comscore

Android Library

Implementation Guide

document version: 6.8.0; released on August 25, 2022

for further information, please contact:
Comscore
Tag Support
+1 866 276 6972

Contents

1 Introduction

Use of the Comscore SDK is subject to the licenses and other terms and conditions set forth herein, including the materials provided in the SDK deliverables. Your use of this SDK and/or transmission of data to Comscore constitutes your agreement to these licenses and other terms and conditions, including the Data Sharing Agreement.

In our continuous efforts to provide the market with highest quality audience measurement data, Comscore Media Metrix uses the *Unified Digital Measurement* methodology.

1.1 Unified Methodology

Unified Digital Measurement is a best-of-breed approach that puts the consumer — the human, not the machine — at the center of Comscore measurement and relies on Panel data as well as Census or Server data. Data from the Comscore panel provides a 360° view of the consumer including demographics, cross-visitation, etc. while Census data from tags provides overall, site-specific usage activity. Through the advent of measuring consumption of content and ads, Comscore has been measuring more entities and campaigns as consumer's media consumption grows across channels, devices, and environments. Grounded in rigorous methodology and by tying this data to panel observations, Comscore uses these assets together in what Comscore calls Unified Digital Measurement (UDM).

We're ensuring our new iteration — UDM 2.0 — will stand the test of time. UDM 2.0 combines first-party data from digital publishers and TV networks in a privacy-preserving manner to ensure audiences are represented with the same granularity and precision as you have come to expect. A separate document about UDM 2.0 is available from your Comscore account team or implementation support team.

A key element for UDM 2.0 is that Comscore's audience reporting services like Media Metrix and Video Metrix will use information from panels, enhanced with *first-party* (*1P*) publisher-specific data of the consumer provided by the publisher. Please refer to *Appendix C: How to add 1P data on page 18* for details on providing additional 1P data in tags. 1P Data can consist of:

- Optional 1P identifier data (e.g., obfuscated value of a login identifier)
- Optional demographics data — age group and gender — provided these are available

1.2 UDM 2.0 measurement with Android library

The Android library provides a (Mobile) Audience Measurement solution designed to accurately capture and report on usage measurements for Android applications. A similar solution is available for other popular platforms from which Comscore reports reach and launches.

If you have any questions or concerns about the instructions in this document, or about elements of the Android library, then please contact your Comscore account team or implementation support team.

1.3 Intended use of Android library

The instructions in this document are intended to be used with **version 6.5.0 and subsequent 6.x.y releases** of the Android library for implementation in **Android applications targeting API level 21 and higher developed in Java** code. The Android OS versions which the Android library can be used with are mentioned in its release notes. If your application is developed in another programming language then please contact your Comscore account team to ask for guidance.

1.4 Preparation

Please complete the following checklist before adding the Android library to your Android applications:

1. Familiarize yourself with the instructions in this document.
2. If you are updating an existing implementation, then please refer to *Appendix A: Updating an existing implementation on page 14* to see if there are any relevant steps mentioned for your situation.
3. Confirm your Comscore *Publisher ID* — also known as the *Client ID* or *c2 value* — which is a number with at least 7 digits, provided by Comscore.
4. If you want to manually include the library in your application project — instead of via Comscore's *Maven repository* — then confirm you have received the library binaries.

About retrieving the Publisher ID…

1. Use a web browser to visit *Comscore Direct (http://direct.comscore.com/clients/MobileApp.aspx)* .
2. Log in to Comscore Direct with your user account and password if you are prompted to.
   If you are a Comscore client but do not have a Comscore user account then please contact your client service representative.
   If you are not yet a Comscore client please sign up for a user account through Comscore Direct.
3. Confirm you are on the *Mobile App* tab and click on *Get Tag* to show a popup which contains the Publisher ID.

1.5 Implementation overview and general instructions

The implementation for Android applications involves the following steps:

1. Include the library in the application project.
2. Update application project configuration as needed for the library to provide all its functionality.
3. Add code statements to configure and start the library.
4. Add code statements to indicate changes in user experience as needed (i.e., video and/or audio content playback).
5. Add code statements to indicate application section changes as needed.

1.5.1 Intended use of library elements

As you work with the library you might see classes, methods or properties which do not appear in this documentation. Those library elements are exposed either because the solution requires it or because they are needed for custom solution implementations for which Comscore provides additional instructions.

Please ensure you do not use any library elements which do not appear in this documentation unless you have received explicit instructions for their use from Comscore.

1.5.2 Use `import` statements

Your IDE ("Integrated Development Environment") will likely take care of importing any classes you use from the `com.comscore` package. If it does not, or if you do not use an IDE then please make sure to include appropriate `import` statements to be able to use the class names without the need to specify the package.

