



UNIFIED DIGITAL MEASUREMENT

comScore Library Streaming Tag

JavaScript Implementation Guide

document version: 4.5.3; released on December 13, 2017

for further information, please contact:

comScore, Inc.

Tag Support

+1 866 276 6972

Contents

Implementation Quick Start	3
1 Introduction	4
1.1 Intended Use	4
1.2 Preparation	4
2 Implement the Streaming Tag	6
2.1 Include the Library	6
2.2 Create a <code>ns_.StreamingAnalytics</code> Instance	7
2.2.1 Stand-alone Implementation in a Web Page	7
2.2.2 Implementation in an Application	7
2.3 Tag Media Asset Changes	7
2.3.1 Create a new Playback Session	8
2.3.2 Set the Current Asset	8
2.3.3 Modify Asset Labels After Setting the Current Asset	8
2.4 Tag Playback State Changes	9
2.4.1 Arguments on the Notification Methods	9
2.4.2 Playback State Change Notification Methods	10
Appendix A: Labels	13
Standard Labels List	13
Classification Type	16
Appendix B: Content Asset Metadata Examples	18
Appendix C: Data Collection Model	19
Player Time Line Representation	20
Creating Playback Sessions	21
Changing Assets	21
Content Media Playback Tagging	22
Advertisement Media Playback Tagging	22
Example Playback Scenario	22
Appendix D: Migrating an Existing Implementation	25

Implementation Quick Start

This section contains abridged instructions for experienced users to implement the Streaming Tag using version 6.x of the comScore Streaming library.



Please do not use classes, methods or properties which do not appear in this documentation unless you have received explicit instructions to do so from your comScore account team or comScore Tag Support.

1. Confirm you have taken care of all the preparation items ([🔗 details on page 4](#)).
2. Include the relevant JavaScript libraries in your web page or application project ([🔗 details on page 6](#)).
 - For stand-alone implementations on a web page intended to be shown in web browsers on desktop and mobile devices: include the comScore streaming JavaScript library.
 - For implementations in OTT application projects: ensure the *latest* version of the comScore OTT JavaScript library is implemented and include the comScore streaming JavaScript library in your project.
 - For implementations in Cordova mobile application projects: ensure the *latest* version of the comScore Cordova plugin is implemented.
3. Create a `ns_.StreamingAnalytics` instance ([🔗 details on page 7](#)).
 - For stand-alone implementations your comScore Publisher ID needs to be provided for initialization:

```
11. | var streamingAnalytics = new ns_.StreamingAnalytics( { publisherId: '1234567' } );
```
 - For implementations in an OTT application or Cordova mobile application no initialization settings are needed:

```
11. | var streamingAnalytics = new ns_.StreamingAnalytics();
```
4. Tag media asset changes ([🔗 details on page 7](#)).
 - a. Create a new `Playback Session` each time the media player loads a new content media (with or without advertisements):

```
21. | streamingAnalytics.createPlaybackSession();
```
 - b. Set the current `Asset` in the each time the media player changes its media (content and/or advertisement):

```
31. | streamingAnalytics.getPlaybackSession().setAsset(assetMetadata);
```
5. Tag the playback state changes by calling the appropriate notification method for each relevant media player playback state change, providing arguments to the method calls as needed ([🔗 details on page 9](#)).

After successfully following these steps the comScore library will be collecting data for your streaming media playback. The library will send measurements whenever playback is started.

1 Introduction

The comScore Streaming Tag provides accurate and comprehensive streaming media analytics functionality. This enables comScore to receive measurement insights critical to answering questions about audio and video stream usage, including advertising messages.

The Streaming Tag is implemented next to, or into, your media player. Simply put, calls to the comScore library will be implemented to respond to activity in your media player, which then send data via HTTP requests to comScore's data collection servers.

This documentation is equally applicable to both video and audio streaming measurement. Please clarify with your comScore account team what type of media you should be implementing this tag onto (video and/or audio). Please do not implement this tag onto media types other than those you have been instructed to by your comScore account team.

This documentation applies to the implementation of comScore streaming library version 6.1.0.170130.

1.1 Intended Use

These instructions and the accompanying comScore library are intended to be used with media players running inside web pages intended to be shown in web browsers on desktop and mobile devices or with media players running inside OTT applications and Cordova mobile application and offer a JavaScript API. If your OTT application is developed in another programming language then please contact your comScore account team to ask for guidance.

1.2 Preparation

Please complete the following checklist before implementing the Streaming Tag in your application project:

1. If you plan on using the Streaming Tag in your OTT application or Cordova mobile application then make sure you have implemented the comScore Application Tag (either the OTT library or the comScore Cordova plugin) in your project.
2. If you are updating an existing Streaming Tag implementation, then please refer to [Appendix D: Migrating an Existing Implementation on page 25](#) to see if there are any relevant steps mentioned for your situation.
3. If you plan on using the Streaming Tag stand-alone, on a web page intended to be shown in web browsers on desktop and mobile devices then confirm your comScore *Publisher ID* - also known as the *Client ID* or *c2 value* - which is a number with at least 7 digits, provided by comScore.
4. Clarify with your comScore account team what type of media you should be implementing the Streaming Tag for (video and/or audio).
5. Familiarize yourself with the tagging instructions and the description of the Streaming Tag data collection model (refer to [Appendix C: Data Collection Model on page 19](#)) in this document.
6. Determine the media asset metadata values that need to be collected. Please refer to [Appendix A: Labels on page 13](#) for more information about media asset metadata collection.
7. Make sure you are using a media player that has an API which allows you to detect the player state and allows you to access details like the current playback position and relevant media asset metadata.

**About retrieving comScore Publisher details and comScore SDKs...**

1. Use a web browser to visit  [comScore Direct \(http://direct.comscore.com/clients/Video.aspx\)](http://direct.comscore.com/clients/Video.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 comScore account team. 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](#).

4. The Publisher ID is shown in the popup.

5. Please contact your comScore account team to get (updates to) comScore SDKs.

2 Implement the Streaming Tag

The implementation of the Streaming Tag involves the following steps:

1. Include the JavaScript library in your web page or application project where needed.
2. Create a `ns_.StreamingAnalytics` object to interact with.
3. Tag media asset changes by creating a new *Playback Session* and setting the current *Asset* by calling the appropriate methods on the `ns_.StreamingAnalytics` object.
4. Tag player playback state changes with calls to the appropriate notification methods on the `ns_.StreamingAnalytics` object.

All data is collected by specifying *Labels*, e.g., with classification values for reporting, in the implementation. Please refer to [Appendix A: Labels on page 13](#) for more information about labels.

As you work with the comScore library you might see classes, methods and properties which do not appear in this documentation. Those elements are typically exposed because that is necessary for the library to work in the way it was intended to or to support customized implementations for which you would receive separate instructions from your comScore account team or comScore Tag Support.

