comscore

JavaScript Library

Implementation Guide

document version: 5.10.0; released on November 8, 2023

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 B: How to add 1P data on page 19* 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 JavaScript library

The JavaScript library provides a (Mobile) Audience Measurement solution designed to accurately capture and report on usage measurements for streaming media players intended to be shown on web pages or OTT applications. A similar solution is available for other popular platforms from which Comscore reports reach and launches.

Comscore combines 1P data from digital publishers and TV networks in a privacy-preserving manner to ensure your audiences are represented with the same granularity and precision as you have come to expect. For PC and Mobile web browsers runtime environments the tag tries to set a third-party cookie. The `PlatformAPI` for these environments (`analytics.PlatformApi.PlatformApis.WebBrowser`) also offers an opt-in functionality which will let it additionally try to set a publisher-specific first-party cookie with the name `_scor_uid`.

While the first-party cookie functionality is disabled by default, **it is strongly advised to enable the first-party cookie functionality feature** by using the `enableFirstPartyCookie` configuration setting, as shown in the implementation instructions to *Configure and start the library on page 8*. Optionally, you can add additional information about the consumer to enhance your results. Please refer to *Appendix B: How to add 1P data on page 19* for details on providing 1P data.

Please note this first-party cookie functionality feature **is not available** in the `PlatformAPI` for other runtime environments.

About first-party cookie lists...

If you enable the first-party cookies in your PC and Mobile web browsers runtime environments and you maintain a list of first-party cookies — for example as part of your privacy statement — then **please make sure to add the `_scor_uid` cookie** to your list. Its purpose is Audience Measurement. Please be aware the Comscore Publisher Tag uses this cookie in accordance with policies surrounding user consent signals as described in *Communicate user consent on page 12*.

The JavaScript library will continue to collect data for the same purposes as approved with your current implementation.

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

1.3 Intended use of JavaScript library

The instructions in this document are intended to be used with **version 7.8.0 and subsequent 7.x.y releases** of the JavaScript library for implementation using JavaScript code in any these environments:

- streaming media players in web sites or web applications intended for PC and Mobile web browsers like Chrome, Safari or Microsoft Edge
- Sony Playstation 3 and 4 as well as other Sony devices which provide a Trilithium or JSMAF runtime environment.
- Samsung Legacy Platform SmartTV TV sets and Bluray devices
- LG TV sets with a Netcast or Cordova runtime environment
- Cordova 2.7 TV apps
- AppleTV 3rd generation
- AppleTV tvOS
- Chromecast Custom Receiver
- Xbox One WinJS environment
- WebOS environment
- Node.js environment
- OTT Applications — not intended for PC and Mobile web browsers — that use generic HTML5/CSS/JavaScript

This documentation refers to the JavaScript code for all these environments as “application” even though you might not consider your web page environment to be an application.

If you are using a different kind of environment or 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 JavaScript library to your streaming media players intended to be shown on web pages or OTT 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 16* 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.

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 streaming media players intended to be shown on web pages or OTT 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.

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

Typically, the file with the Comscore JavaScript library will be named *comscore.js* (it may include a version as part of the file name). This file needs to be part of your application project. To use the library API you need to get a reference to it.

The code examples in this document assume you have created a reference called `analytics` as is used in this example code.

For OTT application use cases

Please include the file with the Comscore JavaScript library in your application project. Please create your own reference to the library API as a shorthand in your application source code:

```
let analytics = ns_.analytics; // Create your own reference.
```

Please note that the library can also be included as a CommonJS module:

```
const analytics = require('./comscore'); // Use the path to the JavaScript library file.
```

For Web browser use cases

For web browser use cases you need to make the JavaScript file available to your web page. You can put the file on your hosting environment or CDN or include as part of a larger JavaScript file together with other libraries you might use. Once you have made the file available, please ensure it is loaded in your web page. It can be loaded dynamically (or asynchronously) as long as you make sure the file is loaded before any of the code statements that use the library API are executed.

Immediately following loading of the file, please create your own reference to the library API to ensure you have access to the version of the API your implementation expects. For example:

```
<script src="path/to/your/hosted/comscore.js"></script>
<script>
    var analytics = ns_.analytics; // Create your own reference.
</script>
```

The reason this needs to happen immediately after loading the file is that web pages could contain first and third party code that load different versions of the library, which will contain the API specific to that version. By creating your own references you are ensuring your implementation will work as intended.

