



Make Live Easy™

About id3as

- 100s of thousands of live events a year
- 1000s of live 24/7 channels
- Early access to technology (LL-HLS, WebRTC...)
- Rare skills (deep media expertise, cloud, PureScript, Erlang, Rust...)
- Good on a Bad Day
- Founders of Greening of Streaming



About Our Deployments

- Cloud native
 - “Infinite” capacity
 - Great for spiky load
- ...but about half of our customers run on-prem or hybrid
 - Source acquisition
 - Density
 - Cost
 - Energy

We Are Going to Need a Bigger Boat...



- 256 physical cores (512 logical)
- 24x 2.5" U.2 hot-swappable bays
- 2U in height
- Good excuse for a new screen!

Why Norsk?

Off-the-shelf products can be limiting.

Building your own media technology stack is
time-consuming and complex.

There has to be a better way!

Norsk: Live Media Manipulation

Made Easy!



Norsk: Picture in Picture

```
const input1 = await norsk.input.srt(srtSettings);
const input2 = await norsk.input.rtmpServer(rtmpSettings);
const input3 = await norsk.input.imageFile(fileInputSettings);
const compose = await norsk.processor.transform.composeOverlay(composeSettings);
const mixer = await norsk.processor.transform.audioMix(mixerSettings);
const output = await norsk.output.whep rtcSettings;

compose.subscribeToPins([
  { source: input1, sourceSelector: videoToPin(background.pin) },
  { source: input2, sourceSelector: videoToPin(embedded.pin) },
  { source: input3, sourceSelector: videoToPin(logo.pin) },
]);
mixer.subscribeToPins([
  { source: input1, sourceSelector: audioToPin('input1') },
  { source: input2, sourceSelector: audioToPin('input2') }
]);
output.subscribe([
  { source: compose, sourceSelector: selectVideo },
  { source: input1, sourceSelector: selectAudio },
]);
```



Norsk: RTMP => WebRTC

```
const norsk = await Norsk.connect();

const input  = await norsk.input.rtmpServer();
const output = await norsk.output.whep();

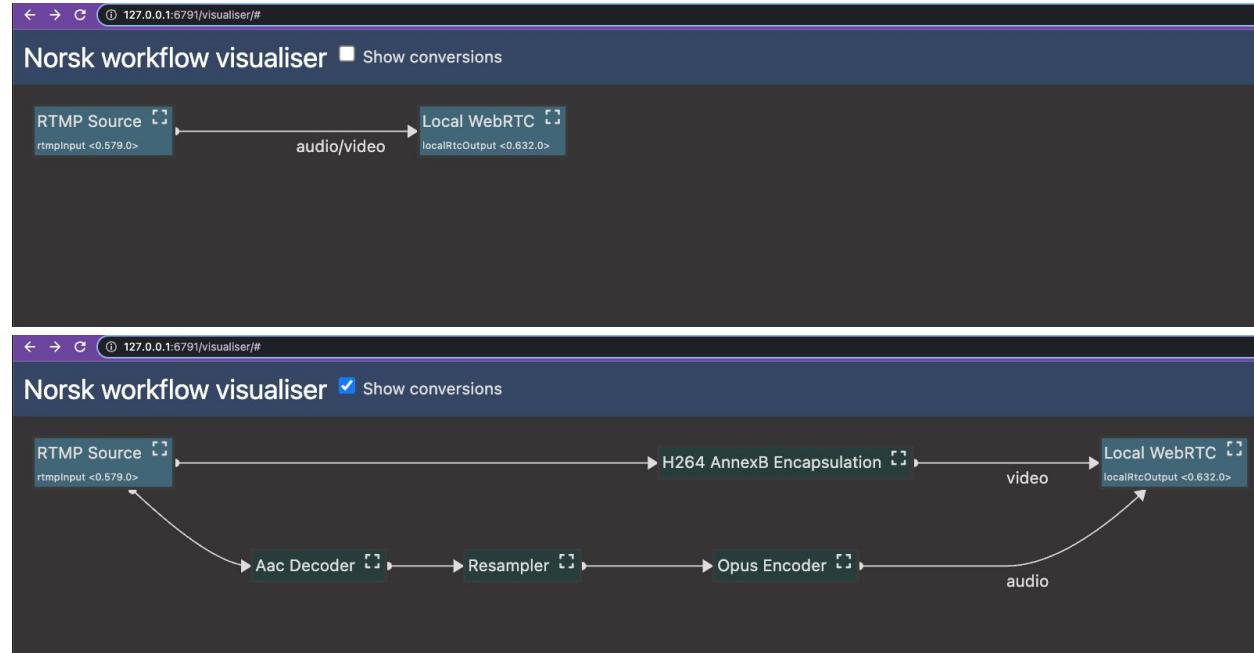
output.subscribe([{ source: input, sourceSelector: selectAV }]);
```



