Instructions to the nexxPlay framework

1. Adding the framework to an existing project

1. Select your project, select your target and open the tab “General”
2. Drag nexxPlay.framework to “Embedded Binaries”
3. Make sure the framework also appears in “Linked Frameworks and Libraries” (by default you can find this section below “Embedded Binaries”)
4. Change your tab from “General” to “Build Phases”
5. Add nexxPlay.framework to the “Copy Bundle resources” section
6. ONLY Objective-C: Change your tab to “Build Settings” and make sure that “Embedded Content Contains Swift Code” is set to “YES” (the default value is “NO”)

2. Simple ViewController examples

Adding a PlayerView

Swift

```
import UIKit
import nexxPlay

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        let v_player = PlayerView(frame: CGRect(x: 0, y: 0, width: 300, height: 300))
        view.addSubview(v_player)

       //possible predefinitions
       v_player.overrideMidroll("http://vastURL", interval:5)
       v_player.overrideAutoPlay(true)


        v_player.startPlay("0", client: "571", playmode: "single", param: "40164")
    }
}
```

Objective-C

```
#import "ViewController.h"
#import <nexxPlay/nexxPlay.h>

@interface ViewController ()
@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    PlayerView* view = [[PlayerView alloc] initWithFrame:CGRectMake(0.0, 0.0, 300.0,
300.0)];
    [self.view addSubview:view];

    //possible predefinitions
    [v_player overrideMidroll:@"http://vastURL" interval:5];
    [v_player overrideAutoPlay:YES];

    [view startPlay:@"0" client:@"571" playmode:@"single" param:@"40164"
startPosition:0 startItem:0];
}

@end
```

Removing a PlayerView

As soon as the PlayerView is deallocated, the player will automatically stop.

3. Public methods

Defining VAST information prior to the player start

The following methods can be called prior to the initial method startPlay() in order to predefine information for the player. In case the server also provides this information when calling startPlay(), only the information that was not provided before startPlay() is used from the server.

```
public func overridePreroll(url:String)
```

Sets the URL for the VAST preroll.

```
public func overrideMidroll(url:String, interval:Int)
```

Sets the URL and time interval (minutes) for the VAST midroll.

```
public func overrideBanner(url:String, interval:Int)
```

Sets the URL and time interval (minutes) for the VAST banner roll.

```
public func overridePostroll(url:String)
```

Sets the URL for the VAST postroll

```
public func overrideAdProvider(provider:String)
```

Sets the ad provider.

```
public func overrideVASTData(provider:String, prerollURL:String,
midrollURL:String, midrollInterval:Int ,bannerURL:String,
bannerInterval:Int,postRollURL:String)
```

Sets all VAST information for the player.

```
func overrideMenu(visibility:Int)
```

Updates the setting defining whether the menu is never visible (0), visible on tap (1) or always visible(2)

```
func overrideAutoPlay(autoPlayMode:Bool)
```

Updates the setting defining whether videos should start automatically

```
func overrideExitMode(exitMode:String)
```

Updates the exitMode after the video, “reload” defines no exit page, “load” will show an exit page

```
func overrideAutoNext(autoNext:Bool)
```

Updates the setting defining whether the next video should start automatically after some seconds

```
func overrideStartMuted(startMuted:Bool)
```

Updates the setting defining whether the player starts muted or not

```
func setDelay(delay:Int)
```

Defines the start position of the video (in seconds).

```
func addObserverForAllNotifications(observer:AnyObject, selector:String)
```

Convenience function to add an `observer` on all notifications the player will send. Whenever a notification is sent by the player, the function defined by `selector` is called on `observer`. For more information see Notifications (4).

```
func removeObserverForAllNotifications(observer:AnyObject)
```

Convenience function to remove an `observer` on all notifications the player will send.

