Vidgyor Player Media3 SDK

Vidgyor Player Media3 SDK

- **Group / Artifact:** co.silverpush:vidgyor-player-media3
- **Latest Version:** 1.0.3
- **Distribution:** Maven Central
- **Min SDK:** 23 (24+ recommended) · **Compile / Target SDK:** 36 · **Java Version:** 17
- **Flavors:** mobile, tv (choose via missingDimensionStrategy)
- **Public API Package:** co.silverpush
- **PlayerView:** com.silverpush.player.ui.SpPlayerView

1. Overview

The **SpPlayer SDK** (artifact: vidgyor-player-media3) is an Android video player library built on top of **AndroidX Media3 (ExoPlayer)**. It provides a turnkey solution for:

- **Live stream** and **VOD** playback (HLS).
- **Pre-roll** and **Mid-roll** ad insertion (VAST / VMAP via IMA).
- **SCTE-35** and **Polling** based mid-roll ad signaling.
- **VMAP multi-break ads on VOD** (pre / mid / post breaks declared in a single VMAP tag).
- **Mid-roll slate fallback** when an ad fails to load.
- **Native IMA ad UI** with full D-pad focus handling on TV (follows IMA SDK customization guidelines).
- **Customizable player overlay** including quality, audio, speed, fullscreen, captions, and live badge.
- **Sensor-based orientation / fullscreen** (auto-rotate, or app-owned rotation).
- Channel **logo overlay** for Live Stream.
- Built-in **analytics** using Firebase and custom Silverpush analytics.
- Automatic network monitoring and error recovery.

IMPORTANT: The Activity hosting the player **must** extend AppCompatActivity or FragmentActivity.

2. Prerequisites

Requirement	Minimum
Android Studio	Ladybug or later
Android Gradle Plugin	9.x

Requirement	Minimum
Min SDK	23 (24+ recommended)
Compile SDK	36
Java / Kotlin	17
Core Library Desugaring	Required (desugar_jdk_libs 2.x)
Firebase Analytics (optional)	Valid google-services.json if enabled

3. Installation

3.1 Maven coordinate

```kotlin
implementation("co.silverpush:vidgyor-player-media3:1.0.3")
```

3.2 App build.gradle.kts

In the defaultConfig block, you must select the flavor using missingDimensionStrategy("device", "mobile"). Ensure isCoreLibraryDesugaringEnabled is set to true and transitive dependencies for Media3, Networking, and Image loading are added.

The SDK publishes its Media3, Retrofit/OkHttp, Glide, and Firebase dependencies as api(...), so they arrive **transitively** through the single SDK dependency — you only need to declare them directly if your own code uses them. (See the Integration Guide for a full snippet)

4. Mobile vs TV Flavor

The SDK uses product flavors with a dimension named device.

Variant	missingDimensionStrategy
Mobile (phones / tablets)	missingDimensionStrategy("device", "mobile")

Variant	missingDimensionStrategy
TV (Android TV)	missingDimensionStrategy("device", "tv")

5. Permissions

The SDK manifest declares the following permissions, which are merged automatically:

- android.permission.INTERNET
- android.permission.ACCESS_NETWORK_STATE
- android.permission.ACCESS_WIFI_STATE
- android.permission.CHANGE_NETWORK_STATE
- android.permission.WAKE_LOCK

