Video Encoding API Flow with Code

1. MediaCodec::createEncoderByType("video/avc")

- **Layer**: Application Layer (MediaCodec.java) → Native Framework (CCodec.cpp)
- **Purpose**: Initializes an H.264 (AVC) encoder by specifying the MIME type "video/avc".
- **Details**:
  - This is the entry point in the Java layer to create an encoder instance.
  - Bridges to native code via JNI to configure the encoder.
  - Requires MediaFormat to specify parameters (e.g., width, height, bitrate).
- **Sample Code**:

```
import android.media.MediaCodec;
import android.media.MediaFormat;

MediaFormat format = MediaFormat.createVideoFormat("video/avc", 1280, 720);
format.setInteger(MediaFormat.KEY_BIT_RATE, 2000000);
format.setInteger(MediaFormat.KEY_FRAME_RATE, 30);
format.setInteger(MediaFormat.KEY_I_FRAME_INTERVAL, 1);

MediaCodec codec = MediaCodec.createEncoderByType("video/avc");
codec.configure(format, null, null, MediaCodec.CONFIGURE_FLAG_ENCODE);
```

- **Next Step**: Triggers `CCodec::configure()` in the native layer.

---

2. Codec2Client::createComponent("c2.x264.encoder")

- **Layer**: Native Framework (CCodec.cpp) → CODEC2 Component (CODEC2Client.cpp)
- **Purpose**: Creates a CODEC2 component instance for the H.264 encoder.
- **Details**:
  - Uses the CODEC2 framework to instantiate a component with the name "c2.x264.encoder".
  - Initializes the encoder’s state and allocates resources.
  - Returns a handle to the component for further operations.
- **Sample Code**:

```
#include <codec2/hidl/client.h>
#include <C2Component.h>

std::shared_ptr<C2Component> component;
c2_status_t status = CreateCodec2Component("c2.x264.encoder", &component);
if (status == C2_OK) {
    // Component created successfully
    component->init();
}
```

- **Next Step**: Calls `C2SoftX264Enc::onInit()` to initialize the HAL.

---

3. C2SoftX264Enc::onInit()

- **Layer**: CODEC2 Component (CODEC2Client.cpp) → CODEC2 HAL (C2SoftX264Enc.cpp)
- **Purpose**: Initializes the H.264 encoder in the CODEC2 HAL layer.
- **Details**:
  - Sets up the encoder’s internal state, buffers, and parameters.
  - Prepares the X264 library for encoding operations.
  - Returns success/failure status to the upper layer.
- **Sample Code**:

```
#include "C2SoftX264Enc.h"

c2_status_t C2SoftX264Enc::onInit() {
    x264_param_t param;
    x264_param_default_preset(¶m, "medium", "zerolatency");
    param.i_width = 1280;
    param.i_height = 720;
    param.i_fps_num = 30;
    param.i_fps_den = 1;
    handle_ = x264_encoder_open(¶m);
    return handle_ ? C2_OK : C2_BAD_VALUE;
}
```

- **Next Step**: Encoder is ready to start.

---

4. MediaCodec::start()

- **Layer**: Application Layer (MediaCodec.java) → Native Framework (CCodec.cpp)
- **Purpose**: Starts the encoder to begin processing video frames.
- **Details**:
  - Transitions the encoder from configured to running state.
  - Initiates the native framework’s encoding pipeline.
  - Requires prior configuration and input buffer setup.
- **Sample Code**:

```
MediaCodec codec = MediaCodec.createEncoderByType("video/avc");
// Configure codec (as in Section 1)
codec.start(); // Start the encoder
```

- **Next Step**: Triggers `CCodec::start()` in the native layer.

---

5. Codec2Client::start()

- **Layer**: Native Framework (CCodec.cpp) → CODEC2 Component (CODEC2Client.cpp)
- **Purpose**: Starts the CODEC2 component for encoding operations.
- **Details**:
  - Activates the component to process queued work items.
  - Propagates the start command to the HAL layer.
  - Ensures all resources are ready for encoding.
- **Sample Code**:

```
c2_status_t Codec2Client::start() {
    return component_->start();
}
```

- **Next Step**: Calls `C2SoftX264Enc::onStart()` in the HAL.

---

6. C2SoftX264Enc::onStart()

- **Layer**: CODEC2 Component (CODEC2Client.cpp) → CODEC2 HAL (C2SoftX264Enc.cpp)
- **Purpose**: Starts the H.264 encoding process in the HAL.
- **Details**:
  - Prepares the encoder to accept input buffers.
  - Initializes the encoding loop or thread if applicable.
  - Signals readiness to process data.
- **Sample Code**:

```
c2_status_t C2SoftX264Enc::onStart() {
    isRunning_ = true;
    return C2_OK;
}
```

- **Next Step**: Ready to process input data.