Norsk: Taking Care of the Detail

Norsk is expert in
the intricate
details of media

So you don't
have to be!



Norsk: Source Switcher

```
const norsk = await Norsk.connect();

const camera1      = await norsk.input.srt(srtCamera1Settings);
const camera2      = await norsk.input.srt(srtCamera2Settings);

const sourceSwitcher = await norsk.processor.control.sourceSwitcher(sourceSwitcherSettings);

const output        = await norsk.output.whep(rtcOutputSettings);

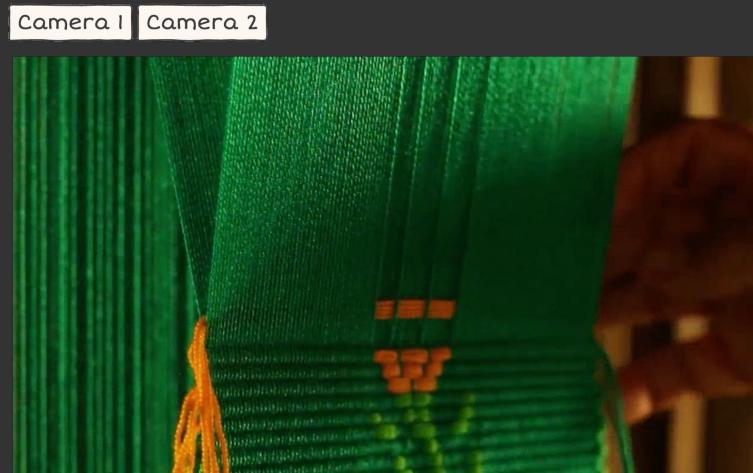
sourceSwitcher.subscribe([
  { source: camera1, sourceSelector: selectAV },
  { source: camera2, sourceSelector: selectAV },
]);
output.subscribe([
  { source: sourceSwitcher, sourceSelector: selectAV }
]);
```