1.5.3 Add *App Set ID SDK* to your project

It is advised to add *App Set ID SDK* as a dependency to your project, because this enables the Comscore library to collect the privacy-friendly publisher-specific device identifier option called *App Set ID* for estimating the number of unique users for audience research purposes. However, the Comscore library does not require *App Set ID SDK* to function correctly (i.e., adding this SDK is optional).

```
...
dependencies {
    ...
    implementation 'com.google.android.gms:play-services-appset:16.+'
    ...
}
```

Optionally, you can add additional information about the consumer to enhance your results. Please refer to *Appendix C: How to add 1P data on page 18* for details on providing 1P data.

1.5.4 Applications which might need special instructions

The library should be implemented in the application in such a way that it can capture regular application starts and usage data. If your application uses multiple processes or contains background services then your implementation likely requires special instructions.

If your application uses multiple processes or background services, then please make sure to only use the library from the main process of your application. If your main application process is automatically restarted — for example by using a background service — then please make sure to only initialize the library when the application is brought to the foreground for the first time after being (re)started.

2 Implementation instructions

This section contains the instructions for mandatory and optional parts of a standard implementation.

2.1 Include the library in your application project

To include the *comscore.aar* library file in your application project, you can include it from the *Maven Central Repository (https://search.maven.org/search?q=com.comscore)*. Using the Maven Central Repository makes updating the library to a newer version much easier, compared to manually including the library in your application project.

The library uses the following Android permissions to provide all its functionality properly and includes a manifest which will cause these permissions to be associated with your application project:

- `android.permission.INTERNET`
- `android.permission.ACCESS_NETWORK_STATE`
- `android.permission.ACCESS_WIFI_STATE`
- `com.google.android.gms.permission.AD_ID`

How you include the library from the Maven Central Repository depends on the development environment you are using. For example, with Android Studio IDE you can configure *Gradle* to include the library by adding the following to the *gradle.build* file of your application project:

1. Maven Central as a source repository: `mavenCentral()`.
2. A dependency to implement the Comscore library as part of the application, using *com.comscore:android-analytics:6.+* as the identifier[(1)].

The following example shows what the relevant parts of your *gradle.build* file could look like:

```
allprojects {
    repositories {
        ...
        mavenCentral()
        ...
    }
}
...
dependencies {
    ...
    implementation 'com.google.android.gms:play-services-appset:16.+'
    implementation 'com.comscore:android-analytics:6.+'
    ...
}
```

If you cannot (or do not want to) use the Maven repository and instead prefer to manually add the *comscore.aar* library file to your application project, then please follow the instructions in *Appendix B: Manually including the Comscore Library on page 17*.

---

(1) The occurrence of *:6.+* will cause the library to be updated with new releases on major version 6.

2.2 Configure and start the library

It is strongly advised to configure and start the library from within the `Application.onCreate()` method, implying the Android `Application` class is extended in your application project.

When you configure and start the library from within the `Application.onCreate()` method this ensures optimal operation of the library. Please contact your Comscore account team or implementation support team if you have any questions about when to configure and start the library in your application.

The library is configured by providing publisher-specific configuration settings and — where applicable — general configuration settings, prior to instructing the library to start collecting data.

2.2.1 Available Publisher Configuration Settings

The library configuration settings used for standard solution implementations are listed below. All other configuration settings you might see are for custom solution implementations and should only be used when instructed by Comscore.

Setting	Type	Presence	Example value
`publisherId`	`String`	**mandatory**	`1234567`
	Provide your Publisher ID value. The Publisher ID is often also referred to as the *Client ID* or *c2 value*.		

2.2.2 Apply Publisher Configuration Settings

The appropriate values for your publishers-specific configuration settings are assigned on a `PublisherConfiguration` object using the *Builder* pattern, after which the `PublisherConfiguration` object is provided to the configuration on the `Analytics` singleton object.

```
11. PublisherConfiguration myPublisherConfig = new PublisherConfiguration.Builder()
12.     .publisherId( "1234567" ) // Provide your Publisher ID here.
13.     .build();
14. Analytics.getConfiguration().addClient( myPublisherConfig );
```

2.2.3 Available General Configuration Settings

Aside from the publisher-specific configuration settings the library also offers general configuration settings. For standard solution implementations the available configuration settings are listed below. All other configuration settings you might see are for custom solution implementations and should only be used when instructed by Comscore.

Any general configuration settings that might need to be used, are expected to only be provided by the publisher who 'owns' the application, i.e., the party who makes the application publicly available to users. Partners and other publishers who might be also tagging the application — for example for traffic sharing purposes — are **not** expected to set any of the general configuration settings.