6. Layout Setup

Add the custom SpPlayerView to your Activity layout. All control layouts are bundled inside the AAR.

```xml
<com.silverpush.player.ui.SpPlayerView
    android:id="@+id/playerView"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

7. Quick Start — Live

To integrate a live stream, configure the SpPlayerConfig, define the LiveMedia item, set up optional listeners, and create the player using SpPlayer(...).createLive(media). Manage the player lifecycle in onResume, onPause, and onDestroy.
See the Integration Guide 4.1 for a full example.

8. Quick Start — VOD

For VOD, you can use a DSL-style config builder. Both mediaId and url are required in the VodMedia model. Create the player using .createVod(media).
See the Integration Guide 4.2.

9. Configuration Reference

`SpPlayerConfig` is a container of four sub-configs. Build it directly via its constructor (`SpPlayerConfig(videoConfig = …, overlayConfig = …, analyticsConfig = …, logoConfig = …)`) or use the DSL helpers `video {}`, `overlay {}`, `analytics {}`, `logo {}`.

VideoConfig

Property	Type	Default	Description
autoPlay	Boolean	true	Auto-start playback when ready.
startPositionMs	Long	0L	Initial playback position (ms).
repeatMode	RepeatMode	REPEAT_MODE_ONE	Playback repeat mode. REPEAT_MODE_OFF, REPEAT_MODE_ONE, REPEAT_MODE_ALL
rendererMode	RendererMode	RENDERER_MODE_AUTO	Renderer mode selection. RENDERER_MODE_OFF, RENDERER_MODE_ON, RENDERER_MODE_AUTO
autoRotate	Boolean	true	true = SDK reacts to sensor rotation (rotate to landscape → fullscreen + sibling views hidden). false = the app owns sensor rotation; the SDK still keeps the fullscreen icon/state in sync and manual fullscreen via the button / changeOrientation() keeps working. Ignored on TV.
slateCacheSizeBytes	Long	64 MB (64 * 1024 * 1024)	Max disk cache used to pre-buffer mid-roll slate clips.
debugTextView	TextView?	null	Optional view to display debug info (resolution, bitrate, etc).

OverlayConfig

Property	Type	Default	Description
autoHideMillis	Int	3000	Overlay auto-hide delay (ms).
qualityControl	Boolean	true	Show quality selection button.
fullScreenControl	Boolean	true	Show fullscreen toggle button.
audioControl	Boolean	true	Show audio track selection button.
speedControlOnVod	Boolean	true	Show speed control for VOD (hidden on live by default).
showLiveBadgeOnLive	Boolean	true	Show the "LIVE" badge on live streams.
captionControl	Boolean	false	Show the captions/subtitles button.
seekForwardIncrementMs	Long	15000	Seek-forward increment in milliseconds (≥ 1).
seekBackIncrementMs	Long	5000	Seek-backward increment in milliseconds (≥ 1).
liveEdgeThresholdInSeconds	Long	30	Threshold (seconds) to consider the player at the live edge.
liveTargetOffsetMs	Long?	null	Target live offset from the live edge (ms). null = use the manifest/Media3 default.
backgroundAlpha	Float	0.8f	Alpha transparency for the overlay background (0.0–1.0).

LogoConfig

Property	Type	Default	Description
enable	Boolean?	null	Enable the logo overlay.
logoUrl	String?	null	URL of the logo image.
scaleType	ImageView.Scale...	FIT_CENTER	Scale type for the logo image.
widthPercentage	Float?	null	Logo width as a fraction of player width (0.0–1.0).
heightPercentage	Float?	null	Logo height as a fraction of player height (0.0–1.0).
horizontalBias	Float?	null	Horizontal position: 0 = left, 0.5 = center, 1 = right.
verticalBias	Float?	null	Vertical position: 0 = top, 0.5 = center, 1 = bottom.

AnalyticsConfig

Property	Type	Default	Description
firebaseEnabled	Boolean?	null	Enable Firebase Analytics tracking. null defers to the SDK default (enabled); pass explicit false to disable.
customEnabled	Boolean?	null	Enable custom (Silverpush) analytics tracking. null defers to the SDK default (enabled); pass explicit false to disable.

10. Media Models

LiveMedia

Includes fields for the stream URL, title, pre-roll ad tags, mid-roll ad tags (VAST/VMAP), and fallback mid-roll slates. The URL can be empty if it is fetched from the channel API.

