nielsen

Engineering

Nielsen App SDK 3.2

Developer's Guide

Revision History

Revision	Date	Change Made	Responsible Engineer
1	01/27/2012	Initial version, combined Nielsen ID3Meter API and Concept of Operations documents	Alan Bosworth Marcos Male Mridula Gudluru Scott Cooper
2	03/20/2012	Additional edits, additional information on deep link URLs. Added appendix.	Scott Cooper
3	04/22/2013	Added enhancements to support ‘in app’ recruitment, Quick Start guide, Nielsen On-Boarding process, etc.	Freny Ann Varghese Nidhish Ramachandran Huey Yu Scott Cooper
4	10/24/2013	Updated document to cover transition from ID3Meter SDK to Nielsen App SDK.	Nidhish Ramachandran Huey Yu Martti Juotasniemi Scott Cooper
5	11/05/2013	Updated document for R1 compatibility.	Scott Cooper
6	11/21/2013	Additional edits for OCR & DPR CMS data.	Nidhish Ramachandran Scott Cooper
7	02/14/2014	Updated playheadPos string.	Huey Yu
8	6/19/2014	Added support for v 1.1 of SDK	Nidhish Ramachandran Huey Yu
9	09/29/2014	Added support for v.1.2 of SDK	Sanjana Murlidhar

Contents

List of Figures

List of Tables

1. Introduction

Nielsen's App SDK (Software Development Kit) is a framework for measuring linear (live) and on-demand TV viewing using mobile devices such as Apple iOS devices (iPads®, iPhones®, iPod Touch®) or Android® devices (tablets and cell phones). The Nielsen App SDK leverages Nielsen's audio watermark technologies for TV audience measurement, the industry supported ID3 metadata tag specification, and Nielsen's Combined Beacon technology. Nielsen's App SDK is integrated into MVPD/CDN/CO (Content Originator) video player applications (VPAs) so that streaming measurement can be provided on mobile devices.

This document covers the following:

- Nielsen App SDK Concept of Operations
- Nielsen App SDK Quick Start and Metadata Guide
- Nielsen App SDK Developer On-Boarding Process
- Nielsen App SDK Framework & API
- Nielsen App SDK Sample Application
- Nielsen Privacy Requirements

2. Concept of Operations

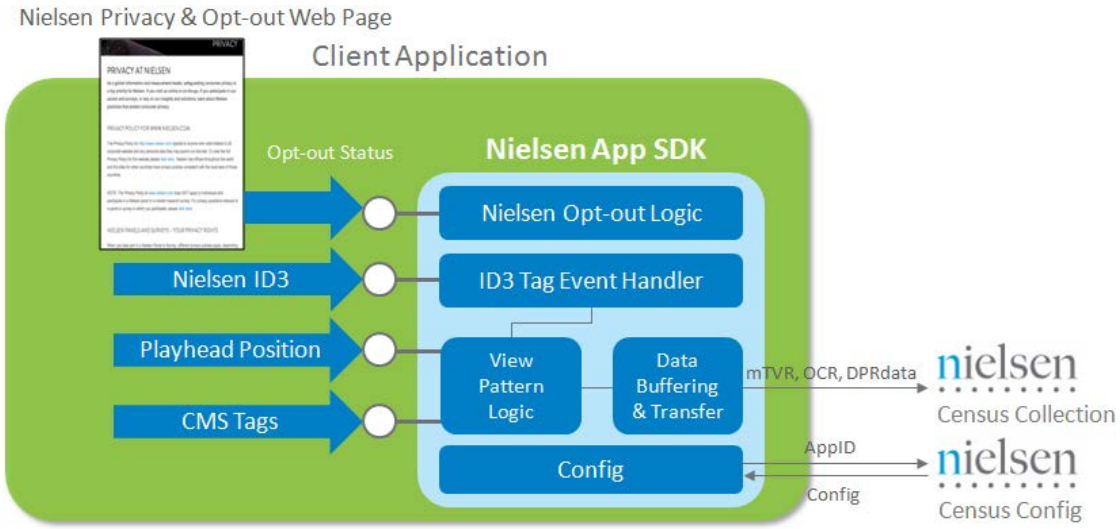
2.1. Purpose

The purpose of this section is to explain the concept of operations behind Nielsen's App SDK measurement framework and provide integration details of software components found within the SDK.

2.2. Description of the System

Figure 1 shows the high-level components that comprise Nielsen's App SDK software.

Figure 1 – Nielsen App SDK High Level Components



The Nielsen App framework does the following things:

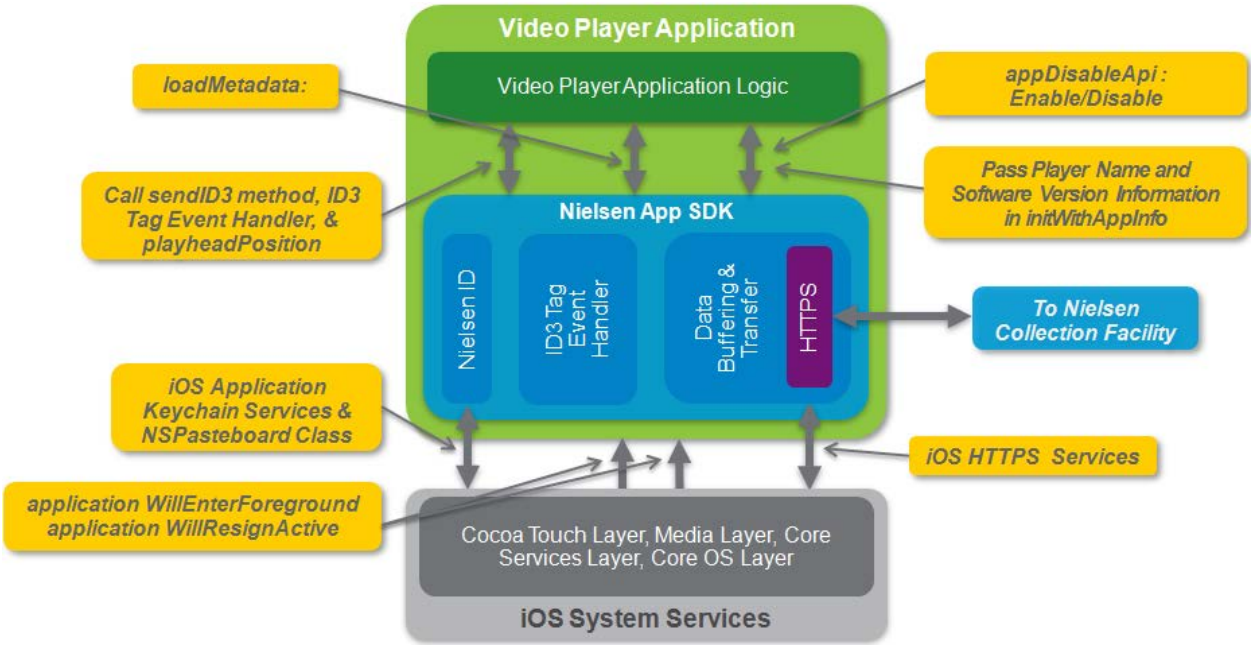
- Creates and stores a Nielsen Unique ID (NUID) token locally. The NUID token is unrelated to Apple's UDID identifier.
- Retrieves a configuration file from Nielsen configuration servers based on the Application ID (AppID) assigned to the application.
- Processes ID3 tags and CMS tags (DPR or OCR) received during video streaming sessions, and discards non-Nielsen specific information (for example, 3rd-party ID3 tags).
- Buffers Nielsen ID3 / CMS tags and transfers them to Nielsen's collection facility.
- Supports ID3 / CMS tag caching when users view downloaded content in situations where there is no network connectivity. Cached ID3 / CMS tags are transferred to Nielsen's collection facility when connectivity returns, however, CMS tags will not be carried over to the next viewing session.
- Supports user opt-out selection - if the user opts out of measurement, the SDK will notify Nielsen's collection facility that the user has opted out of measurement and will stop sending measurement data to Nielsen.

2.3. How Does it Work?

Once integrated into a client's VPA player application, the operational flow of Nielsen's App SDK is as follows:

- Configuring the VPA for Nielsen measurement:

Figure 2 – Nielsen App SDK Key Components



- Handling Nielsen ID3 tags and viewing events:
  - Upon each launch of the Video Player Application (VPA), the VPA initializes a Nielsen App SDK singleton object and sets up an NSNotification Center ID3 tag event listener to handle HLS timed metadata events that come from the iOS media player object.
  - The App SDK component collects general application information, player position (playhead) to ensure accurate metering. The VPA must pass the following information to Nielsen's metering framework:
    - The name of the Video Player Application.
    - A unique application ID (AppID) assigned to you by Nielsen for your player app. Nielsen will assign two AppIDs for your application – a test appID and a production appID. The application must use the test AppID during the development, test, and certification processes. 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.
    - Software revision of the VPA.

