nielsen

Engineering

Nielsen App SDK – Metadata

Quick Start Guide

Revision History

Revision	Date	Change Made	Responsible Engineer
1	05/09/2014	Initial version	Sanjana Murlidhar
2	06/27/2014	Included support for dataSrc and adload	Nidhish Ramachandran
3	07/17/2014	Including opt-out page scaling Instructions and mTVR and mDPR use cases	Suraj Jagga
4	09/29/2014	Included mOCR Integration steps, diagrams, and opt-out URL changes	Suraj Jagga

Contents

List of Figures

1. Introduction

This document goes through various metadata structures passed while integrating with the Nielsen AppSDK for iOS® and Android®.

2. Highlights

- Initialization
- mTVR (Mobile TV Ratings)
- mDPR (Digital Program Ratings)
- mOCR (Online Campaign Ratings)
- mDRM (Digital Radio Measurements)

3. Quick Start

Refer to the user guide for more detail. Parameter values and samples provided below are for demonstration purposes only. Please consult with Nielsen support for values specific to your app.

NOTE All JSON value must be string value. This includes values of true and false and numbers. For example, a value of true should be represented with "true"; a number value, 123, should be "123".

3.1. Instantiate SDK

3.1.1. Init JSON

Initialize a Nielsen App SDK object by calling the initWithAppInfo: method (iOS) or AppSdk.getInstance() method (Android). For these methods, it passes the appname, appid, appversion, and sfcode values as arguments via the AppInfo JSON schema.

Variable Name	Variable Description	Example
appid	appid assigned by Nielsen. It is GUID data type and is specific to your application.	T8A5E427-1970-47E1-BF7D-7575235359A2
appversion	Version of your application.	3.1.1
appname	Name of your application.	Nielsen Sample App

Variable Name	Variable Description	Example
sfcode	Nielsen collection facility to which the SDK should connect.	uat-cert

Optional Parameters that can be passed during init:

Variable Name	Variable Description	Example
longitude	GPS location. May be overwritten by SDK.	123.22
latitude	GPS location. May be overwritten by SDK.	-223.32
dma	Nielsen Designated Market Area code. DMA must be the Nielsen DMA value. Please check with the Technical Account Manager.	123
ccode	If Country Code is provided by your app, it should be done after consultation with your Technical Account Manager.	456

NOTE

- **appid**: You will receive two appids.
  - Test AppID

    The application must use the test AppID during the development, test, and certification processes.
  - Production AppID

    You should only use the production AppID when the app has completed the certification process with Nielsen and is ready to submit to the App Store.
- **sfcode**: sfcode identifies the Nielsen collection facility to which the SDK should connect.

mTVR/DPR

- For production, the sfcode is “us”.
- For development purposes, “uat-cert”.

DRM

- For production, the sfcode for DRM is “drm”.
- For development purposes, “uat-cert”.

3.1.2. Handling Special Characters in JSON Metadata

Character	Character Name	NOTES
\b	Backspace (ASCII code 08)	Do not include this in JSON to SDK.
\f	Form feed (ASCII code 0C)	Do not include this in JSON to SDK.
\n	New line	Do not include this in JSON to SDK.
\r	Carriage return	Do not include this in JSON to SDK.
\t	Tab	Do not include this in JSON to SDK.
\’	Apostrophe or single quote (Only valid in single quoted json strings)	Do not include back slash when passing apostrophe to SDK.
\”	Backslash quote (only valid in double quoted json strings)	Include back slash when passing double quote to SDK.
\\	Backslash character	Use double back slash when passing one back slash to SDK.

Example:

```
{
"type": "\"radio\"",            // To send “radio” in
metadata string
"assetid": "WXYZ-FM'101",
"stationType": "3",
"provider": "\\iHeartRadio"    // To send "\iHeartRadio" in
metadata string
}
```

3.1.3. SDK init JSON – Example

- JSON with basic application information

```
{
"sfcode":"us"
"appid":"18A5E427-1970-47E1-BF7D-7575235359A2",
"appname":"Nielsen SDK Sample QA",
```

```
"appversion":"app.3.1.1"
}
```

- JSON with optional Longitude and Latitude

```
{
"sfcode":"us",
"appid":"18A5E427-1970-47E1-BF7D-7575235359A2",
"appname":"Nielsen Sample App",
"appversion":"app.3.1.1",
"longitude":"123.22",
"latitude":"-223.32",
“dma”:”123”,
“ccode”:”456”
}
```