```kotlin
data class LiveMedia(
    val url: String = "",                    // HLS URL (or empty — resolved by
SDK via channel API)
    val title: String? = null,
    val disablePreRoll: Boolean? = null,     // null → use channel/global config
    val preRollAdTag: String? = null,        // VAST / VMAP URL
    val thumbnailUrl: String? = null,
    val disableMidRoll: Boolean? = null,     // null → use channel/global config
    val midRollAdTag: List<String?>? = null, // VAST / VMAP URLs for mid-rolls
    val midrollSlate: List<String?>? = null  // Fallback video URLs if ads fail
)
```

The URL can be empty if it is fetched from the channel API (using accountId + channelId).

VodMedia

Requires a mediaId for tracking and a content url. It also supports optional pre-roll ad tags and thumbnails.

```kotlin
data class VodMedia(
    val mediaId: String,            // required — analytics primary key
    val url: String,                // required — HLS content URL
    val title: String? = null,
    val preRollAdTag: String? = null, // VAST pre-roll, OR a VMAP that declares
pre/mid/post breaks
    val thumbnailUrl: String? = null
)
```

VOD ads are driven entirely by `preRollAdTag`. A plain VAST tag yields a pre-roll; a **VMAP** tag with multiple `<AdBreak>` elements yields **multi-break** pre/mid/post ads. There is no separate VOD mid-roll field.

11. Player Lifecycle

The PlayerManager interface requires integration with the following Activity callbacks:

- **onResume():** Call player.play().
- **onPause():** Call player.pause().
- **onDestroy():** Call player.release().

WARNING: Failure to call release() will cause memory leaks (it tears down the Media3 player, ad loaders, network monitors, polling/SCTE-35 listeners, and analytics scope).

PlayerManager API

Kotlin

```kotlin
interface PlayerManager {
    fun play()
    fun pause()
    fun changeOrientation()
    fun getOrientation(): Int

    fun showController()
    fun hideController()
    fun toggleController()
    fun release()                               // call in onDestroy()

    fun isPlaying(): Boolean
    fun isAdPlaying(): Boolean
    fun getCurrentPosition(): Long
    fun getDuration(): Long                     // C.TIME_UNSET on Live
    fun getBufferedPosition(): Long

    fun seekTo(positionMs: Long)
    fun seekForward(stepMs: Long = 10_000)
    fun seekBackward(stepMs: Long = 10_000)
    fun seekToLive()                            // no-op on VOD
    fun replay()                                // full re-init from start

    fun setPlaybackSpeed(speed: Float)          // VOD only
    fun getPlaybackSpeed(): Float
}
```

12. Event Listeners

The SDK provides PlayerEventListener, AdsEventListener, and ErrorListener for comprehensive monitoring of playback, ad lifecycles, and critical errors. All listeners are optional.

