Combined Beacon Media Player Integration

JavaScript API

(including sample code for HTML5)

nielsen

Table of Contents

1 Introduction

The Nielsen combined beacon is a JavaScript library that collects video event information suitable for consumption by the following Nielsen products; VideoCensus, SiteCensus, Market Intelligence and IAG video.

Setup consists of some configuration values (provided to you from Nielsen) and timely invocation of the appropriate combined beacon library API calls for VideoCensus, SiteCensus, Market Intelligence and IAG. All video content identifying information will leverage data that is already available from the player itself and values will be assigned, associated and transmitted dynamically to the Nielsen collection servers at the beginning and end of video playback.

Important Notes:

- If you are an **existing Video Census** customer please contact your Nielsen representative before implementing this new beacon. For accurate post data collection reporting purposes, the legacy Video Census beacon should be run **concurrently** with this new combined beacon until both you the client and Nielsen are comfortable that the new beacon is fully deployed to all video players. This requires a specific configuration of this new combined beacon.
- If you are instrumenting live streaming video please see the section in this document on live streaming. There is a specific configuration required.

2 Purpose of This Document

This document describes using the JavaScript API for instrumentation of JavaScript compliant video players. This includes Windows Media Player, Silverlight and HTML5 video players, though the reader should note that there is separate supplemental Nielsen documentation available for these video players that augments this document.

3 Prerequisites and Assumptions

A knowledge of JavaScript.

It is assumed that you have been provided with a Nielsen clientid and that you have been informed of the correct vcid to use. Your Nielsen representative can assist with mapping specific vcid to a specific player instance.

4 Combined Beacon Overview

The combined beacon consists of a javascript library that is referenced and downloaded at runtime. The addition of a few lines of code to the <script> tag where the player is located is all that is needed to enable universal audience measurement.

Messages are sent to the Nielsen imrworldwide.com site where they are collected and logged. The same set of logs also drives processing for the VideoCensus service, the SiteCensus service, Market Intelligence and IAG. The combined beacon collects, formats, and buffers event data about user experience with videos and sends it to Nielsen servers for subsequent processing. It does so using as few messages as possible, while still guarantying that every stream view is counted. This results in an accurate set of numbers that tracks view counts as well as the quality of the video viewing experience.

5 Frequently Asked Questions

1) Is collected data surface in the reporting UI in real-time?

   Data is not currently processed and surfaced in real time.

   Data received before 12 midnight in the time zone declared for a specific publisher will be available in the reporting UI by 9am next day, in the time zone declared for the specific publisher. This is the worst case scenario where a video stream start has been registered but an explicit video stop has not been sent. i.e. up to 8 hours after the video start was logged.

2) Will the beacon generate a lot of network traffic to the Nielsen data collection server?

   No. The optimal (default) configuration is to cache all events other than the initial video play start and video play end.

   The first time a visitor plays a video on any tagged publisher site, the API will make a call to fetch the beacon code from the Nielsen data collection server. The beacon will then be cached on the users machine.

   For each video play, a web request to the Nielsen data collection server is made upon receiving the first play event (event 5) after seeing a load video event (event 3).

   At the end of each video play (upon receiving a video stop event 7), or the detection of a video termination through browser close or URL change, another web request is made to the Nielsen data collection server is made to flush all the detected events during the users video view; scrubbing back and forth, pause, resume, volume up / down.

3) What Qualifies as an Ad?

   - The video is streamed from a 3rd party server (eg: Atlas or DoubleClick) or from a location on a publisher's CDN that is used for serving ads.
   - The video is shown in rotation where other advertising appears (ex: within a commercial "pod" or "chapter break" between segments of an episode).
   - The video is clearly not the primary viewing experience and intended to be shown as advertising.

4) Are Promotional Videos counted as Ads?

   - Promos are only counted when <u>not</u> shown in an ad rotation spot.
   - For example- on a page for a specific show- the user is seeking out the promotional content as their primary viewing experience.
   - If the promo is shown in an ad rotation position (ex: within a commercial "pod" or "chapter break" between segments of an episode) it will be counted as an ad.

5) Are Combined Beacon messages compatible with existing tags?

   Combined Beacon messages are a strict superset of existing messages for VideoCensus, SiteCensus Streaming and Market Intelligence. Current messages will continue to work.

6) Do we need to capture all the API events?

   No. Load Video and Play are the required events. All other events are optional or are automatically generated. But, more data capture leads to better measurement of a viewer's experience in a single video play. When aggregated across a large number of views, this results in accurate profiles of user experience with videos.

7) Why do we need a beacon anyway?

   As the number and types of events that need capture goes up, efficient messaging back to the server becomes paramount. View counts can be achieved without a beacon. Duration metrics are problematic – more accurate numbers need more messages and affect the cost of service. % viewed and other metrics are impractical without a beacon (i.e. code of some sort attached to the player). Emerging metrics for strict auditing, content-ad interaction and interactive videos are impossible without a beacon.

8) Does the combined beacon affect end-user video viewing?

   No. The beacon is a small, lightweight code that has no visible components, is entirely in the background and is designed not to interfere with video experience in any way. Almost all calls to the beacon are asynchronous – this includes the initial loading of the beacon itself and all subsequent messaging to the Nielsen imrworldwide.com collection server.

9) For the purposes of metrics we record the video stop before the actual end of the video content (Also known as the "black screen"). Can we send the stop event before the actual end of the video?

   You can send the stop video (event 7) any time you wish as long as the analytics team understand that you are doing so. If possible you should keep track of the playhead position and throw the stop video (event 7) if the user scrubs forward through the time point in question.

10) Our Flash player executes the content video start a split second before it executes the preroll. The user does not visually see the content video before the preroll video but it does result in the video play event for the content executing before the video play event start executing for the preroll Ad. How would the beacon handle this scenario?

    It is OK to send the start beacon for the content before the start beacon of the preroll Ad. Typically we like to see video stream start and end beacon calls supplied as discrete pairs of calls, however we can accept a Ad video start without seeing the stop event for the already sent content video start. What we CANNOT accept is another additional Ad video start or content video start without first receiving a stop event for the already "open" Ad video and / or content video stream. i.e. we only allow one level of call nesting.

11) I have a number of different player technologies on the website that use different cookie technologies (i.e. Flash cookie with Flash player, browser cookie with Silverlight) but I wish to report the data into the same Nielsen Analytics Account, which Nielsen beacon should I use?

You should choose your cookie type and stick with it for all players. This is important from a methodology perspective as the persistence of a particular cookie can vary. i.e. Flash cookies tend to be more difficult to remove from a users environment and therefore that affects the Unique Viewer metric.

The Javascript API in native mode will use a browser cookie. The Flash AS2 API and AS3 API will use a Flash cookie. The Javascript API in Combo/SWF mode will use a flash cookie if it can, otherwise it will use a browser level cookie.

12) My organization has mandated that we should not force cookie tracking technology on our users. Is there anything I need to do in the video player / API?

There is no beacon specific change required. Individual users may visit a privacy page on the Nielsen website and "opt-out" of tracking. Any subsequent video views / plays that trigger the beacon will not be tracked for that user.

13) We do not, or cannot use Flash cookies. Do you have a video beacon technology that uses regular browser cookies?

Yes. You should use the Javascript API in native mode which will force the use of the Javascript beacon for collecting metrics and will use a browser cookie.

14) We don't use a Flash player, but what would be the advantages to us of switching to a Flash player and using the Flash variant of your API and beacon?

We do not mandate that you use the Flash API but we do strongly encourage you to do so. The Flash API gets bound / compiled into your player and therefore is guaranteed to "travel" with your player if it gets embedded in other publisher websites. This ensures that all end-user plays get captured and registered. If you used the Javascript API to instrument your Flash player then you would have to make sure other publishers that embedded your video player also inserted the Javascript API and instrumentation which would be an additional management overhead.

15) Why has my Nielsen TAM asked me to set **cisuffix** to "gg"?

Many of our existing customers already have the Video Census beacon in place. This new super beacon is designed to also send the same information to Video Census. Many of our customers have communicated a resistance to the extra work in taking the old Video Census beacon down.

As such, we have to allow for the situation where the old Video Census beacon and new super beacon execute on the same property. If left unaddressed, this would result in double-counts in Video Census. This flag, when set to "gg", indicates to our servers that we should ignore the super beacon for the purposes of counts in Video Census.

At some future point, once all the properties in a given client account are tagged with the new super beacon then the old VC beacon can be removed and the **cissuffix** set to "". At that point, Video Census will then be fed counts from the new super beacon.

Also see Dual Beacon topic in the overview section of this document.

16) How do you detect that a user has abruptly closed the browser / video player or abruptly navigated away from the URL hosting this viewing of the video?

We embed some JavaScript that detects these browser conditions and generates the appropriate event for our processing. This gets stored in the beacon buffer and gets sent to the Nielsen server next time a user watches another video, or in fact any video on any publishers website that is tagged with our beacon technology.

The VA backend processing will process messages for a given video stream that are time stamped up to eight hours after the initial video start message (event 15 or event 3 + 5).

If an explicit video stop event (event 7) or an unload video event (event 2) is not seen within that time frame for a video stream then the video will be deemed stopped and the stream count credited. The duration of that video will be approximated based on the last event received for that video stream. This could be any API event; pause, stop, forward, back, periodic play update (event 49)

17) It is already the 7th of the month and the Demographics data is not showing for the previous month. Why?

Demographic data for the previous month is typically available on the 11th of the month. i.e April data will be available in the UI on the 11th of May.

18) What is the attentiveness score?

The attentiveness score is a computed metric that indicates how engaging the video content is. The greater the score, the more engaged the user base is. Maximum score is 100. It is computed from the following user behavior when a video is in play:

- **Viewing**: Time measurement
- **Focus**: Is the video player in a foreground window (in focus) or background window
- **Interaction**: Positive event adjustment such as volume change.
- **Affinity**: Repeat viewing

Contact your Nielsen representative if you require more detail.

19) How is the Referring URL for a video derived?

The Nielsen beacon uses standard JavaScript script (“document.referrer”) to extract the referring URL from the header.

It should be understood that this property can be blocked or manipulated which can lead to misleading information being returned to the beacon when it makes this inquiry.

Common reasons (but not limited to) for this blocking are:-

- The referring URL has been removed or NULL out.
- Anti-virus software on the user PC has masked or removed the referring URL.
- The referring URL can be manipulated / changed by the web server. You would need to ask the webmaster of the client website.

- The web server can block the JavaScript that is used by the beacon to extract the referring URL from the windows.location.referrer property in the HTML DOM. You would need to ask the webmaster of the client website.

20) The Video Stream will not playback in the Reporting UI SynchPlayer. Why?

For each video stream start recorded by the beacon, the second parameter in the event 3 or event 15 must supply the full file path URL for play back purposes. Assuming that the full path is supplied for each video file, then the following conditions also have to be met:-

a) The video file is not encrypted. i.e. RTMPe are usually not playable outside of the publisher’s player.

b) The video file is accessible to the public internet. If the publisher is hosting the video file behind a secure mechanism that requires the video player to pass a hideen sign-on or other mechanism to get access then the Nielsen Video Analytics UI may not have permission to access the file URL.

c) The file path URL is still valid. The URL may have been valid at the time the stream was recorded but subsequently the publisher may have deleted it, or moved it to another location.

d) The file is an unsupported video type. At this time we support playback of FLV video.

Note: Nielsen Video Analytics does not store the video file in it’s database. It stores the URL reference to the video file.

21) What does the Period Analysis data represent?

The numbers represent when the user watched video during the day. The activity is recorded per the users local time zone.

22) How does Video Analytics define the session metric?

A session is closed out when for a specific user (cookie) no video player activity has been recorded for a continuous 30 minute period.

23) Do I have to do anything different for live streams?

Yes. Depending on the type of live stream, you will need to set the meta-data accordingly. Please see separate section in this document on live streaming.

24) In I.E. I get a browser security warning on the dashboard and then thumbnails do not load in Top Content Titles. Why is this?

The Video Analytics UI is attempting to resolve a URL/path to the publisher’s website to retrieve the thumbnail graphic. This is triggering a security policy on your browser / machine.

6 Video Player Tag Business Rules

There are generally two types of content presentation. Single clip content and multi-chapter full episode content. Both types are tracked as the same type in Video Analytics, though multi-chapter content is subject to a specific set of Nielsen business rules.

Video Ads are also tracked by Video Analytics and IAG. Advertising is recorded separately from content because it enables post-buy reporting and advertising effectiveness studies. For futher clarification on what defines an Ad (as apposed to content) see the FAQ section. Three basic types are supported by

Video Analytics; preroll, midroll and postroll. For video Ads that are triggered via a user click on an overlay, they should be indicated as type midroll.

Important: You can only have one instance of an “open” Ad stream tag and content stream tag at any given time. For example, the start (event 3 // 15) of the preroll could have been indicated, and the start (event 3 // 15) of the content could have been indicated, but you cannot send another Ad or content stream start until you have closed the previous open Ad or content with video stop/termination (event 7)

Terminology Note: in this section you will see reference to **dav0** and **dav2**. These are indicators to the Nielsen collection servers that a video stream measurement must start and end respectively. The next section on how to use the beacon test tool will make reference to these indicators.

6.1 Single Clip Content and Ads

Single clip content is the easiest to validate. It will have a video start event 3 + 5 or 15 that will generate a dav0 web request at the start of the video, and a video end event 7 that will generate a dav2 web request.

Often the content will be preceded by a preroll Ad. This should also be measured with its own start event 3 + 5 or 15 (dav0) and a video end event 7 (dav2).

6.2 Full Episode / Multi-Chapter Content and Ads

Full episode content is a little more complex. Typically the publishers will divide their content up into chapters. Between each chapter a midroll Ad will execute.

The complication comes in with regional market differences in the way we have to measure multi-chapter content for stream count purposes. For the US and UK market we have to measure each content chapter separately. For non-US / UK markets it is ok to have one stream for the content irrespective of the number of chapters. All Ads have to be measure discretely irrespective of market.

6.2.1 US // UK Market Only

Nielsen’s methodology is that long-form content should be broken up into segments (typically at cue-points where one or more ads are shown) and each segment view be treated as a separate video play for stream count purposes. To this end, the chapter indicator (parameter 4 of event type 3 / 15) should always be supplied when recording full episode / multi-chapter content.

In order to capture accurate counts in the Nielsen panel, the following rules must be followed in regard to recording distinct video plays for stream count purposes.

A) **A Stream Start SHOULD be sent when the following conditions are met:**

- Send a start call for each “chapter” the first time it plays, regardless of whether an ad plays in between the chapters.
- If the user scrubs forward (ffwd) or backward (rwd) to a different chapter, <u>and</u> this is the first time that chapter has been accessed, send an additional beacon call for that chapter.
- If the user scrubs forward (ffwd) or backward (rwd) to a different chapter, <u>and</u> the user has already viewed part of that chapter, only send an additional beacon call if an ad plays in-between.

Example: a full episode with 5 segments might have 5 start calls, potentially more if the user revisits a prior segment and is interrupted by an ad.

B) **A Stream Start SHOULD NOT be sent:**

- If the user scrubs forward (fast forward) or backward (rewind) to a different chapter, AND the user has already viewed part of that chapter (it has previously been accessed) AND no ad plays in between – do NOT send a beacon.

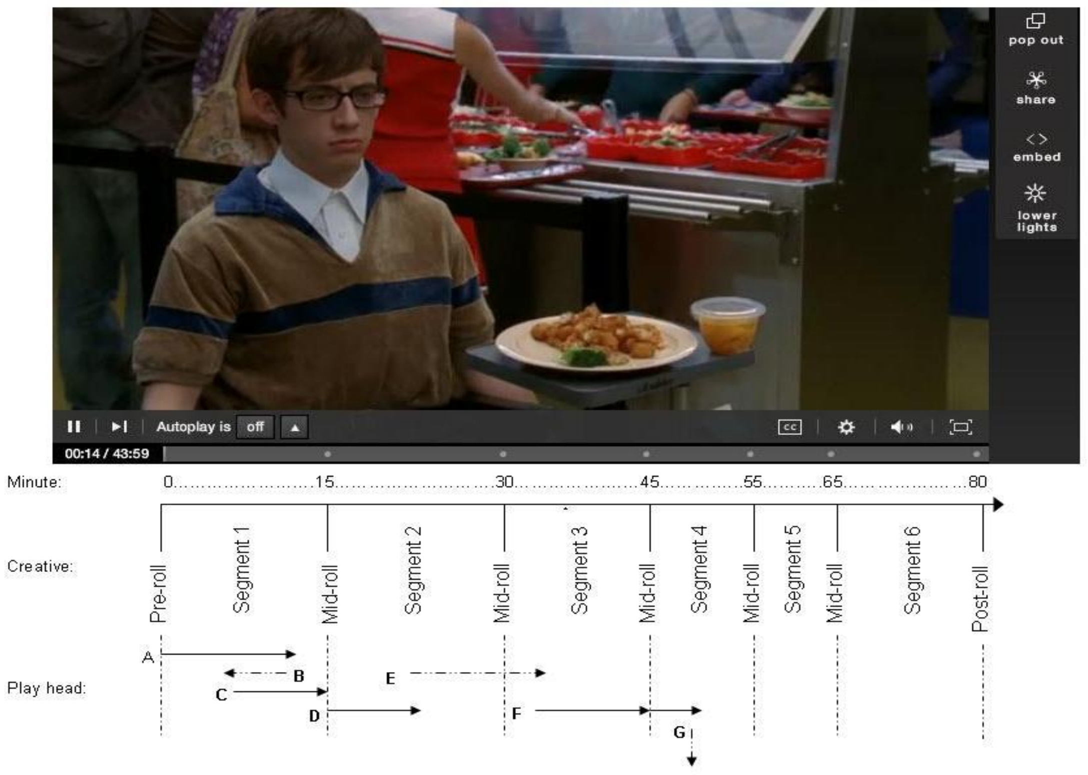
- If the user scrubs all the way back to the beginning of the video (the 0:00 position) – do not count an additional stream.
- Pausing and then resuming – do not count an additional stream.
- Adjusting video quality/resolution (ex: switching to HD) with a seamless transfer (where the video automatically seeks to the same play position that the user was viewing at the previous resolution).
- Changing to full screen or back to regular size.

6.2.2.1 Typical Beacon Firing Pattern

To help translate the above rules into real-world scenarios, below you will find a couple of typical examples of a user moving around a long-form video.

Example A – No Chapter Revisits

In this scenario the user scrubs around the segments but they never revisit a previously watched chapter.



Following the A through G sequence outlined in the above graphic, the events generated would be:-

Step A:

Pre-roll Start - Event 3 or 15

Pre-roll End – Event 7

Pre-roll Unload - Event 4 - Optional

Content Chapter 1 Start - Event 3 or 15

Step B:

<Scrub back within chapter 1> - Event 8

Step C:

<Resume play to end of chapter>

Content Chapter 1 End - Event 7

Step D:

Mid-roll Start – Event 3 or 15

Mid-roll End – Event 7

Mid-roll Unload - Event 4 – Optional

Content Chapter 2 Start – Event 3 or 15

Step E:

<Scrub forward into Chapter 3>

Content Chapter 2 End – Event 7

Note: because the user crosses the chapter 2 / 3 break, AND a mid-roll then triggers, chapter 2 is effectively terminated. Therefore an event 7 should be generated, not a seek event 8.

