# Spil iOS Framework

Generated by Doxygen 1.8.5

Tue Nov 26 2013 15:50:20

# Contents

# Chapter 1

# Hierarchical Index

## 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

# Chapter 2

# Class Index

## 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 3

# Class Documentation

## 3.1 <AppSettingsDelegate> Protocol Reference

`#import <AppSettingsDelegate.h>`

### Instance Methods

- (void) - appSettingsDidLoad:
- (void) - appSettingsDidFailWithError:
- (void) - appSettingsDidStartDownload

### 3.1.1 Detailed Description

Protocol to handle the responses from the App Settings subsystem

### 3.1.2 Method Documentation

#### 3.1.2.1 - (void) appSettingsDidFailWithError: (NSError *) *error* [required]

Method to call back in case the settings couldn't be loaded. Usually the reasons to call this method will be:

- if there is any parsing error in the remote settings and in the local settings.
- if there is a connection error, and the file of the defaults cannot be found locally.

**Parameters**

| | |
|---|---|
| *error* | Error describing what was wrong. |

#### 3.1.2.2 - (void) appSettingsDidLoad: (NSDictionary *) *settings* [required]

Method to call back when the settings are finally loaded. This methods will receive the settings loaded in the form of a dictionary. The developers should know the structure of the dictionary since they created the default settings file.

**Parameters**

| | |
|---|---|
| *settings* | The settings loaded. The format and the values are defined by the developer of the app. |

#### 3.1.2.3 - (void) appSettingsDidStartDownload [required]

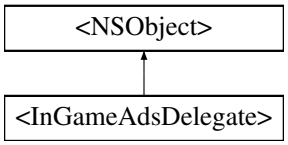
Method to call back when the download of the settings has been started. Can be use to notify the user or do other tasks until everything is downloaded.

The documentation for this protocol was generated from the following file:

- /Users/Shared/Jenkins/Home/workspace/native-ios-framework/src/Classes/public/delegates/AppSettingsDelegate.h

## 3.2 <InGameAdsDelegate> Protocol Reference

Inheritance diagram for <InGameAdsDelegate>:




### Instance Methods

- (void) - adDidGetInGameAd:
- (void) - adDidFailToGetInGameAd:

### 3.2.1 Method Documentation

#### 3.2.1.1 - (void) adDidFailToGetInGameAd: (NSError *) *error*

Method to callback when an error happened trying to retrieve the ad from the server.

**Parameters**

| | |
|---|---|
| *error* | An error code describing the cause of the error. |

#### 3.2.1.2 - (void) adDidGetInGameAd: (UIView *) *image*

Method to callback when a in game ad has been retrieved from the server. This view will handle the display event and will mark the advert as shown.

**Parameters**

| | |
|---|---|
| *image* | A UIView that will respond to the events when it's displayed and clicked. |

The documentation for this protocol was generated from the following file:

- /Users/Shared/Jenkins/Home/workspace/native-ios-framework/src/Classes/public/delegates/InGameAdsDelegate.h

## 3.3 Spil(ABTest) Category Reference

### Instance Methods

- (void) - setABTestDelegate:
- (void) - abtestUpdateUserInfo
- (void) - abtestUpdateUserInfoWith:
- (void) - abtestGetTestDiff
- (void) - abtestGetTestDiffForUser:
- (void) - abtestMarkSucceedTest:withParameters:

### 3.3.1 Method Documentation

#### 3.3.1.1 - (void) abtestGetTestDiff

Sends a request to retrieve the test differences for this user (MAC Address). The differences will be sent asynchronously to the ABTestDelegate implemented and set in the getABTest method.

#### 3.3.1.2 - (void) abtestGetTestDiffForUser: (NSString ∗) *user*

For development purposes only. Send a request to retrieve the test differences for this user. The differences will be sent asynchronously to the ABTestDelegate implemented and set in the getABTest method.

