Android Limelight VideoSDK

version 0.1

Last updated : 8-Dec-2014

By: Kanchan Kumar <[kanchan.kumar@rebaca.com](mailto:kanchan.kumar@rebaca.com)>

# Approval

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Date** |
| LimelightVideoSDK |  |  |
| QA |  |  |
| Ops |  |  |
| Architect |  |  |
| Product |  |  |
| Management |  |  |
| Agile |  |  |
| T4 support |  |  |

# Table of Contents

[Approval](#h.rdx750lsm1e1)

[Table of Contents](#h.3oev3nnzggco)

[Prerequisite](#h.sntl71h7pgia)

[Open Issues](#h.x1c5zcpleqtx)

[Overview](#h.nh977ixs671z)

[Goal](#h.aab4qzq4e6v9)

[Product Requirements](#h.97sffjv3iikb)

[MileStone 0.1](#h.yh661qj6gpr4)

[Milestone 0.2](#h.yck22cppwqxe)

[MileStone 0.3](#h.klzluwchhsr6)

[MileStone 0.4](#h.eqxr172t16vb)

[Product Description](#h.sk95po9r1kxx)

[General strategies](#h.2k8voixqmdfu)

[Naming Conventions](#h.hn4ujg3q12o1)

[LimelightVideoSDK Flow](#h.fzubuygf5fks)

[MileStone 0.1](#h.dt7ujz9c3ppz)

[Public Classes](#h.vgg52hqsn1b9)

[Player](#h.herjzww7u4g5)

[Sample code for developers to use Player:](#h.t2rzoghy03ei)

[LoggerUtil](#h.jgxjms8h9hbw)

[Package Classes](#h.1tzgc3ydovt4)

[TimeUtil](#h.9ne3e48ga91k)

[LogConfigurator](#h.sqhulqc9f853)

[Private Classes](#h.urv3ni99rsmj)

[PlayerControl](#h.aw0f1b2vagu9)

[Public API List](#h.3ozd1yn8wppv)

[Player](#h.vwb9hch3kurx)

[LoggerUtil](#h.mostfbdd25qz)

[Package API List](#h.avu7cs4x5ywt)

[LoggerUtil](#h.53gge97mjbnu)

[Sample code for using LoggerUtil](#h.f8e936k1y0gr)

[TimeUtil](#h.jtvvifep4156)

[LogConfigurator](#h.7mcsvbnm0tei)

[Private API List](#h.sif8xaji5gp1)

[PlayerControl](#h.vtg0ex80zpmk)

[Interfaces](#h.gush5oserra2)

[IPlayerCallback](#h.v9mfd34704td)

[IPlayerControl](#h.9o1fw98dg74p)

[Logging](#h.sdqdnyo36qxv)

[Statement of Work](#h.dx8058ib3w6)

[Monitoring](#h.a8cnmlv0mqnt)

[Cross-functional Dependencies](#h.acpyfy6mdrgu)

[Deployment Process](#h.j3n5ay52pbf5)

[Test Case Recommendations](#h.m1n5y9qy3wcz)

[JIRA Items Addressed](#h.nsobc6rl66k5)

[Opensource Libraries](#h.5cvq7u9fg3j3)

[Copyright Licenses](#h.a8uaj1mcr6t)

[LogConfigurator.java](#h.2vzi7klteb1)

[Log4J](#h.1zrmsknywnum)

[Build Environment](#h.oqd6wwbp0ln1)

[Software Requirements](#h.mgeqsbfdexhg)

[Components](#h.aixkvxhx0ml3)

[Final Deliverable](#h.bgeb5yw8mnx2)

[Development SetUp](#h.n1h7bi22luv9)

[Android LimelightVideoSDK Application](#h.r2ofdq8okqyb)

[Creating a single Jar](#h.wgldcwz3hwtk)

[Creating Application using LimelightVideoSDK in Eclipse](#h.a5qnwsas2df8)

[Creating Application using LimelightVideoSDK in Android Studio](#h.9fmd4srmglui)

[References](#h.mhdcsefdysb8)

# 

# 

# 

# 

# 

# 

# P**rerequisite**

Developer should have a Limelight Orchestrate Video account and they need to supply their secret and access key to the LimelightVideoSDK in order to access media content from Limelight Network.

# Open Issues

No open issue as of now.

# Overview

Currently, there is Video kit for Flash and iOS to enable customer application developers to develop applications for Fetching content from Limelight network and playing it in a video player in a very short time.This way they can start focusing on other aspects of the application.

