Event Notification Server Documentation

Pliable Pixels

Oct 12, 2019

Contents

CHAPTER 1

Installation of the Event Server (ES)

1.1 Download the repo

- Clone the project to some directory `git clone https://github.com/pliablepixels/zmeventnotification.git`
- Edit `zmeventnotification.ini` to your liking. More details about various parts of the configuration are explained later in this readme
- If you are behind a firewall, make sure you enable port `9000`, TCP, bi-directional (unless you changed the port in the code)
- If you are _not_ using machine learning hooks, make sure you comment out the `hook_script` attribute in the `[hook]` section of the ini file or you will see errors and you will not receive push
- We now need to install a bunch of dependencies (as described below)

1.2 Install Dependencies

Note that I assume you have other development packages already installed like `make`, `gcc` etc as the plugins may require them. The following perl packages need to be added (these are for Ubuntu - if you are on a different OS, you'll have to figure out which packages are needed - I don't know what they might be)

(**General note** - some users may face issues installing dependencies via `perl -MCPAN -e "Module::Name"`. If so, its usually more reliable to get into the CPAN shell and install it from the shell as a 2 step process. You'd do that using `sudo perl -MCPAN -e shell` and then whilst inside the shell, `install Module::Name`)

- Crypt::MySQL (if you have updated to ZM 1.34, this is no longer needed)
- Net::WebSocket::Server
- Config::IniFiles (you may already have this installed)
- Crypt::Eksblowfish::Bcrypt (if you have updated to ZM 1.34, you will already have this)

Installing these dependencies is as simple as:

```
sudo perl -MCPAN -e "install Crypt::MySQL"
sudo perl -MCPAN -e "install Config::IniFiles"
sudo perl -MCPAN -e "install Crypt::Eksblowfish::Bcrypt"
```

If after installing them you still see errors about these libraries missing, please launch a CPAN shell - see General Note above.

If you face issues installing Crypt::MySQL try this instead: (Thanks to aaronl)

```
sudo apt-get install libcrypt-mysql-perl
```

If you face issues installing Crypt::Eksblowfish::Bcrypt, this this instead:

```
sudo apt-get install libcrypt-eksblowfish-perl
```

If there are issues installing Config::IniFiles and the errors are related to Module::Build missing, use following command to get this module in debian based systems and install Config::IniFiles again.

```
sudo apt-get install libmodule-build-perl
```

Next up install WebSockets

```
sudo apt-get install libyaml-perl
sudo apt-get install make
sudo perl -MCPAN -e "install Net::WebSocket::Server"
```

Then, you need JSON.pm installed. It's there on some systems and not on others In ubuntu, do this to install JSON:

```
sudo apt-get install libjson-perl
```

Get HTTPS library for LWP:

```
perl -MCPAN -e "install LWP::Protocol::https"
```

If you want to enable MQTT:

```
perl -MCPAN -e "install Net::MQTT::Simple"
```

Some notes on MQTT:

- A minimum version of MQTT 3.1.1 is required
- If your `MQTT:Simple` library was installed a while ago, you may need to update it. A new `login` method was added to that library on Dec 2018 which is required (ref)

Note that starting 1.0, we also use `File::Spec`, `Getopt::Long` and `Config::IniFiles` as additional libraries. My ubuntu installation seemed to include all of this by default (even though `Config::IniFiles` is not part of base perl).

If you get errors about missing libraries, you'll need to install the missing ones like so:

```
perl -MCPAN -e "install XXXX" # where XXX is Config::IniFiles, for example
```

1.3 Configure SSL certificate (Generate new, or use ZoneMinder certs if you are already using HTTPS)

NOTE: If you plan on using picture messaging in zmNinja, then you cannot use self signed certificates. You will need to generate a proper certificate. LetsEncrypt is free and perfect for this.

If you are using secure mode (default) you **also need to make sure you generate SSL certificates otherwise the script won't run** If you are using SSL for ZoneMinder, simply point this script to the certificates.