Setting	Type	Presence	Example value
`applicationName`	`String`	**optional**	`News Magazine`
	By default the library retrieves the application name from your application's label, as returned by the `android.content.pm.PackageManager.getApplicationLabel()` method. Should you want to override the automatically retrieved value then you can provide a string with your preferred application name. Please inform your Comscore account team of the collected application name to ensure proper classification for Audience reporting.		
`usagePropertiesAutoUpdateMode`	`Enum`	**optional**	`UsagePropertiesAutoUpdateMode.FOREGROUND_ONLY`

	This setting controls if the library will update application usage times at a regular interval. The available modes on the enum are:
	`UsagePropertiesAutoUpdateMode.FOREGROUND_ONLY`
	Update usage times only when the application is in the foreground. **This is the default mode.**
	`UsagePropertiesAutoUpdateMode.FOREGROUND_AND_BACKGROUND`
	Update usage times when the application is in the foreground and when the application is in the background *while providing a user experience*. If your application can provide a user experience in the background then please select this mode for the best possible measurement of application usage time. Automatic updates of usage times while the application is in the background will only occur if the application is providing a user experience as indicated by calls to the `notifyUxInactive` and `notifyUxInactive` API methods. Please refer to *Indicate changes in the application user experience on page 9* for more details about user experiences.
	`UsagePropertiesAutoUpdateMode.DISABLED`
	Do not update usage times. It is not advised to select this mode.		

`usagePropertiesAutoUpdateInterval`	`int`	**optional**	`120`
	The interval in seconds at which the library automatically updates usage times if the auto-update is enabled. The default value is `60`, which is also the minimum value.		

2.2.4 Apply General Configuration Settings

The appropriate values for general configuration settings are assigned using setter methods of a configuration object on the `Analytics` singleton object.

```
16. Analytics.getConfiguration().setUsagePropertiesAutoUpdateMode(
    UsagePropertiesAutoUpdateMode.FOREGROUND_AND_BACKGROUND );
```

2.2.5 Start the Comscore Library

To instruct the library to start collecting data it is required to provide a reference to the current `Context` in a call to the `start` method after providing configuration settings.

```
21. Analytics.start( context ); // Provide a reference to the (application) context.
```

2.3 Indicate changes in the application user experience

If your application can provide a user experience – like the playback of music or video – **while the application is in the background**, then it is particularly important to implement calls to the user experience notification methods on the library API for the correct measurement of usage time during background activity.

User experience API methods

Method name	Expected location for calls
`notifyUxActive`	When your application **starts** providing the user experience
`notifyUxInactive`	When your application **stops** providing the user experience

Please make sure to only call these methods if your application can provide a user experience **while it is in the background**. If your application cannot provide a user experience while it is in the background, then you should not call these methods.

For example, to notify the library of the start of playback of audiovisual content:

```
Analytics.notifyUxActive();
```

2.4 Indicate application section changes

These instructions are entirely optional and only need to be followed if you want metrics to be reportable per section of your application.

To have metrics reported per section of your application, you can notify the library when the user changes sections by calling the application event notification method `notifyViewEvent`. The `notifyViewEvent` can be supplied with a `HashMap` argument, of which the key/value pairs specify the *Event-specific Labels*.

Labels are name/value pairs used to collect data. Use the `ns_category` *Label* to indicate the name of the section the user has changed to. You should work with your Comscore account team to establish what the `ns_category` label values should be, based on your desired dictionary goals.

Please make sure to always call the application event notification method whenever your application changes to *another* section. Please make sure to provide the label `ns_category` with section names that are suitable for your application. When the user changes to general sections of your application — such as a home screen or a startup splash screen — then please use an empty string as the value of label `ns_category`. The following example shows the use of section name value “news”:

```
41. // The user changes to the "news" section
42. HashMap<String,String> labels = new HashMap<String,String>();
43. labels.put( "ns_category", "news" );
44. Analytics.notifyViewEvent( labels );
```

2.5 Communicate user consent

Applicable privacy and data protection laws and regulations may require companies to capture and/or document a user's consent for measurement. For example, the European Union's General Data Protection Regulation (“GDPR”) and the Privacy and Electronic Communication Directive 2002/58/EC and the California Consumer Privacy Act (“CCPA”) have requirements regarding capturing user consent or providing consumers the ability to opt-out of the sale of personal information, where appropriate. Please note that the implications of applicable privacy and data protection laws and regulations may vary and are best evaluated by each individual business.

2.5.1 Using a Consent Management Platform

If you are using a Consent Management Platform (CMP) which implements IAB Transparency and Consent Framework (TCF) version 2.0 then the Comscore library integrates with the CMP to automatically collect user consent. No additional steps are necessary to enable this integration, other than to ensure the Comscore library is in the application where it can access the CMP data as per the TCF 2.0 technical specification.

2.5.2 Manually Communicating Consent

This section explains the steps to manually communicate user preference (e.g., did a user opt in or out of measurement), where required, for publishers with an existing implementation of a Comscore library when a Consent Management Platform is not present.

