comscore

iOS, iPadOS, and tvOS Libraries

Implementation Guide

document version: 6.12.0; released on July 11, 2024

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 24* 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 Darwin libraries

The Darwin libraries provide a (Mobile) Audience Measurement solution designed to accurately capture and report on usage measurements for iOS, iPadOS, or tvOS 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 Darwin libraries, then please contact your Comscore account team or implementation support team.

1.3 Intended use of Darwin libraries

The instructions in this document are intended to be used with **version 6.12.0 and subsequent 6.x.y releases** of the Darwin libraries for implementation in **iOS, iPadOS, or tvOS applications developed in Objective-C or Swift** code. The iOS, iPadOS and tvOS versions which the Darwin libraries can be used with are mentioned in their 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 Darwin libraries to your iOS, iPadOS, or tvOS 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 18* 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 *Swift Package Manager* or *CocoaPods* — 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 iOS, iPadOS, or tvOS 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 Applications which might need special instructions

The library should only be configured and started if the application provides a user experience (i.e., its UI is visible). If your application can be launched from notifications then your implementation likely requires special instructions.

If your application can be launched from notifications, please confirm *the reason for launching the application (https://developer.apple.com/documentation/uikit/uiapplicationlaunchoptionskey?language=objc)* , by inspecting the `launchOptions` *provided to* `didFinishLaunchingWithOptions` *(https://developer.apple.com/documentation/uikit/uiapplicationdelegate/1622921-application?language=objc)* . Ensure the library is *only* started when your application was launched with an appropriate reason and provides a user experience (i.e., its UI is visible).

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 appropriate *ComScore.xcframework* in your application project, you can include it via Swift Package Manager or *CocoaPods*. Using Swift Package Manager or CocoaPods makes updating the library to a newer version much easier compared to manually including the library in your application project.

If you cannot (or do not want to) use Swift Package Manager or CocoaPods and instead prefer to manually include the appropriate *ComScore.xcframework* in your application project, then please follow the instructions in *Appendix B: Manually Including the Comscore Library on page 23*.

2.1.1 Include via Swift Package Manager

You can include the Comscore library as a *Package Dependency* in your project settings by using this *Package URL*: https://github.com/comScore/Comscore-Swift-Package-Manager.

It is advised to always use the latest version of the Comscore library, which Xcode will do by default by using *Branch* as the *Dependency Rule*. You may want to limit this to the current major version by using *Up to Next Major version* as the *Dependency Rule* and specifying `6.0.0` as the version.

2.1.2 Include via CocoaPods

If you are already using CocoaPods then you can include the library in your project by adding the following line in your *Podfile*:

```
pod 'Comscore', '~> 6.0'
```

The library is delivered as two frameworks: one for iOS and iPadOS targets and one for tvOS targets, all supported from the same *Podspec*. If your project contains more than one target then you need to follow the CocoaPods instructions to configure your *Podfile* for multiple targets.

2.1.3 Import headers

To be able to use the elements of the library you need to import the necessary headers.

For Projects Using Objective-C

Xcode will likely suggest to import headers when you address the library in your code. Please make sure to import the umbrella header file *<ComScore/ComScore.h>* in any locations where you need to access the library to ensure all necessary header files are automatically included:

```
#import <ComScore/ComScore.h>
```

For Projects Using Swift

To import the library in a project using Swift you need a bridging header to expose Objective-C classes in your Swift code. If you have other Objective-C code in your Swift project then you will already have a bridging header file. Otherwise you will need to create a bridging header file according to Apple's instructions for adding Objective-C code to the Swift version used in your project.

When created, the bridging header file will typically be named after your project - or module - with the suffix *-bridging-header.h*. Please make sure to include the following statement in the bridging header file, to have the Comscore library classes exposed in your Swift project:

```
#import <Comscore/Comscore.h>
```

2.1.4 Add *AdSupport.framework* to Your Project Target(s)

It is advised to add *AdSupport.framework* to your project target(s) even if the application does not contain advertisements because this enables the Comscore library to collect the `advertisingIdentifier` value for estimating the number of unique users for audience research purposes. However, the Comscore library does not require *AdSupport.framework* to function correctly (i.e., adding this framework is optional).