3.1.4. AppSDK Initailization and App Going into the Background/Foreground

Android

Figure 1 – Android AppSDK, Background/Foreground for Video Apps

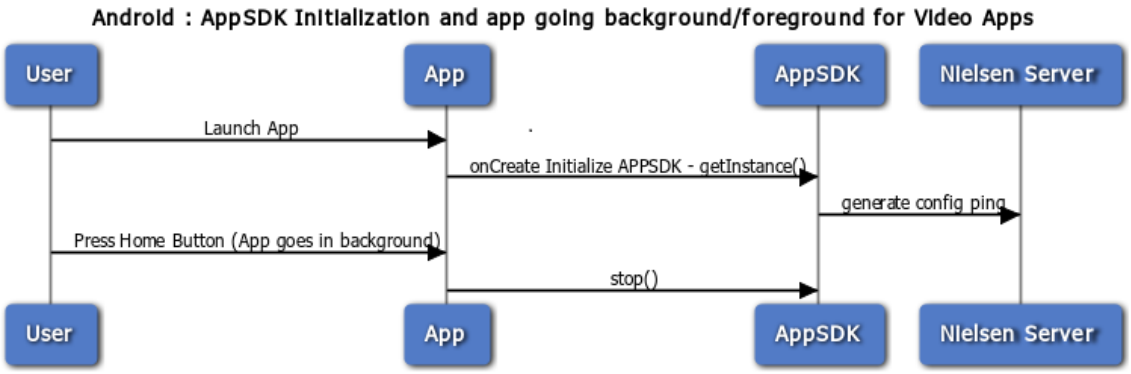
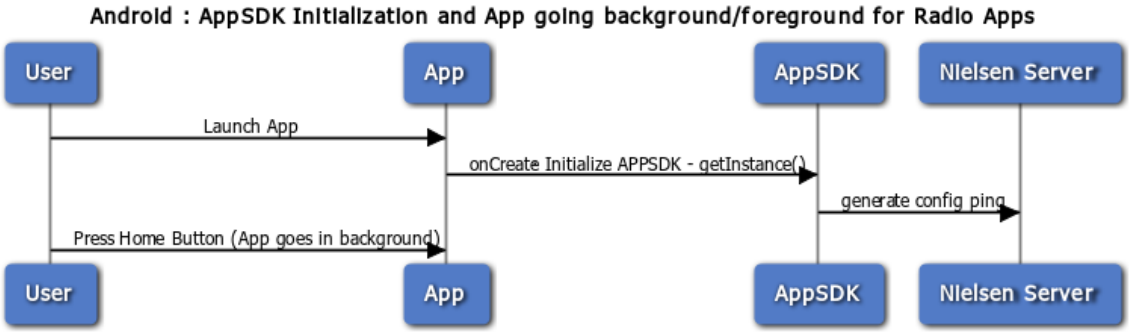
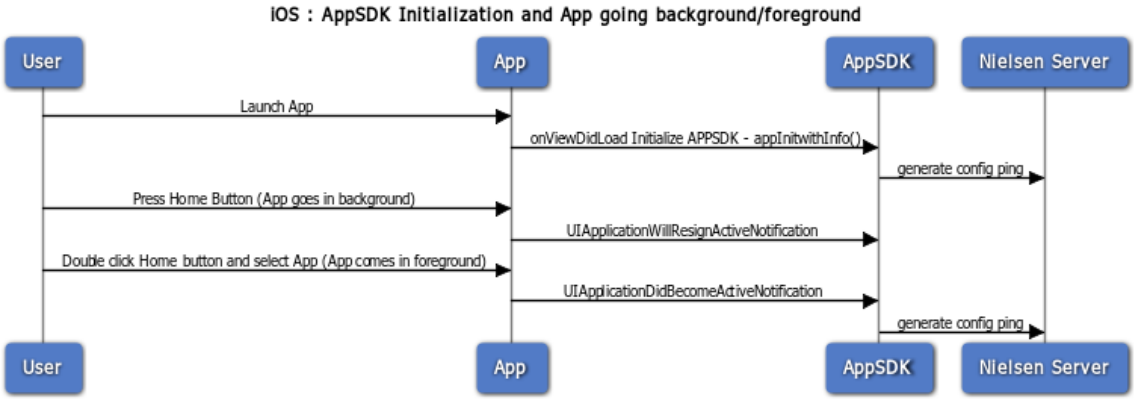


Figure 2 – Android AppSDK, Background/Foreground for Radio Apps



iOS

Figure 3 – iOS AppSDK, Background/Foreground for Video/Radio Apps



3.2. mTVR (Mobile TV Ratings)

For mTVR measurements, the client relies on Nielsen ID3 tags. The Channel Name field is a 32-character free-form text field containing the name of the program or feed being sent (such as ESPN2, Food Network, Breaking Bad, etc.) that must be inserted on a JSON string.