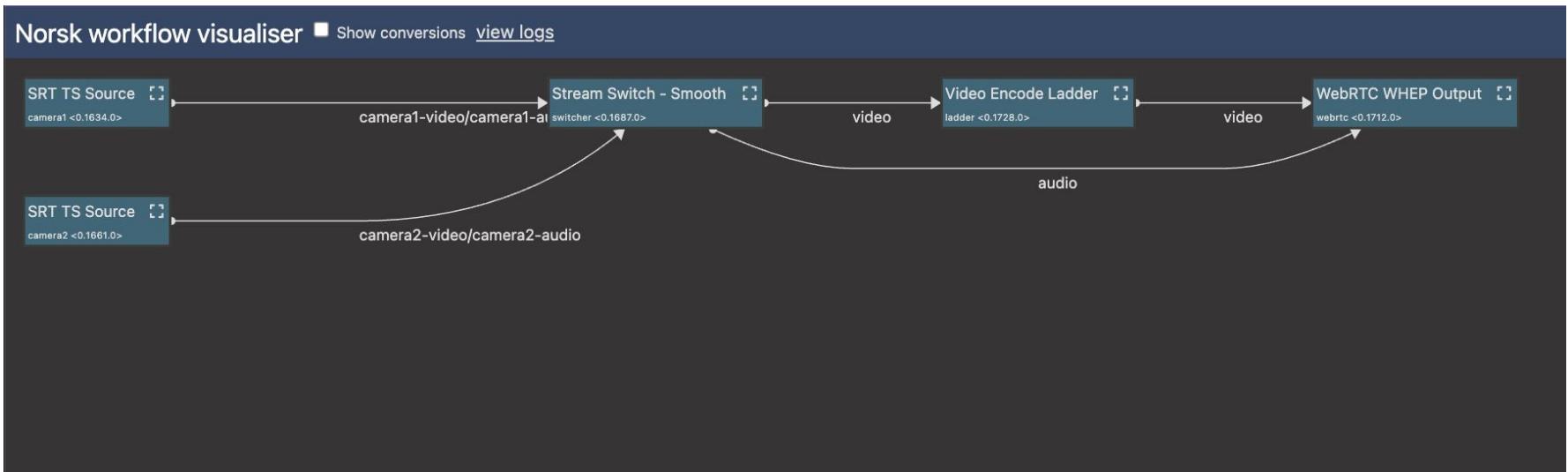
Norsk: Source Switcher

Then just call:

```
sourceSwitcher.switchSource(newSource);
```



Norsk: Taking Care of the Detail

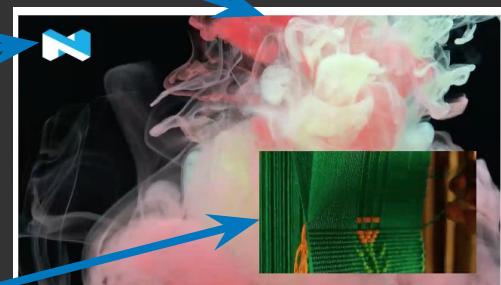


Norsk: Picture in Picture

```
const background: ComposePart<'background'> = {  
  pin: 'background', opacity: 1.0, zIndex: 0,  
  sourceRect: { x: 0, y: 0, width: 100, height: 100 },  
  destRect: { x: 0, y: 0, width: 100, height: 100 },  
};
```

```
const logo: ComposePart<'logo'> = {  
  pin: 'logo', opacity: 1.0, zIndex: 2,  
  sourceRect: { x: 0, y: 0, width: 100, height: 100 },  
  destRect: { x: 5, y: 5, width: 10, height: 8 },  
};
```

```
const embedded: ComposePart<'embedded'> = {  
  pin: 'embedded', opacity: 1.0, zIndex: 1,  
  sourceRect: { x: 0, y: 0, width: 100, height: 100 },  
  destRect: { x: 50, y: 50, width: 45, height: 45 },  
};
```



