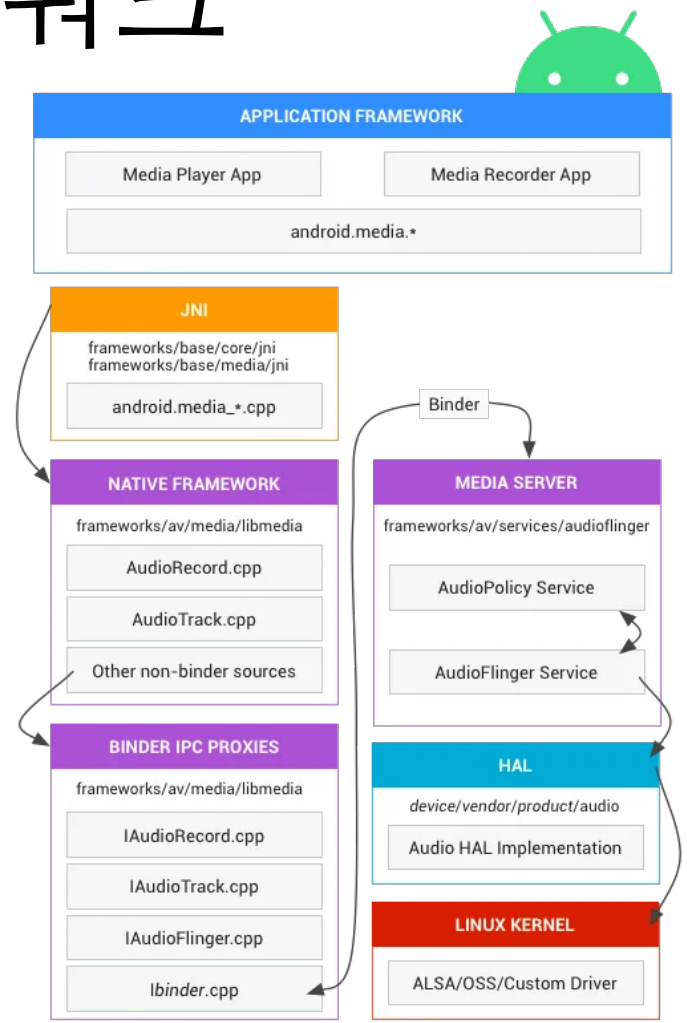
유정현
YOO JEUNGHYUN
Android Engineer

안드로이드 오디오 프레임워크 구조와 동작 원리

Understanding the Architecture and Behavior of Android Audio Framework

목차

오디오 프레임워크



출처: 안드로이드 오디오 (https://source.android.com/docs/core/audio?hl=ko)

Native Layer

	Native Framework	Binder IPC	Media Server
역할	C/C++ 레벨에서 오디오 처리	Native Framework와 Media Server를 연결	안드로이드 미디어 서비스를 관리하는 네이티브 데몬
구성 요소	AudioTrack.cpp, AudioRecord.cpp …		Audio Policy Service, Audio Flinger

Media Server

AudioPolicyService

- 오디오의 우선순위, 볼륨 정책, 라우팅 규칙 등을 결정
- AudioManger의 AudioFocus, AudioAttributes 등으로 정책 결정 가능(Application 레벨)

AudioFlinger

- AudioPolicy에서 결정된 규칙에 따라 오디오를 실제 믹싱/전달하는 작업을 수행

Flow

- 재생(Playback) : 앱 → AudioFlinger → (Policy 확인) → HAL → 스피커/헤드폰
- 녹음(Record) : 마이크/HAL → AudioFlinger → (Policy 확인) → 앱(AudioRecord 등)

부팅 프로세스

1. 로그 및 데몬 초기화
2. HAL 조회와 Sound Trigger 초기화
3. Audio Flinger/ Audio Policy Manager(APM) 구성
4. Codec, Content Provider 및 Device 초기화
    1. APM ↔️ Audio Flinger ↔️ AHAL