The Comscore library adheres to Apple's policies and checks for the user's permission prior to collecting the `advertisingIdentifier` value:

- On iOS, iPadOS and tvOS 13 or older the library uses `advertisingTrackingEnabled`.
- On iOS, iPadOS and tvOS 14 the library uses *AppTrackingTransparency.framework* if the framework is present in the project targets and implemented for use. Otherwise the library falls back to inspecting the `advertisingIdentifier` value to determine if it is allowed to be collected.
- On iOS, iPadOS and tvOS 17 the library continues to use the *AppTrackingTransparency.framework* if the framework is present in the project targets and implemented for use.
  - The library also contains a *Privacy Manifest* which declares the library could collect the `advertisingIdentifier` when it's made available.

If *AppTrackingTransparency.framework* is implemented for use in your application, then you will have set the `NSUserTrackingUsageDescription` key in the *Information Property List* (*Info.plist*). To ensure your description covers all your specific use cases for requesting the users permission, please be aware the Comscore library collects the `advertisingIdentifier` value for analytics and tracking purposes.

About App Store submissions...

As part of the process for submitting the app to the App Store it is required to indicate the reasons for using the *Advertising Identifier* by ticking the appropriate check boxes. If you have included *AdSupport.framework* in your application as per the above instructions, then please make sure to tick the following check boxes when you submit the app to the App Store:

- *'Attribute this app installation to a previously served advertisement'*
- *'Attribute an action taken within this app to a previously served advertisement'*

If your application also contains advertisements, then please also make sure to tick the checkbox for *'Serve advertisements within the app'*.

The Comscore library adheres to the *Limit Ad Tracking* setting. This means that you can safely tick the checkbox for *'Limit Ad Tracking setting in iOS'* or *'Limit Ad Tracking setting in tvOS'* as far as the Comscore library is concerned.

Please contact your Comscore account team or Comscore Tag Support with any questions about the use of the Advertising Identifier within the Comscore library.

If your application also contains advertisements, then please also ensure the *Advertising data* data type — key `NSPrivacyCollectedDataType` with value `NSPrivacyCollectedDataTypeAdvertisingData` — is mentioned in the *Privacy Manifest* of your application.

2.1.5 Collection of first-party device identifiers

The Comscore library automatically collects the privacy-friendly publisher-specific device identifier called *Identifier for Vendors* (*IDFV*) for estimating the number of unique users for audience research purposes. 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 24* for details on providing 1P data.

2.2 Configure and start the library

It is strongly advised to configure and start the library from within the `application didFinishLaunchingWithOptions` method in an application project using Objective-C, or the equivalent location in a project using Swift.

When you configure and start the library from within the aforementioned location 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 publisher-specific configuration settings are assigned on a `SCORPublisherConfiguration` object using the *Builder* pattern, after which the `SCORPublisherConfiguration` object is provided to the configuration on the `SCORAnalytics` singleton object.

For Projects using Objective-C

```
SCORPublisherConfiguration *myPublisherConfig = [SCORPublisherConfiguration
publisherConfigurationWithBuilderBlock:^(SCORPublisherConfigurationBuilder *builder) {
    builder.publisherId = @"1234567"; // Provide your Publisher ID here.
}];
[[SCORAnalytics configuration] addClientWithConfiguration:myPublisherConfig];
```

For Projects using Swift

```
let myPublisherConfig = SCORPublisherConfiguration(builderBlock: { builder in
    builder?.publisherId = "1234567"
})
SCORAnalytics.configuration().addClient(with: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 Comscore library retrieves the application name from your application's *Info.plist* application bundle name as returned by `CFBundleName`. 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**	`SCORUsagePropertiesAutoUpdateModeForegroundOnly`

	This setting controls if the library will update application usage times at a regular interval. The available modes on the enum are:
	
	`SCORUsagePropertiesAutoUpdateModeForegroundOnly`
	Update usage times only when the application is in the foreground. **This is the default mode.**
	
	`SCORUsagePropertiesAutoUpdateModeForegroundAndBackground`
	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 11* for more details about user experiences.
	
	`SCORUsagePropertiesAutoUpdateModeDisabled`
	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 as properties of a configuration object on the `Analytics` singleton object.

For Projects using Objective-C

```
16. [SCORAnalytics configuration].usagePropertiesAutoUpdateMode =
    SCORUsagePropertiesAutoUpdateModeForegroundAndBackground;
```

For Projects using Swift

```
16. SCORAnalytics.configuration().usagePropertiesAutoUpdateMode =
    SCORUsagePropertiesAutoUpdateModeForegroundAndBackground
```

2.2.5 Start the Comscore Library

To instruct the library to start collecting data it is required to call to the `start` method after providing configuration settings.

For Projects using Objective-C