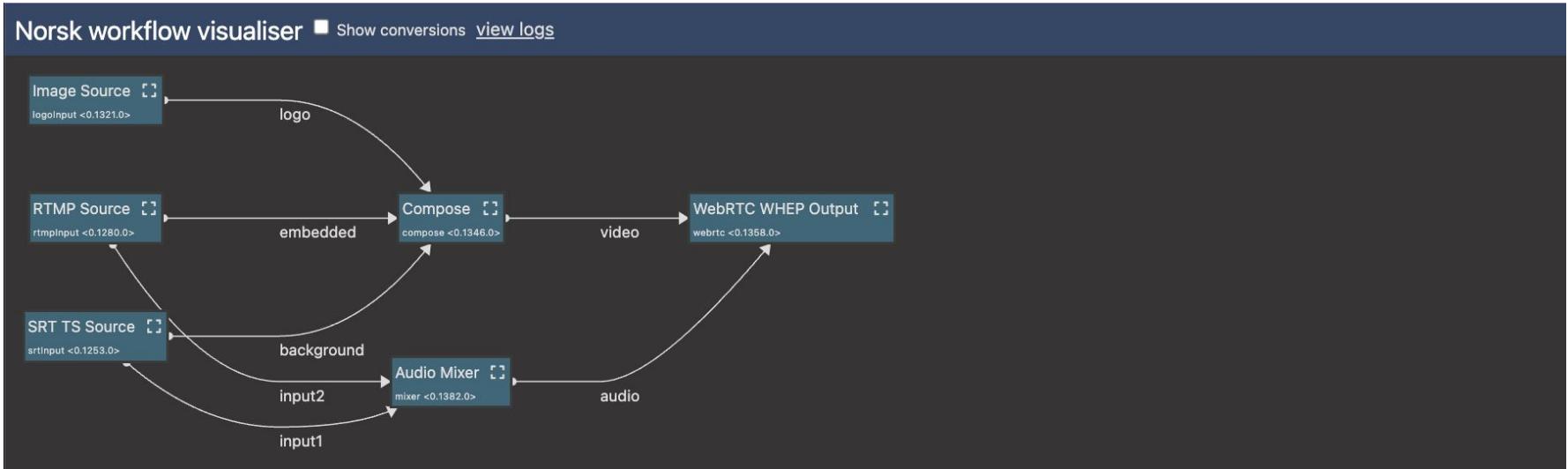
Norsk: Picture in Picture

```
const input1 = await norsk.input.srt(srtSettings);
const input2 = await norsk.input.rtmpServer(rtmpSettings);
const input3 = await norsk.input.imageFile(fileInputSettings);
const compose = await norsk.processor.transform.composeOverlay(composeSettings);
const mixer = await norsk.processor.transform.audioMix(mixerSettings);
const output = await norsk.output.whep rtcSettings;

compose.subscribeToPins([
  { source: input1, sourceSelector: videoToPin(background.pin) },
  { source: input2, sourceSelector: videoToPin(embedded.pin) },
  { source: input3, sourceSelector: videoToPin(logo.pin) },
]);
mixer.subscribeToPins([
  { source: input1, sourceSelector: audioToPin('input1') },
  { source: input2, sourceSelector: audioToPin('input2') }
]);
output.subscribe([
  { source: compose, sourceSelector: selectVideo },
  { source: input1, sourceSelector: selectAudio }
]);
```



Norsk: Taking Care of the Detail



Norsk: Picture in Picture (Dynamic)

```
compose.updateConfig(newComposeSettings);
```



