GDPR Compliance: Capturing User Consent

Addendum for comScore Library Implementations

document version: 1.4.0; released on June 7, 2018

for further information, please contact:
comScore, Inc.
Tag Support
+1 866 276 6972

Contents

Introduction

The General Data Protection Regulation (GDPR) is a new European Union (EU) law that has global impact as it introduces new rules for the governance of personal data, such as a requirement to capture User Consent. This addendum document explains the steps to implement collection of User Consent for Audience Research purposes for publishers with an existing implementation of a comScore library.

About the instructions in this document...

This document contains instructions for multiple versions of the available comScore libraries. For each platform in use, please make sure to confirm the comScore library version — if applicable — in your application project source code and then execute the implementation steps to collect User Consent for that particular version.

Any questions about the instructions, the comScore libraries, code changes, the impact on the collected data or differences between the examples in this documentation and a publisher's environment can be addressed to comScore TagSupport and/or a comScore client account representative.

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 collect User Consent for Audience Research purposes 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 required values for the `cs_ucfr` user consent label are:

Label `cs_ucfr` values for collecting User Consent

Value	Interpretation	Usage
`0`	User has **not** given consent	The publisher uses this value to indicate the user has been asked for consent where the **user did not give consent** to collect data for Audience Research purposes
`1`	User has given consent	The publisher uses this value to indicate the user has been asked for consent where the **user has given consent** to collect data for Audience Research purposes

About including label `cs_ucfr` when not collecting User Consent or when the User Consent value is unknown…

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 comScore library is configured and started, then **do not populate** label `cs_ucfr` as part of the comScore library configuration. Instead, populate label `cs_ucfr` as a *Persistent Label* as soon as the User Consent value is known and subsequently notify the comScore library of a *Hidden Event* where needed as per the *Update the User Consent Value* instructions for each of the available comScore libraries.

1 Android

There are two major versions of comScore library for Android in circulation: the current **version 5** and the previous **version 3**.

1.1 Determine Library Version

The comScore library version can be determined from the configuration code statements required for all implementations.

Determine Android library version

Version	Appearance / Characteristics

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

Version 3	The configuration code statements look like:
	
	`comScore.setCustomerC2( "1234567" ); // Provide your Publisher ID here.`
	`comScore.setPublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ); // Provide your Publisher Secret here.`

Depending on individual publisher data collection needs other configuration settings might be included, but the appearance and structure of these required configuration settings will be as shown above.

1.2 Include User Consent with Version 5 Library

Regardless of exactly what configuration settings appear, for the purpose of collecting User Consent the `persistentLabels` configuration setting needs to be added (or modified) on the `PublisherConfiguration`. This configuration setting accepts persistent labels as a `HashMap` of which the key/value pairs represent label name/value pairs. For example, assuming the user has given consent, the aforementioned configuration code statements would be changed as follows:

```
HashMap<String,String> labels = new HashMap<String,String>();
labels.put("cs_ucfr", "1");
PublisherConfiguration myPublisherConfig = new PublisherConfiguration.Builder()
    .publisherId( "1234567" ) // Provide your Publisher ID here.
    .publisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ) // Provide your Publisher Secret here.
    .persistentLabels(labels)
    .build();
Analytics.getConfiguration().addClient( myPublisherConfig );
```

1.3 Include User Consent with Version 3 Library

Regardless of exactly what configuration settings appear, for the purpose of collecting User Consent the `cs_ucfr` label needs to be added to the persistent labels through the `setLabels` configuration API method. With this API method persistent labels are provided as a `HashMap` of which the key/value pairs represent label name/value pairs. For example, assuming the user has given consent, the aforementioned configuration code would be changed as follows:

```
11. HashMap<String,String> labels = new HashMap<String,String>();
12. labels.put("cs_ucfr", "1");
13. comScore.setCustomerC2( "1234567" ); // Provide your Publisher ID here.
14. comScore.setPublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ); // Provide your Publisher Secret here.
15. comScore.setLabels(labels);
```

1.4 Update the User Consent Value

It could happen that the User Consent value needs to be updated after the comScore library has been configured and started or that the User Consent value is not known at the time the comScore library is configured and started, which means label `cs_ucfr` was not populated in the comScore library configuration at that time.