Video SDK with similar functionalities has to be implemented in Android.

This 0.1 version will include Player development,Time helper development and Logging support.

In subsequent releases, we would like to implement analytics support, URL signing, date utilities, thumbnail image downloader and notification center.

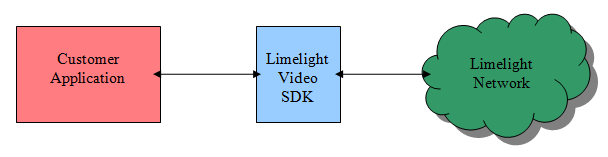


Figure 1: Overall System Diagram

# Goal

To develop LimelightVideoSDK for android similar to iOS Video Kit SDK.

# 

# Product Requirements

## MileStone 0.1

* Player implementation which can be embedded into customer application container view.
* Player should also have a UI control bar.
* Player should support HLS playback.
* Player should support H264 playback.
* Time should be formatted to hh:mm:ss format for hour > 0 and mm:ss for hour < 0.
* Logging should be done using apache Log4J.
* A test application should be provided to test the LimelightVideoSDK functionalities.

## Milestone 0.2

* LimelightVideoSDK should have methods to get MD5 Hash and SHA256 Hash of supplied string.
* Request URL should be signed with developer’s access key and secret.
* LimelightVideoSDK should Implement date extensions.
* LimelightVideoSDK should support for Widevine DRM’d content online and offline playback.

## MileStone 0.3

* LimelightVideoSDK should notify when the given host becomes available and unavailable.
* LimelightVideoSDK should asynchronously download an Image associated with Channel and media.
* LimelightVideoSDK should show a buffering icon when player is out of buffer and connection exists.
* LimelightVideoSDK should show a “Connection Lost message” when player is out of buffer and connection does not exists.
* LimelightVideoSDK should deallocate player if user navigates back in app.
* Implementation of Channel, Media, Library,Delivery,Playlist.

## 

## MileStone 0.4

* LimelightVideoSDK should support playlists.
* LimelightVideoSDK should have analytics reporter.
* LimelightVideoSDK should have analytics observer.
* LimelightVideoSDK should support notifications.
* LimelightVideoSDK should support full screen player.

# Product Description

## General strategies

* The LimelightVideoSDK should do all the heavy lifting like URL signing, authentication, fetching media content, analytics and logging.
* Customer application developer should use APIs exposed by LimelightVideoSDK to develop application and they should not bother about using the Limelight Orchestrate Video Content APIs. This makes their application development easier.
* Customer application will include LimelightVideoSDK in it and this SDK will be used to communicate with Limelight network through LimeLight orchestrate APIs. Customer application will not directly communicate with Network.
* Since the Player has to be an embed-able component, Player cannot be an activity. It has to be either fragment or a layout or just video view so that it can embed in an activity inside the customer application. Here fragment has been chosen because,
  + It is flexible in usage.
  + Fragment lifecycle can be leveraged for managing Player states in a more better way.
  + Fragments can be used to create rich UI customer applications.

Sample usage provided at [sample code to embed player](#h.t2rzoghy03ei)

* Log4J is considered over Java Util Log (JUL) for logging because,
  + Log4J is thread-safe.
  + Log4J is optimized for speed.
  + Log4J supports multiple output appenders per logger.
  + Log4J supports internationalization.
  + Logging behavior can be set at runtime using a configurator.
  + The format of the log output can be easily changed by extending the *Layout* class.
  + The target of the log output as well as the writing strategy can be altered by implementations of the Appender interface.
  + Some important appenders that Log4J has are:

SMTPAppender, SocketAppender and RollingFileAppender.

* + - SMTPAppender can send the logs in mail.
    - SocketAppender can write logs to a socket.
    - RollingFIleAppender can write logs to a file that can be used separately in various manners.
  + So ,whatever JUL can do, Log4j can also do - and more." They differ most in the areas of useful appender/handler implementations, useful formatter/layout implementations, and configuration flexibility.
* We are going to use Log4J 1.2.9 version because it is a stable release and also tested on android.
* The public access specifiers will be used to expose the classes and methods and default/private access specifiers will be used for library internal classes and methods.

## Naming Conventions

* In general, Java naming conventions and coding guidelines will be followed.
* The library name will be LimelightVideoSDK.
* The package name will be com.limelight.videosdk.
* The interfaces names will start with “I”.
* All class names starts with capital letter.
* The detail list of names of public classes, private classes,package classes,public APIs, private APIs and Interfaces are mentioned at [Public Classes](#h.vgg52hqsn1b9) , [Package Classes](#h.1tzgc3ydovt4) , [private classes](#h.urv3ni99rsmj), [Public API List](#h.3ozd1yn8wppv) ,[Package API List](#h.avu7cs4x5ywt), [Private API List](#h.sif8xaji5gp1) and [Interfaces](#h.gush5oserra2).

