comscore

UNIFIED DIGITAL MEASUREMENT

JavaScript Library

Skeleton PlatformAPI Implementation Guide

document version: 2.1.1; released on February 4, 2020

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

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.

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.

The instructions in this document are intended to be used with **version 7.1.0 and subsequent 7.x.y releases** of the JavaScript library for implementation using JavaScript code. The library includes support for a number of environments through a specific *PlatformAPI* implementation for each environment. The *PlatformAPI* defines how the library retrieves appropriate platform-specific information and uses available transmission mechanisms, application-level storage and file system IO support where available.

For environments not supported by any of the *PlatformAPI* implementations the library offers a *Skeleton PlatformAPI*, which does not contain any implementation at all. Its methods are intended to be overridden using code for a platform-specific implementation of the targeted environment(s). This creates a solution for the targeted environment which allows data collection with the library on par (or as much as possible) with the already supported environments.

Calls to the library API will internally use the implemented *Skeleton PlatformAPI*. The implementation of the library API is discussed in a separate document.

About the integration described in this document...

The integration described in this document needs to be executed with involvement from Comscore for at least two reasons:

1. To ensure that the integration is using appropriate resources of the platform.
2. To ensure the collected data matches with Comscore reporting expectations.

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.

2 Implement the Skeleton PlatformAPI

It is expected for *Skeleton PlatformAPI* implementation to be applicable for any application running in the currently unsupported environment. In other words: the implementation of the *Skeleton PlatformAPI* methods is expected to use elements in the environment which are available to all applications running in that environment.

2.1 Override the Skeleton PlatformAPI with a specific interface

The first step is to inform the JavaScript library to use the *Skeleton PlatformAPI* and specify the interface to use:

```
analytics.PlatformApi.setPlatformApi(analytics.PlatformApi.PlatformApis.Skeleton, interfaceObject);
```

The `interfaceObject` is a standard `Object` with specific properties referencing relevant methods of the *Skeleton PlatformAPI* overridden with a suitable implementation for the targeted platform. The *PlatformAPI* methods that can be overridden are listed in *Platform-specific Integration on page 4*.

2.2 Fetch Updated Values During Runtime

After the library is configured and started, it will execute the *PlatformAPI* methods autonomously. Some of the implemented *Skeleton PlatformAPI* methods return platform-specific values which are static in nature, while others return values which are expected to change as the application is running. Aside from accessing parts of the environment to retrieve platform-specific values for data collection, the *Skeleton PlatformAPI* implementation caters for correct operation of the library — for example by storing any data that need to be persisted in application-level storage or arranging transmission of collected data — and the library expects the implemented *Skeleton PlatformAPI* methods **can be called at any time**.

To allow timely updates of the platform-specific values the *PlatformAPI* executes its `onDataFetch` method **first**, prior to executing any of its data retrieval methods or its support functionality methods which return values for data collection. The `onDataFetch` method receives two callback arguments, one of which is expected to be executed when the values are updated. This allows for synchronous and asynchronous retrieval of platform-specific values.

To illustrate, an implementation of the `onDataFetch` *PlatformAPI* method could be:

```
analytics.PlatformApi.setPlatformApi(analytics.PlatformApi.PlatformApis.Skeleton, {
    onDataFetch: function (onSuccessCallback, onErrorCallback) {
        // Execute a function with platform-specific code to retrieve up-to-date information.
        runPlatformSpecificCodeToRetrieveValues(onSuccessCallback, onErrorCallback);
    }
    // ⋮
    // Other overridden PlatformAPI methods, as needed.
});
```

The references to the provided callbacks should be retained so the appropriate callback can be called once the platform-specific code has updated the platform-specific values used by the *PlatformAPI*.

Once the platform-specific code has been executed successfully and the updated platform-specific values are available for the *PlatformAPI* to retrieve them, the `onSuccessCallback` — the first argument on the `onDataFetch` method call — needs to be called. When something occurs which prevents the *PlatformAPI* from retrieving updated values, the `onErrorCallback` — the second argument on the `onDataFetch` method call — needs to be called.