```kotlin
class PlayerListeners(
    val playerEventListener: PlayerEventListener? = null,
    val adsEventListener: AdsEventListener? = null,
    val errorListener: ErrorListener? = null
)
```

PlayerEventListener callbacks: onPlaybackReady(), onPlaybackStarted(), onPlaybackPaused(), onPlaybackCompleted(), onBufferingStarted(), onBufferingEnded(), onIsPlayingChanged(isPlaying), onSeekStarted(positionMs), onSeekCompleted(positionMs), onPositionDiscontinuity(reason), onPlaybackSpeedChanged(speed), onVideoQualityChanged(width, height, bitrate), onAudioTrackChanged(language, label), onFullScreenChanged(isFullScreen), onOverlayVisibilityChanged(isVisible), onAdBreakEnded() (has a default no-op), onPlaybackError(errorMessage).

AdsEventListener callbacks: onAdBreakStarted(info), onAdBreakCompleted(info), onAdLoaded(info), onAdStarted(info), onAdFirstQuartile(info), onAdMidpoint(info), onAdThirdQuartile(info), onAdCompleted(info), onAdSkipped(info), onAdClicked(info), onAdError(error).

ErrorListener callbacks: onPlayerError(error: SdkPlaybackError), onFatalError(error: SdkFatalError).

> **Breaking change in 1.0.3:** PlayerEventListener.onLastBottomReached() has been **removed**. It is now a TV-flavor-only extension — attach it via SpPlayerView.setOnLastBottomReachedListener { ... } (package co.silverpush.listener, src/tv only). Pass null to detach.

13. Listener Model Reference

Detailed models such as AdBreakInfo, AdInfo, SdkAdError, and SdkFatalError are provided to convey specific event data.

Model	Package	Fields
AdInfo	co.silverpush.listener.model.ad	adId, adPosition, totalAdsInBreak, durationMs, isSkippable
AdBreakInfo	co.silverpush.listener.model.ad	breakId, positionMs, totalAds, adType
AdType (enum)	co.silverpush.listener.model.ad	PREROLL, MIDROLL, POSTROLL, SLATE, UNKNOWN
SdkPlaybackError	co.silverpush.listener.model.error	code, message, type (PlaybackErrorType)
SdkAdError	co.silverpush.listener.model.error	code, message, type (AdErrorType), adBreakId, adId, isRecoverable, throwable
SdkFatalError	co.silverpush.listener.model.error	code, message, type (FatalErrorCode), isRetryable, throwable
PlaybackErrorType (enum)	co.silverpush.listener.model.error	NETWORK, SOURCE, DECODER, DRM, UNKNOWN
AdErrorType (enum)	co.silverpush.listener.model.error	LOAD, PLAYBACK, TIMEOUT, VAST, NETWORK, INTERNAL, UNKNOWN
FatalErrorCode (enum)	co.silverpush.listener.model.error	NETWORK, SOURCE, DECODER, DRM, RENDERER, INTERNAL, PLAYER_INIT_FAILED, AD_INIT_FAILED, UNKNOWN
SdkDiscontinuityReason (enum)	co.silverpush.listener.model	AUTO_TRANSITION, SEEK, SEEK_ADJUSTMENT, SKIP, REMOVE, INTERNAL, SILENCE_SKIP, UNKNOWN

14. Fullscreen + Back Press

With autoRotate = true (default) and a sensor-enabled android:screenOrientation (e.g. fullUser), physical rotation drives fullscreen automatically. The fullscreen button / back press applies a temporary orientation lock and hands control back to your Activity's original requestedOrientation once the device is physically rotated to match (YouTube-style). With a fixed orientation (e.g. userPortrait), fullscreen is driven only by the button / back press.

On **Android TV**, the player is always fullscreen; route back-press through changeOrientation() while in fullscreen so the player exits fullscreen first, then finish().

15. ProGuard / R8 Rules

Consumer rules ship inside the AAR; no additional rules are required for normal integration. The public API surface (co.silverpush.**) and the SpPlayerView XML class is kept. If your app reflects/serializes against the SDK's model classes, keep them yourself.

16. Troubleshooting

Solutions are provided for common issues such as Gradle variant-selection failures (add missingDimensionStrategy), Activity restarts on rotation (add configChanges), UnstableApi lint (annotate the call site), and ads not displaying.

17. Changelog (v1.0.3)

- **Native IMA ad UI** — removed the custom ad-UI layer; ads now render with IMA's native UI, including correct D-pad focus handling on TV (per IMA SDK customization guidelines). Fixes skip-button focus on YouTube creatives.
- **Sensor-based orientation / fullscreen** — new VideoConfig.autoRotate; the SDK manages fullscreen transitions and sibling-view hiding, with a YouTube-style hand-back to the app's declared orientation.
- **VOD VMAP multi-break ads** — a VMAP tag in VodMedia.preRollAdTag now yields multiple pre/mid/post ad breaks; fixed multi-break stalls and a placeholder group-count crash.
- **First mid-roll VAST pre-warm** — the first live mid-roll VAST is pre-fetched for a faster, glitch-free ad start.
- **Faster initial load** — ad-tag processing is deferred off the start path and analytics initialise in parallel.
- **TV fixes** — play/pause icon restored on backend-triggered mid-roll resume; replay callback re-registered on every controller rebuild.
- **Analytics** — actionable ad events now reach the tracker; KDoc added across public classes.
- **New config** — OverlayConfig.liveTargetOffsetMs, VideoConfig.slateCacheSizeBytes; new AdType.SLATE.
- **Breaking** — PlayerEventListener.onLastBottomReached() removed; use the TV-only SpPlayerView.setOnLastBottomReachedListener extension instead.

Integration Guide

Android Integration Guide (Mobile/TV)

- **Group / Artifact**: co.silverpush:vidgyor-player-media3
- **Latest Version**: 1.0.3
- **Distribution**: Maven Central
- **Min SDK**: 23 (24+ recommended) · **Compile / Target SDK**: 36 · **Java Version**: 17
- **Flavors**: mobile, tv (choose via missingDimensionStrategy)
- **Public API Package**: co.silverpush
- **PlayerView**: com.silverpush.player.ui.SpPlayerView

1. Add the Dependency

1.1 Repositories

Ensure mavenCentral() and google() are declared in your settings.gradle.kts file.