```
21. [SCORAnalytics start];
```

For Projects using Swift

```
21. SCORAnalytics.start()
```

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:

For Projects using Objective-C

```
[SCORAnalytics notifyUxActive];
```

For Projects using Swift

```
SCORAnalytics.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 `Dictionary` 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”:

For Projects using Objective-C

```
// The user changes to the "news" section
[SCORAnalytics notifyViewEventWithLabels:@{
    @"ns_category": @"news"
}];
```

For Projects using Swift

```
// The user changes to the "news" section
SCORAnalytics.notifyViewEvent(withLabels: [
    "ns_category": "news"
]);
```

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 2.2 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 2.2 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 `SCORPublisherConfiguration`. This configuration setting accepts persistent labels as a `Dictionary` 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:

For Projects using Objective-C

```
SCORPublisherConfiguration *myPublisherConfig = [SCORPublisherConfiguration
publisherConfigurationWithBuilderBlock:^(SCORPublisherConfigurationBuilder *builder) {
    builder.publisherId = @"1234567"; // Provide your Publisher ID here.
    builder.persistentLabels = @{ @"cs_ucfr": @"1" }
}];
[[SCORAnalytics configuration] addClientWithConfiguration:myPublisherConfig];
```

For Projects using Swift

```
let myPublisherConfig = SCORPublisherConfiguration(builderBlock: { builder in
    builder?.publisherId = "1234567" // Provide your Publisher ID here.
    builder?.persistentLabels = [ "cs_ucfr": "1" ]
})
SCORAnalytics.configuration().addClient(with: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.

For Projects using Objective-C

```
// Provide your Publisher ID in the publisherConfigurationWithPublisherId call.
[((SCORClientConfiguration *) [[SCORAnalytics configuration] publisherConfigurationWithPublisherId:@"1234567"])
setPersistentLabelWithName:@"cs_ucfr" value:consentValue];
[SCORAnalytics notifyHiddenEvent];
```

For Projects using Swift

```
// Provide your Publisher ID in the publisherConfiguration call.
SCORAnalytics.configuration().publisherConfiguration(withPublisherId:"1234567").setPersistentLabelWithName("cs_ucfr",
value:consentValue)
SCORAnalytics.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 15*.

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:

For Projects using Objective-C

```
16. [[SCORAnalytics configuration] enableChildDirectedApplicationMode];
17. [SCORAnalytics start];
```

For Projects using Swift

```
16. SCORAnalytics.configuration().enableChildDirectedApplicationMode()
17. SCORAnalytics.start()
```

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:

For Projects using Objective-C

```
16. [[SCORAnalytics configuration] enableImplementationValidationMode];
17. [SCORAnalytics start];
```

For Projects using Swift

```
16. SCORAnalytics.configuration().enableImplementationValidationMode()
17. SCORAnalytics.start()
```

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(1)
`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(2)
`ns_ap_dbt`	Background active duration - in milliseconds - of previous sessions(3)

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(4), i.e., on an iOS device press the home button.
   - Wait for 30 seconds.
3. Bring the application to the foreground(5).
   - Keep the application in the foreground for 45 seconds.

---

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

(2) 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.

(3) 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 11* must be implemented.

(4) Returning to the home screen might terminate the application in some environments.

(5) 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.

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[(6)].

Once you have verified the library is correctly implemented you must resubmit your application to the Apple App Store for approval.

(6) 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:
 	
For Projects using Objective-C
 	
`SCORPublisherConfiguration *myPublisherConfig = [SCORPublisherConfiguration publisherConfigurationWithBuilderBlock:^(SCORPublisherConfigurationBuilder *builder) {`
`    builder.publisherId = @"1234567"; // Will mention your Publisher ID.`
`    builder.publisherSecret = @"7b94840eb66b17e61c3d2f909c3a1163"; // Will mention your Publisher Secret.`
`}];`
`[[SCORAnalytics configuration] addClientWithConfiguration:myPublisherConfig];`
 	
For Projects using Swift
 	
`let myPublisherConfig = SCORPublisherConfiguration(builderBlock: { builder in`
`    builder?.publisherId = "1234567" // Will mention your Publisher ID.`
`    builder?.publisherSecret = "7b94840eb66b17e61c3d2f909c3a1163" // Will mention your Publisher Secret.`
`})`
`SCORAnalytics.configuration().addClientWithConfiguration(myPublisherConfig)`
	5

