
```

Button event onClick

```
sendView.addEventListener(MouseEvent.CLICK, view_MouseClickHandler);

function view_MouseClickHandler(event:MouseEvent):void
{
 DAx.getInstance().notifyEventWithLabels(new EventType(EventType.VIEW), new Array());
}
```

Setup and Start event in Flex(MXML)

```
<?xml version="1.0" encoding="utf-8"?>
  <mx:WindowedApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
                  xmlns:s="library://ns.adobe.com/flex/spark"
                  xmlns:mx="library://ns.adobe.com/flex/mx" applicationComplete="start()">

<mx:Script>
    <![CDATA[
        import com.comscore.analytics.DAx;
        import com.comscore.applications.EventType;

        private var dax:DAx;

        private function start():void
        {
            dax = DAx.getInstance();
            dax.setAppContext(this);
            dax.setPixelURL("http://b.scorecardresearch.com/p2?c2=1000001");
            dax.setSalt("PublisherSecret");
            dax.notifyEventWithLabels(new EventType(EventType.START), new Array());
        }

    ]]>
</mx:Script>
</mx:WindowedApplication>
```

Button event onClick

```
<?xml version="1.0" encoding="utf-8"?>
      <mx:WindowedApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
                      xmlns:s="library://ns.adobe.com/flex/spark"
                      xmlns:mx="library://ns.adobe.com/flex/mx">
  <mx:Button label="Send View Event" click="notifyView()">
  </mx:Button>
  <mx:Script>
      <![CDATA[
```

```
            import com.comscore.analytics.DAx;
            import com.comscore.applications.EventType;

            private var dax:DAx;

            private function notifyView():void
            {
                DAx.getInstance().notifyEventWithLabels(new com.comscore.applications.EventType(com.comsc
            }

        ]]>
    </mx:Script>
</mx:WindowedApplication>
```

TESTING THE IMPLEMENTATION

As you test your comScore-tagged app internally, comScore servers will collect the measurements. For immediate testing, run your test device's web traffic through Wireshark to view the measurements.

WIRESHARK INSTRUCTIONS

First, download and install Wireshark from http://www.wireshark.org/download.html. Then, perform the following steps each time you restart your Mac and want to Wireshark your test device's app

- Allow Wireshark to sniff network traffic
  - Launch Terminal and run the following commands:
    ```
    cd /dev
    sudo chmod a+r bpf*
    ls -l bpf*
    ```
  - You should see `crw-r--r--` in the first column of the file listing after running the final command above
- Share your Mac's wireless network
  - Connect to the internet using an Ethernet cable
  - Click on the Airport icon at the top right of the screen, select *Create Network*; remember the name you give this network
  - Go to System Preferences, open the *Sharing* pane, highlight (but don't check) *Internet Sharing*
  - Share connection from Ethernet, share computers using AirPort
  - Click *AirPort Options*, and enter the network name you chose above
  - Check the *Internet Sharing* box
  - Click *Start*
  - Connect your test device to the network you just created
- Launch Wireshark
  - Click OK if an error pops up
- Configure Wireshark to look at the `HTTP` traffic from your wireless card
  - In the menu bar, navigate to *Capture* > *Options*
  - Interface `en1` (which is most likely your wireless card)
  - Click *Capture filter*
  - Set the *filter string* to `host b.scorecardresearch.com`

- Click *Start*

If you see `HTTP` analytic requests matching the above pattern, data is being measured correctly.

Once you have verified the SDK is correctly implemented you can add your Flash/Flex widget into HTML page either run it as standalone application.

FAQS

How can I validate my app measurement is working as intended?
There are two ways to validate your app measurements. You could test the measurements yourself by following the Wireshark instructions above. In addition, you may cold-start your app at least 10 times, and email sdksupport@comscore.com with your *c2* and *Publisher Secret*. We will confirm that we received your measurements within two business days.

Do I have to re-submit my app?
Yes. Once you have tested and confirmed your implementation of the comScore measurement code, resubmit your app.

How can I change my application's name that is reported to comScore?
Before you notify any event to the comScore SDK, you can change your reported application name, which defaults to your *swf*'s file name, by calling:

```
DAx.getInstance().setAppName("myApplication");
```

Will placing comScore measurement code slow the application?
No. The comScore code is extremely lightweight, and will not affect the performance of your application.

Will comScore receive measurements if my application user is not connected to the internet?
No. The comScore measurement code only works if the device has internet connectivity.

Will comScore receive measurements if my application user is currently roaming?
If the application launches where your user may incur roaming charges, a warning will appear on their screen. However, if the application starts and the user has internet connectivity we will receive the measurements.

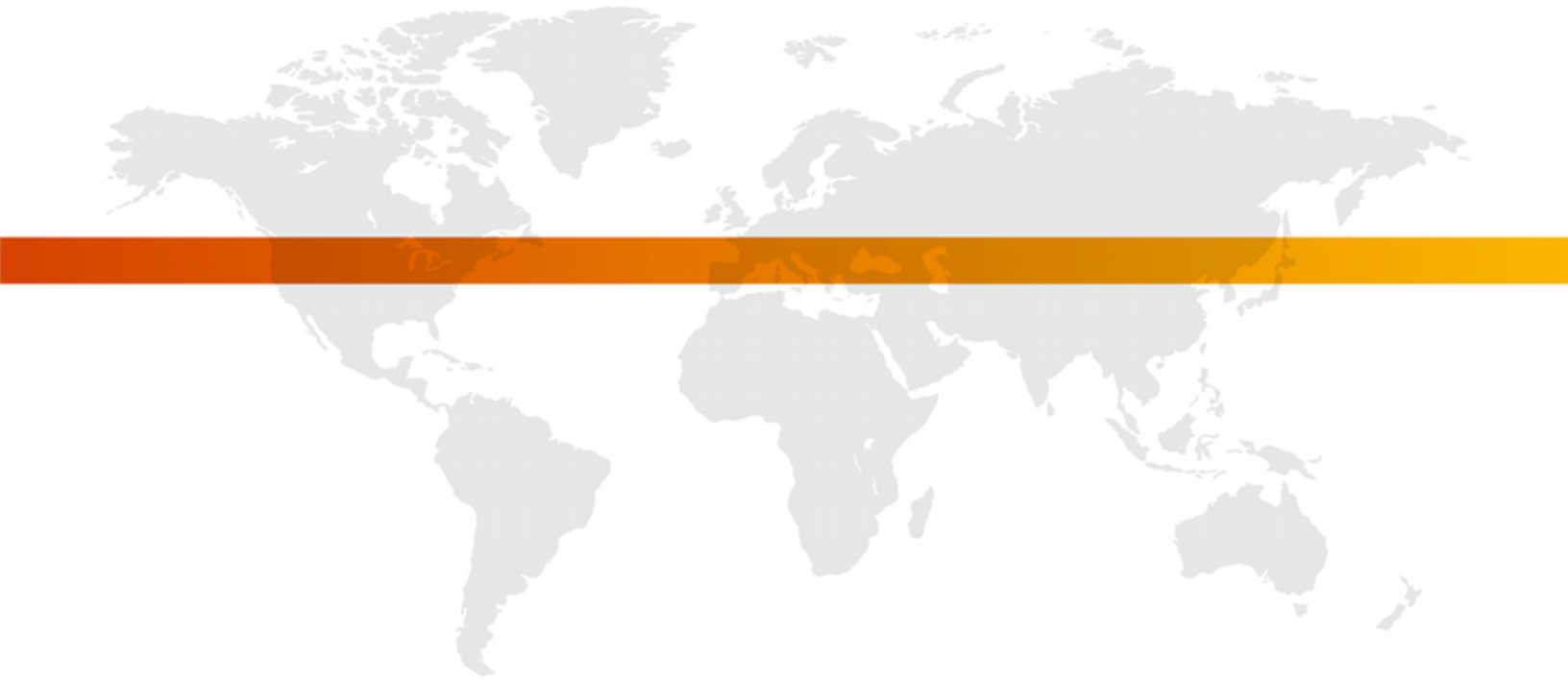
Measuring the digital world.™

DIGITAL ANALYTIX™

Android Mobile Application Measurement

Tag Specification Document

May 2012



FOR FURTHER INFORMATION, PLEASE CONTACT:

comScore, Inc.
+1 866 276 6972
sdksupport@comscore.com

OVERVIEW

comScore Mobile Application measurement solutions are designed to provide easy usage reporting about the application start and duration.

MOBILE APPLICATION MEASUREMENTS

On the Android platform, applications may use the *libComScore* in order to tag the following events:

- `Start`
- `View`
- `Hidden`
- `Aggregate` (no network transmission)

Start

One very important event is the *Application Start* event, which collects information about the current application version, the last time it was executed, and the last version used if there was an update – among others.

View

You may report a *View* event whenever your application users enter a new context, launch an action or the screen changes sufficiently.

Hidden

Hidden events are meant to report any other action, process or data than *Views* or any other event.

Aggregate

Aggregated events don't generate any network request. Instead, any custom label you send with this event will be taken and stored in memory instead of being transmitted immediately. When more than one aggregated event is fired consecutively, its values are appended or added to those previously stored in patterns we'll later describe.

When the first non-aggregated event occurs, the whole of the aggregated data will be appended and transmitted out.

Aggregation patterns

Label values are always `Strings` but their aggregation pattern may vary depending on their value format, defined as:

Numbers

If a label's value can be casted to number then this label will be summed consecutively, to report the total sum as value.

Lists

We'll define Lists as *"comma-separated lists of strings not containing spaces and with at least one comma even for single-item lists"*. Aggregation here will record the number of times each single value appeared, reporting a concatenation with semicolons of each value plus a colon, plus the total number of times it was received.
Example:

Event 1 `"gems=blue,"`
Event 2 `"gems=blue,red,green"`
Event 3 `"gems=blue,green"`
Result: `gems=blue:3;red:1;green:2` → `gems%3Dblue%3A3%3Bred%3A1%3Bgreen%3A2`

Strings

Disregarding the above types, strings will be defined as *"any group of characters not containing commas, or containing commas and spaces"*. The aggregation method will store each unique value received, reporting a comma-separated list of values.
Example:

Event 1 `"gems=blue"`
Event 2 `"gems=red"`
Event 3 `"gems=green"`
Event 4 `"gems=red"`
Result: `gems=blue,red,green` → `gems%3Dblue%2Cred%2Cgreen`

REQUIREMENTS

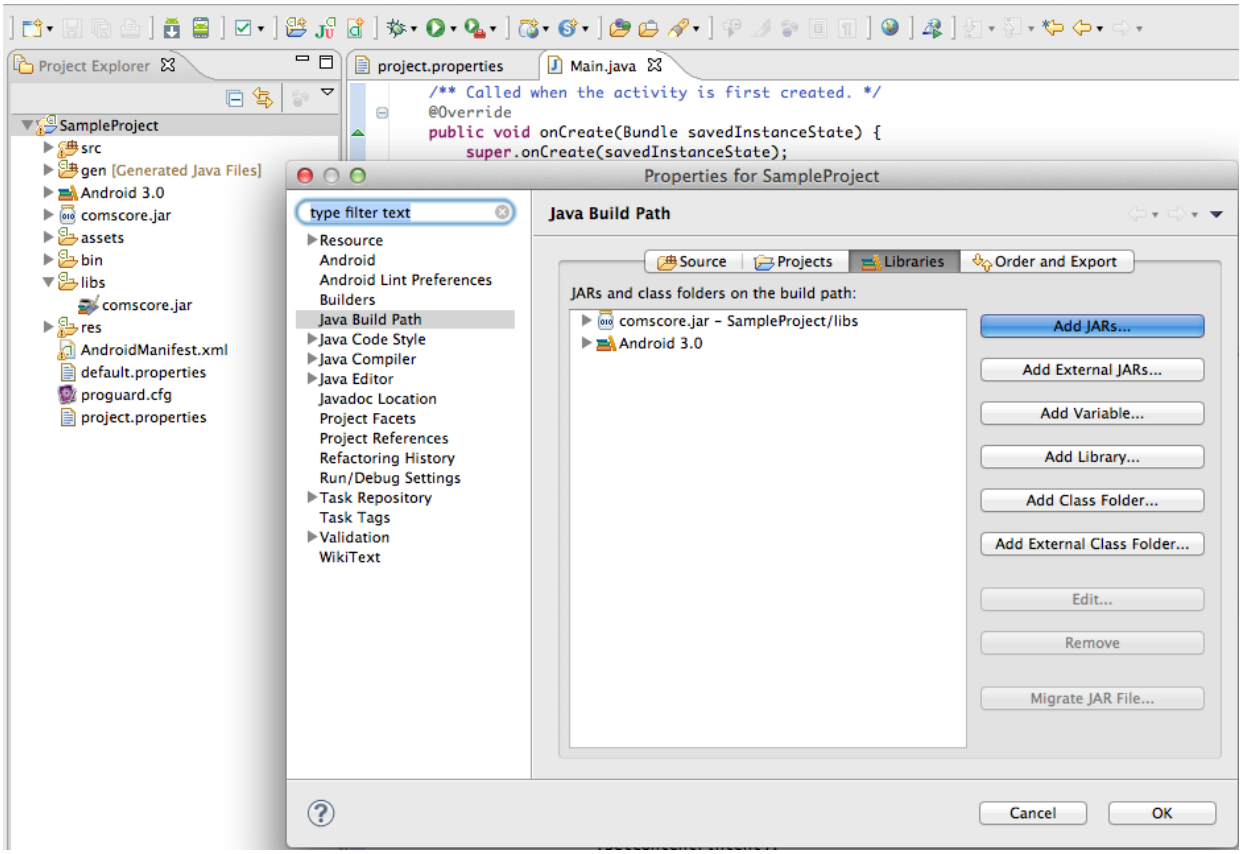
- Your comScore Client ID, a number of at least 7 digits. This parameter is the C2 value.
  - Available in the comScore Direct tag, found at direct.comscore.com
  - Log in to comScore Direct with your client Login ID and password
  - Click "Get Tag"
  - The number after `c2:` is your Client ID
  - Contact your client service representative or sdksupport@comscore.com with questions
- A *Publisher Secret*, supplied by comScore
  - The *Publisher Secret* is a text string used to obfuscate your application users' UDID when measurements are sent to comScore servers
  - It is the same for all of your mobile applications, also the *iOS* ones, but unique to you
  - This is required for security, and to protect the privacy of your application's users
  - This will be sent by your comScore Client Service representative
- *libComScore SDK*'s `comscore.jar`
  - This is the comScore Android app tagging SDK. Please request it to your comScore Client Service representative.

IMPLEMENTING DIGITAL ANALYTIX™

INCLUDE THE LIBRARY

To begin, `comscore.jar` must be moved inside a directory called `libs` within your project location, and included in your Java Build Path.

- In Eclipse:
  - Right-click your project and select *Properties*.
  - Go to the *Java Build Path* panel.
  - Click *Add JARs…* and navigate to the `comscore.jar` file you were provided. It should be inside the `libs` directory.



CODE INITIALIZATION

Import statement

For your convenience, import the classes you are going to use afterwards:

```
import com.comscore.analytics.DAx;
import com.comscore.applications.EventType;
```

Set the Application Context

The *Application Context* needs to be set **just once** before any analytic request is made. It is available in your Activity's `onCreate` method, by calling `getApplicationContext()`.

```
DAx.getInstance().setAppContext(getApplicationContext());
```

Set the Publisher Secret

The *Publisher Secret* is a comScore-supplied `String` used to obfuscate your application users' UDID when measurements are sent to comScore servers. It is the same for all of your applications, but unique to you, and is required to protect the privacy of your application's users.

```
DAx.getInstance().setSalt("PublisherSecret");
```

Set the Pixel URL

The *Pixel URL* is the combination of comScore's base URL plus your *comScore account ID*.
E.g: "`http://b.scorecardresearch.com/p2?c2=1000001`"

```
DAx.getInstance().setPixelURL("http://b.scorecardresearch.com/p2?c2=1000001");
```

Custom Data: Labels

We call *labels* to each `name=value` pair of custom data you send over on each event. You can have any number of those, although there is a hard limit of 2048 bytes for the overall request URL and any exceeding labels will be cut out.

To set application-wise persistent labels, sent for each of your application's events, use the methods available for that purpose. Examples:

```
// Set a single label
DAx.getInstance().setCustomLabel( "labelName", "myValue" );

// Set any number of labels
HashMap<String, String> labels = new HashMap<String,String>();
labels.put("labelName1", "myValue1");
labels.put("labelName2", "myValue2");

DAx.getInstance().setCustomLabels( labels );
```

DISPATCHING EVENTS

Fire the Application Start

Fire the Application Start event as soon as you are notified of a successful application launch.

```
DAx.getInstance().notify(EventType.Start, new HashMap<String,String>());
```

Firing Page Views

On web analytics, Page Views usually have a name to identify them. You can set it by passing a custom label *name*.

```
HashMap<String, String> labels = new HashMap<String,String>();
labels.put("name", "homepage");
DAx.getInstance().notify(EventType.View, labels);
```

Firing Hidden Events

Any particular event may be reported with a Hidden Event. Also usually carries a name.

```
HashMap<String, String> labels = new HashMap<String,String>();
labels.put("name", "hiddenEvent");
DAx.getInstance().notify(EventType.Hidden, labels);
```

SAMPLES

Setup and Start event

```
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  DAx.getInstance()
    .setAppContext(getApplicationContext())
    .setSalt("myPublisherSecret")
    .notify("http://b.scorecardresearch.com/p2?c2=1000001",
      EventType.Start,
      new HashMap<String,String>());
}
```

View event on Click

```
button.setOnClickListener(new View.OnClickListener() {
  public void onClick(View v) {
    DAx.getInstance().notify( EventType.View, new HashMap<String, String>());
  }
});
```

TESTING THE IMPLEMENTATION

As you test your comScore-tagged app internally, comScore servers will collect the measurements. For immediate testing, run your test device's web traffic through Wireshark to view the measurements.

WIRESHARK INSTRUCTIONS

First, download and install Wireshark from http://www.wireshark.org/download.html. Then, perform the following steps each time you restart your Mac and want to Wireshark your test device's app

- Allow Wireshark to sniff network traffic
  - Launch Terminal and run the following commands:
    ```
    cd /dev
    sudo chmod a+r bpf*
    ls -l bpf*
    ```
  - You should see `crw-r--r--` in the first column of the file listing after running the final command above
- Share your Mac's wireless network
  - Connect to the internet using an Ethernet cable
  - Click on the Airport icon at the top right of the screen, select *Create Network*; remember the name you give this network
  - Go to System Preferences, open the *Sharing* pane, highlight (but don't check) *Internet Sharing*
  - Share connection from Ethernet, share computers using AirPort
  - Click *AirPort Options*, and enter the network name you chose above
  - Check the *Internet Sharing* box

    - Click *Start*
    - Connect your test device to the network you just created
- Launch Wireshark
    - Click OK if an error pops up
- Configure Wireshark to look at the `HTTP` traffic from your wireless card
    - In the menu bar, navigate to *Capture* > *Options*
    - Interface `en1` (which is most likely your wireless card)
    - Click *Capture filter*
    - Set the *filter string* to `host b.scorecardresearch.com`
    - Click *Start*

If you see `HTTP` analytic requests matching the above pattern, data is being measured correctly.

Once you have verified the SDK is correctly implemented you must resubmit your App to Android Market and/or any other application marketplace for approval.

FAQS

How can I validate my app measurement is working as intended?
There are two ways to validate your app measurements. You could test the measurements yourself by following the Wireshark instructions above. In addition, you may cold-start your app at least 10 times, and email sdksupport@comscore.com with your *c2* and *Publisher Secret*. We will confirm that we received your measurements within two business days.

Do I have to re-submit my app?
Yes. Once you have tested and confirmed your implementation of the comScore measurement code, resubmit your app.

Will placing comScore measurement code slow the application?
No. The comScore code is extremely lightweight, and will not affect the performance of your application.

Will comScore receive measurements if my application user is not connected to the internet?
No. The comScore measurement code only works if the device has internet connectivity.

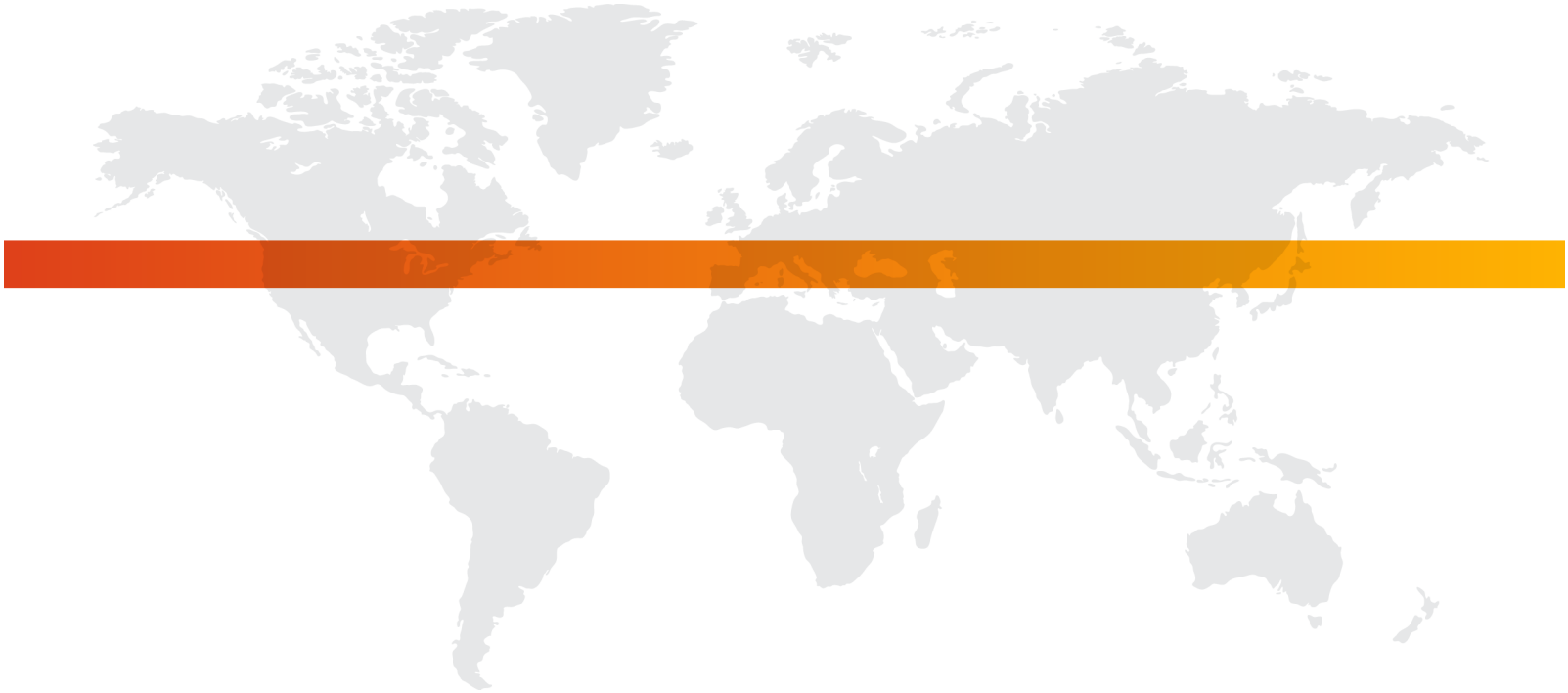
Will comScore receive measurements if my application user is currently roaming?
If the application launches where your user may incur roaming charges, a warning will appear on their screen. However, if the application starts and the user has internet connectivity we will receive the measurements.

Measuring the digital world.™

Unified Digital Measurement

Flash Tag Code Document

July, 2012



Introduction

This document outlines the process for implementing comScore's Unified Digital Measurement (UDM) Flash Tag to support data collection for comScore's audience measurement and (if desired) web analytics products. The document provides you with information about tagging conventions and guidelines for implementing the comScore UDM Flash Tag on your web site.

At time of writing the Flash Tag is provided for use with Flash ActionScript 3 projects. This includes projects using the Flex framework as well as projects that are Adobe Air applications.

Checklist

Before deploying the tag on your site, please complete the following checklist first:

- Obtain your client ID (referred to as the "c2 parameter") through comScore Direct or from your Account Manager.
- Obtain the Flash UDM Tag ActionScript 3 classes.

1. ComScore UDM Flash Tag Implementation

The comScore UDM Flash Tag code consists of a set of classes that need to be imported into your project:

```
1  import com.comscore.udm.Beacon;
```

Using the UDM Flash Tag

To call the tag, simple call the request() method and provide the measurement URL. See the example below for a simple call that contains only the required information:

```
1  Beacon.request("http://b.scorecardresearch.com/b?c1=2&c2=1234567");
```

Note: This serves only as an example. Measurement URLs need more information to get most out of Digitial Analytix reports. This is discussed further in this document.

Passing Values in the Tag

comScore tags are simple client-side HTTP get requests to `b.scorecardresearch.com`.

The get requests include a query string with additional information. Within this query string, there are a variety of parameters. Most common are the "c-parameters" which represent different pieces of information.

One of these c-parameters is the client ID in the c2 parameter. In the call displayed above a sample c2 parameter with a value of **1234567** is used (and highlighted). Please make sure you provide own client ID as the c2 parameter value in both the script and no script portions of the code.

Note: Please note that the request in the example call displayed above specifies non-secure requests.

To generate secure requests, please modify the measurement URL:

```
1  Beacon.request("https://sb.scorecardresearch.com/b?c1=2&c2=1234567");
```

The changes are in the hostname – `sb.scorecardresearch.com` instead of `b.scorecardresearch.com` – and the use of the `https` protocol.

2. Additional Functionality

The comScore UDM Flash Tag is the ActionScript 3 equivalent of the JavaScript comScore Inline Tag. The Flash Tag also supports some of the additional functionality provided by the Inline Tag. The Flash Tag has no JavaScript counterpart, nor does it use additional JavaScript functionality. Any functionality that is supported directly from the measurement URL is also available to the Flash Tag.

Additional Functionality Supported From the Measurement URL

Functionality supported from the measurement URL includes:

- Digital Analytix page names
- Online campaign tracking
- On-site search
- Custom information

This functionality is described in an appendix of the comScore Inline Tag Code Document: “Appendix A – Additional Tag Functionality”. If the appendix did not accompany this document, you can obtain it through comScore Direct or from your Account Manager.

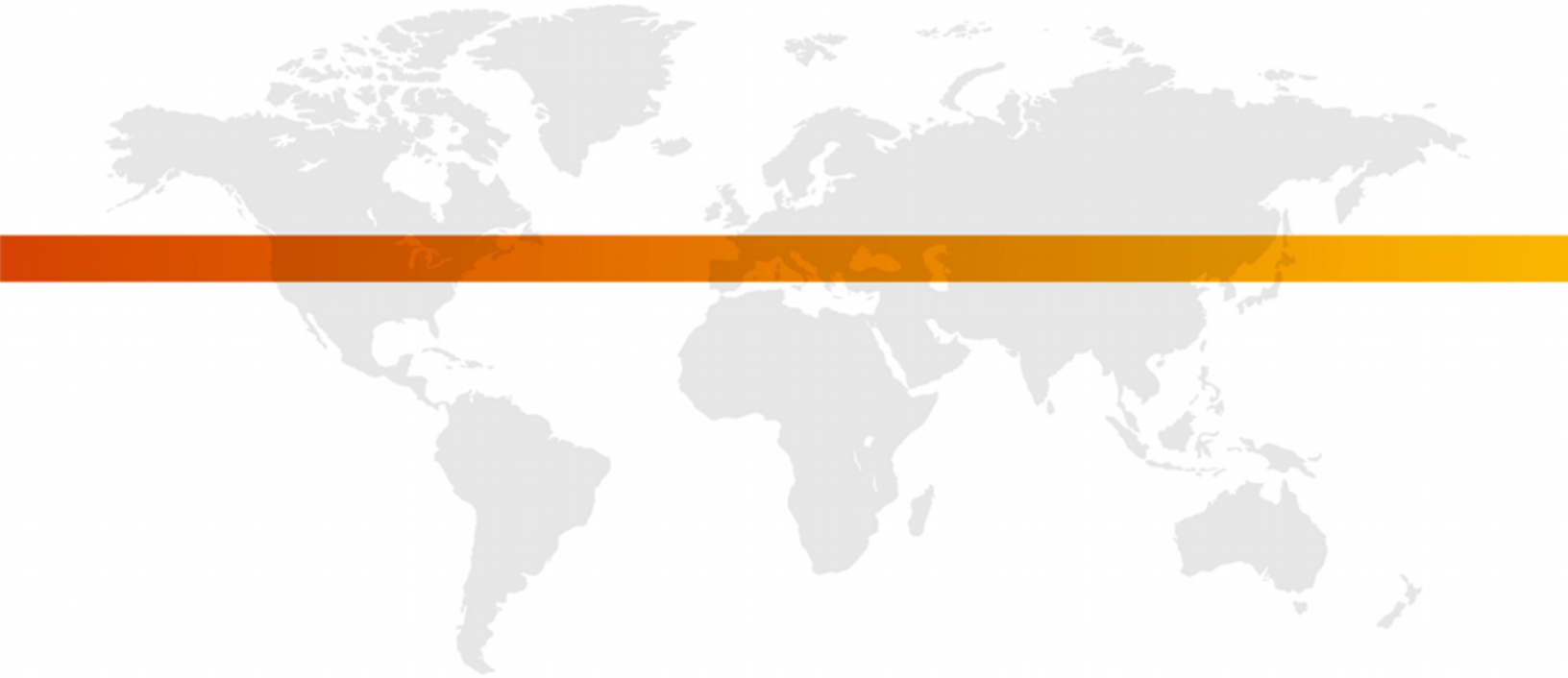
Measuring the digital world.™

DIGITAL ANALYTIX™

iOS Mobile Application Measurement

Tag Specification Document

March 2012



FOR FURTHER INFORMATION, PLEASE CONTACT:

comScore, Inc.
+1 866 276 6972
sdksupport@comscore.com

OVERVIEW

comScore Mobile Application measurement solutions are designed to provide easy usage reporting about the application start and duration.

MOBILE APPLICATION MEASUREMENTS

On the iOS platform, applications may use the *comScore.framework* in order to tag the following events:

- `Start`
- `View`
- `Hidden`
- `Aggregate` (no network transmission)

Start

One very important event is the *Application Start* event, which collects information about the current application version, the last time it was executed, and the last version used if there was an update – among others.

View

You may report a *View* event whenever your application users enter a new context, launch an action or the screen changes sufficiently.

Hidden

Hidden events are meant to report any other action, process or data than *Views* or any other event.

Aggregate

Aggregated events don't generate any network request. Instead, any custom label you send with this event will be taken and stored in memory instead of being transmitted immediately. When more than one aggregated event is fired consecutively, its values are appended or added to those previously stored in patterns we'll later describe.

When the first non-aggregated event occurs, the whole of the aggregated data will be appended and transmitted out.

Aggregation patterns

Label values are always `Strings` but their aggregation pattern may vary depending on their value format, defined as:

Numbers

If a label's value can be casted to number, then this label will be summed consecutively, reporting the total sum as value.

Lists

We'll define Lists as *"comma-separated lists of strings not containing spaces and with at least one comma even for single-item lists"*. Aggregation here will record the number of times each single value appeared, reporting a concatenation with semicolons of each value plus a colon, plus the total number of times it was received.
Example:

Event 1 `"gems=blue,"`
Event 2 `"gems=blue,red,green"`
Event 3 `"gems=blue,green"`
Result: `gems=blue:3;red:1;green:2` → `gems%3Dblue%3A3%3Bred%3A1%3Bgreen%3A2`

Strings

Disregarding the above types, strings will be defined as *"any group of characters not containing commas, or containing commas and spaces"*. The aggregation method will store each unique value received, reporting a comma-separated list of values.
Example:

Event 1 `"gems=blue"`
Event 2 `"gems=red"`
Event 3 `"gems=green"`
Event 4 `"gems=red"`
Result: `gems=blue,red,green` → `gems%3Dblue%2Cred%2Cgreen`

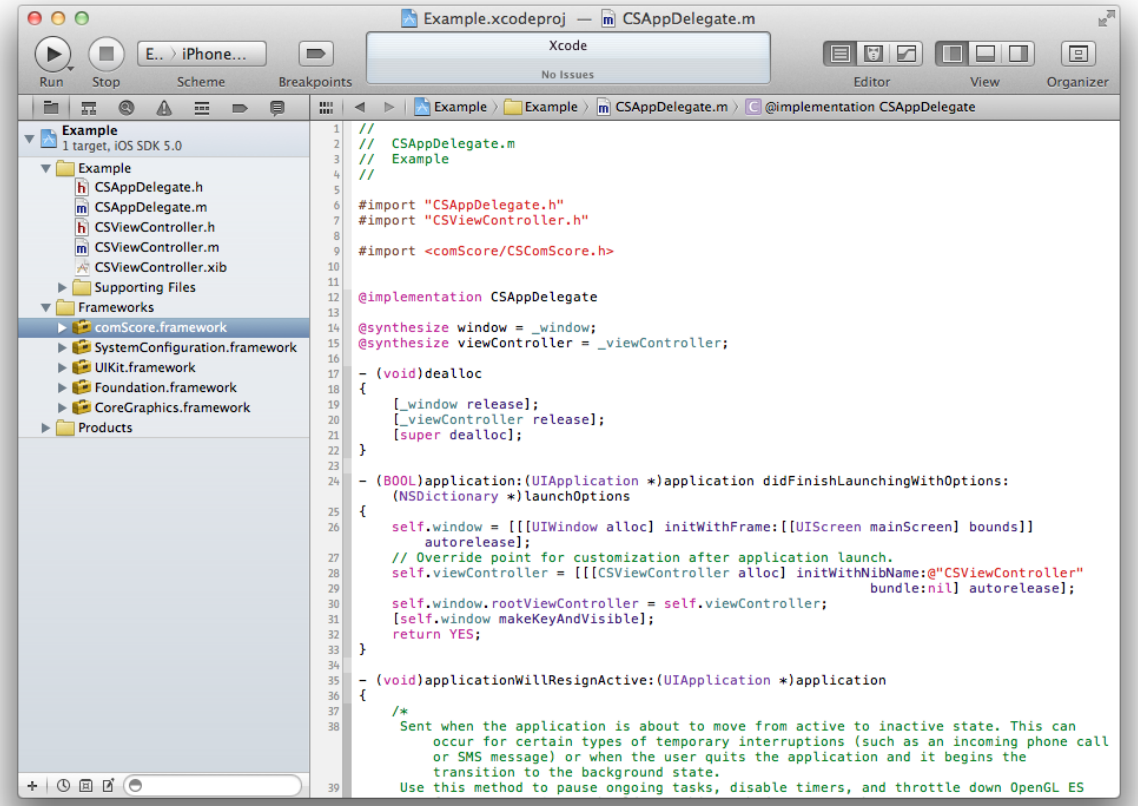
REQUIREMENTS

- Your comScore Client ID, a number of at least 7 digits. This parameter is the C2 value.
  - Available in the comScore Direct tag, found at direct.comscore.com
  - Log in to comScore Direct with your client Login ID and password
  - Click "Get Tag"
  - The number after `c2:` is your Client ID
  - Contact your client service representative or sdksupport@comscore.com with questions
- A *Publisher Secret*, supplied by comScore
  - The *Publisher Secret* is a text string used to obfuscate your application users' UDID when measurements are sent to comScore servers
  - It is the same for all of your applications, but unique to you
  - This is required for security, and to protect the privacy of your application's users
  - This will be sent by your comScore Client Service representative
- SystemConfiguration.framework
  - Make sure this API is loaded in your project
- comScore.framework
  - This is the comScore iOS app tagging SDK. This will be sent by your comScore Client Service representative.

IMPLEMENTING DIGITAL ANALYTIX™

INCLUDE THE FRAMEWORK

Once you received the comScore SDK for iOS, you must include it in your XCode project in order to make use of it. Then drag the entire *comScore.framework* folder into your project, and once there move it into your project's *Frameworks* folder.



Note that **SystemConfiguration.framework** is also required.

IMPORT STATEMENT

Include the library headers with a statement like:

```
#import <comScore/CSComScore.h>
```

Note when upgrading

If you are upgrading from a previous comScore SDK, be aware you must also change the import statement, as it differs slightly when importing *frameworks*: tags instead of quotes, and the directory path.

CODE INITIALIZATION

Set the Publisher Secret

The *Publisher Secret* is a comScore-supplied `String` used to obfuscate your application users' UDID when measurements are sent to comScore servers. It is the same for all of your applications, but unique to you, and is required to protect the privacy of your application's users.