Comscore collects data through the use of name/value pairs, typically called ‘labels’ in Comscore tagging implementation documents. Data collection is autonomous and controlled by the Comscore libraries, influenced by library API method calls in the application project source code.

To manually communicate user consent a publisher must **add label** `cs_ucfr` to the Comscore library configuration code statements **as a *Persistent Label***. This will cause the label and its value to be persisted through the application run and included by the Comscore library in all collected data transmissions. In this process the publisher **should not change any other configuration settings**.

The accepted values for the `cs_ucfr` user consent label are:

Label `cs_ucfr` values for communicating user consent

Value	Interpretation	Usage
`0`	User has not given consent or has opted out	Use this value to indicate the user 1. has been asked for consent where the user did not give consent, or 2. enabled the option to opt out (e.g., opt out of the sale of personal information)
`1`	User has given consent	Use this value to indicate the user has been asked for consent where the user has given consent to collect data for measurement
	User has not taken an action	Use an empty string value (i.e., blank) to indicate the user has not taken an action

About including label `cs_ucfr` when not collecting user consent opt-in or when the user consent value is unknown…

In countries that do not require explicit opt-in consent for measurement the following may be applicable:

- If **consent is not collected** for a user, then **do not populate** label `cs_ucfr`.
- If **the user consent value is not known** when the library is configured and started, then **populate label `cs_ucfr` with an empty string value (i.e., blank)** as part of the library configuration. Please populate label `cs_ucfr` as a *Persistent Label* with the appropriate value as soon as the user consent value is known and subsequently notify the library of a *Hidden Event*.

If the user consent value is **not** known at the time the library is configured and started, then the `cs_ucfr` can be populated with an empty string value (i.e., blank) as a *Persistent Label* as part of the library configuration. Regardless of exactly what configuration settings appear, for the purpose of collecting user consent the `persistentLabels` configuration setting needs to be added (or modified) on the `PublisherConfiguration`. This configuration setting accepts persistent labels as a `HashMap` of which the key/value pairs represent label name/value pairs.

For example, assuming the user has given consent, the aforementioned configuration code statements would be changed as follows:

```
11. HashMap<String,String> labels = new HashMap<String,String>();
12. labels.put( "cs_ucfr", "1" );
13. PublisherConfiguration myPublisherConfig = new PublisherConfiguration.Builder()
14.     .publisherId( "1234567" ) // Provide your Publisher ID here.
15.     .persistentLabels( labels )
16.     .build();
17. Analytics.getConfiguration().addClient( myPublisherConfig );
```

To update the user consent value after the library is started — or to set the value in case it was previously unknown — label `cs_ucfr` can be populated with the appropriate value as shown below with a subsequent notification of a *Hidden Event* so Comscore can collect the updated user consent value.

```
41. // Provide your Publisher ID in the getPublisherConfiguration() call.
42. Analytics.getConfiguration().getPublisherConfiguration( "1234567" ).setPersistentLabel( "cs_ucfr", consentValue );
43. Analytics.notifyHiddenEvent();
```

After changing the implementation as instructed in this section the collected data should contain label `cs_ucfr` with its assigned value any time the library transmits collected data. The collected data is transmitted as HTTP requests of which the URL query string parameters should contain label `cs_ucfr` with its assigned value. More details about inspecting HTTP requests can be found in *Test your implementation on page 12*.

2.6 Child directed applications

If the application is considered child directed and/or is categorized within the Comscore Client Focus Dictionary as a “Kids Sub-Category”, the child directed application mode configuration setting should be enabled. When the setting is enabled there is no collection of the advertising identifier within the particular application, regardless of user settings. All other information collected will be processed in accordance with Comscore's internal privacy rules.

To enable this feature you can include a call to the `enableChildDirectedApplicationMode` configuration method immediately before calling the `start` method:

```
16.  Analytics.getConfiguration().enableChildDirectedApplicationMode();
17.  Analytics.start( context );
```

If the application is not child directed please ensure you either request the category change through your Direct Account or speak with your Client Success representative.

If you have followed the implementation instructions up to this point then the library will be collecting data for your application.

3 Test your implementation

As you test your application with the implemented library internally, Comscore servers will collect the analytics data via secure transmissions using the HTTPS protocol. To simplify the process of testing your implementation the library has a “validation mode” feature causing it to output the request URLs on standard output. To enable this feature you can include a call to the `enableImplementationValidationMode` configuration method immediately before calling the `start` method:

```
16.  Analytics.getConfiguration().enableImplementationValidationMode();
17.  Analytics.start( context );
```

With the feature enabled, you will see lines with the text `Comscore:` followed by a URL to a host on the *scorecardresearch.com* domain appear on standard output. You can then use your preferred debugging method to review this output.