- If the video content (programming or advertising) supports the use of CMS tags, these tags are to be supplied to the SDK using the SDK's loadMetadata: method.
- Streaming session start events, referred to as "play: channelName" events
- Video playheadPosition must be sent to the SDK once every two seconds if you are using the SDK for Nielsen's DPR and DRM products.
- Streaming session stop events, referred to as "stop" events

o The Nielsen App framework requires access to the following iOS system services in order to function properly:

- Application DidBecomeActive notification – This system notification will cause the App SDK framework to instantiate its class objects and retrieve its config file from Nielsen's configuration servers and to be ready to process viewing data.
- Application WillResignActive notification – This notification causes Nielsen's App SDK framework to transmit all unsent messages to Nielsen's collection facility and perform general housekeeping chores before going to the background; this includes ending the viewing session, freeing up memory, etc.
- iOS Application Keychain Services and iOS generalPasteboard of the UIPasteboard Class – These two system services are used to persist the Nielsen ID on the device in the event the user deletes/re-installs the VPA player app and to share the Nielsen ID with other Nielsen-enabled player apps.
- iOS HTTPS functions – Nielsen's App SDK uses HTTPS to securely transfer streaming initial configuration data to Nielsen's collection facilities.
- Persistent memory – Persistent memory is used to store App SDK state information, variables, unsent data, etc. The SDK is capable of caching ID3 tag data for up to 10 days. In extreme cases, the SDK will use about 30MB of persistent memory to cache unsent data. Once network availability is back, the SDK will retry sending the cached data.

o When the user starts viewing media from an HLS HTTP Adaptive Streaming session, the iOS operating system detects timed ID3 metadata events and invokes Nielsen App SDK's sendID3 method with the owner ID string in the PRIV frame of Nielsen's ID3 tag. Nielsen's sendID3 method validates the owner ID string, processes its data, and queues data for transfer to Nielsen's collection facility.

Note	VPA is responsible of obtaining owner ID string in the PRIV frame of ID3 tags. It is not necessary for the VPA to call the appDisableAPI method to enable/disable Nielsen metering. We recommend that you only call this method when you want to completely disable measurement.

- Data transfer to the Nielsen collection facility:
  - The transfer of viewing data to Nielsen's collection facility is done using a HTTPS secure sockets connection. Data transferred to Nielsen's collection facility include: Nielsen ID3 tags, channelName descriptive information including CMS tags for DPR, DRM, VC, IAG, or OCR, viewing patterns, and general VPA application information (VPA player name, software revision, Nielsen App SDK revision, etc.).
  - All data collected and transferred to Nielsen's collection facility is carefully protected to ensure the anonymity of the user is preserved and the integrity of the Nielsen sample is maintained.

3. Quick Start Guide and Metadata Guide

There are 4 steps you must go through in order to deploy Nielsen-enabled applications with Nielsen's App SDK. For more detailed technical information, see Section 5 of this document.

The following data elements must be in place before your application can send data to Nielsen.

Table 1 – Data Elements

Data Item	Description
appid	The `appid` is provided to you by Nielsen's Technical Account Manager. You will receive two appIDs; a test appID and a production appID. The developer must use the test AppID during the development, test, and certification process. You should only use the production AppID when the app has complete process with Nielsen and is ready to submit to App Store.
sfcode	`sfcode` specifies which Nielsen collection facility the app should connect to. During development you will be connecting to a development server.
dma	If your app is supplying the `dma`, it should be done after consultation with your Technical Account Manager.
ccode	If your app is supplying the country code, it should be done after consultation with your Technical Account Manager.
CMS Metadata	Unique CMS metadata details must be discussed with your Technical Account Manager prior to launch.

3.1. Setting up your Xcode Development Environment

Ensure that you unzip the Nielsen App SDK sample app and copy the "NielsenAppApi.Framework" to the Frameworks folder of your Xcode VPA player project. Also, add the following import file to your View Controller's header file:

```
#import <NielsenAppAPI/NielsenAppAPI.h>
```

Ensure the Nielsen App SDK is included in the Linked Frameworks and Libraries list (located in the project's Summary settings). Also, verify the iOS security framework is listed in the Linked Frameworks and Libraries list, as Nielsen's metering framework makes use of a number of functions in this library.

In order to create an instance of the Nielsen App SDK object, the application has to also import the following frameworks and libraries:

```
<UIKit/UIKit.h>
<Foundation/Foundation.h>
<AdSupport/AdSupport.h>
<SystemConfiguration/SystemConfiguration.h>
<Security/Security.h>
<libsqlite3.dylib>
```

Nielsen's SDK includes C++ modules. As such, the application will need to add **–lstdc++** to the "Other Linker Flags" in the Xcode Build Settings (or, the program file that imports the SDK header may need to have .mm as its file extension or set Objective-C++ Source as its file type in Xcode).

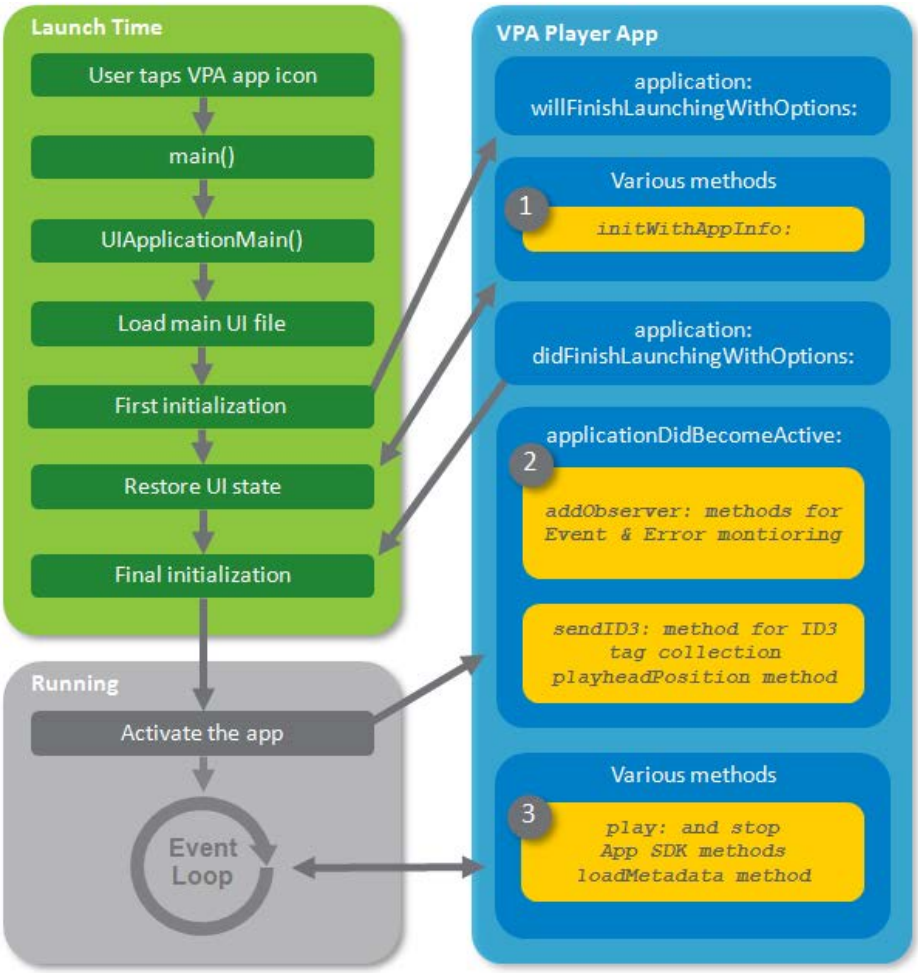
3.2. Step 1: Initializing the Nielsen App SDK Object

The first thing you must do is initialize a Nielsen App SDK object by calling the `initWithAppInfo:` method. VPA player parameters (`appid, appversion, appname, sfcode`) must be passed with this method via the AppInfo JSON schema described in Appendix A. The appID value is obtained from Nielsen operational support and is unique to your VPA player application.

Note

The `appID` is provided to you by Nielsen's Technical Account Manager (TAM). The appID is a GUID data type and is specific to your application. The appID has the following type of format: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX. The developer must use the test `appid` during the development, test, and certification process.