In these cases label `cs_ucfr` must be populated as a *Persistent Label* with the appropriate value after the comScore library has been started and the comScore library must be notified of a *Hidden Event* so comScore can collect the updated User Consent value.

With Version 5 Library

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

With Version 3 Library

```
41. comScore.setLabel("cs_ucfr", consentValue);
42. comScore.hidden();
```

2 iOS, tvOS and watchOS

There are two major versions of comScore libraries for iOS, tvOS and watchOS in circulation: the current **version 5** and the previous **version 3**.

2.1 Determine Library Version

The comScore library version can be determined from the reference to the singleton object used in code statements in the implementation.

Determine iOS, tvOS or watchOS library version

Version	Appearance / Characteristics
Version 5	The code uses references to `SCORAnalytics`.
Version 3	The code uses references to `CSComScore`.

2.2 Locate Library Configuration Code Statements

To collect User Consent the comScore library configuration code statements need to be changed. The following table shows the configuration code statements required for all implementations, which can help to locate the configuration code statement in the application project source code.

Locate iOS, tvOS or watchOS library configuration code statements

Version	Appearance / Characteristics

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

Version	Appearance / Characteristics

Version 3	The configuration code statements look like:
	
	For Projects using Objective-C
	
	`[CSComScore setCustomerC2:@"1234567"]; // Provide your Publisher ID here.`
	`[CSComScore setPublisherSecret:@"7b94840eb66b17e61c3d2f909c3a1163"]; // Provide your Publisher Secret here.`
	
	For Projects using Swift
	
	`CSComScore.customerC2 = "1234567" // Provide your Publisher ID here.`
	`CSComScore.publisherSecret = "7b94840eb66b17e61c3d2f909c3a1163" // Provide your Publisher Secret here.`

Depending on individual publisher data collection needs other configuration settings might be included, but the appearance and structure of these required configuration settings will be as shown above.

2.3 Include User Consent with Version 5 Library

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 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.publisherSecret = @"7b94840eb66b17e61c3d2f909c3a1163"; // Provide your Publisher Secret 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?.publisherSecret = "7b94840eb66b17e61c3d2f909c3a1163" // Provide your Publisher Secret here.
    builder?.persistentLabels = [ "cs_ucfr": "1" ]
})
SCORAnalytics.configuration().addClient(with:myPublisherConfig)
```

2.4 Include User Consent with Version 3 Library

Regardless of exactly what configuration settings appear, for the purpose of collecting User Consent the `cs_ucfr` label needs to be added to the persistent labels through the `setLabels` configuration API method. With this API method persistent labels are provided 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

```
[CSComScore setCustomerC2:@"1234567"]; // Provide your Publisher ID here.
[CSComScore setPublisherSecret:@"7b94840eb66b17e61c3d2f909c3a1163"]; // Provide your Publisher Secret here.
[CSComScore setLabels:@{ @"cs_ucfr": @"1" }];
```

For Projects using Swift

```
CSComScore.customerC2 = "1234567" // Provide your Publisher ID here.
CSComScore.publisherSecret = "7b94840eb66b17e61c3d2f909c3a1163" // Provide your Publisher Secret here.
CSComScore.labels = [ "cs_ucfr": "1" ]
```

2.5 Update the User Consent Value

It could happen that the User Consent value needs to be updated after the comScore library has been configured and started or that the User Consent value is not known at the time the comScore library is configured and started, which means label `cs_ucfr` was not populated in the comScore library configuration at that time.

In these cases label `cs_ucfr` must be populated as a *Persistent Label* with the appropriate value after the comScore library has been started and the comScore library must be notified of a *Hidden Event* so comScore can collect the updated User Consent value.

With Version 5 Library

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 publisherConfigurationWithPublisherId() call.
SCORAnalytics.configuration().publisherConfiguration(withPublisherId:"1234567").setPersistentLabelWithName("cs_ucfr",
value: consentValue)
SCORAnalytics.notifyHiddenEvent()
```