2.2 Set the correct `PlatformAPI`

First, you need to inform the library of the environment it is running in by calling the `analytics.PlatformApi.setPlatformAPI( environment )` method and indicate the environment your application is running in. The available `environment` argument values are shown below.

Runtime environment `PlatformAPI` values

Runtime environment	`PlatformAPI` value
PC and Mobile web browsers (**this is the default**)	`analytics.PlatformApi.PlatformApis.WebBrowser`
Sony Playstation(1) Trilithium runtime	`analytics.PlatformApi.PlatformApis.Trilithium`
Sony Playstation JSMAF runtime	`analytics.PlatformApi.PlatformApis.JSMAF`
Samsung Legacy Platform SmartTV TV sets and Bluray devices	`analytics.PlatformApi.PlatformApis.SmartTV`
LG TV Netcast runtime environment	`analytics.PlatformApi.PlatformApis.Netcast`
LG TV Cordova runtime environment(2)	`analytics.PlatformApi.PlatformApis.Cordova`
Apple TV runtime environment	`analytics.PlatformApi.PlatformApis.AppleTV`
Chromecast Custom Receiver	`analytics.PlatformApi.PlatformApis.Chromecast`
Xbox One WinJS environment	`analytics.PlatformApi.PlatformApis.Xbox`
webOS environment	`analytics.PlatformApi.PlatformApis.webOS`
Node.js environment	`analytics.PlatformApi.PlatformApis.nodejs`
Generic HTML5 environment (not intended for PC and Mobile web browsers)	`analytics.PlatformApi.PlatformApis.html5`
Unimplemented Skeleton API (for use with additional instructions from Comscore)	`analytics.PlatformApi.PlatformApis.Skeleton`

For example, to set the `PlatformAPI` to match a Chromecast environment:

```
10. analytics.PlatformApi.setPlatformAPI(analytics.PlatformApi.PlatformApis.Chromecast);
```

After setting the `PlatformAPI` you can configure the library by providing publisher-specific configuration settings and — where applicable — general configuration settings, prior to instructing the library to start collecting data.

2.3 Configure and start the library

It is strongly advised to configure and start the library when your application starts. For web browser use cases this would be when the web page containing your streaming media player(s) is loaded. For OTT application this would be when you initialize your own code in the application.

2.3.1 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
`setPublisherId`	`String`	**mandatory**	`1234567`
	Provide your Publisher ID value. The Publisher ID is often also referred to as the *Client ID* or *c2 value*.		

2.3.2 Apply Publisher Configuration Settings

The appropriate values for your publishers-specific configuration settings are assigned through a simple object during the construction of a `PublisherConfiguration` object instance, after which the `PublisherConfiguration` object instance is provided to the library API.

---

(1) Also supports other Sony devices with the same runtime.

(2) This should not be confused with mobile applications developed with Cordova, which have their own Comscore library solution.

```
var myPublisherConfig = new analytics.configuration.PublisherConfiguration({
    'publisherId': '1234567' // Provide your Publisher ID here.
});
analytics.configuration.addClient( myPublisherConfig );
```

2.3.3 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
`setApplicationName`	`String`	**optional**	`News Magazine`
	By default the library retrieves the application name from the targeted environment, for example the HTML document title in case of a web page. 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.		
`setApplicationVersion`	`String`	**optional**	`1.3.2-5f5e7a`
	By default the library does not retrieve an application version. You can provide a string with your application version.		
`enableFirstPartyCookie`	By default the library does not enable the first party cookie. You can enable it by using this setting. It does not take any arguments.		
`setUsagePropertiesAutoUpdateMode`	Object property	**optional**	`analytics.configuration.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:
	`analytics.configuration.UsagePropertiesAutoUpdateMode.FOREGROUND_ONLY`
	Update usage times only when the application is in the foreground. **This is the default mode.**
	`analytics.configuration.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 11* for more details about user experiences.
	`analytics.configuration.UsagePropertiesAutoUpdateMode.DISABLED`
	Do not update usage times. It is not advised to select this mode.		

`setUsagePropertiesAutoUpdateInterval`	integer `Number`	**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.		

2.3.4 Apply General Configuration Settings

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

```
16. analytics.configuration.enableFirstPartyCookie();
17. analytics.configuration.setUsagePropertiesAutoUpdateMode(
    analytics.configuration.UsagePropertiesAutoUpdateMode.FOREGROUND_AND_BACKGROUND );
```

2.3.5 Start the Comscore Library

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

```
21. analytics.start();
```

2.4 Indicate changes in the application life cycle

You **only** need to follow the instructions in this section if you are implementing in an OTT application. For web browser use cases you should **not** indicate changes in the application life cycle.