```
[[CSComScore comScore] setSalt:@"PublisherSecret"];
```

Set the Pixel URL

The *Pixel URL* is the combination of comScore's base URL plus your *comScore account ID*.
E.g: "`http://b.scorecardresearch.com/p2?c2=1000001`"

```
[[CSComScore comScore] setPixelURL:@"http://b.scorecardresearch.com/p2\?c2=1000001"];
```

DISPATCHING EVENTS

Fire the Application Start

Fire the Application Start event as soon as you are notified of a successful application launch.

```
NSDictionary *dict = [[NSDictionary alloc] init];
[[CSComScore comScore] notifyWithApplicationEventType:Start andLabels:dict];
[dict release];
```

Firing Page Views

On web analytics, Page Views usually have a name to identify them. You can set it by passing a custom label *name*.

```
NSDictionary *dict = [[NSDictionary dictionaryWithObjectsAndKeys:
                       @"mypagename", @"name", nil] retain];
[[CSComScore comScore] notifyWithEventType:View andLabels:dict];
[dict release];
```

Firing Hidden Events

Any particular event may be reported with a Hidden Event. Also usually carries a name.

```
NSDictionary *dict = [[NSDictionary dictionaryWithObjectsAndKeys:
                       @"myevent", @"name", nil] retain];
[[CSComScore comScore] notifyWithEventType:Hidden andLabels:dict];
[dict release];
```

SAMPLES

Setting the PixelURL and firing the Start event

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Override point for customization after app launch.
    [self.window addSubview:viewController.view];
    [self.window makeKeyAndVisible];

    [[CSComScore comScore] setPixelURL:@"http://b.scorecardresearch.com/p2\?c2=1000001"];

    NSDictionary *dict = [[NSDictionary alloc] init];
    [[CSComScore comScore] notifyWithApplicationEventType:Start andLabels:dict];
    [dict release];

    return YES;
}
```

View event on tab change

```
-(void) tabBar:(UITabBar *)tabBar didSelectItem:(UITabBarItem *)item
{
    //favorites, history and bookmarks are declared UIViewController's
    //favItem, histItem and bookItem are declared UITabBarItem's

    NSMutableDictionary *labels = [[NSMutableDictionary alloc] init];

    if (item == favItem) {
        [self.view bringSubviewToFront:favorites.view]; //show view

        //key_field and value_field are declared UITextField's
        [labels setObject:key_field.text forKey:value_field.text];

        [[CSComScore comScore] notifyWithEventType:View andLabels:labels];

    }
    else if (item == histItem) {
        [self.view bringSubviewToFront:history.view]; //show view

        [[CSComScore comScore] notifyWithEventType:View
andLabels:labels];
    }
    else if (item == bookItem) {
        [self.view bringSubviewToFront:bookmarks.view]; //show view

        [[CSComScore comScore] notifyWithEventType:View
andLabels:labels];
    }
    [labels release];
}
```

Hidden event on touch

```
// Handles the end of a touch event.
-(void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event
{
    NSMutableDictionary *labels = [[NSMutableDictionary alloc] init];
    [labels setObject:@"test_label_key" forKey:@"text_label_value"];

    [[CSComScore comScore] notifyWithEventType:Hidden andLabels:labels];
    [labels release];
}
```

Aggregated event on UIPickerView value change

```
-(void)pickerView:(UIPickerView *)pickerView didSelectRow:(NSInteger)row
    inComponent:(NSInteger)component
{
    // numberPicker is a declared UIPickerView with values 5, 10, 20, 50, 100.
    // numberData is a declared NSArray that contains the UIPickerView data in NSString format.
    // row is the selected index in the UIPickerView for user.
    if (pickerView == numberPicker) {
        NSMutableDictionary *labels = [[NSMutableDictionary alloc] init];

        //set aggregate label
        [labels setObject:[numberData objectAtIndex:row] forKey:@"picker_number"];

        //Notify Aggregate event (no request send)
        [[CSComScore comScore] notifyWithApplicationEventType:Aggregate andLabels:labels];

        [labels release];
    }
}
```

TESTING THE IMPLEMENTATION

As you test your comScore-tagged app internally, Digital Analytix™ servers will collect the measurements.
For immediate testing, run your test device's web traffic through Wireshark on a Mac to view the measurements.

WIRESHARK INSTRUCTIONS

First, download and install Wireshark from http://www.wireshark.org/download.html. Then, perform the following steps each time you restart your Mac and want to Wireshark your test device's app

- Allow Wireshark to sniff network traffic
  - Launch Terminal and run the following commands:
    ```
    cd /dev
    sudo chmod a+r bpf*
    ls -l bpf*
    ```
  - You should see `crw-r--r--` in the first column of the file listing after running the final command above
- Share your Mac's wireless network
  - Connect to the internet using an Ethernet cable

    - Click on the Airport icon at the top right of the screen, select *Create Network*; remember the name you give this network
    - Go to System Preferences, open the *Sharing* pane, highlight (but don't check) *Internet Sharing*
    - Share connection from Ethernet, share computers using AirPort
    - Click *AirPort Options*, and enter the network name you chose above
    - Check the *Internet Sharing* box
    - Click *Start*
    - Connect your iOS test device to the network you just created
- Launch Wireshark
    - Click OK if an error pops up
- Configure Wireshark to look at the `HTTP` traffic from your wireless card
    - In the menu bar, navigate to *Capture* > *Options*
    - Interface `en1` (which is most likely your wireless card)
    - Click *Capture filter*
    - Set the *filter string* to `host b.scorecardresearch.com`
    - Click *Start*

If you see `HTTP` analytic requests matching the above pattern, data is being measured correctly.

Once you have verified the SDK is correctly implemented you must resubmit your App to the iTunes App store for approval.

FAQS

How can I validate my app measurement is working as intended?
There are two ways to validate your app measurements. You could test the measurements yourself by following the Wireshark instructions above. In addition, you may cold-start your app at least 10 times, and email sdksupport@comscore.com with your *c2* and *Publisher Secret*. We will confirm that we received your measurements within two business days.

Do I have to re-submit my app to the App Store?
Yes. Once you have tested and confirmed your implementation of the comScore measurement code, resubmit your app to the App Store. Your app should be accepted quickly, if the comScore code is the only change you made since the last application update.

How can I change my application's name that is reported to comScore?
Before you notify any event to the comScore SDK, you can change your reported application name, which defaults to your app's *Info.plist* application bundle name (`CFBundleName`), by calling:

```
[[CSComScore comScore] setAppName: @"myApplication"];
```

Will placing comScore measurement code slow the application?
No. The comScore code is extremely lightweight, and will not affect the performance of your application.

Will comScore receive measurements if my application user is not connected to the internet?
No. The comScore measurement code only works if the device has internet connection.

Will comScore receive measurements if my application user is currently roaming?
If the application launches where your user may incur roaming charges, a warning will appear on their

screen. However, if the application starts and the user has internet connectivity we will receive the measurements.

Measuring the digital world.™

DIGITAL ANALYTIX™

BlackBerry Smartphone Application Measurement

Tag Specification Document

April 2012

FOR FURTHER INFORMATION, PLEASE CONTACT:

comScore, Inc.
+1 866 276 6972
sdksupport@comscore.com

OVERVIEW

comScore Mobile Application measurement solutions are designed to provide easy usage reporting about the application start and duration.

MOBILE APPLICATION MEASUREMENTS

On the BlackBerry Smartphone platform, applications may use the *libComScore* in order to tag the following events:

- `Start`
- `View`
- `Hidden`
- `Aggregate` (no network transmission)

Start

One very important event is the *Application Start* event, which collects information about the current application version, the last time it was executed, and the last version used if there was an update – among others.

View

You may report a *View* event whenever your application users enter a new context, launch an action or the screen changes sufficiently.

Hidden

Hidden events are meant to report any other action, process or data than *Views* or any other event.

Aggregate

Aggregated events don't generate any network request. Instead, any custom label you send with this event will be taken and stored in memory instead of being transmitted immediately. When more than one aggregated event is fired consecutively, its values are appended or added to those previously stored in patterns we'll later describe.

When the first non-aggregated event occurs, the whole of the aggregated data will be appended and transmitted out.

Aggregation patterns

Label values are always `Strings` but their aggregation pattern may vary depending on their value format, defined as:

Numbers

If a label's value can be casted to number then this label will be summed consecutively, to report the total sum as value.

Lists

We'll define Lists as *"comma-separated lists of strings not containing spaces and with at least one comma even for single-item lists"*. Aggregation here will record the number of times each single value appeared, reporting a concatenation with semicolons of each value plus a colon, plus the total number of times it was received.
Example:

Event 1 `"gems=blue,"`
Event 2 `"gems=blue,red,green"`
Event 3 `"gems=blue,green"`
Result: `gems=blue:3;red:1;green:2` → `gems%3Dblue%3A3%3Bred%3A1%3Bgreen%3A2`

Strings

Disregarding the above types, strings will be defined as *"any group of characters not containing commas, or containing commas and spaces"*. The aggregation method will store each unique value received, reporting a comma-separated list of values.
Example:

Event 1 `"gems=blue"`
Event 2 `"gems=red"`
Event 3 `"gems=green"`
Event 4 `"gems=red"`
Result: `gems=blue,red,green` → `gems%3Dblue%2Cred%2Cgreen`

REQUIREMENTS

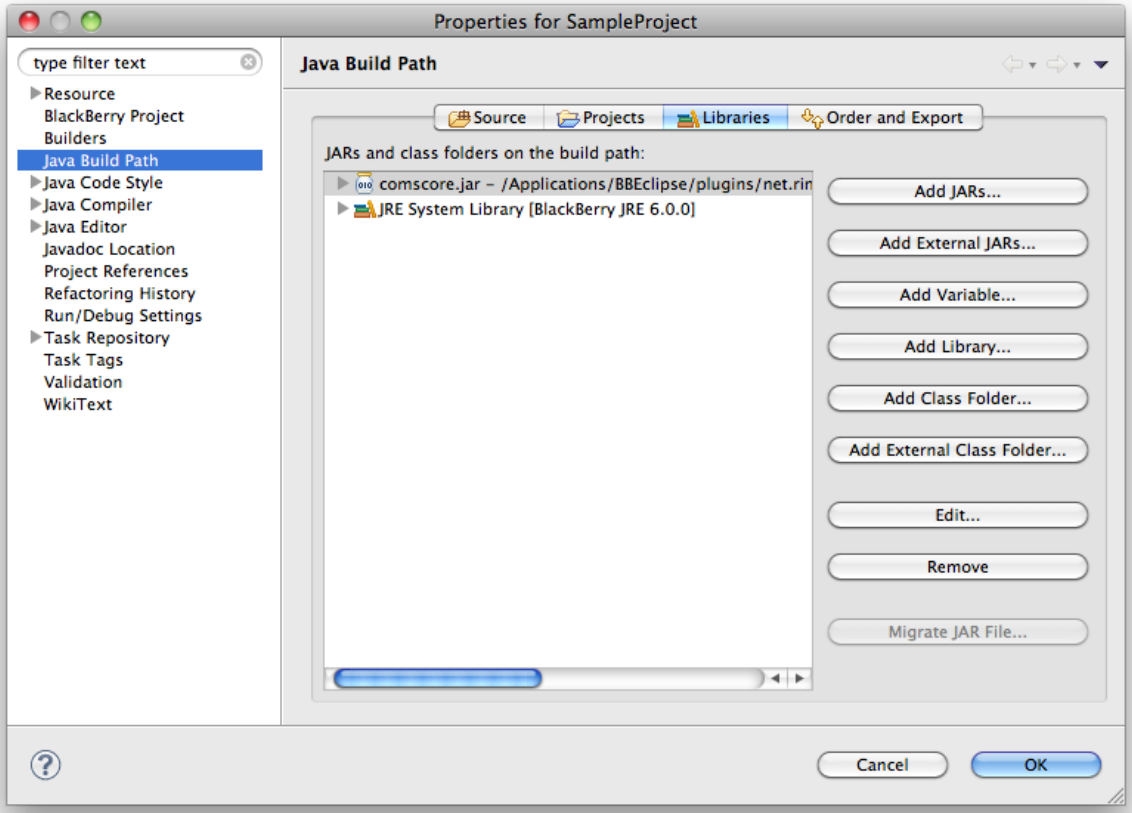
- Your comScore Client ID, a number of at least 7 digits. This parameter is the C2 value.
  - Available in the comScore Direct tag, found at direct.comscore.com
  - Log in to comScore Direct with your client Login ID and password
  - Click "Get Tag"
  - The number after `c2:` is your Client ID
  - Contact your client service representative or sdksupport@comscore.com with questions
- A *Publisher Secret*, supplied by comScore
  - The *Publisher Secret* is a text string used to obfuscate your application users' UDID when measurements are sent to comScore servers
  - It is the same for all of your mobile applications, also the *iOS* ones, but unique to you
  - This is required for security, and to protect the privacy of your application's users
  - This will be sent by your comScore Client Service representative
- *libComScore SDK*'s `comscore.jar`
  - This is the comScore BlackBerry smartphone app tagging SDK. Please request it to your comScore Client Service representative.
- Application permissions for:
  - *Allow Connections*
  - *Allow Interactions*
  - Your app will automatically request these permissions without requiring any action.