Figure 3 – Apple's iOS Application Life Cycle



3.3. Step 2: Making Connections to the App SDK Object

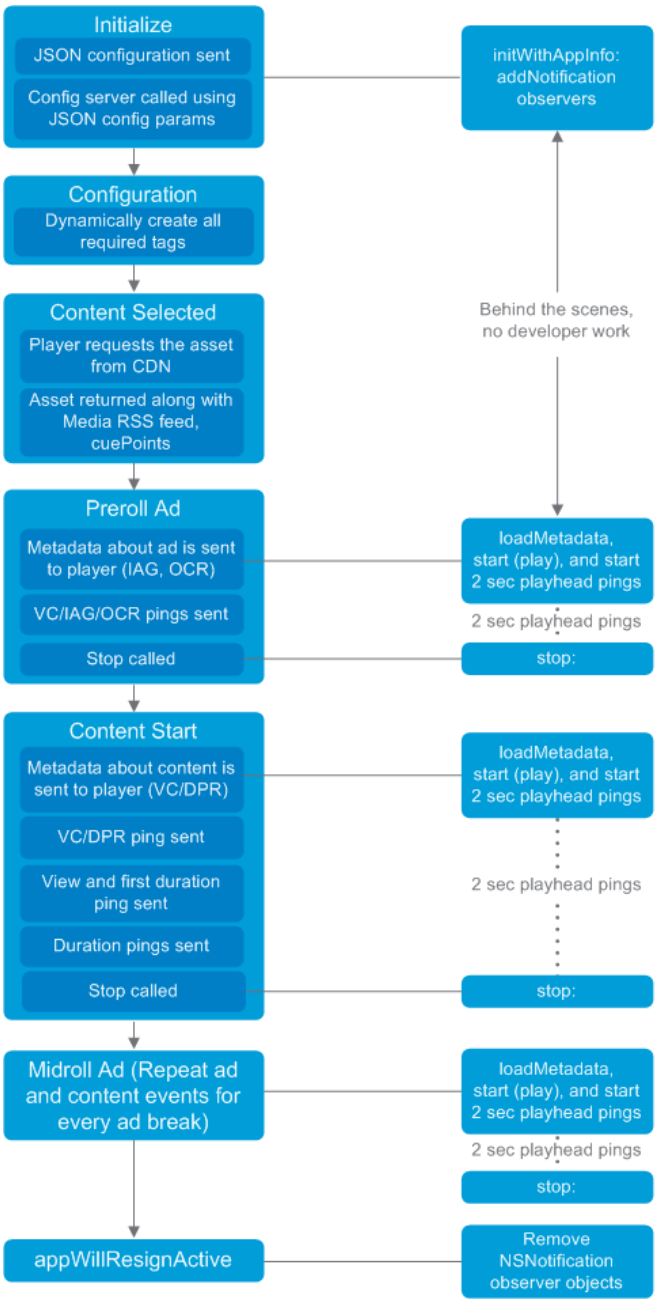
Once the NielsenAppApi object has been initialized, the next step is to enable notifications. The SDK makes use of iOS's Notification Center and NSNotification Class methods for:

- Receiving event and error messages from the SDK framework
- Collecting HLS timed metadata events (ID3 tags) during viewing sessions. As such, you will need to set up a timed metadata event listener method for receiving ID3 tags and calling Nielsen's `sendID3:` method. You will also need to add NSNotification Center observers to your application using the `addObserver:`method to receive event/error information from the `kNol_notifyAppApiEventStatusUpdate:` and `kNol_notifyAppApiErrorStatusUpdate:` selector methods.

3.4. Step 3: Nielsen App SDK Streaming Sessions

After you have set up observers for SDK events/errors and a listener method to process incoming Nielsen ID3 tags, the next step is to load CMS metadata using the `loadMetadata:` method, call the `play:channelName` method when starting a streaming session, and the `stop` method when ending a viewing session. During session play, call the `playheadPosition:` method once every two seconds until the stream is stopped. Nielsen's App SDK object handles filtering of ID3 tags and CMS tags, queuing/buffering of data, and all communications with Nielsen's collection facility.

Figure 4 – Nielsen DPR Streaming Session Lifecycle



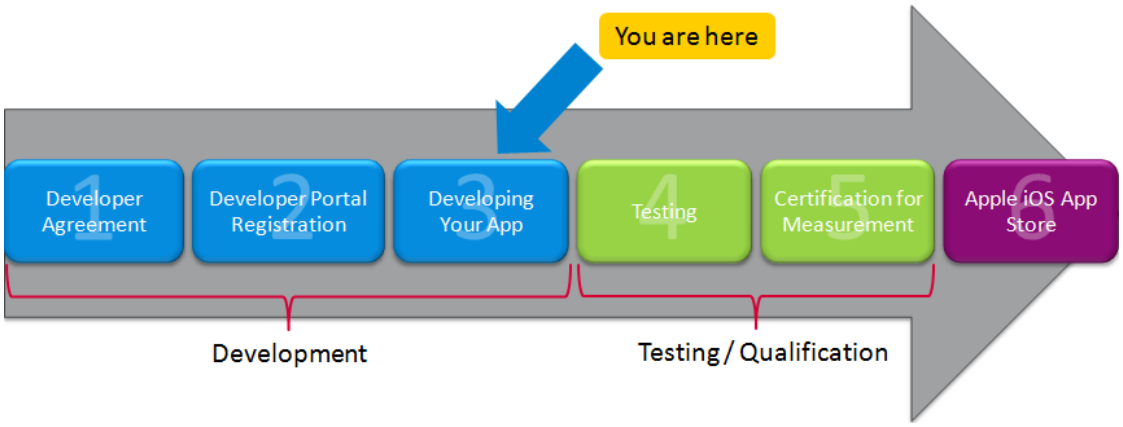
3.5. Step 4: Tying up the Loose Ends

Ensure that you remove NSNotification observer objects using the `removeObject:` method when you clean up and/or prepare to go to iOS's background mode.

4. Nielsen App SDK On-Boarding Process

4.1. Steps to Releasing a Nielsen-enabled Application

Figure 5 – Nielsen On-Boarding Process Timeline



The following steps are necessary to deploy Nielsen-enabled applications:

1. Client signs the Nielsen App SDK Evaluation Agreement and sets up a developer account via Nielsen's Mobile Measurement Developer Registration Portal.
2. Client fills out form on Registration Portal and submits it to Nielsen.
3. Nielsen's Mobile Measurement team verifies that legal paperwork is in order and provides FTP account information (username & password) to you for downloading the Nielsen App SDK. The Mobile Measurement team also provides Nielsen-assigned appids to you for use in your iOS player application.

> **Note** Clients only need a single agreement and single download of the Nielsen App SDK. Each application that integrates the Nielsen App SDK, however, will need to have its own set of Nielsen-assigned `appIDs`; a Test `appID` and Production `appID`.

4. Client develops the player app with the App SDK and sets the `appID`, the application name (appname), and assigns a software version (appversion) to the app. Client uses the test appID during development, test, and certification.
5. Once the application is certified and CMS variable mapping for DPR, VC, IAG, and OCR data has been confirmed by your Nielsen Technical Account Manager, your app is ready for the measurement. Change the `appID` to the production `appID` and prepare to submit your application to the App Store for distribution.

6. Submit player application to the Apple Store.

5. SDK Framework

5.1. Purpose

This section of the Nielsen App SDK Developer's guide describes the Objective-C API used to interface to Nielsen's App SDK. This framework provides Nielsen measurement services to video streaming applications. By integrating the SDK into a VPA player application, clients are able to collect streaming media consumption data for Nielsen mobile device measurement.

5.2. Overview of the Nielsen App SDK

The Nielsen App SDK allows the calling application to pass data that Nielsen uses to accurately measure audience viewing. The VPA app must also pass ID3 tags and application information, such as the VPA application's name, version, and `appid`. The Nielsen App object automatically relays this information to Nielsen's collection facility.

Nielsen's App SDK is compatible with Apple iOS versions 6.0+.

Note	While Nielsen's App SDK is compatible with Apple's iOS 5.x, only iOS versions 6.0+ are qualified for measurement due to Apple IDFA changes.