If you are not already using SSL for ZoneMinder and don't have certificates, generating them is as easy as:

(replace /etc/zm/apache2/ssl/ with the directory you want the certificate and key files to be stored in)

```
sudo openssl req -x509 -nodes -days 4096 -newkey rsa:2048 -keyout /etc/zm/apache2/ssl/
↪zoneminder.key -out /etc/zm/apache2/ssl/zoneminder.crt
```

It's **very important** to ensure the `Common Name` selected while generating the certificate is the same as the hostname or IP of the server. For example if you plan to access the server as `myserver.ddns.net` Please make sure you use `myserver.ddns.net` as the common name. If you are planning to access it via IP, please make sure you use the same IP.

Once you do that please change the following options in the config file to point to your SSL certs/keys:

```
[ssl]
cert = /etc/zm/apache2/ssl/zoneminder.crt
key = /etc/zm/apache2/ssl/zoneminder.key
```

1.3.1 IOS Users

On some IOS devices and when using self signed certs, I noticed that zmNinja was not able to register with the event server when it was using WSS (SSL enabled) and self-signed certificates. To solve this, I had to email myself the zoneminder certificate (`zoneminder.crt`) file and install it in the phone. Why that is needed only for WSS and not for HTTPS is a mystery to me. The alternative is to run the eventserver in WS mode by disabling SSL.

1.4 Making sure everything is running (in manual mode)

- I am assuming you have downloaded the files to your current directory in the step below
- Make sure you do a `chmod a+x ./zmeventnotification.pl`
- Start the event server manually first using `sudo -u www-data ./zmeventnotification.pl --config ./zmeventnotification.ini` (Note that if you omit `--config` it will look for `/etc/zm/zmeventnotification.ini` and if that doesn't exist, it will use default values) and make sure you check syslogs to ensure its loaded up and all dependencies are found. If you see errors, fix them. Then exit and follow the steps below to start it along with Zoneminder. Note that the `-u www-data` runs this command with the user id that apache uses (in some systems this may be `apache` or similar). It is important to run it using the same user id as your webserver because that is the permission zoneminder will use when run as a daemon mode.
- Its is HIGHLY RECOMMENDED that you first start the event server manually from terminal, as described above and not directly dive into daemon mode (described below) and ensure you inspect syslog to validate all logs are correct and THEN make it a daemon in ZoneMinder. If you don't, it will be hard to know what is going wrong. See *this section* later that describes how to make sure its all working fine from command line.

1.5 Install the server (optionally along with hooks)

NOTE : By default `install.sh` moves the ES script to `/usr/bin`. If your ZM install is elsewhere, like `/usr/local/bin` please modify the `TARGET_BIN` variable in `install.sh` before executing it.

- You can now move the ES to the right place by simply doing `sudo ./install.sh` and following prompts. Other options are below:
- Execute `sudo ./install.sh --no-install-hook` to move the ES to the right place without installing machine learning hooks
- In ZM 1.32.0 and above, go to your web interface, and go to `Options->Systems` and enable `OPT_USE_EVENTNOTIFICATION` and you are all set.

The rest of this section is NOT NEEDED for 1.32.0 and above!

Deprecated since version 1.32.0.

WARNING : Do NOT do this before you run it manually as I've mentioned above to test. Make sure it works, all packages are present etc. before you add it as a daemon as if you don't and it crashes you won't know why

(Note if you have compiled from source using cmake, the paths may be `/usr/local/bin` not `/usr/bin`)

- Edit `/usr/bin/zmdc.pl` and in the array `@daemons` (starting line 89 or so, may change depending on ZM version) add `'zmeventnotification.pl'` like this
- Edit `/usr/bin/zmpkg.pl` and around line 275 (exact line # may change depending on ZM version), right after the comment that says `#this is now started unconditionally` and right before the line that says `runCommand( "zmdc.pl start zmfilter.pl" );` start zmeventnotification.pl by adding `runCommand( "zmdc.pl start zmeventnotification.pl" );` like this
- Make sure you restart ZM. Rebooting the server is better - sometimes zmdc hangs around and you'll be wondering why your new daemon hasn't started
- To check if its running do a `zmdc.pl status zmeventnotification.pl`