The library typically cannot automatically detect changes in the application life cycle of OTT applications. Please inform the library of background and foreground transitions of the application by calling the application life cycle notification methods.

Application life cycle API methods

Method name	Purpose	Expected location for calls
`notifyEnterForeground`	Informs the library of the application **moving to foreground**	To be called when your application determines it has moved to the foreground
`notifyExitForeground`	Informs the library of the application **leaving the foreground** (e.g., **moving to background**)	To be called when your application determines it has moved to the background

About calling the application life cycle notification methods at appropriate times…

If you have added the code statements for configuring and starting the library in the advised location then that ensures optimal operation of the library.

The library relies on these application life cycle notifications to properly collect analytics data on application usage. If you do not follow these instructions then the library will not be able to correctly measure application usage.

For example, to notify the library of a transition to foreground when your application is started:

```
analytics.notifyEnterForeground();
```

2.5 Indicate changes in the application user experience

You **only** need to follow the instructions in this section if you are implementing in an OTT application. For web browser use cases you should **not** implement calls to user experience notification methods.

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();
```

2.6 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 OTT application. For web browser use cases you should **not** indicate application section changes.

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 `Object` 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”:

```
// The user changes to the "news" section
analytics.notifyViewEvent({
    'ns_category': 'news'
});
```

2.7 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.

2.7.1 Using a Consent Management Platform

If you are using the library in streaming media players in web sites or web applications intended for PC and Mobile web browser, and you are using a Consent Management Platform (CMP) which implements iAB Global Privacy Platform (GPP) 1.1 or iAB Transparency and Consent Framework (TCF) version 2.2, then the 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 can access the CMP data as per the GPP 1.1 or TCF 2.2 technical specification.

If you are using the library in an iframe and the library is expected to use its TCF 2.2 CMP integration to automatically retrieve and collect user consent data, then you are expected to emulate either `__tcfapi()` inside the iframe. An example script that emulates the in-frame `__tcfapi()` call, is provided by the *TCF 2.2 specification (https://github.com/InteractiveAdvertisingBureau/GDPR-Transparency-and-Consent-Framework/blob/master/TCFv2/IAB%20Tech%20Lab%20-%20CMP%20API%20v2.md#is-there-a-sample-iframe-script-call-to-the-cmp-api)* . This script can be copied verbatim from the TCF 2.2 specification and implemented in your iframe.

The library currently does not support the use of its GPP 1.1 CMP integration when the library is used in an iframe.

2.7.2 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 `Object` 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. var myPublisherConfig = new analytics.configuration.PublisherConfiguration({
12.     'publisherId': '1234567' // Provide your Publisher ID here.
13.     'persistentLabels': {
14.         'cs_ucfr': '1'
15.     }
16. });
17. analytics.configuration.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.configuration.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 14*.

2.8 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.configuration.enableChildDirectedApplicationMode();
17. analytics.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 console. To enable this feature you can include a call to the `enableImplementationValidationMode` configuration method immediately before calling the `start` method:

```
16. analytics.configuration.enableImplementationValidationMode();
17. analytics.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 console. 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

Parameter	Description / Comments
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(3)
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(4)
ns_ap_dbt	Background active duration - in milliseconds - of previous sessions(5)

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(6), i.e., on an iOS device press the home button.
   - Wait for 30 seconds.
3. Bring the application to the foreground(7).
   - 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(8).

---

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

(4) 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.

(5) 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.

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

(7) 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.

(8) 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.

> 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.

With older library major versions, the solution for streaming media players in web sites or web applications intended for PC and Mobile web browsers only uses the Streaming Tag. You will first need to identify if that is the case in your implementation.

Determine Implementation Type

The implementation type can be determined from the presence of specific references and/or code statements to create object instances in the implementation.