The following is an example of how the Nielsen App SDK is used:

1. Instantiate (**init**) the Nielsen App object within the *viewDidLoad* view controller delegate method using the `initWithAppInfo:` method. The `initWithAppInfo:` method passes the `appname`, `appid`, `appversion`, and `sfcode` values as arguments via the AppInfo JSON schema.
2. At the start of each streaming session, call the *play:* method and use the `channelInfo` parameter to pass channel descriptor information. 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. See Section 5.4.3 for more information.
3. Load CMS data using the `loadMetadata:` method at the start of each asset being played.
4. During a timed metadata event, use the `sendID3:` method to pass the data object to the Nielsen SDK object for processing.
5. Call `playheadPosition` once every two seconds to pass content view position to the SDK object.

6. At the end of each streaming session, channel change, or stoppage of playing for any reason, call the `stop` method. Although the Nielsen App object does observe didBecomeActive and willResignActive, it cannot trap the user's action within the application. So, the app must notify the Nielsen App object via the `stop` or `play:` method call that the player within the app is "in or out of view", such as for occurrences of the viewDidDisappear or viewDidAppear events.
7. Release the Nielsen App object within the `dealloc` viewController delegate method. This allows the Nielsen App object to do housecleaning operations before moving to the background. Also, ensure that you remove NSNotification observer objects using the `removeObject:` method when you clean up and/or prepare to go to iOS's background mode.

5.3. Variable Name Mapping

Several SDK methods take JSON formatted strings as an input parameter. This means that the data is formatted as key-value pairs, the key being the variable name and the value being the string the key is set to. For example, a JSON string for setting variables "title" and "program length" to values MyEpisodeTitle and 3600 respectively would be as follows:

```
{\"title\" : \"MyEpisodeTitle\", \"length\" : \"3600\"}";
```

Now, consider a case where the player application was implemented using different variable names due to different naming conventions used within your CMS system. As an example, the player app might have implemented the example above as:

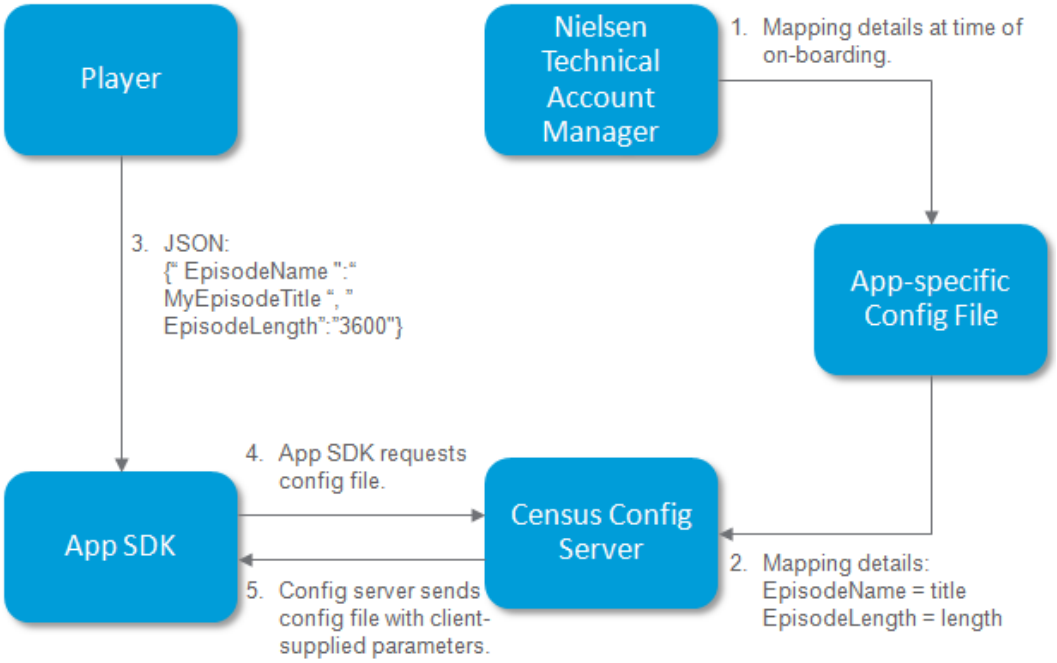
```
{\"EpisodeName\" : \"MyEpisodeTitle\", \"EpisodeLength\" :
\"3600\"}";
```

It is possible to support this option without modifying application code by using Variable Name Mapping. The SDK has functionality for supporting different variable names, but the mapping information must be provided to Nielsen through your Technical Account Manager (TAM).

The new variable name mapping will be updated in Nielsen's configuration server and passed to the SDK via an updated config file. Once changes are made, the SDK will know that "EpisodeName " is synonymous with "title" and "EpisodeLength" actually is the same as "length".

Variable name mapping gives flexibility to the client side implementation. It is recommended, however, that you use the Nielsen-defined names whenever possible to simplify long-term maintainability.

Figure 6 – Variable Name Mapping



1. Mapping details provided to Nielsen Technical Account Manager.
2. Config server handles the URL separately for each application based on initial parameter mapping.
3. Player App uses custom names in the JSON strings with whatever key values agreed upon during Nielsen's on-boarding process.
4. SDK requests config file upon initialization.
5. The config file is customized for each app.

For more information, contact your Nielsen Technical Account Manager.

5.4. Interfaces Provided by the SDK

This section provides a general description of the NielsenAppApi class and its public API.

5.4.1. NielsenAppApi Class Description

The NielsenAppApi class is the primary application interface to the Nielsen App SDK. For example, after an instance object of the NielsenAppApi class is created and initialized, it can be used by the calling application to collect HLS timed metadata using the SDK's `sendID3:` method.

These are the public methods and properties exposed by the NielsenAppApi class:

```
#import <NielsenAppApi/NielsenAppApi.h>

   @interface NielsenAppApi : NSObject {
         }

   - (id)initWithAppInfo:(NString *)appInfo;
   + (NSString *)getMeterVersion;

   - (void)loadMetadata:(NSString *)jsonMetadata;
   - (void) play:(NSString *)channelInfo;
   - (void) sendID3:(NSString *)data;
   - (void) playheadPosition:(long long)playheadPos;
   - (void) stop;

   + (NSString *)getLastEvent;
   + (NSDictionary *)getLastOccurredEvent;
   + (NSDictionary *)getLastOccurredError;

   - (BOOL)userOptOut:(NSString *)optOut;
   - (void)appDisableApi:(BOOL)apiDisable;
   - (NSString *)optOutURLString;

   @end
```

5.4.2. Nielsen App API Class Properties

The NielsenAppApi class inherits from the NSObject class and exposes no properties to the calling application. All configuration and operational settings are made through methods calls described in the following section.

5.4.3. Nielsen App API Methods

Overview

The NielsenAppAPI class methods provide the main interface to the Nielsen iOS device metering framework. This class creates a singleton object that allows you to enable and disable the meter, start and stop viewing sessions, and access status events or error information from the metering framework.

Tasks

Initializing a NielsenAppApi Object

`- initWithAppInfo`

Configuring the NielsenAppApi Object

- appDisableApi
- optOutURLString
- userOptOut

Working with a NielsenAppApi Object

- loadMetadata
- play
- playheadPosition
- sendID3
- stop

Accessing NielsenAppApi Log Information

- getLastEvent
+ getLastOccurredError
+ getLastOccurredEvent
+ getMeterVersion

Receiving ID3Meter Error/Event Notifications

- kNol notifyAppApiErrorStatusUpdate
- kNol notifyAppApiEventStatusUpdate

Instance Methods

initWithAppInfo:

Initializes the App SDK metering framework. The initializer for the App SDK has been enhanced with additional input parameters.

```
- (id)initWithAppInfo:(NSString *)jsonAppInfo
```

Parameters

`jsonAppInfo`

The parameter jsonAppInfo is an NSString object that defines the application information in JSON string format. The required items are application name (`appname`), application ID (`appid`), application version (`appversion`), designated market area (`dma`),Nielsen-assigned data node (`sfcode`), and the country code (`ccode`). The following is an example of a properly formed jsonAppInfo string (note that all quotes inside the string are escaped with a backslash):

```
@"{\"appname\":\"PlayerApp\",\"appid\":\"F1CFBFE7-C1BB-
4B86-87BD-46DB0D0A3604\",\"appversion\":\"2.0.0.4
\",\"sfcode\":\"us\"}"
```

Return Value

Initialized NielsenAppApi object of type `id`. Returns a nil object if the initialization failed. The application should check for a valid return value.

Discussion

This method is the initialization method for the App SDK metering framework. All parameters in the `initWithAppInfo:` method are mandatory and should not be NULL values.