Mid-roll Start – Event 3 or 15

Mid-roll End – Event 7

Mid-roll Unload - Event 4 - Optional

Content Chapter 3 Start – Event 3 or 15

Step F:

<Play to end of chapter>

Content Chapter 3 End – Event 7

Mid-roll Start – Event 3 or 15

Mid-roll End – Event 7

Mid-roll Unload - Event 4 – Optional

Content Chapter 4 Start – Event 3 or 15

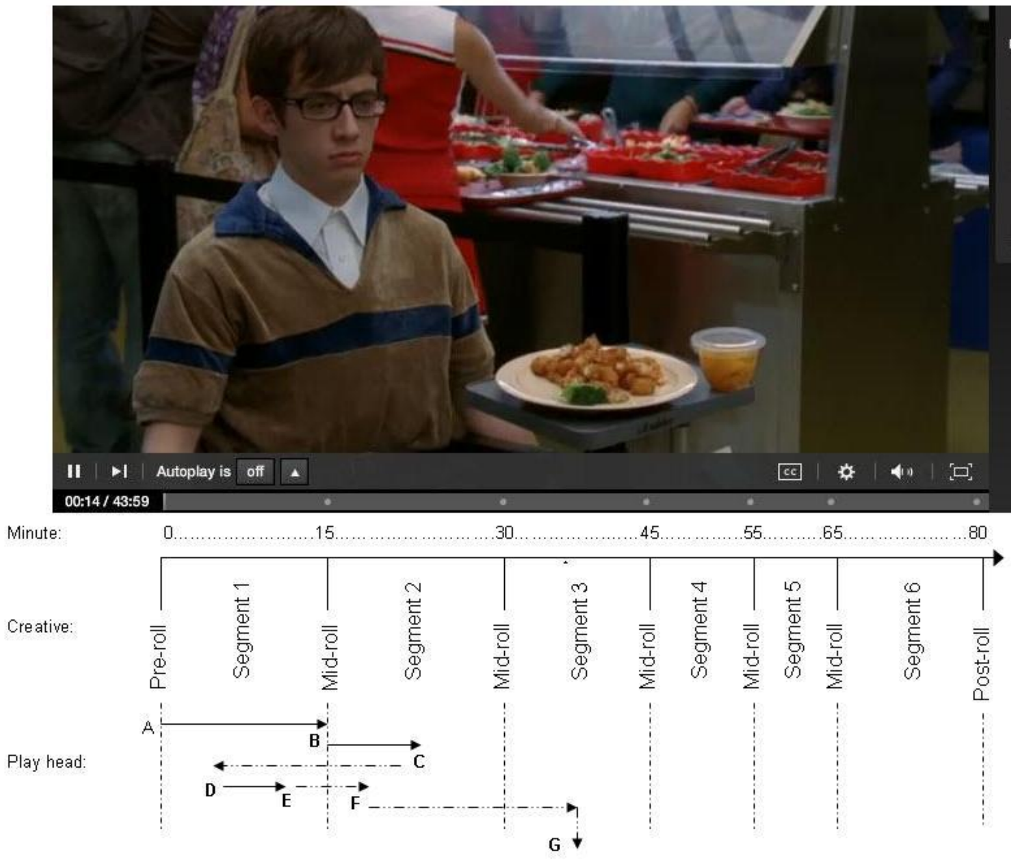
Step G:

<the user abruptly exits a few seconds into chapter 4>

Beacon detects URL or browser close – Event 2 auto-generated

Example B – Chapter Revisits

In this scenario the user scrubs across chapter breaks, including revisits to previously watched chapters. In addition, with this publisher if a user scrubs back across a previously encountered chapter break then the mid-roll Ad associated with that chapter break is not triggered. I.e. the previously watched chapter immediately resumes play. Therefore the beacon for this previously watched chapter should NOT be fired.



Following the A through G sequence outlined in the above graphic, the events generated would be:-

Step A:

Pre-roll Start - Event 3 or 15

Pre-roll End – Event 7

Pre-roll Unload - Event 4 - Optional

Content Chapter 1 Start - Event 3 or 15

Content Chapter 1 End – Event 7

Step B:

Mid-roll Start – Event 3 or 15

Mid-roll End – Event 7

Mid-roll Unload - Event 4 – Optional

Content Chapter 2 Start – Event 3 or 15

Step C:

<Scrub back into chapter 1> - Event 8

Content Chapter 2 End – Event 7

Note: because the user scrubbed out of chapter 2 back into chapter 1 then the chapter 2 is effectively done with from the Nielsen perspective. For this publisher because the mid-roll associated with this break between chapter 1 and chapter 2 has already been shown once before it is their policy NOT to show an Ad again. Therefore per the Nielsen rules, chapter 1 should not be measured again.

Step D:

<Scrub forward within chapter 1>

Note: again, because we are not currently measuring this chapter then there is no reason to send an event 8. However if it is problematic to selectively suppress the event it will be ignored if we receive it. This is because there is no "open" event 15 at this time.

Step E:

<Scrub forward into chapter 2>

Note: because the user scrubbed out of chapter 1 back into chapter 2 then the chapter 1 is effectively done with from the Nielsen perspective. For this publisher because the mid-roll associated with this break between chapter 1 and chapter 2 has already been shown once before it is their policy NOT to show an Ad again. Therefore per the Nielsen rules, chapter 2 should not be measured again.

Step F:

<Scrub forward into chapter 3>

Note: this time because the user is scrubbing over a new mid-roll break not previously triggered then the publisher forces the Ad to be shown before starting chapter 3. This satisfies the Nielsen rule for sending an event 15 for the chapter 3. Even if the mid-roll did NOT execute, the user is also visiting chapter 3 for the first time so the event 15 should be sent for chapter 3 anyway.

Content Chapter 3 Start – Event 3 or 15

Step G:

<the user abruptly exits a few seconds into chapter 4>

Beacon detects URL or browser close – Event 2 auto-generated

See appendix G for another example with actual transcript of actual beacon calls and decrypted event data.

6.2.2 Non US // UK Market Only

Because Video Census has yet to launch in other markets, and full-episode video streams tend to be single video creatives now, it has been decided that for non-UK/US markets we will measure the video content stream once irrespective of any intervening Ad behavior. Ad's will continue to be measured every time they show.

A typical sequence of video start (dav0) and video end (dav2) for multi-chapter content is outlined below. Note that the "Unload" entries are NOT optional in this scenario because the Mid-roll Ad Start/End pairs effectively overlaps the Content Start / End pair.

Preroll Start – Event 3 or 15

Preroll End – Event 7

Preroll Unload – Event 4 - Mandatory

Content Start– Event 3 or 15

Midroll Start – Event 3 or 15

Midroll End – Event 7

Midroll Unload – Event 4 - Mandatory

Midroll Start – Event 3 or 15

Midroll End – Event 7

Midroll Unload - Event 4 - Mandatory

Content End – Event 7

Content Unload - Event 4 - Mandatory

This is valid as there is only one “open” Ad video stream and only one “open” content stream at any given time.

7 Cookie / Unique Browser Measurement

7.1 Video Analytics

The combined beacon currently supports two types of cookie technology for the tracking of unique browsers / visitors in Video Analytics; the standard Nielsen IMR cookie and a Flash cookie.

In order to maintain consistent measurement across all video streams recorded, ideally one cookie type should be chosen and stuck with on a per publisher basis.

If the publisher insists on using multiple players that use different cookie types then the cookie type reported to the Nielsen Video Ops team during setup of the account should be Nielsen IMR. This is because the backend processing uses the Nielsen IMR methodology as the lowest common denominator for Unique Browser calculations, even for Flash cookie tracked metrics.

Note: Although Video Analytics at the time of writing does not support rollups at the brand level for channels defined within it, if you had a channel instrumented using an IMR cookie and another channel instrumented using a Flash cookie rolling up to a brand then you would have inconsistent UB at the brand level due to the differences in methodologies.

Video players are generally built around specific technologies such as Flash, Silverlight, Windows Media Player, or JavaScript.

Often driven by commercial considerations, publishers are often restricted in the video player technology that they can employ or have present on their website. I.e. Publisher X has signed an exclusive agreement with Microsoft to only have Silverlight technology on the website.

Taking all the above into consideration, the new combined beacon offers the following support for video player types, technology platforms and cookie types:-

Player Type / Brand	Cookie Type	Beacon Flavor and Mode Supported
Brightcove v2, v3 or v4	Flash	Brightcove specific beacon

thePlatform	Flash	thePlatform specific beacon
Microsoft Windows Media Player	Nielsen IMR	JavaScript API running in native mode
	Flash	JavaScript API running in SWF mode (JavaScript player events triggering the Flash beacon)
Microsoft Silverlight	Nielsen IMR	JavaScript API running in native mode
	Flash	JavaScript API running in SWF mode (JavaScript player events triggering the Flash beacon)
Flash 8 Players (AS 2)	Flash	Flash AS2 integration method OR JavaScript API running in SWF mode (JavaScript player events triggering the Flash beacon)
	Nielsen IMR	JavaScript API running in native mode
Flash 9 & 10 Players (AS 3)	Flash	Flash AS3 integration method OR JavaScript API running in SWF mode (JavaScript player events triggering the Flash beacon)
	Nielsen IMR	JavaScript API running in native mode

7.2 All Other Nielsen Products (Video Census, Market Intelligence, Site Census Streaming)

Independent of the underlying video player technology being used (Flash, Silverlight, etc.) and the beacon flavor being implemented, the cookie passed to other Nielsen products will ALWAYS be the Nielsen IMR cookie.

Therefore you should be aware that Unique Browser counts in Site Census Streaming, Market Intelligence and Video Census might be slightly different from those surfaced in Video Analytics due to methodology differences between a standard cookie (Nielsen IMR) and a Flash cookie.

8 Initial Setup & Beacon Configuration

The JavaScript API for the Combined Beacon library is at: http://secure-us.imrworldwide.com/novms/js/2/ggcmb370.js

Integration consists of initialization and calls to supply data about player events.

The JavaScript Combined Beacon library communicates with the Nielsen message collection servers at imrworldwide.com.

8.1 Setting the Brand/Channel Id (ClientID / VCID)

In order for Nielsen to correctly credit video streams to a publisher / Ad network, a central tracking mechanism is used called the Nielsen Market View Dictionary. This dictionary is used across many Nielsen products including Site Census, Video Census, and Video Analytics. Ask your Nielsen representative if you wish to learn more about the Market View Dictionary.

For the purposes of video player integration, you will need to know which id's are associated with specific websites. Your Nielsen TAM will be able to assist with this.

If you reporting all video traffic into one Market View Dictionary ID ("*clientid*" and "*vcid*" pair) then you can simply hard-code the VCID. Find *_nolggGlobalParams* and change initial value of *clientid* to Nielsen-supplied *clientid* for your company. e.g. clientid: "us-12345". Also change initial value of *vcid* to Nielsen-supplied VideoCensus id for the lowest level in the Marketview hierarchy. E.g. vcid: “c01”. It should be included and initialized in the <body> of your HTML page. See below for an example.

If your integration is spanning many websites and reporting video traffic into many different Market View Dictionary ID’s then you will need some other mechanism (such as a Content Management System) to dynamically supply the correct *clientid* and *vcid* pair at runtime. These are part of the global variable *_nolggGlobalParams*.

Example:

```
<script type="text/javascript">
   var _nolggGlobalParams = {

             clientid: “ my Nielsen assigned client id ”,

             vcid: “ my Nielsen assigned vc id ”,

             cisuffix: ““,

             sfcode: "Nielsen assigned data node “,

             prod: " Nielsen assigned product code "

};

</script>
```

8.2 Page Bootstrap

```
<script type="text/javascript"
     src=" http://secure-XX.imrworldwide.com/novms/js/2/ggcmb370.js">

</script>
```

Where “XX” is the Nielsen supplied country code for your region. For example, it is “us” for customers in USA.

Complete the initialization by including the following lines.

```
<script>

             var canUseSWF = false; // retained for backwards API compatibility – ignored post v360

             var uid = 0;

             var detectBrowser = true; //optional – used to disable window object calls

             var gg1 = new gg();

             gg1.ggInitialize(_nolggGlobalParams,uid,canUseSWF,detectBrowser);

</script>
```

8.3 Message Interval (MSGINT)

The MSGINT variable is defined in nolggGlobalParams. By default, MSGINT is null.

By default there are two messages (GET or POST) per stream - once at the start of the stream and one at the end of stream. There may be a lot of events generated by the API, but they get buffered up in the client browser / machine memory and get sent at the end of the stream.

Customers may occasionally want milestone messages when the play has reached a certain stage.

You must get approval from Nielsen before changing the MSGINT parameter as it may affect the traffic load on the Nielsen IMRWorldwide collection nodes. This may also incur an additional monetary cost depending on the amount of traffic generated.

It can be of the form "x1,x2,x3.." where each xN is either a number (position in seconds) or a number and % (position as a fraction of length in seconds). So for a 200sec video, "10%,25%,75%" may send up to 3 additional messages, at 10 seconds, 50 seconds and 150 seconds.

```
<script type="text/javascript">
  var _nolggGlobalParams = {

         msgint: “10%, 25%, 75% ”

};

</script>
```

The MSGINT window is used in combination with the Periodic Play (event 49) event to provide more granular analytics around users who terminate a video play through an abrupt browser close or URL change mid-flight through a video play.

If you change the default MSGINT configuration then you should also ensure that at least one event 49 is dispatched “inside” each window of time defined in the MSGINT. The recommended interval for event 49 is every 15 seconds for content, and every 5 seconds for Ads.

Special Note for Market Intelligence**:** If the beacon is to provide data to Nielsen Market Intelligence then it is recommended that MSGINT be set to 25%, 50% and 75%. It is also recommended that event 49 (Periodic Play) be sent every 30 seconds.

9 Player Integration and Example Code

This section will step you though the collection and identification of meta-data and subsequent instrumentation of the video player.

See appendix D for a complete example of instrumenting a JW Player using JavaScript.

See appendix E for a complete example of instrumenting a HTML5 <video> tag using JavaScript

9.1 Basic API

The combined beacon can collect up to 30 individual video events. These range from basic events such as play, stop, pause, rewind, and fast forward, to more specific events such as Click through URL. See appendix A for a complete list of Video Analytics events.

In a typical scenario, the Video Player is loaded when its HTML page is loaded; it then plays one or more content videos, zero or more ad videos and stops. The Player is unloaded when the browser closes or moves to a new URL.

The beacon automatically generates Load Player and Unload Player messages. The bulk of the player integration effort consists of calling one or more Video Analytics functions for events that happen in the context of a single video.

A typical sequence goes like this:

Load Video *(Event 3)*

Play *(Event 5)*

… zero or more events for pause, seek, still playing etc.

Stop *(Event 7)*

9.1.1 Example Code

Call ggPM for various events as described in Section 3. Almost all interaction with Combined Beacon library is through the ggPM call, which is of the form:

```
gg1.ggPM(  functionType,param1,param2,param3,param4  )
```

functionType is a required argument (see Events below). Other parameters may be optional or required depending on the function code.

Here is an example of calls you have to make for a video that is started, played half way and stopped:

```
//Load Player  is automatically handled. You don’t need to insert that call.
// Load "Content" Video
gg1.ggPM(15,
          "http://cust.com/videos/content1.mp4",
          "content",
          "<uurl>http://cust.com/?w=73ydbhH </uurl><length>300</length>");

gg1.ggPM(49, 30);

// Stop at location 150 (second)
gg1.ggPM (7, 150);

// Unload Player is automatically called when the browser closes or moves to a new page.
```

Here’s another example of calls for a video with a 15 sec preroll ad.

```
gg1.ggPM (15,
          "http://adnetwork.com/ad1.mp4",
          "preroll",
          "<uurl>http://adnwrk.com?adid=773GGSG </uurl><length> 15 </length>");
gg1.ggPM (7,15);
gg1.ggPM (4, 15, "preroll");

gg1.ggPM (3,
           "http://cust.com/videos/content1.mp4",
          "content",
          "<uurl>http://cust.com?w=hhd766dbFF </uurl><length> 300 </length> ");

gg1.ggPM (5, 0);

gg1.ggPM(49,30);
gg1.ggPM (7, 150);
```

When the video viewer’s browser moves to a new HTML page, it is automatically unloaded and flushes buffered messages.

9.2 Additional API Calls & Metadata Handling

In order for Video Analytics to provide accurate and meaningful metrics then a certain amount of meta-data must be supplied. Rules around meta-data are summarized in this section.

Metadata refers to generally static information that describes a video (content or ad) that is playing. Category, title, etc. constitute metadata for the playing video. You will see this referenced as *VideoInfo*. See appendix B for a complete list, including a summary of mandatory meta-data items.

Metadata parameters are passed during the API call for the Load Video (event 3) or Load and Play Video (event 15). Typically metadata like title etc. is available from the playlist definitions (i.e. SMIL) or you get these values from multiple services orchestrated by your video player. i.e. a CMS

9.2.1 MetaData: No Special Encoding!

The XML style sent in parameter 3 of event 3 or event 15 should be sent as a plain text UTF-8 string.

If your environment / setup requires you to HTML encode out of Flash and HTML decode when it is passed to the JavaScript then you should use something like encodeURIComponent in Flash, and decodeURIComponent in JavaScript to preserve the original plain UTF-8 string.

To illustrate the difference between HTML and plain UTF-8, see the below.

HTML encoded:

&lt;uurl**&gt;**http://tv.repubblica.it/home**&lt;**/uurl**&gt;&lt;**title**&gt;**Crisi, allItalia serve un altro governo**&lt;**/title&gt;&lt;length&gt;820&lt;/length&gt;&lt;category&gt;rubriche###repubblica_domani&lt;/category&gt;&lt;subcategory&gt;Dossier speciale su Sara Carbonero&lt;/subcategory&gt;

Plain UTF-8:

<uurl>http://tv.repubblica.it/home</uurl><title>Crisi, allItalia serve un altro governo</title><length>820</length><category>rubriche###repubblica_domani</category><subcategory>Dossier speciale su Sara Carbonero</subcategory>

9.2.2 Content and Ad Meta-Data: Categorization, Title, Video Length, etc.

In order to differentiate between content video streams and Ad streams, event 3 or event 15 accepts a parameter to indicate what type of video is playing.

Note: Video Ads of type overlay are specified as midroll.

Param2 of event 3 or event 15 calls is used to specify ads. If param2 is preroll, midroll or postroll, the load refers to an ad. Otherwise, the load refers to a content video. Ad play events can be nested inside the Begin and End of a content play. However, events from two content views or two ad views cannot be nested.

9.2.3 MetaData: Content Video

Video Analytics provides two levels of categorization within a specific MarketView Brand / Channel. Although optional, for a better reporting experience in Video Analytics you are strongly urged to categorize your video streams.

<length> = length of the content clip / chapter. This is a mandatory item. *See a Live Streaming section for guidance on setting live clip length.*

<category> - this is the first level of categorization on a per video basis. An example would be "Sports". Video Analytics will default the value to "General" if you do not supply a value.

<subcategory> - this is the second level of categorization on a per video basis. It is hierarchically below **<category>**. Video Analytics will default the value to "Misc" if you do not supply a value.

<censuscategory> - enables client defined reporting in Video Census. See section on Video Census for more details.

<title> - You are strongly urged to specify a readable, descriptive title value for reporting purposes. This is a mandatory item. It is accepted that some video player implementations may not have access to a descriptive title from a CMS. In these instances you should populate the UURL value into the <title>. If <UURL> is not being populated then you should specify the video filename in <title>. Video filename is parameter 2 of event 3 / event 15.

9.2.4 Meta Data: Advertising Video

For advertising video, these attributes can usually be populated with meaningful data from the publisher CMS or other meta-data source such as a playlist.

Typical population is as follows:-

<length> = length of the Ad. This is a mandatory item.

<category> = advertiser name (i.e. Unilever) or a unique id that can be tracked back to an advertiser name.

<subcategory> = campaign name, flight name, or brand name. . i.e. AT&T

<title> = campaign name, creative name, or creative URL. This is a mandatory item. i.e. AT&T Go Phone April

However, it is not uncommon for the Ad networks to not share meta-data information about the video. In these cases we recommend the following rules (in order stated below) be followed for the population of <title>:

1. If a valid, distinct <uurl> (that points to the original source of the Ad video asset) value is available then set <title> = <uurl>
2. If a valid, distinct <uurl> (that points to the original source of the Ad video asset) value is NOT available then omit <uurl> altogether and set <title> to some other unique Ad video asset id. This unique Ad video asset id MUST be a value that is recognizable to the research team who are the typical consumers of the resultant Video Analytics metrics. Typical examples of an Ad video asset id is **campaignId** or **assetId**.