Determine JavaScript library implementation type

Appearance / Characteristics	Version	Type
The implementation has code statements that use the `ns_.comScore` reference. In cases where tagging of video consumption is included, the implementation will *also* contain code statements that create Streaming Tag object instances.	6	**Application Tag**

For use in traditional web pages the implementation will not contain any references to `ns_.comScore`. Instead, the implementation only uses Streaming Tag object instance which are created using either
`var sa = new ns_.StreamingAnalytics({ publisherId: '1234567' }); // Will mention your Publisher ID.`
or
`var sa = new ns_.ReducedRequirementsStreamingAnalytics({ publisherId: '1234567' });`
		'stand-alone' Streaming Tag

Migrate from major version 6 to 7 with Application Tag

1. Replace the JavaScript file with the Comscore library code and create your own reference to the library API as a shorthand, as described in *Include the library in your application project on page 7*:

```
7.  var analytics = ns_.analytics; // Create your own reference.
```

2. Replace occurrences of `ns_.PlatformAPIs` with `analytics.PlatformAPIs`.
3. Locate existing library configuration code statements.
   Typically the following configuration code statements are present in an existing library implementation:

```
11. ns_.comScore.setAppContext( platformApi );
12. ns_.comScore.setCustomerC2( clientId );
13. ns_.comScore.setPublisherSecret( publisherSecret );
```

4. 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:

```
analytics.PlatformApi.setPlatformAPI( platformApi );
var myPublisherConfig = new analytics.configuration.PublisherConfiguration({
    'publisherId': clientId
});
analytics.configuration.addClient( myPublisherConfig );
analytics.start();
```

5. Replace occurrences of into and modify method calls to account for changes in naming[(9)]:

Existing call	Migrated call
`ns_.comScore.onEnterForeground()`	`analytics.notifyEnterForeground()`
`ns_.comScore.onExitForeground()`	`analytics.notifyExitForeground()`
`ns_.comScore.onUxActive()`	`analytics.notifyUxActive()`
`ns_.comScore.onUxInactive()`	`analytics.notifyUxInactive()`
`ns_.comScore.setLabel( name, value )`	`analytics.configuration.setPersistentLabel( name, value )`
`ns_.comScore.setLabels( labels )`	`analytics.configuration.addPersistentLabels( labels )`
`ns_.comScore.setAppName( name )`	`analytics.configuration.setApplicationName( name );`
`ns_.comScore.setAppVersion( version )`	`analytics.configuration.setApplicationVersion( name );`
`ns_.comScore.start()`	Please remove the call.

6. In cases where tagging of video consumption is included, update the Streaming Tag implementation as per migration instructions in the Streaming Tag's implementation guide.

Migrate from major version 6 to 7 with 'stand-alone' Streaming Tag

1. Replace the JavaScript file with the Comscore library code and create your own reference to the library API as a shorthand **immediately following loading of the file**, as described in *Include the library in your application project on page 7*:

```
<script src="path/to/your/hosted/comscore.js"></script>
<script>
    var analytics = ns_.analytics; // Create your own reference.
</script>
```

2. Locate existing Streaming Tag code statements, which typically look like either

```
var sa = new ns_.StreamingAnalytics({ 'publisherId': clientId });
```

or

```
var sa = new ns_.ReducedRequirementsStreamingAnalytics({ 'publisherId': clientId });
```

3. Make sure the library is configured and started **once**, prior to the use of any of the Streaming Tag code statements you have located. In the configuration you will copy over the `clientId` value:

```
analytics.PlatformApi.setPlatformAPI( analytics.PlatformApi.PlatformApis.WebBrowser );
var myPublisherConfig = new analytics.configuration.PublisherConfiguration({
    'publisherId': clientId
});
```

---

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

```
analytics.configuration.addClient( myPublisherConfig );
analytics.start();
```

4. Update the Streaming Tag implementation as per migration instructions in the Streaming Tag's implementation guide.

Appendix B: 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`	Mandatory(10)	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.**

(10) `cs_fpid` is required if you choose not to enable first-party cookie functionality. Please refer to *UDM 2.0 measurement with JavaScript library on page 3* for more details about the first-party cookie functionality.

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 `Object` 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:

```
var myPublisherConfig = new analytics.configuration.PublisherConfiguration({
    'publisherId': '1234567' // Provide your Publisher ID here.
    'persistentLabels': {
        'cs_fpid': '1605266069802_50777152',
        'cs_fpit': 'c',
        'cs_fpdm': '39642313001',
        'cs_fpdt': '01'
    }
});
analytics.configuration.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.

```
// Provide your Publisher ID in the getPublisherConfiguration() call.
analytics.configuration.getPublisherConfiguration( '1234567' ).setPersistentLabel( 'cs_fpdm', '39642313001' );
analytics.configuration.getPublisherConfiguration( '1234567' ).setPersistentLabel( 'cs_fpdt', '01' );
analytics.notifyHiddenEvent();
```

Appendix C: 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 14*.

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 10*.

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

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.