The id singleton object returned by the `initWithAppInfo:` method will be nil if the Nielsen App SDK is initialized on devices with iOS versions 5.x and earlier. This is because Nielsen's App SDK utilizes functionality associated with Apple's Identifier for Advertising (IDFA), which was introduced in iOS 6. The SDK will not initialize, and instead, return a nil object if the Nielsen App SDK is unable to find the NSJSONSerialization class. Check to ensure that the `initWithAppInfo:` method returns a valid object after its use.

Declared In

`NielsenAppApi.h`

loadMetadata:

The VPA player application is to use this method to send CMS metadata to the SDK. This is done by constructing a JSON formatted string of key-value pairs and calling the `loadMetadata:` method.

```
- (void)loadMetadata:(NSString *)jsonMetadata
```

Parameters

`jsonMetadata`

The metadata string should contain at least the video type "`type`" and the asset id "`assetid`". If type is marked as "content" the whole JSON dictionary gets saved as new CMS metadata inside the SDK. DPR or OCR content can be flagged with the "dprflag" and "ocrflag" respectively. For instance, the following is an example of a jsonMetadata parameter string:

```
@"{\"tv\" : \"true\",\"category\" :
\"MyProgramCategory\",\"assetid\":\"MyContentAssetId\",
\"title\" : \" MyEpisodeTitle \", \"length\" : \"3600\",
\"type\":\"content\"}"
```

For more information, see Appendix A.

Return Value

Void

Declared In

`NielsenAppApi.h`

kNol_notifyAppApiErrorStatusUpdate:

This event adds an entry to the iOS Notification Center's dispatch table for receiving notifications whenever a NielsenAppApi application error occurs.

`kNol_notifyAppApiErrorStatusUpdate`

Return Value

1. SDK Event when SDK is ready: {

"Event Description" = "Nielsen App SDK Version, ai.3.2.1.12 is initialized by the Player...";

EventStatus = 2001;

TimeStamp = "2014-09-19 14:48:35 +0000";

}

2. SDK Event when SDK is stopped: {

"Event Description" = "Nielsen App SDK is shutting down ...";

EventStatus = 2002;

TimeStamp = "2014-09-19 14:48:40 +0000";

}

Discussion

The NielsenAppApi framework makes use of iOS's NSNotification Class for notifying the host application that error events have occurred. For more information on individual error codes, see the enumerated constant, `AppApiErrorCode`, defined in Section 6. An example of an `addObserver` implementation for setting up notifications is as follows:

```
[[NSNotificationCenter defaultCenter] addObserver:self
     selector:@selector(notifyError:)
     name:@"kNol_notifyAppApiErrorStatusUpdate"
     object:nil];
```

Declared In

`NielsenAppApi.h`

kNol_notifyAppApiEventStatusUpdate:

This method adds an entry to the iOS Notification Center's dispatch table for recieving notifications whenever NielsenAppApi application events occur.

```
- (void)kNol_notifyAppApiEventStatusUpdate
```

Parameters

None

Return Value

Void

Discussion

The NielsenAppApi framework makes use of iOS's NSNotification Class for notifying the host application that events have occurred. For more information on individual event codes, see the enumerated constant, `AppApiEventCode`, defined in Section 6. An example of an `addObserver` implementation for setting up notifications is as follows:

```
[[NSNotificationCenter defaultCenter] addObserver:self
     selector:@selector(notifyEvent:)
     name:@"kNol_notifyAppApiEventStatusUpdate"
     object:nil];
```

Declared In

`NielsenAppApi.h`

playheadPosition:

For situations where you are using CMS data in conjunction with Nielsen's DPR / OCR products, your application needs to call the `playheadPosition:` method in the SDK once every 2 seconds. The position is the current location of the player from the beginning of the asset in seconds.

Note	For proper reporting, the SDK should receive a playhead position update every 2 seconds.

```
- (void) playheadPosition:(long long)playheadPos
```

Parameters

playheadPos

An Integer value representing the time in content in seconds.

Return Value

Void

Discussion

The player application must send the value of the play head to the SDK once every 2 seconds. The VPA app can schedule a timer with 2 second period and in a timer event and send the playhead position in the following manner:

```
NielsenAppApi *nielsenMeter;
AVPlayer *player;

CMTime curTime=[player currentTime];
int pos = CMTimeGetSeconds(curTime);
[nielsenMeter playheadPosition:pos];
```

Declared In

`NielsenAppApi.h`

sendID3:

This method is a receiver for HLS timed metadata events (ID3 tags) provided through iOS's NSNotificationCenter notification system. This method filters out Nielsen-specific ID3 tags from the system and buffers the data for transfer to Nielsen's collection facility.

```
- (void)sendID3:(NSString *)data
```

Parameters

`data`

An NSString object that contains only the owner ID in the PRIV frame of an ID3 tag.

Return Value

Void

Discussion

During the HLS timed metadata event, only the PRIV frame's owner ID of the payload (or a copy of the payload) from the NSNotificationCenter notification system shall be passed into this method. This method ignores any owner ID that does not pertain to Nielsen. Since ID3 tags are continuously streamed, every timed metadata event must be captured, stored, and transferred for accuracy of metering.

Note	It is not necessary for the player application to Enable/Disable the Nielsen App SDK component depending on whether Nielsen ID3 tags are present in the stream. This is handled automatically by the SDK.

Declared In

`NielsenAppApi.h`

appDisableApi:

Enable or disable meter functions.

```
- (BOOL)appDisableApi:(BOOL)apiDisable
```

Parameters

`apiDisable`

A Boolean value set by the app developer to enable or disable Nielsen measurement.

Return Value

FALSE: To enable the meter

TRUE: To disable the meter

Discussion

To disable meter, the `appDisableApi` method should be called with the value set to TRUE.

Declared In

`NielsenAppApi.h`

play:

The SDK is started by calling the `play` method with the `channelInfo` parameter. Normally this method is called when the user presses the play button on the player.

```
- (void)play:(NSString *)channelInfo
```

Parameters

`channelInfo`

The `channelInfo` NSString objects contains the channel name (channelName) defined in JSON string format. For example:

```
@"{\"channelName\":\"TheMovieTitle\"}"
```

Return Value

Void

Discussion

This method must be called at the start of every streaming session. The maximum length of this string is 32 characters. For more information, see Setting the channelName Field. Special Considerations:

Note	The text format of the `channelInfo` string description is free-form.

Declared In

`NielsenAppApi.h`

stop

Records a channel `stop` event.

```
- (void)stop
```

Parameters

None

Return Value

Void

Discussion

Indicates the end of a streaming session. This method is to be called when the user either stops streaming content or switches to a different HLS stream, in which case, the VPA application should call the `play:` method when starting the new stream.

Declared In

`NielsenAppApi.h`

userOptOut:

Enable or disable the VPA player application for Nielsen measurement.

```
- (BOOL)userOptOut:(NSString *)optOut;
```

Parameters

`optOut`

NSString value received back from the Nielsen Opt-out web page. This value is set by the web page and the application is responsible for passing this value back to the SDK without modification.

Return Value

FALSE: The command response or URL was not intended for the Nielsen SDK. Retrieve the next URL if requested.

TRUE: This command/URL was handled by the Nielsen SDK.

Discussion

Note	The value of the `userOptOut` state is persisted across application launches or applicationWillEnterForeground mode changes.

This method is to be called when the user has selected to Opt-in or Opt-out of Nielsen measurement.

```
- (BOOL)userOptOut:(NSString *)optOut;
```

Declared In

`NielsenAppApi.h`

optOutURLString

Queries the metering framework for the Nielsen opt-out web page.

```
- (NSString*)optOutURLString;
```

Parameters

None

Return Value

Returns an NSString object with the URL of Nielsen opt-out page. In special cases, it is possible that a nil value can be returned.

Note	If OptOutURL is nil then handle it gracefully and retry it later.

Discussion

```
NSString *webAddress = [nielsenMeter optOutURLString];
```

Get the URL of the web page that is used to give the user a chance to opt out from the Nielsen measurement.

Declared In

`NielsenAppApi.h`

Class Methods

getLastEvent

Queries the NielsenAppApi framework for the last event that occurred.

```
+ (NSString *)getLastEvent
```

Parameters

None

Return Value

Returns a JSON NSString object with the following event information:

Key	Value
lastEvent	AppApi EventCode
timestamp	NSDate

`lastEvent`

An NSEvent JSON NSString object is returned that provides specific information on the event. The NSEvent contains descriptive information about the event that occurred, which is predefined by the Nielsen App SDK and the time it occurred. See Section 6 for more information on the AppApiEventCode enumeration types.

`timestamp`

An NSDate object containing a time stamp at which the last event occurred.