9.2.5 Meta Data: Uniquely Identifying a Video Play in Video Analytics

Video Analytics creates a new record for a video stream based on the combination of values supplied in the tags **<title>, <category>, <subcategory>** and **<uurl> / mediaFile**. See appendix B for specific explanations of each tag. The **mediaFile** is the video URL or video unique identifier supplied in parameter 2 of event 3 or event 15.

There are two methods that can be used to uniquely identify a video record. Method one is the default method and will be assumed during initial account setup. If you wish to use method two then you should contact your Nielsen client services representative.

Method One: Combination of <title> / <category> / <subcategory> / <uurl> or <mediaFile>

This is the default method.

The valid combination rules of <title>, <category>, <subcategory>, <uurl> / <mediaFile> are represented in the state table below. In the case of the UURL not being supplied in the XML style meta-data of the tag then the system defaults to using the mediaFile in the state rules.

Important Note the <uurl> value must be unique for the life time of the creative video asset that it points too. If the value cannot be guaranteed to be the same then omit <uurl> all together. If <uurl> is not supplied in the tag then the mediaFile entry is assumed to be unique.

It is strongly recommended that you discuss/socialize these rules with the consumers of the resultant Video Analytics. I.e. the research department.

Title	Category	SubCategory	UURL / mediaFile	New Record Created?
The same	The same	The same	The same	No
The same	The same	Different	The same	No
The same	Different	The same	The same	No
Different	The same	The same	The same	No
Different	The same	Different	The same	No
Different	Different	The same	The same	No
The same	Different	Different	The same	No
Different	Different	Different	The same	No
The same	The same	The same	Different	Yes
The same	The same	Different	Different	Yes
The same	Different	The same	Different	Yes
Different	The same	The same	Different	Yes
The same	Different	Different	Different	Yes
Different	Different	The same	Different	Yes
Different	The same	Different	Different	Yes
Different	Different	Different	Different	Yes

Method Two: Combination of <title> / <category> / <subcategory>

This method defines uniqueness as a property of the combination of <title, <category> and <subcategory>. <uurl> and <mediaFile> are NOT factored into the evaluation per method one.

For instance if the same creative UURL or mediaFile is mounted one or more times with a different combination of <title> / <category> / <subcategory> then a distinct record in Video Analytics will be created.

Although this method is very flexible, it does assume that the publisher has good controls in place when mounting content and Ads in the content management system / authoring system behind the video player. If firm controls are not in place around the specification of <category>, <subcategory> and <title> then records can become fragmented in Video Analytics.

9.2.6 Meta Data: Live Streaming

Important Note for Market Intelligence: *Market Intelligence requires that live streaming be measured in the same way as Video on Demand. Therefore if the video beacon is to serve both Market Intelligence and Video Analytics a specific configuration needs to be made in the Video Analytics Database during account creation.*

Live streams are processed slightly differently from video on demand streams. This is because live streams can be more ad-hoc in nature. A full break down of all the live stream types along with examples of the parameter 1 of each event type (5, 6, 7 or 8) is listed out in detail below.

Note: only event type 3 (load Meta data) followed by event type 5 (video play start) can be used. You CANNOT use event type 15 (combined load Meta data and video play start).

1) **Setting Meta-data in Event 3**

<LIVESTREAM>

Each live stream must be identified explicitly with the <livestream> tag in the XML style parameter 3 of event 3.

i.e. <livestream>yes</livestream>

<LENGTH>

Live streams fall into one of the following classifications:-

a) **Known start time, planned for duration:** An example of this would be a live sporting event where the start point is known and an expected end time (i.e. known duration). With live events the end time may over run (i.e. baseball game) so Video Analytics will take this into account if it sees a duration that is longer than the initially declared duration in <length>

So for example, a two hour baseball game would have a parameter 3, event 3 <length> of 7200

b) **Known start time, unknown duration:** Although not very common, some publishers may have technology that does not support the specification of expected live event duration. In these cases, in alignment with Video Census the duration should be set at 4 hours.

So for example, parameter 3, event 3 <length> would be set to 14400.

c) **Unknown start time, unknown duration:** Also known as rolling 24/7 feeds. In these cases, the parameter 3, event 3 <length> should be set to 86400.

Note: often, upon further exploration, these 24/7 feeds are usually made up of segments that are of expected defined length and duration. As such, you are encouraged to treat each segment of the feed as type (a) above.

2) **Indicating play head position in Event 5 (initial play start)**

Market Intelligence Note: *If the beacon is going to be used to send video stream data to Market Intelligence then you may ignore the below and set parameter 1 of event 5 to zero.*

For live streams that have a defined start time, the play head position supplied in the event 5 should represent the offset time (in seconds) that the user joined the live stream.

So for instance, if a publisher started playing the live stream at 1pm and the user joins the broadcast at 1.30pm, the offset in seconds would be 1800 (30m x 60s = 1800).

```
gg1.ggPM(5, 1800);
```

If the user has activated the video player before the broadcast has begun and is waiting for the broadcast to begin then the value upon stream commencement will be zero.

For live streams that do not have a specific start time, the play head offset should be calculated as an offset from midnight.

3) **Indicating play head position in Event 5 (resume play start after a pause)**

Market Intelligence Note: *If the beacon is going to be used to send video stream data to Market Intelligence then you may ignore the below and set parameter 1 of event 5 to zero.*

Not many live stream video players allow resume from pause (they will have a stop button, so subsequently clicking play again should be treated as a new video stream) but for those that do, the play head position supplied in the event 5 should represent the offset time (in seconds) that the user has resumed the live stream.

So for instance, if a publisher started playing the live stream at 1pm and the user joins the broadcast at 1.30pm, subsequently pauses the stream at 2pm and then resumes play at 2.30pm (90 minutes into the broadcast), the offset in seconds would be 7200 (90m x 60s = 7200).

```
gg1.ggPM(5, 7200);
```

For live streams that do not have a specific start time, the play head offset should be calculated as an offset from midnight.

4) **Indicating play in progress in Event 49**

Market Intelligence Note: *If the beacon is going to be used to send video stream data to Market Intelligence then you should set the parameter 1 of event 49 in the same way as you would for Video on Demand streams. As such you may ignore the below.*

As noted previously in this section on live streaming, it is possible for the stream to be long in duration and potentially open ended with a high chance of never sending an event 7. The stream will effectively be terminated via URL change or browser close detection.

As such, accurate duration requires that an event 49 be sent on a regular basis. It is recommended to generate an event 49 every 15 seconds. Parameter 1 of the event 49 should include any offset provided in event 5.

So for instance, if a publisher started playing the live stream at 1pm and the user joins the broadcast at 1.30pm, the offset in seconds starting 15 seconds into the viewing session would be 1815 (30m15s x 60s = 1815). The subsequent Event 49 sequence every 15 seconds would be:-

```
gg1.ggPM(49, 1815);
gg1.ggPM(49, 1830);
gg1.ggPM(49, 1845);
```

5) **Indicating pause in Event 6**

Market Intelligence Note: *If the beacon is going to be used to send video stream data to Market Intelligence then you should set the parameter 1 of event 6 in the same way as you would for Video on Demand streams. As such you may ignore the below.*

The value in parameter 1 of event 6 should be calculated relative to the offset value supplied in parameter 1 of event 5.

So for instance, if a publisher started playing the live stream at 1pm and the user joins the broadcast at 1.30pm, and then paused at 2pm, the offset in seconds would be 3600 (60m x 60s = 3600). The subsequent Event 6 would be:-

```
gg1.ggPM(6, 3600);
```

6) **Indicating play end in Event 7**

Market Intelligence Note: *If the beacon is going to be used to send video stream data to Market Intelligence then you should set the parameter 1 of event 6 in the same way as you would for Video on Demand streams. As such you may ignore the below.*

The value in parameter 1 of event 7 should be calculated relative to the value supplied in parameter 1 of event 5.

So for instance, if a publisher started playing the live stream at 1pm and watched it through to completion at 3.30pm then the play head position for event 7 would be 12600 (3h 30m * 60s = 12600).

```
gg1.ggPM(7, 12600);
```

If the event over ran by 30 minutes to 4pm then the value would be 14400 (4h * 60s = 14400).

```
gg1.ggPM(7, 14400);
```

9.2.7 Meta Data: Xtag Custom Categorization

Similar to <category> and <subcategory> but not restricted to the two level limit that <category> and <subcategory> impose, the Xtag is an unlimited mechanism for grouping videos together within a hierarchy to provide aggregated numbers for attentiveness, average percent watched, average time spent viewing and stream view. Xtags are NOT linked to <category> and<subcategory> other than implicitly through the video <title>.

Syntax: **<xtag>{tagvalue1}||{tagvalue2}||{tagvalue3}||…</xtag>**

Example: **<xtag>Sport||Football||EuroCup||Wembley</xtag>**

Note: The current version Xtag functionality requires that the hierarchical relationships between tag values be set up in advance by the Nielsen Operations department. It is recommended that tag values and the inter-relationships be defined and recorded using a spreadsheet with your Nielsen client services representative.

9.2.8 API: Periodic Play / Ping Update

In order to provide accurate Video Analytic metrics, it is strongly advised to generate a "periodic play" event every 5 seconds for Ad's ("preroll", "midroll", and "postroll") and every 15 seconds for "content". This does not result in any additional network traffic as the periodic plays are cached and subsequently flushed to the server upon receiving the video stop/end event.

Here is an example where an event 49 is generated every 15 seconds.

```
gg1.ggPM(49, 15);
gg1.ggPM(49, 30);
gg1.ggPM(49, 45);
```

9.2.9 Meta Data: Populating the Page URL and Referring URL

The page URL is the web page URL that the video player (if embedded in a web page) executes on.

The referring URL is the web page URL that the user was referred from before landing on the publishers site.

Upon generating an event 15 or event 3, an additional event is automatically generated that captures the page URL and referring domain URL. The beacon executes a JavaScript call to query window.location.href for the current page URL and document.referrer for the referring URL.

You may decide that you want to override page URL or referring URL before it is passed into Video Analytics. To do so you should set parameter 1 of event 51 to the desired *page URL* and parameter 2 to the desired *referring URL*. You should do this immediately before the event 3 or event 15 calls.

Note 1: when making an event 51 call, parameter 1 is mandatory, parameter 2 is optional.

Note 2: the page URL is also used as the value passed into the OU parameter for Market Intelligence. See section 12 for more detail on Market Intelligence and this API.

Below is an example.

```
ggCom.getInstance().PM (51,http://abclocal.go.com/wabc, "http://www.google.com");
ggCom.getInstance().PM(15, …);
```

9.2.10 Meta Data: URI Custom Variables

A number of clients request and pay for custom reporting that leverages customized data passed on the URI / HTTP Request at video play back start.

The following is generated by a video start on the Hulu video player: http://secure-us.imrworldwide.com/cgi-bin/m?ci=us-600346gg&c6=vc,c71&cc=1&cg=Monster%20Fish&c4=mn,1&c7=hg,4CDD13FDB31C7A660A6E57BE43522D02&c9=pv,hulu&c5=gn,38&tl=dav0-%5BNational%20Geographic%5DMonster%20Fish%20of%20America_1_6&rnd=4850&tp=gg

In the above, C4, C5, C7, and C9 are custom variables sent by Hulu. They are ignored by the core Video Census product.

Important Note: There are a number of reserved C*x* variables that are used by the core Video Census product. Duplication / corruption of these reserved variables will probably result in the URI being ignored completely by Video Census. **Reserved C*x* variables are C6, C3, CI, CG and CC**.

To populate a custom variable, use the following scheme in parameter 3 (XML style data) of event 3 or event 15:-

<nol_*cx*>{value}</nol_*cx*>

Please consult with your Nielsen Techncial Account Manager before using this custom variable feature.

9.2.11 Meta Data: Overlay Ads

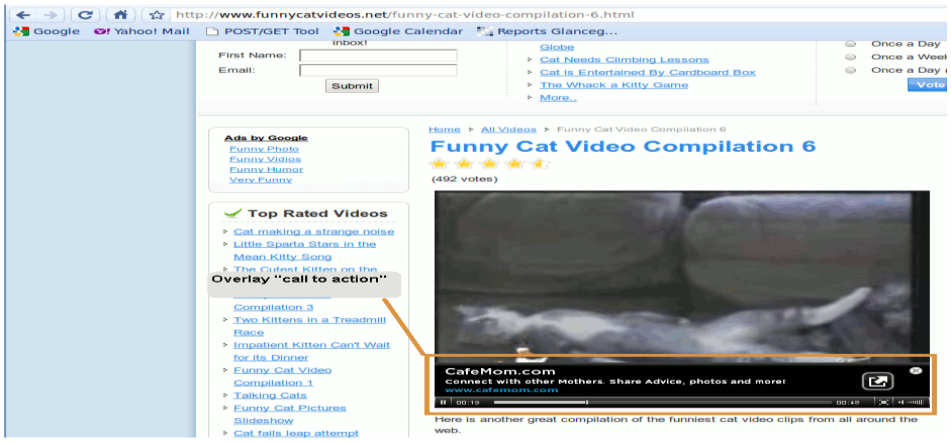
The combined beacon ONLY tracks video Ads that execute as a the result of an auto-play or user invoked play when the overlay "call to action" appears over the content video stream.

If you wish to track the overlay "call to action" itself then you should contact your IAG representative who will assist you with the installation of the IAG pixel tag.

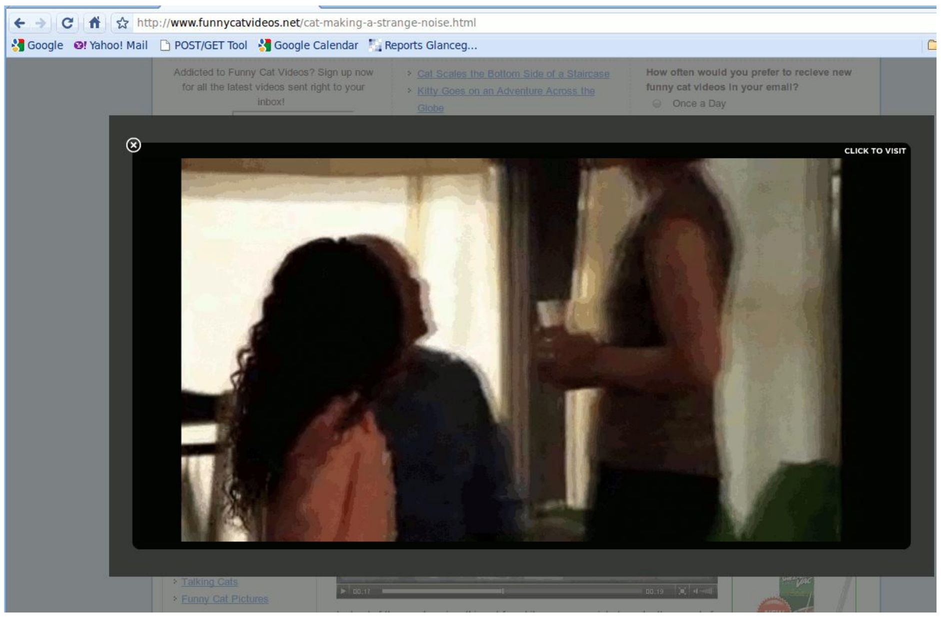
Note: At this time, the combined beacon API does not identify overlays Ads as a specific type (i.e. preroll, midroll, postroll). Until this status changes then you should indicate overlay video Ads as type "midroll".

Below is a typical scenario where the combined beacon is triggered to record the video play as a result of the user clicking on the initial overlay "call to action".

1. The content video play is started. The overlay "call to action" appears at the 5 second mark, and the user clicks on the overlay image at the 10 second mark…


http://www.funnycatvideos.net/funny-cat-video-compilation-6.html
Funny Cat Video Compilation 6
Top Rated Videos
Overlay "call to action"
CafeMom.com
Connect with other Mothers. Share Advice, photos and more!
www.cafemom.com

2. The action of the user clicking on the overlay "call to action" causes the play of the underlying content video to be paused. The video Ad is started…


http://www.funnycatvideos.net/cat-making-a-strange-noise.html
CLICK TO VISIT

3. Once the video Ad completes then the video Ad window closes and control is passed back to the publisher video player and the content resumes play.

The above scenario should be represented as the following API calls in the combined beacon:

1. Content video start - Event 15 or Event 3 + Event 5
2. Overlay "call to action" appears at 5 seconds - Event 13 (Optional)

{User clicks on overlay at 10 second mark}

3. Content video pause - Event 6
4. Video Ad Start (indictated as type "midroll") - Event 15 or Event 3 + Event 5
5. Video Ad end - Event 7
6. Video Ad unload - Event 4 (indictated as type "midroll")
7. Content video resumes Play - Event 5 (at same time marker indicated when content paused)

{Video progresses without further user interruption}

8. Overlay "call to action" disappears from view - Event 14 (Optional)
9. Content video end - Event 7
10. Content video unload - Event 4 (with video type indicator of "content")

10 VideoCensus Support

Please refer to VideoCensus Tagging Implementation Guide from Nielsen for a definition of these parameters.

Variable called prod in nolggGlobalParams must be set to "vc" to enable VideoCensus tagging. This variable can also be simply omitted in nolggGlobalParams, in which case it will default to "vc".

10.1 Clientid and Vcid

Clientid ("ci" parameter) and vcid ("c6" parameter) are specified in _nolggGlobalParams. "c3" (Stream Type) and "tl" (Title) are automatically picked up from parameters to the LoadVideo or LoadandplayVideo API calls.

If a <title> is not supplied, the URL of the flash file will be used for the "tl" parameter.

10.2 Census Category

<censuscategory> specifies whether a play should be counted and surfaced under "client-defined category" in VideoCensus syndicated reporting (the "cg" parameter). Often, this refers to a Show (e.g. The Simpsons).

The <category> value in Video Analytics serves the same purpose. You can specify either or both in *videoinfo*.

If <censuscategory> is specified, but <category> is not, <category> value defaults to that of <censuscategory>. The converse is not true; <censuscategory> must be explicitly specified or it is null.

<censuscategory> needs to be specified in *videoinfo* as opposed to nolggGlobalParams because it applies to a single video play and can be different for different videos.

See section *Metadata Handling* for *videoinfo* code examples.

11 SiteCensus Support

Please refer to SiteCensus Tagging Implementation Guide from Nielsen for a definition of these parameters. This document refers only to the streaming portion of SiteCensus. The rest of SiteCensus tagging remains unchanged.

ci, tl and c3 parameters are handled exactly like VideoCensus. <censuscategory>, if specified, this works exactly the same way as in VideoCensus above, but becomes the "pg" parameter.

Variable called prod in nolggGlobalParams must be set to "sc" to enable data collection for SiteCensus streaming.

12 Market Intelligence Support

Tagging for the Nielsen MarketIntelligence service is same as SiteCensus tagging. One additional requirement is the Ownership URL (ou) parameter. For Market Intelligence this is set to the domain owner.

Configuration prerequisite: Variable called **prod** in nolggGlobalParams must be set to "sc" to enable the below rules and the correct build of the OU.

Setting the OU: Combined beacon generates an ownership URL by default based on the rules laid out below. In order of precedence, the value of the ou parameter is decided like this:

a) If the URL is set explicitly in the API (Event 51), that will be the value. Call to event 51 must be made before the start of the first video. i.e.

```
gg1.ggPM(51,"http://abclocal.go.com/wabc");
gg1.ggPM (15, …);
```

b) Otherwise, if Javascript access is available to the plugin, it will be the actual URL of the containing HTML page where the video is playing (using window.location.href)

c) Otherwise, it will be the URL of the JS file (e.g. http://mysite.com/player.js).

For example, see the OU in this GET generated by the combined beacon for an existing client:

http://secure-us.imrworldwide.com/cgi-bin/m?ci=us-801075&c6=vc,b01&cc=1&**ou=%3A%2F%2Fabclocal%2Ego%2Ecom%2Fwabc**&c3=st,a&tp=gg&tl=dav0%2Dhttp%3A%2F%2Fwdig%2Evo%2Ellnwd%2Enet%2Fo2%2Fcngabcfamily%2Fads%2Famericanlicorice...