IMPLEMENTING DIGITAL ANALYTIX™

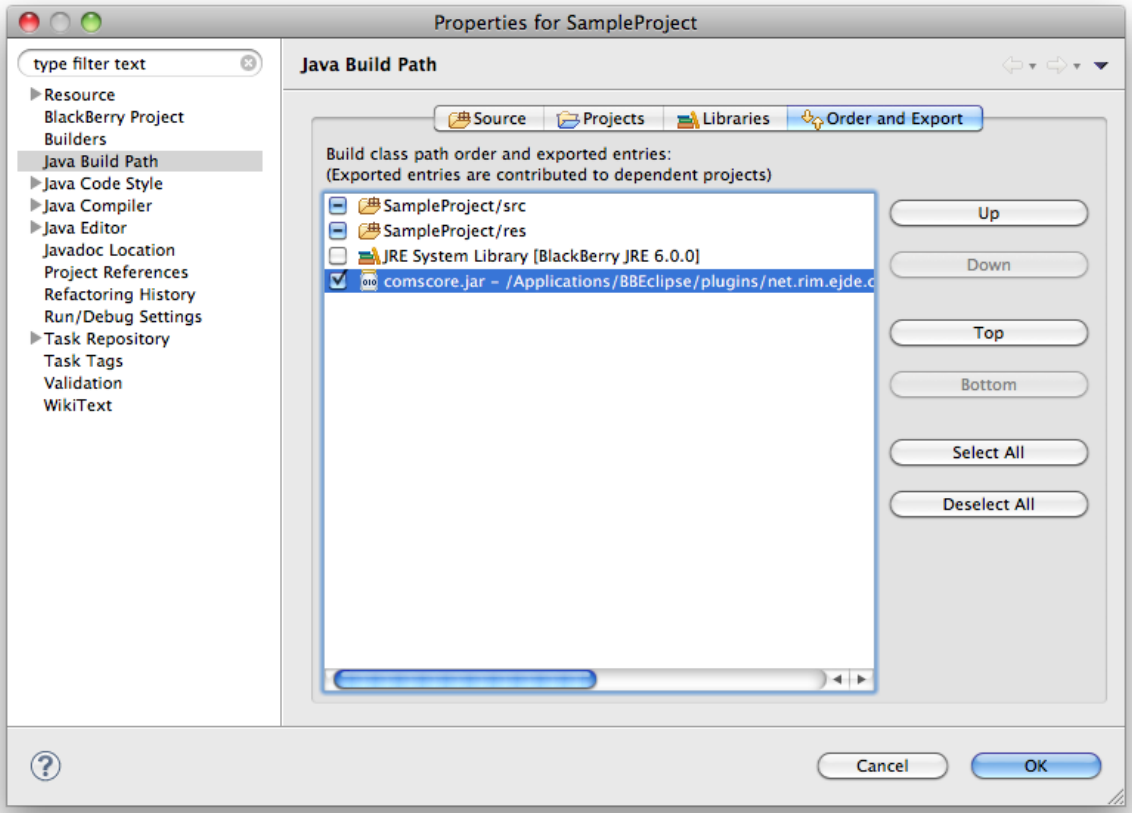
INCLUDE THE LIBRARY

To begin, `comscore.jar` must be included in your Java Build Path.

- In Eclipse:
  - Right-click your project and select *Properties*.
  - Go to the *Java Build Path* panel.
  - Click *Add External JARs…* and navigate to the `comscore.jar` file you were provided.



- Go to Order and Export section and select `comscore.jar` file you have just imported.



CODE INITIALIZATION

Import statement

For your convenience, import the classes you are going to use afterwards:

```
import com.comscore.analytics.DAx;
import com.comscore.applications.EventType;
```

Set the Application Context

The *Application Context* needs to be set **just once** before any analytic request is made. It is available by calling `ApplicationDescriptor.currentApplicationDescriptor()`.

```
DAx.getInstance().setAppContext(ApplicationDescriptor.currentApplicationDescriptor());
```

Set the Publisher Secret

The *Publisher Secret* is a comScore-supplied `String` used to obfuscate your application users' UDID when measurements are sent to comScore servers. It is the same for all of your applications, but unique to you, and is required to protect the privacy of your application's users.

```
DAx.getInstance().setSalt("PublisherSecret");
```

Set the Pixel URL

The *Pixel URL* is the combination of comScore's base URL plus your *comScore account ID*.
E.g: "`http://b.scorecardresearch.com/p2?c2=1000001`"

```
DAx.getInstance().setPixelURL("http://b.scorecardresearch.com/p2?c2=1000001");
```

Custom Data: Labels

We call *labels* to each `name=value` pair of custom data you send over on each event. You can have any number of those, although there is a hard limit of 2048 bytes for the overall request URL and any exceeding labels will be cut out.

To set application-wise persistent labels, sent for each of your application's events, use the methods available for that purpose. Examples:

```
// Set a single label
DAx.getInstance().setCustomLabel( "labelName", "myValue" );

// Set any number of labels
Hashtable labels = new Hashtable();
labels.put("labelName1", "myValue1");
labels.put("labelName2", "myValue2");
labels.put("labelName3", "myValue3");

DAx.getInstance().setCustomLabels( labels );
```

DISPATCHING EVENTS

Fire the Application Start

Fire the Application Start event as soon as you are notified of a successful application launch.

```
DAx.getInstance().notify(new com.comscore.applications.EventType(
  com.comscore.applications.EventType.START), new Hashtable());
```

Firing Page Views

On web analytics, Page Views usually have a name to identify them. You can set it by passing a custom label *name*.

```
Hashtable labels = new Hashtable();
labels.put("name", "homepage");
DAx.getInstance().notify(new com.comscore.metrics.EventType(
  com.comscore.metrics.EventType.VIEW), labels);
```

Firing Hidden Events

Any particular event may be reported with a Hidden Event. Also usually carries a name.

```
Hashtable labels = new Hashtable();
labels.put("name", "hiddenEvent");
DAx.getInstance().notify(new com.comscore.metrics.EventType(
  com.comscore.metrics.EventType.HIDDEN), labels);
```

SAMPLES

Setup and Start event

```
public MyScreen() {

  DAx.getInstance()
    .setAppContext(ApplicationDescriptor.currentApplicationDescriptor())
    .setSalt("myPublisherSecret")
    .notify("http://b.scorecardresearch.com/p2?c2=1000001",
      new com.comscore.applications.EventType(com.comscore.applications.EventType.START),
      new Hashtable());
}
```

Button event FieldChange

```
button.setChangeListener(new FieldChangeListener() {
  public void fieldChanged(Field field, int context) {
    DAx.getInstance().notify(new com.comscore.applications.EventType(
      com.comscore.applications.EventType.VIEW), new Hashtable());
  }
});
```

TESTING THE IMPLEMENTATION

As you test your comScore-tagged app internally, comScore servers will collect the measurements. For immediate testing, run your test device's web traffic through Wireshark to view the measurements.

WIRESHARK INSTRUCTIONS

First, download and install Wireshark from http://www.wireshark.org/download.html. Then, perform the following steps each time you restart your Mac and want to Wireshark your test device's app

- Allow Wireshark to sniff network traffic
  - Launch Terminal and run the following commands:
    ```
    cd /dev
    sudo chmod a+r bpf*
    ls -l bpf*
    ```
  - You should see `crw-r--r--` in the first column of the file listing after running the final command above
- Share your Mac's wireless network
  - Connect to the internet using an Ethernet cable
  - Click on the Airport icon at the top right of the screen, select *Create Network*; remember the name you give this network
  - Go to System Preferences, open the *Sharing* pane, highlight (but don't check) *Internet Sharing*
  - Share connection from Ethernet, share computers using AirPort
  - Click *AirPort Options*, and enter the network name you chose above
  - Check the *Internet Sharing* box
  - Click *Start*
  - Connect your test device to the network you just created
- Launch Wireshark
  - Click OK if an error pops up
- Configure Wireshark to look at the `HTTP` traffic from your wireless card
  - In the menu bar, navigate to *Capture* > *Options*
  - Interface `en1` (which is most likely your wireless card)
  - Click *Capture filter*
  - Set the *filter string* to `host b.scorecardresearch.com`
  - Click *Start*

If you see `HTTP` analytic requests matching the above pattern, data is being measured correctly.

Once you have verified the SDK is correctly implemented you must resubmit your App to BlackBerry App World and/or any other application marketplace for approval.

FAQS

How can I validate my app measurement is working as intended?
There are two ways to validate your app measurements. You could test the measurements yourself by

following the Wireshark instructions above. In addition, you may cold-start your app at least 10 times, and email sdksupport@comscore.com with your *c2* and *Publisher Secret*. We will confirm that we received your measurements within two business days.

Do I have to re-submit my app?
Yes. Once you have tested and confirmed your implementation of the comScore measurement code, resubmit your app.

Will placing comScore measurement code slow the application?
No. The comScore code is extremely lightweight, and will not affect the performance of your application.

Will comScore receive measurements if my application user is not connected to the internet?
No. The comScore measurement code only works if the device has internet connectivity.

Will comScore receive measurements if my application user is currently roaming?
If the application launches where your user may incur roaming charges, a warning will appear on their screen. However, if the application starts and the user has internet connectivity we will receive the measurements.

Measuring the digital world.™

DIGITAL ANALYTIX™

BlackBerry Tablet Application Measurement

Tag Specification Document

April 2012

FOR FURTHER INFORMATION, PLEASE CONTACT:

comScore, Inc.
+1 866 276 6972
sdksupport@comscore.com

OVERVIEW

comScore Mobile Application measurement solutions are designed to provide easy usage reporting about the application start and duration.

MOBILE APPLICATION MEASUREMENTS

On the BlackBerry Tablet platform, applications may use the *libComScore* in order to tag the following events:

- `Start`
- `View`
- `Hidden`
- `Aggregate` (no network transmission)

Start

One very important event is the *Application Start* event, which collects information about the current application version, the last time it was executed, and the last version used if there was an update – among others.

View

You may report a *View* event whenever your application users enter a new context, launch an action or the screen changes sufficiently.

Hidden

Hidden events are meant to report any other action, process or data than *Views* or any other event.

Aggregate

Aggregated events don't generate any network request. Instead, any custom label you send with this event will be taken and stored in memory instead of being transmitted immediately. When more than one aggregated event is fired consecutively, its values are appended or added to those previously stored in patterns we'll later describe.

When the first non-aggregated event occurs, the whole of the aggregated data will be appended and transmitted out.

Aggregation patterns

Label values are always `Strings` but their aggregation pattern may vary depending on their value format, defined as:

Numbers

If a label's value can be casted to number then this label will be summed consecutively, to report the total sum as value.