Discussion

The `getLastEvent` method allows the player application to query the NielsenAppApi Class for details on the last event that occurred.

Declared In

`NielsenAppApi.h`

getLastOccurredError

Queries the App API framework for the last error that occurred.

```
+ (NSDictionary *)getLastOccurredError
```

Parameters

None

Return Value

Returns an NSDictionary object with the following error information:

Key	Value
lastErrorOccurred	NSError
timestamp	NSDate

`lastErrorOccurred`

An NSError NSDictionary object is returned that provides information on the error domain, the error code, and details on the specific error. The `userInfo` dictionary object in the NSError contains descriptive information about the error that occurred, which is predefined by the Nielsen App SDK and the time it occurred. See Section 6 for more information on the AppApiErrorCode enumeration types.

`timestamp`

An NSDate object containing a time stamp at which the last event occurred.

Discussion

The `getLastOccurredError` method allows the player application to query the Nielsen AppApi Class for details on the last error that occurred.

Declared In

`NielsenAppApi.h`

getLastOccurredEvent

Queries the metering framework for the last event that occurred.

```
+ (NSDictionary *)getLastOccurredEvent
```

Parameters

None

Return Value

Returns an NSDictionary object with the following event information:

Key	Value
eventCode	NSInteger
eventDescription	NSString
timestamp	NSDate

`eventCode`

An NSInteger which specifies the NielsenAppApi event code and when it occurred. See Section 6 for more information on the `AppApiEventCode` enumeration types.

`eventDescription`

An NSString object that describes the actual event.

`timestamp`

An NSDate object containing a time stamp at which the last event occurred.

Discussion

The `getLastOccurredEvent` can be used by player application to query the NielsenAppApi Class for details on the last event that has occurred.

Declared In

`NielsenAppApi.h`

getMeterVersion

Returns version of Nielsen App SDK framework being used.

```
+ (NSString *)getMeterVersion
```

Parameters

None

Return Value

Returns an NSString object containing the ID3Meter SDK version. This allows the developer to display version information such as, "Nielsen App SDK –V3.0.0.4", in the About section of the application's settings view.

Discussion

This API call returns the current version of the Nielsen App SDK.

Declared In

`NielsenAppApi.h`

6. Constants/Enumerations

The following are NS Notification Center selector methods defined in the NielsenAppApi SDK:

Notification Names

```
#define kNotifyError @"kNol_notifyAppApiErrorStatusUpdate"
#define kNotifyEvent @"kNol_notifyAppApiEventStatusUpdate"
```

Notification listener methods must be registered using iOS's `addObserver:` NSNotification method.

AppApiErrorCode

```
typedef enum {
    AppApiNetworkConnectionFailure =1001,
    AppApiFileWriteFailure =1002,
    AppApiFileReadFailure = 1003,
    AppApiEmptyValue = 1004,
    AppApiEmptyAppName = 1005,
    AppApiEmptyAppVersion = 1006,
    AppApiEmptyAppId = 1007,
    AppApiAnExceptionOccured = 1008
} AppApiErrorCode;
```

An enumeration with predefined error codes which the App SDK object can generate.

AppApiEventCode

```
typedef enum {
     AppApiStartup = 2001,
     AppApiShutdown = 2002,
}AppApiEventCode;
```

An enumeration with predefined App SDK event state transition codes.

7. Opt-Out and Nielsen Metering Information

7.1. Providing Nielsen Metering Information to the User

In accordance with Nielsen's SDK licensing agreement, developers are required to provide basic informational data to users about Nielsen's Privacy Policy, and where additional information on Nielsen measurement can be obtained.

In the App store DESCRIPTION, include a short description about Nielsen measurement.

Sample Description:

> This app features Nielsen's proprietary measurement software which will allow you to contribute to market research, like Nielsen's TV Ratings.
>
> Please see http://www.nielsen.com/digitalprivacy for more information.

Refer to the Nielsen Privacy Requirements document for more information.

7.2. Opt-Out

The SDK provides a means for users to opt-out of Nielsen measurement. A user can use this feature if they would prefer not to participate in any of Nielsen's online measurement research.

The application must provide a user interface for the user to opt-out from Nielsen measurement. This must be a link in the application's terms and conditions page that takes the user to the Nielsen Privacy URL web page.

The opt-out happens by opening a Nielsen defined web page and passing the user choice from the web view command. The URL to this webpage should be called from the SDK and opened in a web view within the app, not within Safari or Chrome outside of the app.

Below are the steps to implement user opt-out using UIWebView:

1. When a user clicks Opt-Out, the application should invoke optOutURLString to get the link to the Nielsen Privacy page from SDK.

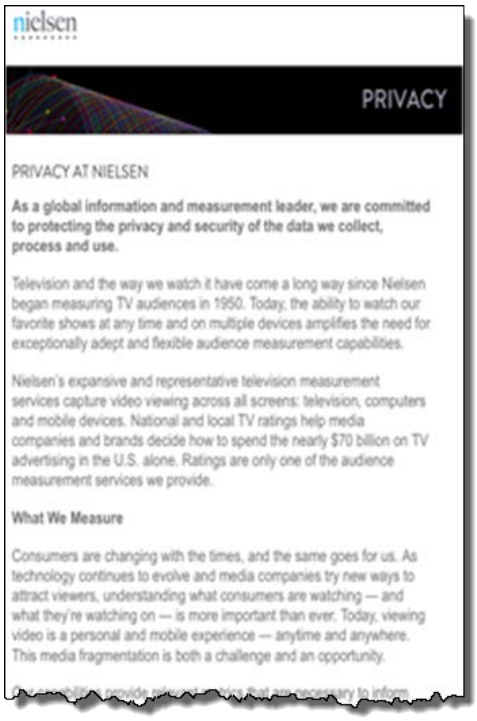
```
NielsenAppApi *NID3Meter;
```

```
@property(strong) UIWebView *optOutView;
// create web view and set self as delegate
self.optOutView=[[UIWebViewalloc]initWithFrame:self.view.bo
unds];
self.optOutView.delegate = self;
// get Nielsen defined web address and load the page
NSString *webAddress = [NID3Meter optOutURLString];
If(webAddress == nil)
{ // Handle it gracefully and retry later} else
{[optOutView loadRequest:[NSURLRequest
requestWithURL:webAddress]];
// show the view to the user
```

[self.view addSubview:optOutView]; }

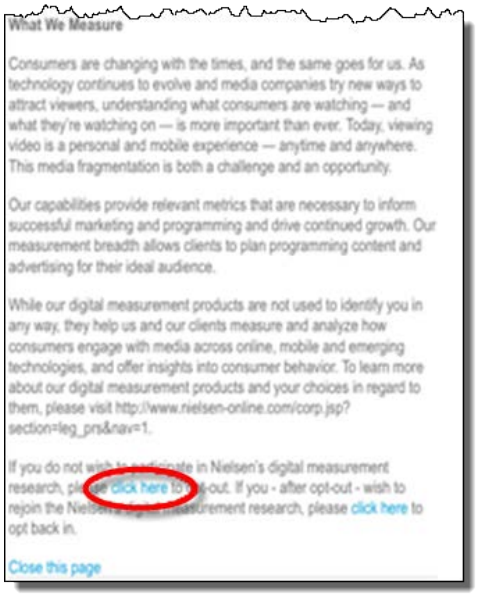
2. The Nielsen Privacy page appears using UIWebView (Figure 7).

Figure 7 – Nielsen Privacy Page



3. The user scrolls down the Nielsen Privacy page and selects the Opt-Out link (Figure 8). A confirmation message appears.

Figure 8 – Nielsen Privacy Page – Opt-Out Link



4. The user taps OK to confirm.

The application needs to implement the following UIWebView delegate method to capture the user selection, and pass the user selection back to the SDK via the userOptOut method.

```
- (BOOL)webView:(UIWebView *)webView
shouldStartLoadWithRequest:(NSURLRequest *)request
navigationType:(UIWebViewNavigationType)navigationType
```

The app gets the user selection string via web views shouldStartLoadWithRequest and invokes userOptOut with user selection. The delegate method handles the webView URL requests, interprets the commands, and calls the SDK accordingly.

Sample Implementation

```
(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 reqeusted
        return (![NID3Meter userOptOut:command]);
    }
```

Where [NlsAppApi *userOptOut*:command]); passes the user's selection on Nielsen Privacy page to the SDK to allow the SDK to perform the required functions.