Periodic PlayHead Position Update: Market Intelligence uses a high-water marking methodology for computing duration. This value is passed in the DU variable when a dav1 or dav2 is produced. To ensure a reasonably accurate play head position is available to the beacon, you must specify an event 49 periodically. For MI purposes we would recommend generating an event 49 every 15 seconds.

```
gg1.ggPM(49, 15);
gg1.ggPM(49, 30);
gg1.ggPM(49, 45);
```

MI Configuration: You will also need to do the following in Market Intelligence (MI):

1) Link the client id (ci/cid) to a domain in the MI portal.
2) Make sure the OU patterns in MI match what is being passed in the OU value from the combined beacon.

Contact your Nielsen Technical Account Manager to make sure Market Intelligence is configured correctly.

13 Nielsen IAG Support

Combined Beacon can also automatically generate parameters needed for the Nielsen IAG service. To enable IAG, the following variables must be set in nolggGlobalParams structure:

- **prod** variable in nolggGlobalParams to “iag”. If both VideoCensus and IAG are needed, set the **prod** to “vc,iag” in ggCom.as (Action Script 3) / ggBase.as (Action Script 2)
- **sid** variable should be set to Source Id supplied to your company by Nielsen IAG.
- **tfid** variable should be set the Tag Format Id supplied to your company by Nielsen IAG.

13.1 VideoInfo parameters for IAG

The majority of IAG parameters that appear on the request URL are defaulted in from existing data collected by the beacon when the video meta-data is supplied.

Direct specification of IAG data that cannot be derived are all specified in the VideoInfo (param3) of LoadVideo (event type 3 – appendix a) or LoadAndPlayVideo (event type 15 – appendix a) call to the combined beacon.

Example event type 15:

```
gg1.ggPM(15,http://cust.com/videos/content1.flv,”content”,”<uurl>http://cust.com?w=hhd766dbFF
</uurl><length>300</length><censuscategory>Bones</censuscategory><category>Drama</cate
gory><subcategory>Suspense</subcategory><title>The
Knife</title><iagcategory>Mystery</iagcategory><IAG_epi>The
Knife_S2_E1</IAG_epi><pd>Fox.com</pd><IAG_seg>2</IAG_seg><IAG_fp>fp</iag_fp>”,1);
```

Some clients may have custom engagements with IAG and there might be slightly different data collection requirements. You should confirm these custom requirements with your IAG representative. For example the creative URL (cte) might not be accessible through the URL supplied, or the Program indicator (pgm) required might be different from the standard value. For these cases, with prior agreement from the IAG client services team, an alternative value might suffice. These exceptions can be passed through “override” meta-data parameters. Available override parameters are discussed below, and indicated in appendix C against the corresponding standard parameter.

Your IAG representative can guide you through those items below that are required for your particular IAG engagement.

Important Note: the meta-data tags are case-sensitive and should be specified in lower-case I.e. <iag_pgm> is not the same as <IAG_pgm>

IAG Category (Program Name)

<iagcategory> is specified in the <videoinfo> string.

If <iagcategory> is specified in event 3 or event 15, it will populate the value of the **IAG pgm** parameter in the request URL. Otherwise, if <category> has been provided then that will become the **IAG pgm** parameter in the request URL.

Example: **<iagcategory>***category***</iagcategory>**

Long-form Content Notes:

a) IAG category is optional but strongly encouraged to be set to the show name if long form content.
b) For long-form content (pre-roll / multi-chapter / mid-roll) the **IAG pgm** needs to be explicitly set for the Ad’s. This is done using the **<iag_pgm>** over-ride. This should be set to the same value as **<iagcategory>** in the content chapter that immediately follows the Ad.
c) <iag_pgm> is ignored for content.

IAG BCR (Broadcaster)

By default this value will be set to the **clientid** sent into the beacon.

You may override this automatic mapping by directly specifying the brand in **<iag_bcr>***broadcaster***</iag_bcr>**

IAG BRN (Brand Name)

By default this value will be set to the **clientid** sent into the beacon.

You may override this automatic mapping by directly specifying the brand in **<iag_brn>***brand***</iag_brn>**

IAG EPI (Episode Title)

Value in **<title>** will be defaulted into the **IAG epi** parameter in the request URL.

You may override this automatic mapping by directly specifying the Episode Title in **<iag_epi>***video_title***</iag_epi>**

Important Note: if the series and episode information is available (but not already included in the standard video_title information) then IAG would strongly urge that these details are passed into epi along with the title information using the **<iag_epi>** override variable for both the content and Ad streams. The value supplied in the Ad call should be the same as that supplied for the content chapter that immediately follows the Ad.

IAG OAD (Original Air Date)

<oad> is specified in the <videoinfo> string and becomes the **IAG oad** parameter in the request URL.

Example: **<oad>***05/06/2010***</oad>**

You may override this automatic mapping for Ads by directly specifying the OAD in **<IAG_oad>***05/06/2010***</IAG_oad>**

Important Note: oad is optional. If <oad> is not supplied then the <title> needs to uniquely identify the Name, Season, and Episode when tracking long-form content. If <oad> is known then it should be supplied for both the content chapter and the Ad's that execute immediately before hand.

IAG PD (Partner Distribution)

<pd> can be specified in <videoinfo> and becomes the **IAG pd** parameter in the request URL.

Example: **<pd>***fox.com***</pd>**

You may override this automatic mapping for Ads by directly specifying the PD in **<IAG_pd>***05/06/2010***</IAG_pd>**

Important Note: pd is optional but strongly recommended. If <pd> is known then it should be supplied for both the content chapter and the Ad's that execute immediately before hand. In the Ad call it should be specified as **<iag_pd>**

IAG CTE (Creative)

Defaulted to the path/filename of the video file being tracked. This is captured by the beacon in parameter one of event 3 or event 15 and is used to populate the **IAG cte** parameter in the request URL.

You may over ride the default value by specifying the creative reference in **<iag_cte></IAG_cte>**.

<iag_cte> is ignored when called for content.

Example: **<iag_cte>**http%3A//media.cwtv.com/cwtv/Prime/Season/0809/Ads/hp30-crosscountrybaby-102510.flv**</iag_cte>**

Important Note: It is essential to supply a resolvable reference ID for the media file of the advertisement so it maybe viewed. This is typically a URL. This creative ID or creative URL must be unique across all content and ads in your system.

IAG SEG (Segment/Chapter)

Defaulted to the chapter number of the content that preceeded the Ad. Therefore the default is only applicable for mid-roll Ads and post-roll Ads. Becomes the **IAG seg** parameter in the request URL.

You may over ride the default value by specifying the chapter/segment in **<iag_seg></iag_seg>**. This is typically used to specify the chapter/segment that comes immediately after the Ad. i.e. pre-roll Ads.

Example: **<iag_seg>***2***</iag_seg>**

Important Note: seg is optional but strongly recommended for long-form content.

IAG FP (Form)

<iag_fp> can be specificed in <videoinfo> and becomes the **IAG fp** parameter in the request URL.
Valid values are “sf” (Short Form) or “lf” (Long Form)

Example: **<iag_fp>***lf***</iag_fp>**

Important Note: fp is optional but strongly encouraged.

IAG POD (Ad Pod)

<iag_pod> can be specified in <videoinfo> and becomes the **IAG pod** parameter in the request URL.

Important Note: pod is optional. If supplied then it should have four numeric integer parameters delimited using the “_” character. If the information is not known for a given parameter then the value **0** (Zero) should be supplied. The functional decomposition of each position is:-

{numpods}_{podnumber}_{podadcount}_{placementwithinpod}

Example 1: **<iag_pod>***5_2_1_1***</iag_pod>**

Example 2 (no placement known): **<iag_pod>***5_2_1_0***</iag_pod>**

IAG APT (Ad Type)

<iag_apt> can be specified in <videoinfo> and becomes the **IAG apt** parameter in the request URL.

Example: **<iag_apt>***be***</iag_apt>**

Important Note: apt is optional. If supplied then it should be one of the following four values; **ol** (overlay), **be** (branded entertainment), **as** (ad selector), **na** (regular video ad)

IAG CUST1 (Custom Field)

<iag_cust1> can be specified in <videoinfo> and becomes the **IAG cust1** parameter in the request URL.

Example: **<iag_cust1>***Some custom data***</iag_cust1>**

Important Note: cust1 is optional. Used for bespoke data to be sent to IAG.

14 Beacon Test/QA Checklist

This section details what you should observe in the browser when the publisher video player triggers the beacon. This enables you to trap and correct basic instrumentation errors at source rather than trying to reverse engineer from metrics that surface in the reporting UI. Because many individual events in the video player are used to compute a basic metric in the reporting UI it can often be difficult, if not impossible, to diagnose an incorrectly instrumented video player from the reporting UI.

The section below summarises the basic measurement rules that have to be followed for different content and Ad types. You will then find two checklists that you should complete when using the test tool.

14.1 CHECKLIST 1: Basic Video Start Event and Stop Event

Note: Two tables are detailed below. First is for Flash players, second is for Javascript instrumented players. You should pick the appropriate table for the video player type that has been implemented.

Flash Players		
Step	**Pass/Fail**	**Test Step**
1		A POST with 'dav0' and the following parameters is generated upon or shortly after a video play is initiated • tp=gg • tl=dav0%2DSome%20Title • ci={clientid} • c6={prod},{vcid} • cc=1 • ou={url} • sd={Number}
2		A POST with 'dav2' and the following parameters is generated upon or shortly after a video stop is initiated, or the video completes

		• tp=gg • tl=dav0%2DSome%20Title • ci={clientid} • c6={prod},{vcid} • cc=1 • ou={url} • du={Number}
3		Upon decrypting the HEX40 in the 'dav0' POST the following are observed for event type 3 (Load Video) **OR** event 15 (load and play video) **Parameter 1** Mandatory: • valid, playable video filename in parameter 1 of event 3 or event 15. If the video filename is not directly playable then it must, at a minimum, uniquely identify the video asset. The customer should also be made aware that if the video is not directly playable from the value supplied then it will not be playable in the SynchPlayer. **Parameter 2** Mandatory: • the value "content", "preroll", "midroll", or "postroll" in parameter 2,depending on the video type executing **Parameter 3** Mandatory: • <length> attribute that matches the length of the video indicated in the video player • <title> attribute that matches that of the video being tested, or if not <title> then a valid <uurl> that refers to the source video file • <censuscategory> for full episode players. Should be populated with the show name. Optional but desirable: • <category>, <subcategory> Optional • <vidtype>, <imgurl> **Parameter 4** Mandatory for videos of type "content": • An increment number indicating the current chapter. Default is "1" Note: For IAG implementations (i.e. the configuration will have "iag" specified in the **prod** variable in ggCom.as or ggBase.as) please see separate documentation.
4		Upon decrypting the HEX40 in the 'dav0' POST the following is observed for event

		type 5 (Play Video) • Parameter 2 contains a valid numeric indicating play start position. Typically this will be zero (0) **Note: if event 15 was specified then you will not see a distinct event 5**
5		Upon decrypting the HEX40 in the 'dav2' POST the following are observed for event type 7 (Stop Video) • Parameter 2 contains a valid numeric indicating the play stop position. If this is the end of the video then it should be reasonable close to the end time position indicated in the player UI.

Javascript Players (includes WMP and Silverlight)		
Step	**Pass/Fail**	**Test Step**
1		A GET with 'dav0' and the following parameters is generated upon or shortly after a video play is initiated • tp=gg • tl=dav0%2DSome%20Title • ci={clientid} • c6={prod},{vcid} • cc=1 • ou={url} • sd={Number}
2		A GET with 'dav2' and the following parameters is generated upon or shortly after a video stop is initiated, or the video completes • tp=gg • tl=dav0%2DSome%20Title • ci={clientid} • c6={prod},{vcid} • cc=1 • ou={url} • du={Number}
3		Upon decrypting the HEX40 in the 'dav0' GET the following are observed for event type 3 (Load Video) **OR** event 15 (load and play video) **Parameter 1** Mandatory: • valid, playable video filename in parameter 1 of event 3 or event 15. If the video filename is not directly playable then it must, at a minimum, uniquely identify the video asset. The customer should also be made aware that if the video is not directly playable from the value supplied then it will not be

		playable in the SynchPlayer. **Parameter 2** Mandatory: • the value “content”, “preroll”, “midroll”, or “postroll” in parameter 2,depending on the video type executing **Parameter 3** Mandatory: • <length> attribute that matches the length of the video indicated in the video player • <title> attribute that matches that of the video being tested, or if not <title> then a valid <uurl> that refers to the source video file • <censuscategory> for full episode players. Should be populated with the show name. Optional but desirable: • <category>, <subcategory> Optional • <vidtype>, <imgurl> **Parameter 4** Mandatory for videos of type “content”: • An increment number indicating the current chapter. Default is “1” Note: For IAG implementations (i.e. the configuration will have “iag” specified in the **prod** variable in ggCom.as or ggBase.as) please see separate documentation.
4		Upon decrypting the HEX40 in the ‘dav0’ GET the following is observed for event type 5 (Play Video) • Parameter 2 contains a valid numeric indicating play start position. Typically this will be zero (0) **Note: if event 15 was specified then you will not see a distinct event 5**
5		Upon decrypting the HEX40 in the ‘dav2’ GET the following are observed for event type 7 (Stop Video) • Parameter 2 contains a valid numeric indicating the play stop position. If this is the end of the video then it should be reasonable close to the end time position indicated in the player UI.

14.2 CHECKLIST 2: Additional Video Player Events

All Player Types		
Step	**Pass/Fail**	**Test Step**
1		Player Pause – an event type 6 should be generated with parameter 2 set to the time position as indicated in the video player when the pause control was clicked

		Note: if the user clicks on Play again then step 2 below must also be validated.
2		Play Resume after Pause – an event type 5 should be generated with parameter 2 set to the time position as indicated in the video player when the play button was clicked after previously pausing
3		Player Seek Forward – an event type 8 should be generated with parameter 1 set to the time position as indicated when the seek control was activated, and parameter 2 set to the time position as indicated when the seek control was released at the new position forwards of the current time position
4		Player Seek Backwards – an event type 8 should be generated with parameter 1 set to the time position as indicated when the seek control was activated, and parameter 2 set to the time position as indicated when the seek control was released at the new position backwards of the current time position
5		Player Mute – an event type 9 should be generated with value of ‘1’ if mute is turned on, value of ‘0’ if mute is turned off.
6		Player Volume Up/Down – an event type 11 should be generated with a numeric value between 1 and 100 indicating the new volume setting

14.3 Using the Test Tool

To aid the process of validating that the beacon is correctly integrated with the video player, an online tool is provided that decrypts the HEX part of the GET / POST data section.

Using a tool such as HTTPFox for Firefox, you can filter for calls to the Nielsen data collection server at http://secure-XX.imrworldwide.com (where XX is the appropriate country code such as “us” for United States. If you do not know your local collection node then please ask your Nielsen representative)

14.3.1 Identifying the GET / POST Requests

In the example below, this graphic shows a video in flight with calls filtered on the string “imrworldwide”. There are a number of products that use the IMRWorldwide collection server and depending on the services you have implemented you may see numerous GET or POST calls.

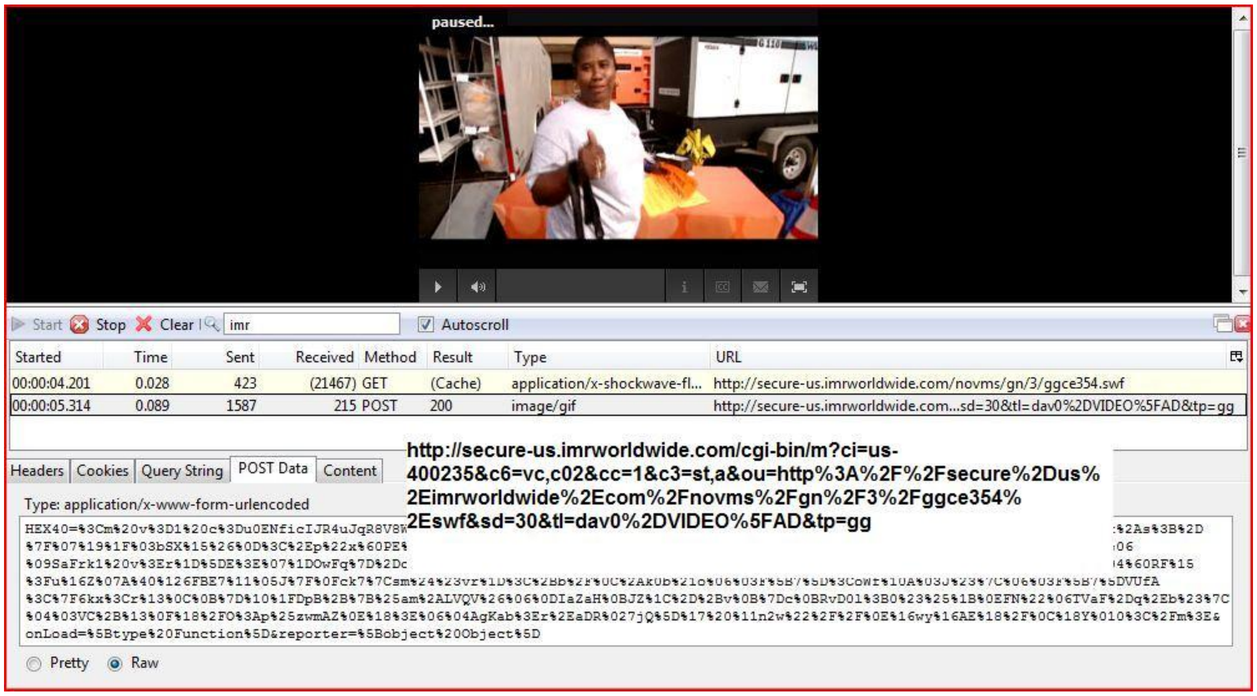
For the purposes of this beacon testing you are looking for GET or POST calls that have the parameter **“tp=gg”** included on the parameter line and of type **dav0, dav1** or **dav2**.

Dav0 – Generated upon the beacon receiving the API event 3 followed immediately by an event 5 **OR** an event 15.

Dav1 – Intermediate POST of messages in the beacon buffer. For long play video you may see a dav1 generated if the buffer reaches a predetermined limit.

Dav2 – Generated upon the beacon receiving the API event 7.

For each of the above, there will be some accompanying parameter data. This is compressed for the purposes of minimizing memory use and speed of transmission to the Nielsen server. The rest of this section shows examples of **dav0** and **dav2**. The next section will detail how this data can be uncompressed and examined.