Lists

We'll define Lists as *"comma-separated lists of strings not containing spaces and with at least one comma even for single-item lists"*. Aggregation here will record the number of times each single value appeared, reporting a concatenation with semicolons of each value plus a colon, plus the total number of times it was received.
Example:

Event 1 `"gems=blue,"`
Event 2 `"gems=blue,red,green"`
Event 3 `"gems=blue,green"`
Result: `gems=blue:3;red:1;green:2` → `gems%3Dblue%3A3%3Bred%3A1%3Bgreen%3A2`

Strings

Disregarding the above types, strings will be defined as *"any group of characters not containing commas, or containing commas and spaces"*. The aggregation method will store each unique value received, reporting a comma-separated list of values.
Example:

Event 1 `"gems=blue"`
Event 2 `"gems=red"`
Event 3 `"gems=green"`
Event 4 `"gems=red"`
Result: `gems=blue,red,green` → `gems%3Dblue%2Cred%2Cgreen`

REQUIREMENTS

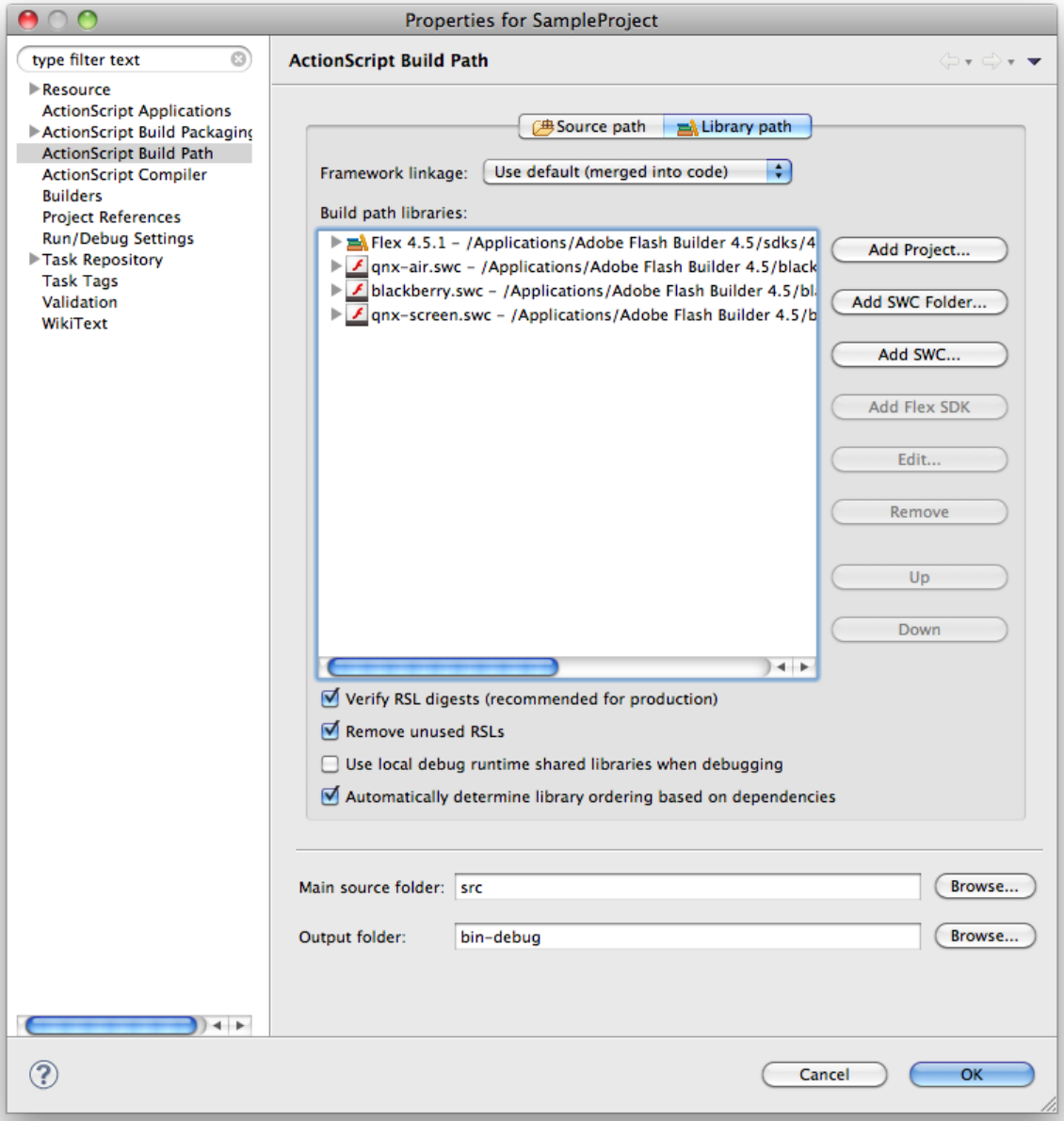
- Your comScore Client ID, a number of at least 7 digits. This parameter is the C2 value.
  - Available in the comScore Direct tag, found at direct.comscore.com
  - Log in to comScore Direct with your client Login ID and password
  - Click "Get Tag"
  - The number after `c2:` is your Client ID
  - Contact your client service representative or sdksupport@comscore.com with questions
- A *Publisher Secret*, supplied by comScore
  - The *Publisher Secret* is a text string used to obfuscate your application users' UDID when measurements are sent to comScore servers
  - It is the same for all of your mobile applications, also the *iOS* ones, but unique to you
  - This is required for security, and to protect the privacy of your application's users
  - This will be sent by your comScore Client Service representative
- *libComScore SDK*'s `comscore.swc`
  - This is the comScore BlackBerry Tablet app tagging SDK. Please request it to your comScore Client Service representative.
- Application permission: `read_device_identifying_information`
  - This is required in order to uniquely identify your visitors.
  - When not set, the SDK will auto-generate an ID for your visitors, but it will be lost if your app gets uninstalled.

IMPLEMENTING DIGITAL ANALYTIX™

INCLUDE THE LIBRARY

To begin, `comscore.swc` must be included in your ActionScript Build Path.

- In Flash Builder:
  - Right-click your project and select *Properties*.
  - Go to the *ActionScript Build Path* panel.
  - Click *Add SWC…* and navigate to the `comscore.swc` file you were provided.



CODE INITIALIZATION

Import statement

For your convenience, import the classes you are going to use afterwards:

```
import com.comscore.analytics.DAx;
import com.comscore.applications.EventType;
```

Set the Application Context

The *Application Context* needs to be set **just once** before any analytic request is made. It is available by calling `NativeApplication.nativeApplication.applicationDescriptor`.

```
DAx.getInstance().setAppContext(NativeApplication.nativeApplication.applicationDescriptor);
```

Set the Publisher Secret

The *Publisher Secret* is a comScore-supplied `String` used to obfuscate your application users' UDID when measurements are sent to comScore servers. It is the same for all of your applications, but unique to you, and is required to protect the privacy of your application's users.

```
DAx.getInstance().setSalt("PublisherSecret");
```

Set the Pixel URL

The *Pixel URL* is the combination of comScore's base URL plus your *comScore account ID*.
E.g: "`http://b.scorecardresearch.com/p2?c2=1000001`"

```
DAx.getInstance().setPixelURL("http://b.scorecardresearch.com/p2?c2=1000001");
```

Custom Data: Labels

We call *labels* to each `name=value` pair of custom data you send over on each event. You can have any number of those, although there is a hard limit of 2048 bytes for the overall request URL and any exceeding labels will be cut out.

To set application-wise persistent labels, sent for each of your application's events, use the methods available for that purpose. Examples:

```
// Set a single label
DAx.getInstance().setCustomLabel( "labelName", "myValue" );

// Set any number of labels
var labels:Array = new Array();
labels["labelName1"] = "myValue1";
labels["labelName2"] = "myValue2";
labels["labelName3"] = "myValue3";

DAx.getInstance().setCustomLabels( labels );
```

DISPATCHING EVENTS

Fire the Application Start

Fire the Application Start event as soon as you are notified of a successful application launch.

```
DAx.getInstance().notifyEventWithLabels(new com.comscore.applications.EventType(
  com.comscore.applications.EventType.START), new Array());
```

Firing Page Views

On web analytics, Page Views usually have a name to identify them. You can set it by passing a custom label *name*.

```
var labels:Array = new Array();
labels["name"] = "homepage";
DAx.getInstance().notifyEventWithLabels(new com.comscore.applications.EventType(
  com.comscore.applications.EventType.VIEW), labels);
```

Firing Hidden Events

Any particular event may be reported with a Hidden Event. Also usually carries a name.

```
var labels:Array = new Array();
labels["name"] = "hiddenEvent";
DAx.getInstance().notifyEventWithLabels(new com.comscore.applications.EventType(
  com.comscore.applications.EventType.HIDDEN), labels);
```

SAMPLES

Setup and Start event

```
public function MainClass() {

  DAx.getInstance()
    .setAppContext(NativeApplication.nativeApplication.applicationDescriptor)
    .setSalt("myPublisherSecret")
    .notifyEvent("http://b.scorecardresearch.com/p2?c2=1000001",
      new com.comscore.applications.EventType(com.comscore.applications.EventType.START),
      new Array());
}
```

Button event onClick

```
private function onClickButton(e:MouseEvent):void
{
  DAx.getInstance().notifyEventWithLabels(new com.comscore.applications.EventType(
    com.comscore.applications.EventType.VIEW), new Array());
}
```

TESTING THE IMPLEMENTATION

As you test your comScore-tagged app internally, comScore servers will collect the measurements. For immediate testing, run your test device's web traffic through Wireshark to view the measurements.

WIRESHARK INSTRUCTIONS

First, download and install Wireshark from http://www.wireshark.org/download.html. Then, perform the following steps each time you restart your Mac and want to Wireshark your test device's app

- Allow Wireshark to sniff network traffic
  - Launch Terminal and run the following commands:
    ```
    cd /dev
    sudo chmod a+r bpf*
    ls -l bpf*
    ```
  - You should see `crw-r--r--` in the first column of the file listing after running the final command above
- Share your Mac's wireless network
  - Connect to the internet using an Ethernet cable
  - Click on the Airport icon at the top right of the screen, select *Create Network*; remember the name you give this network
  - Go to System Preferences, open the *Sharing* pane, highlight (but don't check) *Internet Sharing*
  - Share connection from Ethernet, share computers using AirPort
  - Click *AirPort Options*, and enter the network name you chose above
  - Check the *Internet Sharing* box
  - Click *Start*
  - Connect your test device to the network you just created

- Launch Wireshark
  - Click OK if an error pops up
- Configure Wireshark to look at the `HTTP` traffic from your wireless card
  - In the menu bar, navigate to *Capture* > *Options*
  - Interface `en1` (which is most likely your wireless card)
  - Click *Capture filter*
  - Set the *filter string* to `host b.scorecardresearch.com`
  - Click *Start*

If you see `HTTP` analytic requests matching the above pattern, data is being measured correctly.

Once you have verified the SDK is correctly implemented you must resubmit your App to BlackBerry App World and/or any other application marketplace for approval.

FAQS

How can I validate my app measurement is working as intended?
There are two ways to validate your app measurements. You could test the measurements yourself by following the Wireshark instructions above. In addition, you may cold-start your app at least 10 times, and email sdksupport@comscore.com with your *c2* and *Publisher Secret*. We will confirm that we received your measurements within two business days.

Do I have to re-submit my app?
Yes. Once you have tested and confirmed your implementation of the comScore measurement code, resubmit your app.

Will placing comScore measurement code slow the application?
No. The comScore code is extremely lightweight, and will not affect the performance of your application.

Will comScore receive measurements if my application user is not connected to the internet?
No. The comScore measurement code only works if the device has internet connectivity.

Will comScore receive measurements if my application user is currently roaming?
If the application launches where your user may incur roaming charges, a warning will appear on their screen. However, if the application starts and the user has internet connectivity we will receive the measurements.

Measuring the digital world.™

DIGITAL ANALYTIX™

Symbian Java Application Measurement

Tag Specification Document

February 2012

FOR FURTHER INFORMATION, PLEASE CONTACT:

comScore, Inc.
+1 866 276 6972
sdksupport@comscore.com

OVERVIEW

comScore Mobile Application measurement solutions are designed to provide easy usage reporting about the application start and duration.

MOBILE APPLICATION MEASUREMENTS

On the Symbian platform, applications may use the *libComScore* in order to tag the following events:

- `Start`
- `View`
- `Hidden`
- `Aggregate` (no network transmission)

Start

One very important event is the *Application Start* event, which collects information about the current application version, the last time it was executed, and the last version used if there was an update – among others.

View

You may report a *View* event whenever your application users enter a new context, launch an action or the screen changes sufficiently.

Hidden

Hidden events are meant to report any other action, process or data than *Views* or any other event.

Aggregate

Aggregated events don't generate any network request. Instead, any custom label you send with this event will be taken and stored in memory instead of being transmitted immediately. When more than one aggregated event is fired consecutively, its values are appended or added to those previously stored in patterns we'll later describe.

When the first non-aggregated event occurs, the whole of the aggregated data will be appended and transmitted out.

Aggregation patterns

Label values are always `Strings` but their aggregation pattern may vary depending on their value format, defined as:

Numbers

If a label's value can be casted to number then this label will be summed consecutively, to report the total sum as value.