## 

## 

## 

## LimelightVideoSDK **Flow**

### MileStone 0.1

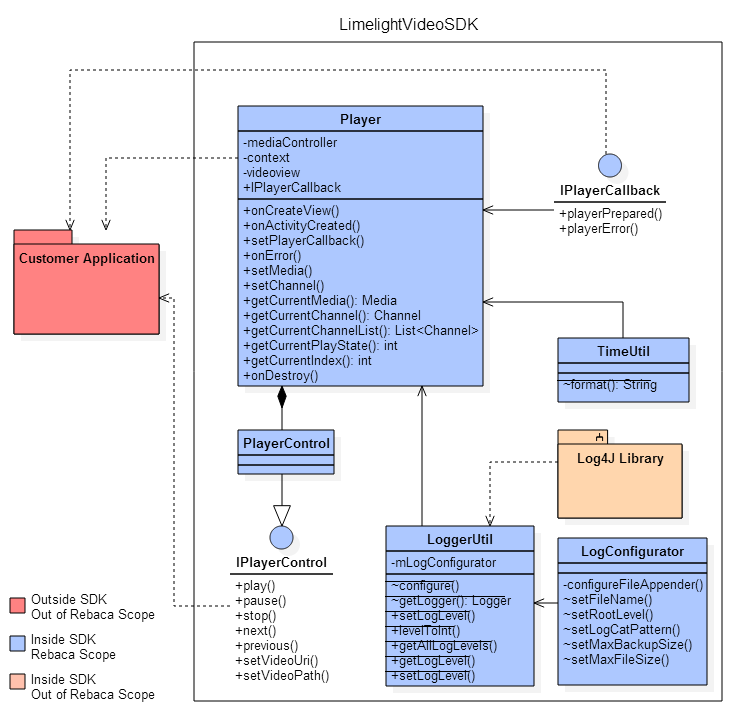


Figure 2: 0.1 Deliverable Related Design Diagram

This design diagram is specifically related to 0.1 delivery. Player is the public API that will be used by customer application to access LimelightvideoSDK Player. Log4J library is an open source library and is used for Logging purpose. It will be part of LimeLightVideoSDK but its development is out of scope of this tech spec.

## Public Classes

### Player

This class is the customized fragment which also contains native android player.

It also includes media controller object which is the UI controller for the player. Media Controller has UI controls like seekbar, play button,pause button,next button, previous button.Currently it is using default mediacontroller, if needed we will extend the media controller class for customizing the UI control.

It also has PlayerControl as an inner final class.This is the implementation of [IPlayerControl](#h.4zzdlh2o977r). This is the control to the Player. It is supplied to the Activity which requested for the Player.

It also has IPlayerCallback which holds the IPlayerCallback implementation sent from activity. It is used to interact from Player to activity which requested for Player.

Player will be available as an embed-able component in customer application container view. It is made embed-able by extending the fragment.

Since the Player has to be an embed-able component, Player can not be an activity. It has to be either fragment or a layout so that it can embed in an activity inside the customer application. Here fragment has been chosen because of its flexibility in usage.

Player along with List support like channel list and media list will be handled in subsequent releases

#### Sample code for developers to use Player:

Player player = new Player();

getFragmentManager().beginTransaction().add(R.id.container,player).commit();// Embeds the //player into container layout.

player.setVideoUri(mediaUri)

player.setPlayerCallback(IPlayerCallback);

Player class is public and has many public APIs. Refer [Public Api List](#h.3ozd1yn8wppv)

### LoggerUtil

This class has methods to configure logging and to log the messages. It uses Apache Open source logging library Log4J for logging. As of now, we will use RollngFileAppender which can log information to a particular file as configured by LimelightVideoSDK. In future, if required we can use SMTPAppender or SocketAppender or so.

This class will be accessible to customer application so that they can do some little configuration for logging. As of now only few methods like setLogLevel() will be exposed for customer application. With this method customer application developer can set the level of logging done by SDK.

The priority order of levels is : 1.Fatal 2.Error 3. Warn 4. Info 5. Debug

Logs below or equal to current level will only be logged. eg: If current level is set to error, then only error and fatal logs will be logged. Off turns off the logging.