The Nielsen Privacy page must be open via UIWebView. Opening the opt-out page in the Safari browser takes the application control out of the Nielsen appSDK context. The reason being that there will be no way to interact/send notification to the appSDK regarding the user selection. The App SDK does not expose any direct http/https interface to interact with outer world to get the user selection on Privacy page.

Note | The SDK manages the user's choice (opt-out/opt-in) so this status is not required or need to be managed by the app.

8. Nielsen Sample Application

The Nielsen App SDK includes a simple, fully-documented sample application to demonstrate the use of all of the components included within the SDK. The Nielsen iPad player sample application consists of a simple video player using the Apple Media Player Framework integrated with a Nielsen App SDK object. The player demonstrates all the supported functions of the API in the Nielsen SDK.

Included in the SDK package are a test URL and sample HLS streams that the developer can use to test ID3 tag extraction and confirm that data is being properly collected.

8.1. Development Environment

The software included in the sample application is written in Objective-C and can be built and run on a system with the following specifications:

- Mac OS X Lion (10.7) or later operating system
- Xcode 4.4 and above
- iOS Simulator 4.4 and above

8.2. Integrating the Nielsen App SDK Sample App

In order to integrate the Nielsen App SDK into the VPA, the following must be done:

1. Add the "NielsenAppApi.Framework" to the application. While adding the files, the option "Copy items into destination group's folder (if needed)" must be selected.

2. Import <NielsenAppApi/NielsenAppApi.h> in "MyStreamingMovieViewController.m" file.

```
#import "MyStreamingMovieViewController.h"
#import "MyPlayerLayerView.h"
#import <AVFoundation/AVFoundation.h>
#import <NielsenAppApi/NielsenAppApi.h>
```

Now, the NielsenAppApi Class Instance Methods of the Nielsen App SDK can be called from the video player application.

8.3. Key Classes and Methods used in the Sample VPA

The frameworks used for the Nielsen App SDK sample application are:

- UIKit
- Foundation
- CoreGraphics
- AVFoundation
- CoreMedia
- Security
- NielsenAppApi
- CoreText
- SystemConfiguration
- libsqlite3.dylib

For more information on the sample app, refer to developer comments in the sample application and Apple's iOS Developer portal.

8.3.1. Major Functions in Sample App

Table 2 – Major Functions in Sample App

Class Name	Method	Description
MyStreamingMovieViewController.m	play:(id)sender	Plays the video
MyStreamingMovieViewController.m	pause:(id)sender	Stop/pause the playing video
MyStreamingMovieViewController.m	startFromBegin:(id)sender	Play the video from the beginning

Class Name	Method	Description
MyStreamingMovieViewController.m	channelUp:(id)sender channelDown:(id)sender	Switch channels up and down
MyStreamingMovieViewController.m	ForwardRewind:(id)sender	Forward or rewind the playing video
MyStreamingMovieViewController.m	onUserOptOut:(id)sender	Sends the user's privacy status. It invokes a Nielsen privacy page using NSURLRequest.
MyStreamingMovieViewController.m	webView:(UIWebView *) webView shouldStartLoadWithRequest: (NSURLRequest *)request navigationType:(UIWebView NavigationType)navigationType	This is a delegate method of UI web view, which collects the response of the user's privacy status and sends it to the SDK.
MyStreamingMovieViewController.m	handleTimedMetadata:(AVMetadataItem*)timedMetadata	1. Sends timed metadata for a video to the collection facility by calling sendID3: timedMetadata method of Nielsen AppApi. 2. Displays the ID3 tags and metadata in the actual order of the ID3 tags being sent in a running list.
MyStreamingMovieViewController.m	playerItemDidReachEnd:(NSNotification*) aNotification	Called when the player item has played to its end time. It enables the play button.
MyStreamingMovieViewController.m	playerItemDuration	Gets the duration for a AVPlayerItem.

Appendix A – JSON String Formats

JSON Schemas for NSString Objects

The following details provide the JSON schema used in the initWithAppInfo and loadMetadata methods.

JSON Schema for the AppInfo String

```
{
      "title": "App Information",
      "type": "object",
      "properties": {
            "appid": {
            "title": "Unique AppID assigned by Nielsen",
            "type": "string"
            },
            "appversion": {
            "title": "Version of VPA App",
            "type": "string"
            },
            "appname": {
            "title": "Name of VPA App",
            "type": "string"
            },
            "sfcode": {
"title": "Nielsen assigned Collection Facility",
            "type": "string"
            },
"dma": {
            "title": " Nielsen Designated Market Area
code",
            "type": "string"
            },
            "ccode": {
            "title": "Country code",
            "type": "string"
            },

      },
      "required": ["appid", "appversion", "appname",
"sfcode"]
}
```

Note DMA and Country Code are optional parameters. DMA codes supplied should be standard Nielsen-recognized DMA codes. If an unrecognized `dma` value is used, it will be overwritten by Nielsen specified values.

Note sfcode identifies the Nielsen collection facility to which the SDK should connect. The default `sfcode` is "US". Please consult your Technical Account Manager before using DMA and Country Code to ensure these parameters are applicable for your application.

JSON Schema for the ProgramInfo String

```
{
      "title": "Program Information",
      "type": "object",
      "properties": {
            "assetid": {
            "title": "ID of asset from CMS",
            "type": "string"
            },

            "title": {
"title": "Descriptive title value for reporting purposes.
Use URL if title not available",
            "type": "string"
            },
            "length": {
            "title": "Episode Length",
            "type": "integer",
            "minimum": 0
            },
            "type": {
            "title": "Type of content played",
"enum" : [“content”, “preroll”, “midroll”, “postroll”,
“ad”]
            "type": "string"
            },
            "category": {
            "title": "Category of the Episode",
            "type": "string"
            },
            "censuscategory": {
"title": "Enables client defined reporting in Video
Census",
            "type": "string",
            },
```

```
            "ocrtag": {
            "title": "OCR tags"
            "type": "string"
            },
            "tv": {
"title": "Is DPR enabled for this content – true, false"
            "type": "boolean"
            },
            "prod": {
"title": "Type of product. For IAG, set to iag. If both
VideoCensus and IAG are used set the prod value to vc,iag"
            "type": "string",
            },
            "pd": {
"title": "IAG Partner Distribution - Call sign or short
abbreviation of partner"
            "type": "string"
            },
            "tfid": {
"title": "IAG Tag Format ID supplied by Nielsen IAG"
            "type": "string"
            },
            "sid": {
"title": "IAG Source ID supplied by Nielsen IAG"
            "type": "string"
            },
      },
}
```

See Section 5.3 for information on variable name re-mapping if your company uses different CMS naming conventions to refer to these fields.

For more information, consult your Technical Account Manager.

Note

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 (for example, The Simpsons). The <category> value in Video Analytics serves the same purpose. You can specify either parameter or both. For more information, consult your Technical Account Manager.

Nielsen IAG Support

Nielsen's App SDK can automatically generate parameters needed for Nielsen's IAG service. Additional IAG parameters can be accepted by the SDK. IAG parameters can also be configured in the Nielsen configuration servers. For more information on IAG, consult your Technical Account Manager.

Nielsen 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 Content Management System (CMS) identifying that the video contains ad content. This information should be passed as a JSON string via the `loadMetadata:` method with OCR contents.

OCR Tags

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

SDK Integration

Table 3 – Online Campaign Ratings Variables

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 supplied by your Technical Account Manager. This should include the complete URL including the http portion. The ocrtag should be properly URI encoded.	http://secure- uat-cert.imrworldwide.com/cgi-bin/m?ci=ENTXX5&am=3&ep=1&at=view&rt=banner&st=image&ca=XX1717&cr=crvXX35&pc=plc1234

Example

```
NSDictionary* metaDatainformation = @{
          @"type" : @"ad",
          @"ocrtag" : @"URL received from Ad server"
};
NSData* jsonOCRInfo = [NSJSONSerialization
      dataWithJSONObject:metaDatainformation
            options:0 error:nil];

NSString* ocrInfo = [[NSString alloc]
      initWithBytes:[jsonOCRInfo bytes]
            length:[jsonOCRInfo length]
            encoding:NSUTF8StringEncoding];

 [NID3Meter loadMetadata:ocrInfo];
```

The ocrtags received from your CMS system should be transformed into a JSON string and passed in via the `loadMetadata:` method.

