# Signals & Processing

We're seeing an increase in the number of hardware units that are configurable over a custom API, and we need to develop the tools to manage them.

One of these hardware units is a device used for realtime streaming that we need to configure via its API - Mainly used to monitor our channels remotely.

We have added a few imaginary features to the units described here, thus we won't be able to do a real-world test of the solution.

There are between 10 and 35 units online at any given time, and we need to be able to monitor them all.

## Requirements

Write a tool that can be used to configure a hardware unit over a custom API. The tool should be able to:

- Read the current state of the hardware unit and transmit it to a remote monitoring system. (Datadog, Grafana, etc.)
  - See **Appendix A** for the API call and response documentation.
  - The remote server can be Datadog, Grafana, or any other monitoring system you prefer.
- Read the current configuration of the hardware unit.
  - See **Appendix B** for the API call and response documentation.

Note; The API in this assigment is designed around the *Blackmagic Webpresenter HD*.

## Deliverables

You get to choose how to implement this, use the tool you're most comfortable with, or you think would fit this case the best. There are no restrictions on the language or framework you can use. We prefer the tool to be able to run on Linux, but if you want to use a different OS, that's fine too.

What do we want:

- Source files (Zipped or single file).
- Documentation (Readme, or similar).
- Optional configuration files.
- Optional tests or test scripts.

## Evaluation

We will mostly look at the percieved quality of the code. How does it work, and how is it structured? Feel free to use AI tools to help you with the code, but please make sure to understand what it does.

# Stretch goals

If you find that you have the time to implement some of the stretch goals, you are free to do so. These are not requirements for the solution, as we just want to see how you solve the main problem. You can expect that we will ask you about the stretch goals in the interview, there are bonus questions attached to each goal to help you prepare.

In no particular order - Solve as many as you want.

## → Updating the configuration

Our users want a better looking stream and have decided that we must increase the resolution and framerate. They specifically want to transmit in 1080p and 50fps.

**Appendix B** contains the API call and response documentation, see the section about updating the configuration.

*Q: Are there other things we can do, to increase the quality for the user?*

## → Devices are not always online

Some of the web presenters are in the back of our OB vans, and are not always online. Your tool needs to handle the case where the device is not online, without it being a critical error.

*Q: There are also certain units that must always be online. How would you handle this?*

## → Alerting

We want to know if a device is suddenly interrupted, or if the buffer is getting too full (<90%). The alert can be sent directly to a messaging platform of your choice, Slack, Teams, etc.

*Q: How would you make this monitor run in intervals?*

## → Starting and stopping a stream

After updating the device, the API has changed slightly, and we now need to stop a stream before updating, and start it manually again afterwards.

See **Appendix C** for the API call and response documentation.

*Q: Not all units might have been updated yet, how would you handle this?*

## → Security

We have been required to add authentication calls to the solution, The vendor has supplied commands, which should be translated into your solution.

See **Appendix D** for the API call and response documentation.

*Q: How would you store the password and username in a secure way?*

# Appendix

Here you will find the available documentation for the Blackmagic Webpresenter HD API.

## Appendix A

Getting the current state of the hardware unit Reading the configuration of the hardware unit is done by sending a GET request to the following URL:

```
curl -H "Accept:application/json" -H "Content-Type:application/json" \
-X GET https://IP-ADDRESS/control/api/v1/livestreams/0
```

- This will return the current status of the hardware unit in JSON.
- The expected status code is `200 OK`.

And the response looks like this

```
{
  "status": "Idle",
  "bitrate": 7500000,
  "effectiveVideoFormat": "1280x720p25",
  "duration": 1234,
  "buffer_pct": 7
}
```

The Status response might be one of the following: - Idle - Connecting - Streaming - Interrupted

## Appendix B

Configuration

Updating the configuration of the hardware unit is done by sending a GET request to the following URL:

```
curl -H "Accept:application/json" -H "Content-Type:application/json" \
-X GET https://IP-ADDRESS/control/api/v1/livestreams/customPlatforms/Custom.json
```

- This will return the current configuration of the hardware unit in JSON.
- The expected status code is `200 OK`.

And the response looks like this:

```
{
  "server": "srt://10.12.0.3:40052",
  "audio_bitrate": 128000,
  "video_bitrate": 7500000,
  "resolution": "720p",
  "fps": 25,
  "codec": "H264"
}
```

#### Updating the configuration

To update the configuration, you have to make a PUT request with the body specified above, to the following URL:

```
curl -H "Accept:application/json" -H "Content-Type:application/json" \
-X PUT https://IP-ADDRESS/control/api/v1/livestreams/customPlatforms/Custom.json -d '{
  "server": "srt://10.12.0.3:40052",
  "audio_bitrate": 128000,
  "video_bitrate": 7500000,
  "resolution": "720p",
  "fps": 25,
  "codec": "H264"
}'
```

- This will return an empty response.
- The expected status code is `204 NO CONTENT`.

## Appendix C

Stopping and Starting a Stream

To **stop** a stream, send a PUT request to the following endpoint:

```
curl -H "Accept:application/json" -H "Content-Type:application/json" \
-X PUT https://IP-ADDRESS/control/api/v1/livestreams/0/stop
```

- This will stop the currently running stream.
- The expected response status code is `204 No Content`.
- Stopping an already stopped stream will return a `409 Conflict` status code.

To **start** a stream, send a PUT request to the following endpoint:

```
curl -H "Accept:application/json" -H "Content-Type:application/json" \
-X PUT https://IP-ADDRESS/control/api/v1/livestreams/0/start
```

- This will start the stream with the current configuration.
- The expected response status code is `204 No Content`.
- Starting an already started stream will return a `409 Conflict` status code.

**Typical workflow for updating configuration:**

1. Stop the stream using the `/stop` endpoint.
2. Update the configuration as needed (see Appendix B).
3. Start the stream again using the `/start` endpoint.

## Appendix D

To authenticate with the API, you need to send a POST request to the login endpoint with your credentials.

```
curl -i -k -c cookie.txt -H "Accept:application/json" -H "Content-Type:application/json" \
-X POST https://IP-ADDRESS/user/login \
-d "{\"username\" : \"default\", \"password\": \"default\"}"
```

Use the cookie received in the response for subsequent requests:

```
curl -i -k -b cookie.txt -H "Accept:application/json" -H "Content-Type:application/json" \
-X GET https://IP-ADDRESS/control/api/v1/livestreams/0
```