Started	Time	Sent	Received	Method	Result	Type	URL
00:00:04.201	0.028	423	(21467)	GET	(Cache)	application/x-shockwave-fl...	http://secure-us.imrworldwide.com/novms/gn/3/ggce354.swf
00:00:05.314	0.089	1587	215	POST	200	image/gif	http://secure-us.imrworldwide.com...sd=30&tl=dav0%2DVIDEO%5FAD&tp=gg

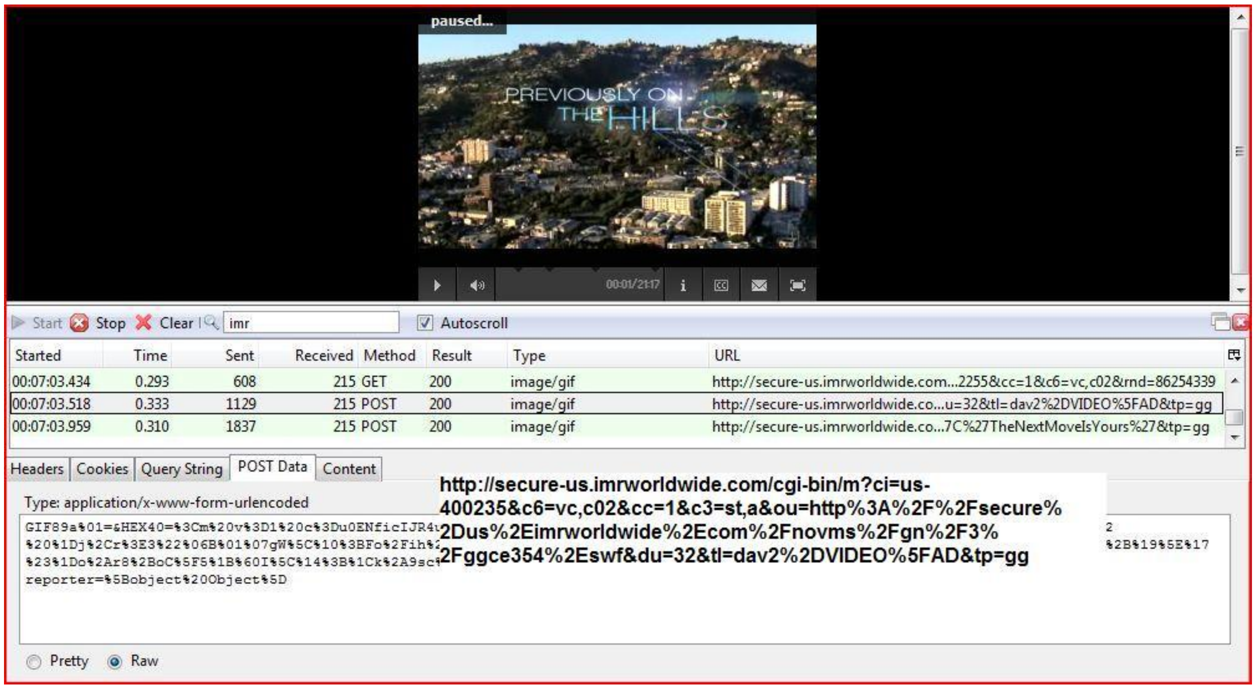
http://secure-us.imrworldwide.com/cgi-bin/m?ci=us-400235&c6=vc,c02&cc=1&c3=st,a&ou=http%3A%2F%2Fsecure%2Dus%2Eimrworldwide%2Ecom%2Fnovms%2Fgn%2F3%2Fggce354%2Eswf&sd=30&tl=dav0%2DVIDEO%5FAD&tp=gg

Type: application/x-www-form-urlencoded

```
HEX40=%3Cm%20v%3D1%20c%3Du0ENficIJR4uJqR8V8W                                   :%2As%3B%2D
%7F%07%19%1F%03bSX%15%26%0D%3C%2Ep%22x%60PE%                                   :06
%09SaFrk1%20v%3Er%1D%5DE%3E%07%1DOwFq%7D%2Dc                                   )4%60RF%15
%3Fu%16Z%07A%40%126FBE7%11%05J%7F%0Fck7%7Csm%24%23vr%1D%3C%2Bb%2F%0C%2Ak0b%21o%06%03F%5B7%5D%3CoWf%10A%03J%23%7C%06%03F%5B7%5DVUfA
%3C%7F6kx%3Cr%13%0C%0B%7D%10%1FDpB%2B%7B%25am%2ALVQV%26%06%0DIaZaH%0BJZ%1C%2D%2Bv%0B%7Dc%0BRvD01%3B0%23%25%1B%0EFN%22%06TVaF%2Dq%2Eb%23%7C
%04%03VC%2B%13%0F%18%2FO%3Ap%25zwmAZ%0E%18%3E%06%04AgKab%3Er%2EaDR%027jQ%5D%17%20%11n2w%22%2F%2F%0E%16wy%16AE%18%2F%0C%18Y%010%3C%2Fm%3E&
onLoad=%5Btype%20Function%5D&reporter=%5Bobject%20Object%5D
```

The above example is a POST of type dav0 for a preroll Ad. It is distinguishable by the parameter that says “c3=st,a”. This dav0 gets executed when the player sends an event 5 (see next section for a functional decomposition of basic event types that you need to test for) indicating video start.

When the video ends and an event 7 is sent then you will see a POST of type dav2. Again it has the parameter of “c3=st,a”. See highlighted below which shows the video end for the preroll Ad.



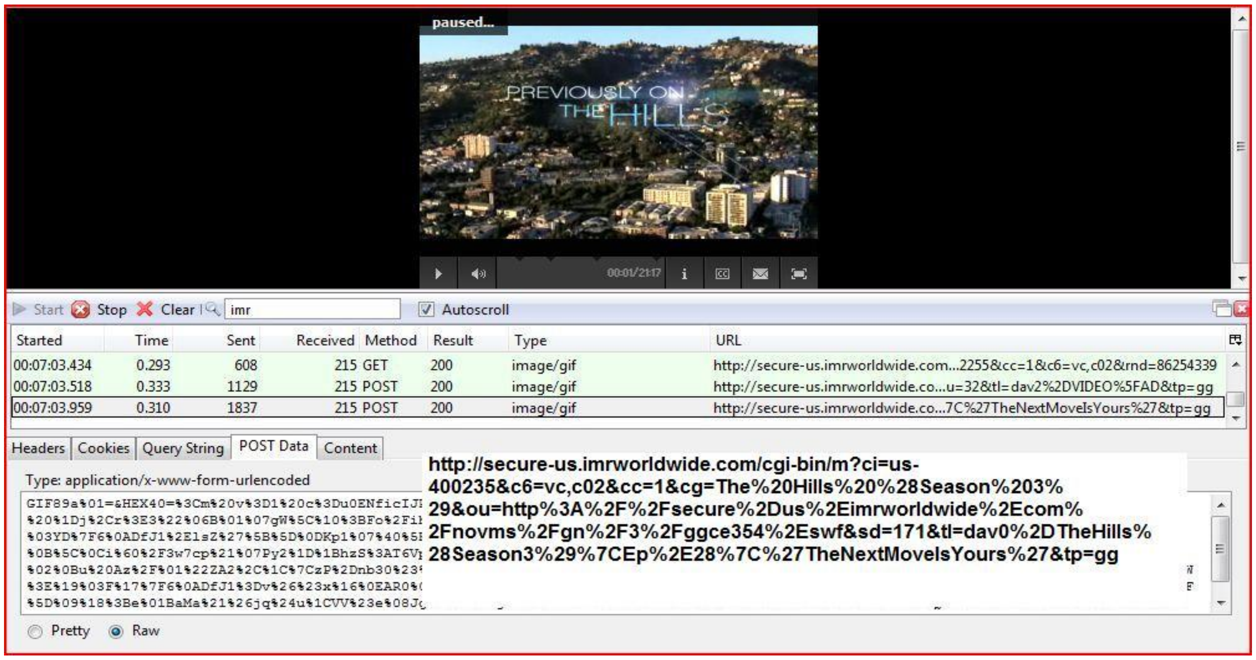
Started	Time	Sent	Received	Method	Result	Type	URL
00:07:03.434	0.293	608	215	GET	200	image/gif	http://secure-us.imrworldwide.com...2255&cc=1&c6=vc,c02&rnd=86254339
00:07:03.518	0.333	1129	215	POST	200	image/gif	http://secure-us.imrworldwide.co...u=32&tl=dav2%2DVIDEO%5FAD&tp=gg
00:07:03.959	0.310	1837	215	POST	200	image/gif	http://secure-us.imrworldwide.co...7C%27TheNextMoveIsYours%27&tp=gg

http://secure-us.imrworldwide.com/cgi-bin/m?ci=us-400235&c6=vc,c02&cc=1&c3=st,a&ou=http%3A%2F%2Fsecure%2Dus%2Eimrworldwide%2Ecom%2Fnovms%2Fgn%2F3%2Fggce354%2Eswf&du=32&tl=dav2%2DVIDEO%5FAD&tp=gg

Type: application/x-www-form-urlencoded

```
GIF89a%01=&HEX40=%3Cm%20v%3D1%20c%3Du0ENficIJR4                                 2
%20%1Dj%2Cr%3E3%22%06B%01%07gW%5C%10%3BFo%2Fih%2                                %2B%19%5E%17
%23%1Do%2Ar8%2BoC%5F5%1B%60I%5C%14%3B%1Ck%2A9sc%
reporter=%5Bobject%20Object%5D
```

The next highlighted entry (see next screen shot) is the POST of type dav0 which is the video start for Chapter 1 of the content. You will notice that it does NOT have the parameter “cs=st,a” which indicates that it is a video of type content.



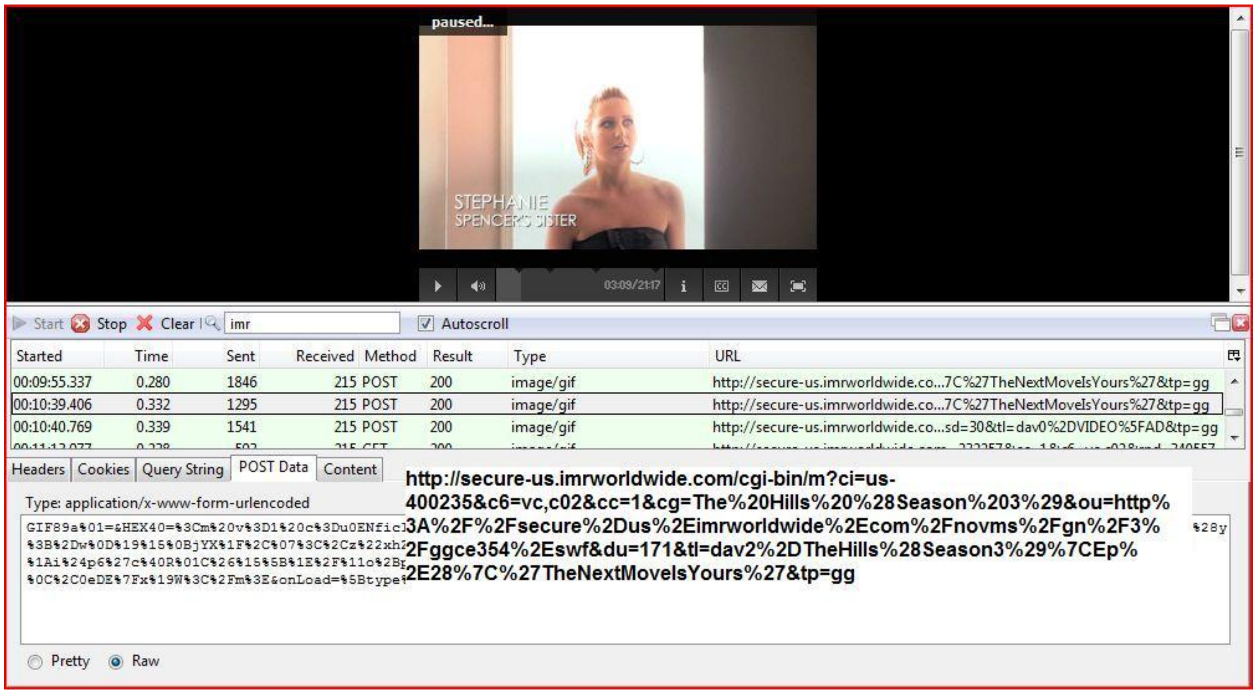
Started	Time	Sent	Received	Method	Result	Type	URL
00:07:03.434	0.293	608	215	GET	200	image/gif	http://secure-us.imrworldwide.com...2255&cc=1&c6=vc,c02&rnd=86254339
00:07:03.518	0.333	1129	215	POST	200	image/gif	http://secure-us.imrworldwide.co...u=32&tl=dav2%2DVIDEO%5FAD&tp=gg
00:07:03.959	0.310	1837	215	POST	200	image/gif	http://secure-us.imrworldwide.co...7C%27TheNextMoveIsYours%27&tp=gg

Type: application/x-www-form-urlencoded

```
GIF89a%01=&HEX40=%3Cm%20v%3D1%20c%3Du0ENficIJ
%20%1Dj%2Cr%3E3%22%06B%01%07gW%5C%10%3BFo%2Fil
%03YD%7F6%0ADfJ1%2E1sZ%27%5B%5D%0DKp1%07%40%5I
%0B%5C%0Ci%60%2F3w7cp%21%07Py2%1D%1BhzS%3AT6V
%02%0Bu%20Az%2F%01%22ZA2%2C%1C%7CzP%2Dnb30%23
%3E%19%03F%17%7F6%0ADfJ1%3Dv%26%23x%16%0EAR0%
%5D%09%18%3Be%01BaMa%21%26jq%24u%1CVV%23e%08Jq
```

http://secure-us.imrworldwide.com/cgi-bin/m?ci=us-400235&c6=vc,c02&cc=1&cg=The%20Hills%20%28Season%203%29&ou=http%3A%2F%2Fsecure%2Dus%2Eimrworldwide%2Ecom%2Fnovms%2Fgn%2F3%2Fggce354%2Eswf&sd=171&tl=dav0%2DTheHills%28Season3%29%7CEp%2E28%7C%27TheNextMoveIsYours%27&tp=gg

And the next highlighted entry is again the POST of type dav2 for the video stop indicating end of Chapter 1 of the content. Once again, you will notice that it does NOT have the parameter “cs=st,a” which indicates that it is a video of type content.



Started	Time	Sent	Received	Method	Result	Type	URL
00:09:55.337	0.280	1846	215	POST	200	image/gif	http://secure-us.imrworldwide.co...7C%27TheNextMoveIsYours%27&tp=gg
00:10:39.406	0.332	1295	215	POST	200	image/gif	http://secure-us.imrworldwide.co...7C%27TheNextMoveIsYours%27&tp=gg
00:10:40.769	0.339	1541	215	POST	200	image/gif	http://secure-us.imrworldwide.co...sd=30&tl=dav0%2DVIDEO%5FAD&tp=gg

Type: application/x-www-form-urlencoded

```
GIF89a%01=&HEX40=%3Cm%20v%3D1%20c%3Du0ENficI                                         %28y
%3B%2Dw%0D%19%15%0BjYX%1F%2C%07%3C%2Cz%22xh
%1Ai%24p6%27c%40R%01C%26%15%5B%1E%2F%11o%2B
%0C%2C0eDE%7Fx%19W%3C%2Fm%3E&onLoad=%5Btype
```

http://secure-us.imrworldwide.com/cgi-bin/m?ci=us-400235&c6=vc,c02&cc=1&cg=The%20Hills%20%28Season%203%29&ou=http%3A%2F%2Fsecure%2Dus%2Eimrworldwide%2Ecom%2Fnovms%2Fgn%2F3%2Fggce354%2Eswf&du=171&tl=dav2%2DTheHills%28Season3%29%7CEp%2E28%7C%27TheNextMoveIsYours%27&tp=gg

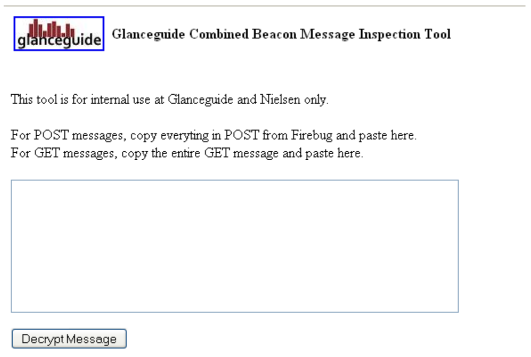
14.3.2 Decrypting the GET/POST Data (HEX40)

Once you have identified the GET or POST requests that you are interested in, you then need to validate that events contain the correct information. The next section outlines the business rules that should be observed for particular video player scenarios (full episode, clip, etc.) and the event data that you should be observing.

An example video of a QA test can be viewed here:
http://www.nielsenonlinesupport.com/clientsupport/va_test/va_test_tools.html

The rest of this section details the basic mechanism for decrypting the message and observing the contents. Basic validation can be done by simply observing the GET or POST parameters in the request. To validate the data for Video Analytics you will have to cut and paste the RAW contents of the GET or POST data into the decryptor tool.

The decryptor tool is hosted at http://l2.glanceguide.com/djunpack.html.

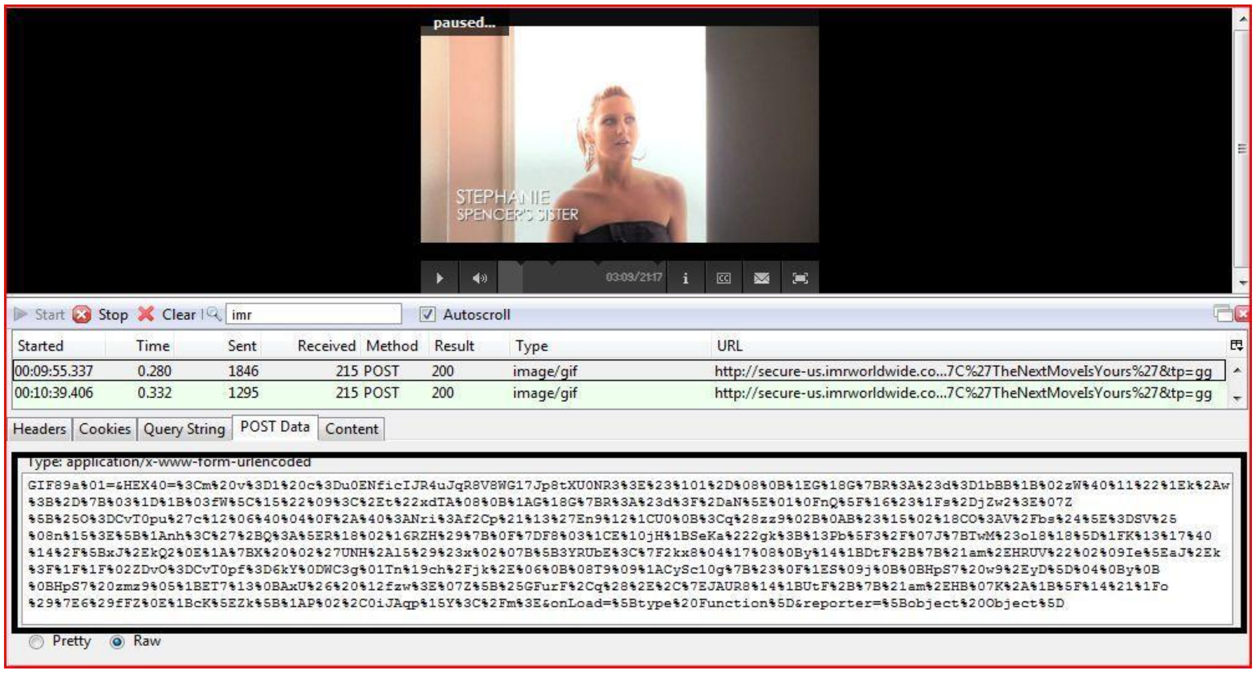

Glanceguide Combined Beacon Message Inspection Tool

This tool is for internal use at Glanceguide and Nielsen only.

For POST messages, copy everyting in POST from Firebug and paste here.
For GET messages, copy the entire GET message and paste here.

The rest of this chapter steps you through the process of grabbing the raw compressed data from HTTPFox and decrypting using the tool.