AudioRecord

```
val audioFormat = AudioFormat.Builder()
    .setEncoding(AudioFormat.ENCODING_PCM_16BIT)
    .setSampleRate(44100)
    // 입력용 채널은 일반적으로 setChannelMask()를 사용함.
    .setChannelMask(AudioFormat.CHANNEL_IN_MONO)
    .build()
val audioRecord = AudioRecord.Builder()
    .setAudioSource(MediaRecorder.AudioSource.MIC)
    .setAudioFormat(audioFormat)
    .build()
```

- Audio Source
- Audio Format
- Device

Input Device

```
Device 1:
  - id: 15
  - tag name: Built-In Mic
  - type: AUDIO_DEVICE_IN_BUILTIN_MIC
  - supported encapsulation modes: 0
  - supported encapsulation metadata types: 0
  - address: bottom
  - Profiles:
      Profile 0:
          - format: AUDIO_FORMAT_PCM_16_BIT
          - sampling rates: 8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000
          - channel masks: 0x000c, 0x0010, 0x0030
          - encapsulation type: 0
```

AudioTrack

```
// AudioFormat: 오디오 데이터의 형식(인코딩, 샘플링 속도, 채널 구성)을 정의함
val audioFormat = AudioFormat.Builder()
    .setEncoding(AudioFormat.ENCODING_PCM_16BIT) // PCM 16비트 형
식
    .setSampleRate(44100) // 샘플링 속도: 44.1kHz (일반 미디어 재생용)
    .setChannelMask(AudioFormat.CHANNEL_OUT_MONO) // 출력 채널 구
성: 단일 채널 (모노)
    .build()

// 디바이스가 지원하는 최소 버퍼 크기를 계산 (재생 안정성 확보를 위해 필요)
val bufferSize = AudioTrack.getMinBufferSize(
    44100,                                  // 샘플링 속도
    AudioFormat.CHANNEL_OUT_MONO,           // 채널 구성
    AudioFormat.ENCODING_PCM_16BIT          // 인코딩 방식
)

// AudioTrack 생성: 실제 오디오 출력을 담당
val audioTrack = AudioTrack.Builder()
    .setAudioFormat(audioFormat)            // 필수: 오디오 데이터 형식
지정
    .setBufferSizeInBytes(bufferSize)       // 필수: 최소 버퍼 크기 설
정
    .setTransferMode(AudioTrack.MODE_STREAM) // 필수: 스트리밍 모드
(실시간 데이터 전송)
    .build()
```

- Audio Source
- Audio Format
- Buffer Size
- Transfer Mode (STREAM/STATIC)
- Audio Attributes
- Device

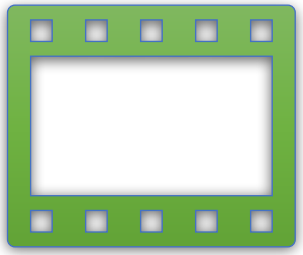
Output Device

```
Device 1:
  - id: 2
  - tag name: Earpiece
  - type: AUDIO_DEVICE_OUT_EARPIECE
  - supported encapsulation modes: 0
  - supported encapsulation metadata types: 0
  - Profiles:
      Profile 0:
          - format: AUDIO_FORMAT_PCM_16_BIT
          - sampling rates: 48000
          - channel masks: 0x0001
          - encapsulation type: 0

Product Strategies dump:
  - STRATEGY_PHONE (id: 14)
    Selected Device: {type:AUDIO_DEVICE_OUT_EARPIECE, @:}
     Group: 13 stream: AUDIO_STREAM_VOICE_CALL
      Attributes: { Content type: AUDIO_CONTENT_TYPE_UNKNOWN Usage: AUDIO_USAGE_VOICE_COMMUNICATION Source
AUDIO_SOURCE_DEFAULT Flags: 0x0 Tags:  }
```

제어

1. 영상 시청



2. 음악 재생

3. 전화 수신

현재는 전화 소리만 재생

Audio Focus

Audio Focus: 재생 중지

첫 번째 앱(현재 재생 중인 앱)의 조건:

- AudioAttributes.USAGE_MEDIA / AudioAttributes.USAGE_GAME
- AUDIOFOCUS_GAIN 방식으로 오디오 포커스 보유
- AudioAttributes.CONTENT_TYPE_SPEECH 아닌 오디오 콘텐츠 재생