The use of those elements without instructions from comScore could severely impact the behaviour of the comScore library, the quality of the collected data or reporting capabilities.



Please do not use classes, methods or properties which do not appear in this documentation unless you have received explicit instructions to do so from your comScore account team or comScore Tag Support.

If you have any questions or concerns about the instructions in this document or about any classes, methods or properties which you find in the comScore library then please contact your comScore account team or comScore Tag Support.

2.1 Include the Library



About the instructions in this section...

The instructions in this section do not apply if you are implementing the Streaming Tag in a Cordova mobile application. The comScore Cordova plugin already includes the Streaming Tag library. No further steps are required to include the library.

To start using the Streaming Tag you will have to make sure the JavaScript file with the comScore streaming library is available in your HTML document. You can load this JavaScript file from the HTML document's `<head>` section, using:

```
<script src="comscore.streaming.min.js" type="text/javascript"></script>
```

This file contains JavaScript code with definitions of classes used by the Streaming Tag implementation. You can load the file dynamically (or asynchronously) as long as you make sure the file is loaded before any code statements that use the comScore streaming library are executed.

2.2 Create a `ns_.StreamingAnalytics` Instance

Depending on your implementation environment the instance of `ns_.StreamingAnalytics` needs to be created with or without a configuration object.

2.2.1 Stand-alone Implementation in a Web Page

If you are implementing the Streaming Tag stand-alone, on a web page intended to be shown in web browsers on desktop and mobile devices, then you need to create a new instance and initialize it with configuration settings which specify your comScore Publisher ID:

```
11. | var streamingAnalytics = new ns_.StreamingAnalytics( { publisherId: '1234567' } );
```

Note that the `publisherId` property name is case-sensitive.

2.2.2 Implementation in an Application

If you are implementing the Streaming Tag in an OTT application or Cordova application, then you need to also implement the Application Tag with the comScore OTT library or comScore Cordova plugin. The Streaming Tag will reuse the initialization settings from the Application Tag implementation. Create an instance without providing any configuration settings:

```
11. | var streamingAnalytics = new ns_.StreamingAnalytics();
```

2.3 Tag Media Asset Changes

Please refer to [Appendix C: Data Collection Model on page 19](#) for an in-depth discussion of the Streaming Tag data collection model. It explains the representation of the various elements of a streaming media player environment and the terminology used in Streaming Tag implementations. You will also find example describing when to use the API methods presented in these implementation instructions.

The Streaming Tag data collection model can be summarized as follows:

- The *Playback Session* represents the collection of the content media and its related advertisement media.
- Each discrete content media and individual advertisement media is represented by exactly one *Asset*.

2.3.1 Create a new Playback Session

Each time your media player is instructed to load different content for playback - or the first time it loads an advertisement related to that content - you should instruct the `ns_.StreamingAnalytics` object to create a new *Playback Session*, using the `createPlaybackSession` method on the `ns_.StreamingAnalytics` object:

```
31. | streamingAnalytics.createPlaybackSession();
```

Advertisements that are played in relation to content should be in the same *Playback Session* as their related content. When advertisements are involved you would typically change the current *Playback Session* after any post-rolls and before any pre-rolls so that the content and its related advertisements end up in the same *Playback Session*.

Please refer to [Appendix C: Data Collection Model on page 19](#) for examples of when creating a new *Playback Session* is and is not expected.

2.3.2 Set the Current Asset

As your media player changes between media assets - content or advertisements - you need to make this known to the `ns_.StreamingAnalytics` object by informing it about the *Asset* the media player has changed to, and provide the media metadata as *Asset Labels*.

To do this, use the `setAsset` method on the *PlaybackSession* object (accessible through the `ns_.StreamingAnalytics` object) and provide a *Object* as an argument of which the key/value pairs are set to the name/value pairs of the *Asset Labels*:

```
51. | streamingAnalytics.getPlaybackSession().setAsset(assetMetadata);
```

Please refer to [Appendix A: Labels on page 13](#) for more information about providing asset metadata labels, which labels are mandatory and optional to use and how to determine appropriate values.



About identifying *Segments* in the *Asset Labels*...

If your media player supports mid-roll ad breaks then please make sure to set the *Segment Number* label (`ns_st_pn`) to the appropriate value in the *Object* with *Asset Labels* prior to providing it on the `setAsset` call.

Please refer to [Appendix C: Data Collection Model on page 19](#) for examples of when changing the current *Asset* is and is not expected. There you will also find details about the representation of the media player *Time Line* and *Segments*.

The call to `setAsset` takes an optional second argument - `loop` - which is used to specify a boolean flag that indicates playback of the *entire Playback Session* is looping, i.e., (automatically) being repeated. This is used for advanced reporting scenarios. It is not required to provide a value for this second argument unless you have received specific instructions to do so from your comScore account team or comScore Tag Support.

2.3.3 Modify Asset Labels After Setting the Current Asset

At times it might be necessary to update *Asset Label* values after setting the current asset. For example if the media player has an updated value of the asset length. You can set, modify or unset⁽¹⁾ *Asset Labels* after you have set the current *Asset*:


```

61. // Set a single label
62. streamingAnalytics.getPlaybackSession().getAsset().setLabel("labelName", "labelValue");
63.
64. // Set any number of labels at once
65. streamingAnalytics.getPlaybackSession().getAsset().setLabels({
66.     labelName1: "labelValue1",
67.     labelName2: "labelValue2"
68. });

```

2.4 Tag Playback State Changes

After changing the current media asset the player will start going through the lifecycle of the asset - accompanied by playback state changes - as it buffers data, starts playback and performs (user-initiated) activities like seeking and continuing playback, pausing and resuming playback, re-buffering, etc. The lifecycle of the current media ends when the media player reaches the end of the media either naturally or through user intervention, or when the media player needs to change to other media.

For each of the playback state changes that have been identified as relevant to the Streaming Tag implementation there is a notification method on the `ns_.StreamingAnalytics` object that should be called.

Please note that calling the notification methods will cause the internal state of the `ns_.StreamingAnalytics` object to be updated - i.e., the first notification will start the media lifecycle - and transition between playback states. Depending on the transition the `ns_.StreamingAnalytics` object will determine if transmission is necessary, implying that calls to the notifications methods do not necessarily cause HTTP requests.



About creating a new *Playback Session* or changing the *Asset* during playback...

If the `createPlaybackSession` method is called during playback, then the `ns_.StreamingAnalytics` object will internally consider playback of all *Assets* of the previous *Playback Session* to be ended.

Likewise, if the `setAsset` method is called during playback, then the `ns_.StreamingAnalytics` object will internally consider playback of the previous *Asset* to be ended.

Calls to these methods should occur when playback has ended (or is about to end), ideally only after the `notifyEnd` playback state notification method has been called to indicate playback has ended.