LoggerUtil also converts the log levels into some integer values ranging from 0 to 6. Log4j Levels integer values are difficult to use in code as they are not in sequence. So this method maps Log4J level values to normal continuous integer values.

## Package Classes

### TimeUtil

This is a utility class to format the time into hh:mm:ss format for hour> 0 and mm:ss for hour< 0.

This class will not be accessible to customer application as it will be given package access specifier. TimeUtil will be used only within LimelightVideoSDK.

### LogConfigurator

LogConfigurator class configures the logger like resetting the configuration, adding appender to the logger, configuring file appender which includes maximum number of backed up log files,maximum size of log file until rolling and logging format.

## 

## Private Classes

### PlayerControl

This is an inner final class of Player.This is the implementation of [IPlayerControl](#h.4zzdlh2o977r). This is the control to the Player.It is supplied to the Activity which requested for the Player.

## 

## Public API List

### Player

onCreateView() - Called to have the fragment instantiate its user interface view.

setPlayerPreparedListener() - To set the IPlayerPreparedListener implementation.

onActivityCreated() - Called when the fragment's activity has been created and this fragment's view hierarchy instantiated.

setMedia() - To set the media in player, but does not starts playing.

setChannel() - To set a channel fetched from Limelight network but does not starts playing.

setDataUri() - To set a data uri in player, but does not starts playing.

getCurrentMedia - To get the current media loaded/set in player.

getCurrentChannel() - To get the current channel loaded/set in player.

getCurrentIndex() - Returns the index of the current video in the channel.

getCurrentPlayState() - Returns the state of the player.

### LoggerUtil

setLogLevel() - Developer can set the level of logging done by SDK.

getLogLevel() - Method to return the current log level.

getAllLogLevels() - Method to return all levels.

levelToInt() - This method converts the log levels into some integer values ranging from 0 to 6.

## 

## 

## Package API List

### LoggerUtil

configure() - Method to configure Logging.

getLogger() - Method to return a logger object.

#### Sample code for using LoggerUtil

LoggerUtil.configure( "LLVSDK.log" , "%d - [%p::%c::%t] - %m%n" , 5 , 1024);

LoggerUtil.getLogger( “SDK” ).error( "XXXX" );

### TimeUtil

format() - Method to format the time into hh:mm:ss format for hour> 0 and mm:ss for hour< 0.

### LogConfigurator

configure() - Method to configure logging.It resets the configuration. It adds appender to the logger.

configureFileAppender() - Method to configure file appender.It includes maximum number of backed up log files,maximum size of log file until rolling and logging format.

setFileName() - Sets the name of the log file

setMaxBackupSize() - Sets the maximum number of backed up log files

setMaxFileSize() - Sets the maximum size of log file until rolling

## Private API List

### PlayerControl

play() - To programmatically play player.

stop() - To programmatically stop player.

pause() - To programmatically pause player.

next() - To programmatically go to next in player.

previous() - To programmatically go to previous in player.

setVideoUri(Uri uri) - To set a video content uri in player, but does not starts playing.

setVideoPath(String path) - To set a video content path in player, but does not starts playing.

## Interfaces

### IPlayerCallback

This is the callback to be sent to activity from Player. This is the communication mechanism between activity in application and player in SDK.

Once Activity holding the player fragment gets created, callback is sent to activity. Activity can then use playerPrepared() to start interacting with Player.The rationale behind this approach is that customer application should start using Player only after activity holding the player fragment gets created.

Player can also send error information back to the activity in Application.Activity then can use this error information.

This interface is public and has to be implemented in activity which wants to use Player.

playerPrepared(IPlayerControl control) - Method will be implemented in activity. IPlayerControl will be used to control Player.

playerError(int what,int extra) - Method will be implemented in activity. Player will send error information like type of error that has occurred and an extra code, specific to the error to the customer application.

### 

### IPlayerControl

This interface is the control to the Player.

A child object of this interface is supplied to the Activity which requested for the Player.

This interface is public and and has to be used in activity which wants to use Player.

play() - To programmatically play player.

stop() - To programmatically stop player.

pause() - To programmatically pause player.

next() - To programmatically go to next in player.

previous() - To programmatically go to previous in player.

setVideoUri(Uri uri) - To set a video content uri in player, but does not starts playing.

setVideoPath(String path) - To set a video content path in player, but does not starts playing.

# Logging

Logging for this project will be done by LoggerUtil and configured by LogConfigurator class. These will be backed up by Apache open source library Log4J. Please refer [General strategies](#h.2k8voixqmdfu) for rationale behind choosing Log4J.