Variable Name	Variable Description	Example
channelName	Name of the channel or channel URL.	ESPN2
adModel **	Identifier to add model uses by the station. 0 - Default: Crediting is based on ad model break-out from the ID3 tag. 1 – Linear: Content will have the same linear ads 2 – Dynamic: Content will have ads dynamical served and not the same as linear ads.	1
** Only applicable for SDK 1.1 and above.		

Use Case 1: Content is ID3-tagged and content will have the same linear ads

- call play api with channelName JSON as below

```
{
        "channelName": "ESPN2"
```

```
}
```

- call loadMetadata api with JSON metadata as below

```
{
”adModel”: ”1”
}
```

- call sendID3 api with the ID3 payload popped-out during playback

3.3. DPR (Digital Program Ratings)

DPR measurement content is originated for digital-only or linear TV content with dynamic ad-model. Players should retrieve these data items from the Content Management System. A proper JSON string must be created to pass this information via the SDK’s loadMetaData: method.

Usage: TV App		
Variable Name	**Variable Description**	**Example**
dataSrc**	Source of the data. If content is coming from Content Management System and content is not ID3 tagged, dataSrc should be specified as ”cms”. If content is ID3 tagged.dataSrc should be specified as”id3”.	cms
type	Type of video content (preroll, midroll, postroll, content)	Content
assetid	Unique ID of content	vid-123
tv	JSON for TV **content** should be true. **Note** "true" for TV related contents (which was broadcasted on TV). "false" for the non-TV content (youtube videos etc.) Default value: "false"	true
category	Program name	NCIS
title	Episode title	S2, E1
length	Length of content in sec	3600
** Only applicable for SDK 1.1 and above.		

Use Case 1: Content is coming from Content Management System and content is not ID3-tagged

- call play api with channelName JSON as below

```
{
   ”channelName”: ”ESPN2”
}
```

- call loadMetadata api with JSON metadata as below

```
{
        ”dataSrc”: ”cms”,
        "type”: "content",
        "assetid”: "vid-123",
        "tv”: "true",
        "title":"My Episode Title",
        "category":"MyProgram",
        "length":"3600"
}
```

- call setPlayheadPosition api every 2 secs

Use Case 2: Content is ID3-tagged and content will have dynamical served ads (overidding the National ad-load)

- call play api with channelName JSON as below

```
{
   ”channelName”: ”ESPN2”
}
```

- call loadMetadata api with JSON metadata as below

```
{
”dataSrc”: ”id3”,
”adModel”: ”2”
}
```

- call sendID3 api with the ID3 payload popped-out during playback

3.3.1. Playhead Position During Playing

Depending on the type of content streamed, provide the appropriate playhead position value.

- If streaming live content, the client must pass the current UTC time in seconds.

- If streaming VOD (video on demand), pass the offset from the beginning of the file.

Content	Description	Example
Live	UTC of the live content.	Seconds since1970.
Video On Demand	Position is taken from beginning of the content in seconds.	Current player position from beginning of the content.

3.3.2. Live Content

- iOS

```
long long pos = [[NSDate date] timeIntervalSince1970];
[nAppApiObject playheadPosition:pos];
```

- Android

```
Calendar c = Calendar.getInstance();
long pos = (c.getTimeInMillis()/ 1000);
mAppSdk.setPlayheadPosition(pos);
```

3.3.3. On-demand Content

- iOS

```
CMTime curTime=[player currentTime];
long pos=CMTimeGetSeconds(curTime);
[nAppApiObject playheadPosition:pos];
```

- Android

```
long pos = mPlayer.videoPosition() / 1000;
mAppSdk.setPlayheadPosition(pos);
```

3.4. OCR (Online Campaign Ratings)

OCR tags are used to track commercial delivery for campaign ratings. OCR tags are received during the streaming session from the ad network or exchange for tracking campaign for video or static ads. This information should be passed as a JSON string via the loadMetadata: method with OCR contents.

