Player SDK for iOS

Programming Guide

Introduction

RayV Player SDK for iOS is a set of libraries for developing iOS players that can stream and display RayV based live TV channels. The SDK supports iOS 3.0 or higher on both iPhone and iPad although some higher level APIs require iOS 3.2 or higher. Applications using the SDK should be built with iOS SDK 4.2 or higher. The SDK can be used with both iOS devices and iOS simulators.

For other operating systems, RayV provides a C/C++ SDK with which provides low level streaming APIs and requires developer to provide their own media rendering code. However since iOS does not provide any API for decoding H.264 video, implementing an efficient, hardware accelerated, renderer is not a simple task. Therefore RayV provides an iOS specific layer that translates, locally on the device, the RayV proprietary protocol into standard HTTP Live Streaming protocol (HLS). This layer provides the following benefits:

- A simplified Objective-C API for the RayV Player SDK.
- Standard HLS protocol output allows using standard iOS media player with all the standard user interface and features.
- Efficient hardware accelerated media decoding and rendering allowing live channels with high quality.

A RayV specific movie player is also part of the SDK, hiding all the RayV specific code and making it even easier to develop RayV based applications.

This document provide high level guide for using the SDK. More information can be found in the SDK header files.

More information regarding playing RayV channels on iOS device can be can be found in *RayV iOS Player Operations Guide*.

RayV Player SDK for iOS files

The SDK is packaged in a zip file and should be extracted to a folder on the development machine. Note that only Release configuration version of the libraries are provided. The Debug configuration folders simply contain a symbolic links to the Release version for convenience.

The SDK include the following folders and files:

Common

ViewerCoreEvents.h – Constants with keys and values that are used by the SDK notifications.

RayV_iPhone_P2PStreaming

RVMoviePlayerController.h – RayV movie player object that plays RayV channels and has a similar API to MPMoviePlayerController.
RVMediaPlayer.strings – strings file that is used by RVMoviePlayerController user interface.
SupportInfoController.h – Object for displaying RayV related support information.
HttpStreamer.h – The object that allows streaming a RayV channels and receiving its media via HLS protocol.
IPhonePlayerCore.h – An Objective-C wrapper of the RayV Player SDK
IPhonePlayerSession.h – An Objective-C wrapper of the RayV Player SDK viewing session. Can be ignored in most applications since the HttpStreamer uses this object internally and provides all the needed interfaces.

build

Folder containing different builds of the RayV_P2PStreaming_iPhone library.

RayV_SDK

build

Folder containing different builds of the RayV_SDK_iPhone library.

test

RayV SDK for iOS Test

A sample application that demonstrates playing a RayV channel using the RVMoviePlayerController object. The sample also uses SupportInfoController to display RayV related support information. The sample supports only iOS 3.2 or higher and does not support multitasking.

IPhonePlayerCore

IPhonePlayerCore is the object that creates and manages the RayV Player SDK. It is in charge of the following:

- Connecting to the RayV Grid Server, passing the information about the application and SDK and keeping the connection alive.
- Creating, configuring and managing the P2P engine.
- Creating viewing sessions for playing RayV channels.
- Providing internal, support/debugging related, information to the application.
- Handles application entering background state.
- Handles WiFi/3G switching.

The IPhonePlayerCore is a singleton and only one instance of it can be created. It is created with the following code:

```
IPhonePlayerCore* playerCore =[[IPhonePlayerCore alloc]
initWithDistirbutor:@"distributorId" versionMajor:1 versionMinor:0 versionRevision:0
versionBuild:0];
```

Each company using RayV has its own distributor ID and should use it to identify its players. The distributor ID and application version are used for statistics and service rules.

The application should then start the SDK using the following code:

```
[playerCore start];
```

This will connect to the Grid and then start the P2P engine.

When the application terminates it should stop the SDK using the following code:

```
[playerCore stop];
```

This will disconnect from the Grid and stop the P2P engine.

When the application enters background state in iOS 4.0 or higher, IPhonePlayerCore will stop everything internally, keeping only session objects alive but not streaming. When the application enters foreground state IPhonePlayerCore will resume everything and the sessions will continue streaming.

If IPhonePlayerCore uses 3G and WiFi becomes available then IPhonePlayerCore will silently switch to use WiFi for connecting to the RayV Grid. It will do so only when no video is streaming or right after video streaming stops.

RVMoviePlayerController

RVMoviePlayerController is an object that internally uses RayV HttpStreamer and Apple MPMoviePlayerController to play RayV channels and takes care of the following:

- Create and manage an HttpStreamer object.
- Display user interface before the MPMoviePlayerController is displayed.
- Handle session state and display session progress user interface.
- Display peer-to-peer information for debugging.
- Handles iOS notifications for efficient use of device resources (e.g. stops peer-to-peer when playback is paused by user, application becomes inactive or device goes to sleep).

RVMoviePlayerController supports only iOS 3.2 or higher. If iOS 3.0/3.1 support is required then use HttpStreamer directly.

RVMoviePlayerController does not yet support multitasking and its state might be undefined when entering foreground state after being in background state.