Please make sure to remove the call to the `enableImplementationValidationMode` configuration method before deploying the implementation into production.

3.1 How to review your collected data

The collected data will be present on the request URL as query string parameters. There are a number of key query string parameters you can check to confirm the library is implemented correctly:

Key query string parameters to check

Parameter	Description / Comments
`c1`	Fixed value `19`, indicating the collected data is sent from an application
`c2`	The Publisher ID, also known as *Client ID* or *c2 value*
`ns_ap_an`	The application name as it will show up in reporting[2]

Parameter	Description / Comments
ns_ap_sv	The library version number
ns_ap_cs	Number of cold starts of the application
ns_ap_dft	Foreground duration - in milliseconds - of previous sessions(3)
ns_ap_dbt	Background active duration - in milliseconds - of previous sessions(4)

3.2 Execute a simple test scenario

Please execute the following simple test scenario to confirm application cold starts and foreground duration are collected properly:

1. Cold start the application.
   - An HTTP request to Comscore's servers should appear. Please check the value of parameter ns_ap_dft which should have a low value (usually well below 300).
   - Keep the application in the foreground for 15 seconds.
2. Return to the device / operating system's home screen(5), i.e., on an iOS device press the home button.
   - Wait for 30 seconds.
3. Bring the application to the foreground(6).
   - Keep the application in the foreground for 45 seconds.
4. Return to the device / operating system's home screen.
   - Wait for 20 seconds.
5. Terminate the application while it is in the background.
6. Wait 50 seconds.
7. Cold start the application again.
   - An HTTP request to Comscore's servers should appear. Please check the value of parameter ns_ap_dft which should be close to 60000 if you've followed the timings in this scenario(7).

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

(2) Please reach out to your Comscore account team to ensure this value is properly classified for Audience reporting.
(3) The first time the application is cold started after installing it on a device the value of ns_ap_dft will be close to 0. The second time the application is cold started the value of ns_ap_dft will be the time the application was running in the foreground in the previous session.
(4) This is only relevant to applications that can be active when put in the background or minimized, like (streaming) music players or map/navigation applications. For ns_ap_dbt to have a non-zero value the calls for *Indicate changes in the application user experience on page 9* must be implemented.
(5) Returning to the home screen might terminate the application in some environments.
(6) If your app was terminated in the previous step, then an HTTP request to Comscore's servers should appear here. Please check the value of parameter ns_ap_dft which should be close to 15000 if you've followed the timings in this scenario.
(7) If your app was terminated when you went back to the home screen, then the value of parameter ns_ap_dft should be close to 45000 if you've followed the timings in this scenario.

Appendix A: Updating an existing implementation

Updating within the same library major version typically are drop-in replacements. When *upgrading* to a newer major version some code changes might be required as major versions usually include API changes.

Your development environment will typically point out when your code uses API methods which no longer exist or have a different signature. This appendix aims to give an impression of what can be involved in a migration.

> It could be that some of the mentioned library classes, API methods or method arguments do not appear in your implementation. If your implementation contains elements which are not mentioned in these migration instructions then please contact your Comscore account team or implementation support team for additional instructions.

There are a few Comscore library major versions in circulation. You can determine which one you have implemented from the required configuration code statements.

Determine library version

Appearance / Characteristics	Major Version

The configuration code statements look like:
`PublisherConfiguration myPublisherConfig = new PublisherConfiguration.Builder()`
`.publisherId( "1234567" ) // Will mention your Publisher ID.`
`.publisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ) // Will mention your Publisher Secret.`
`.build();`
`Analytics.getConfiguration().addClient( myPublisherConfig );`
	5

The configuration code statements look like:
`comScore.setCustomerC2( "1234567" ); // Will mention your Publisher ID.`
`comScore.setPublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ); // Will mention your Publisher Secret.`
	2 and 3

Migrate from major version 5 to 6

1. Remove any calls to the `publisherSecret` method on `PublisherConfiguration.Builder()`. Typically the following configuration code statements are present, where the highlighted line should be removed:

```
11. PublisherConfiguration myPublisherConfig = new PublisherConfiguration.Builder()
12.     .publisherId( "1234567" ) // Will mention your Publisher ID.
13.     .publisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ) // Will mention your Publisher Secret.
14.     .build();
15. Analytics.getConfiguration().addClient( myPublisherConfig );
```

2. Move an calls to specify an alternative application name on `PublisherConfiguration.Builder()` to the general configuration settings. For example, the highlighted configuration code statement could be present:

```
11. PublisherConfiguration myPublisherConfig = new PublisherConfiguration.Builder()
12.     .publisherId( "1234567" ) // Will mention your Publisher ID.
13.     .publisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ) // Will mention your Publisher Secret.
14.     .applicationName( value ) // Will mention your chosen application name.
15.     .build();
16. Analytics.getConfiguration().addClient( myPublisherConfig );
```