Step 1 – Grab the raw POST Data

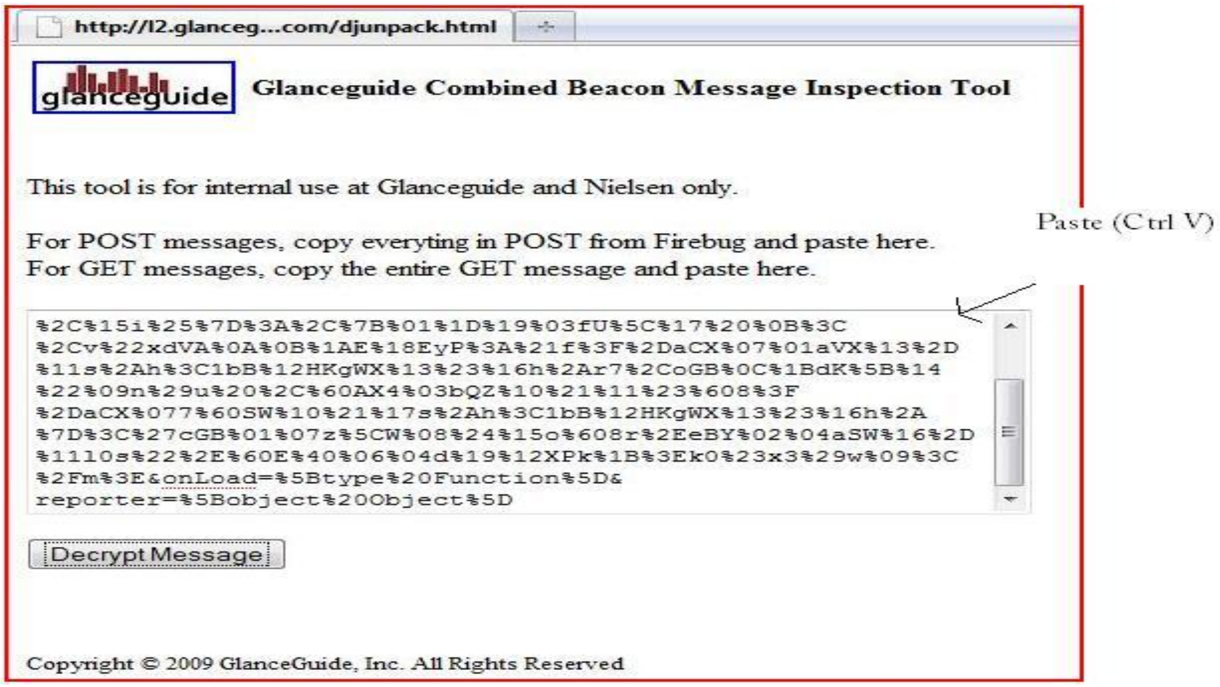


Started	Time	Sent	Received	Method	Result	Type	URL
00:09:55.337	0.280	1846	215	POST	200	image/gif	http://secure-us.imrworldwide.co...7C%27TheNextMoveIsYours%27&tp=gg
00:10:39.406	0.332	1295	215	POST	200	image/gif	http://secure-us.imrworldwide.co...7C%27TheNextMoveIsYours%27&tp=gg

Type: application/x-www-form-urlencoded

```
GIF89a%01=&HEX40=%3Cm%20v%3D1%20c%3Du0ENficIJR4uJqR8V8WG17Jp8tXU0NR3%3E%23%101%2D%08%0B%1EG%18G%7BR%3A%23d%3D1bBB%1B%02zW%40%11%22%1Ek%2Aw
%3B%2D%7B%03%1D%1B%03fW%5C%15%22%09%3C%2Et%22xdTA%08%0B%1AG%18G%7BR%3A%23d%3F%2DaN%5E%01%0FnQ%5F%16%23%1Fs%2DjZw2%3E%07Z
%5B%25O%3DCvT0pu%27c%12%06%40%04%0F%2A%40%3ANri%3Af2Cp%21%13%27En9%12%1CU0%0B%3Cq%28zz9%02B%0AB%23%15%02%18CO%3AV%2Fbs%24%5E%3DSV%25
%08n%15%3E%5B%1Anh%3C%27%2BQ%3A%5ER%18%02%16RZH%29%7B%0F%7DF8%03%1CE%10jH%1BSeKa%222gk%3B%13Pb%5F3%2F%07J%7BTwM%23ol8%18%5D%1FK%13%17%40
%14%2F%5BxJ%2EkQ2%0E%1A%7BX%20%02%27UNH%2A15%29%23x%02%07B%5B3YRUbE%3C%7F2kx8%04%17%08%0By%14%1BDtF%2B%7B%21am%2EHRUV%22%02%09Ie%5EaJ%2Ek
%3F%1F%1F%02ZDvO%3DCvT0pf%3D6kY%0DWC3g%01Tn%19ch%2Fjk%2E%06%0B%08T9%09%1ACySc10g%7B%23%0F%1ES%09j%0B%0BHpS7%20w9%2EyD%5D%04%0By%0B
%0BHpS7%20zmz9%05%1BET7%13%0BAxU%26%20%12fzw%3E%07Z%5B%25GFurF%2Cq%28%2E%2C%7EJAUR8%14%1BUtF%2B%7B%21am%2EHB%07K%2A%1B%5F%14%21%1Fo
%29%7E6%29fFZ%0E%1BcK%5EZk%5B%1AP%02%2C0iJAqp%15Y%3C%2Fm%3E&onLoad=%5Btype%20Function%5D&reporter=%5Bobject%20Object%5D
```

Step 2 – Paste into the Decrypt Tool and Click <Decrypt Message>



http://l2.glanceg...com/djunpack.html
glanceguide
Glanceguide Combined Beacon Message Inspection Tool
Paste (Ctrl V)
Decrypt Message

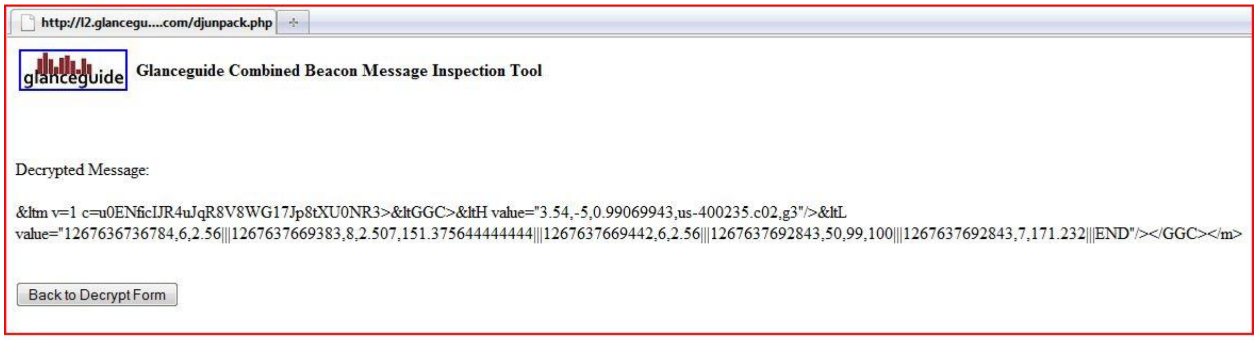
This tool is for internal use at Glanceguide and Nielsen only.

For POST messages, copy everyting in POST from Firebug and paste here.
For GET messages, copy the entire GET message and paste here.

```
%2C%15i%25%7D%3A%2C%7B%01%1D%19%03fU%5C%17%20%0B%3C
%2Cv%22xdVA%0A%0B%1AE%18EyP%3A%21f%3F%2DaCX%07%01aVX%13%2D
%11s%2Ah%3C1bB%12HKgWX%13%23%16h%2Ar7%2CoGB%0C%1BdK%5B%14
%22%09n%29u%20%2C%60AX4%03bQZ%10%21%11%23%608%3F
%2DaCX%077%60SW%10%21%17s%2Ah%3C1bB%12HKgWX%13%23%16h%2A
%7D%3C%27cGB%01%07z%5CW%08%24%15o%608r%2EeBY%02%04aSW%16%2D
%1110s%22%2E%60E%40%06%04d%19%12XPk%1B%3Ek0%23x3%29w%09%3C
%2Fm%3E&onLoad=%5Btype%20Function%5D&
reporter=%5Bobject%20Object%5D
```

Copyright © 2009 GlanceGuide, Inc. All Rights Reserved

Step 3 – Decrypt the contents



http://l2.glancegu....com/djunpack.php
glanceguide
Glanceguide Combined Beacon Message Inspection Tool
Back to Decrypt Form

Decrypted Message:

```
&ltm v=1 c=u0ENficIJR4uJqR8V8WG17Jp8tXU0NR3>&ltGGC>&ltH value="3.54,-5,0.99069943,us-400235.c02,g3"/>&ltL
value="1267636736784,6,2.56|||1267637669383,8,2.507,151.375644444444|||1267637669442,6,2.56|||1267637692843,50,99,100|||1267637692843,7,171.232|||END"/></GGC></m>
```

You can now see the events captured during the video play immediately before and including the video stop event.

The above can be broken down as follows:

A) **<m v=1 c=u0ENficIJR4uJqR8V8WG17Jp8tXU0NR3><GGC><H value="3.54,-5,0.99069943,us-400235.c02,g3"/>**

Basic header information that contains the cookie id, browser version and Nielsen clientid and vcid. Nothing to validate from a video player event perspective.

B) **<L value="1267636736784,6,2.56|||1267637669383,8,2.507,151.375644444444|||1267637692843,50,99,100|||1267637692843,7,171.232|||END"/></GGC></m>**

Each of the “|||” separated values represent a video player event that has been captured by the beacon. The first number is each block represents a unique internal sequence number. Each block breaks down as follows:

1267636736784,6,2.56 = Player pause event 6 with parameter 1 set to 2.56 seconds.

1267637669383,8,2.507,151.375644444444 = Player seek (scrub) event 8 with parameter 2 set to the “from” time mark (2.507 seconds) and parameter 3 set to the “to” time mark (151.37 seconds)

1267637692843,50,99,100 = Auto generated event. Should be present, but nothing to validate if it is.

1267637692843,7,171.232 = Player stop event 7 with parameter 1 set to 171.23 seconds.

Appendix A: Tracked Events

Events highlighted in gray below are mandatory where the video player controls are provided. All other events are optional, but the more of them you track, the better metrics of user experience you will get.

Note: you can choose to use event 3 OR event 15.

Func. Code	**Type**	**Param1**	**Param2**	**Param3**	**Param4**	**Auto Gen?**
1	Load Player (auto generated)	Containing page URL	Referring Page URL; optional, default null			Y
2	Unload Player (auto generated)					Y
3	Load Video	URL of Video that identifies .flv or stream *Also know as "mediaFile"*	Video Type: "content","preroll"," midroll","postroll" (can also be passed in the *Videoinfo* xml string as <vidtype>value</vidtype>)	*Videoinfo* xml string	Chapter Number; set to "1" for single clip videos	Y
4	Unload Video ***Only send after an event 7***	Current Position in secs; optional	Video Type same as matching Load Video; optional			Y
5	Play video	Current Position in secs				Y
6	Pause Video	Current Position in secs				Y
7	Stop	Current Position in secs				Y
8	Seek (fwd or rewind)	Current Position in secs	New Position in secs			Y
9	Mute	0 or 1 ("true" or "false" and "On" or "Off" is also acceptable)				Y
10	Full screen	0 or 1 ("true" or "false" is also acceptable)				Y
11	setVolume	Number (0 to 100)				Y
12	Download					

Func. Code	**Type**	**Param1**	**Param2**	**Param3**	**Param4**	**Auto Gen?**
13	Load Overlay	Current Position in secs	URL of overlay SWF or image			
14	Unload Overlay	Current Position				
15	Load and Play Video	URL of Video that identifies .flv or stream *Also know as “mediaFile”*	Video Type: “content”,”preroll”,” midroll”,”postroll” (can also be passed in the *Videoinfo* xml string as <vidtype>value</vidtype>)	*Videoinfo* xml string	Chapter Number; set to “1” for single clip videos	Y
16	ClickURL (instream)	URL of advertiser or other link				
17	Email Video Link	URL of video				
18	Bookmark Link	URL of video				
19	Comment	URL of Video				
20	Publish	URL of Video				
21	Interact	URL in overlay	Type (number 1 to 255)			
22	ClickURL (external)	URL of advertiser				
23	Rate	Value (Number 1 to 5)				
24	StreamStartDelay	Time spent buffering before play started (in millisecs)	CDN Name; optional	Bandwidth (number, optional)		Y
25	StreamBuffer	Number of buffering events	Total buffering time in millisecs	Max value of a single buffering event in millisecs		Y
26	StreamFailure	Number of failure events				Y

Func. Code	Type	Param1	Param2	Param3	Param4	Auto Gen?
49	Playing (periodic play position update)	Current position in secs				
50	GGStatus (auto generated by GG)					Y
51	setPageURL	URL of HTML page that contains the player	URL of Referring Page; Optional			

Appendix B: VideoInfo (Meta-Data)

Meta-data Identifier	Required?	Description
<length>	Y	Number that contains the length of the video in seconds. Should round to two decimal places. i.e. 103.34
<uurl>	N	URL that uniquely identifies the video asset. This URL often points to the original file system location of the creative asset. Video Analytics requires that this be a static value for correct aggregation of numbers against a particular video creative asset. ***Important*** If this value changes over the life time of it's use then Video Analytics will treat it as a new creative asset. You should omit the UURL if the value changes for the same creative asset.
<category>	N *(but strongly recommended –* see section on Meta-Data Categorization)	Category of the video content (e.g. The Simpsons) or ad. *Note: if you leave this value blank, but specify <censuscategory> then the value in <censuscategory> is populated in <category>*
<subcategory>	N *(but strongly recommended –* see section on Meta-Data Categorization)	Subcategory of the video content (e.g. Season 20) or ad
<title>	Y *(if <title> cannot be supplied then you will need to populate the <uurl> tag – see above)*	Title of video (e.g. Mypods and Boomsticks) *Note: also used by VideoCensus*
<xtag>	N	Enables additional grouping of videos by a user defined tag. Is additional (and separate) from <category> and <subcategory>.
<imgurl>	N	URL of thumbnail image
<noskip>	N	True or false; if the video is ad, can it be skipped?
<censuscategory>	N	Customer-defined "category" ("cg" parameter) for VideoCensus client defined entity reporting.

<vidtype>	N	“content”, “preroll”, “midroll”, “postroll”. If no value is specified then “content” is assumed.
<iagcategory>	N	Value passed to IAG product if enabled. If no value is specified then the value from <category> is used.
<pd>	N	Distribution partner 9a short string value). Used by IAG
<oad>	Y *(if tagging for IAG)*	Original Air Date in MM/DD/YY format. Used by IAG.
<nol_*XXX*>	N	Creates and populates a variable on the HTTP request named *XXX* and assigns the value specified in the tag. Used by Nielsen custom reporting. Not used by Video Analytics.
<livestream>	N	Valid values are **yes** or **no**

Appendix C: IAG (Meta-Data)

Meta-data Identifier	Required for Content?	Required for Ads?	Field Definition	Default Rules	Override Variable
sid	Y	Y	Source ID	Customer specific. Provided by IAG at start of implementation.	Not applicable
tfid	Y	Y	Tag format ID	Customer specific. Provided by IAG at start of implementation.	Not applicable
bcr	Y	Y	Broadcaster or Client	Auto-generated from Video Census clientid (CI)	<iag_bcr>
pgm	Y	Y	Program – this should be the name of the program being played, or the name of the program the ad is included in.	Auto-generated from Video Analytics <category> if specified. Can be explicitly populated via <iagcategory>	Not applicable
epi	Y	Y	Episode – this should be the name of the episode being played, or the name of the episode the ad is included in. This field should also contain an indicator for short form vs long form content. (sf or lf) Example: <episode name> - lf	Auto-generated from <title> (Video Census TL)	<iag_epi>
seg	Y	Y	Segment - For content, this is the segment or chapter of the content. For ads, this is the segment or chapter that the ad precedes.	Auto-generated from Video Analytics chapter number – parameter 4 of event 3 or event 15.	<iag_seg>
cp	Y	Y	Content position- possible values are: soc (start of content), cep (content end point), cmp (content mid point)	Automatically generated	None

pd	Y	Y	Partner distribution-possible values: call sign or short abbreviation of Partner (example: MySpace) If you do not have access to this data, fill in this parameter with NA.	Must explicitly supply in the event 3 or event 15 call with <pd> meta-data	<iag_pd>
oad	Y	Y	Original air date of Episode on TV: Example 01-24-2010	Must explicitly supply in the event 3 or event 15 call with <oad> meta-data	<iag_oad>
brn	N	Y	Brand or client	Auto-generated from clientid and VCID	<iag_brn>
cte	N	Y	Creative- must be a URL that links to the creative or an id that can be used to view the creative. This id or URL must be unique across all creatives in your system.	Auto-generated from parameter 1 (video file name) of event 3 or event 15	<iag_cte>
ap	N	Y	Ad Placement – possible values: pre, mid, post	Auto-generated from parameter 2 (video type; content, preroll, midroll, postroll) of event 3 or event 15	Not applicable
pod	N	N	This information is used to quantify the effectiveness levels (through IAG measurement) based on the number of pods in the stream, thespecific pod number in the stream and the position of the ad within the pod. For example, if ads in pods with two ads get lower recall than ads in pods with just a single ad. Or ads in the second pod compared to ads in the fifth pod in terms of Recall. Publishers are	Must explicitly supply in the event 3 or event 15 call with <iag_pod> meta-data	Not applicable

			increasingly experimenting with new formats and ad models for their premium video offerings. They hope to use this data to establish pricing, and to prove performance to their advertiser/agency clients.		
fp	N	N	Indicates long-form ("fp") or short form / clips ("sp")	Must explicitly supply in the event 3 or event 15 call with <iag_fp> meta-data	Not applicable
apt	N	N	Indicates ad placement type. Valid values: "ol" – overlay "be" – branded entertainment "as" – ad selector "na" – regular ad	Must explicitly supply in the event 3 or event 15 call with <iag_apt> meta-data	Not applicable
cust1	N	N	Custom field for passing bespoke data to IAG	Must explicitly supply in the event 3 or event 15 call with <iag_cust1> meta-data	Not applicable

Appendix D: Sample code for JavaScript Instrumenting JW Player

Two examples are included in this section.

Example 1: The following code shows a very basic implementation of the JavaScript beacon.

In this example the code has been split into page level code, and then a self-contained example handler library (ggCom.js) that binds to the events exposed by most video players; in this case JW Player 5.6 Flash Player . If you wish to have a copy of ggCom.js then please contact Nielsen client services.

Page code: needs to be executed prior to video player / object draw.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>JW Player 5.6 with Nielsen VA Beacon</title>

<script type="text/javascript" src="ggcmb370.js"></script>

<script type="text/javascript" src="ggCom.js"></script>

<script type="text/JavaScript">

             var _nolggGlobalParams = {

                    clientid : "us-502202",

                    vcid : "c01",

                    sfcode : "us",

                    cisuffix : "gg",

                    prod : "sc"

                    };

</script>

<script type="text/javascript">

             var canUseSWF = false; // retained for backwards compatibility

             var uid = 0;

             var detectBrowser = true; //optional - used to disable window object call
for non-browser (app) use

             var gg1 = new gg();

             gg1.ggInitialize(_nolggGlobalParams,uid,canUseSWF,detectBrowser);

</script>

</head>

<body>

<script type='text/javascript' src='jwplayer.js'></script>
```

```
   </script>
             <div id="left" style="height:100%; width:550px">
                    <div class="class1" id="div1">
                           This is &lt;div&gt; 1.
                    </div>
          </div>
      <script type="text/javascript">
       function setup(id) {
           $jw(id).setup({
               levels: [
               {file: 'bunny.mp4', image:'bunny.jpg'},
               {file: 'bunny.ogg', image:'bunny.jpg'},
               {file: 'bunny.webm', image:'bunny.jpg'}
               ],
               players:[
                   {type:'html5'},
                                 {type:'flash', src:'player.swf'}
               ],
               components: {
                   playlist: {
                       position: 'right',
                       size: '200'
                   },
                   controlbar: {
                       position: 'over'
                   }
               },
               width:'550',
               height:'300',
               skin: '',
               events: {
                   onReady: function() {
                       bindEvents (gg1, this);
                   }
               }
           });
```

```
        }
        setup('div1');
    </script>
</script>
</body>
</html>
```

Event Library for Example 1: an example definition of binding the events exposed by the video player to the Nielsen JS library. This file is called ggCom.js and is referenced as a script in the page code on the previous page of this document.

