Send WeChat Notifications

Introduction

In this tutorial, we will walk you through a simple use case for BrainFrame: getting WeChat Notification when there is no cashier in the checkout area. You can find the complete script on our GitHub repository.

Setup The Environment

In a previous tutorial, we installed the BrainFrame server, client, and Python API libraries. In this tutorial, the API functions we are going to use are:

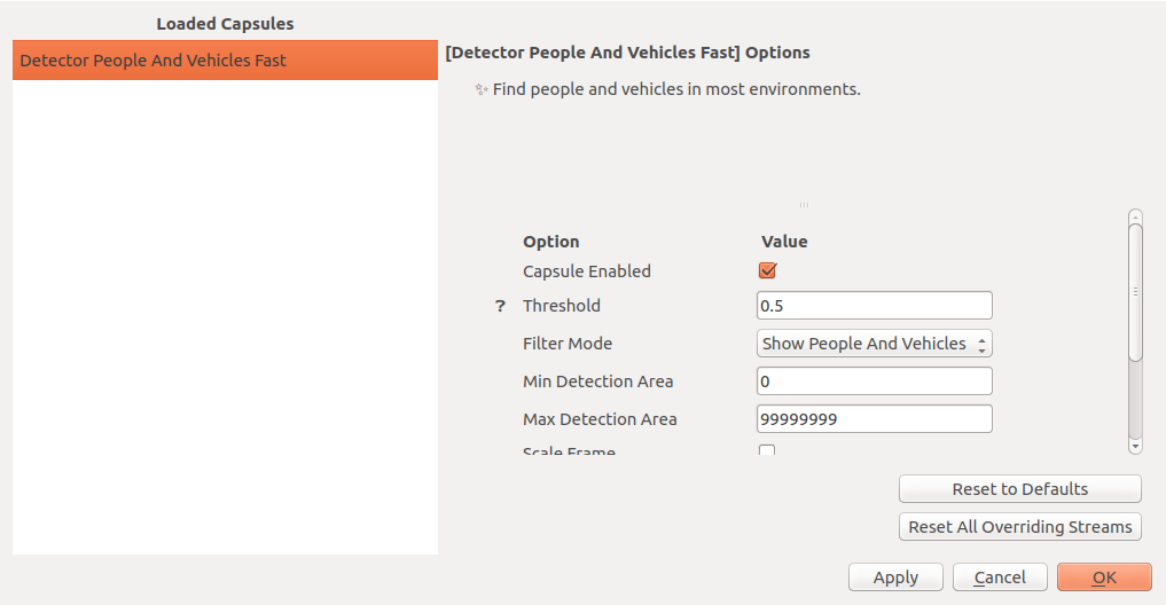
- `api.set_stream_configuration(...)`
- `api.set_zone(...)`
- `api.get_latest_zone_statuses()`
- `api.get_zone_status_stream()`

We will be using a third-party library called itchat to send notifications to WeChat. We'll install it using `pip`:

```
pip3 install itchat
```

We will also use one of our publicly available capsules, `detector_people_and_vehicles_fast`. You can grab it from our downloads page.

Before we start, you should have the BrainFrame server and client running, and capsules ready.



