Media: Audio
+ Foreground Service
+ Widget

android-audio-compose
(on GitHub)

Media3 Architecture

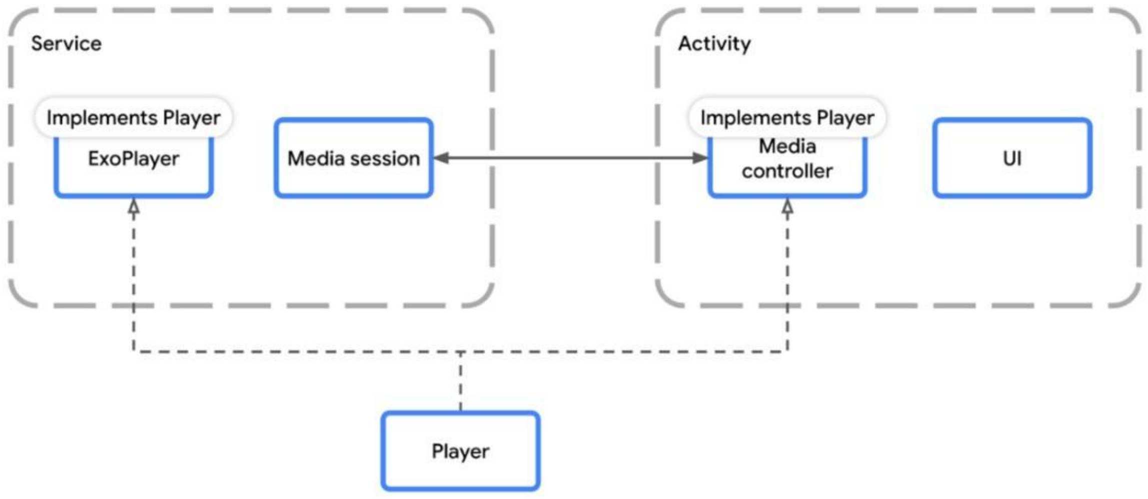
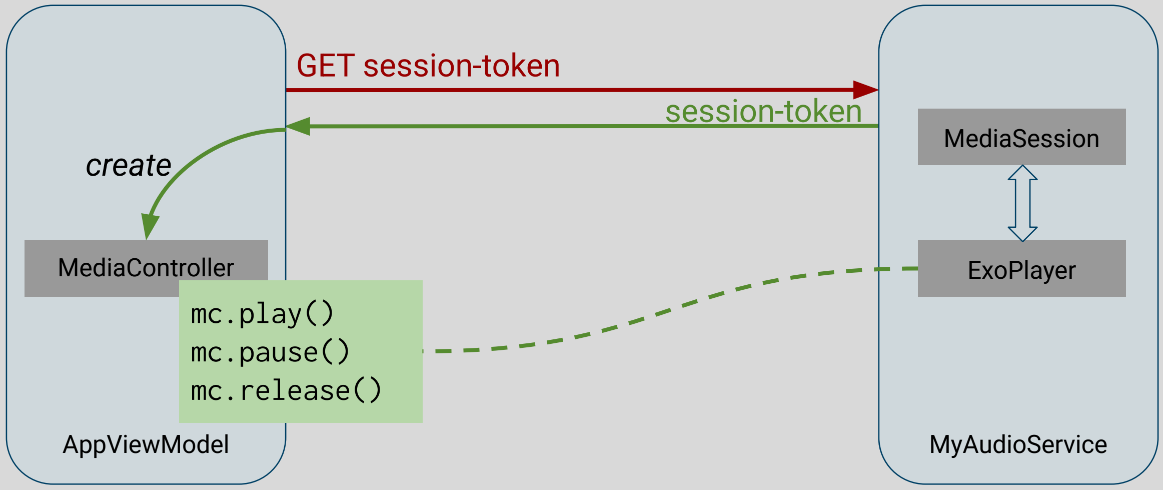


Figure 1: *The* `Player` *interface plays a key role in the architecture of Media3.*

Workflow



```
mc.play()
mc.pause()
mc.release()
```

Step 1: Add Dependencies (Media v3)

```
dependencies {
     // Media3
     implementation("androidx.media3:media3-exoplayer:1.8.0")
     implementation("androidx.media3:media3-common-ktx:1.8.0")
     implementation("androidx.media3:media3-session:1.8.0")
}
```