Lists

We'll define Lists as *"comma-separated lists of strings not containing spaces and with at least one comma even for single-item lists"*. Aggregation here will record the number of times each single value appeared, reporting a concatenation with semicolons of each value plus a colon, plus the total number of times it was received.
Example:

Event 1 `"gems=blue,"`
Event 2 `"gems=blue,red,green"`
Event 3 `"gems=blue,green"`
Result: `gems=blue:3;red:1;green:2` → `gems%3Dblue%3A3%3Bred%3A1%3Bgreen%3A2`

Strings

Disregarding the above types, strings will be defined as *"any group of characters not containing commas, or containing commas and spaces"*. The aggregation method will store each unique value received, reporting a comma-separated list of values.
Example:

Event 1 `"gems=blue"`
Event 2 `"gems=red"`
Event 3 `"gems=green"`
Event 4 `"gems=red"`
Result: `gems=blue,red,green` → `gems%3Dblue%2Cred%2Cgreen`

REQUIREMENTS

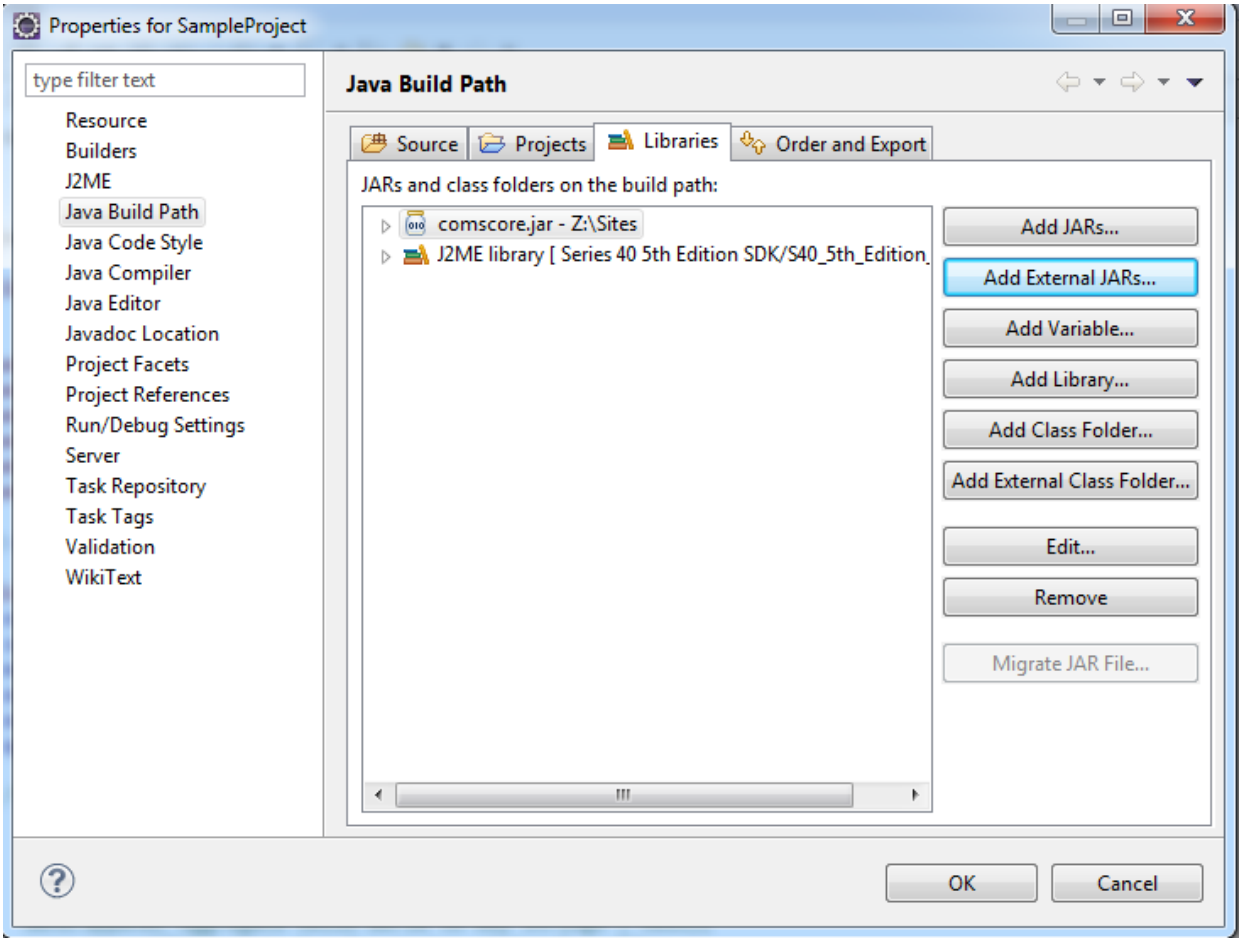
- Your comScore Client ID, a number of at least 7 digits. This parameter is the C2 value.
  - Available in the comScore Direct tag, found at direct.comscore.com
  - Log in to comScore Direct with your client Login ID and password
  - Click "Get Tag"
  - The number after `c2:` is your Client ID
  - Contact your client service representative or sdksupport@comscore.com with questions
- A *Publisher Secret*, supplied by comScore
  - The *Publisher Secret* is a text string used to obfuscate your application users' UDID when measurements are sent to comScore servers
  - It is the same for all of your mobile applications, also the *iOS* ones, but unique to you
  - This is required for security, and to protect the privacy of your application's users
  - This will be sent by your comScore Client Service representative
- *libComScore SDK*'s `comscore.jar`
  - This is the comScore Symbian Java app tagging SDK. Please request it to your comScore Client Service representative.
- Allow your application to use network and send or receive data
  - Your app will automatically request this setting when it starts.

IMPLEMENTING DIGITAL ANALYTIX™

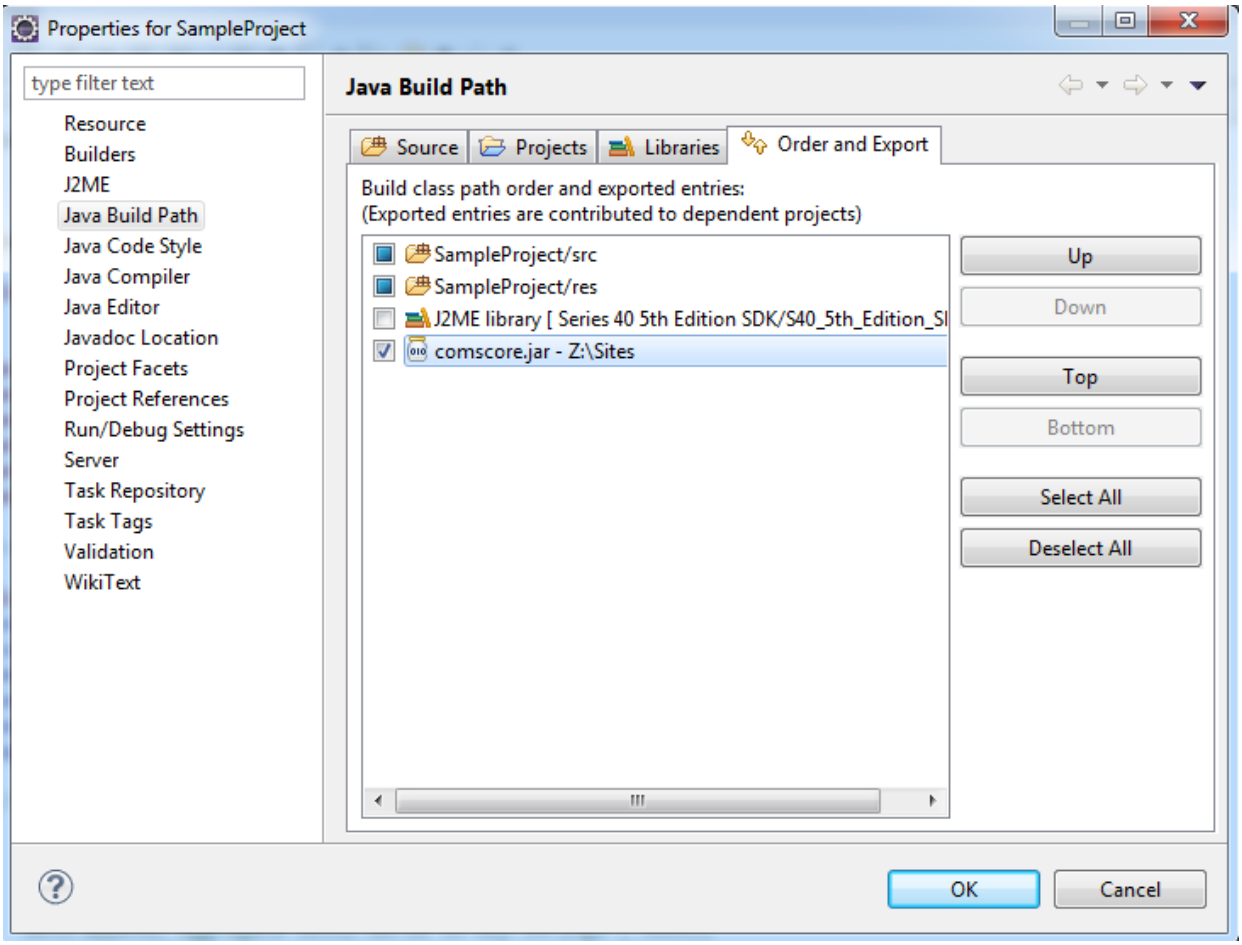
INCLUDE THE LIBRARY

To begin, `comscore.jar` must be included in your Java Build Path.

- In Eclipse:
  - Right-click your project and select *Properties*.
  - Go to the *Java Build Path* panel.
  - Click *Add External JARs…* and navigate to the `comscore.jar` file you were provided.



- Go to Order and Export section and select `comscore.jar` file you have just imported.



CODE INITIALIZATION

Import statement

For your convenience, import the classes you are going to use afterwards:

```
import com.comscore.analytics.DAx;
import com.comscore.applications.EventType;
```

Set the Application Context

The *Application Context* needs to be set **just once** before any analytic request is made. It is available by calling `this`, which is reference to the current MIDlet object.

```
DAx.getInstance().setAppContext(this);
```

Set the Publisher Secret

The *Publisher Secret* is a comScore-supplied `String` used to obfuscate your application users' UDID when measurements are sent to comScore servers. It is the same for all of your applications, but unique to you, and is required to protect the privacy of your application's users.

```
DAx.getInstance().setSalt("PublisherSecret");
```

Set the Pixel URL

The *Pixel URL* is the combination of comScore's base URL plus your *comScore account ID*.
E.g: "`http://b.scorecardresearch.com/p2?c2=1000001`"

```
DAx.getInstance().setPixelURL("http://b.scorecardresearch.com/p2?c2=1000001");
```

DISPATCHING EVENTS

Fire the Application Start

Fire the Application Start event as soon as you are notified of a successful application launch.

```
DAx.getInstance().notify(new com.comscore.applications.EventType(
    com.comscore.applications.EventType.START), new Hashtable());
```

Firing Page Views

On web analytics, Page Views usually have a name to identify them. You can set it by passing a custom label *name*.

```
Hashtable labels = new Hashtable();
labels.put("name", "homePage");
DAx.getInstance()
    .notify(new com.comscore.metrics.EventType(
        com.comscore.metrics.EventType.VIEW), labels);
```

Firing Hidden Events

Any particular event may be reported with a Hidden Event. Also usually carries a name.

```
Hashtable labels = new Hashtable();
labels.put("name", "hiddenEvent");
DAx.getInstance()
    .notify(new com.comscore.metrics.EventType(
        com.comscore.metrics.EventType.HIDDEN), labels);
```

SAMPLES

Setup and Start event

```
public class SampleProject extends MIDlet{
    protected void startApp() throws MIDletStateChangeException {
        DAx.getInstance()
            .setAppContext(this)
            .setSalt("myPublisherSecret")
            .notify("http://b.scorecardresearch.com/p2?c2=1000001",
                new com.comscore.applications.EventType(
                    com.comscore.applications.EventType.START
                ),
                new Hashtable());
    }
}
```

Listener for any command

```
public void commandAction(Command arg0, Displayable arg1) {
    DAx.getInstance()
        .notify(new com.comscore.applications.EventType(
            com.comscore.applications.EventType.VIEW), new Hashtable()
        );
}
```

TESTING THE IMPLEMENTATION

As you test your comScore-tagged app internally, comScore servers will collect the measurements. For immediate testing, run your test device's web traffic through Wireshark to view the measurements.