That code statement should be removed and the specified value supplied on the general configuration:

```
PublisherConfiguration myPublisherConfig = new PublisherConfiguration.Builder()
    .publisherId( "1234567" ) // Will mention your Publisher ID.
    .build();
Analytics.getConfiguration().addClient( myPublisherConfig );
Analytics.getConfiguration().setApplicationName( value );
```

3. Move calls to specify alternative usage properties auto-update settings on `PublisherConfiguration.Builder()` to the general configuration settings. For example, the highlighted configuration code statements could be present:

```
PublisherConfiguration myPublisherConfig = new PublisherConfiguration.Builder()
    .publisherId( "1234567" ) // Will mention your Publisher ID.
    .publisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ) // Will mention your Publisher Secret.
    .usagePropertiesAutoUpdateMode( modeValue ) // Will mention an enum value.
    .usagePropertiesAutoUpdateInterval ( intervalValue ) // Will mention a value in seconds
    .build();
Analytics.getConfiguration().addClient( myPublisherConfig );
```

Those code statements should be removed and the specified values supplied on the general configuration:

```
PublisherConfiguration myPublisherConfig = new PublisherConfiguration.Builder()
    .publisherId( "1234567" ) // Will mention your Publisher ID.
    .build();
Analytics.getConfiguration().addClient( myPublisherConfig );
Analytics.getConfiguration().setUsagePropertiesAutoUpdateMode( modeValue );
Analytics.getConfiguration().setUsagePropertiesAutoUpdateInterval( intervalValue );
```

4. Replace occurrences of `setPersistentLabels` to `addPersistentLabels`.

Migrate from major versions 2 and 3 to 6

1. Remove the existing *comscore.jar* library file from the application project.
2. Add the *comscore.aar* library file to the application project as per the instructions in *Include the library in your application project on page 6*.
3. Remove existing import statements which import classes from the `com.comscore.analytics` package.
4. Locate existing library configuration code statements.
   Typically the following configuration code statements are present in an existing library implementation:

```
comScore.setAppContext( this.getApplicationContext() );
comScore.setCustomerC2( clientId );
comScore.setPublisherSecret( publisherSecret );
comScore.enableAutoUpdate( 60, true );
```

5. Replace existing library configuration statements, migrating the settings to be provided to the new API. In the new API *publisherSecret* is no longer used. Migrating the configuration code statements shown above would result in the following:

```
PublisherConfiguration myPublisherConfig = new PublisherConfiguration.Builder()
    .publisherId( clientId )
    .build();
Analytics.getConfiguration().addClient( myPublisherConfig );
Analytics.getConfiguration().setUsagePropertiesAutoUpdateMode(
UsagePropertiesAutoUpdateMode.FOREGROUND_ONLY );
Analytics.getConfiguration().setUsagePropertiesAutoUpdateInterval( 60 );
Analytics.start( this.getApplicationContext() );
```

6. Replace occurrences of into and modify method calls to account for changes in naming[8]:

Existing call	Migrated call
`comScore.onEnterForeground()`	`Analytics.notifyEnterForeground()`
`comScore.onExitForeground()`	`Analytics.notifyExitForeground()`
`comScore.onUxActive()`	`Analytics.notifyUxActive()`
`comScore.onUxInactive()`	`Analytics.notifyUxInactive()`
`comScore.setLabel( name, value )`	`Analytics.getConfiguration().setPersistentLabel( name, value )`
`comScore.setLabels( labels )`	`Analytics.getConfiguration().addPersistentLabels( labels )`
`comScore.setAppName( name )`	`Analytics.getConfiguration().setApplicationName( name );`
`comScore.start()`	Please remove the call.

7. Ensure your IDE has included the relevant imports for the classes which are used in the migrated code statements and that the imports for classes which no longer exist have been removed from your code.

Migrate from using Bintray to using Maven Central Repository

Previously, Comscore offered the library through Bintray where the repository was added to *gradle.build* file in the `repositories` section:

```
maven {
    url"https://comscore.bintray.com/Analytics"
}
```

Additionally, the library was then included as follows in the `dependencies` section:

```
compile "com.comscore:android-analytics:6.+"
```

As Bintray is being sunset on May 1, 2021, the Comscore library will from now on be offered on Maven Central Repository. Please remove the aforementioned lines from the *gradle.build* file and refer to *Include the library in your application project on page 6* for contemporary instructions.

(8) Any existing arguments can be copied as-is, unless specified otherwise.

Appendix B: Manually including the Comscore Library

If you prefer to manually include the *comscore.aar* library file in your application project, then please follow these instructions.