2.4.1 Arguments on the Notification Methods

All notification methods on the `ns_.StreamingAnalytics` object accept the following two *optional* arguments:

Current Playback Position

This argument is expected to be a value in milliseconds which is either `0` or a positive *integer* value otherwise. The provided value represents the position the player is at - on the time line *relative to the loaded media asset* - when the corresponding playback state change occurs.

(1) Specifying `null` as the label value will unset the label.

For example: if the player pauses at 10 minutes into the content, then the notification of the playback state change would use a value of 10 minutes.

Event-specific Labels

This argument is expected to be a **Object** of which the key/value pairs are set to the name/value pairs which apply to the particular notification they are provided for. Your implementation should not specify any event-specific labels unless you have received explicit instructions to do so from your comScore account team or comScore Tag Support.



About providing position values...

Position values are always provided as integer values in *milliseconds*. For example: 10 minutes should be provided as **600000**.

The `ns_.StreamingAnalytics` object keeps an internal position value which progresses together with natural time from the last known position. It is typically only needed to supply a value for the optional position argument when indicating playback is *starting*, *resuming after pausing* or *continuing after seeking* to inform the `ns_.StreamingAnalytics` object about the *Current Playback Position*.

If the position argument is not provided in those cases then the `ns_.StreamingAnalytics` object will assume playback either *starts* from position **0**, *resumes after pausing* from the last known position or *continues after seeking* from the position where seeking took place⁽²⁾.

2.4.2 Playback State Change Notification Methods

The following table lists the relevant playback state changes and the notification method which should be called in each case. As described above, all arguments for these notification methods are *optional* unless indicated otherwise in the *Comments* column, in which case further instructions for the use of the argument(s) will be provided.

Playback state change notification methods on the `ns_.StreamingAnalytics` object

New playback state	Notification method	Comments
buffering started	<code>notifyBufferStart</code>	<p>Indicates the player has started buffering streaming data and is not currently playing.</p> <p>Expected to be called prior to the start of playback (tracks startup buffering time) and/or during playback (tracks re-buffering).</p> <p>Usually the call is expected to contain no arguments or just the position argument. If the position argument is provided then it should reflect the position where buffering started.</p>
buffering ended	<code>notifyBufferStop</code>	<p>Indicates the player has finished buffering streaming data.</p> <p>If buffering started prior to the start of playback then the <code>ns_.StreamingAnalytics</code> object will assume the player is now idle and waiting to start playback. If buffering started during playback (i.e., re-buffering) then the <code>ns_.StreamingAnalytics</code> object will assume playback resumes.</p> <p>Usually the call is expected to contain no arguments or just the position argument. If the position argument is provided then it should reflect the position where playback will <i>start</i>, <i>resume after pausing</i> or <i>continue after seeking</i>.</p>

(2) Please note that this means the internal position of the `ns_.StreamingAnalytics` object remains unchanged - implying that the seeking activity *did not involve a position change* - which is typically not to be expected when tagging seeking activity.

New playback state	Notification method	Comments
playback activated	<code>notifyPlay</code>	<p>Indicates playback has either <i>started</i>, <i>resumed after pausing</i> or <i>continued after seeking</i>.</p> <p>The call is expected to contain a value for the position argument so that the <code>ns_.StreamingAnalytics</code> object can update its internal position with an appropriate value. The position argument should reflect the position where playback <i>started</i>, <i>resumed after pausing</i> or <i>continued after seeking</i>.</p>
playback paused	<code>notifyPause</code>	<p>Indicates playback was paused and the media player is not currently playing.</p> <p>It is expected for this method to be called after playback has <i>started</i>, <i>resumed after pausing</i> or <i>continued after seeking</i>.</p> <p>Usually the call is expected to contain no arguments or just the position argument. If the position argument is provided then it should reflect the position where playback was paused.</p>
playback ended	<code>notifyEnd</code>	<p>Indicates the playback - or the media lifecycle - has ended.</p> <p>This methods is typically called in the following cases:</p> <ul style="list-style-type: none"> Playback naturally reached the end of the asset (content or advertisement). The user interacted with the media player either causing playback to end or during pausing, or to interrupt seeking or buffering (for example by using the 'stop playback' button). Playback of the current asset ends because the media player needs to change media - for example to load an advertisement for an ad break - even if the media player technically pauses one playback component to load and play the other media in another component. The media player encountered a fatal error during playback, pausing, seeking or buffering and playback (or other activity) cannot continue. <p>Usually this call is expected to contain no arguments or just the position argument. If the position argument is provided then it should reflect the position where playback ended (or the fatal error occurred).</p>
seeking started	<code>notifySeekStart</code>	<p>Indicates the media player has started seeking.</p> <p>Typically, it is expected for this method to be called only when seeking occurs during playback, i.e., not for seeking that occurs while playback is paused.</p> <p>Usually this call is expected to contain no arguments or just the position argument. If the position argument is provided then it should reflect the position when playback was interrupted as seeking started.</p> <p>For example: the media player seeks during playback from position 15 minutes to position 20 minutes. The position value for the <code>notifySeekStart</code> method call is expected to be <code>900000</code> (i.e., 15 minutes in milliseconds). After seeking ends and playback <i>continues after seeking</i> then the position value for the <code>notifyPlay</code> method call is expected to be <code>1200000</code> (i.e., 20 minutes in milliseconds).</p> <p>If the player is playing a live stream with DVR capabilities, then the call is expected to not contain a value for the position argument, as per the guidelines on page 19.</p>

For example, to notify the `ns_.StreamingAnalytics` object that playback is continuing after seeking from position 20 minutes:

```
71. | streamingAnalytics.notifyPlay(1200000);
```

It is possible to have multiple calls to a notification method appear throughout the implementation with your media player. Some media players are known to require calls to tag, for example, buffering and pausing in multiple locations (e.g., in response to multiple media player API events). Please refer to [Appendix C: Data Collection Model on page 19](#) for example sequences of playback state change notifications.

If you have followed the implementation instructions up to this point then the comScore library will be collecting data for your streaming media playback. The library will send measurements whenever playback is started.

Appendix A: Labels

All data is collected in name/value pairs called [Labels](#). Typically, labels are assigned in the implementation via [Objects](#). A number of standard labels are defined by comScore. It might be possible that your comScore account team or comScore Tag Support asks you to implement labels which are not listed in this table for example for custom reporting purposes.

Oftentimes, the label values you want to provide contain characters that cannot be passed in a URL (spaces, exclamation marks, commas, etc.). You do not have to URL encode or escape these characters. The comScore library will take care of URL encoding these characters.








Standard Labels List













The table below lists all standard labels allowed for use in your Streaming Tag implementation, with their intended purpose and expected values and the comScore products they are typically used for. If you have any questions about which values to populate or if any of this data is not available for your implementation to use, then please contact your comScore account team.