Usage: Online Campaign		
Variable Name	**Variable Description**	**Example**
type	Type identifies the tag content type. For OCR, the data type should be always set to “ad”.	ad
ocrTag	The complete tag/URL is	http://secure-aws.imrworldwide.com/cgi-bin/m?ci=

Usage: Online Campaign		
Variable Name	**Variable Description**	**Example**
	supplied by your Technical Account Manager. This should include the complete URL including the http portion. The ocrtag should be properly URI encoded.	ENTXX5&am=3&ep=1&at=view&rt=banner&st=image&ca=XX1717&cr=crvXX35&pc=plc1234

3.4.1. Typical JSON of OCR Metadata

- JSON for OCR

```
{
     "type”: "ad",
   "ocrtag”: “http://secure-aws.imrworldwide.com/cgi-bin/m?ci=ENT29825&am=3&ep=1&at=view&rt=banner&st=image&ca=jjjj1717&cr=crv194435&pc=plc12340”

  }
```

3.4.2. Introduction – Nielsen OCR Tag

Nielsen OCR tag or beacon may come in different forms from an ad service via VAST XML or an ad service integrated player framework library:

1. A standalone http string:

   http://secure-us.imrworldwide.com/cgi-bin/m?ci=ent30986&am=22&ep=1&at=view&rt=banner&st=image&ca=cmp97144&cr=1186239&pc=3739659&c13=C8CFD744-1C79-4F77-A014-B73B6552E37B&r=2011370876

2. A http string data field in a larger http string: (value of CR field below)

   http://cue.v.fwmrm.net/ad/l/1?s=a029&n=171224%3B171224&t=1390846739204167007&f=&r=171224&adid=3575811&reid=2448362&arid=0&auid=&cn=defaultImpression&et=i&_cc=3575811,2448362,,,1390846739,1&tpos=0&iw=&uxnw=&uxss=&uxct=&metr=121&init=1&cr=http%3A//secure-us.imrworldwide.com/cgi-bin/m%3Fci%3Dent30986%26am%3D22%26ep%3D1%26at%3Dview%26rt%3Dbanner%26st%3Dimage%26ca%3Dcmp97144%26cr%3D1186239%26pc%3D3575811%26c13%3C8CFD744-1C79-4F77-A014-B73B6552E37B%26r%3D1290866613

3. A http string wrap in CDATA of an xml file (for example, VAST XML):

   <![CDATA[http://secure-us.imrworldwide.com/cgi-bin/m?ci=abc2011-ca&at=view&rt=banner&st=image&ca=ABC2014&cr=2121321&pc=3826745&ce=g492154&pr=iag.sid,2500013149&pr=iag.tfid,5181&pr=iag.pageid,3826745&pr=iag.brn,294227&pr=iag.cte,2121321&pr=iag.cmpid,ABC2014&pr=iag.stid,g492154&c13=C8CFD744-1C79-4F77-A014-B73B6552E37B&r=1546128247]]>

4. Previous two cases combined:

   <![CDATA[http://cue.v.fwmrm.net/ad/l/1?s=a029&n=171224%3B171224&t=1390846739204167007&f=&r=171224&adid=3575811&reid=2448362&arid=0&auid=&cn=defaultImpression&et=i&_cc=3575811,2448362,,,1390846739,1&tpos=0&iw=&uxnw=&uxss=&uxct=&metr=121&init=1&cr=http%3A//secure-us.imrworldwide.com/cgi-bin/m%3Fci%3Dent30986%26am%3D22%26ep%3D1%26at%3Dview%26rt%3Dbanner%26st%3Dimage%26ca%3Dcmp97144%26cr%3D1186239%26pc%3D3575811%26c13%3C8CFD744-1C79-4F77-A014-B73B6552E37B%26r%3D1290866613]]>

3.4.3. Steps to Integrate mOCR with AppSDK

After getting a string from an ad service to integrate the Nielsen SDK for an OCR tag, the app should first identify a Nielsen beacon in the use cases mentioned in Section 3.4.2 above:

1. If the input string is a CDATA string, get the value of the CDATA as a new input string.
2. If the input string (or new input string) starts with http and the hostname ends with “imrworldwide.com”, the input string is a Nielsen beacon (case 1 and case 3) in Section 3.4.2.
3. If the input string (or new input string) starts with http and the CR field exists:
   a. Get the value of the CR field.
   b. If the CR value starts with http and the hostname of the CR value ends with “imrworldwide.com”, the CR value is a Nielsen beacon (case 2 and case 4).
   c. If the Nielsen beacon exists, remove the value of the CR field from the input string and process the updated string for a non-OCR beacon.
4. If a Nielsen beacon exists:
   a. Have the URL decode the Nielsen beacon.
   b. If the c13 field exists in the Nielsen beacon, remove the c13 from the Nielsen beacon.
   c. Create a JSON with the Nielsen beacon:

   JSON:

```
{
      “type”:”ad”,
      “ocrtag”:”<Nielsen beacon>”
}
```

Sample JSON string with the OCR string (decoded http string without the c13):

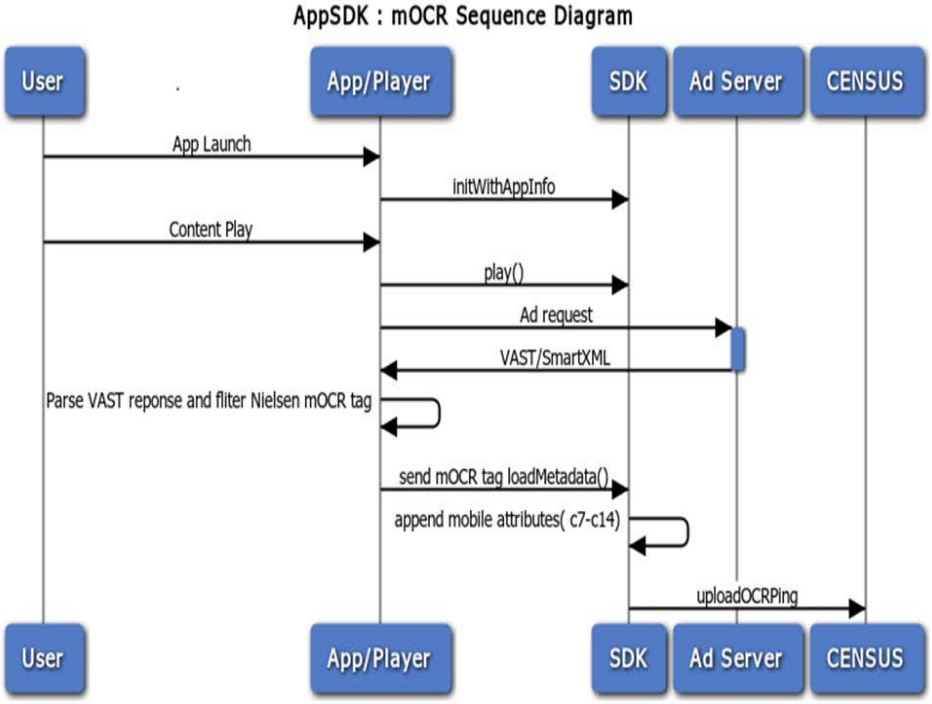
```
{
      “type”:”ad”,
“ocrtag”:”http://secure-us.imrworldwide.com/cgi-
bin/m?ci=ent30986&am=22&ep=1&at=view&rt=banner&st=image&ca=
cmp97144&cr=1186239&pc=3739659&r=2011370876”
}
```

d. Send the JSON to the Nielsen SDK using loadMetadata.

5. If the Nielsen beacon does not exist, process the input string for the non-OCR beacon.

3.4.4. mOCR Sequence Diagram

Figure 4 – AppSDK mOCR Sequence Diagram



3.5. DRM (Digital Radio Measurement)

Radio clients should provide following parameters for Digital Radio Measurement.

Usage: Radio App		
Variable Name	**Variable Description**	**Example**
dataSrc **	Source of the data. For DRM datasrc should be passed as	Cms

Usage: Radio App		
Variable Name	**Variable Description**	**Example**
	“cms”.	
type	Type of content . For DRM mention type as “radio”	Radio
assetid	Station identifier.Should include Call letters and Band.	WXYZ-FM
stationType	OTA station flag and/or OTA station type. 0: Custom station built per user. 1: OTA Streaming station with same ad load. 2: OTA station with different ad load. 3: Multicast eRadio or online station. 4. On Demand Audio(Podcasting)	3
provider	Name of Provider	iHeartRadio
** Only applicable for SDK 1.1 and above.		

