Triton Digital Mobile SDK 2.2.2 for iOS

2015-10-02

TRITON™
DIGITAL

Publicaton Information

© 2015 Triton Digital

Published by Triton Digital Inc.

1440 Ste-Catherine W, Suite 1200
Montreal QC H3G 1R8
Canada

514-448-4037
www.tritondigital.com

Document Version

Triton Digital Mobile SDK 2.2.2 for iOS

2015-10-02

Trademarks

Triton Digital Product Names are trademarks of Triton Digital Inc. All other trademarks belong to their respective owners.

Disclaimer Notice

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, translated into any other language in any form or by any means, electronic or mechanical, including photocopying or recording, for any purpose, without the express permission of Triton Digital.

Triton Digital Inc. has made every effort to ensure the accuracy of the information contained herein. However, due to continuing product development, the information is subject to change without notice.

Customer Support
supportdesk@tritondigital.com
1-866-448-4037 extension 1

Contents

Introduction

The Triton Mobile SDK is designed to help you play your radio station or display on-demand ads very easily in your own application. There are two versions; one for Android and one for iOS. This is the iOS version.

The SDK includes a ZIP file that contains the API reference, and a sample application that is ready to compile showing the most common uses of the SDK.

Document Change Log

Version	Date	Change
2.2.2 (iOS)	2015-10-02	• Added Token Authorization (Self Signed) to the Using the iOS SDK section.

ZIP File

Download the ZIP file here: http://triton-sdk.media.streamtheworld.com/mobile_sdk/

(*Be sure to take the iOS one.*)

The ZIP file contains:

- API reference (HTML format)
- A sample application ready to be compiled that shows the most common uses of this SDK.
- The SDK!

© 2015 Triton Digital

Main Features

The main features of the SDK:

- Play Triton streams.
- Receive meta-data synchronized with the stream.
- Advertising:
  - Sync banners
  - On-demand audio and video interstitial ads
  - Support for VAST and DAAST formats
- Receive Now Playing History information.

Client-specific Information

This SDK should come with the following client-specific information:

Info	Description	Use
Broadcaster	Your company name.	• Playing a station
Mount	Identifies your station stream (e.g., MOBILEFM_AACV2).	• Playing a station
Station name	Identifies your station (e.g., MOBILEFM).	• Playing a station • On-Demand ads
Station ID	Identifies your station (e.g., 12345). For on-demand ads, this can be used instead of the station name.	• On-Demand ads
On-Demand Ad Server	Server to use for on-demand advertising.	• On-Demand ads

Glossary

Term	Definition
Ad	Contains a main creative and multiple companion banners.
Ad Request	URL used to request an ad from Triton Digital's server.
Banner	Part of an ad that can be displayed in a *BannerView*.
BannerView	UI component used to display banners.
Interstitial	Full-screen display of a video or audio ad over an application.
Main Creative	Main part of an ad; an audio or a video file.