With Version 3 Library

For Projects using Objective-C

```
[CSComScore setLabel:@"cs_ucfr" value:consentValue];
[CSComScore hidden];
```

For Projects using Swift

```
CSComScore.setLabel("cs_ucfr", consentValue)
CSComScore.hidden()
```

3 JavaScript (web or OTT)

For applications developed with JavaScript source code as well as for media players used in traditional web pages — those aimed at delivery to regular web browsers — comScore offers different solutions. Each of these solutions covers a specific type of implementation, which needs to be identified first from the code statements that appear in the implementation.

3.1 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

Type	Appearance / Characteristics
Application Tag	The Application Tag is not used for implementations in traditional web pages. Implementations of this type will have 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. Typically these Streaming Tag object instances are created *without providing any arguments for their instantiation*.

'stand-alone' Streaming Tag	For use in traditional web pages the Application is not used, meaning the implementation *does not contain any references to `ns_.comScore`*. Instead, the implementation *only uses Streaming Tag object instances* which are created using any of the following example code statements (`1234567` is an example value for the Publisher ID):
	
	`var streamingAnalytics = new ns_.StreamingAnalytics({ publisherId: '1234567' });`
	
	`var streamSense = new ns_.StreamSense({}, 'http://b.scorecardresearch.com/p?c1=2&c2=1234567');`

'stand-alone' Reduced Requirements Streaming Tag	For use in traditional web pages the Application is not used, meaning the implementation *does not contain any references to `ns_.comScore`*. Instead, the implementation *only uses Reduced Requirements Streaming Tag object instances* which are created using any of the following example code statements:
	
	`var streamingAnalytics = new ns_.ReducedRequirementsStreamingAnalytics({ publisherId: '1234567' });`
	
	`var myStreamingTag = new ns_.StreamingTag({ customerC2: '1234567' });`

3.2 Include User Consent with Application Tag

To collect User Consent the comScore library configuration code statements need to be changed. For most publishers, the configuration code statements required for all implementations will look as follows in the application project source code:

```
11. ns_.comScore.setCustomerC2( "1234567" ); // Provide your Publisher ID here.
12. ns_.comScore.setPublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ); // Provide your Publisher Secret here.
```

Depending on individual publisher data collection needs other configuration settings might be included, but the appearance and structure of these required configuration settings will be as shown above.

Regardless of exactly what configuration settings appear, for the purpose of collecting User Consent the `cs_ucfr` label needs to be added to the persistent labels through the `setLabels` configuration API method. With this API method persistent labels are provided as an `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:

```
ns_.comScore.setCustomerC2( "1234567" ); // Provide your Publisher ID here.
ns_.comScore.setPublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ); // Provide your Publisher Secret here.
ns_.comScore.setLabels({ "cs_ucfr": "1" });
```

In cases where tagging of video consumption is included, the code statements using the Streaming Tag do not require any changes.

3.3 Include User Consent with 'stand-alone' Streaming Tag

Regardless of exactly what Streaming Tag object instance is created, for the purpose of collecting User Consent the `cs_ucfr` label needs to be added to the persistent labels through the `setLabels` API method. With this API method persistent labels are provided as an `Object` of which the key/value pairs represent label name/value pairs. For example, assuming the user has given consent, the API method would be called on an instance as follows:

```
streamingAnalytics.setLabels({ "cs_ucfr": "1" });
```

3.4 Include User Consent with 'stand-alone' Reduced Requirements Streaming Tag

The Reduced Requirements Streaming Tag does not offer a way to set *Persistent Labels*. For the purpose of collecting User Consent, the `cs_ucfr` label needs to be added to the metadata `Object` provided as argument in the API methods on the Reduced Requirements Streaming Tag object instances.

With JavaScript library version **below 4.1411.18** the metadata `Object` can only be provided as an argument to the `playContentPart` notification method, as the `playAdvertisement` notification method does not accept any arguments. For JavaScript library versions **4.1411.18 and higher** the metadata `Object` can be provided as an argument to all playback notification methods (i.e., `playVideoContentPart`, `playAudioContentPart`, `playVideoAdvertisement` and `playAudioAdvertisement` all accept a metadata argument).