**Parameters**

| | |
|---|---|
| *user* | The user to force the different variants of the A/B test. |

#### 3.3.1.3 - (void) abtestMarkSucceedTest: (NSString ∗) *name* withParameters:(NSDictionary ∗) *params*

Marks a particular resource as a success with the parameters that lead to that success. This method should be called with the exact name of the resource and also it must be called with the control version to be able to compare results.

**Parameters**

| | |
|---|---|
| *name* | The name of the resource to mark as a successful one. |
| *params* | A dictionary with extra parameters relevant for the analysis of the action called. |

#### 3.3.1.4 - (void) abtestUpdateUserInfo

Updates the user basic information to create segments and improve A/B tests. This method will send:

- Country

- Language
- Device (ipad, ipod, iphone)
- OS Platform (version)

#### 3.3.1.5 - (void) abtestUpdateUserInfoWith: (NSDictionary ∗) *extraInfo*

Updates the user information to create segments and improve A/B tests. This method will send the basic information plus all the information included in the extra info

**Parameters**

| | |
|---|---|
| *extraInfo* | A dictionary with all the extra parameter we want to submit. |

**See Also**

- abtestUpdateUserInfo

#### 3.3.1.6 - (void) setABTestDelegate: (id< ABTestDelegate >) *delegate*

Sets the ABTestDelegate and receive the proper notifications from it. Without the delegate this subsystem is disabled.

**Parameters**

| | |
|---|---|
| *delegate* | The delegate to handle the events generated by the A/B test subsystem. |

The documentation for this category was generated from the following file:

- /Users/Shared/Jenkins/Home/workspace/native-ios-framework/src/Classes/public/Spil+ABTest.h

## 3.4 Spil(Ads) Category Reference

### Instance Methods

- (void) - setAdsDelegate:
- (void) - adsEnabled:
- (void) - adsShowMoreGames
- (void) - adsNextInterstitial
- (void) - adsNextInterstitial:
- (void) - adsCacheNextInterstitial
- (void) - adsCacheNextInterstitial:
- (void) - setInGameAdsDelegate:
- (BOOL) - adsRequestInGameAd:
- (BOOL) - adsRequestInGameAd:atLocation:
- (void) - adsRequestInGameAdAsset:
- (void) - adsRequestInGameAdAsset:atLocation:
- (void) - adsMarkInGameAdAsShown:

### 3.4.1 Method Documentation

#### 3.4.1.1 - (void) adsCacheNextInterstitial

Caches the next interstitial image to speed up the load time. Uses the default location.

**See Also**

- adsCacheNextInterstitial:

#### 3.4.1.2 - (void) adsCacheNextInterstitial: (NSString ∗) *location*

Caches the next interstitial image to speed up the load time. Use the location parameter to indicate where the ad is being displayed, for instance, "mainmenu", "store", "pausemenu". This helps to improve the advertisement campaigns.

**Parameters**

| | |
|---|---|
| *location* | Location to be used for this interstitial |

#### 3.4.1.3 - (void) adsEnabled: (BOOL) *state*

Turns on/off if the ads should be displayed. The ads are displayed by default.

**Parameters**

| | |
|---|---|
| *state* | Indicates if the ads should be displayed or not. |

#### 3.4.1.4 - (void) adsMarkInGameAdAsShown: (NSString ∗) *adId*

Marks the ad as shown, this will be use only for the unity plugin. Use this method if and only if you used adsRequestInGameAdAsset to get the asset and render the ad by yourself. The usage of this method if discourage and should only be used if you know the whole Chartboost's workflow properly.

**Parameters**

| | |
|---|---|
| *adId* | The advert id returned by chartboost assets lib. |

#### 3.4.1.5 - (void) adsNextInterstitial

Shows an ad right away, using the default location.

**See Also**

- adsNextInterstitial:

#### 3.4.1.6 - (void) adsNextInterstitial: (NSString ∗) *location*

Shows an ad right away, using the specified location. Use the location parameter to indicate where the ad is being displayed, for instance, "mainmenu", "store", "pausemenu". This helps to improve the advertisement campaigns.