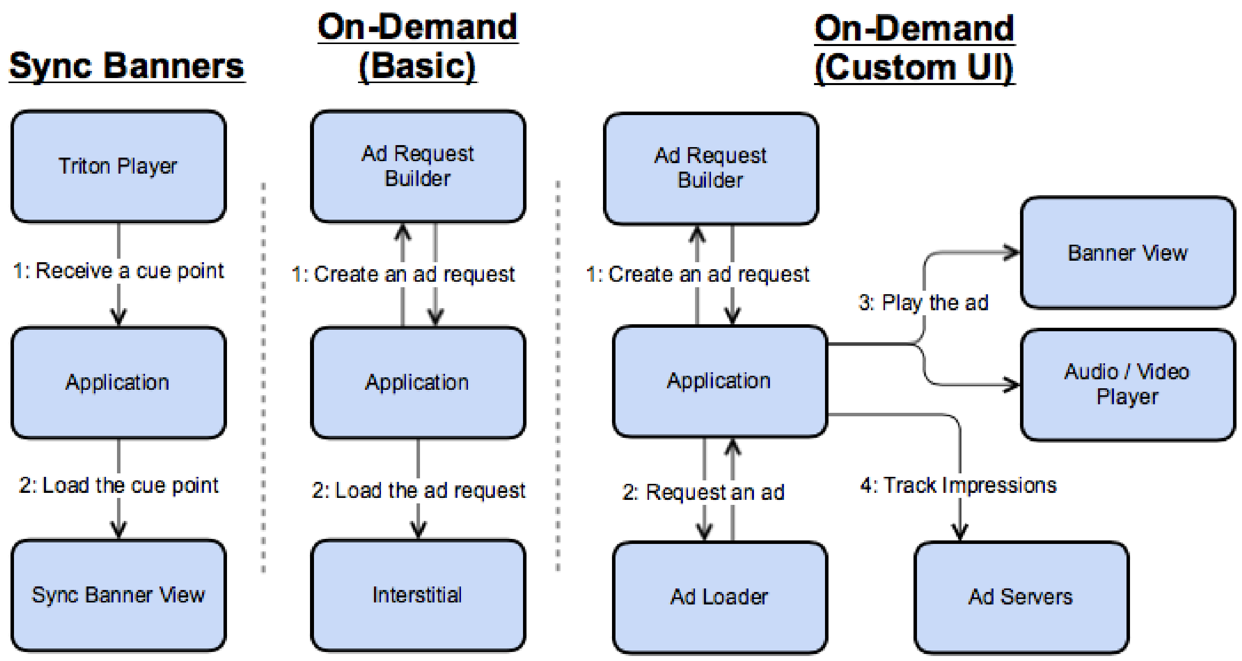
Technical Support

Triton Digital will provide technical support only for problems that can be reproduced in the sample application of the lastest SDK version.

supportdesk@tritondigital.com / 1-866-448-4037 extension 1

© 2015 Triton Digital

Ad Request Process



iOS SDK

- The library is distributed as a static framework for Xcode called **TritonMobileSdk**.
- **Minimum requirements:**
  - Xcode 6
  - iOS 7.0
- **API reference:**
  - http://triton-sdk.media.streamtheworld.com/mobile_sdk/ios_api_reference/

Dependencies

In order to link TritonMobileSdk, the frameworks **SystemConfiguration**, **AdSupport**, **AVFoundation**, **MediaPlayer** and **CoreMedia** must also be linked in Xcode.

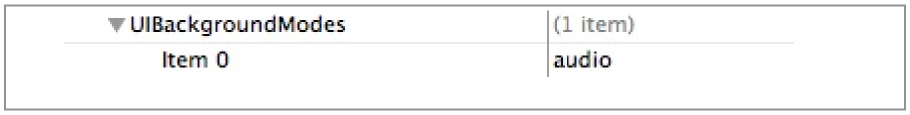
Submitting an Application to the AppStore Using TritonMobileSdk

TritonMobileSdk uses Advertising Identifier (IDFA) for targeting its advertisements. It respects the end user's "Limit Ad Tracking" setting in iOS when collecting tracking data. Apple requires developers to confirm their use of Advertising Identifier when submitting an application through iTunes Connect. Therefore, when submitting an application using TritonMobileSdk, we recommend you state that the application uses IDFA, and also to check the option "Serve advertisements within the app," along with the clause that confirms that the developer honors the end user's "Limit Ad Tracking" setting.

The developer should also check other cases according to their specific use of advertising in the application. Failing to do so may cause app rejection by Apple.

Support for Background Application Streaming

If the application is intended to continue streaming when in the background, you must declare that the app plays audible content while in the background in the application's **info.plist** file through the key **UIBackgroundModes**, as shown below:



▼UIBackgroundModes	(1 item)
Item 0	audio

The SDK configures the audio session with the proper category to enable playback in the background.

Using the iOS SDK

The following section describes how to accomplish the most common tasks using TritonMobileSdk. The SDK package includes a **sample Xcode project** that showcases the available features and shows how to integrate the SDK into a streaming application.

Playing a Station

Stream functionality is provided by the **TritonPlayer** class. In order to be able to play, you must provide the settings for your station; at a minimum, provide the station and broadcaster's name and the mount. Contact Triton Digital if you don't have this information. See the *API Reference* for other available settings.

```
NSDictionary *settings = @{SettingsStationNameKey : @"MOBILEFM",
                           SettingsBroadcasterKey : @"Triton Digital",
                           SettingsMountKey : @"MOBILEFM_AACV2"
                           };

self.tritonPlayer = [[TritonPlayer alloc] initWithDelegate:self andSettings:settings];

...

[self.tritonPlayer play];
```

Changing Station

When changing stations, you need to provide the settings for the new station. Use the method **-updateSettings:** to configure this. It will override the previous settings but will only be effective next time the player stops and plays again.

```
// Stop previous stream
[self.tritonPlayer stop];

...

// Configure and play the new station
NSDictionary *settings = @{SettingsStationNameKey : @"WMSC_S01",
                           SettingsBroadcasterKey : @"Triton Digital",
                           SettingsMountKey : @"WMSC_S01AAC"
                           };

[self.tritonPlayer updateSettings:settings];
[self.tritonPlayer play];
```

Alternatively, you can recreate the TritonPlayer every time you change stations, but the procedure above is preferred.

