# Livepeer Process

If you need support a good place to go is Livepeer Discord ([here](https://discord.gg/uaPhtyrWsF)), under the *#studio-support* channel

* Let’s build this against our Serraform Livepeer tenant:
  + Livepeer username: [streaming@livepeer.com](mailto:streaming@livepeer.com)
  + Livepeer password: *(sent in Signal)*

Asset = VOD uploaded asset, recorded live stream asset  
Streams = Live stream  
Sessions = Each instance of a live stream (*session* is a child of *stream*)

**VOD**

* 1. Upload an asset ([here](https://docs.livepeer.studio/reference/api/create-asset))
     1. Create a new Direct Upload URL
     2. Upload the file (direct upload)
        1. **NOTE:** Livepeer only support mp4 files encoded with H264 video and AAC audio at the moment
  2. Delete an asset ([here](https://docs.livepeer.studio/reference/api/delete-asset))
     1. **Verify:** does deleting an asset after creating a stream, break the stream? In other words, can we clean up the asset after we’ve made a stream from it?
  3. Get an asset ([here](https://docs.livepeer.studio/reference/api/get-assets))
     1. This will be used to get the asset information (if needed).
  4. Get tasks ([here](https://docs.livepeer.studio/reference/api/get-tasks))
     1. This may be useful for our API to query the status of tasks like stream encoding, so that when a stream is being prepared, we have the ability in the UI to show the user that it’s not completed yet.
  5. Create a stream ([here](https://docs.livepeer.studio/reference/api/create-stream))
     1. We need to set the profiles as follows:

🡪Prabh to verify quality is good enough once we can test it

"profiles": [

{

"name": "720p",

"bitrate": 2000,

"fps": 30,

"width": 1280,

"height": 720

},

{

"name": "480p",

"bitrate": 1750,

"fps": 30,

"width": 854,

"height": 480

},

{

"name": "360p",

"bitrate": 1500,

"fps": 30,

"width": 640,

"height": 360

}

**LIVE**

* 1. Get a stream ([here](https://docs.livepeer.studio/reference/api/get-stream))
     1. This will be used to get the stream information (if needed). We may just want to keep the stream information in table storage so that our API can retrieve it from there for users to access?
  2. Get streams ([here](https://docs.livepeer.studio/reference/api/get-streams))
     1. Not sure how or if we will need this.
  3. Delete a stream ([here](https://docs.livepeer.studio/reference/api/delete-stream))
     1. We will delete a stream as part of the cleanup job. Any stream that is suspended for 7days AND zero funds deposited in users account gets deleted.
  4. Suspend a stream ([here](https://docs.livepeer.studio/reference/api/update-stream#suspend-stream))
     1. We suspend a stream when the users funds are exhausted from their account (we do not delete until 7 days later – see “Delete a stream” for more info)
  5. Get sessions ([here](https://docs.livepeer.studio/reference/api/get-session) and [here](https://docs.livepeer.studio/reference/api/get-sessions))
     1. Not sure what a “session” is or whether we need this. Justen has reached out to support to understand what this is.
  6. Toggle recording ([here](https://docs.livepeer.studio/reference/api/record-on-off))
     1. Each live stream should have recording on IF the user selects to do so.
  7. Get recorded session ([here](https://docs.livepeer.studio/reference/api/get-recorded-sessions))

Examples ([here](https://docs.livepeer.studio/reference/examples) and [here](https://github.com/livepeer/studio-sample-app))