LoggerUtil class has methods to configure logging and to log the messages. LoggerUtil class will be accessible to customer application so that they can do some little configuration for logging. As of now only setLogLevel() will be exposed for customer application. With this method customer application developer can set the level of logging done by SDK.

LogConfigurator class configures the logger like resetting the configuration, adding appender to the logger, configuring file appender which includes maximum number of backed up log files,maximum size of log file until rolling and logging format.

We are currently using RollngFileAppender which can log information to a particular file as configured by LimelightVideoSDK. In future, if required we can use SMTPAppender or SocketAppender or so. The file size and number of files can be configured for RollingFileAppender. SMTPAppender can send the logs in mail. SocketAppender can write logs to a socket.

We are going to use Log4J 1.2.9 version because it is a stable release and also tested on android

# Statement of Work

As this is an ongoing project, we have story breakdown in PMS

TBD : PMS story link here

# Monitoring

TBD

# 

# Cross-functional Dependencies

This project uses some open source libraries. Apache Log4J library is used for logging because of its above mentioned benefits. Log4J is an ongoing project but we are using the tested stable release of Log4J.

# Deployment Process

TBD

# Test Case Recommendations

End to end tests will be done to test the functionality supported by LimelightVideoSDK . A test application will be written to do functionality test.

# JIRA Items Addressed

TBD Jira link here

# **Opensource** Libraries

* Log4J - Open source library for logging. (1.2.9 version)

# Copyright Licenses

## LogConfigurator.java

* + Apache License, Version 2.0 - LogConfigurator.java is under the Apache License. The copyright license header is mentioned in the file.Copy of the License can be found at http://www.apache.org/licenses/LICENSE-2.0.

## Log4J

* + Apache License, Version 2.0 - Log4J is under the Apache License.Copy of the License can be found at http://www.apache.org/licenses/LICENSE-2.0.

# Build Environment

## Software Requirements

* Android SDK 4.0
* Eclipse ADT Bundle ver. 23 OR Android Studio 0.8
* Java 1.6

## 

## Components

* Android LimelightVideoSDK
* Open source library – Log4J
* TestLimelightVideoSDK - Test Application

## Final Deliverable

* TestLimelightVideoSDK - Test application
* Android LimelightVideoSDK.jar

## Development SetUp

### Android LimelightVideoSDK Application

1. Create an android application in ADT bundle's eclipse. Name it as LimelightVideoSDK.
2. Mark isLibrary to true in LimelightVideoSDK application properties.
3. Include Open source Library Log4J jar in the libs folder and refer it in LimelightVideoSDK application class path.
4. Create a package named com.limelight.videosdk under /src folder.
5. Start writing classes in the package.
6. Build the LimelightVideoSDK application.
7. The LimelightVideoSDK.jar can be retrieved from /bin folder.

### 

### 

### Creating a single Jar

To Obtain a single Jar, we need to modify the LimelightVideoSDK.jar retrieved from LimelightVideoSDK /bin folder. This Jar includes the Rebaca Developed class files.

* Open LimelightVideoSDK.jar using 7zip.
* Copy/paste the .class files from open source jars into this opened LimelightVideoSDK.jar.
* Close opened LimelightVideoSDK.jar.

Now the LimelightVideoSDK.jar will contain all the classes. This Single Jar will be the deliverable.

### 

### Creating Application using LimelightVideoSDK in Eclipse

1. Create an android application in ADT bundle’s eclipse.
2. Include LimelightVideoSDK.jar in the libs folder and add it as a jar in application class path by configuring application build path.
3. Create a package and write classes and methods to access LimelightVideoSDK APIs in “src” folder. Write layout and resources under “res” folder.

### Creating Application using LimelightVideoSDK in Android Studio

1. Create a new Project in Android Studio.
2. Write classes and methods to access LimelightVideoSDK APIs under “app” folder.Write layouts and resources under “res” folder
3. Copy LimelightVideoSDK.jar in the libs folder and add it as a file dependency using Project Structure – app – dependency -- + -- File Dependency.Or modify build.gradle file of the app by adding compile files('libs/limelightvideosdk.jar') under dependency.

# References

* Android Sdk\_Proposal\_engineering 09.19.2014.pptx
* LL\_iOS\_SDK\_docs
* VPS-iOSSDKFunctionalSpecification-101114-0543-220
* New Player Design & Customization - Light Weight (LVP-00071) - Video PM
* <http://support.video.limelight.com/support/docs/player_api/>