To illustrate the use of the content media metadata labels some examples with a selection of labels are provided in [Appendix B: Content Asset Metadata Examples on page 18](#). The examples show what kind of label values are expected for commonly used types of content media.

Standard labels available for the Streaming Tag implementation









Products Legend  = Video Metrix  = Cross Platform Product Suite



Label	Item	Description	Presence	Products	Example value(s)
Asset Labels					
For Assets which represent advertisements it is expected to provide the metadata of the content media which the advertisement is related to, such as its title and genre. Exceptions are, of course, those labels which provide information about the individual Asset , like the asset length.					
ns_st_ci	Unique Content ID	Provide your internal unique identifier for the content media Asset . Used in the report calculations logic to identify unique content assets. Provide value 0 if your media player does not use or have access to unique content media identification.	mandatory		13784
ns_st_cl	Asset Length	The length of the individual Asset , i.e., the available amount of media. Expects a value in milliseconds . If your media player or content metadata database reports media length values in seconds then please multiply the reported values by 1000. If the media length is unknown or cannot be determined then please provide value 0.	mandatory	 	1260000
ns_st_pu	Publisher Brand Name	Collect the consumer-facing brand name of the media publisher that owns the content.	mandatory	 	<ul style="list-style-type: none"> ABC ESPN CNN NFL
ns_st_pr	Program Title	Top level content title (i.e., the name of the overall program, show, or content series). Can be used with label Episode Title (ns_st_ep) to tag TV shows on program and episode level.	mandatory	 	<ul style="list-style-type: none"> Modern Family Harry Potter 7 Game 16: Eagles vs Patriots

Label	Item	Description	Presence	Products	Example value(s)
ns_st_tpr	Program ID	Top level content ID, which can be used for matching and grouping purposes (for example when the program title appears with multiple variations for the same program). <i>(This should not be confused with the Unique Content ID ns_st_ci which identifies an individual asset.)</i>	optional	 	<ul style="list-style-type: none"> 53617155
ns_st_ep	Episode Title	Sub level content title (i.e., the title of the specific episode). Can be used with label Program Title (ns_st_pr) to tag TV shows on program and episode level.	mandatory		<ul style="list-style-type: none"> Rash Decisions Season 2 Teaser
ns_st_tep	Episode ID	Sub level content ID, which can be used for matching and grouping purposes (for example when the episode title appears with multiple variations for the same episode of a specific program). <i>(This should not be confused with the Unique Content ID ns_st_ci which identifies an individual asset.)</i>	optional	 	<ul style="list-style-type: none"> 53617155
ns_st_sn	Episode Season Number	The season number for episodic content. It is recommended to use a value with 2 digits. Can be omitted or left blank for non-episodic content.	mandatory		05
ns_st_en	Episode Number	The episode number for episodic content. It is recommended to use a value with 2 digits ⁽³⁾ .	mandatory		17
ns_st_ge	Content Genre	Content genre description. Multiple values can be provided as a comma-separated string.	mandatory	 	<ul style="list-style-type: none"> Comedy Sports Fantasy, Drama
ns_st_ct	Classification Type	4-character identifier which distinguishes advertisement stream types from content stream types. Please refer to the complete description on page 16 for a list of available values and instructions to decide which value to provide.	mandatory		vc12
ns_st_ia	Advertisement Load Flag	<p>Indicates whether the streamed media carries the same advertisement load that was used during the TV airing. Use value 1 if the advertisement load is the same. Can be omitted otherwise.</p> <p>This flag helps comScore differentiate if the stream is carrying the same ad load as TV. Often digital video inventory is clubbed together with TV inventory and is served with the same ad load. The CPM⁽⁴⁾ for digital inventory with TV ad load is different from the CPM for any other ad load.</p> <p>If for any reason your backend or workflow requires all media metadata to have values for the same set of labels, then please make sure you use value 0 for any streamed media which did not carry the same advertisement load as during the TV airing.</p> <p>The flag is considered 'set' when the label is present with any non-empty, non-zero value.</p>	mandatory		1
ns_st_ddt	Digital Airdate	<p>The date on which the content was made available for streaming consumption. Expects a value in <code>yyyy-mm-dd</code> format.</p> <p>This airdate helps comScore establish monetization windows (live, day +1, day +3, ...) for any given episode or show. The monetization windows are used to calculate commercial and program ratings.</p>	mandatory		2011-08-30

(3) For episodic content with more than 99 episodes in a season it is recommended to use values with 3 digits, e.g., 017.

(4) CPM is the advertising cost per impression.

Label	Item	Description	Presence	Products	Example value(s)
<code>ns_st_tdt</code>	TV Airdate	<p>The date on which the content aired on TV. Expects a value in <code>yyyy-mm-dd</code> format.</p> <p>This airdate helps comScore establish monetization windows (live, day +1, day +3, ...) for any given episode or show. The monetization windows are used to calculate commercial and program ratings.</p>	mandatory		<code>2011-08-30</code>
<code>ns_st_st</code>	Station Title	The title of the station or channel for which content was recorded or where content is made available.	mandatory	 	<ul style="list-style-type: none"> ESPN3 BBC2
<code>c3</code> , <code>c4</code> , <code>c6</code>	Video Metrix Dictionary Classification	<p>These labels determine which entity the clip will credit to in the Video Metrix dictionary.</p> <p>These mandatory values are used to determine the entity the clip will credit to in the Video Metrix dictionary. They do not have specific pre-defined meanings. You should work with your comScore account team to establish what these labels' values should be, based on your desired dictionary goals.</p> <p>All three of these labels must always be passed. Unused labels must still be passed with the literal string value <code>*null</code>.</p>	mandatory		<code>*null</code>
Asset Labels Specifically for Content					
These labels are expected to only be supplied for Assets which represent content .					
<code>ns_st_pn</code>	Segment Number	<p>Indicates the current segment of the content media. Is expected to start with value <code>1</code> and to increment when the current Asset is changed back to the content after a mid-roll ad break.</p> <p>Please refer to Content Media Playback Tagging on page 22 for more information about working with segment numbers.</p>	mandatory		<code>1</code>
<code>ns_st_tp</code>	Total Number of Segments	<p>Indicates the total number of segments of the content media, which is equal to the number of mid-roll ad breaks + 1. Use value <code>1</code> if there are no mid-roll ad breaks for the content media.</p> <p>Use value <code>0</code> if the total number of segments of the content media cannot be determined as the current Asset is changed, like when mid-roll ad breaks are assigned dynamically) during content playback.</p>	optional		<code>3</code>
<code>ns_st_ce</code>	Complete Episode Flag	<p>Specifies the content media to be a full episode rather than an excerpt. Use value <code>1</code> when the content media is a full episode. Can be omitted otherwise.</p> <p>This flag helps us identify if the streaming content is episodic, long-form, or premium in nature. It also indicates whether the show or episode will be explicitly broken out in the dictionary.</p> <p>If for any reason your backend or workflow requires all media metadata to have values for the same set of labels, then please make sure you use value <code>0</code> for any media (content and/or advertisements) which is not a full content episode.</p> <p>The flag is considered 'set' when the label is present with any non-empty, non-zero value.</p>	mandatory	 	<code>1</code>