```
var ggCom1;
function bindEvents(beacon, sender) {
       ggCom1 = new ggCom(beacon, sender);
       var arg;
       sender.onMeta(function(data, player) {ggCom1.onMediaMeta(data, player)});
       sender.onPlay(function(args) {ggCom1.onCurrentStateChanged(args)});
       sender.onPause(function(args) {ggCom1.onCurrentStateChanged(args)});
       sender.onComplete(function(args) {ggCom1.onCurrentStateChanged(args)});
       sender.onIdle(function(args) {ggCom1.onCurrentStateChanged(args)});
       sender.onTime(function(args) {ggCom1.updatePos(args)});
       sender.onFullscreen(function(args) {ggCom1.onFullscreen(args)});
       sender.onMute(function(args) {ggCom1.onMute(args)});
       sender.onVolume(function(args) {ggCom1.onVolume(args)});
}
function ggCom(beacon, player) {
       this.gg = beacon;
       this.player = player;
       this.useSWF = false;
       this.media;
       this.ie = navigator.userAgent.indexOf("MSIE") != -1;
       this.cur_movie;
       this.cur_position;
       this.cur_volume;
       this.cur_mute;
       this.timer;
       this.playStateTimer;
       this.queryInterval = 2;  //in seconds
```

```
        this.duration;

        this.fullScreen;

        this.cur_movie_url;

        this.content = "content";

        this.playerStr = "";

        this.metaData={};

        this.prevState = 0;

        this.movieStatus = {loaded:false, played:false, stopped:true};

        this.uniqueID = '';

        this.debug = true;

}


ggCom.prototype.onCurrentStateChanged = function(args) {

        if(this.player.getState() != "BUFFERING" && this.prevState != 'Closed' &&
this.prevState != this.player.getState()) {

                var cur_pos = this.player.getPosition();

                var send_pos;

                if(cur_pos >= this.cur_position && cur_pos <= this.duration) {

                        send_pos = cur_pos;

                } else if(cur_pos > this.duration && this.duration > 0) {

                        send_pos = this.duration;

                } else {

                        send_pos = this.cur_position;

                }

                switch (this.player.getState())

                {

                        case 'IDLE':

                                if(!this.movieStatus.stopped)

                                {

                                        this.movieStatus.stopped = true;

                                        this.movieStatus.loaded = false;

                                        this.gg.ggPM(7,send_pos);

                                        this.cur_position = 0;

                                        if(this.debug)

                                                this.logger("this.gg.ggPM(7,"+send_pos+")");

                                }
```

```
                    break;
                    case 'PAUSED':
                           if(args.oldstate == "PLAYING")
                           {
                                  if(this.debug)
                                         this.logger("this.gg.ggPM(6,"+cur_pos+")");
                                  this.gg.ggPM(6,send_pos);
                           }
                    break;
                    case 'PLAYING':
                           this.movieStatus.played = true;
                           if(this.debug)
                                  this.logger("this.gg.ggPM(5,"+send_pos+")");
                           this.movieStatus.stopped = false;
                           this.gg.ggPM(5,send_pos);
                           var that = this;
                    break;
                    case 'Closed':
                           if(!this.movieStatus.stopped)
                           {
                                  if(this.debug)

      this.logger("this.gg.ggPMclose(7,"+send_pos+")");
                                  this.movieStatus.stopped = true;
                                  this.movieStatus.loaded = false;
                                  this.gg.ggPM(7,send_pos);
                           }
                    break;
             }
             this.prevState = this.player.getState();
      }
}
ggCom.prototype.onMediaMeta = function(args, player) {
      if(this.movieStatus.loaded == false && args.metadata && args.metadata.duration)
      {
             this.movieStatus.loaded = true;
```

```
              this.cur_movie = args.metadata.title;

              this.duration = args.metadata.duration.toFixed(1);

              this.cur_movie_url = args.metadata.source;

              this.cur_position = 0;

              var media = this.player.getPlaylistItem();

              var that = this;

              this.prevState = "Loading";

              this.cur_movie = media.title ? media.title : media.mediaid;

              this.duration = media.duration.toFixed(1);

              this.cur_mute;

              this.cur_volume;

              this.uniqueID = '';

              var type = "content";

              this.type = type;

              this.timer = 0;

              var l_metaData = "";

              this.prevState = 0;

              var ppp = this.player.container.getElementsByTagName('video')[0];

              this.cur_movie_url = ppp.currentSrc;

              if(this.movieStatus.played) {

                     if(this.debug)

                            this.logger("this.gg.ggPM(15,"+this.cur_movie_url+",
"+type+",
<title>"+this.cur_movie+"</title><uurl>"+this.cur_movie_url+"</uurl><length>"+String(t
his.duration)+"</length>"+l_metaData+")");

                     this.gg.ggPM(15, this.cur_movie_url, type,
"<title>"+this.cur_movie+"</title><uurl>"+this.cur_movie_url+"</uurl><length>"+String(
this.duration)+"</length>"+l_metaData);

              }

              else

              {

                     if(this.debug)

                            this.logger("this.gg.ggPM(3,"+this.cur_movie_url+",
"+type+",
<title>"+this.cur_movie+"</title><uurl>"+this.cur_movie_url+"</uurl><length>"+String(t
his.duration)+"</length>"+l_metaData+")");

                     this.gg.ggPM(3, this.cur_movie_url, type,
"<title>"+this.cur_movie+"</title><uurl>"+this.cur_movie_url+"</uurl><length>"+String(
this.duration)+"</length>"+l_metaData);
```

```
                }
        }
}
ggCom.prototype.onMediaEnded = function(sender, args) {
        var media = sender.findName(this.mediaName);
        if(!this.movieStatus.stopped) {
                var cur_pos = parseFloat(media.Position.Seconds.toFixed(1));
                var send_pos = 0;
                if(parseFloat(cur_pos) >= parseFloat(this.cur_position) &&
parseFloat(cur_pos) <= parseFloat(this.duration)) {
                        send_pos = cur_pos;
                } else if(parseFloat(cur_pos) > parseFloat(this.duration) &&
parseFloat(this.duration) > 0) {
                        send_pos = this.duration;
                } else {
                        send_pos = this.cur_position;
                }
                this.movieStatus.stopped = true;
                if(this.debug)
                        this.logger("this.gg.ggPM(7,"+String(send_pos)+")");
                this.gg.ggPM(7,send_pos);
        }
}
ggCom.prototype.onMediaUnload = function() {
        var media = this.media;
        var cur_pos = parseFloat(media.Position.Seconds.toFixed(1));
        var send_pos;
        if(parseFloat(cur_pos) >= parseFloat(this.cur_position) && parseFloat(cur_pos)
<= parseFloat(this.duration)) {
                send_pos = cur_pos;
        } else if(parseFloat(cur_pos) > parseFloat(this.duration) &&
parseFloat(this.duration) > 0) {
                send_pos = this.duration;
        } else {
                send_pos = this.cur_position;
        }
        if(!this.movieStatus.stopped) {
                if(this.debug)
                        this.logger("this.gg.ggPMunload(7,"+send_pos+")");
                this.gg.ggPM(7,send_pos);
                this.movieStatus.stopped = true;
        }
        if(this.movieStatus.loaded && this.movieStatus.played) {

                if(this.debug)
                        this.logger("this.gg.ggPM(4,"+send_pos+", "+this.content+")");
                this.gg.ggPM(4, send_pos, this.content);
                this.movieStatus.loaded = false;
        }
}
ggCom.prototype.updatePos = function(caller) {
        if(this.movieStatus.stopped)
                return;
        if(Math.abs(this.cur_position - Math.round(caller.position)) > 2)
                this.onPositionChange(this.cur_position, Math.round(caller.position));
        this.cur_position = caller.position.toFixed(1);
        if(Math.abs((this.timer - this.cur_position)) > this.queryInterval)
        {
                this.timer = this.cur_position;
                if(this.debug)
                        this.logger("this.gg.ggPM(49,"+this.cur_position+")");
                this.gg.ggPM(49, this.cur_position);
        }
}
ggCom.prototype.onFullscreen = function(args) {
                this.fullscreen = args.fullscreen;
                if(this.debug)
                        this.logger("this.gg.ggPM(10,"+this.fullscreen+")");
                this.gg.ggPM(10, String(this.fullscreen));
}
ggCom.prototype.onMute = function(args) {
                this.mute = args.mute;
                if(this.debug)
```

```
                    this.logger("this.gg.ggPM(9,"+this.mute+")");
             this.gg.ggPM(9, String(this.mute));
}
ggCom.prototype.onVolume = function(args) {
             this.volume = args.volume;
             if(this.debug)
                    this.logger("this.gg.ggPM(11,"+this.volume+")");
             this.gg.ggPM(11, String(this.volume));
}
ggCom.prototype.onPositionChange = function(oldPos, newPos) {

             if(this.debug)
                    this.logger("this.gg.ggPM(8,"+oldPos+", "+newPos+")");
             this.gg.ggPM(8, String(oldPos), String(newPos));
             //this.cur_position = args.newPos;
}
ggCom.prototype.logger = function(logStr) {
             try {
                    console.log(logStr);
             } catch (e) {
                    // no logging
             }
}
```

Example 2: In this example, the coder instantiates a SWFObject

```
<script type="text/javascript" src="http://secure-us.imrworldwide.com/novms/js/2/ggcmb370.js"></script>
...
     <script type="text/javascript">
        var _nolggGlobalParams = {
          clientid: "us-802832",
          vcid: "b01",
          cisuffix: "",
          sfcode: "",
          prod: "sc"
        };
        var canUseSWF = false; // retained for backwards compatibility
        var uid = 0;

       var detectBrowser = true; //optional – used to disable window object call for
non-browser (app) use

        var gg1 = new gg();

        gg1.ggInitialize(_nolggGlobalParams,uid,canUseSWF, detectBrowser);

      </script>

...

      <script type="text/javascript">

        var s1 = new SWFObject('/mediaplayer/player.swf','player1','640','379','9');

        s1.addParam('allowfullscreen','false');

        s1.addParam('allowscriptaccess','always');

s1.addParam('flashvars','file=/JSES/video/YMSE1603.flv&image=/JSES/preview/YMSE1603.jp
g&autostart=true&id=player1&name=player1&plugins=drelated-
1&drelated.dxmlpath=/JSES/XML/relatedYMSE1603.xml&drelated.dposition=bottom&drelated.d
skin=/mediaplayer/grayskin.swf&drelated.dtarget=_self');

        s1.write('player');

      </script>

      <script type="text/javascript">

        var player;

        var cur_pos;

        var last_pos;

        var duration;

        var timeoutIDduration;

        var timeoutIDload;

        var nielsenLoaded = false;

        function playerReady(obj){

          player = document.getElementById(obj['id']);

          player.addModelListener("STATE","toggleState");

          player.addModelListener("TIME","timeChange");

          player.addViewListener("SEEK","playerSeek");

          player.addViewListener("VOLUME","playerVolume");

          checkForDuration();

        };

        function playerVolume(obj){

          gg1.ggPM(11, obj['percentage']);

        };

        function playerSeek(obj){
          gg1.ggPM(8, last_pos, obj['position']);
        };
        function timeChange(obj){
          cur_pos = obj['position'];
          if(cur_pos > 0){
            last_pos = cur_pos;
          }
          duration = obj['duration'];
        };
        function checkForDuration(){
          if(duration !== undefined){
            clearTimeout(timeoutIDduration);
            gg1.ggPM(3,
"http://supplements.jshoulderelbow.org/JSES/video/YMSE1603.flv", "content",
"<uurl>http://supplements.jshoulderelbow.org/content/osteochondritis-dissecans-
capitellum</uurl><length>" + duration +
"</length><category>Orthopedics</category><subcategory>JSES</subcategory><title>Osteoc
hondritis Dissecans of the Capitellum</title>", 1);
            nielsenLoaded = true;
          }else{
            timeoutIDduration = setTimeout("checkForDuration();", 100);
            nielsenLoaded = false;
          }
        };
        function checkForLoad(){
          if(nielsenLoaded === true){
            clearTimeout(timeoutIDload);
            gg1.ggPM(5, last_pos);
          }else{
            timeoutIDload = setTimeout("checkForLoad();", 100);
          }
        };
        function toggleState(obj){
          if(obj.newstate == 'PLAYING'){
            checkForLoad();
          }
          if(obj.newstate == 'PAUSED'){
            gg1.ggPM(6, last_pos);
          }
          if(obj.newstate == 'COMPLETED'){
            gg1.ggPM(7, last_pos);
          }
          if(obj.newstate == 'IDLE'){
            gg1.ggPM(7, last_pos);
          }
        };
      </script>
```

Appendix E: Sample Code for Instrumenting HTML5 <video>

This is a complete example of using the HTML5 <video> tag method. It is based on the JWPlayer HTML5 approach.

In this example the code has been split into page level code, and then a self-contained example handler library (ggCom_embed.js) that binds to the events exposed by most video players; in this case JW Player 5.6 Flash Player . If you wish to have a copy of ggCom_embed.js then please contact Nielsen client services.

Page code: needs to be executed prior to video player / object draw.

```
<!DOCTYPE html>

<html>

<head>

    <title>Nielsen HTML5 using Combined Video Beacon</title>

    <script type="text/javascript">

        // listener function changes src

        function myNewSrc()

             {

                    var myVideo = document.getElementsByTagName('video')[0];

                    if(navigator.userAgent.indexOf("MSIE") != -1)

                           myVideo.getElementsByTagName('source')[0].src =
"ed_1024.mp4";

                    else if(myVideo.canPlayType("video/ogg"))

                           myVideo.src = "ed_1024.ogg";

                    else

                           myVideo.src = "ed_1024.mp4";

                    myVideo.load();

                    myVideo.play();

        }

        // add listener function to ended event

        function myAddListener()

             {

              var myVideo = document.getElementsByTagName('video')[0];

             bindTagEvents(gg1,myVideo); // this function is in ggCom_embed.js

              myVideo.addEventListener('ended',myNewSrc,false);

        }

    </script>

</head>
```

```
<body onLoad="myAddListener()">
    <script type="text/javascript" src="ggcmb370.js"></script>


       <script type="text/javascript" src="ggCom_embed.js"></script>
       <script type="text/javascript">
              var _nolggGlobalParams = {
                            clientid:"us-502202", // provided by Nielsen
                            vcid : "c09",
                            sfcode : "us",
                            cisuffix : "gg",
                            prod : "vc,sc,iag"
              };
              var gg1 = new gg();
              var uid = 0; //provided by Nielsen
              var oldFlashDetect = false; // no longer used (dummy placeholder for
legecy implementations)
              var detectBrowser = true; //optional -- used to disable window object
call for non-browser use
              gg1.ggInitialize(_nolggGlobalParams, uid, oldFlashDetect, detectBrowser);
    </script>
    <video controls="true" style="border:0.2cm groove black">
        <source src="bunny.mp4" />
        <source src="bunny.ogg" />
    </video>
<br>Nielsen Tutorial - Native HTML5 Video Tag Implementation
<br>Note: to replay the video you will have to refresh the page
</body>
</html>
```

Event Library for HTML5: an example definition of binding the events exposed by the video player to the Nielsen JS library. This file is called ggCom_embed.js and is referenced as a script in the page code on the previous page of this document.

```
var ggCom1;
function bindTagEvents(beacon, sender) {
       ggCom1 = new ggCom(beacon, sender);
       var arg;
       sender.addEventListener('loadedmetadata', function(args)
{ggCom1.onMediaMeta(args)}, false);
```

```
        sender.addEventListener('play', function(args) {ggCom1.onPlayEvt(args)},
false);
        sender.addEventListener('pause', function(args) {ggCom1.onPauseEvt(args)},
false);
        sender.addEventListener('ended', function(args) {ggCom1.onEndEvt(args)},
false);
        sender.addEventListener('timeupdate', function(args) {ggCom1.updatePos(args)},
false);
        sender.addEventListener('seeked', function(args) {ggCom1.onSeeked(args)},
false);
        sender.addEventListener('seeking', function(args) {ggCom1.onSeeking(args)},
false);
        sender.addEventListener('volumechange', function(args) {ggCom1.onVolume(args)},
false);
}
function ggCom(beacon, player) {
        this.gg = beacon;
        this.player = player;
        this.useSWF = false;
        this.media;
        this.ie = navigator.userAgent.indexOf("MSIE") != -1;
        this.cur_movie;
        this.cur_position = 0;
        this.seekPosition = 0;
        this.seeking = false;
        this.cur_volume;
        this.cur_mute;
        this.timer;
        this.playStateTimer;
        this.queryInterval = 2; //in seconds
        this.duration;
        this.fullScreen;
        this.cur_movie_url;
        this.content = "content";
        this.playerStr = "";
        this.metaData={};
        this.prevState = 0;
        this.movieStatus = {loaded:false, played:false, stopped:true};
        this.uniqueID = '';
        this.debug = true;  //enables and disables logging of events to the console
}
ggCom.prototype.onPlayEvt = function (arg)
{
        this.movieStatus.played = true;
        this.movieStatus.stopped = false;
        if(!this.movieStatus.loaded)
        {
                this.onMediaMeta(arg);
                return;
        }
        this.gg.ggPM(5,arg.target.currentTime.toFixed(1));
        if(this.debug) {

        this.logger("this.gg.ggPM(5,"+arg.target.currentTime.toFixed(1)+")");
        }
}
ggCom.prototype.onPauseEvt = function (arg)
{
        if(this.debug) {
                this.logger("this.gg.ggPM(6,"+arg.target.currentTime.toFixed(1)+")");
        }
        this.gg.ggPM(6,arg.target.currentTime.toFixed(1));
}
ggCom.prototype.onEndEvt = function (arg)
{
        if(!this.movieStatus.stopped && this.movieStatus.loaded) {
                this.movieStatus.stopped = true;
                this.movieStatus.loaded = false;
                this.gg.ggPM(7,arg.target.currentTime.toFixed(1));
                this.cur_position = 0;
                if(this.debug) {

        this.logger("this.gg.ggPMstop(7,"+arg.target.currentTime.toFixed(1)+")");
                }
        }
```

```
}
ggCom.prototype.onMediaMeta = function(args) {
        if(this.movieStatus.loaded == false && args.target.duration)
        {
                this.movieStatus.loaded = true;
                this.cur_movie = args.target.title;
                this.duration = args.target.duration.toFixed(1);
                this.cur_movie_url = args.target.src ? args.target.src :
args.target.currentSrc;
                var media = args.target;
                var that = this;
                this.prevState = "Loading";
                this.cur_movie = media.title ? media.title : this.cur_movie_url;
                this.cur_mute;
                this.cur_volume;
                this.uniqueID = '';
                var type = "content";
                var category = "TestCat";
                var subcategory = "TestSubCat";
                var censuscategory = "TestCat";
                this.type = type;
                var l_metaData = "";
                this.prevState = 0;
                this.timer = 0;
                if(this.movieStatus.played)
                {
                        if(this.debug) {
                                this.logger("this.gg.ggPM(15,"+this.cur_movie_url+",
"+type+",
<title>"+this.cur_movie+"</title><uurl>"+this.cur_movie_url+"</uurl><length>"+String(t
his.duration)+"</length>"+"<censuscategory>"+censuscategory+"</censuscategory>"+"<cate
gory>"+category+"</category>"+"<subcategory>"+subcategory+"</subcategory>"+l_metaData+
")");
                        }
                        this.gg.ggPM("15", this.cur_movie_url, type,
"<title>"+this.cur_movie+"</title><uurl>"+this.cur_movie_url+"</uurl><length>"+String(
this.duration)+"</length>"+"<censuscategory>"+censuscategory+"</censuscategory>"+"<cat
egory>"+category+"</category>"+"<subcategory>"+subcategory+"</subcategory>"+l_metaData
);
```

```
            }

            else

            {

                    if(this.debug) {

                            this.logger("this.gg.ggPM(3,"+this.cur_movie_url+",
"+type+",
<title>"+this.cur_movie+"</title><uurl>"+this.cur_movie_url+"</uurl><length>"+String(t
his.duration)+"</length>"+"<censuscategory>"+censuscategory+"</censuscategory>"+"<cate
gory>"+category+"</category>"+"<subcategory>"+subcategory+"</subcategory>"+l_metaData+
")");