두 번째 앱(재생할 앱)의 조건:

- AUDIOFOCUS_GAIN 오디오 포커스 요청

Audio Focus: Ducking (소리 감소)

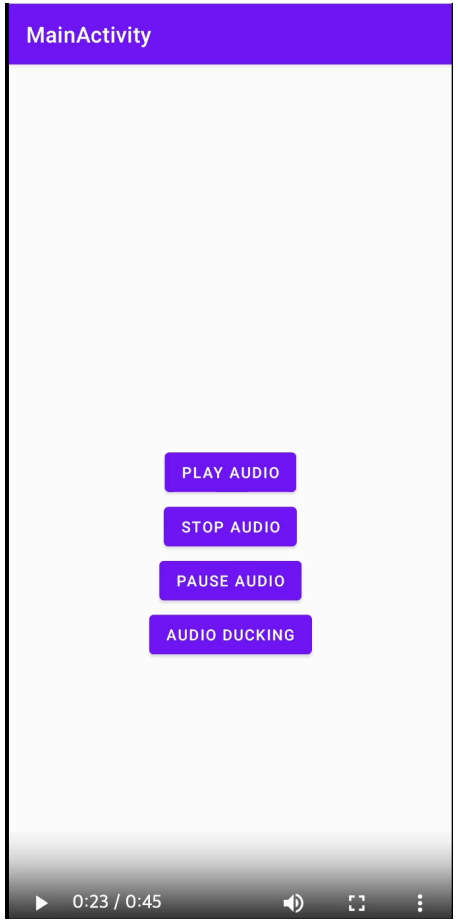
첫 번째 앱(현재 재생 중인 앱)의 조건:

- AudioFocusRequest.Builder.setWillPauseWhenDucked(true)를 설정하지 않음
- 종류와 상관없이 오디오 포커스 보유
- AudioAttributes.CONTENT_TYPE_SPEECH 아닌 오디오 콘텐츠 재생

두 번째 앱(재생할 앱)의 조건:

- AUDIOFOCUS_GAIN_TRANSIENT_MAY_DUCK 오디오 포커스 요청

Audio Focus 적용



Audio Focus 적용

```
findViewById<View>(R.id.btn_play).setOnClickListener { v: View? ->
    val result = audioManager!!.requestAudioFocus(
        AudioFocusRequest.Builder(AudioManager.AUDIOFOCUS_GAIN)
            .setOnAudioFocusChangeListener(focusListener)
            .setAcceptsDelayedFocusGain(true)
            .build()
    )
    if (result == AudioManager.AUDIOFOCUS_REQUEST_GRANTED) {
        mediaPlayer!!.start()
    }
}
```

Focus 요청 후 재생

```
private val focusListener =
    OnAudioFocusChangeListener { focusChange ->
        when (focusChange) {
            AudioManager.AUDIOFOCUS_LOSS -> mediaPlayer!!.pause()
            AudioManager.AUDIOFOCUS_LOSS_TRANSIENT -> mediaPlayer!!.pause()
            AudioManager.AUDIOFOCUS_LOSS_TRANSIENT_CAN_DUCK -> mediaPlayer!!.setVolume(
                0.2f,
                0.2f
            )
            AudioManager.AUDIOFOCUS_GAIN -> {
                mediaPlayer!!.setVolume(1.0f, 1.0f)
                mediaPlayer!!.start()
            }
        }
    }
```

Focus Listener 설정

```
--------------------
requestAudioFocus() from uid/pid 10210/10243 AA=USAGE_MEDIA/CONTENT_TYPE_UNKNOWN clientId=android.media.AudioManager@4dd6945com.example.audioframework.MainActivity$$Extern
AudioFocus Requested
requestAudioFocus() from uid/pid 10167/9920 AA=USAGE_MEDIA/CONTENT_TYPE_UNKNOWN clientId=android.media.AudioManager@ad9fdf7affu@e3c9464 callingPack=com.google.android.yout
AudioFocus Granted
dispatching onAudioFocusChange(-1) to android.media.AudioManager@4dd6945com.example.audioframework.MainActivity$$ExternalSyntheticLambda6@8b7089a
```

고생하셨습니다