Target Audience for Audio Ads

In order to target your audience for audio ads (and respective companion banners), you must pass targeting parameters to the TritonPlayer's settings. Suppose you want to target males, 30 years old, and provide their location coordinates:

```
NSDictionary *settings = @{SettingsStationNameKey : @"MOBILEFM",
                           SettingsBroadcasterKey : @"Triton Digital",
                           SettingsMountKey : @"MOBILEFM_AACV2",
                           SettingsEnableLocationTrackingKey : @(YES),
                           SettingsStreamParamsExtraKey : @{@"gender": @"m",
                                                            @"age": @"30"}
                          };
self.tritonPlayer = [[TritonPlayer alloc] initWithDelegate:self andSettings:settings];

...

[self.tritonPlayer play];
```

When **SettingsEnableLocationTrackingKey** is YES, it will use **CLLocationManager** to provide the user's location (See "Location Tracking"). The other parameters (including manual location) are specified in a dictionary in the **SettingsStreamParamsExtraKey**. For all available targeting parameters see the *Triton Digital Streaming Guide*.

Token Authorization (self signed)

```
// Create the targeting parameters
NSDictionary *targetingParams =  @{@"gender": @"m",
                                   @"age": @"30"
                                 };

// Create the authorization token
NSString *token = [TDAuthUtils createJWTTokenWithSecretKey:@"MySecretKey"
                                                andSecretKeyId:@"MySecretKeyId"
andRegisteredUser:YES andUserId:@"foo@bar.com" andTargetingParameters:targetingParams];


// Create the player settings.
NSDictionary *settings = @{SettingsStationNameKey : @"MOBILEFM",
                           SettingsBroadcasterKey : @"Triton Digital",
                           SettingsMountKey : @"MOBILEFM_AACV2",
                           SettingsEnableLocationTrackingKey : @(YES),
                           StreamParamExtraAuthorizationTokenKey: token,
                           SettingsStreamParamsExtraKey : targetingParams
                          };

// Create the player.
self.tritonPlayer = [[TritonPlayer alloc] initWithDelegate:self andSettings:settings];
[self.tritonPlayer play];
```

Receive Stream Metadata (Cue Points)

Once the player is streaming, it is capable of receiving cue points with stream metadata (e.g., current playing song, ad metadata, etc.). To receive cue points, you need to implement TritonPlayerDelegate's **-player:didReceiveCuePointEvent:** in the class that will be the TritonPlayer's delegate. **CuePointEvent.h** defines all the constants for cue point types and keys for retrieving its information.

```
@interface ViewController() <TritonPlayerDelegate>
@end

@implementation ViewController
...
- (void)viewDidLoad {
    [super viewDidLoad];
    ...
    self.tritonPlayer = [[TritonPlayer alloc] initWithDelegate:self
andSettings:settings];
}

...
[self.tritonPlayer play];
...

- (void)player:(TritonPlayer *)player didReceiveCuePointEvent:(CuePointEvent *)
cuePointEvent {

    // Check if it's an ad or track cue point
    if ([cuePointEvent.type isEqualToString:EventTypeAd]) {
        // Handle ad information (ex. pass to TDBannerView to render companion banner)

    } else if ([cuePointEvent.type isEqualToString:EventTypeTrack]) {
        NSString *currentSongTitle = [inNowPlayingEvent.data
objectForKey:CommonCueTitleKey];
        NSString *currentArtistName = [inNowPlayingEvent.data
objectForKey:TrackArtistNameKey];
        NSString *currentAlbumName = [inNowPlayingEvent.data
objectForKey:TrackAlbumNameKey];

        // Do something with track data (update interface, notify user etc.)
    }
}
...
@end
```