Norsk: Browser Overlay

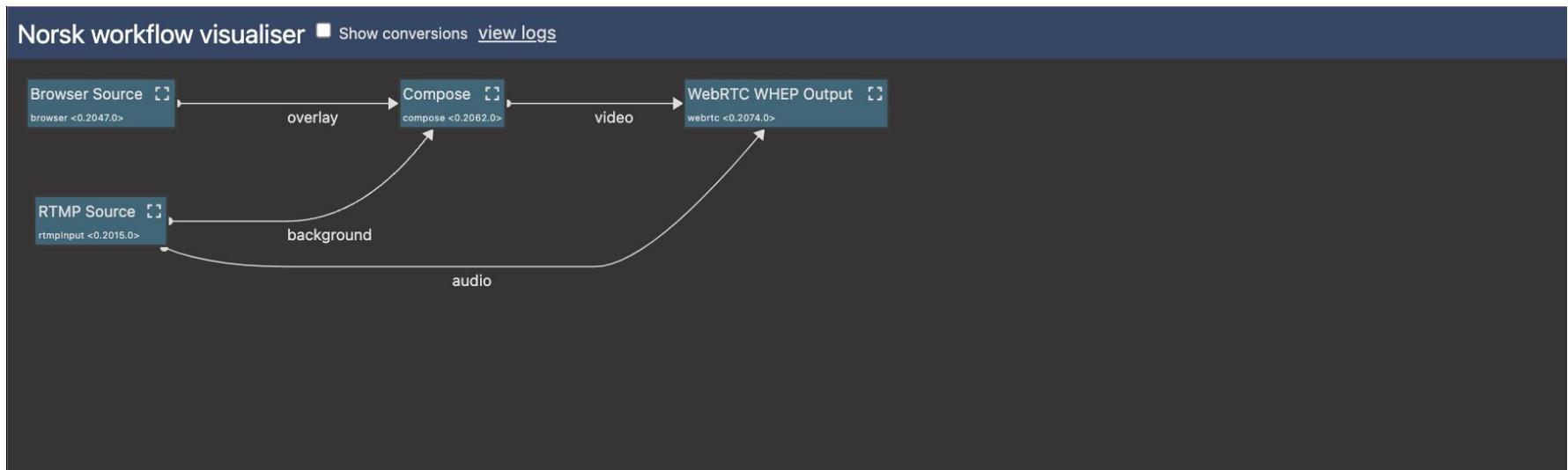
```
const source  = await norsk.input.rtmpServer(rtmpSettings);
const browser = await norsk.input.browser(browserSettings);
const compose = await norsk.processor.transform.videoCompose(composeSettings);
const output   = await norsk.output.whep(webrtcSettings);

compose.subscribeToPins([
  { source: source,  sourceSelector: videoToPin(background.pin) },
  { source: browser, sourceSelector: videoToPin(overlay.pin) },
]);

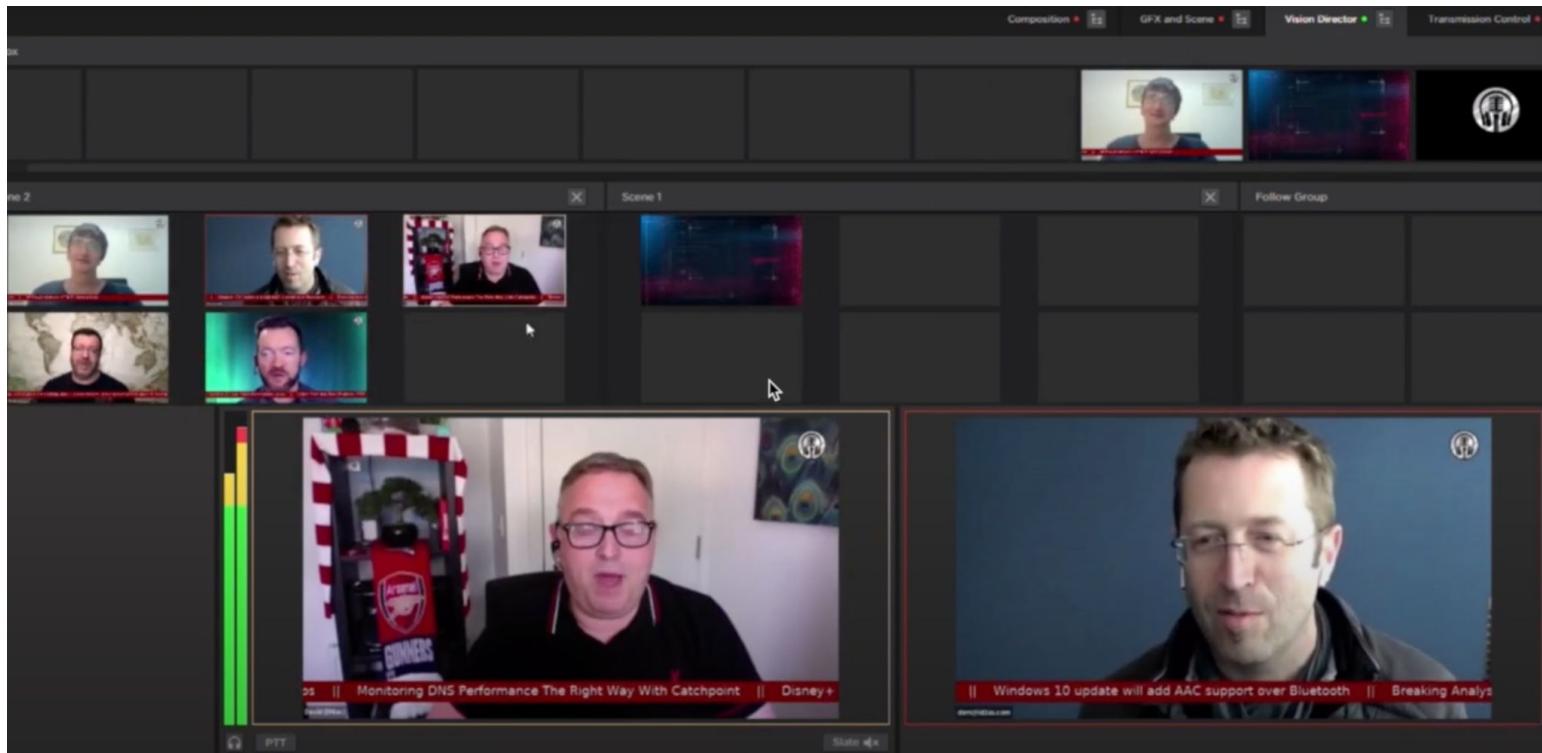
output.subscribe([
  { source: compose, sourceSelector: selectVideo },
  { source: source,  sourceSelector: selectAudio },
]);
```



Norsk: Taking Care of the Detail



That's everything you need to build...