The configuration code statements look like:
 	
For Projects using Objective-C
 	
`[CSComScore setCustomerC2:@"1234567"]; // Will mention your Publisher ID.`
`[CSComScore setPublisherSecret:@"7b94840eb66b17e61c3d2f909c3a1163"]; // Will mention your Publisher Secret.`
 	
For Projects using Swift
 	
`CSComScore.customerC2 = "1234567" // Will mention your Publisher ID.`
`CSComScore.publisherSecret = "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 the `builder`. Typically the following configuration code statements are present, where the highlighted line should be removed:

   ### For Projects using Objective-C

   ```
   SCORPublisherConfiguration *myPublisherConfig = [SCORPublisherConfiguration
   publisherConfigurationWithBuilderBlock:^(SCORPublisherConfigurationBuilder *builder) {
       builder.publisherId = @"1234567"; // Will mention your Publisher ID.
       builder.publisherSecret = @"7b94840eb66b17e61c3d2f909c3a1163"; // Will mention your Publisher Secret.
   }];
   [[SCORAnalytics configuration] addClientWithConfiguration:myPublisherConfig];
   ```

   ### For Projects using Swift

   ```
   let myPublisherConfig = SCORPublisherConfiguration(builderBlock: { builder in
       builder?.publisherId = "1234567" // Will mention your Publisher ID.
       builder?.publisherSecret = "7b94840eb66b17e61c3d2f909c3a1163" // Will mention your Publisher Secret.
   })
   SCORAnalytics.configuration().addClientWithConfiguration(myPublisherConfig)
   ```

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

   ### For Projects using Objective-C

   ```
   SCORPublisherConfiguration *myPublisherConfig = [SCORPublisherConfiguration
   publisherConfigurationWithBuilderBlock:^(SCORPublisherConfigurationBuilder *builder) {
       builder.publisherId = @"1234567"; // Will mention your Publisher ID.
       builder.publisherSecret = @"7b94840eb66b17e61c3d2f909c3a1163"; // Will mention your Publisher Secret.
       builder.applicationName = value; // Will mention your chosen application name.
   }];
   [[SCORAnalytics configuration] addClientWithConfiguration:myPublisherConfig];
   ```

   ### For Projects using Swift

   ```
   let myPublisherConfig = SCORPublisherConfiguration(builderBlock: { builder in
       builder?.publisherId = "1234567" // Will mention your Publisher ID.
       builder?.publisherSecret = "7b94840eb66b17e61c3d2f909c3a1163" // Will mention your Publisher Secret.
       builder?.applicationName = value // Will mention your chosen application name.
   })
   SCORAnalytics.configuration().addClientWithConfiguration(myPublisherConfig)
   ```

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

   ### For Projects using Objective-C

   ```
   SCORPublisherConfiguration *myPublisherConfig = [SCORPublisherConfiguration
   publisherConfigurationWithBuilderBlock:^(SCORPublisherConfigurationBuilder *builder) {
       builder.publisherId = @"1234567"; // Will mention your Publisher ID.
   }];
   [[SCORAnalytics configuration] addClientWithConfiguration:myPublisherConfig];
   [SCORAnalytics configuration].applicationName = value;
   ```

   ### For Projects using Swift

   ```
   let myPublisherConfig = SCORPublisherConfiguration(builderBlock: { builder in
   ```

```
    builder?.publisherId = "1234567" // Will mention your Publisher ID.
})
SCORAnalytics.configuration().addClientWithConfiguration(myPublisherConfig)
SCORAnalytics.configuration().applicationName = value
```

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

For Projects using Objective-C

```
SCORPublisherConfiguration *myPublisherConfig = [SCORPublisherConfiguration
publisherConfigurationWithBuilderBlock:^(SCORPublisherConfigurationBuilder *builder) {
    builder.publisherId = @"1234567"; // Will mention your Publisher ID.
    builder.publisherSecret = @"7b94840eb66b17e61c3d2f909c3a1163"; // Will mention your Publisher Secret.
    builder.usagePropertiesAutoUpdateMode = modeValue; // Will mention an enum value.
    builder.usagePropertiesAutoUpdateInterval = intervalValue; // Will mention a value in seconds.
}];
[[SCORAnalytics configuration] addClientWithConfiguration:myPublisherConfig];
```