WIRESHARK INSTRUCTIONS

First, download and install Wireshark from http://www.wireshark.org/download.html. Then, perform the following steps each time you restart your Mac and want to Wireshark your test device's app

- Allow Wireshark to sniff network traffic
  - Launch Terminal and run the following commands:
    ```
    cd /dev
    sudo chmod a+r bpf*
    ls -l bpf*
    ```
  - You should see `crw-r--r--` in the first column of the file listing after running the final command above
- Share your Mac's wireless network
  - Connect to the internet using an Ethernet cable
  - Click on the Airport icon at the top right of the screen, select *Create Network*; remember the name you give this network
  - Go to System Preferences, open the *Sharing* pane, highlight (but don't check) *Internet Sharing*
  - Share connection from Ethernet, share computers using AirPort
  - Click *AirPort Options*, and enter the network name you chose above
  - Check the *Internet Sharing* box
  - Click *Start*
  - Connect your test device to the network you just created
- Launch Wireshark
  - Click OK if an error pops up
- Configure Wireshark to look at the `HTTP` traffic from your wireless card
  - In the menu bar, navigate to *Capture* > *Options*
  - Interface `en1` (which is most likely your wireless card)
  - Click *Capture filter*
  - Set the *filter string* to `host b.scorecardresearch.com`
  - Click *Start*

If you see `HTTP` analytic requests matching the above pattern, data is being measured correctly.

Once you have verified the SDK is correctly implemented you must resubmit your App to any application marketplace for approval.

FAQS

How can I validate my app measurement is working as intended?
There are two ways to validate your app measurements. You could test the measurements yourself by following the Wireshark instructions above. In addition, you may cold-start your app at least 10 times, and email sdksupport@comscore.com with your *c2* and *Publisher Secret*. We will confirm that we received your measurements within two business days.

Do I have to re-submit my app?
Yes. Once you have tested and confirmed your implementation of the comScore measurement code, resubmit your app.

Will placing comScore measurement code slow the application?
No. The comScore code is extremely lightweight, and will not affect the performance of your application.

Will comScore receive measurements if my application user is not connected to the internet?
No. The comScore measurement code only works if the device has internet connectivity.

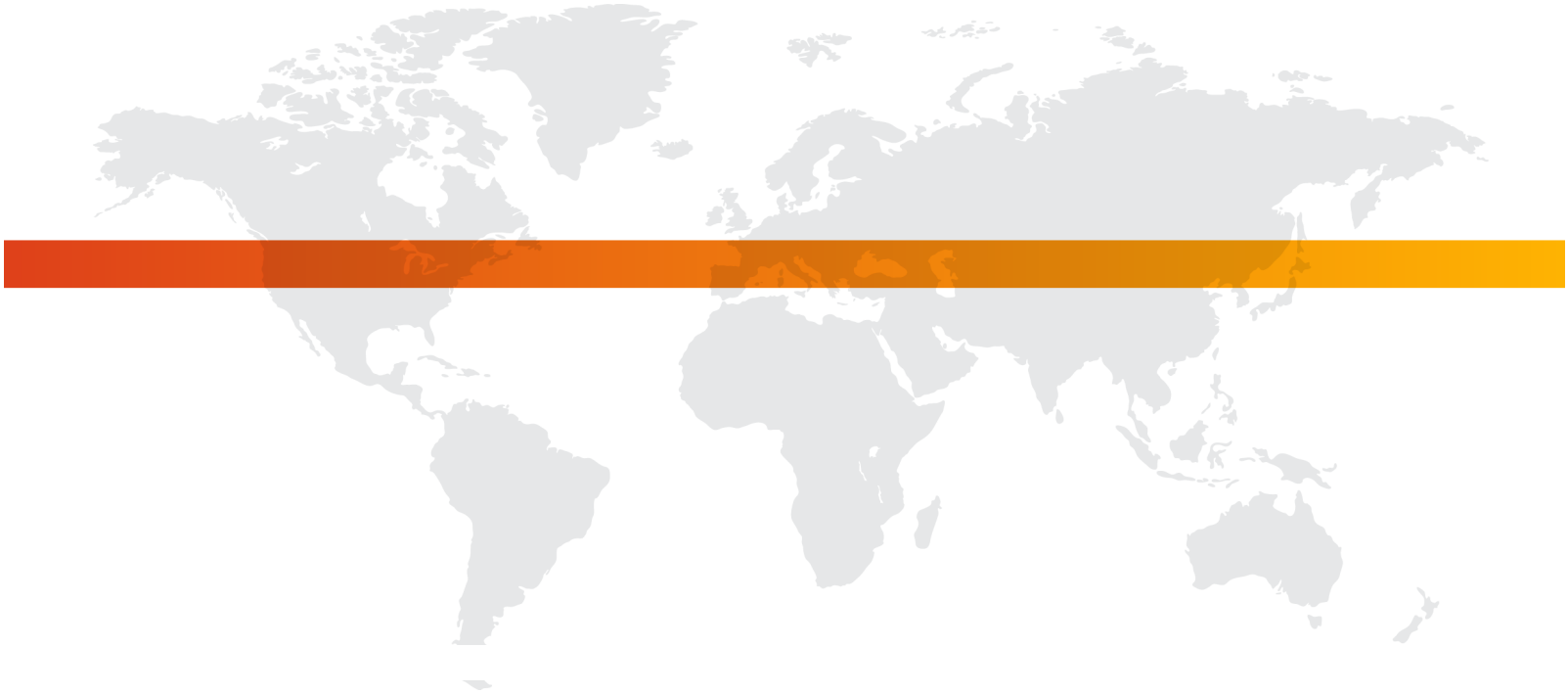
Will comScore receive measurements if my application user is currently roaming?
If the application launches where your user may incur roaming charges, a warning will appear on their screen. However, if the application starts and the user has internet connectivity we will receive the measurements.

Measuring the digital world.™

comScore

Unified Digital Measurement

Inline Tag Code Document



Introduction

This document outlines the process for implementing comScore's Unified Digital Measurement Inline Tag to support data collection for comScore's audience measurement and (if desired) web analytics products. The document provides you with information about tagging conventions and guidelines for implementing the Inline Tag on your web site.

Please note that the Inline Tag consists of two components:

1) The Inline Tag code (with JavaScript measurement code and a `<noscript>` HTML tag)
2) An externally hosted JavaScript file (cs.js)

As a best practice, it is recommended that you copy the Inline Tag code and the HTML `<script>` statement for loading the externally hosted script file from the Inline Tag code text file. This ensures that the syntax of the code is correct.

This text file should be delivered to you together with an electronic version of this document. If you have not received this text file, please request it through comScore Direct or from your Account Manager.

Checklist

Before deploying the tag on your site, please complete the following checklist first:

- Obtain your client ID (referred to as the "c2 parameter") through comScore Direct or from your Account Manager.
- Validate and test that the code has been copied and pasted properly (as displayed in this document) onto a test page.

1. The Inline Tag Code

The Inline Tag code consists of a `<script>` HTML tag and a `<noscript>` HTML tag, which have the following structure:

```
<!-- Begin comScore Inline Tag 1.1111.15 -->
<script type="text/javascript">
// <![CDATA[
function udm_(a){var
b="comScore=",c=document,d=c.cookie,e="",f="indexOf",g="substring",h="length",i=2048,j,k="&ns_",l="&",m,n,o,p,q=window,r=q.encodeURICom
ponent||escape;if(d[f](b)+1)for(o=0,n=d.split(";"),p=n[h];o<p;o++)m=n[o][f](b),m+1&&(e=l+unescape(n[o][g](m+b[h])));a+=k+"_t="+ +(new
Date)+k+"c="+(c.characterSet||c.defaultCharset||"")+"&c8="+r(c.title)+e+"&c7="+r(c.URL)+"&c9="+r(c.referrer),a[h]>i&&a[f](l)>0&&(j=a[g]
(0,i-8).lastIndexOf(l),a=(a[g](0,j)+k+"cut="+r(a[g](j+1)))[g](0,i)),c.images?(m=new
Image,q.ns_p||(ns_p=m),m.src=a):c.write("<","p","><",'img src="',a,'" height="1" width="1" alt="*"','><','/p',">")}
udm_('http'+(document.location.href.charAt(4)=='s'?'s://sb':'://b')+'.scorecardresearch.com/b?c1=2&c2=1234567');
// ]]>
</script>
<noscript><p><img src="http://b.scorecardresearch.com/p?c1=2&amp;c2=1234567" height="1" width="1" alt="*"></p></noscript>
<!-- End comScore Inline Tag -->
```

Please refer to the text file with the Inline Tag code to copy this measurement code.

Tag Placement

The Inline Tag code should be placed immediately (or as close as possible) after the opening `<body>` tag at the top of the HTML web page.

Passing Values to the Inline Tag

The Inline Tag generates simple client-side HTTP get requests to `b.scorecardresearch.com`. The `<script>` part of the Inline Tag automatically detects secure pages and will use HTTPS get requests instead. The URL for such a HTTP get request is what is called the **measurement URL**.

Note: There is small difference between measurement URLs for the <script> and <noscript> part of the Inline Tag. The measurement URL in the `<script>` part needs to use "**/b?**" while the measurement URL in the `<noscript>` part needs to use "**/p?**". **This is a requirement for the tag to work properly.**

The measurement URL includes a query string with additional information. Within this query string, there are a variety of parameters. Most common are the "c-parameters" which represent different pieces of information.

One of these c-parameters is the client ID in the c2 parameter. In the code displayed above a sample c2 parameter with a value of **1234567** is used (and highlighted). Please make sure you provide own client ID as the c2 parameter value in both the script and no script portions of the code.

Note: The examples only show c-parameters c1 and c2. For any tag implementation these two c-parameters are mandatory. The various Audience measurement products support different sets of required and optional c-parameters. **Please make sure all required or optional c-parameters are added as query parameters in the measurement URL!**

No Script Portion of the Inline Tag

The `<noscript>` part of the Inline Tag ensures that a measurement is requested when the browser does not have JavaScript enabled.

Note: Please note that the image request in the measurement code displayed above specifies non-secure requests. This should be used on non-secure pages only. Unlike the `<script>` part of the tag the `<noscript>` part cannot automatically detect if the page is secure.

Websites that contain secure pages should modify the `<noscript>` tag on those secure pages to have it generate secure HTTP get requests. The URL should appear as follows (below is only an example and should only be used as a complete replacement for the `<noscript>` HTML tag as displayed above):

```
8  <noscript><p><img src="https://sb.scorecardresearch.com/p?c1=2&amp;c2=1234567" height="1" width="1" alt="*"></p></noscript>
```

The changes are in the hostname – `sb.scorecardresearch.com` instead of `b.scorecardresearch.com` – and the use of the `https` protocol.

Additional Functionality Supported From the Measurement URL

The following additional functionality is supported directly from the measurement URL in the Inline Tag:

- Digital Analytix page names
- Digital Analytix online campaign tracking
- Digital Analytix on-site search
- Custom information/data collection

This functionality is described in an appendix to this document: “Appendix A – Additional Tag Functionality”. If the appendix did not accompany this document, you can obtain it through comScore Direct or from your Account Manager.

2. The External JavaScript File (cs.js)

The Inline Tag also contains an external JavaScript file that works in tandem with the Inline Tag code. This file is hosted by comScore.

Note: Please make sure to add the cs.js file in the implementation of the Inline Tag. **This is not an optional part**.

Placing the JavaScript (cs.js) on Your Site

The JavaScript file is loaded using the HTML markup code for an HTML `<script>` tag as displayed below.

```
<script language="JavaScript1.3" src="http://b.scorecardresearch.com/c2/1234567/cs.js"></script>
```

This HTML markup code should be placed right (or as as close as possible) before the closing `</body>` tag at the bottom of the HTML web page. The `language` attribute needs to be present and the value needs to be `"JavaScript1.3"`.

Note: The URL of the cs.js file must contain your client ID. In the code displayed above a sample value of **1234567** is used (and highlighted). Please make sure you provide your own client ID.

Websites that contain secure pages should modify the URL to the cs.js file on those secure pages. As with the `<noscript>` part of the Inline Tag the changes are in the hostname – `sb.scorecardresearch.com` instead of `b.scorecardresearch.com` – and the use of the `https` protocol. **.** The URL should appear as follows:

```
<script type="text/javascript" language="JavaScript1.3" src="https://sb.scorecardresearch.com/c2/1234567/cs.js"></script>
```