Step 2: Add Service Class

```
class MyAudioService: MediaSessionService() {
  private var mediaSession: MediaSession? = null

  @UnstableApi
  override fun onCreate() {
    // more details
  }
  override fun onDestroy() {
    // more details
  }

  override fun onGetSession(/*args*/): MediaSession? {
    return mediaSession
  }
}
```

```
override fun onCreate() {
  super.onCreate()
  val player = ExoPlayer.Builder(this)
                .build()
  mediaSession = MediaSession
                  .Builder(this, player)
                  .build()
}
```

```
override fun onDestroy() {
  mediaSession?.player.release()
  mediaSession?.release()
  mediaSession?.mediaSession = null
  super.onDestroy()
}
```

Step 3a: Add Permissions (AndroidManifest.xml)

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

  <uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
  <uses-permission android:name="android.permission.FOREGROUND_SERVICE_MEDIA_PLAYBACK" />
  <application android:allowBackup="true">
    <activity android:name=”.MainActivity”>
      <intent-filter>
        <action android:name="androidx.intent.action.MAIN" />"
        <category android:name="androidx.intent.category.LAUNCHER" />"
      </intent-filter>
    </activity>
  </application>
</manifest>
```

Step 3b: Add Service (AndroidManifest.xml)

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
  <application android:allowBackup="true">
    <activity android:name=”.MainActivity”>
    </activity>
    <service android:name=".MyAudioService"
      android:foregroundServiceType="mediaPlayback" android:exported="true">
      <intent-filter>
        <action android:name="androidx.media3.session.MediaSessionService" />"
      </intent-filter>
    </service>
  </application>
</manifest>
```

Step 4: Create MediaController (in ViewModel)

```
class AppViewModel(val app: Application): AndroidViewMode(app) {
  private var medCtrl: MediaController? = null

  init {
    val context = app.applicationContext
    val token = SessionToken(context, ComponentName(context, MyAudioService::class.java)
    val mcBuilder = MediaController.Builder(context, token).buildAsync()
    mcBuilder.addListener({
      medCtrl = mcBuilder.get()
    }, MoreExecutors.directExecutor())
  }
}
```

Step 5a: Stop/Play MediaPlayer (in ViewModel)

```
class AppViewModel(val app: Application): AndroidViewMode(app) {
  private var medCtrl: MediaController? = null

  fun playAudio() {
    val mediaItem = MediaItem.fromUri(____)
    // medCtrl?.repeatMode = REPEAT_MODE_ONE
    medCtrl?.setMediaItem(mediaItem)
    medCtrl?.prepare()
    medCtrl?.play()
  }

  fun stopAudio() {
    mediaCtrl?.stop()
  }
}
```

Location of Audio Files (.mp3)

- Bundled into the project
  - `app/src/main/res/raw/_____.mp3`
- On the device file system
  - `data/data/`*`your_app_package_name`*`/files/audio/______.mp3`
- On the Internet
  - http://somedomainname/______.mp3

Step 5b: Preparing MediaItem (in ViewModel)

```
// From URL
val mediaFromURL = MediaItem.fromUri("http://podcastapple.com/89iuakhads")

// From File System
val audioFile = File(context.filesDir, "myAudioFiles/test.mp3")
val audioUri = FileProvider.getUriForFile(context,
   context.packageName + ".provider", audioFile)
val mediaFromFile = MediaItem.fromUri(audioUri)

// From Android Resource   app/res/raw/test.mp3
val resUri = Uri.Builder()
         .schema(ContentResolver.SCHEMA_ANDROID_RESOURCE)
         .authority(app.packageName)
         .appendPath(R.raw.test.toString())
         .build()
val mediaFromRes = MediaItem.fromUri(resUri)
```

Adding Waveform Animation (optional)

Understanding the math
(Desmos Graphing Calculator)

Sum of Two Sine Waves

```
fun sinePath(path: Path, size: Size, freq1: Float, freq2: Float) {
   path.moveTo(0f, size.height/2)
   // Use a sum of two sine waves of different frequency
   for (x in 0..size.width.toInt()) {
       val y1 = Math.sin(x * freq1 * Math.PI / size.width).toFloat()
       val y2 = Math.sin(x * freq2 * Math.PI / size.width).toFloat()
       val ySum = (0.4f * y1 + 0.6f * y2) * size.height / 2 + size.height / 2
       path.lineTo(x.toFloat(), ySum)
   }
}
```