Nielsen DPR:(Digital Program Ratings)

Nielsen's DPR product requires the following data parameters:

Table 4 – Digital Program Ratings Variables

Usage: TV App			
Variable Name	**Variable Description**	**SDK Generated / Client Defined Value**	**Example**
dataSrc**	Source of the data. For DPR datasrc should be specified as "cms".	Client-Defined	cms
type	Type of play event, preroll, midroll, postroll.	Client-Defined	Content
assetid	Unique ID of content	Client-Defined	vid-123

Usage: TV App			
Variable Name	**Variable Description**	**SDK Generated / Client Defined Value**	**Example**
tv	TV-enabled flag should be set as true if the content relates to any program that has been aired on TV. **Note** "True" for demographics data. "False" for non-demographic data Default value False.	Client-Defined	true
category	Program name	Client-Defined	NCIS
title	Episode title	Client-Defined	S2, E1
length	Length of content in sec	Client-Defined	3600
** Only applicable for SDK 1.1 and above.			

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.

Note The episode title will be the episode name used for reporting. It is recommended to use the following naming convention: Episode Name - Season Number - Episode Number - Shortform (SF)/longform (LF) content (*e.g.* Episode Name - S2 - E1 - LF*).*

Your CMS system might refer to these data items with alternate names such as:

- videoID – assetID
- program – program
- episode – title
- duration – length
- vidType – type
- tv – dprflag

In the above example, the CMS mappings would need to be captured by Nielsen's Technical Account Manager during the on-boarding process. The mapping ensures that the player can create a proper JSON string using your current schema KEYS.

See Section 5.3 for information on variable name re-mapping if your company uses different CMS naming conventions to refer to these fields.

Sample JSON input for DPR:

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

Sample Code

```
//Create the  necessary JSON input
NSString *dprMetadata=@"{\"type\" : \"content\",\"assetid\"
: \"MyContentAssetId\",\"tv\" : \"true\",\"program\" :
\"MyProgram\",\"title\" : \"MyEpisodeTitle\",
\"category\":\"ReportingCategory\", \"length\" :
\"3600\"}";

// Pass the JSON structure to API
 [NID3Meter loadMetadata:dprMetadata];
```

DPR viewing is driven from the playhead position. The current playhead position (position in the video asset) must be passed to the SDK via the `playheadPosition:` method every 2 seconds.

Example:

playheadTimer=[NSTimer scheduledTimerWithTimeInterval:nPlayHeadInterval target:self selector:@selector(playHeadTimeEvent) userInfo:nil repeats:YES];

where nPlayHeadInterval = 2

```
- (void)playHeadTimeEvent
{
// player : Instance of AVPlayer used to play the video
stream
    CMTime curTime=[player currentTime];
    int pos = CMTimeGetSeconds(curTime);
// NID3Meter : Instance of AppApiSDK    [NID3Meter
playheadPosition:pos];
}
```

Example JSON Schemas for VC / IAG loadMetadata Methods

Example for a VC / IAG for Content Tags

```
{
"length":"600".
"title":"TestContentTitle",
"censuscategory":"TestContentCensusCategory",
"type":"content",
"assetid":"245671"
}
```

Example for a VC / IAG Advertisement (Type Preroll) Tags

```
{
“length”:”16”,
“title”:”FOX_StateFarm_SFSSP_Video_VAST_DFA1”,
“iag_epi”:”Season 3 - Guess Who's Back?”,
“iag_seg”:”1”,
“iag_pgm”:”New Girl”,
“category”:”STATE FARM”,
“subcategory”:”2013 State Farm Online Video”,
“type”:”preroll”
}
```

The following is an example of how you might load CMS data into an NSDictionary object and JSON string-ify it for sending to Nielsen's iOS App SDK using the `loadMetadata:` method:

```
NSDictionary* vcInfoDict = @{
@"length": @"16",
@"title": @"FOX_StateFarm_SFSSP_Video_VAST_DFA",
@"iag_epi": @"Season 3 - Guess Who's Back?",
@"iag_seg": @"1",
@"iag_pgm": @"NewGirl",
@”type” : “ preroll”,
@"catagory": @"STATE FARM",
};
```

NSData* jsonDataVcInfo = [NSJSONSerialization dataWithJSONObject:vcInfoDict options:0 error:nil];

NSString *sVCString = [[NSString alloc] initWithBytes:[jsonDataVcInfo bytes] length:[jsonDataVcInfo length] encoding:NSUTF8StringEncoding];

[NID3Meter loadMetadata:sVCString];

where

iag_epi – Episode Name

iag_seg – Segment Number

iag_pgm – Program

Nielsen mTVR (Mobile TV Ratings)

Setting the channelName Field

The *channelName* field is a freeform text field in the protocol between the device and Nielsen's Collection Facility. This field is used to convey a text description of the channel or asset being selected by the user for viewing on his or her device. In an effort to enhance this field's usefulness across different VPA player apps, this field has been sub-divided to support the creation of different channelInfo types with the goal that MVPDs/CNDs/COs (Content Originators) would select one of these types for use in their particular application. For example, the use of CRIDs for on-demand assets allows Nielsen's collection facility to isolate on-demand content through the use of an asset's CRID Authority.

Table 5 – Mobile TV Ratings channelName Field

Name	Description	Sample Value
channelName	Name of the channel. You can pass any friendly name for the content being played.	ESPN2
dataSrc **	Data Source Identifier. For mTVR data source should be specified as "id3".	Id3
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.		

JSON for Channel Name at Play

```
{
”channelName”: ”ESPN2”,
”dataSrc”: ”id3”,
”adModel”: ”1”
}
```

Nielsen DRM (Digital Radio Measurement)

SDK can accept latitude and longitude values in the case of DRM.

Sample JSON input for DRM:

```
{
"sfcode":"drm",
"appid":" XXXXE427-1970-47E1-BF7D-75752353XXXX ",
"appname":"Nielsen SDK Sample QA",
"appversion":"app.3.1.1",
"longitude":"123.223",
"latitude":"-223.323"
}
```

Sample

```
// Create the necessary JSON input
NSDictionary* appInformation = @{
            @"appName" : appName,
            @"appVersion" : appVersion,
            @"sfcode"     : sfCode,
            @"appId"     : appId,
            @"latitude"  : appLatitude, // optional
            @"longitude"  :appLongitude  // optional
};
```

DRM viewing is driven from the playhead position. The current playhead position (position in the video asset) must be passed to the SDK via the playheadPosition method every 2 seconds.

Load Content Metadata at Play

Name	Description	Sample Value
dataSrc **	Source of the data. For DRM datasrc should be passed as "cms".	cms
type	Type of content. For DRM mention type as "radio"	radio
assetid	Station identifier. Should include Call letters and Band.	WXYZ-FM
stationType	Station type indicating the type of usage and the ad load. 0: Custom station built per client 1: Streaming station corresponds to an OTA station and has same ad load as OTA 2: Streaming station corresponds to an OTA but with different ad load 3: Multicast e-station or online station	0, 1, 2, 3
provider	Name of Provider.	providerName
** Only applicable for SDK 1.1 and above.		

The name "provider" can be changed as long as the new names are provided to Nielsen.

Typical JSON of Content Metadata

JSON for radio content:

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

Glossary

C

CO

Content Originator. Content Originators are Nielsen clients that aggregate content and distribute it to end users, either as a linear feed (examples include traditional cable networks such as A&E, ESPN, TNT, etc.), or as on-demand content (examples include iPad applications such as A&E, WatchESPN, TNT for iPad, etc.).

CMS

Content Management System

D

DPR

Digital Program Ratings

H

HLS

HTTP Live Streaming. An HTTP-based media streaming communications protocol implemented by Apple and proposed as an Internet Draft Standard.

HTTPS

Hypertext Transfer Protocol – Secure. Standard secure connection protocol used on the web.

I

IAG

Nielsen acquired IAG Research in 2008. IAG measures consumer engagement with television programs, national commercials, and product placements.

ID3

A metadata container convention that has been adapted for use with HLS-streamed content.

M

MVPD

Multichannel Video Programming Distributor. MVPDs are server providers that deliver video programming feeds to end users. MVPDs may be cable television systems (CATV), direct-broadcast satellite provides (DBS), or traditional telcos. Examples include Comcast, Time Warner, DirecTV, DISH, FiOS, A&T U-Verse, etc.

N

Nielsen Watermark

A proprietary algorithm developed by Nielsen to insert sub-audible watermarks in audio portion of media files and/or streams. Nielsen's watermarks enable audience measurement.

O

OCR

Online Campaign Ratings

S

SID

Source Identifier. Source IDs and Program IDs are associated with the tags that are injected into the ID3 tags by Nielsen for retrieval by the component

V

VC

Video Census

Viewing Session

The period of time from the moment the VPA player starts streaming content, to the point where the user either shuts down the application, sends it to the background, or stops streaming content.