---

7. MediaCodec::queueInputBuffer()

- **Layer**: Application Layer (MediaCodec.java) → Native Framework (CCodec.cpp)
- **Purpose**: Queues raw video frames for encoding.
- **Details**:
  - Passes ByteBuffer containing raw YUV data to the encoder.
  - Requires valid input buffer indices from `dequeueInputBuffer`.
  - Triggers the encoding pipeline in the native layer.
- **Sample Code**:

```
ByteBuffer inputBuffer = codec.getInputBuffer(inputBufferIndex);
inputBuffer.put(rawVideoData);
codec.queueInputBuffer(inputBufferIndex, 0, rawVideoData.length, 0, 0);
```

- **Next Step**: Forwards to `CCodec::queueInputBuffer()`.

---

8. Codec2Client::queueWork()

- **Layer**: Native Framework (CCodec.cpp) → CODEC2 Component (CODEC2Client.cpp)
- **Purpose**: Queues the input buffer as a work item for encoding.
- **Details**:
  - Wraps the input buffer in a CODEC2 work item.
  - Schedules the encoding task in the component.
  - Passes the work to the HAL layer.
- **Sample Code**:

```
c2_status_t Codec2Client::queueWork(const std::shared_ptr<C2Buffer> &buffer) {
    return component_->queue(buffer);
}
```

- **Next Step**: Triggers `C2SoftX264Enc::onEncode()`.

---

9. C2SoftX264Enc::onEncode()

- **Layer**: CODEC2 Component (CODEC2Client.cpp) → CODEC2 HAL (C2SoftX264Enc.cpp)
- **Purpose**: Encodes the input buffer into an H.264 frame.
- **Details**:
  - Converts input buffer to X264-compatible format.
  - Calls the X264 library to perform encoding.
  - Produces an encoded output buffer.
- **Sample Code**:

```
c2_status_t C2SoftX264Enc::onEncode(const std::shared_ptr<C2Buffer> &input) {
    x264_picture_t pic;
    x264_picture_init(&pic);
    pic.img.plane[0] = input->data().data(); // Simplified
    x264_nal_t *nal;
    int nNal;
    x264_encoder_encode(handle_, &nal, &nNal, &pic, nullptr);
    return C2_OK;
}
```

- **Next Step**: Calls `x264_encoder_encode()`.

---

10. x264_encoder_encode()

- **Layer**: CODEC2 HAL (C2SoftX264Enc.cpp) → X264 HAL (x264_encoder.c)
- **Purpose**: Performs the actual H.264 encoding using the X264 library.
- **Details**:
  - Processes the input picture and generates NAL units.
  - Outputs encoded H.264 frames (NAL units).
  - Returns success/failure and encoded data.
- **Sample Code**:

```
int x264_encoder_encode(x264_t *h, x264_nal_t **pp_nal, int *pi_nal,
                        x264_picture_t *pic_in, x264_picture_t *pic_out) {
    return x264_encoder_encode(h, pp_nal, pi_nal, pic_in, pic_out);
}
```

- **Next Step**: Encoded H.264 frame is ready.

---

11. Codec2Client::getOutputWork()

- **Layer**: CODEC2 Component (CODEC2Client.cpp) → Native Framework (CCodec.cpp)
- **Purpose**: Retrieves the encoded H.264 frame from the component.

- **Details**:
  - Polls for completed work items containing encoded data.
  - Returns the output buffer to the native framework.
  - Handles buffer management and synchronization.
- **Sample Code**:

```
c2_status_t Codec2Client::getOutputWork(std::shared_ptr<C2Work> *work) {
    return component_->getWork(work);
}
```

- **Next Step**: Passes to `CCodec::dequeueOutputBuffer()`.

---

12. MediaCodec::dequeueOutputBuffer()

- **Layer**: Application Layer (MediaCodec.java) ← Native Framework (CCodec.cpp)
- **Purpose**: Retrieves the encoded H.264 frame for the application.
- **Details**:
  - Polls for available output buffers with encoded data.
  - Returns buffer index and metadata (e.g., flags, timestamp).
  - App can process or render the encoded frame.
- **Sample Code**:

```
MediaCodec.BufferInfo info = new MediaCodec.BufferInfo();
int outputBufferIndex = codec.dequeueOutputBuffer(info, 0);
ByteBuffer outputBuffer = codec.getOutputBuffer(outputBufferIndex);
byte[] encodedData = new byte[info.size];
outputBuffer.get(encodedData);
```

- **Next Step**: App uses the encoded output.

---

13. Summary

- **Flow Overview**:
  - Application (MediaCodec.java) → Native Framework (CCodec.cpp) → CODEC2 Component (CODEC2Client.cpp) → CODEC2 HAL (C2SoftX264Enc.cpp) → X264 HAL (x264_encoder.c).
- **Key APIs with Code**: Demonstrated initialization, start, queue, encode, and dequeue operations with sample implementations.
- **Result**: Encodes raw video into H.264 format efficiently. **End of Document**