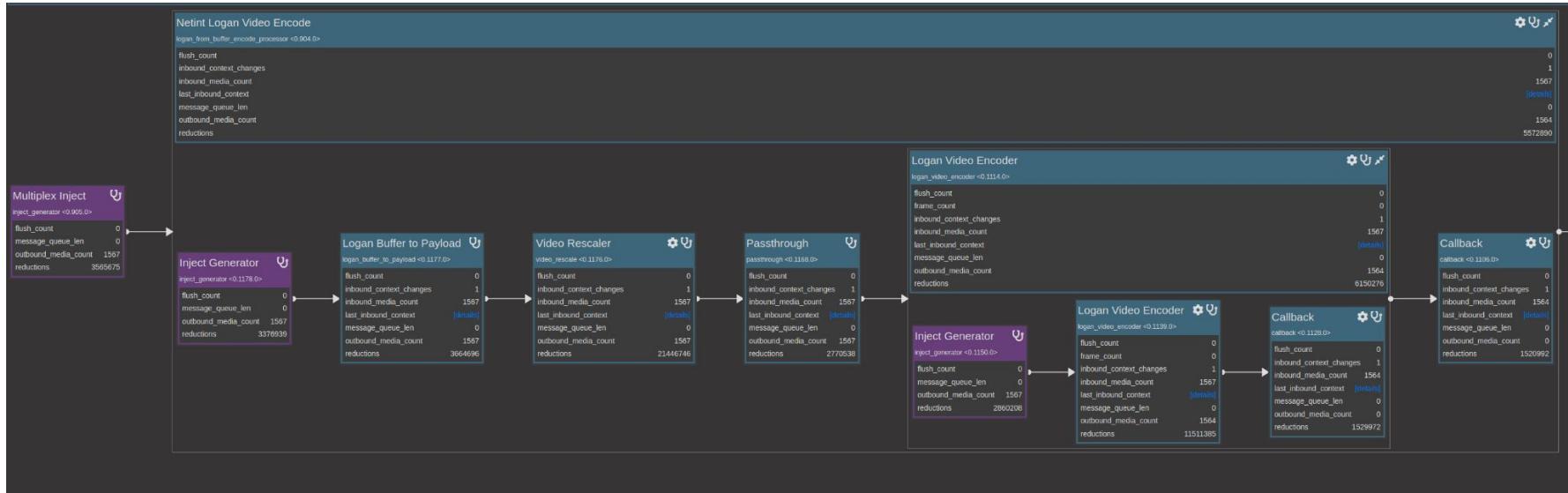
Norsk: Control and Flexibility

```
const highRung: VideoEncodeRung =  
  if (process.env.ENVIRONMENT !== "DEV") return {  
    name: "high",  
    width: 1920,  
    height: 1080,  
    frameRate: { frames: 60, seconds: 1 },  
    codec: {  
      type: "quadra-h264",  
      profile: "high",  
      level: 5.1,  
      bitrate: 2_800_000,  
      intraPeriod: 50,  
      rcEnable: 1,  
      intraQp: 30,  
    }  
  };
```