For example, assuming the user has given consent, the statements to set the metadata and provide it to the `playVideoContentPart` notification method could look like this:

```
var metadata = {
    "ns_st_pu": "ABC",
    "ns_st_pr": "Modern Family",
    "ns_st_ep": "Rash Decisions",
    "ns_st_sn": "1",
    "ns_st_en": "2",
    "ns_st_st": "Hulu",
    "ns_st_ge": "Comedy",
    "ns_st_ce": "1",
    "cs_ucfr": "1"
}
myStreamingTag.playVideoContentPart(metadata,
ns_.ReducedRequirementsStreamingAnalytics.ContentType.LongFormOnDemand);
```

3.5 Update the User Consent Value

It could happen that the User Consent value needs to be updated after the **Application Tag** or **'stand-alone' Streaming Tag** has been configured, or that the User Consent value is not known at the time the comScore library was configured and started, which means label `cs_ucfr` was not populated in the comScore library configuration at that time.

About updating the User Consent value with 'stand-alone' Reduced Requirements Streaming Tag…

The mechanism for populating label `cs_ucfr` with the **'stand-alone' Reduced Requirements Streaming Tag** already uses the updated User Consent value and does not require any extra code statements to update the value.

With Application Tag

In these cases label `cs_ucfr` must be populated as a *Persistent Label* with the appropriate value after the comScore library has been started and the comScore library must be notified of a *Hidden Event* so comScore can collect the updated User Consent value.

```
41. ns_.comScore.setLabel("cs_ucfr", consentValue);
42. ns_.comScore.hidden();
```

With 'stand-alone' Streaming Tag

In these cases label `cs_ucfr` must be populated as a *Persistent Label* with the appropriate value at a later time. The updated value will immediately be used by the 'stand-alone' Streaming Tag.

```
streamingAnalytics.setLabel("cs_ucfr", consentValue);
```

4 ActionScript3/AIR

To collect User Consent with ActionScript3 and/or AIR applications the comScore library configuration code statements need to be changed. For most publishers, the configuration code statements required for all implementations will look as follows in the application project source code:

```
11. comScore.setCustomerC2( "1234567" ); // Provide your Publisher ID here.
12. comScore.setPublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ); // Provide your Publisher Secret here.
```

Depending on individual publisher data collection needs other configuration settings might be included, but the appearance and structure of these required configuration settings will be as shown above.

Regardless of exactly what configuration settings appear, for the purpose of collecting User Consent the `cs_ucfr` label needs to be added to the persistent labels through the `setLabels` configuration API method. With this API method persistent labels are provided as an `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 labels = new Object();
12. labels["cs_ucfr"] = "1";
13. comScore.setCustomerC2( "1234567" ); // Provide your Publisher ID here.
14. comScore.setPublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ); // Provide your Publisher Secret here.
15. comScore.setLabels(labels);
```

4.1 Update the User Consent Value

It could happen that the User Consent value needs to be updated after the comScore library has been configured and started or that the User Consent value is not known at the time the comScore library is configured and started, which means label `cs_ucfr` was not populated in the comScore library configuration at that time.

In these cases label `cs_ucfr` must be populated as a *Persistent Label* with the appropriate value after the comScore library has been started and the comScore library must be notified of a *Hidden Event* so comScore can collect the updated User Consent value.