For Projects using Swift

```
let myPublisherConfig = SCORPublisherConfiguration(builderBlock: { builder in
    builder?.publisherId = "1234567" // Will mention your Publisher ID.
    builder?.publisherSecret = "7b94840eb66b17e61c3d2f909c3a1163" // Will mention your Publisher Secret.
    builder?.usagePropertiesAutoUpdateMode = modeValue // Will mention an enum value.
    builder?.usagePropertiesAutoUpdateInterval = intervalValue // Will mention a value in seconds.
})
SCORAnalytics.configuration().addClientWithConfiguration(myPublisherConfig)
```

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

For Projects using Objective-C

```
SCORPublisherConfiguration *myPublisherConfig = [SCORPublisherConfiguration
publisherConfigurationWithBuilderBlock:^(SCORPublisherConfigurationBuilder *builder) {
    builder.publisherId = @"1234567"; // Will mention your Publisher ID.
}];
[[SCORAnalytics configuration] addClientWithConfiguration:myPublisherConfig];
[SCORAnalytics configuration].usagePropertiesAutoUpdateMode = modeValue;
[SCORAnalytics configuration].usagePropertiesAutoUpdateInterval = intervalValue;
```

For Projects using Swift

```
let myPublisherConfig = SCORPublisherConfiguration(builderBlock: { builder in
    builder?.publisherId = "1234567" // Will mention your Publisher ID.
})
SCORAnalytics.configuration().addClientWithConfiguration(myPublisherConfig)
SCORAnalytics.configuration().usagePropertiesAutoUpdateMode = modeValue
SCORAnalytics.configuration().usagePropertiesAutoUpdateInterval = intervalValue
```

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

Migrate from major versions 2 and 3 to 6

1. Remove the existing *comscore* folder from the *Application Project*.

2. Remove *libComscore.a* from the *Build Phases* panel of your project target options.
3. If *Library Search Paths* in the *Build Settings* panel of your project target options is still pointing to the old Comscore library location, then remove that location.
4. Add the *ComScore.xcframework* to your project as per the instructions in *Include the library in your application project on page 6*.

About the code statements in an existing Comscore library implementation…

It could be that some of the mentioned Comscore 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 Comscore Tag Support for additional instructions.

5. Replace occurrences of `#import "CScomScore.h"`, `#import "CSStreamSense.h"` and `#import "CSStreamingTag.h"` with `#import <ComScore/ComScore.h>`.
6. Locate the existing library configuration code statements.
   Typically the following configuration code statements would appear in an existing Comscore library implementation:

```
[CSComScore setAppContext];
[CSComScore setCustomerC2:clientId];
[CSComScore setPublisherSecret:publisherSecret];
[CSComScore enableAutoUpdate:60 foregroundOnly:YES];
```

7. Replace existing library configuration statements, migrating the settings to be provided to the new API.
   Migrating the configuration code statements shown above would result in the following:

```
SCORPublisherConfiguration *myPublisherConfig = [SCORPublisherConfiguration
publisherConfigurationWithBuilderBlock:^(SCORPublisherConfigurationBuilder *builder) {
    builder.publisherId = clientId;
}];
[SCORAnalytics configuration].usagePropertiesAutoUpdateMode =
SCORUsagePropertiesAutoUpdateModeForegroundOnly;
[SCORAnalytics configuration].usagePropertiesAutoUpdateInterval = 60;
[[SCORAnalytics configuration] addClientWithConfiguration:myPublisherConfig];
[SCORAnalytics start];
```

8. Replace occurrences of `CSComScore` with `SCORAnalytics` and modify method calls to account for changes in naming[7]:

Existing call	Migrated call
`[CSComScore onEnterForeground]`	`[SCORAnalytics notifyEnterForeground]`
`[CSComScore onExitForeground]`	`[SCORAnalytics notifyExitForeground]`
`[CSComScore onUxActive]`	`[SCORAnalytics notifyUxActive]`
`[CSComScore onUxInactive]`	`[SCORAnalytics notifyUxInactive]`
`[CSComScore setLabel:name value:value]`	`[[SCORAnalytics configuration] setPersistentLabelWithName:name value:value]`
`[CSComScore setLabels:labels]`	`[[SCORAnalytics configuration] addPersistentLabels:labels]`
`[CSComScore setAppName:name]`	`[SCORAnalytics configuration].applicationName = name`

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

Existing call	Migrated call
`[CSComScore start]`	Please remove the call.

Appendix B: Manually Including the Comscore Library

If you prefer to manually include the appropriate *ComScore.xcframework* libraries in your application project, then please follow these instructions.