Log In to WeChat

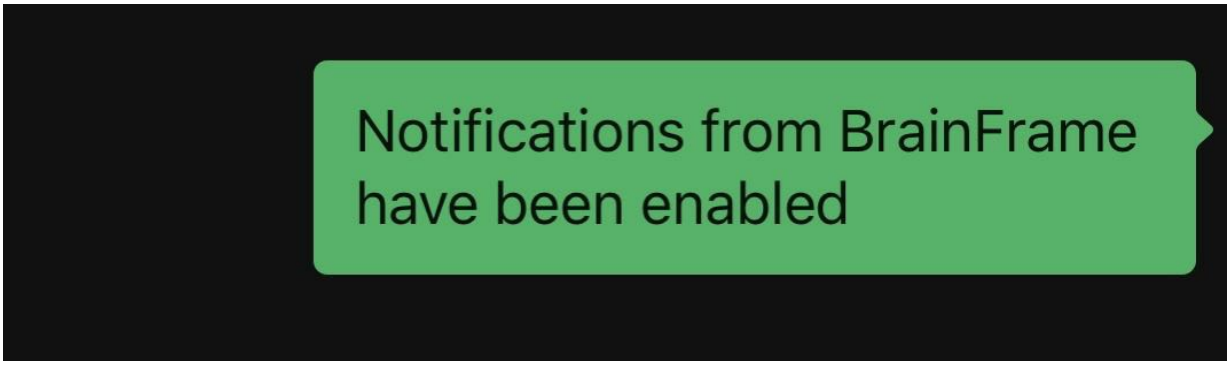
As usual, we will begin by importing our dependencies:

```
from pathlib import Path
import itchat as wechat
from brainframe.api import BrainFrameAPI, bf_codecs
```

Then, let's log in to our WeChat account and send a test message:

```
wechat.auto_login()
wechat.send_msg(f"Notifications from BrainFrame have been enabled",
                toUserName="filehelper")
```

The script will display a QR code, Scan it with your WeChat app and login. Your File Helper will then receive the message.



Create a New Stream from a Local File

First set the BrainFrame URL:

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

We will reuse the code snippet introduced in the previous tutorial to create a stream configuration on the BrainFrame server. We're going to use a simulated video file for this demo, but it will work with live video streams as well.

```
# Upload the local file to the BrainFrame server's database and get its storage
# ID
storage_id = api.new_storage(
    data=Path("../videos/shopping_cashier_gone.mp4").read_bytes(),
    mime_type="application/octet-stream"
)

# Create a StreamConfiguration with the storage ID
new_stream_config = bf_codecs.StreamConfiguration(
    # The display name on the client side
    name="Demo",
    # Specify that we're using a file
    connection_type=bf_codecs.ConnType.FILE,
    connection_options={
        # The storage id of the file
        "storage_id": storage_id,
    },
    runtime_options={},
    premises_id=None,
)

# Send the StreamConfiguration to the server to have it connect
new_stream_config = api.set_stream_configuration(new_stream_config)

# Tell the server to start analysis on the new stream
api.start_analyzing(new_stream_config.id)
```

You can download the demo video from our tutorial scripts repository. We recorded a video simulating a cashier serving customers.

Create a Zone and Setup an Alarm

In BrainFrame, alarms are associated with zones, and you can configure them through the client or through the API. You can check our documentation on Zones and Alarms for more information.

Using the API, we will create a zone around the check-out counter, and an alarm that will be triggered if no people are in that zone.