The execution of either callback will cause the *PlatformAPI* to continue its processing. If the `onErrorCallback` was executed then the *PlatformAPI* will not access its methods to retrieve updated platform data values. The next time the library requests the *PlatformAPI* to retrieve updated values the `onDataFetch` method will again be executed first to allow the values to be updated appropriately.

3 Platform-specific Integration

The table further below contains a list of all *PlatformAPI* properties which are candidates to be overridden in the *Skeleton PlatformAPI* implementation. The data collection methods are listed separate from the functionality methods. The *Skeleton PlatformAPI* implementation needs to adhere to the following:

- The origin and nature of unique device identifier source values needs to be disclosed to Comscore to ensure the values are appropriate for Comscore reporting.
- All implemented — i.e., overridden — data collection methods are expected to return `String` values.
- If a data collection method is not implemented then the *PlatformAPI* will use the listed default value. Please confirm with Comscore if this is acceptable.
- There could be cases where a data collection method can be implemented but needs to be able to return a default value. Please confirm with Comscore if this is acceptable.
- Typically, the support functionality methods will use one of the implementations already provided inside the *PlatformAPI*.

About the implementation of the *Skeleton PlatformAPI*...

Please make sure Comscore is consulted on **all** implemented *Skeleton PlatformAPI* methods. For some of the data collection methods Comscore needs to indicate the expected value. For most of the support functionality methods Comscore needs to be aware of how they will be implemented. In general, Comscore needs to ensure the collected data matches with the expectations of its reporting processes.

This is extremely important to ensure correct reporting of data collected from implemented applications.

The table lists the type of the properties on the *Skeleton PlatformAPI*. Please ensure the expected type is implemented. Some of the functionality properties are expected to be an object, or are expected to be instantiable. If there is a need to implement these, then please consult with Comscore to get a specification of their interface.

About the properties on the *Skeleton PlatformAPI*...

You might notice properties on the *Skeleton PlatformAPI* which are not mentioned in this document. **Please ensure any *Skeleton PlatformAPI* properties not mentioned in this document are not overridden, unless you have received explicit instructions from Comscore to do so.**

PlatformAPI properties

Item	Type	PlatformAPI method	Description	Examples / available value(s)	Default value / implementation
Data collection properties					
Platform OS name	function	`getPlatformName()`	A constant value with an identifier for the platform.	`"js"`	`"js"`
Platform OS version	function	`getPlatformVersion()`	The platform os version (could be the same as *Runtime version*).	`"4.4.2"`	`"unknown"`
Runtime name	function	`getRuntimeName()`	A constant value with an identifier of the runtime environment.	▪ `"tvos"` ▪ `"trilithium"` ▪ `"winjs"`	`"unknown"`
Runtime version	function	`getRuntimeVersion()`	The runtime environment version (could be the same as *Platform OS version*).	`"3.4.78-c"`	`"unknown"`
Cross-publisher unique device identifier	function	`getCrossPublisherUniqueDeviceId()`	A unique identifier which has the same value across all apps on the same device, even if the apps are from different publishers. Advertising identifiers are a good example of such an identifier.	`"5ec85f7c-27b6-4f08-9e0c-d6402a0b978f"`	`null`
Publisher-specific unique device identifier	function	`getPublisherSpecificUniqueDeviceId()`	A unique identifier which at least has the same value across all apps from the same publisher on the same device (could be the same value as *Cross-publisher unique identifier*).	`"ENU7N16116000821"`	`(+new Date()) + (~~(Math.random() * 1000));`
Publisher-specific unique device identifier descriptor	function	`getPublisherSpecificUniqueDeviceIdSuffix()`	A two-digit value that indicates the degrees to which the publisher-specific unique identifier value is persistent as well as its commonality.	`"31"`	`"72"`
Device model name	function	`getDeviceModel()`	An identifier for the device make/model.	▪ `"AppleTV 3,1"` ▪ `"ps4"` ▪ `"xbox one"`	`"unknown"`
Application package / bundle	function	`getPackageName()`	The application's package or bundle identifier.	`"com.comscore.NewsReader"`	`null`
Application name	function	`getApplicationName()`	The name of the application, typically the name with which the application is presented to users.	`"Comscore Newsreader"`	`"unknown"`
Application version	function	`getApplicationVersion()`	The version of the application.	`"1.12.6"`	`"unknown"`