**Parameters**

| | |
|---|---|
| *location* | Location to be used for this interstitial |

#### 3.4.1.7 - (BOOL) adsRequestInGameAd: (CGSize) *size*

Makes a request to get an advert (on the default location) and return it to the invoker when it's done through the InGameAdsDelegate implementation set up prior the call to this method. This methods returns right away and gives NO if there is not a valid chartboost instance, YES otherwise. Although this method returns YES, it doesn't mean the ad will be in fact retrieved and returned.

**See Also**

- adsRequestInGameAd:atLocation:

**Parameters**

| | |
|---|---|
| *size* | The width and height desired for this ad. |

**Returns**

NO if there chartboost provider is not valid. YES otherwise.

#### 3.4.1.8 - (BOOL) adsRequestInGameAd: (CGSize) *size* atLocation:(NSString ∗) *location*

Makes a request to get an advert and return it to the invoker when it's done through the InGameAdsDelegate implementation set up prior the call to this method. Use the location parameter to indicate where the ad is being displayed, for instance, "mainmenu", "store", "pausemenu". This helps to improve the advertisement campaigns. This methods returns right away and gives NO if there is not a valid chartboost instance, YES otherwise. Although this method returns YES, it doesn't mean the ad will be in fact retrieved and returned.

**Parameters**

| | |
|---|---|
| *size* | The width and height desired for this ad. |
| *location* | Location to be used for this interstitial |

**Returns**

NO if there chartboost provider is not valid. YES otherwise.

#### 3.4.1.9 - (void) adsRequestInGameAdAsset: (NSDictionary ∗) (NSError ∗) *callback*

Retrieves the JSON description of the assets, and pass it back to the invoker block a decoded JSON format. The invoker is the responsable for scale and download the asset if it's present. The usage of this method if discourage and should only be used if you know the whole Chartboost's workflow properly, use adsRequestInGameAd: instead.

**See Also**

- adsRequestInGameAd:

**Parameters**

| | |
|---|---|
| *callback* | A callback that will receive the information as a dictionary that can be serialized. |

#### 3.4.1.10 - (void) adsRequestInGameAdAsset: (NSDictionary *) (NSError *) *callback* atLocation:(NSString *) *location*

Retrieves the JSON description of the assets, and pass it back to the invoker block a decoded JSON format. The invoker is the responsable for scale and download the asset if it's present. Use the location parameter to indicate where the ad is being displayed, for instance, "mainmenu", "store", "pausemenu". This helps to improve the advertisement campaigns. The usage of this method if discourage and should only be used if you know the whole Chartboost's workflow properly, use adsRequestInGameAd:atLocation: instead.

**Parameters**

| | |
|---|---|
| *callback* | A callback that will receive the information as a dictionary that can be serialized. |
| *location* | Location to be used for this interstitial |

#### 3.4.1.11 - (void) adsShowMoreGames

Shows the More Games screen right away.

#### 3.4.1.12 - (void) setAdsDelegate: (id< AdsDelegate >) *delegate*

Sets the AdsDelegate and receive the proper notifications from it. Without the delegate this subsystem is disabled.

**Parameters**

| | |
|---|---|
| *delegate* | The delegate to handle the events generated by the Ads subsystem. |

#### 3.4.1.13 - (void) setInGameAdsDelegate: (id< InGameAdsDelegate >) *delegate*

Sets the delegate to handle the events received by the in-game ads system. Without this delegate the InGameAds subsytem is disabled.

**Parameters**

| | |
|---|---|
| *delegate* | The ads delegate who is going to handle the events. |

The documentation for this category was generated from the following file:

- /Users/Shared/Jenkins/Home/workspace/native-ios-framework/src/Classes/public/Spil+Ads.h

## 3.5 Spil(AppSettings) Category Reference

### Instance Methods

- (void) - setAppSettingsDelegate:

### 3.5.1 Method Documentation

#### 3.5.1.1 - (void) setAppSettingsDelegate: (id< AppSettingsDelegate >) *delegate*

Sets the AppSettingsDelegate and receive the proper notifications from it. A delegate is required in order to deliver the settings downloaded from the server or loaded from the default files. Without the delegate this subsystem is disabled.