With the Xcode IDE ("Integrated Development Environment") frameworks are typically placed inside the application project and added to the *Frameworks* folder in the *Project Explorer*.

1. Put the appropriate *ComScore.xcframework* library folder inside your application project folder.
2. Drag the *ComScore.xcframework* library folder into the *Frameworks* folder in the *Project Explorer*.
3. Ensure the library is available for linking.
    a. Go to the *Build Phases* panel of your project target options.
    b. Expand *Link Binary with Libraries* to reveal the list of frameworks and libraries that will be linked when building a binary.
    c. Ensure *ComScore.xcframework* appears in the list beneath *Link Binary with Libraries* and otherwise drag *ComScore.xcframework* from the *Project Explorer* into the list. This will also automatically cause the *ComScore.xcframework* library folder to be added to the *Library Search Paths* in the *Build Settings* panel of your project target options.

The Comscore library requires the *SystemConfiguration.framework* and *Security.framework* frameworks as well as the *C++ Standard Library* (i.e., *libc++*) to be present in the compiled application. There is no need to add those frameworks and that library to your project as long as the *Enable Modules (C and Objective-C)* and *Link Frameworks Automatically* settings are both enabled in the *Apple LLVM - Language Modules* section on the *Build Settings* panel of your project target options in Xcode, which will cause the aforementioned frameworks and library to automatically be linked against when your project is compiled.

By default these settings are enabled on projects created with Xcode[(8)]. If your project settings differ from what is described above then you will likely need to add the mentioned frameworks and library to your project targets.

---

(8) This applies to any projects created with Xcode 5 or newer.

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 If `cs_fpit` is not included alongside `cs_fpid` then Comscore will assume the identifier is a 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 `SCORPublisherConfiguration`. This configuration setting accepts persistent labels as a `Dictionary` 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:

For Projects using Objective-C

```
SCORPublisherConfiguration *myPublisherConfig = [SCORPublisherConfiguration
publisherConfigurationWithBuilderBlock:^(SCORPublisherConfigurationBuilder *builder) {
    builder.publisherId = @"1234567"; // Provide your Publisher ID here.
    builder.persistentLabels = @{
        @"cs_fpid": @"1605266069802_50777152",
        @"cs_fpit": @"c",
        @"cs_fpdm": @"39642313001",
        @"cs_fpdt": @"01"
    }
}];
[[SCORAnalytics configuration] addClientWithConfiguration:myPublisherConfig];
```

For Projects using Swift

```
let myPublisherConfig = SCORPublisherConfiguration(builderBlock: { builder in
    builder?.publisherId = "1234567" // Provide your Publisher ID here.
    builder?.persistentLabels = [
        "cs_fpid": "1605266069802_50777152",
        "cs_fpit": "c",
```

```
            "cs_fpdm": "39642313001",
            "cs_fpdt": "01"
        ]
})
SCORAnalytics.configuration().addClient(with: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.

For Projects using Objective-C

```
// Provide your Publisher ID in the publisherConfigurationWithPublisherId call.
[((SCORClientConfiguration *) [[SCORAnalytics configuration] publisherConfigurationWithPublisherId:@"1234567"])
setPersistentLabelWithName:@"cs_fpdm" value:@"39642313001"];
[((SCORClientConfiguration *) [[SCORAnalytics configuration] publisherConfigurationWithPublisherId:@"1234567"])
setPersistentLabelWithName:@"cs_fpdt" value:@"01"];
[SCORAnalytics notifyHiddenEvent];
```

For Projects using Swift

```
// Provide your Publisher ID in the publisherConfiguration call.
SCORAnalytics.configuration().publisherConfiguration(withPublisherId:"1234567").setPersistentLabelWithName("cs_fpdm",
value:"39642313001")
SCORAnalytics.configuration().publisherConfiguration(withPublisherId:"1234567").setPersistentLabelWithName("cs_fpdt",
value:"01")
SCORAnalytics.notifyHiddenEvent()
```

About declaring collection of first-party data in your application's Privacy Manifest...

If you add a 1P identifier value, then please also ensure the *User ID* data type — key `NSPrivacyCollectedDataType` with value `NSPrivacyCollectedDataTypeUserID` — is mentioned in the *Privacy Manifest* of your application. Please indicate it can be used for tracking and is linked to the user's identity.

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 15*.

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 11*.

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

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 Comscore 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.