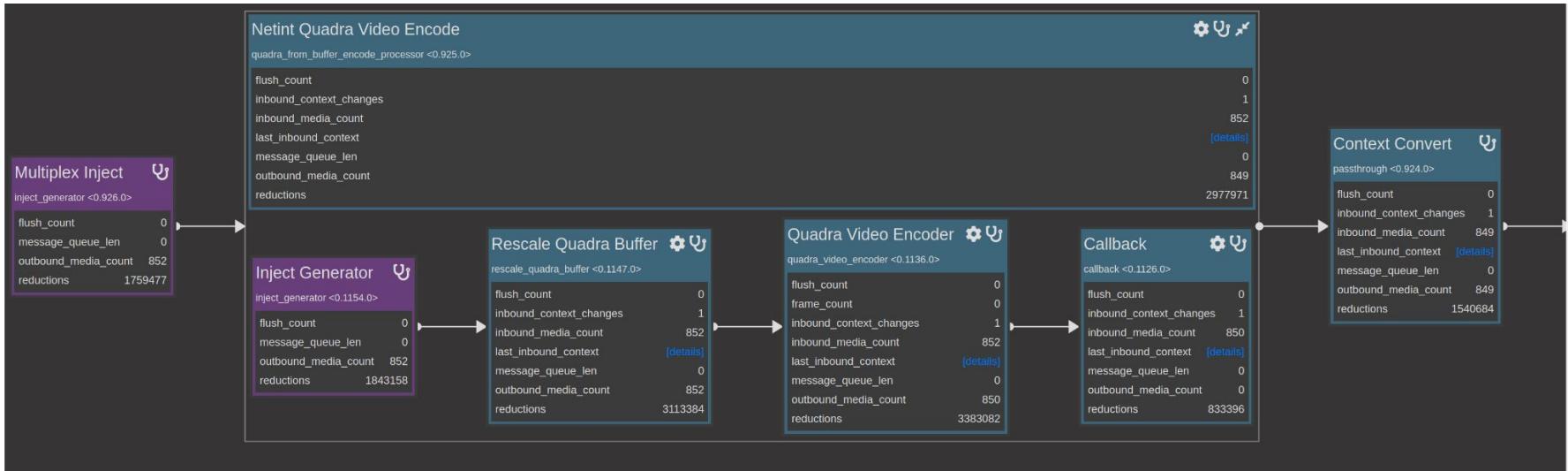
```
else return {  
  name: "high",  
  width: 1280,  
  height: 720,  
  frameRate: { frames: 30, seconds: 1 },  
  codec: {  
    type: "x264",  
    profile: "high",  
    level: 4.1,  
    bitrateMode: { value: 800_000, mode: "abr" },  
    keyFrameIntervalMax: 50,  
    keyFrameIntervalMin: 50,  
    sceneCut: 0,  
    preset: "ultrafast",  
    tune: "zerolatency",  
  }  
};
```



Norsk: Automated Efficiency & Flexibility



Norsk: Automated Efficiency & Flexibility



Norsk: Make Live Practical

The image is a collage of several screenshots illustrating the Norsk Media Typescript SDK:

- Developer Documentation:** A screenshot of the "Norsk Media Typescript SDK" developer documentation showing the "Input" section. It includes a tree view of methods like "audioSignal", "browser", "deckLink", etc., and a detailed view of the "NorskInput" class with its signature and parameters.
- Monitoring Dashboard:** A Grafana dashboard titled "abc-demo" showing system metrics over a 1.2 week period. It includes sections for Uptime, CPU usage (with a graph from 10:39 to 10:56), and Load averages (with graphs for abc-demo-rece, abc-demo-user, and abc-demo-system).
- Processing Flow Diagram:** A complex flowchart titled "NorskInput" showing the data pipeline. It starts with "Adjust Teletext Timestamp" and "Source Gap Filter", followed by "Drop PCR", "Gap Notifier", "Timestamp Reporter", "AVP Transcode Engine", "Missing Signal Filter", "AVP Video Transcode", "Audio Transcode", and ends with "Output". The flowchart includes numerous status boxes and a preview of multiple video streams on the right.
- Video Preview:** On the right side of the flowchart, there are nine preview windows showing different video frames and metadata. Examples include "511: h264 1080 25.00 fps", "510: 25.00 fps AC3", "510: deu ac3", "570: mpegvideo 576i 25.00 fps", "570: ZDF", "570: hallo", "570: 25.00 fps", "570: 25.00 fps", and "570: 25.00 fps".



Norsk: Make Live Easy™