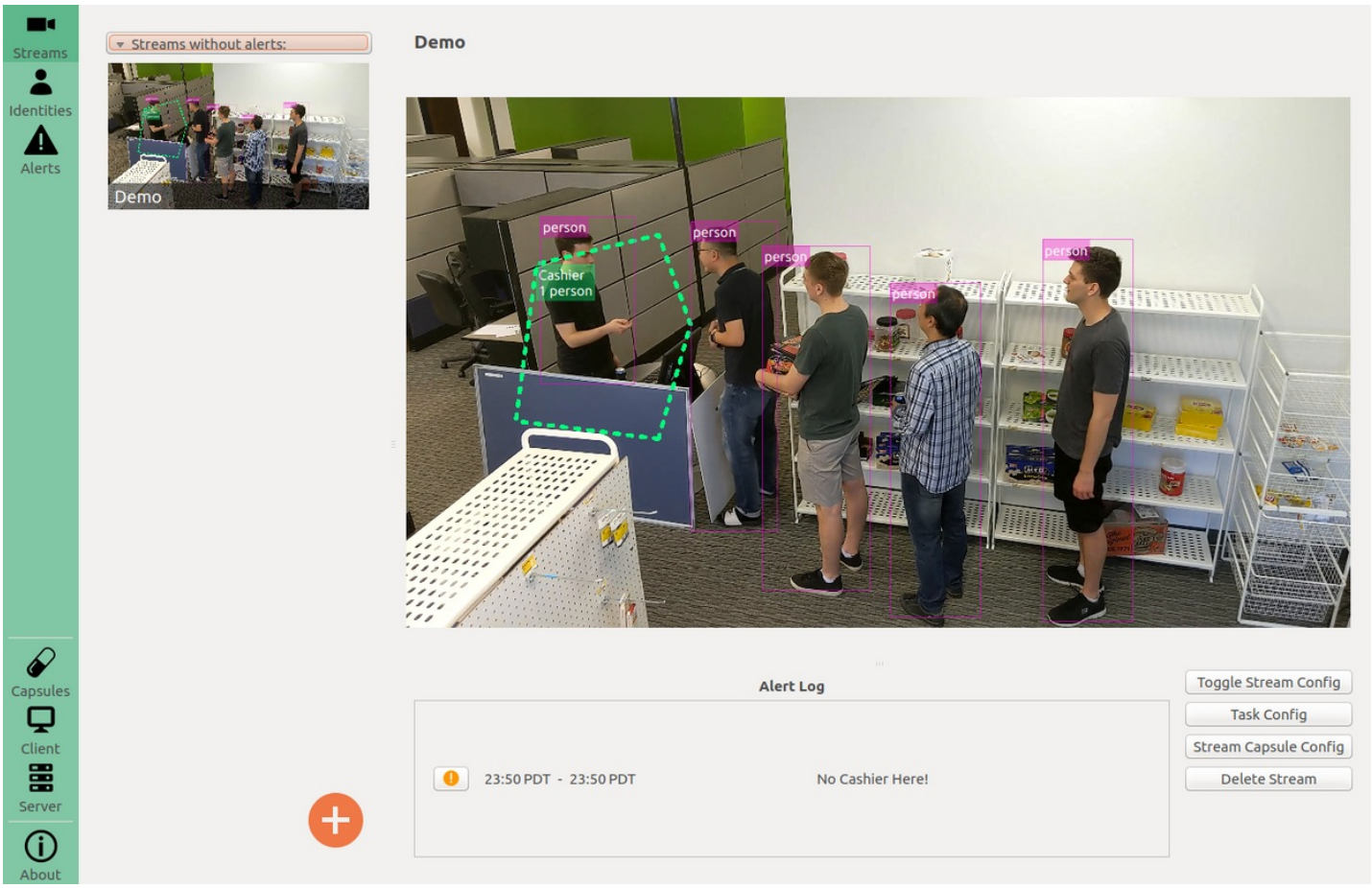
```
# Condition for the Alarm that will trigger when there is <1 person in the zone
# that it is assigned to
no_cashier_alarm_condition = bf_codecs.ZoneAlarmCountCondition(
    test=bf_codecs.CountConditionTestType.LESS_THAN,
    check_value=1,
    with_class_name="person",
    with_attribute=None,
    window_duration=5.0,
    window_threshold=0.5,
    intersection_point=bf_codecs.IntersectionPointType.BOTTOM,
)

# Create the ZoneAlarm. It will be active all day, everyday and will be
# triggered if the detection results satisfy the condition we created. Because
# use_active_time==False, the active end/start times will be ignored.
no_cashier_alarm = bf_codecs.ZoneAlarm(
    name="Missing Cashier!",
    count_conditions=[no_cashier_alarm_condition],
    rate_conditions=[],
    use_active_time=False,
    active_start_time="00:00:00",
    active_end_time="23:59:59",
)

# Create a Zone object with the above alarm
cashier_zone = bf_codecs.Zone(
    name="Cashier",
    stream_id=new_stream_config.id,
    alarms=[no_cashier_alarm],
    coords=[[513, 695], [223, 659], [265, 340], [513, 280], [578, 462]]
)

# Send the Zone to BrainFrame
api.set_zone(cashier_zone)
```

In the client, you will be able to see the zone there:



Get Zone Status

In BrainFrame, we use the `ZoneStatus` data structure to represent the inference results of frames. Let's use it to get ours.