3.5.1. Typical JSON of Content Metadata

- JSON for radio content

```
{
        ”dataSrc”: ”cms”,
        "type”: ”radio”,
        ”assetid”: ”WXYZ-FM”,
        ”stationType”: "3",
        ”provider”: ”iHeartRadio”
}
```

3.5.2. Playhead Position During Playing

Depending on the type of content streamed, provide the appropriate playhead position value.

- If streaming live content, the client must pass the current UTC time in seconds.
- If streaming on demand station or content, pass the offset from the beginning of the file.

Content	Description	Example
Live	UTC of the live content.	Seconds Since1970

Content	Description	Example
Video On Demand	Position is taken from beginning of the content in seconds.	Current player position from beginning of the content.

3.5.3. Live Content

- iOS

```
long long pos = [[NSDate date] timeIntervalSince1970];
[nAppApiObject playheadPosition:pos];
```

- Android

```
Calendar c = Calendar.getInstance();
long pos = (c.getTimeInMillis()/ 1000);
mAppSdk.setPlayheadPosition(pos);
```

3.5.4. On-demand Content

- iOS

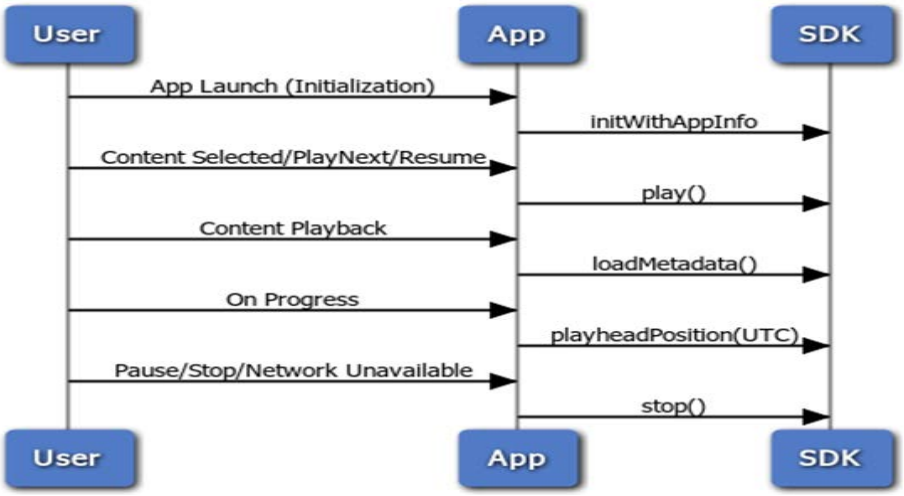
```
CMTime curTime=[player currentTime];
long pos=CMTimeGetSeconds(curTime);
[nAppApiObject playheadPosition:pos];
```

- Android

```
long pos = mPlayer.videoPosition() / 1000;
mAppSdk.setPlayheadPosition(pos);
```

3.6. Nielsen App SDK Call Sequence Diagram

Figure 5 – Nielsen App SDK Call Sequence Diagram



3.7. Nielsen Measurement Opt-Out

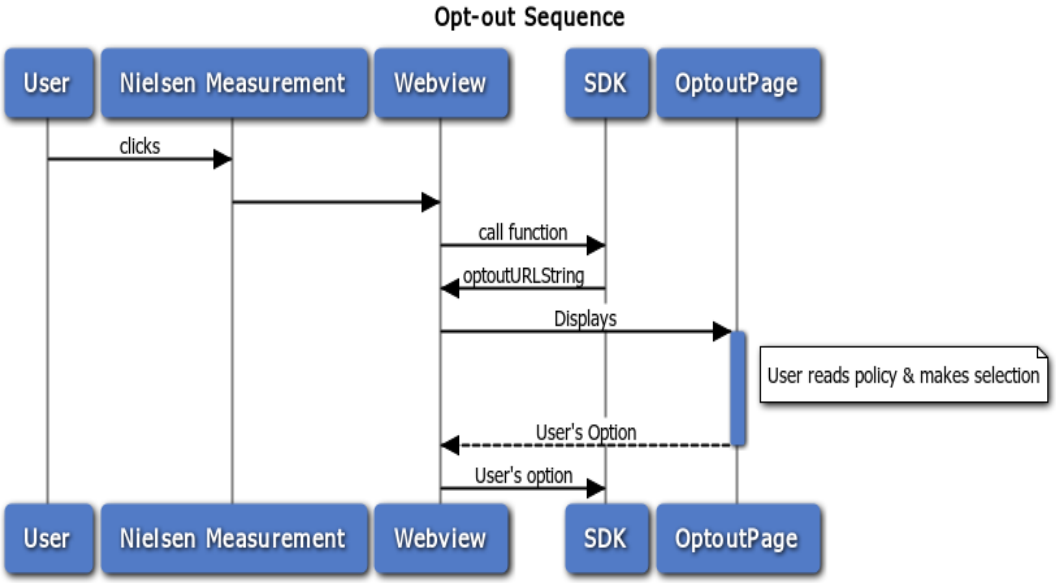
The app must provide a means for the user to opt-out, or opt back in of Nielsen Measurement. A user can use this feature if they would prefer not to participate in any of Nielsen's online measurement research.

