Getting Started

Introduction

In this tutorial, we will connect video streams to the BrainFrame server. You can find the complete script on our GitHub repository.

Before we start, you should have BrainFrame Server and Client installed on your machine, If you don't have them yet, please follow the setup instructions.

This tutorial will use our Python library that wraps around the BrainFrame REST API. The library makes programming for BrainFrame in Python easier. If you're using Python, We strongly recommend you to use our Python API. Otherwise, you can always follow our REST API documentation to use the REST API directly.

Setup Environment

First, let's install the BrainFrame Python API library and setup the environment. Run the following commands. You can either do this in your virtual environment (recommended) or a root environment.

```
pip3 install brainframe-api
```

The Python API is now installed and ready for use.

The following APIs will be used in this tutorial:

- `api.get_stream_configurations()`
- `api.set_stream_configuration(...)`
- `api.start_analyzing(stream_id=...)`

Check existing streams

Now let's create a new, empty script. The first thing you want to do is to import the Python API library.

```
from pathlib import Path
from brainframe.api import BrainFrameAPI, bf_codecs
```

Then, initialize an API instance with the BrainFrame server URL. In this tutorial, we will connect to the BrainFrame server instance running on our local machine.

```
api = BrainFrameAPI("http://localhost")
```

The server is now connected, and we can start working with BrainFrame. First, let's see if there are any streams already connected to BrainFrame.

```
stream_configs = api.get_stream_configurations()
print("Existing streams: ", stream_configs)
```

If you run the script, and you only have a freshly-installed BrainFrame server, you should see just an empty list. Otherwise, the list of streams you have already connected will appear.

Create a New Stream Codec

Next, we will create a new stream configuration. The API function we will use is `api.set_stream_configuration(...)`. Looking at the function, it takes just a stream configuration codec as input. You can check the definition of different codecs in the Python library documentation.

Currently, we support three types of video sources:

- IP cameras
- Webcams
- Local files

For different types of video source, you need to set the different connection types and connection options. For more information, check this documentation.

IP Camera

For an IP camera, the connection type will be `IP_CAMERA`. In the `connection_options`, a valid `url` is required.

```
# Create a new IP camera StreamConfiguration codec
new_ip_camera_stream_config = bf_codecs.StreamConfiguration(
    # The display name on the client/in API responses
    name="IP Camera",
    connection_type=bf_codecs.ConnType.IP_CAMERA,
    connection_options={
        # The url of the IP camera
        "url": "your_ip_camera_url",
    },
    runtime_options={},
    premises_id=None,
)
```

Webcam

For a Webcam, the connection type is `WEBCAM`. Note: This must be connected to the server, not the client. In the `connection_options`, the device ID of the webcam is required. On Linux, you can find the device ID using:

```
ls /dev/ | grep video
```

After you have the device ID, use it in the codec.

```
# Create a local file StreamConfiguration codec
new_web_camera_stream_config = bf_codecs.StreamConfiguration(
    # The display name on the client/in API responses
    name="Webcam",
    connection_type=bf_codecs.ConnType.WEBCAM,
    connection_options={
        # The device ID of the web camera
        "device_id": 0,
    },
    runtime_options={},
    premises_id=None,
)
```

Local File

For a local file, you have to first upload the video file to the BrainFrame server's database and get a storage ID. The connection type is `FILE`. In the `connection_options`, the storage ID of the file is required.

```
# Upload the local file to the database and create a storage id
storage_id = api.new_storage(
    data=Path("../videos/shopping_cashier_gone.mp4").read_bytes(),
    mime_type="application/octet-stream"
)
# Create a local file stream configuration codec
new_local_file_stream_config = bf_codecs.StreamConfiguration(
    # The display name on the client side
    name="Local File",
    connection_type=bf_codecs.ConnType.FILE,
    # The storage id of the file
    connection_options={
        "storage_id": storage_id,
    },
    runtime_options={},
    premises_id=None,
)
```

Create a StreamConfiguration on the Server Side

Once we have the StreamConfiguration codec, we can tell BrainFrame Server to connect to it. In this tutorial, we will use the file-based codec. If you have an IP camera or a Webcam connected to the server, you can try using those as well.