```Kotlin
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        mavenCentral()
        google()
    }
}
```

1.2 App build.gradle.kts

Update your application module's build configuration to include the SDK and its requirements.

```Kotlin
android {
    compileSdk = 36

    defaultConfig {
        minSdk = 24 // SDK requires >= 23
        targetSdk = 36
```

```
        // REQUIRED — selects the SDK flavor.
        missingDimensionStrategy("device", "mobile") // or "tv"
    }

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_17
        targetCompatibility = JavaVersion.VERSION_17
        isCoreLibraryDesugaringEnabled = true
    }

    kotlin { jvmToolchain(17) }
}

dependencies {
    coreLibraryDesugaring("com.android.tools:desugar_jdk_libs:2.1.5")
    implementation("co.silverpush:vidgyor-player-media3:1.0.3")

    // The SDK transitively pulls Media3, Retrofit/OkHttp, simple-xml, Glide,
    // and Firebase Analytics (declared as `api(...)`). You only need to add
    // these directly if your OWN code uses them.
}
```

Note: Raw `.aar` fallback (not recommended):

2. Manifest Configuration

The SDK manifest permissions are merged automatically. For proper orientation and fullscreen handling, configure your Activity as follows:

```
XML
<activity
    android:name=".PlayerActivity"
    android:configChanges="screenSize|smallestScreenSize|screenLayout
|orientation|keyboardHidden|uiMode"
    android:screenOrientation="fullUser"
    android:resizeableActivity="false" />
```

- `configChanges` is **mandatory** — it keeps Android from recreating the Activity on rotation, so the player (and any in-flight ad) survives the transition.

- `android:screenOrientation="fullUser"` enables sensor-driven fullscreen. Use `userPortrait` if you want fullscreen reachable only via the player's fullscreen button.
- For Android TV, drop `screenOrientation` / `resizeableActivity` and keep `configChanges`.

3. Layout Setup

Add the SpPlayerView to your XML layout. This component bundles all necessary control layouts.

```xml
<com.silverpush.player.ui.SpPlayerView
    android:id="@+id/playerView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:keepScreenOn="true" />
```

4. Initialize the Player

`SpPlayer` is the entry point and returns a `PlayerManager`.

```kotlin
class SpPlayer(
    context: Context,
    accountId: String,
    channelId: String,
    spPlayerView: SpPlayerView,
    spPlayerConfig: SpPlayerConfig = SpPlayerConfig(),
    playerListener: PlayerListeners = PlayerListeners()
)
```

4.1 Live Stream Setup

Example implementation for Live streaming:

```kotlin
class LivePlayerActivity : AppCompatActivity() {

    private lateinit var spPlayer: PlayerManager

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_live_player)
```

```
        val playerView = findViewById<SpPlayerView>(R.id.playerView)

        val media = LiveMedia(
            // url is fetched from the channel API — no need to pass a Live URL
            disablePreRoll = false,
            preRollAdTag = "https://example.com/preroll.xml",
            disableMidRoll = false,
            midRollAdTag = listOf(
                "https://example.com/midroll_ad1.xml",
                "https://example.com/midroll_ad2.xml"
            ),
            midrollSlate = listOf(
                "https://example.com/slate1.mp4",
                "https://example.com/slate2.mp4"
            )
        )

        spPlayer = SpPlayer(
            context = this,
            accountId = "<YOUR_ACCOUNT_ID>",
            channelId = "<YOUR_CHANNEL_ID>",
            spPlayerView = playerView
        ).createLive(media)
    }

    override fun onResume()  { spPlayer.play();    super.onResume() }
    override fun onPause()   { spPlayer.pause();   super.onPause() }
    override fun onDestroy() { spPlayer.release(); super.onDestroy() }
}
```

> **In case of warning: use @UnstableApi** — `SpPlayer` / `SpPlayerConfig` use Media3's `@UnstableApi`. Annotate the call site with `@androidx.media3.common.util.UnstableApi`.

4.2 VOD Setup

Example implementation for Video on Demand (VOD):

Kotlin

```
class VodPlayerActivity : AppCompatActivity() {

    private lateinit var spPlayer: PlayerManager

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
        setContentView(R.layout.activity_vod_player)

        val playerView = findViewById<SpPlayerView>(R.id.playerView)

        val media = VodMedia(
            mediaId = "unique-id-123",
            url = "https://example.com/video.m3u8",
            title = "Sample VOD",
            preRollAdTag = "https://example.com/preroll.xml", // VAST pre-roll, or
a VMAP for multi-break
            thumbnailUrl = "https://example.com/thumbnail.jpg"
        )

        spPlayer = SpPlayer(
            context = this,
            accountId = "<YOUR_ACCOUNT_ID>",
            channelId = "<YOUR_CHANNEL_ID>",
            spPlayerView = playerView
        ).createVod(media)
    }

    override fun onResume()  { spPlayer.play();    super.onResume() }
    override fun onPause()   { spPlayer.pause();   super.onPause() }
    override fun onDestroy() { spPlayer.release(); super.onDestroy() }
}
```

4.3 Configuration example

Kotlin

```
val config = SpPlayerConfig().apply {
    video {
        autoPlay        = true
        startPositionMs = 0L
        repeatMode      = RepeatMode.REPEAT_MODE_ONE
        rendererMode    = RendererMode.RENDERER_MODE_AUTO
        autoRotate      = true
    }
    overlay {
        autoHideMillis    = 3000
        qualityControl    = true
        audioControl      = true
        captionControl    = false
        fullScreenControl = true
    }
```

```
    analytics {
        firebaseEnabled = true
        customEnabled   = true
    }
    logo {
        enable           = true
        logoUrl          = "https://cdn.example.com/logo.png"
        scaleType        = ImageView.ScaleType.CENTER_CROP
        widthPercentage  = 0.15f
        heightPercentage = 0.12f
        horizontalBias   = 0.95f   // 0 = left, 1 = right
        verticalBias     = 0.03f   // 0 = top,  1 = bottom
    }
}

spPlayer = SpPlayer(this, accountId, channelId, playerView, spPlayerConfig =
                        config)
spPlayer.createLive(liveMedia) // For Live
spPlayer.createVod(vodMedia) // For Vod
```

5. Configuration Reference

See the Vidgyor Player Media3 SDK - 9. Configuration Reference for the full property tables (VideoConfig, OverlayConfig, AnalyticsConfig, LogoConfig).

6. Player Lifecycle

Mandatory lifecycle integration is required to prevent memory leaks and manage background services.

Activity Callback	Required Method Call
onResume()	spPlayer.play()
onPause()	spPlayer.pause()
onDestroy()	spPlayer.release()

7. TV: Last-Bottom-Reached Callback

On the TV flavor, attach the bottom-of-UI D-pad-down signal (your cue to move focus to content below the player) via an extension function:

```kotlin
import co.silverpush.listener.setOnLastBottomReachedListener

playerView.setOnLastBottomReachedListener {
    // Move focus to your related-content rail below the player.
}
// Pass null to detach.
```

8. Troubleshooting

Symptom	Cause / Resolution
Gradle: Could not resolve flavor	Add missingDimensionStrategy("device", "mobile" | "tv") to defaultConfig.
Activity restarts on rotation	Add configChanges to the Manifest entry
UnstableApi Lint Error	Annotate with @androidx.media3.common.util.UnstableApi
Mid-roll ads never fire on Live	Confirm the stream emits SCTE-35, or supply a midRollAdTag; check disableMidRoll.
SurfaceView / ArrayIndexOutOfBoundsException on some streams	Handle ErrorListener.onFatalError with !isRetryable by recreating the Activity.

For technical support, contact the integration team.
Documentation Date: Jun 29, 2026
Technical Contact: Parth Desai Rahul Gupta

9. Sample Code for Mobile/TV

Mobile: SampleMobilePlayer_with_v1.0.3.zip

TV: SampleTvPlayer_with_v1.0.3.zip

Support

Support

For technical support or questions about implementation:

- Email: support@vidgyor.com

FAQs

Frequently Asked Questions (FAQs)

General Integration

- **Do I need to enable core library desugaring?**
  Yes — isCoreLibraryDesugaringEnabled = true and desugar_jdk_libs:2.x. Several SDK code paths use java.time APIs that require it on minSdk < 26.

- **What's the entry point?**
  co.silverpush.SpPlayer. Construct it once with context, accountId, channelId, spPlayerView (and optionally spPlayerConfig, playerListener), then call createLive(media) or createVod(media).

- **Can a single SpPlayer instance switch between Live and VOD?**
  No. Each createLive() / createVod() call returns a fresh PlayerManager. To switch modes, release() the current one and create a new one.

- **What does PlayerManager.release() actually do?**
  Releases the Media3 player, ad loaders, network monitors, polling/SCTE-35 listeners, and analytics scope. Skipping it leaks the player and leaves background coroutines/services running.

- **Can I host the player inside a Fragment instead of an Activity?**
  Yes — pass the host Activity as context and the SpPlayerView from the Fragment's view hierarchy. Drive lifecycle from onResume / onPause / onDestroyView.

Live Playback

- **Do I have to provide a stream URL on LiveMedia?**
  No. If LiveMedia.url is empty, the SDK fetches it from the channel API using accountId + channelId. The remote URL takes precedence even when you pass one.

- **How is mid-roll triggered on a live stream?**
  Via **SCTE-35**, **polling**, or **VMAP** — the active mode is controlled by the channel's enableMidrollVia field on the backend, and the SDK auto-selects based on that config.

- **What's the "mid-roll slate"?**
  A fallback video clip (or list of clips) played in place of the ad when the ad fails to load. Pass URLs in LiveMedia.midrollSlate.

- **Can I disable ads programmatically?**

Yes — set LiveMedia.disablePreRoll = true and/or disableMidRoll = true. This overrides the remote channel configuration. Leave them null (default) to defer to the channel config.

VOD Playback

- **Why is mediaId required?**
  It's the analytics primary key. Custom analytics events, Firebase events, and resume-position tracking key off it. Use a stable, content-unique value.

- **Are mid-rolls supported on VOD?**
  Yes, via VMAP. Pass a **VMAP** tag in VodMedia.preRollAdTag — its <AdBreak> elements can declare pre, mid, and post breaks (multi-break is supported as of 1.0.3). A plain VAST tag yields a pre-roll only. There is no separate VOD mid-roll field.

- **Does VOD support resumes from last position?**
  Pass VideoConfig.startPositionMs to start at a specific offset. The SDK doesn't auto-persist position; you store/restore it from the host.

UI / Overlay

- **How do I hide the quality button?**
  OverlayConfig(qualityControl = false). Same pattern for audioControl, captionControl, speedControlOnVod, fullScreenControl.

- **Can I customize overlay colors / branding?**