```
41. comScore.setLabel("cs_ucfr", consentValue);
42. comScore.hidden();
```

5 Roku SceneGraph and BrightScript

Roku BrightScript applications can be developed in SceneGraph or with 'regular' BrightScript. To collect User Consent the comScore library configuration code statements need to be changed. For most publishers, the configuration code statements required for all implementations will look as follows in the application project source code:

For SceneGraph projects

```
11. cs.SetCustomerC2( "1234567" ) // Provide your Publisher ID here.
12. cs.SetPublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ) // Provide your Publisher Secret here.
```

For 'regular' BrightScript projects

```
11. CSComScore().SetCustomerC2( "1234567" ) // Provide your Publisher ID here.
12. CSComScore().SetPublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ) // Provide your Publisher Secret here.
```

Depending on individual publisher data collection needs other configuration settings might be included, but the appearance and structure of these required configuration settings will be as shown above.

Regardless of exactly what configuration settings appear, for the purpose of collecting User Consent the `cs_ucfr` label needs to be added to the persistent labels through the `SetLabels` configuration API method. With this API method persistent labels are provided 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 SceneGraph projects

```
11. cs.SetCustomerC2( "1234567" ) // Provide your Publisher ID here.
12. cs.SetPublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ) // Provide your Publisher Secret here.
13. cs.SetLabels({ "cs_ucfr": "1" })
```

For 'regular' BrightScript projects

```
11. CSComScore().SetCustomerC2( "1234567" ) // Provide your Publisher ID here.
12. CSComScore().SetPublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ) // Provide your Publisher Secret here.
13. CSComScore().SetLabels({ "cs_ucfr": "1" })
```

5.1 Update the User Consent Value

It could happen that the User Consent value needs to be updated after the comScore library has been configured and started or that the User Consent value is not known at the time the comScore library is configured and started, which means label `cs_ucfr` was not populated in the comScore library configuration at that time.

In these cases label `cs_ucfr` must be populated as a *Persistent Label* with the appropriate value after the comScore library has been started and the comScore library must be notified of a *Hidden Event* so comScore can collect the updated User Consent value.

For SceneGraph projects

```
41. cs.SetLabel("cs_ucfr", consentValue)
42. cs.Hidden()
```

For 'regular' BrightScript projects

```
41.  CSComScore().SetLabel("cs_ucfr", consentValue)
42.  CSComScore().Hidden()
```

6 Roku C++

To collect User Consent with Roku C++ applications the comScore library configuration code statements need to be changed. For most publishers, the configuration code statements required for all implementations will look as follows in the application project source code:

```
std::shared_ptr<PublisherConfiguration> myPublisherConfig = PublisherConfiguration::Builder()
    .publisherId( "1234567" ) // Provide your Publisher ID.
    .publisherSecret( "9c455c81a801d3832a2cd281843dff30" ) // Provide your Publisher Secret.
    .build();
Analytics::getConfiguration()->addClient( myPublisherConfig );
```

Depending on individual publisher data collection needs other configuration settings might be included, but the appearance and structure of these required configuration settings will be as shown above.

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 `ComScore::StringPairArray` 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:

```
StringPairArray labels;
labels.set("cs_ucfr", "1");
std::shared_ptr<PublisherConfiguration> myPublisherConfig = PublisherConfiguration::Builder()
    .publisherId( "1234567" ) // Provide your Publisher ID.
    .publisherSecret( "9c455c81a801d3832a2cd281843dff30" ) // Provide your Publisher Secret.
    .persistentLabels( labels )
    .build();
Analytics::getConfiguration()->addClient( myPublisherConfig );
```

6.1 Update the User Consent Value

It could happen that the User Consent value needs to be updated after the comScore library has been configured and started or that the User Consent value is not known at the time the comScore library is configured and started, which means label `cs_ucfr` was not populated in the comScore library configuration at that time.

In these cases label `cs_ucfr` must be populated as a *Persistent Label* with the appropriate value after the comScore library has been started and the comScore library must be notified of a *Hidden Event* so comScore can collect the updated User Consent value.

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

7 Cordova

For Cordova applications the comScore Cordova plugin will be installed into the Cordova application development environment and instrumented through JavaScript source code.

To collect User Consent the comScore library configuration code statements need to be changed. For most publishers, the configuration code statements required for all implementations will look as follows in the application project source code:

```
11. ns_.comScore.setCustomerC2( "1234567" ); // Provide your Publisher ID here.
12. ns_.comScore.setPublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ); // Provide your Publisher Secret here.
```

Depending on individual publisher data collection needs other configuration settings might be included, but the appearance and structure of these required configuration settings will be as shown above.

Regardless of exactly what configuration settings appear, for the purpose of collecting User Consent the `cs_ucfr` label needs to be added to the persistent labels through the `setLabels` configuration API method. With this API method persistent labels are provided as an `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. ns_.comScore.setCustomerC2( "1234567" ); // Provide your Publisher ID here.
12. ns_.comScore.setPublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ); // Provide your Publisher Secret here.
13. ns_.comScore.setLabels({ "cs_ucfr": "1" });
```

7.1 Update the User Consent Value

It could happen that the User Consent value needs to be updated after the comScore library has been configured and started or that the User Consent value is not known at the time the comScore library is configured and started, which means label `cs_ucfr` was not populated in the comScore library configuration at that time.

In these cases label `cs_ucfr` must be populated as a *Persistent Label* with the appropriate value after the comScore library has been started and the comScore library must be notified of a *Hidden Event* so comScore can collect the updated User Consent value.

```
41. ns_.comScore.setLabel("cs_ucfr", consentValue);
42. ns_.comScore.hidden();
```

8 Unity

To collect User Consent with Unity applications the comScore library configuration code statements need to be changed. For most publishers, the configuration code statements required for all implementations will look as follows in the application project source code:

```
11. PublisherConfiguration myPublisherConfig = PublisherConfiguration.Builder()
12.     .PublisherId( "1234567" ) // Provide your Publisher ID here.
13.     .PublisherSecret( "9c455c81a801d3832a2cd281843dff30" ) // Provide your Publisher Secret here.
14.     .Build();
15. Analytics.Configuration.AddClient( myPublisherConfig );
```

Depending on individual publisher data collection needs other configuration settings might be included, but the appearance and structure of these required configuration settings will be as shown above.

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 `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:

```
11. PublisherConfiguration myPublisherConfig = PublisherConfiguration.Builder()
12.     .PublisherId( "1234567" ) // Provide your Publisher ID here.
13.     .PublisherSecret( "9c455c81a801d3832a2cd281843dff30" ) // Provide your Publisher Secret here.
14.     .PersistentLabels(new Dictionary<string, string> { { "cs_ucfr", "1" } })
15.     .Build();
16. Analytics.Configuration.AddClient( myPublisherConfig );
```

8.1 Update the User Consent Value

It could happen that the User Consent value needs to be updated after the comScore library has been configured and started or that the User Consent value is not known at the time the comScore library is configured and started, which means label `cs_ucfr` was not populated in the comScore library configuration at that time.

In these cases label `cs_ucfr` must be populated as a *Persistent Label* with the appropriate value after the comScore library has been started and the comScore library must be notified 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();
```

9 Xamarin

There are two major versions of comScore library for Xamarin in circulation: the current **version 2** and the previous **version 1**.

9.1 Determine Library Version

The comScore library version can be determined from the configuration code statements required for all implementations.

Determine Xamarin library version

Version	Appearance / Characteristics

Version 2	The configuration code statements look like:
	`var myPublisherConfig = PublisherConfiguration.Builder()`
	`.PublisherId( "1234567" ) // Provide your Publisher ID here.`
	`.PublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ) // Provide your Publisher Secret here.`
	`.Build();`
	`Analytics.Configuration.AddClient( myPublisherConfig );`

Version 1	The configuration code statements look like:
	`comScore.Analytics.Instance.CustomerC2( "1234567" ); // Provide your Publisher ID here.`
	`comScore.Analytics.Instance.PublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ); // Provide your Publisher Secret here.`

Depending on individual publisher data collection needs other configuration settings might be included, but the appearance and structure of these required configuration settings will be as shown above.

9.2 Include User Consent with Version 2 Library

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 `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:

```
var myPublisherConfig = PublisherConfiguration.Builder()
    .PublisherId( "1234567" ) // Provide your Publisher ID here.
    .PublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ) // Provide your Publisher Secret here.
    .PersistentLabels(new Dictionary<string, string> { { "cs_ucfr", "1" } })
    .Build();
Analytics.Configuration.AddClient( myPublisherConfig );
```

9.3 Include User Consent with Version 1 Library

Regardless of exactly what configuration settings appear, for the purpose of collecting User Consent the `cs_ucfr` label needs to be added to the persistent labels through the `SetLabels` configuration API method. With this API method persistent labels are provided 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 would be changed as follows:

```
11.  comScore.Analytics.Instance.CustomerC2( "1234567" ); // Provide your Publisher ID here.
12.  comScore.Analytics.Instance.PublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ); // Provide your Publisher Secret
     here.
13.  comScore.Analytics.Instance.SetLabels(new Dictionary<string, string> { { "cs_ucfr", "1" } });
```

9.4 Update the User Consent Value

It could happen that the User Consent value needs to be updated after the comScore library has been configured and started or that the User Consent value is not known at the time the comScore library is configured and started, which means label `cs_ucfr` was not populated in the comScore library configuration at that time.

In these cases label `cs_ucfr` must be populated as a *Persistent Label* with the appropriate value after the comScore library has been started and the comScore library must be notified of a *Hidden Event* so comScore can collect the updated User Consent value.

With Version 2 Library

```
41.  // Provide your Publisher ID in the GetPublisherConfiguration() call.
42.  Analytics.Configuration.GetPublisherConfiguration("1234567").SetPersistentLabel("cs_ucfr", consentValue);
43.  Analytics.NotifyHiddenEvent();
```

With Version 1 Library

```
41.  comScore.Analytics.Instance.SetLabel("cs_ucfr", consentValue);
42.  comScore.Analytics.Instance.Hidden();
```

10 Windows, Windows Phone and Xbox

To collect User Consent with Windows applications — including applications for Windows Phone, Mobile, Xbox and Silverlight — the comScore library configuration code statements need to be changed. For most publishers, the configuration code statements required for all implementations will look as follows in the application project source code:

```
11. comScore.Analytics.Instance.CustomerC2( "1234567" ); // Provide your Publisher ID here.
12. comScore.Analytics.Instance.PublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ); // Provide your Publisher Secret
    here.
```

Depending on individual publisher data collection needs other configuration settings might be included, but the appearance and structure of these required configuration settings will be as shown above.

Regardless of exactly what configuration settings appear, for the purpose of collecting User Consent the `cs_ucfr` label needs to be added to the persistent labels through the `SetLabels` configuration API method. With this API method persistent labels are provided 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:

```
11. comScore.Analytics.Instance.CustomerC2( "1234567" ); // Provide your Publisher ID here.
12. comScore.Analytics.Instance.PublisherSecret( "7b94840eb66b17e61c3d2f909c3a1163" ); // Provide your Publisher Secret
    here.
13. comScore.Analytics.Instance.SetLabels(new Dictionary<string, string> { { "cs_ucfr", "1" } });
```

10.1 Update the User Consent Value

It could happen that the User Consent value needs to be updated after the comScore library has been configured and started or that the User Consent value is not known at the time the comScore library is configured and started, which means label `cs_ucfr` was not populated in the comScore library configuration at that time.

In these cases label `cs_ucfr` must be populated as a *Persistent Label* with the appropriate value after the comScore library has been started and the comScore library must be notified of a *Hidden Event* so comScore can collect the updated User Consent value.

```
41. comScore.Analytics.Instance.SetLabel("cs_ucfr", consentValue);
42. comScore.Analytics.Instance.Hidden();
```

11 Confirm User Consent is Collected

After changing the implementation as instructed in this document the collected data should contain label `cs_ucfr` with its assigned value any time the comScore library transmits collected data and the User Consent value was known. The collected data is transmitted as HTTP requests, which can be inspected with an HTTP proxy such as Fiddler or Charles Proxy.

The collected data will be transmitted to the host `b.scorecardresearch.com`(1) where the URL query string parameters of the HTTP request should contain label `cs_ucfr` with its assigned value.

About confirming expected behavior...

As a good practice, it is advised that publishers confirm this expected behavior for each application that is tagged with the comScore library.

(1) For secure transmissions using HTTPS the hostname will be `sb.scorecardresearch.com`.