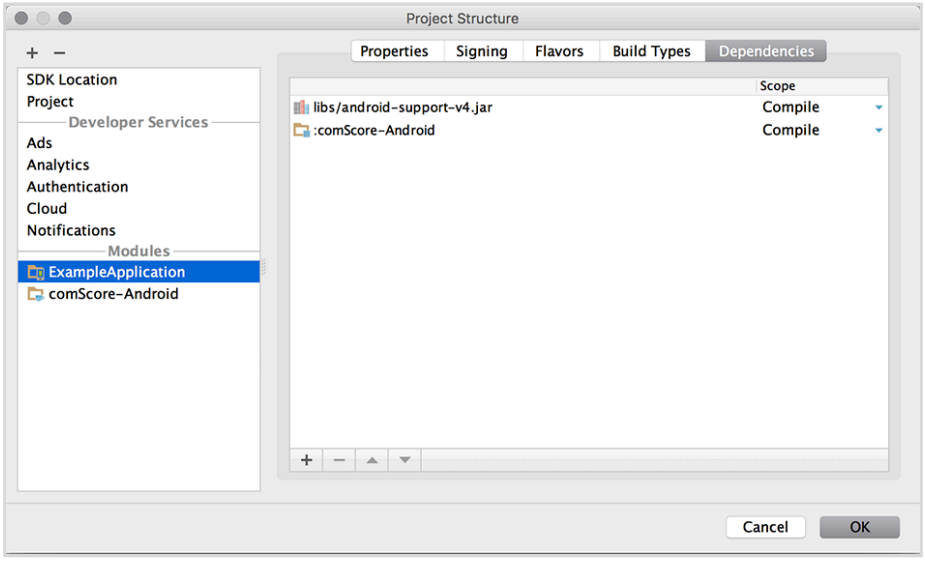
Libraries are manually included in application projects by placing them within the application project folder and adding them to the project configuration to be included in the application. How you include *comscore.aar* library file in your application project depends on the development environment you are using. For example using Android Studio IDE, as shown below.

1. From the *File* menu choose *New* → *Module*.
2. The *Create New Module* modal opens where you should select *Import .JAR/.AAR Package* and click on the *Next* button.
3. For the *File name*, navigate to the *comscore.aar* library file and select it.
4. The name of the library file will should populate *Subproject name*, which you can change to fit your needs[(9)].

Depending on IDE and project configuration the library might not yet be added as a dependency, so that needs to be confirmed.

5. From the *File* menu choose *Project Structure…* which will open the *Project Structure* modal.
6. Confirm the *Subproject name* you added in *step 4* appears in the *Modules* section.
   If it does not appear then please retry the steps to include the *comscore.aar* library file as a module in your application project.
7. In the *Modules* section, select the module which corresponds with your application.
8. On the *Dependencies* panel, confirm the *Subproject name* you added in *step 4* appears in the list.
   If it does not appear then use the *+* button to add the *Subproject name* as a dependency.

This image shows an example of what the *Dependencies* panel could look like after performing these steps:



9. Click on the *OK* button to close the *Project Structure* modal.

---

(9) Typically, changing the pre-populated name to remove the version identification will make updating the library easier as it will then suffice to just replace the *comscore.aar* library file with a newer version.

Appendix C: How to add 1P data

The following optional parameters can be added as *Persistent Labels* as part of the library configuration to provide 1P identifier data and additional demographics data about the consumer:

Tag parameters for 1P data

Parameter	Required or Optional	Description	Explanation	Example value
`cs_fpid`	Optional	1P identifier	Contains the pseudonymized 1P identifier value, which could be either a user_id, login (preferred), OpenID or a first-party cookie.	`1570113661206_6059410249`
`cs_fpit`	Optional	Type of identifier in `cs_fpid`	Specifies the type of identifier ▪ `li`: logged-in ID ▪ `lo`: logged-out ID ▪ `o`: OpenID ▪ `c`: first-party cookie	`c`
`cs_fpdm`	Optional	1P demographics data	Contains an obfuscated value of the demographics data which belong to the 1P identifier. This obfuscation calculation is explained further below this table.	`39642313001`
`cs_fpdt`	Optional	Type of 1P demographics data in `cs_fpdm`	Indicates the origin of the demographics values. Accepted values: ▪ `01`: collected by publisher ▪ `02`: collected through / purchased from third party ▪ `03`: mixed sources or modelled ▪ `99`: unknown origin	`01`

Please ensure that all four parameters are provided together. If you cannot populate a suitable value for any of the parameters — e.g., when demographics data is not available — then please use value `*null` instead.

Demographics data should not be collected for children. If the consumer is a child (ages 0 - 17) then please use value `*null` for `cs_fpdm`.

The 1P identifier is expected to stick to the same consumer in the same application where possible, or even across applications if a user- or login identifier is used. 1P demographics data is calculated as `19991999999` added to to the concatenation of birth date, age group and gender, where birth date has the format `yyyyymmdd`, age group has the format `xx` and gender has the format `z`. In other words: `yyyyymmddxxz + 19991999999`.