We can use the API to get the latest `ZoneStatus` objects from BrainFrame.

```
zone_statuses = api.get_latest_zone_statuses()
print("Zone Statuses: ", zone_statuses)
```

The above code will print out the latest `ZoneStatus` objects for each stream with analysis/inference enabled. Warning: it can be a very long data structure, depending on how many streams there are and what capsules are loaded.

This is the most direct way to get the most recent inference results from BrainFrame. However, you have to call this function each time you want new results, which is a hassle.

A different API function, `get_zone_status_stream()` helps alleviate this issue. Instead of having relying on you polling for `ZoneStatus` objects, this function will return an iterable object to you. Each time BrainFrame has a new result available, it will be pushed to the iterator.

```
zone_status_iterator = api.get_zone_status_stream()
for zone_statuses in zone_status_iterator:
    print("Zone Statuses: ", zone_statuses)
```

This script will print the zone statuses as fast as the capsules can process the frames.

Get Alarms and Send Notifications to WeChat

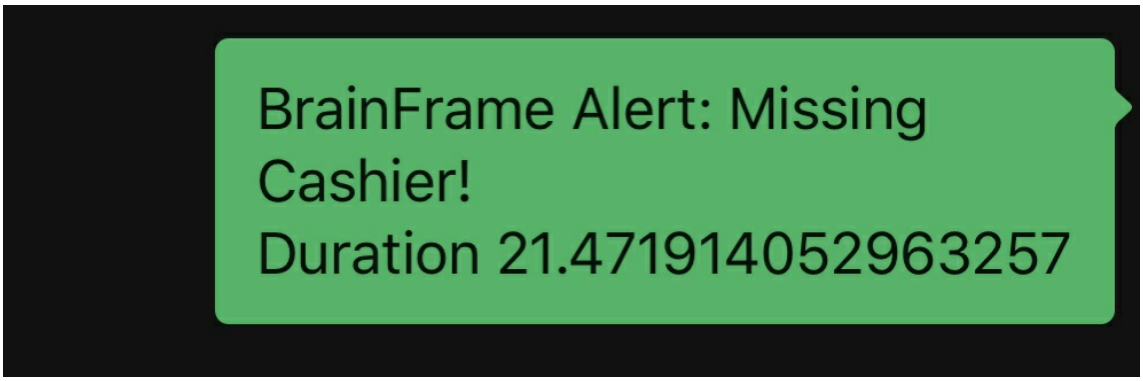
We can iterate through the zone status packets and check if there are any alerts that recently terminated after lasting >5 seconds. If there were, we send a notification. Note that for this example, the alert will only trigger *after* the cashier returns to the counter, a situation that is not as useful outside of the demo environment. The script will also only send one notification before exiting, to avoid sending too many notifications.

```
# Iterate through the zone status packets
for zone_status_packet in zone_status_iterator:
    for stream_id, zone_statuses in zone_status_packet.items():
        for zone_name, zone_status in zone_statuses.items():
            for alert in zone_status.alerts:
                # Check if the alert has ended
                if alert.end_time is None:
                    continue

                total_time = alert.end_time - alert.start_time
                # Check if the alert lasted for more than 5 seconds
                if total_time > 5:
                    alarm = api.get_zone_alarm(alert.alarm_id)
                    wechat.send_msg(
                        f"BrainFrame Alert: {alarm.name} \n"
                        f"Duration {total_time}", toUserName="filehelper")

                    # Stop here, for demo purposes
                    exit()
```

The script will send an alert to your WeChat File Helper if the cashier has been missing for more than 5 seconds. It will then exit the loop.



Logout Your WeChat Account

Finally, before we exit the script, don't forget to log out your WeChat account. Put the following code above `exit()`.

```
wechat.logout()
```