什么是 MediaCodec？

MediaCodec 是 Android 提供的一个音视频编解码器，它允许应用程序对音频和视频数据进行编码（压缩）和解码（解压缩）。通过 MediaCodec，我们可以实现音视频的播放、录制、转码等功能。

下面以编码器为例子进行说明，如下图所示，MediaCodec存在两个缓冲区，输入缓冲区（input）和输出缓冲区（output）。编码视频流时即不断从输入缓冲区中读取原始YUV数据，经过Codec编码后，从输出缓冲区获取压缩后的H264数据。

————————————————

3. 处理输入数据

一旦配置完成，我们就可以开始向编码器输入数据或从解码器接收数据了。对于编码器，我们使用 codec.queueInputBuffer() 方法向编码器输入原始数据：

int inputBufferIndex = codec.dequeueInputBuffer(timeout);

if (inputBufferIndex >= 0) {

ByteBuffer inputBuffer = codec.getInputBuffer(inputBufferIndex);

inputBuffer.put(data);

codec.queueInputBuffer(inputBufferIndex, 0, data.length, presentationTimeUs, 0);

}

4. 处理输出数据

最后，需要处理输出的数据。对于编码器，我们使用 codec.dequeueOutputBuffer() 方法获取编码后的数据：

MediaCodec.BufferInfo bufferInfo = new MediaCodec.BufferInfo();

int outputBufferIndex = codec.dequeueOutputBuffer(bufferInfo, timeout);

if (outputBufferIndex >= 0) {

ByteBuffer outputBuffer = codec.getOutputBuffer(outputBufferIndex);

// 处理编码后的数据

codec.releaseOutputBuffer(outputBufferIndex, false);

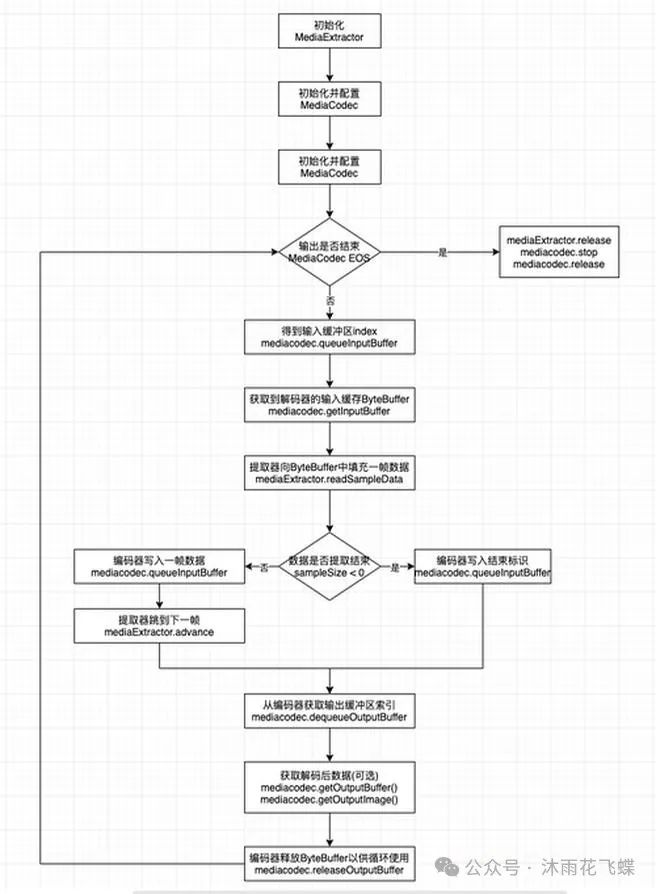
}

5. 销毁编码器

codec.stop();