You can/should run it manually at first to check if it works

CHAPTER 2

Machine Learning Hooks

Note: Before you install machine learnings hooks, please make sure you have installed the Event Notification Server (*Installation of the Event Server (ES)*) and have it working properly

Important: Please don't ask me basic questions like "pip3 command not found" or "cv2 not found" - what do I do? Hooks require some terminal knowledge and familiarity with troubleshooting. I don't plan to provide support for these hooks. They are for reference only

2.1 Limitations

- Only tested with ZM 1.32+. May or may not work with older versions
- Needs Python3 (I used to support Python2, but not any more). Python2 will be deprecated in 2020. May as well update.

2.2 What

Kung-fu machine learning goodness.

This is an example of how you can use the `hook` feature of the notification server to invoke a custom script on the event before it generates an alarm. I currently support object detection and face recognition.

Please don't ask me questions on how to use them. Please read the extensive documentation and ini file configs

2.3 Installation

2.3.1 Option 1: Automatic install

- You need to have `pip3` installed. On ubuntu, it is `sudo apt install python3-pip`, or see this
- Clone the event server and go to the `hook` directory

```
git clone https://github.com/pliablepixels/zmeventnotification # if you don't already␣
↪have it downloaded

cd zmeventnotification
```

- (OPTIONAL) Edit `hook/detect_wrapper.sh` and change:
  - `CONFIG_FILE` to point to the right config file, if you changed paths

```
sudo -H ./install.sh # and follow the prompts
```

Note: if you want to add "face recognition" you also need to do

```
sudo apt-get install libopenblas-dev liblapack-dev libblas-dev  # not mandatory, but␣
↪gives a good speed boost!
sudo -H pip3 install face_recognition # mandatory
```

Takes a while and installs a gob of stuff, which is why I did not add it automatically, especially if you don't need face recognition.

Note, if you installed `face_recognition` earlier without blas, do this:

```
sudo -H pip3 uninstall dlib
sudo -H pip3 uninstall face-recognition
sudo apt-get install libopenblas-dev liblapack-dev libblas-dev # this is the␣
↪important part
sudo -H pip3 install dlib --verbose --no-cache-dir # make sure it finds openblas
sudo -H pip3 install face_recognition
```

2.3.2 Option 2: Manual install

If automatic install fails for you, or you like to be in control:

```
git clone https://github.com/pliablepixels/zmeventnotification # if you don't already␣
↪have it downloaded
```

- Install object detection files:

```
cd zmeventnotification/
sudo -H pip3 install hook/
```

Note: if you want to add "face recognition" you also need to do

```
sudo apt-get install libopenblas-dev liblapack-dev libblas-dev  # not mandatory, but␣
↪gives a good speed boost!
sudo -H pip3 install face_recognition # mandatory
```

Takes a while and installs a gob of stuff, which is why I did not add it automatically, especially if you don't need face recognition.

Note, if you installed `face_recognition` without blas, do this:

:: sudo -H pip3 uninstall dlib sudo -H pip3 uninstall face-recognition sudo apt-get install libopenblas-dev liblapack-dev libblas-dev # this is the important part sudo -H pip3 install dlib –verbose –no-cache-dir # make sure it finds openblas sudo -H pip3 install face_recognition