To opt-out, users must be able to access Nielsen's "About Nielsen Measurement" page from within the app. The URL to this webpage should be called from the SDK and opened in a webview within the app, not within Safari or Chrome outside of the app.If the AppSDK returns null as opt-out URL, please handle it and retry later.

The opt-out occurs by opening a Nielsen-defined web page and passing the user choice from the webview commands. In order to do this, the application needs to implement the UIWebView delegate method to open the Nielsen Privacy web page, capture the user selection, and pass the user selection back to the SDK via the userOptOut method.

Note When the webview is closed, the status returned from the webview must be passed to the SDK within the app. The SDK manages the user's choice (opt-out/opt-in) so this status is not required or needed to be manage by the app.

Figure 6 – Opt-Out Sequence



3.7.1. Sample

iOS

- Present webview with Nielsen opt-out URL

```
NSString *optOutURL = [nAppApiObject optOutURLString];
If(optOutURL == nil)
{
  //Handle it gracefully and Retry later
```

```
}
else
{
self.optOutView = [[UIWebView alloc]
initWithFrame:self.view.bounds];
self.optOutView.delegate = self;
self.optOutView.scalesPageToFit = YES;
[optOutView loadRequest:[NSURLRequest
requestWithURL:[NSURL URLWithString:optOutURL]]];
[self.view addSubview:optOutView];
}
```

- Capture and forward user selection

```
-(BOOL)webView:(UIWebView *)webView
shouldStartLoadWithRequest:(NSURLRequest *)request
navigationType:(UIWebViewNavigationType)navigationType
{
NSString *command = [NSString stringWithFormat:@"%@",
request.URL];
if ([command isEqualToString:kNielsenWebClose]) {
// Close the web view
[self performSelector:@selector(closeOptOutView)
withObject:nil afterDelay:0];
return NO;
}
// Retrieve next URL if it’s not opt-in/out selection
return (![nAppApiObject userOptOut:command]);
}
```

Android

- Create a web view with Nielsen opt-out URL

```
optOutUrl = mAppSdk.userOptOutURLString();
if(optOutUrl){
mWebView = (WebView) findViewById(R.id.webView);
mWebView.getSettings().setJavaScriptEnabled(true);
mWebView.getSettings().setBuiltInZoomControls(true);
mWebView.getSettings().setDisplayZoomControls(false);
mWebView.getSettings().setLoadWithOverviewMode(true);
mWebView.getSettings().setUseWideViewPort(true);
mWebView.getSettings().setLayoutAlgorithm(LayoutAlgorithm.S
INGLE_COLUMN);
mWebView.setWebViewClient(new MonitorWebView());
mWebView.setWebChromeClient(new WebChromeClient());
mWebView.loadUrl(optOutUrl);
}else{
```

```
//Handle it gracefully and Retry later
}
```

- Capture and forward user selection

```
private class MonitorWebView extends WebViewClient
{
   public void onPageFinished(WebView view, String url)
   {
      cancelDialog();
   };
   public void cancelDialog()
   {
      if (dialog != null)
      {
         dialog.cancel();
         dialog = null;
      }
   }
   @Override
public boolean shouldOverrideUrlLoading(WebView view,
String url)
   {
      if (url.indexOf("nielsen") == 0)
      {
         mAppSdk.userOptOut(url);
         finish();
         return false;
      }
      else
      {
         dialog = ProgressDialog.show(OptOutActivity.this,
         "OptOut", "Loading...");
         return true;
      }
   }
}
```