Label	Item	Description	Presence	Products	Example value(s)
Asset Labels Specifically for Advertisements					
These labels are expected to only be supplied for <i>Assets</i> which represent advertisements .					
<i>ns_st_ad</i>	Advertisement Flag	<p>Specifies the asset to be an advertisement. Provide one of the following values to specify the type type of the ad break for which the advertisement is shown:</p> <ul style="list-style-type: none"> <i>pre-roll</i> <i>mid-roll</i> <i>post-roll</i> <p>Use value 1 when the type of ad break cannot be determined. If for any reason your backend or workflow requires all media metadata to have values for the same set of labels, then please make sure you use value 0 for any content media.</p> <p>The flag is considered 'set' when the label is present with any non-empty, non-zero value.</p>	mandatory		1
Asset Labels Specifically for Live Streams					
These labels are expected to only be supplied for <i>Assets</i> which represent live streamed media (both content and advertisement). Live streamed media are provided to viewers as multicast, simulcast, or unicast streams where the viewer typically cannot control the playback position.					
<i>ns_st_ft</i>	Feed Type	<p>Specified the type of feed provided on the live stream. Intended to be used on live streams using the same feed as was used for the live TV broadcast.</p> <p>Currently only used for implementations in the US where it can have the following values:</p> <ul style="list-style-type: none"> <i>EASTHD</i> <i>WESTHD</i> <i>EASTSD</i> <i>WESTHD</i> 	optional		

Classification Type

The *Classification Type* label (*ns_st_ct*) is critical for enabling comScore to distinguish advertisement streams from content streams. Please use the table below as a guide for determining which label value to provide.

Classification Types

Description		Label value	
		video+audio	audio-only
Content^A			
PREMIUM Content with strong brand equity or brand recognition. Premium content is usually created or produced by media and entertainment companies using professional-grade equipment, talent, and production crews that hold or maintain the rights for distribution and syndication.	Short Form ^B Video On Demand	vc11	ac11
	Long Form ^B Video On Demand	vc12	ac12
	Live Streaming	vc13	ac13
USER-GENERATED Content with little-to-no brand equity or brand recognition. User-generated content (UGC) has minimal production value, and is uploaded to the Internet by non-media professionals.	Short Form ^B Video On Demand	vc21	ac21
	Long Form ^B Video On Demand	vc22	ac22
	Live Streaming	vc23	ac23
BUMPERS^C Bumpers - also known as billboards or slates - are static promotional items which usually run before content and usually last less than 5 seconds.		vc99	ac99

Description		Label value	
		video+audio	audio-only
OTHER Used if none of the above categories apply.		vc00	vc00
Advertisement^A			
LINEAR - VIDEO ON DEMAND Linear advertisements delivered into a media player and presented before, in the middle of, or after video content is consumed by the user. The advertisement completely takes over the full view of the media player.	Linear Pre-Roll	va11	aa11
	Linear Mid-Roll	va12	aa12
	Linear Post-Roll	va13	aa13
LINEAR - LIVE Linear advertisements delivered before, in the middle of, or after a live stream of content. The advertisement completely takes over the full view of the media player.		va21	aa21
BRANDED ENTERTAINMENT Media that a user may intentionally view (like content), or it may be served to a user during an ad break (like an advertisement).	During Linear Pre-Roll	va31	aa31
	During Linear Mid-Roll	va32	aa32
	During Linear Post-Roll	va33	aa33
	As Content	va34	aa34
	During Live Streaming	va35	aa35
OTHER Used if none of the above categories apply.		va00	aa00

^A In cases where defining a stream as advertisement or content is ambiguous, streams should be classified as content if they can be monetized. A stream can be monetized if it could (or did) have advertisements run against it. Conversely, a stream should be classified as advertisement if it is not in a position to have advertisements run against it due to the promotional nature of its subject matter.

^B Long form video on demand is differentiated from short form video on demand in that long form content always has a content arc with a beginning, middle, and end which in its entirety typically lasts longer than 10 minutes.

^C Bumpers (billboards, slates) do not have to be tagged. With some implementations tagging of bumpers cannot be avoided. In those cases these values can be used to identify streams as bumpers.

The following types of video streams should **not** be tagged using the Streaming Tag unless otherwise directed by your comScore account team.

- **In-banner video advertisements**

In-banner video advertisements are the same as standard image/flash banner advertisements prevalent on the Internet, except they have a streamed video associated within them, (or consist entirely of a video). They leverage the banner space to deliver a video experience as opposed to another static or rich media format. The format relies on the existence of display advertisement inventory on the page for its delivery. Video banner advertisements can also have interactive rich media elements within them and can pop out of their banners to display larger video advertisements.

- **Overlay advertisements**

Overlay advertisements are non-linear video advertisements that are delivered as text, graphical banners/buttons, or as video and are placed within the media player window, either over the video content itself or directly on the top edge or bottom edge of the video content during the content play.

- **In-Text video advertisements**

In-text video advertisements are delivered as a pop over when a user chooses to mouse-over relevant, apparently hyperlinked words within a block of text.

Appendix B: Content Asset Metadata Examples

There are different types of video content out there on the internet and each type has certain nuances about how it should be tagged in order to be reported correctly in comScore's Audience measurement products. This section will guide you how to populate the video metadata parameters for the most common types of content available on the internet.

Content Asset Metadata Examples

Label	TV Show Episode	TV Show Trailer	Live Sports Content	Sports Highlight Clip	Movie	Movie Trailer	Online News Content	Music Video
Station Title (ns_st_st)	Hulu	Youtube	ESPN3	Youtube	Hulu	Youtube	Huffington Post	VEVO
Publisher Brand Name (ns_st_pu)	ABC	ABC	ESPN	NFL	Warner Bros.	Warner Bros.	Huffington Post	VEVO
Program Title (ns_st_pr)	Modern Family	Modern Family	Game 16: Eagles vs Patriots	Game 16: Eagles vs Patriots	Harry Potter 7	Harry Potter 7	Huff Post Live	Taylor Swift
Episode Title (ns_st_ep)	Rash Decisions	Season 2 Teaser	*null	*null	*null	Harry Potter 7 Trailer #3	All is not Well in Hillaryland	Wildest Dreams
Episode Season Number (ns_st_sn)	1	*null	*null	*null	*null	*null	*null	*null
Episode Number (ns_st_en)	2	*null	*null	*null	*null	*null	*null	*null
Genre (ns_st_ge)	Comedy	Comedy	Sports	Sports	Fantasy, Drama	Fantasy, Drama	News	Music
Complete Episode Flag (ns_st_ce)	1	0	1	0	1	0	0	0