**Parameters**

| | |
|---|---|
| *delegate* | The delegate to handle the response of the AppSettings subsystem. |

The documentation for this category was generated from the following file:

- /Users/Shared/Jenkins/Home/workspace/native-ios-framework/src/Classes/public/Spil+AppSettings.h

## 3.6 Spil(Tracking) Category Reference

### Instance Methods

- (void) - trackPage:
- (void) - trackEvent:
- (void) - trackEvent:action:label:value:
- (void) - trackEvent:withParams:
- (void) - trackTimedEvent:
- (void) - trackEndTimedEvent:withParams:
- (void) - trackEndTimedEvent:
- (void) - trackError:message:exception:

### 3.6.1 Method Documentation

#### 3.6.1.1 - (void) trackEndTimedEvent: (NSString ∗) *event*

Tracks the end of an event that was started. If the tracking system support it natively the equivalent method will be used. Otherwise, an event with the end timestamp(epoch) is issued.

**See Also**

- trackTimedEvent:

**Parameters**

| | |
|---|---|
| *event* | The event to track. Should match with the starting event. |

#### 3.6.1.2 - (void) trackEndTimedEvent: (NSString ∗) *event* withParams:(NSDictionary ∗) *params*

Tracks the end of an event that was started. If the tracking system support it natively the equivalent method will be used. Otherwise, an event with the end timestamp(epoch) is issued.

**See Also**

- trackTimedEvent:

**Parameters**

| | |
|---|---|
| *event* | The event to track. Should match with the starting event. |
| *params* | The parameters when the event was finished. |

#### 3.6.1.3 - (void) trackError: (NSString ∗) *event* message:(NSString ∗) *msg* exception:(NSException ∗) *exception*

Tracks an error/crashes that has occured in the application. This errors appear in special sections of the analytics dashboards, therefore only use them to reflect app crashes or fatal errors. Send minor warnings with this method will clutter the ability to detect actual crashes causes.

**Parameters**

| | |
|---|---|
| *event* | The event to track. |
| *msg* | The message with the detail of the error. |
| *exception* | The exception that causes the error. |

#### 3.6.1.4 - (void) trackEvent: (NSString ∗) *event*

Tracks an event. The event could be actions taken on some object like unlocking an achievement, or a getting a hiscore.

**Parameters**

| | |
|---|---|
| *event* | The event to track. |

#### 3.6.1.5 - (void) trackEvent: (NSString ∗) *category* action:(NSString ∗) *action* label:(NSString ∗) *label* value:(int) *value*

Tracks an event under a particular category. The parameters match with the google analytics' ones. For flurry, a event with parameters is issued, where the category is the event name.

**Parameters**

| | |
|---|---|
| *category* | The category for this event. |
| *action* | The action took on that category. |
| *label* | Optional. A string label to especify something about the action. |
| *value* | Optional. A integer value. useful to especify error codes. |

#### 3.6.1.6 - (void) trackEvent: (NSString ∗) *event* withParams:(NSDictionary ∗) *params*

Tracks an event with particular parameters. This match the Flurry's logEvent:withParameters:

**Parameters**

| | |
|---|---|
| *event* | The event to track |
| *params* | Additional parameters to attach to the event. |

#### 3.6.1.7 - (void) trackPage: (NSString ∗) *page*

Tracks a particular page. It can be used to keep track of the current screen separetely from the events. If the session is not started yet, this request is ignored.

**Parameters**

| | |
|---|---|
| *page* | The page name/url to track. |

#### 3.6.1.8 - (void) trackTimedEvent: (NSString ∗) *event*

Tracks an event that should end in a finite time. If the tracking system support it natively the equivalent method will be used. Otherwise, an event with the start timestamp(epoch) is issued.

**Parameters**

| | |
|---|---|
| *event* | The event to track. |

The documentation for this category was generated from the following file:

- /Users/Shared/Jenkins/Home/workspace/native-ios-framework/src/Classes/public/Spil+Tracking.h

# Index