Item	Type	PlatformAPI method	Description	Examples / available value(s)	Default value / implementation
Connection type	function	`getConnectionType()`	An identifier for the type of connection. Allowed values are: ▪ `emu` for emulator ▪ `wifi` for wireless ethernet ▪ `wwan` for cellular ▪ `eth` for wired ethernet ▪ `bt` for Bluetooth ▪ `unknown` when none of the above applies ▪ `none` if there is no networking connection	`"wifi"`	`"unknown"`
Device language	function	`getLanguage()`	An identifier for the language the application (or platform OS environment) is using, typically an IETF language tag.	▪ `"en"` ▪ `"en-us"` ▪ `"pt-br"`	`"unknown"`
Device display resolution	function	`getDisplayResolution()`	The display resolution of the visual output device (expected format: `<width>x<height>`).	▪ `"1440x2392"` ▪ `"1920x1080"`	`"0x0"`
Application resolution	function	`getApplicationResolution()`	The size of the application user interface (could be the same as *Device display resolution*).	`"1280x720"`	`"0x0"`
Device architecture	function	`getDeviceArchitecture()`	The architecture of the processor in the device.	▪ `"arm"` ▪ `"x64"` ▪ `"x86"` ▪ `"cell"`	`"unknown"`
Functionality properties					
Data retrieval handler	function	`onDataFetch(onSuccessCallback, onErrorCallback)`	The code of this implemented method needs to store the references to the callbacks and trigger the execution of code which retrieves the platform-specific values used by the other *PlatformAPI* methods. Once retrieval of the values is finished the appropriate callback needs to be called.	*See example in Fetch Updated Values During Runtime on page 3*	`onSuccessCallback();`
Live transmission	function	`httpGet(url, callback)`	A function for HTTP GET live transmission. The default `imgHttpGet` uses `Image` objects if available. Existing platform-specific *PlatformAPI* implementations use one of the listed examples for which Comscore can provide the code.	▪ `imgHttpGet` ▪ `ajaxHttpGet` ▪ `voidHttpGet`	`imgHttpGet`
Offline cache transmission	function	`httpPost(url, data, callback)`	A function for HTTP POST transmission of offline cache flushes. Existing platform-specific *PlatformAPI* implementations use one of the listed examples for which Comscore can provide the code.	▪ `ajaxHttpPost` ▪ `voidHttpPost`	`voidHttpPost`(1)

(1) This default implementation does not support offline cache flushes. The offline caching feature should not be enabled in the library implementation unless a suitable *Offline cache transmission* mechanism is configured on the *Skeleton PlatformAPI*.

Item	Type	PlatformAPI method	Description	Examples / available value(s)	Default value / implementation
Implementation validation logging	function	`standardOutputLog( line )`	A function to output logging lines generated by the implementation validation mode functionality. Typically, the lines are sent to standard output such as the console.	`null`	`null`
Storage	object definition	`Storage()`	Please note this property will be accessed by instantiating it. **Two levels of storage** are used through this property: 1. *Device-level storage* is used for the offline caching feature. 2. *Application-level storage* is used to store values which needs to be persisted across application usage sessions. The property is expected to reference an object definition, i.e., a function which will be instantiated and used by the *PlatformAPI*. The description of this property's interface goes beyond the scope of this documentation*Please consult with Comscore for details of the interface if the default implementation needs to be changed to support the targeted environment.*	`null`	`null`
Platform-specific setTimeout	function	`setTimeout(func, msec)`	A platform-specific setTimeout function where the first argument is a function and the second argument is a timeout value in milliseconds and the method returns a reference to the created timeout.	`return setTimeout(func, msec)`	`return setTimeout(func, msec)` *(2)*
Platform-specific clearTimeout	function	`clearTimeout(id)`	A platform-specific setTimeout function where the first argument is a reference to a previously created timeout.	`return setTimeout(func, msec);`	`return clearTimeout(id)` *(3)*

(2) The default implementation uses an existing `setTimeout` function.

(3) The default implementation uses an existing `clearTimeout` function.