                    }

                    this.gg.ggPM(15, this.cur_movie_url, type,
"<title>"+this.cur_movie+"</title><uurl>"+this.cur_movie_url+"</uurl><length>"+String(
this.duration)+"</length>"+"<censuscategory>"+censuscategory+"</censuscategory>"+"<cat
egory>"+category+"</category>"+"<subcategory>"+subcategory+"</subcategory>"+l_metaData
);

            }

            // If you need to also call the Legacy Video Census beacon for dual-
beaconing purposes then this is the trigger point.  I.e. video start

            _scdav(this.cur_movie,censuscategory);

        }

}

ggCom.prototype.updatePos = function(caller) {

        if(this.movieStatus.stopped)

                return;

        if(Math.abs(this.cur_position - caller.target.currentTime.toFixed(1)) > 2 &&
!this.seeking) //added to allow click scrubbing in FF and IE

        {

                this.seeking = true;

                this.seekPosition = this.cur_position;

                if(this.debug) {

                        this.logger("updatePos for seek == "+this.seekPosition);

                }

        }

        this.cur_position = caller.target.currentTime.toFixed(1);

        if(Math.abs((this.timer - this.cur_position)) > this.queryInterval)

        {

                this.timer = this.cur_position;

                if(this.debug) {

                        this.logger("this.gg.ggPM(49,"+this.cur_position+")");
            }
            this.gg.ggPM(49, this.cur_position);
      }
}
ggCom.prototype.onFullscreen = function(args) {
            this.fullscreen = args.fullscreen;
            if(this.debug) {
                  this.logger("this.gg.ggPM(10,"+this.fullscreen+")");
            }
            this.gg.ggPM(10, String(this.fullscreen));
}
ggCom.prototype.onMute = function(args) {
            this.mute = args.target.muted;
            if(this.debug) {
                  this.logger("this.gg.ggPM(9,"+this.mute+")");
            }
            this.gg.ggPM(9, String(this.mute));
}
ggCom.prototype.onVolume = function(args) {
            if(args.target.muted != this.mute)
                  this.onMute(args);
            if(args.target.mute)
                  return;
            this.volume = parseInt(args.target.volume * 100);
            if(this.debug) {
                  this.logger("this.gg.ggPM(11,"+this.volume+")");
            }
            this.gg.ggPM(11, String(this.volume));
}

ggCom.prototype.onSeeked = function(caller) {
      if(this.debug) {
            this.logger("seeked == "+this.seekPosition+",
"+caller.target.currentTime.toFixed(1));
      }
      if(this.seeking && Math.abs(this.seekPosition -
caller.target.currentTime.toFixed(1)) > 1)
```

```
        {

                if(this.debug) {
                        this.logger("this.gg.ggPM(8,"+this.seekPosition+",
"+caller.target.currentTime.toFixed(1)+")");
                }
                this.gg.ggPM(8, String(this.seekPosition),
String(caller.target.currentTime.toFixed(1)));
                this.seeking = false;
        }
}

ggCom.prototype.onSeeking = function(caller) {
        if(!this.seeking)
        {
                this.seeking = true;
                this.seekPosition = this.cur_position;
                if(this.debug) {
                        this.logger("seeking == "+this.seekPosition);
                }
        }
}

ggCom.prototype.logger = function(logStr)
{
        try {
                        console.log(logStr);
                } catch (e) {
                        // no logging
                }
}
```

Appendix F: Example Set of Full-Episode / Multi-Chapter Beacon Calls

For the US market, Video Census measurement rules dictate that each chapter / segment start is recorded as a distinct stream. See section Video Player Tag Rules for specific details on when the stream start should / should not trigger.

The following is an actual full-episode play from a large well-known US Network website. It consists of one or more pre-roll Ads followed by six chapters / segments each of which is separated by one or more mid-roll Ad. Occasionally the mid-roll will not trigger if an Ad was viewed within the past 30 seconds, or the user scrubs the play head back into a previously watched chapter.

Note: Each URL is accompanied by its associated decrypted HEX40 POST DATA entry.

a. Pre-roll 1

Pre-roll Start: http://secure-us.imrworldwide.com/cgi-bin/m?ci=us-800560gg&c6=vc,b01&cc=1&pr=iag.sid,2500011683&pr=iag.tfid,1542&pr=iag.bcr,us-800560&pr=iag.pgm,One%20Tree%20Hill&pr=iag.epi,One%20Tree%20Hill%2007%20Every%20Picture%20Tells%20a%20Story&pr=iag.seg,1&pr=iag.pd,cwtv.com&pr=iag.brn,us-800560&pr=iag.ap,pre&pr=iag.cte,227331154&pr=iag.oad,2010-04-26&pr=iag.fp,lf&pr=iag.pod,6_1_1_1&pr=iag.apt,na&pr=iag.cp,soc&c3=st,a&tl=dav0-Every%20Picture%20Tells%20a%20Story&rnd=37716&tp=gg

&ltm v=1 c=8cvC9segXVKO8dWBx4EkJYB305y1ENw3>&ltGGC>&ltH value="3.60,-4,0.50167035,us-800560.b01,g3"/>&ltL value="1282747558621,1,http://demos.digitalsmiths.com/demos/sites/latest/cwtv/index.html?mediaKey=ea560a92-5d9e-43ab-8eb8-e36297f31a13,|||1282747813608,15,227331154,preroll,&ltlength>30</length>&ltcategory>One Tree Hill</category>&ltsubcategory>227331154</subcategory>&ltiag_pgm>One Tree Hill</iag_pgm>&ltiag_epi>One Tree Hill 07 Every Picture Tells a Story</iag_epi>&ltiagcategory>One Tree Hill</iagcategory>&ltiag_fp>lf</iag_fp>&lttitle>Every Picture Tells a Story</title>&ltpd>cwtv.com</pd>&ltiag_pd>cwtv.com</iag_pd>&ltiag_oad>2010-04-26</iag_oad>&ltiag_seg>1</iag_seg>&ltiag_pod>6_1_1_1</iag_pod>&ltap>pre</ap>&ltapt>na</apt>&ltiag_apt>na</iag_apt>,1|||END"/></GGC></m>

Pre-roll Termination: http://secure-us.imrworldwide.com/cgi-bin/m?ci=us-800560gg&c6=vc,b01&cc=1&pr=iag.sid,2500011683&pr=iag.tfid,1542&pr=iag.bcr,us-800560&pr=iag.pgm,One%20Tree%20Hill&pr=iag.epi,One%20Tree%20Hill%2007%20Every%20Picture%20Tells%20a%20Story&pr=iag.seg,1&pr=iag.pd,cwtv.com&pr=iag.brn,us-800560&pr=iag.ap,pre&pr=iag.cte,227331154&pr=iag.oad,2010-04-26&pr=iag.fp,lf&pr=iag.pod,6_1_1_1&pr=iag.apt,na&pr=iag.cp,cep&c3=st,a&tl=dav2-Every%20Picture%20Tells%20a%20Story&tp=gg

&ltm v=1 c=8cvC9segXVKO8dWBx4EkJYB305y1ENw3>&ltGGC>&ltH value="3.60,-4,0.50167035,us-800560.b01,g3"/>&ltL value="1282747843780,50,99,100|||1282747843780,7,30|||END"/></GGC></m>

b. Content - Chapter 1 / Segment 1

Content Chapter 1 Start: http://secure-us.imrworldwide.com/cgi-bin/m?ci=us-800560gg&c6=vc,b01&cc=1&pr=iag.sid,2500011683&pr=iag.tfid,1542&pr=iag.bcr,us-800560&pr=iag.pgm,One%20Tree%20Hill&pr=iag.epi,One%20Tree%20Hill%2007%20Every%20Picture%20Tells%20a%20Story&pr=iag.seg,1&pr=iag.pd,cwtv.com&pr=iag.oad,2010-04-26&pr=iag.fp,lf&pr=iag.cp,soc&cg=One%20Tree%20Hill&tl=dav0-Every%20Picture%20Tells%20a%20Story&rnd=17093&tp=gg

&ltm v=1 c=8cvC9segXVKO8dWBx4EkJYB305y1ENw3>&ltGGC>&ltH value="3.60,-4,0.50167035,us-800560.b01,g3"/>&ltL value="1282747843780,15,rtmpe://wbworldtv.fcod.llnwd.net/a2246/o23/mp4:cwtv/videos/2010/06/23/OTH719-everypicture_b4e5e837a_700kbps.mp4,content,&ltlength>431</length>&ltcategory>One Tree Hill</category>&ltsubcategory>07</subcategory>&lttitle>Every Picture Tells a Story</title>&ltcensuscategory>One Tree Hill</censuscategory>&ltiag_epi>One Tree Hill 07 Every Picture Tells a Story</iag_epi>&ltiagcategory>One Tree Hill</iagcategory>&ltiag_seg>1</iag_seg>&ltiag_fp>lf</iag_fp>&ltfp>lf</fp>&ltoad>2010-04-26</oad>&ltpd>cwtv.com</pd>,1|||END"/></GGC></m>

Note: At the 30 second mark, the user scrubbed forward out of Chapter 1 into Chapter 2. The nature of this publisher's player forces the play head position to the beginning of Chapter 2 after playing the mid-roll Ads. You will notice in the decrypted data below that the event 7 (highlighted) has a parameter of 31. This reflects the point at which Chapter 1 had reached when the user scrubbed out.

Content Chapter 1 Termination: http://secure-us.imrworldwide.com/cgi-bin/m?ci=us-800560gg&c6=vc,b01&cc=1&pr=iag.sid,2500011683&pr=iag.tfid,1542&pr=iag.bcr,us-800560&pr=iag.pgm,One%20Tree%20Hill&pr=iag.epi,One%20Tree%20Hill%2007%20Every%20Picture%20Tells%20a%20Story&pr=iag.seg,1&pr=iag.pd,cwtv.com&pr=iag.oad,2010-04-26&pr=iag.fp,lf&pr=iag.cp,cep&cg=One%20Tree%20Hill&tl=dav2-Every%20Picture%20Tells%20a%20Story&tp=gg

&ltm v=1 c=8cvC9segXVKO8dWBx4EkJYB305y1ENw3>&ltGGC>&ltH value="3.60,-4,0.50167035,us-800560.b01,g3"/>&ltL value="1282747848030,6,2|||1282748011750,5,2|||1282748014046,6,5|||1282748272427,5,5|||1282748298038,50,7,100|||1282748298038,7,31|||END"/></GGC></m>

Midroll 1

Midroll Start: http://secure-us.imrworldwide.com/cgi-bin/m?ci=us-800560gg&c6=vc,b01&cc=1&pr=iag.sid,2500011683&pr=iag.tfid,1542&pr=iag.bcr,us-800560&pr=iag.pgm,One%20Tree%20Hill&pr=iag.epi,One%20Tree%20Hill%2007%20Every%20Picture%20Tells%20a%20Story&pr=iag.seg,2&pr=iag.pd,cwtv.com&pr=iag.brn,us-800560&pr=iag.ap,mid&pr=iag.cte,228164128&pr=iag.oad,2010-04-26&pr=iag.fp,lf&pr=iag.pod,6_2_4_1&pr=iag.apt,na&pr=iag.cp,soc&c3=st,a&tl=dav0-Every%20Picture%20Tells%20a%20Story&rnd=93277&tp=gg

&ltm v=1 c=8cvC9segXVKO8dWBx4EkJYB305y1ENw3>&ltGGC>&ltH value="3.60,-4,0.50167035,us-800560.b01,g3"/>&ltL value="1282748298303,15,228164128,midroll,&ltlength>15</length>&ltcategory>One Tree Hill</category>&ltsubcategory>228164128</subcategory>&ltiag_pgm>One Tree

Hill</iag_pgm>&ltiag_epi>One Tree Hill 07 Every Picture Tells a Story</iag_epi>&ltiagcategory>One Tree Hill</iagcategory>&ltiag_fp>lf</iag_fp>&lttitle>Every Picture Tells a Story</title>&ltpd>cwtv.com</pd>&ltiag_pd>cwtv.com</iag_pd>&ltiag_oad>2010-04-26</iag_oad>&ltiag_seg>2</iag_seg>&ltiag_pod>6_2_4_1</iag_pod>&ltap>mid</ap>&ltapt>na</apt>&ltiag_apt>na</iag_apt>,1|||END"/></GGC></m>

Midroll Termination: http://secure-us.imrworldwide.com/cgi-bin/m?ci=us-800560gg&c6=vc,b01&cc=1&pr=iag.sid,2500011683&pr=iag.tfid,1542&pr=iag.bcr,us-800560&pr=iag.pgm,One%20Tree%20Hill&pr=iag.epi,One%20Tree%20Hill%2007%20Every%20Picture%20Tells%20a%20Story&pr=iag.seg,2&pr=iag.pd,cwtv.com&pr=iag.brn,us-800560&pr=iag.ap,mid&pr=iag.cte,228164128&pr=iag.oad,2010-04-26&pr=iag.fp,lf&pr=iag.pod,6_2_4_1&pr=iag.apt,na&pr=iag.cp,cep&c3=st,a&tl=dav2-Every%20Picture%20Tells%20a%20Story&tp=gg

&ltm v=1 c=8cvC9segXVKO8dWBx4EkJYB305y1ENw3>&ltGGC>&ltH value="3.60,-4,0.50167035,us-800560.b01,g3"/>&ltL value="1282748313835,50,99,100|||1282748313835,7,15|||END"/></GGC></m>

Midroll 2

Midroll Start: http://secure-us.imrworldwide.com/cgi-bin/m?ci=us-800560gg&c6=vc,b01&cc=1&pr=iag.sid,2500011683&pr=iag.tfid,1542&pr=iag.bcr,us-800560&pr=iag.pgm,One%20Tree%20Hill&pr=iag.epi,One%20Tree%20Hill%2007%20Every%20Picture%20Tells%20a%20Story&pr=iag.seg,2&pr=iag.pd,cwtv.com&pr=iag.brn,us-800560&pr=iag.ap,mid&pr=iag.cte,227331154&pr=iag.oad,2010-04-26&pr=iag.fp,lf&pr=iag.pod,6_2_4_2&pr=iag.apt,na&pr=iag.cp,soc&c3=st,a&tl=dav0-Every%20Picture%20Tells%20a%20Story&rnd=78132&tp=gg

EX40=&ltm v=1 c=8cvC9segXVKO8dWBx4EkJYB305y1ENw3>&ltGGC>&ltH value="3.60,-4,0.50167035,us-800560.b01,g3"/>&ltL value="1282748314085,15,227331154,midroll,&ltlength>30</length>&ltcategory>One Tree Hill</category>&ltsubcategory>227331154</subcategory>&ltiag_pgm>One Tree Hill</iag_pgm>&ltiag_epi>One Tree Hill 07 Every Picture Tells a Story</iag_epi>&ltiagcategory>One Tree Hill</iagcategory>&ltiag_fp>lf</iag_fp>&lttitle>Every Picture Tells a Story</title>&ltpd>cwtv.com</pd>&ltiag_pd>cwtv.com</iag_pd>&ltiag_oad>2010-04-26</iag_oad>&ltiag_seg>2</iag_seg>&ltiag_pod>6_2_4_2</iag_pod>&ltap>mid</ap>&ltapt>na</apt>&ltiag_apt>na</iag_apt>,2|||END"/></GGC></m>

Midroll Termination: http://secure-us.imrworldwide.com/cgi-bin/m?ci=us-800560gg&c6=vc,b01&cc=1&pr=iag.sid,2500011683&pr=iag.tfid,1542&pr=iag.bcr,us-800560&pr=iag.pgm,One%20Tree%20Hill&pr=iag.epi,One%20Tree%20Hill%2007%20Every%20Picture%20Tells%20a%20Story&pr=iag.seg,2&pr=iag.pd,cwtv.com&pr=iag.brn,us-800560&pr=iag.ap,mid&pr=iag.cte,227331154&pr=iag.oad,2010-04-26&pr=iag.fp,lf&pr=iag.pod,6_2_4_2&pr=iag.apt,na&pr=iag.cp,cep&c3=st,a&tl=dav2-Every%20Picture%20Tells%20a%20Story&tp=gg

&ltm v=1 c=8cvC9segXVKO8dWBx4EkJYB305y1ENw3>&ltGGC>&ltH value="3.60,-4,0.50167035,us-800560.b01,g3"/>&ltL value="1282748344461,50,99,100|||1282748344461,7,30|||END"/></GGC></m>

c. Content - Chapter 2 / Segment 2

Chapter Start: http://secure-us.imrworldwide.com/cgi-bin/m?ci=us-800560gg&c6=vc,b01&cc=1&pr=iag.sid,2500011683&pr=iag.tfid,1542&pr=iag.bcr,us-800560&pr=iag.pgm,One%20Tree%20Hill&pr=iag.epi,One%20Tree%20Hill%2007%20Every%20Picture%20Tells%20a%20Story&pr=iag.seg,2&pr=iag.pd,cwtv.com&pr=iag.oad,2010-04-26&pr=iag.fp,lf&pr=iag.cp,soc&cg=One%20Tree%20Hill&tl=dav0-Every%20Picture%20Tells%20a%20Story&rnd=86397&tp=gg

&ltm v=1 c=8cvC9segXVKO8dWBx4EkJYB305y1ENw3>&ltGGC>&ltH value="3.60,-4,0.50167035,us-800560.b01,g3"/>&ltL value="1282748389557,15,rtmpe://wbworldtv.fcod.llnwd.net/a2246/o23/mp4:cwtv/videos/2010/06/23/OTH719-everypicture_b4e5e837a_700kbps.mp4,content,&ltlength>254</length>&ltcategory>One Tree Hill</category>&ltsubcategory>07</subcategory>&lttitle>Every Picture Tells a Story</title>&ltcensuscategory>One Tree Hill</censuscategory>&ltiag_epi>One Tree Hill 07 Every Picture Tells a Story</iag_epi>&ltiagcategory>One Tree Hill</iagcategory>&ltiag_seg>2</iag_seg>&ltiag_fp>lf</iag_fp>&ltfp>lf</fp>&ltoad>2010-04-26</oad>&ltpd>cwtv.com</pd>,2|||END"/></GGC></m>

Note: Around the 10 minute mark, the user scrubbed backward out of Chapter 2 into Chapter 1. This publisher does not force the user to sit through another set of mid-roll Ads if you have visited the destination chapter previously. I.e. you are crossing a chapter / segment break that you have triggered previously during this episode view. You will notice in the decrypted data below that the event 7 (highlighted) has a parameter of 253.4. This reflects the point at which Chapter 2 had reached when the user scrubbed back to Chapter 1

Chapter Termination: http://secure-us.imrworldwide.com/cgi-bin/m?ci=us-800560gg&c6=vc,b01&cc=1&pr=iag.sid,2500011683&pr=iag.tfid,1542&pr=iag.bcr,us-800560&pr=iag.pgm,One%20Tree%20Hill&pr=iag.epi,One%20Tree%20Hill%2007%20Every%20Picture%20Tells%20a%20Story&pr=iag.seg,1&pr=iag.pd,cwtv.com&pr=iag.oad,2010-04-26&pr=iag.fp,lf&pr=iag.cp,cep&cg=One%20Tree%20Hill&tl=dav2-Every%20Picture%20Tells%20a%20Story&tp=gg

&ltm v=1 c=8cvC9segXVKO8dWBx4EkJYB305y1ENw3>&ltGGC>&ltH value="3.60,-4,0.40551255,us-800560.b01,g3"/>&ltL value="1282852997226,50,2,100|||1282852997226,7,10|||END"/></GGC></m>

Note: At this point, when Chapter 1 resumes play at the point the play head was positioned by the user when they scrubbed back, a beacon call is not made. This conforms to the Nielsen rules on not recounting chapters that are not interrupted by an Ad play.

The play resumes in the previously watched Chapter with no new start beacon call. The next beacon call you should see is for any new mid-roll Ads executed as a result of reaching a new previously UNWATCHED chapter.