Display an In-stream Companion Banner (Sync Banner)

```
@interface ViewController() <TritonPlayerDelegate>
@property (strong, nonatomic) TDSyncBannerView *adBannerView;
@end

@implementation ViewController
...
- (void)viewDidLoad {
    [super viewDidLoad];
    ...
    self.tritonPlayer = [[TritonPlayer alloc] initWithDelegate:self
andSettings:settings];

    // Create and configure a 320x50 sync banner with a fallback size of 300x50
    self.adBannerView = [[TDSyncBannerView alloc] initWithWidth:320 andHeight:50
andFallbackWidth:300 andFallbackHeight:50];
    [self.view addSubview:self.adBannerView];
}
...
- (void)player:(TritonPlayer *)player didReceiveCuePointEvent:(CuePointEvent *)
cuePointEvent {

    if ([cuePointEvent.type isEqualToString:EventTypeAd]) {
        [self.adBannerView loadCuePoint:cuePointEvent];
    }
    ...
}
...
```

Display an Interstitial Ad

Interstitial ads (pre-rolls, mid-rolls) are full-screen ads that are displayed modally in an app. Usually they are displayed at natural app transition points such as before or after playing or changing a radio station.

```
// Create a TDAdRequestBuilder to build the request string
TDAdRequestURLBuilder *request = [TDAdRequestURLBuilder builderWithHostURL:kRequestUrl];
request.stationId = 12345;

// Create a TDInterstitialAd
self.interstitial = [[TDInterstitialAd alloc] init];

// Create a TDAdLoader to load the ad from the request
self.adLoader = [[TDAdLoader alloc] init];
[self.adLoader loadAdWithBuilder:requestBuilder completionHandler:^(TDAd *loadedAd,
NSError *error) {
        if (error) {
            // Handle error
        } else {
            // Load the ad into the interstitial
            [self.videoInterstitial loadAd:loadedAd];
        }
    }
];
```

Once the interstitial is loaded, it must be showed at the appropriate time. The application must check if the interstitial is loaded before attempting to show it. If **-loaded** doesn't return YES, the ad won't be displayed. When showing an interstitial, a view controller must be passed. It will be the presenting view controller of the interstitial.

```
- (void)playStation {
  if ([self.interstitial loaded]) {
    [self.interstitial presentFromViewController:self];
  }
  // Rest of method logic
}
```

Receiving Now Playing History Information

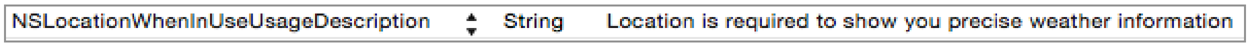
TDCuePointHistory is used for accessing Now Playing History information from Triton Servers. It returns a list of CuePointEvents for a specific mount. It also allows the specification of filters and limiting the maximum number of elements returned.

```
@property (strong, nonatomic) TDCuePointHistory *cuePointHistory;
...
 NSString *mount = @"MOBILEFM_AACV2";
 NSInteger maxItems = 20;
 NSArray *filter = @[EventTypeTrack, EventTypeAd];

 // Request the history list from the server. The result will be a NSArray of
CuePointEvent objects.
 [self.cuePointHistory requestHistoryForMount:mount withMaximumItems:maxItems
eventTypeFilter:filter completionHandler:^(NSArray *historyItems, NSError *error) {
     if (error) {
         // Handle the error
     }

     // Process the cue point list. Ex. display them in a UITableView
     self.cuePointList = historyItems;
     [self.tableView reloadData];
     ...
 }];
```

Location Tracking

TritonMobileSdk provides automatic location tracking to better target its ads. **This feature is disabled by default**. It can be enabled by setting **SettingsEnableLocationTrackingKey** to YES in the settings NSDictionary (see the "Player Settings" section).

In order for location tracking to work properly, starting from iOS 8 the application must declare the location usage description in the application **Info.plist** file. An example is shown in the image below:



NSLocationWhenInUseUsageDescription String Location is required to show you precise weather information

Triton Player SDK requests the **WhenInUse** authorization from the user, meaning the application receives device location updates only when the application is being used. The message shown to the user when requesting authorization is defined in the **NSLocationWhenInUseUsageDescription** plist key described above.

By enabling location tracking, iOS will prompt users to activate location service in their device the first time the application runs. The user can later disable location tracking in the device's System Settings; the framework will properly handle all the changes in permissions.

© 2015 Triton Digital