- You now need to download configuration and weight files that are required by the machine learning magic. Note that you don't have to put them in `/var/lib/zmeventnotification` -> use whatever you want (and change variables in `detect_wrapper.sh` script if you do)

```
sudo mkdir -p /var/lib/zmeventnotification/images
sudo mkdir -p /var/lib/zmeventnotification/models

# if you are using face recognition, create this folder
# after that you need to copy images of faces you want to detect
# to this folder
sudo mkdir -p /var/lib/zmeventnotification/known_faces

# if you want to use YoloV3 (slower, accurate)
sudo mkdir -p /var/lib/zmeventnotification/models/yolov3 # if you are using YoloV3
sudo wget https://raw.githubusercontent.com/pjreddie/darknet/master/cfg/yolov3.cfg -O␣
↪/var/lib/zmeventnotification/models/yolov3/yolov3.cfg
sudo wget https://raw.githubusercontent.com/pjreddie/darknet/master/data/coco.names -
↪O /var/lib/zmeventnotification/models/yolov3/yolov3_classes.txt
sudo wget https://pjreddie.com/media/files/yolov3.weights -O /var/lib/
↪zmeventnotification/models/yolov3/yolov3.weights

--OR--

# if you want to use TinyYoloV3 (faster, less accurate)
sudo mkdir -p /var/lib/zmeventnotification/models/tinyyolo # if you are using␣
↪TinyYoloV3
sudo wget https://pjreddie.com/media/files/yolov3-tiny.weights -O /var/lib/
↪zmeventnotification/models/tinyyolo/yolov3-tiny.weights
sudo wget https://raw.githubusercontent.com/pjreddie/darknet/master/cfg/yolov3-tiny.
↪cfg -O /var/lib/zmeventnotification/models/tinyyolo/yolov3-tiny.cfg
sudo wget https://raw.githubusercontent.com/pjreddie/darknet/master/data/coco.names -
↪O /var/lib/zmeventnotification/models/tinyyolo/yolov3-tiny.txt
```

- Copy over the object detection config file

```
sudo cp objectconfig.ini /etc/zm
```

- Now make sure it all RW accessible by `www-data` (or `apache`)

```
sudo chown -R www-data:www-data /var/lib/zmeventnotification/ #(change www-data␣
↪to apache for CentOS/Fedora)
```

- (OPTIONAL) Edit `detect_wrapper.sh` and change:
  - `CONFIG_FILE` to point to the right config file, if you changed paths
- Now copy your detection file to `/usr/bin`

```
sudo cp detect.py /usr/bin
```

2.4 Post install steps

- Make sure you edit your installed `objectconfig.ini` to the right settings. You MUST change the `[general]` section for your own portal.
- Make sure the `CONFIG_FILE` variable in `detect_wrapper.sh` is correct

2.5 Test operation

```
sudo -u www-data /usr/bin/detect_wrapper.sh <eid> <mid> # replace www-data with␣
↪apache if needed
```

This will try and download the configured frame for alarm and analyze it. Replace with your own EID (Example 123456) The files will be in `/var/lib/zmeventnotification/images` For example: if you configured `frame_id` to be `bestmatch` you'll see two files `<eid>-alarm.jpg` and `<eid>-snapshot.jpg` If you configured `frame_id` to be `snapshot` or a specific number, you'll see one file `<eid>.jpg`

The `<mid>` is optional and is the monitor ID. If you do specify it, it will pick up the right mask to apply (if it is in your config)

The above command will also try and run detection.

If it doesn't work, go back and figure out where you have a problem

- Other configuration notes, after you get everything working
  - Set `delete_after_analyze` to `yes` so that downloaded images are removed after analysis. In the default installation, the images are kept in `/var/lib/zmeventnotification/images` so you can debug.
  - Remember these rules:
    * `frame_id=snapshot` will work for any ZM >= 1.32
    * If you are running ZM < 1.33, to enable `bestmatch` or `alarm` you need to enable the monitor to store JPEG frames in its ZM monitor->storage configuration in ZM
    * If you are running ZM >= 1.33, you can use all fid modes without requiring to enable frames in storage

2.6 Upgrading

To upgrade at a later stage, see *How do I safely upgrade zmeventnotification to new versions?*.

2.7 Logging

Starting version 4.0.x, the hooks now use ZM logging, thanks to a python wrapper I wrote recently that taps into ZM's logging system. This also means it is no longer as easy as enabling `log_level=debug` in `objdetect.ini`. Infact, that option has been removed. Follow standard ZM logging options for the hooks. Here is what I do:

- In `ZM->Options->Logs`:
  - LOG_LEVEL_FILE = debug
  - LOG_LEVEL_SYSLOG = Info

- LOG_LEVEL_DATABASE = Info
- LOG_DEBUG is on
- LOG_DEBUG_TARGET = `_zmesdetect` (if you have other targets, just separate them with `|` - example, `_zmc|_zmesdetect`). If you want to enable debug logs for both the ES and the hooks, your target will look like `_zmesdetect|_zmeventnotification`. You can also enabled debug logs for just one monitor's hooks like so: `_zmesdetect_m5|_zmeventnotification`. This will enable debug logs only when hooks are run for monitor 5.