Age groups and gender are shown in the following tables. **The age groups in the range 0 - 17 are not listed because demographics data should not be collected for children.**

Age groups identifiers

Identifier	Age group
`00`	00
`06`	18 - 20
`07`	21 - 24
`08`	25 - 34
`09`	35 - 44
`10`	45 - 54
`11`	55 - 64
`12`	65+

Gender identifiers

Identifier	Age group
`0`	Unknown
`1`	Male
`2`	Female
`3`	Unspecified / Other

Age group and birth date are complementary, but the preference is to collect birth date. When birth date is provided, age group can be omitted by using `0` values and vice versa. To illustrate:

Demographics values examples

Demographics of consumer	Demographics value calculation	Description
Female born on March 13, 1965	`19650313002 + 19991999999 = 39642313001`	The age group uses zeroes because the birth date is known
Male in the age between 35-45	`00000000091 + 19991999999 = 19992000090`	The birth date uses zeroes because only the age group was available

The `persistentLabels` configuration setting needs to be added (or modified) on the `PublisherConfiguration`. This configuration setting accepts persistent labels as a `HashMap` of which the key/value pairs represent label name/value pairs. For example, assume the following 1P data values:

- Identifier: `1605266069802_50777152`
- Identifier type is first-party cookie: `c`
- Demographics values: `39642313001`
- Demographics collected by publisher: `01`

In this example the aforementioned configuration code statements would be changed as follows:

```
11. HashMap<String,String> labels = new HashMap<String,String>();
12. labels.put( "cs_fpid", "1605266069802_50777152" );
13. labels.put( "cs_fpit", "c" );
14. labels.put( "cs_fpdm", "39642313001" );
15. labels.put( "cs_fpdt", "01" );
16. PublisherConfiguration myPublisherConfig = new PublisherConfiguration.Builder()
17.     .publisherId( "1234567" ) // Provide your Publisher ID here.
18.     .persistentLabels( labels )
19.     .build();
20. Analytics.getConfiguration().addClient( myPublisherConfig );
```

To update 1P data after the library is started — or to set the value in case it was previously unknown — the label can be populated with appropriates value as shown below, with a subsequent notification of a *Hidden Event* so Comscore can collect the updated user consent value.

For example, assume demographics values were previously unavailable and set as `*null` and have become available after the library was started. The following code updates the labels values and notifies of the Hidden Event.

```
41. // Provide your Publisher ID in the getPublisherConfiguration() call.
42. Analytics.getConfiguration().getPublisherConfiguration( "1234567" ).setPersistentLabel( "cs_fpdm", "39642313001" );
43. Analytics.getConfiguration().getPublisherConfiguration( "1234567" ).setPersistentLabel( "cs_fpdt", "01" );
44. Analytics.notifyHiddenEvent();
```

Appendix D: Frequently Asked Questions

How can I validate my application measurement is working as intended?

There are two ways to validate your application measurements. You could test the measurements yourself by following the instructions in *Test your implementation on page 12*.

In addition, you could start your application at least 10 times and let the application run for at least 30 seconds on each start. After this, you can contact your Comscore account team with your Publisher ID. Your Comscore account team will confirm whether or not Comscore has received your application measurements within two business days.

Will adding the Comscore library slow down the application?

No. The code in the Comscore library is extremely lightweight and should not affect the performance of your application.

How can I change my application name that is reported to Comscore?

The reported application name is automatically retrieved by the library. Before you call any of the Comscore library's notification methods, you can change your reported application name by providing your preferred value in the startup configuration. To review which source the library uses to automatically retrieve the application name or to learn more about the *Application Name* startup configuration parameter, please refer to *Start the Comscore Library on page 9*.

Please reach out to your Comscore account team to ensure this value is properly classified for Audience reporting.

My project uses ProGuard. Does the Comscore library require some specific ProGuard configuration?

To be able to compile an application using ProGuard the following lines should be added to `proguard-project.txt`:

```
-keep class com.comscore.** { *; }
-dontwarn com.comscore.**
```

The Comscore library uses static classes and the code is already optimized. These settings inform ProGuard to add the library as-is.

Do I have to re-submit my app?

Yes. Once you have tested and confirmed your implementation of the Comscore library, resubmit your application. If the changes you have made since your last application update are only related to the library then your applications should pass review in a timely manner.

Will Comscore receive measurements if my application user is not connected to the internet?

Yes. If device has no internet connectivity, all the measurements are being collected. They will be received by Comscore once internet connectivity is re-enabled.

Will Comscore receive measurements if my application user is currently roaming?

If an application launches where your user may incur roaming charges, the operating system of the device might display a warning on the screen. When the application is active and the device has internet connectivity Comscore will receive the measurements.