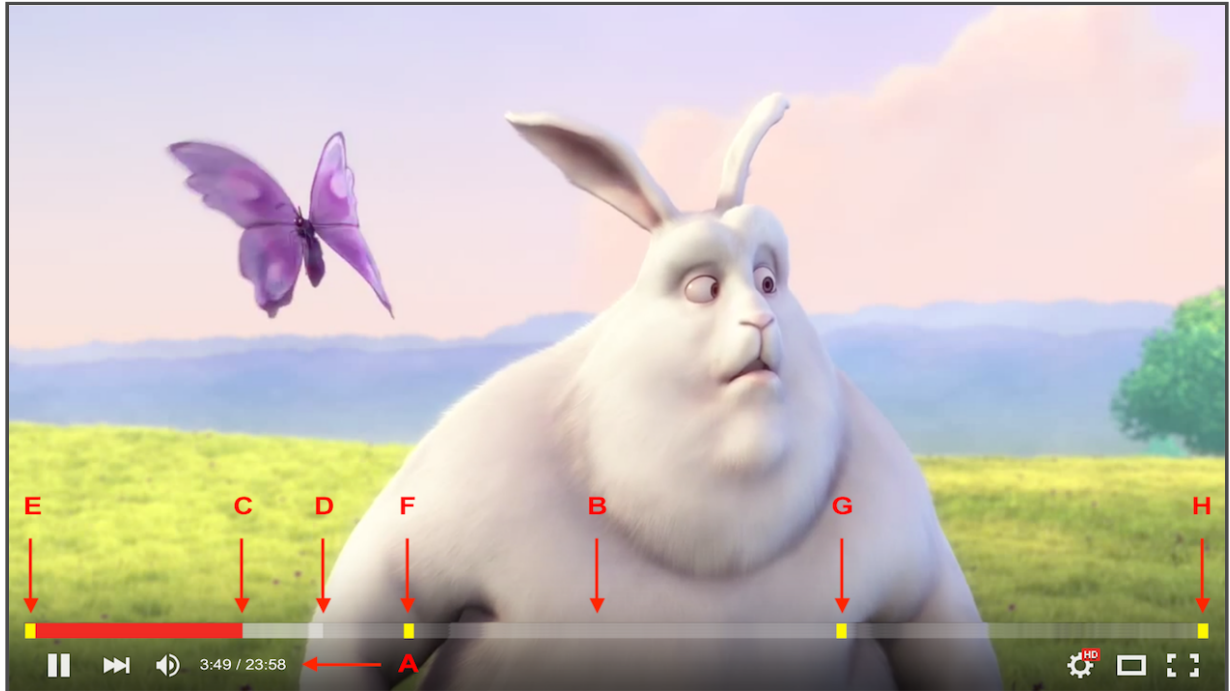
A list of suggested Genre (ns_st_ge) values is provided below:

- Action / Adventure
- Documentary
- Holiday
- Music
- Science Fiction
- Variety
- Adult
- Drama
- Home & Garden / Home Improvement
- News
- Soap Opera
- Animation
- Educational
- Home Shopping
- Paid Programming
- Sports
- Awards
- Fantasy
- Kids
- Politics / Public Affairs
- Talk
- Comedy
- Foreign Language
- Lifestyle
- Reality
- Thriller / Horror
- Food
- Game Show
- Movies
- Religious
- Travel

Appendix C: Data Collection Model

At its core the Streaming Tag data collection model tracks *intervals of playback* associated with streaming media by interpreting playback state changes. The implementation of the Streaming Tag is next to, or into, a media player. The data collected for these intervals of playback ranges from playback state and position information to media metadata to classification data and even technical details about playback⁽⁵⁾. All data is collected in name/value pairs called *Labels*.

The following image shows a screen shot of an example media player as it was playing a video, with reference markers added for various elements its user interface. This player will serve as an example in this section.



Video still taken from 'Big Buck Bunny' movie⁽⁶⁾

- **A** indicates the *Current Playback Position* relative to the length of the current media. The media player is at position 3m49s of the content media, which has a length of 23m58s
- **B** is a visual representation of the *Time Line* for the current media. It represents the media in its entirety and shows a number of relevant details which give an indication of what the media player can be expected to do next.
- **C** visually represents the *Current Playback Position* on the *Time Line*.
- **D** visually represents the amount of media data downloaded by the media player. The player has not yet downloaded the entire content media, so seeking to a position further into the content will likely cause buffering to occur. Or, seeking to a further position might not be possible altogether depending on how the media player has been programmed to behave in such cases.
- **E, F, G** and **H** are cue point markers for ad breaks. At these positions - relative to the content media - the player is potentially going to halt playback of the content media and load and play advertisement media.

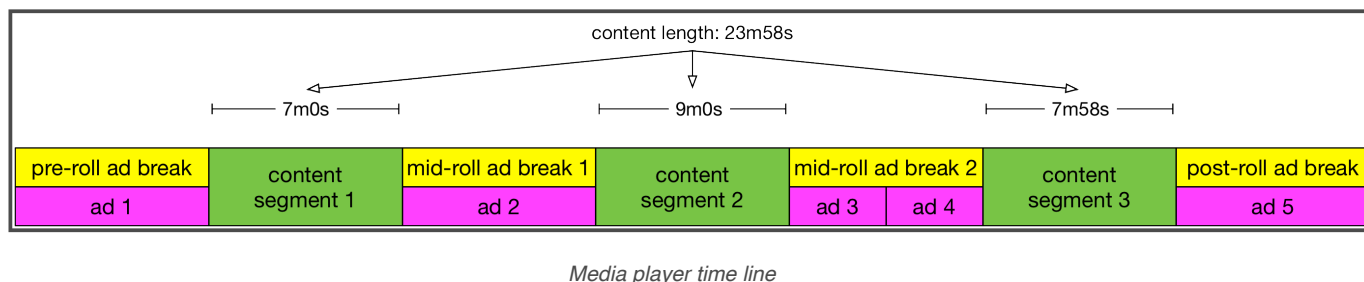
(5) The extent to which these different types of data are collected depends on the degree of implementation.

(6) © 2008, Blender Foundation, www.bigbuckbunny.org (<http://www.bigbuckbunny.org/>)

- **E** represents a pre-roll ad break.
- **F** and **G** represent mid-roll ad breaks.
- **H** represents a post-roll ad break.

Player Time Line Representation

The media player time line for the content is divided into 3 *Segments* or parts by the two mid-roll ad breaks.