Controlling the player

```
func startPlay(sessionID:String, client:String, playmode:String,
param:String, startPosition:Int = 0, startItem:Int = 0)
```

Initial method to get the client data, ad data and video data. The attributes startPosition and startItem are optional.

```
func swap(param:String)
```

This method changes the video ID, thus the player will load the necessary information about the new video and restart with the new data

```
func swapToPos(newParam:String)
```

In case the player currently plays a playlist, it will switch to the video with the provided ID (newParam)

```
func swapComplex(playMode:String, param:String, startPosition:Int = 0,
startItem:Int = 0)
```

Like swap() this method changes the video ID but also lets you define a new playmode as well as the first item to be played (for playlists) and the position, the video is started from (in seconds). The attributes startPosition and startItem are optional.

```
func play()
```

The player starts/continues playing.

```
func pause()
```

The player pauses.

```
func mute()
```

The player is muted.

```
func unmute()
```

The player is unmuted.

```
func seek(time:Int)
```

The current video is set to the defined position (in seconds).

```
func getMediaData() -> NexxPlayMediaData?
```

Returns details about the current video. These details contain:

- mediaID: String
- hash: String
- title: String
- subtitle: String?
- teaser: String?
- description: String?
- runtime: String? (format “hh:mm:ss“)
- actors: String?
- channel: String?

If there is no current video, the method returns nil. If the method returns an object, the attributes mediaID, hash and title are definitely set. All other attributes are optional and may not be available.

4. Notifications

In order to observe the player's behavior the framework provides multiple notifications for the application. The notifications are:

nexxPlayErrorNotification
The player shows the error view

nexxPlayStartPlaybackNotification
The player did load the video data

nexxPlayStartPlayNotification
The player starts playing the content video. If there is an ad before the actual content video, this notification is fired once the ad is ended and the content will start

nexxPlayPlayNotification
Whenever the video starts to play (also after pause)

nexxPlayPauseNotification
Whenever the video pauses

nexxPlayEndedNotification
Whenever the video is finished

nexxPlayPlayPosNotification
The player switches to a new video in the playlist

nexxPlayAdStartedNotification
The player starts a video ad

nexxPlayAdEnded Notification
A video ad is finished (or skipped if possible)

nexxPlayAdError Notification
An error regarding the video ad occured

nexxPlayAdClicked Notification
The video ad was clicked

nexxPlayAdResumedNotification
The video ad continues playing after the user clicked on it

Observing notifications

The notifications are sent by the NSNotificationCenter. To receive a notification you can use the following code snippet:

Swift

```
NSNotificationCenter.defaultCenter().addObserver(self, selector:
"notificationReceived:", name: nexxPlay.nexxPlayErrorNotification, object: v_player)
```

Objective-C

```
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(notificationReceived:) name:@"NexxPlayErrorNotification"
object:playerView];
```

The appropriate function that is called when the notification is received:

Swift

```
@objc func notificationReceived(notification:NSNotification) {
        println(notification.name)           // prints “NexxPlayErrorNotification”
        if notification.name == nexxPlay.nexxPlayPlayPosNotification {
            if let userInfo = notification.userInfo {
                if let position = userInfo["position"] as? NSNumber {
                    println(“position: \(position.intValue)”)
                }
            }
        }

}
```

Objective-C

```
- (void)notificationReceived:(NSNotification*)notification {
    NSLog(notification.name);
    if ([notification.name isEqualToString:@"NexxPlayPlayPosNotification"]) {
        if (notification.userInfo != nil) {
            NSNumber* position = [notification.userInfo valueForKey:@"position"];
            if (position != nil) {
                NSLog(@"position: %d", position.intValue);
            }

        }
    }
}
```

- If you have multiple players you can determine the player that sent the notification by checking `notification.object`
- The notification `nexxPlayNotificationPlayPos` contains additional data , this data is stored in `notification.userInfo` with the key “position”. To ensure compatibility with Swift and Obj-C, this value is stored as a NSNumber.

If you want to receive all notifications from the player, there is a convenience function:

Swift

```
v_player.addObserverForAllNotifications(observer, selector:”notificationReceived:”)
```

Objective-C

```
[playerView addObserverForAllNotifications:self selector:@"notificationReceived:"];
```

Whenever a notification from `v_player` is received by `observer`, the function `notificationReceived:` is called. With `notification.name` you can determine which notification was received and act recordingly.

5. Fullscreen and orientation behavior

Fullscreen

When the player switches to fullscreen, a subview (containing the video and all controls) of the PlayerView is added to the applications keyWindow (`UIApplication.sharedApplication().keyWindow`). Once the user switches back, the subview is added back to the PlayerView.

Orientation behavior

When the device is rotated, the video automatically rotates accordingly.