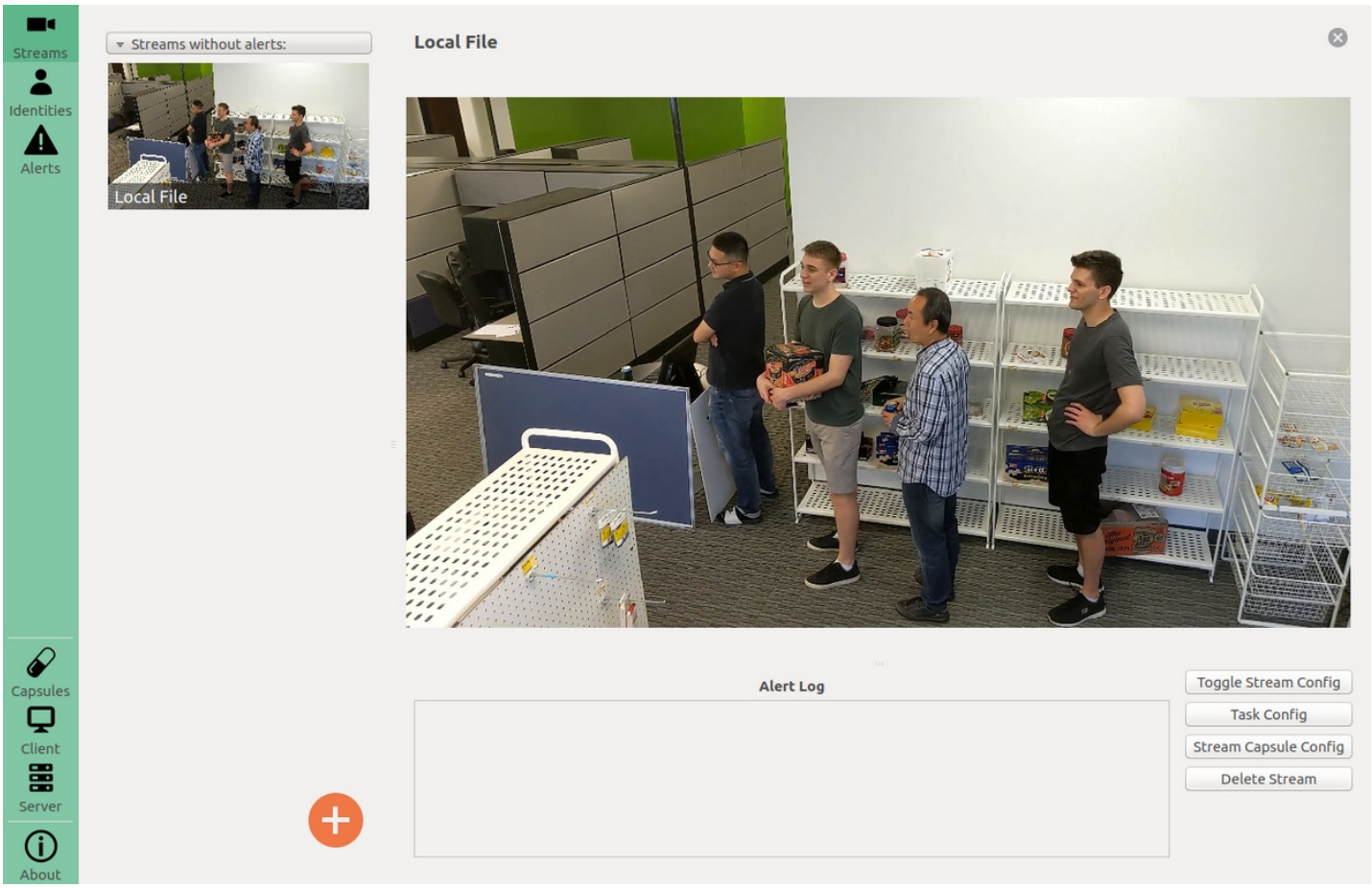
```
# Tell the server to connect to the stream configuration
new_local_file_stream_config = api.set_stream_configuration(
    new_local_file_stream_config)
```

Once the server receives the stream configuration, it will connect to it, assign a stream ID to it, and send it back. It is helpful to keep track of the IDs of the streams you have added using the return value.

Finally, don't forget to tell BrainFrame to start analyzing/preforming inference on the stream.

```
# Start analysis on the stream
api.start_analyzing(new_local_file_stream_config.id)
```

Now you should be able to see that stream in the BrainFrame client.


Streams
Identities
Alerts
Capsules
Client
Server
About
Streams without alerts:
Local File
Local File
Alert Log
Toggle Stream Config
Task Config
Stream Capsule Config
Delete Stream