Assume the following break down of ad breaks is applies to the context described above. The 3 segments have the lengths 7m0s, 9m0s and 7m58s, totalling to 23m58s (not surprisingly, that is equal to the content media length). To be clear, this puts the cue points at positions 0m0s (the pre-roll ad break), 7m0s (the first mid-roll ad break), 16m0s (the second mid-roll ad break) and 23m58s (the post-roll ad break). Except for the second mid-roll ad break all ad breaks have one advertisement assigned.

The data collection model represents each coherent content media and each individual advertisement media with exactly one *Asset*. Here 'coherent' refers to the content as a whole, regardless of whether or not the data stream for the content media is cut up into separate streams or files⁽⁷⁾ that are seamlessly switched during playback.

The example media player time line has six *Assets*: one for the content and five for the advertisements. Note that content media is represented with a single *Asset* regardless of the amount of *Segments* the content media is divided into by mid-roll ad breaks.

The *Playback Session* represents the collection of the content *Asset* and all its related advertisement *Assets* which are played on the pre-roll, mid-roll and post-roll ad breaks. If the media player does not play advertisements then each *Playback Session* will contain only the content *Assets*.



About the difference between *Playback Session* and media player playlist...

The *Playback Session* is not necessarily the same as the concept of a playlist which some media players have and present to the user. In those media players their playlist typically shows a list of (related) content media which the user can select from or which the media player will automatically play, one after the other.

If the media are not strictly related to each other, then each content media on such a playlist could be in its own *Playback Session*, together with all the advertisement media related to that particular content media. For example, the playlist contains a user-controlled selection of individual episodes of a TV Show. This is typical for most implementations.

⁽⁷⁾ This refers to cases where, for example, content with a length of 40m0s is cut into two pieces: one stream which contains positions 0m0s to 21m0s and another stream that contains positions 21m0s to 40m0s.

However, if the playlist contains media with a specific grouping purpose, then it is allowed to put all media from the playlist in the same *Playback Session* (including any advertisements). For example, when a playlist contains multiple recordings of mistakes, goofs and bloopers to create a blooper reel or when news item clips about a certain topic are presented together on a playlist.

Creating Playback Sessions

When our example media player loads its time line, a new *Playback Session* will be created. All 6 *Assets* in the example scenario will be part of this *Playback Session*. Note that - at this point - it does not matter whether or not the ad breaks at the specified cue points are utilised and show advertisements. The *Playback Session* is created regardless because the player has loaded 'new' content media.

If we assume that the user would end playback of the video in our example scenario and instruct the media player to play another movie then that action would cause another *Playback Session* to be created as the media player loads the other movie.

The following scenarios are additional examples of when you would typically create a new *Playback Session*:

- When the media player is instructed to play a TV show episode, regardless of the presence of cue points for advertisements⁽⁸⁾.
- When the media player is instructed to play a collection of short clips which are represented to the viewer as a whole, regardless of the presence of intermediate advertisements.

To be clear, in the following scenarios it is **not** expected to create a new *Playback Session*:

- When the media player is changing from advertisement media in a pre-roll ad break to the content media for which the pre-roll ad break was started.
- When the media player is changing from the content media to a mid-roll ad break.
- When the media player is changing from a mid-roll ad break back to the content media for which the mid-roll ad break was started.
- When the media player is changing from the content media to a post-roll ad break.
- When the media player is changing from the content media - or a post-roll ad break - to a pre-roll ad break for the same content media, i.e., looped playback or (automatically) repeated playback of the same content and it's advertisements.
- When the media player is looping playback or (automatically) repeating the same content media, but this time does not play any advertisement for the ad breaks, i.e., the next media asset that plays is the same content.

Changing Assets

Each discrete, coherent content media and each individual advertisement media is represented by exactly one *Asset*. In our example scenario there are 6 *Assets*. The Streaming Tag implementation will make the `ns_.StreamingAnalytics` object aware of each time the media player changes media by changing the current *Asset*. This implies that the first media change follows immediately after creating a new *Playback Session*.

The following scenarios are examples of when the implementation would typically change the current *Asset*:

- When the media player loads a TV show episode (or movie, short clip, or any other type of content) to play next.
- When the media player loads an advertisement to play next (regardless of the ad break the advertisement appears in).

⁽⁸⁾ These could be pre-roll, mid-roll or post-roll ad breaks.

To be clear, in these scenarios it is **not** expected to change the current [Asset](#):

- When the media player changes from one stream URL/location used for one segment of the media⁽⁹⁾ to another stream URL/location used for another segment of the same media.
- When the media player is changing the stream URL/location to change bit rate or streaming quality.

Content Media Playback Tagging

When your media player loads content media your implementation will include setting the current [Asset](#) and provide [Asset Labels](#). One of those asset labels will be the [Segment Number](#) label (`ns_st_pn`) which needs to be updated to reflect the segment which will be played next.

As your media player is playing content it might encounter a mid-roll ad break for which it plays advertisements. When your media player changes back to the content media - after playing any mid-roll advertisements - your implementation should update the [Segment Number](#) label value on the [Asset Labels](#) accordingly.

For example, let's assume the media player is now going to play the second [Segment](#). The current [Asset](#) has not yet been changed from the previous advertisement media to the content media. The value of the [Segment Number](#) label (`ns_st_pn`) should be set to `2` before the [Asset Labels](#) are used to set the current [Asset](#).

The [Segment Number](#) label only needs to be updated when setting the current [Asset](#). It is not needed to update the [Segment Number](#) label during playback.

Advertisement Media Playback Tagging

With some media players and advertisement servers it might not be possible to get proper media change notifications or playback state change notifications for individual advertisements. The ideal implementation - i.e., one following the instructions in this implementation document - represents each discrete media asset in your media player with exactly one [Asset](#), which might not be possible in such environments.

In those environments it is acceptable to fall back to representing each discrete content media asset with exactly one [Asset](#) and representing each *ad break* with exactly one [Asset](#). In practice this means that the ad break will be treated as if it is one coherent advertisement, rather than containing individual advertisements.

With this approach all playback state changes for the ad breaks are expected to be tagged as per the provided instructions.

Example Playback Scenario

Let's take [the example media player context from page 19](#) and assume the media player plays through the entire media player time line exactly once, where the ad server returns the specified video advertisements for each of the ad breaks. All media are played in their entirety.

(9) The media can be content - a TV show episode, movie, short clip, or any other type of content - or advertisement.

The transcript below shows how the Streaming Tag implementation should react to this playback scenario and how it instruments the `ns_.StreamingAnalytics` object. It is assumed that object is already created and available for use with the reference `sa`.