The RVMoviePlayerController implements a subset of MPMoviePlayerController API so the original MPMoviePlayerController documentation can be used as a reference. Here is the API reference with a description of issues related to RVMoviePlayerController only:

Instance Methods

```
- (id)initWithContentURL:(NSURL *)url
```

The URL should have the format "rayv:channelkey". For example for channel iphone1demo the URL should be "rayv:iphone1demo".

```
- (void) rvMakeRotateViewByOrientation:(UIInterfaceOrientation)interfaceOrientation
```

This is a method not available in MPMoviePlayerController. This method tells the RVMoviePlayerController the current orientation so it can display correctly its overlay in full screen mode.

```
- (void)play
- (void)pause
```

Identical to MPMoviePlayerController methods. Ignored until the video starts.

Properties

```
@property(nonatomic, readonly) UIView *view
```

The RVMoviePlayerView view can added to the application view hierarchy right away without dependency on the streaming progress.

```
@property(nonatomic, retain) NSURL *contentURL
```

This property can be used to retrieve the current channel or to switch channels, using the same URL scheme described in initWithContentURL.

```
@property(nonatomic) BOOL fullscreen
@property(nonatomic, readonly) MPMoviePlaybackState playbackState
@property(nonatomic, readonly) MPMovieLoadState loadState
@property(nonatomic) MPMovieScalingMode scalingMode
```

Identical to MPMoviePlayerController properties. Note that setting the properties until the video start is ignored.

Notifications

The following notifications are supported and are identical to the MPMoviePlayerController notifications, just replacing MP prefix with RV:

- RVMoviePlayerDidEnterFullscreenNotification
- RVMoviePlayerDidExitFullscreenNotification
- RVMoviePlayerWillEnterFullscreenNotification
- RVMoviePlayerWillExitFullscreenNotification
- RVMoviePlayerScalingModeDidChangeNotification
- RVMoviePlayerLoadStateDidChangeNotification
- RVMoviePlayerPlaybackStateDidChangeNotification

SupportInfoController

This is a View Controller that can be used to display RayV related support information. If the view is part of a navigation view controller then a refresh button will be added to the navigation bar for allowing refresh of the support information.

The SuppotInfoController is used as any standard UIViewController and has one method to refresh the information.

The SuppotInfoController retrieves support information from IPhonePlayerCore by calling the infoOfType method with “support” as the type parameter.

HttpStreamer

It is highly recommended to use RVMoviePlayerController for faster development and for gaining future RVMoviePlayerController developments done by RayV. However sometimes there is a need to use the lower level HttpStreamer, for example in order to support iOS 3.0/3.1 or for implementing a movie player with a different user interface.

The HttpStreamer uses the IPhonePlayerCore to create a player session, stream a RayV channel and stream the channel media using the HLS protocol to a standard MPMoviePlayerController object.

HttpStreamer does not yet support multitasking and its state might be undefined when entering foreground state after being in background state.

The HttpStreamer is created with the following code:

```
HttpStreamer* httpStreamer = [[HttpStreamer alloc] initWithDelegate:self];
```

Assuming the object creating it is implementing the HttpStreamerDelegate protocol for receiving session related events.

The channel is then set using the following code:

```
httpStreamer.channel = @”yourchannel”;
```

HttpStreamer is then started with the following code:

```
[httpStreamer start];
```

In order to start streaming the session should be activated with the following code:

```
[httpStreamer setSessionActive:YES];
```

Your object that implements HttpStreamerDelegate will now start to receive session state messages to:

```
(void)sessionStateDidChange:(NSDictionary*)event
```

The session state can be queried by the following code:

```
[event valueForKey:EVENT_NEW_SESSION_STATE]
```

If an error occurred the error can be retrived by the following code:

```
[event valueForKey:EVENT_ERROR]
```

If the session goes into SESSION_INACTIVE state and the error is not an empty string then something went wrong. If all is fine then the session will always try to resume streaming, even if connection was gone for a while.

Once enough media was streamed, your object will receive the following message:

```
(void)mediaSegmentsReady
```

The local HLS stream is now ready and can be used for playback using the standard MPMoviePlayerController object. To get the HLS URL use the following code:

```
NSString* url = [NSString stringWithFormat:@"http://localhost:%d/playlist.m3u8",
httpStreamer.port];
```

The local HLS stream will be available as long as the HttpStreamer is not stopped, even if there is no external Internet connection.

Note that the MPMoviePlayerController might go to pause state if pause by the user or if the application becomes inactive (call or message received, device goes to sleep, etc.). The HttpStreamer is not aware of the pause and will continue to stream and waste device resources. In order to stop the streaming but keep the HttpStreamer alive and connected to the MPMoviePlayerController, the session should be stopped using the following code:

```
[httpStreamer setSessionActive:NO];
```

To stop the HttpStreamer completely (and the viewing session it uses internally) use the following code:

```
[httpStreamer stop];
```

The HttpStreamer must be stopped and released before stopping and releasing the IPhonePlayerCore!

Linking with the SDK libraries

In your project (or target) settings add the following to the Other Linker Flags setting:

SDK_FOLDER/build/$(CONFIGURATION)$(EFFECTIVE_PLATFORM_NAME)/libRayV_SDK_iPhone.a

SDK_FOLDER/RayV_iPhone_P2PStreaming/build/$(CONFIGURATION)$(EFFECTIVE_PLATFORM_NAME)/libRayV_P2PStreaming_iPhone.a

Where SDK_FOLDER is the path where the SDK is installed.

In addition your project will need to link with the C++ runtime library. The easiest way to force Xcode to do this is to include some C++ file in the project. In the SDK sample this was achieved by using main.mm instead of main.m for the main application file.

Your project will also need to link with the following frameworks: SystemConfiguration, Foundation.framework and CFNetwork.framework.