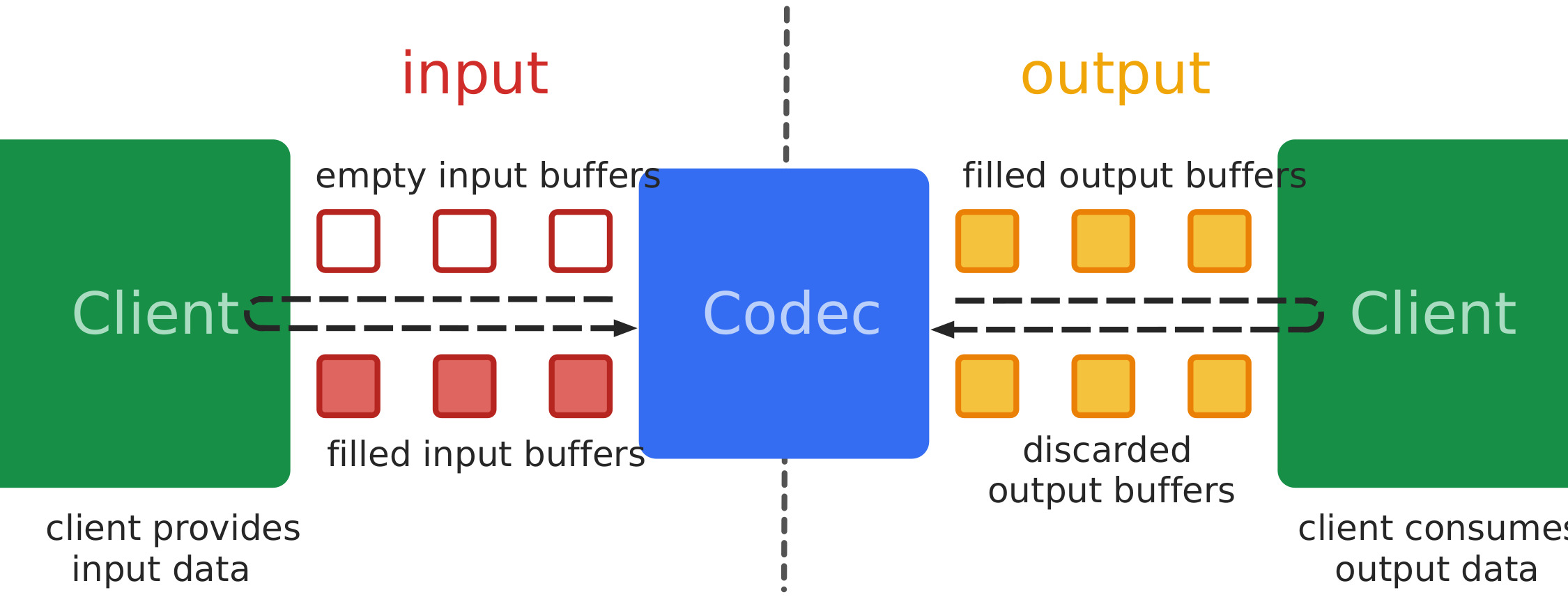
codec.release();

MediaCodec的编解码流程包括创建对象、配置参数、启动、处理输入数据、处理输出数据以及停止和释放资源等步骤。



# **MediaCodec 的数据处理**

调用了 start 方法后，MediaCodec 就开始处理数据了。MediaCodec 的处理流程如下图所示：



MediaCodec 处理输入数据(input data)以生成输出数据(output data)，有同步和异步两种数据处理方式。每个 Codec 都维护了一组 input buffers 和 output buffers，这些 buffer 通过 buffer-ID 标识。简单的讲，我们通过以下步骤使用 MediaCodec：

1.请求(或接收)一个空的输入缓冲区(empty input buffer)

。在同步模式下，调用 MediaCodec.dequeueInputBuffer() 方法从 codec 获取 input buffer

。在异步模式下，我们需要通过 MediaCodec.Callback.onInputBufferAvailable() 回调

方法获取可用的input buffer

// 1. 创建 MediaCodec 的实例

MediaCodec codec = MediaCodec.createByCodecName(name);

MediaFormat mOutputFormat;

// 2. 设置回调，使 MediaCodec 异步处理数据

codec.setCallback(new MediaCodec.Callback() {

@Override

void onInputBufferAvailable(MediaCodec mc, int inputBufferId) {

// 拿到 inputBuffer

ByteBuffer inputBuffer = codec.getInputBuffer(inputBufferId);

// 省略填充数据到 inputBuffer

codec.queueInputBuffer(inputBufferId, …);

}

@Override

void onOutputBufferAvailable(MediaCodec mc, int outputBufferId, …) {

// 拿到 outputBuffer

ByteBuffer outputBuffer = codec.getOutputBuffer(outputBufferId);

mOutputFormat = codec.getOutputFormat(outputBufferId);

// 省略使用 outputBuffer

codec.releaseOutputBuffer(outputBufferId, …);

}

@Override

void onOutputFormatChanged(MediaCodec mc, MediaFormat format) {

// 后续数据将使用新 format

// 使用 getOutputFormat 方法获取了 MediaFormat 后，该回调可省略

mOutputFormat = format;

}

@Override

void onError(…) {

// 错误处理省略

}

@Override

void onCryptoError(…) {

// 解密错误省略

}

});

// 3. 配置 MediaCodec

codec.configure(format, …);

// 获取 MediaFormat

mOutputFormat = codec.getOutputFormat();

// 4. 开始

codec.start();

...

// 5. 停止

codec.stop();

// 6. 释放

codec.release();

2.向 empty input buffer 中填入数据

3.传递带输入数据的输入缓冲区给编解码器(Codec)进行处理

调用 MediaCodec.queueInputBuffer 方法将其提交给Codec；如果需要解密视频，则需要使用 queueSecureInputBuffer。注意除非是 csd 数据，否则不同的 input buffer 不能具有相同的时间戳。

4.编解码器处理数据

5.处理完成后，Codec 将处理结果输出至一个空的输出缓冲区(empty output buffer)

6.请求(或接收)一个已填充结果数据的 output buffer

。在同步模式下，调用 MediaCodec.dequeueOutputBuffer() 方法从 codec 获取 output buffer

。在异步模式下，我们需要通过 MediaCodec.Callback.onOutputBufferAvailable() 回调方法获取可用的 output buffer

7.使用 output buffer

8.使用完缓冲区后，调用 releaseOutputBuffer 方法将 buffer 返回给 codec