Example Playback Scenario Transcript

Step	Activity	Executed code statement(s)
1	The media player is instructed to load new content media for playback. Create a new <i>Playback Session</i> .	11. <code>sa.createPlaybackSession();</code>
2	From the available cue points on the content media the media player now loads the pre-roll ad break. The ad server returns 1 advertisement. The current asset is now set to this advertisement.	12. <code>// Argument is an Object with the relevant label values.</code> 13. <code>sa.getPlaybackSession().setAsset(ad1Metadata);</code>
3	Playback of the advertisement starts from position 0m0s.	21. <code>sa.notifyPlay(0);</code>
4	The media player reaches the end of the advertisement and playback ends.	22. <code>sa.notifyEnd(); // Will assume the position is equal to the length of the advertisement.</code>
5	The media player is going to start playing the content. The current asset is now set to the content media.	31. <code>// Ensure contentMetadata indicates segment 1 by specifying label ns_st_pn=1.</code> 32. <code>sa.getPlaybackSession().setAsset(contentMetadata);</code>
6	Playback of the content starts from position 0m0s.	33. <code>sa.notifyPlay(0);</code>
7	The media player encounters the cue point for the first mid-roll ad break. Playback of the content ends. The ad server returns 1 advertisement. The current asset is now set to this advertisement.	41. <code>sa.notifyEnd(); // Will assume the position is approximately 7m0s.</code> 42. <code>sa.getPlaybackSession().setAsset(ad2Metadata);</code>
8	Playback of the advertisement starts from position 0m0s.	51. <code>sa.notifyPlay(0);</code>
9	The media player reaches the end of the advertisement and playback ends.	52. <code>sa.notifyEnd();</code>
10	The media player is going to playing the second segment of the content. The current asset is now set to the content media.	61. <code>// Ensure contentMetadata indicates segment 2 by specifying label ns_st_pn=2.</code> 62. <code>sa.getPlaybackSession().setAsset(contentMetadata);</code>
11	Playback of the content starts from position 7m0s.	63. <code>sa.notifyPlay(420000);</code>
12	The media player encounters the cue point for the second mid-roll ad break. Playback of the content ends. The ad server returns 2 advertisements. The current asset is now set to the first advertisement in this ad break.	71. <code>sa.notifyEnd(); // Will assume the position is approximately 16m0s.</code> 72. <code>sa.getPlaybackSession().setAsset(ad3Metadata);</code>
13	Playback of the advertisement starts from position 0m0s.	81. <code>sa.notifyPlay(0);</code>

Step	Activity	Executed code statement(s)
14	The media player reaches the end of the advertisement and playback ends. The current asset is now set to the second advertisement in this ad break.	<pre> 82. sa.notifyEnd(); 83. sa.getPlaybackSession().setAsset(ad4Metadata); </pre>
15	Playback of the advertisement starts from position 0m0s.	<pre> 91. sa.notifyPlay(0); </pre>
16	The media player reaches the end of the advertisement and playback ends.	<pre> 92. sa.notifyEnd(); </pre>
17	The media player is going to playing the third segment of the content. The current asset is now set to the content media.	<pre> 101. // Ensure contentMetadata indicates segment 3 by specifying label ns_st_pn=3. 102. sa.getPlaybackSession().setAsset(contentMetadata); </pre>
18	Playback of the content starts from position 16m0s.	<pre> 103. sa.notifyPlay(960000); </pre>
19	The media reaches the end of the content after 7m58s and playback ends. The post-roll ad break is encountered. The ad server returns 1 advertisement. The current asset is now set to this advertisement.	<pre> 111. sa.notifyEnd(); // Will assume the position is 23m58s. 112. sa.getPlaybackSession().setAsset(ad5Metadata); </pre>
20	Playback of the advertisement starts from position 0m0s.	<pre> 121. sa.notifyPlay(0); </pre>
21	The media player reaches the end of the advertisement and playback ends.	<pre> 122. sa.notifyEnd(); </pre>

Appendix D: Migrating an Existing Implementation

These steps apply to Streaming Tag implementations which are migrating from comScore streaming library version 4.x or 5.x.

1. If you are updating the Streaming Tag in your OTT application or Cordova mobile application then make sure you have updated the comScore Application Tag (either the OTT library or the comScore Cordova plugin) in your project to the latest version.
2. Take note of the new File name of the comScore streaming tag library JavaScript File and update your environment accordingly. The old File name contained the word 'streamsense', which is no longer the case.
3. Replace occurrences of `ns_.StreamSense` with `ns_.StreamingAnalytics`.

About the code statements in an existing comScore library implementation...

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

4. Locate the code statement which creates the `ns_.StreamingAnalytics` object instance. Depending on your implementation it could be using a string argument with the URL for transmission of measurements. The code statement will look like this:

```
11. | var sa = new ns_.StreamingAnalytics( {}, transmissionURL );
```

5. If your implementation uses that string argument, then change your implementation as follows to specify the URL as the live endpoint URL in a configuration object:

```
11. | var sa = new ns_.StreamingAnalytics( { liveEndpointURL: transmissionURL } );
```

6. Assuming your reference to the `ns_.StreamingAnalytics` object is called `sa`, modify the following method calls to account for changes in naming⁽¹⁰⁾:

Existing call	Migrated call
<code>sa.setPlaylist()</code>	<code>sa.createPlaybackSession()</code> If the existing call specified a non- <code>null</code> argument value then copy over the argument value: <code>sa.createPlaybackSession(labels)</code>
<code>sa.setClip(labels)</code>	<code>sa.getPlaybackSession().setAsset(labels)</code>
<code>sa.getPlaylist().setLabel(name, value)</code>	<code>sa.getPlaybackSession().setLabel(name, value)</code>
<code>sa.getPlaylist().setLabels(labels)</code>	<code>sa.getPlaybackSession().setLabels(labels)</code>
<code>sa.getClip().setLabel(name, value)</code>	<code>sa.getPlaybackSession().getAsset().setLabel(name, value)</code>
<code>sa.getClip().setLabels(labels)</code>	<code>sa.getPlaybackSession().getAsset().setLabels(labels)</code>
<code>sa.notify(ns_.StreamSense.PlayerEvents.BUFFER, position)</code>	<code>sa.notifyBufferStart(position)</code>
<code>sa.notify(ns_.StreamSense.PlayerEvents.PLAY, position)</code>	<code>sa.notifyPlay(position)</code>

⁽¹⁰⁾ Any existing arguments to these can be copied as-is unless specified otherwise. Please note that this will be a 1:1 migration of the existing implementation. This kind of migration does not apply any improvements, simplifications or new features to the implementation.

Existing call	Migrated call
<code>sa.notify(ns_.StreamSense.PlayerEvents.PAUSE, position)</code>	<code>sa.notifyPause(position)</code>
<code>sa.notify(ns_.StreamSense.PlayerEvents.END, position)</code>	<code>sa.notifyEnd(position)</code>
<code>sa.notify(ns_.StreamSense.PlayerEvents.CUSTOM, labels, position)</code>	<code>sa.notifyCustomEvent(position, labels)</code>