  Theme overrides are not part of the public API. LogoConfig overlays a channel logo on live content; other branding ships from the AAR.

- **How do I disable auto-rotation?**
  VideoConfig(autoRotate = false). Ignored on TV.

- **What does the last-bottom-reached callback mean?**
  TV-only. Fires when the user presses D-pad-down past the bottom-most overlay control. Attach it via SpPlayerView.setOnLastBottomReachedListener { ... } (it is no longer a PlayerEventListener method as of 1.0.3). Use it to move focus to a "related content" row below the player.

- **How long does the overlay stay visible?**
  OverlayConfig.autoHideMillis (default 3000 ms). Reset on every user interaction.

Ads

- **VAST or VMAP — which should I use for mid-rolls?**
  Either works. For VOD multi-break (pre/mid/post) use a single VMAP in preRollAdTag. For Live, supply VAST/VMAP URLs in midRollAdTag.

- **Why do I see onAdError with type TIMEOUT?**
  The IMA SDK couldn't fetch/parse the VAST tag in time (ad-server latency, malformed XML, blocked URL). The SDK falls back to the slate and resumes content.

- **Are ads skippable by default?**
  Skip behavior is dictated by the VAST tag (<Skipoffset>). The SDK honors it; AdInfo.isSkippable reflects the per-ad value.

- **How can I track ad revenue events?**
  Implement AdsEventListener and forward onAdStarted / onAdCompleted / onAdClicked to your analytics. AdInfo.adId and AdBreakInfo.breakId are stable correlators.

Errors & Recovery

- **What's the difference between onPlayerError and onFatalError?**
  onPlayerError(SdkPlaybackError) is recoverable — the SDK attempts retry/resume internally. onFatalError(SdkFatalError) is terminal — check isRetryable; if false, release and re-create the player/Activity.

- **What happens when the network drops mid-stream?**
  A NetworkMonitor watches connectivity. On loss the SDK pauses, shows a "No Internet" overlay, and auto-resumes when connectivity returns.

- **I get ERROR_CODE_PARSING_MANIFEST_MALFORMED — what now?**
  The HLS manifest is malformed or returns the wrong content-type. Verify the URL; ensure Content-Type: application/vnd.apple.mpegurl.

- **Crash on minified release builds — where's the ProGuard config?**
  Consumer rules ship inside the AAR. If issues remain, keep your own model classes that you reflect/serialize against.

Lifecycle & Threading

- **Can I call play() / pause() from a background thread?**
  No. All PlayerManager methods must be called from the main thread.

- **What happens during a configuration change (rotation)?**
  The SDK handles orientation internally. With the configChanges flags from the Integration Guide §2, you don't need to forward anything.

- **Will background audio playback work?**
  Not currently — the SDK pauses on onPause() by design. A media-session-backed background mode is on the roadmap.

TV Specifics

- **What's the difference between Mobile and TV?**
  Same public API. The TV variant ships a leanback overlay, fixed-fullscreen presentation, D-pad navigation, native IMA ad-UI focus handling, and the setOnLastBottomReachedListener extension.

- **Can I publish a single APK/AAB for phone and TV?**
  Possible but not recommended — the TV AAR's manifest declares leanback as required. Ship two flavors of your app, one per SDK variant.

- **Why doesn't the fullscreen button appear on TV?**
  TV is always fullscreen. OverlayConfig.fullScreenControl is silently ignored on the TV variant.

Analytics

- **What gets tracked?**
  Playback lifecycle events, ad lifecycle events, quality changes, errors, and session metadata. Forwarded to Firebase and Silverpush's tracker endpoint.

- **Can I disable all analytics?**
  AnalyticsConfig(firebaseEnabled = false, customEnabled = false). Pass explicit false to force-disable (the default null defers to the SDK's enabled-by-default behavior).

- **Do I need user consent before analytics fire?**
  The SDK starts analytics on play(). Gate the play() call behind your consent prompt to comply with GDPR / DPDP.