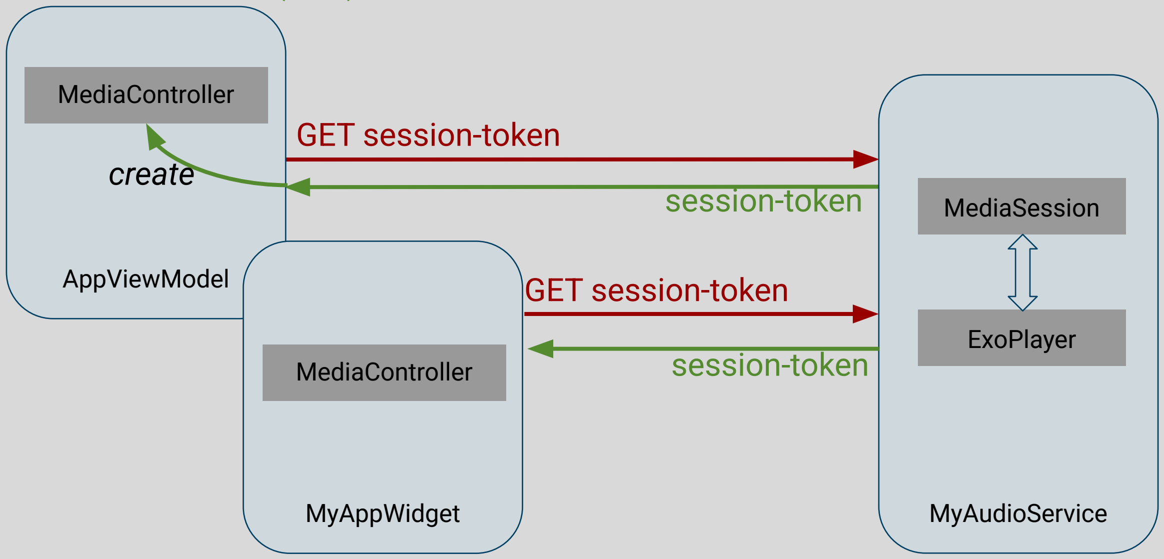
@Composable: Canvas & LaunchedEffect

```
@Composable AudioScreen() {
   LaunchedEffect(isPlayingAudio) {
       launch(Dispatchers.Default) {
           while (isPlayingAudio) {
               freq1 = 20 * Random.nextFloat()
               freq2 = 10 + 30 * Random.nextFloat()
               delay(100)
           }
       }
   }

    Canvas(modifier = Modifier.fillMaxWidth()) {
        val p = Path()
        sinePath(p, size, freq1, freq2)
        drawPath(p, brush = SolidColor(Color.Blue), style = Stroke(width = 4f))
    }
}
```

Adding Widget

Workflow (x2)



Ingredients

Component Needed	Why Needed?	Implementation
BroadcastReceiver	Manage lifecycle of the widget	● `MyAppWidgetReceiver` ● `<receiver>` section in AndroidManifest.xml
MediaController	The second copy of media controller (connected to the **same Foreground service**)	`Inside MyAppWidget`
UI design for the Widget	*self-explanatory*	● `MyAppWidget` ● `res/xml/my_app_widget_info`

Step A: Add Dependencies (Glance Widget)

```
dependencies {
    // Glance Widget
    implementation("androidx.glance:glance-appwidget:1.1.1")

    // For interop APIs with Material 3
    implementation("androidx.glance:glance-material3:1.1.1")
}
```

Step B: AppWidgetReceiver & AppWidget

```
class MyAppWidget : GlanceAppWidget() {
   private var mc: MediaController? = null
   override suspend fun provideGlance(context: Context, id: GlanceId) {
       val sessionToken = SessionToken(context,
           ComponentName(context, MyAudioService::class.java))
       val mcBuilder = MediaController.Builder(context, sessionToken).buildAsync()
       mcBuilder.addListener({
           mc = mcBuilder.get()
       }, MoreExecutors.directExecutor())

       provideContent {
           Row {
               Button("Play", onClick = { mc?.play() })
               Button("Stop", onClick = { mc?.stop() })
           }
       }
   }
}
```

```
class MyAppWidgetReceiver : GlanceAppWidgetReceiver() {
   override val glanceAppWidget: GlanceAppWidget = MyAppWidget()
}
```

Step C: Add Receiver

(AndroidManifest.xml)

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
  <application android:allowBackup="true">
    <activity android:name=”.MainActivity”>
    </activity>
    <receiver android:name=".MyAppWidgetReceiver" android:exported="true">
      <intent-filter>
        <action android:name="android.appwidget.action.APP_WIDGETUPDATE" />"
      </intent-filter>
      <meta-data android:name=”android.appwidget.provider”
                 android:resource=”@xml/my_app_widget_info”>
      </meta-data>
    </service>
  </application>
</manifest>
```

```
// Save this into app/src/main/res/xml/my_app_widget_info.xml
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:initialLayout=”@layout/glance_default_loading_layout”
    android:targetCellWidth=”3” android:targetCellHeight=”1” />
```