The above config. will store debug logs in my `/var/log/zm` directory, while Info level logs will be recorded in syslog and DB.

You will likely need to restart ZM after this.

So now, to view hooks/detect logs, all I do is:

```
tail -f  /var/log/zm/zmesdetect*.log
```

Note that the detection code registers itself as `zmesdetect` with ZM. When it is invoked with a specific monitor ID (usually the case), then the component is named `zmesdetect_mX.log` where `X` is the monitor ID. In other words, that now gives you one log per monitor (just like `/var/log/zm/zmc_mX.log`) which makes it easy to debug/isolate.

2.8 Troubleshooting

- In general, I expect you to debug properly. Please don't ask me basic questions without investigating logs yourself
- Always run `detect_wrapper.sh` in manual mode first to make sure it works
- To get debug logs, Make sure your `LOG_DEBUG` in ZM Options->Logs is set to on and your `LOG_DEBUG_TARGET` option includes `_zmesdetect` (or is empty)
- You can view debug logs for detection by doing `tail -f /var/log/zm/zmesdetect*.log`
- One of the big reasons why object detection fails is because the hook is not able to download the image to check. This may be because your ZM version is old or other errors. Some common issues:
  - Make sure your `objectconfig.ini` section for `[general]` are correct (portal, user,admin)
  - For object detection to work, the hooks expect to download images of events using `https://yourportal/zm/?view=image&eid=<eid>&fid=snapshot` and possibly `https://yourportal/zm/?view=image&eid=<eid>&fid=alarm`
  - Open up a browser, log into ZM. Open a new tab and type in `https://yourportal/zm/?view=image&eid=<eid>&fid=snapshot` in your browser. Replace `eid` with an actual event id. Do you see an image? If not, you'll have to fix/update ZM. Please don't ask me how. Please post in the ZM forums
  - Open up a browser, log into ZM. Open a new tab and type in `https://yourportal/zm/?view=image&eid=<eid>&fid=alarm` in your browser. Replace `eid` with an actual event id. Do you see an image? If not, you'll have to fix/update ZM. Please don't ask me how. Please post in the ZM forums

2.9 Types of detection

You can switch detection type by using `model=<detection_type1>,<detection_type2>,...` in your `objectconfig.ini`

Example:

`model=yolo,hog,face` will run full Yolo, then HOG, then face recognition.

Note that you can change `model` on a per monitor basis too. Read the comments in `objectconfig.ini`

If you select yolo, you can add a `model_type=tiny` to use tiny YOLO instead of full yolo weights. Again, please readd the comments in `objectconfig.ini`

2.9.1 How to use license plate recognition

Two ALPR options are provided:

- Plate Recognizer . It uses a deep learning model that does a far better job than OpenALPR (based on my tests). The class is abstracted, obviously, so in future I may add local models. For now, you will have to get a license key from them (they have a free tier that allows 2500 lookups per month)
- OpenALPR . While OpenALPR's detection is not as good as Plate Recognizer, when it does detect, it provides a lot more information (like car make/model/year etc.)

To enable alpr, simple add *alpr* to *models*. You will also have to add your license key to the `[alpr]` section of `objdetect.ini`

This is an example config that uses plate recognizer:

```
models = yolo,alpr

[alpr]
alpr_service=plate_recognizer
# If you want to host a local SDK https://app.platerecognizer.com/sdk/
#alpr_url=https://localhost:8080
# Plate recog replace with your api key
alpr_key=KEY
# if yes, then it will log usage statistics of the ALPR service
platerec_stats=no
# If you want to specify regions. See http://docs.platerecognizer.com/#regions-
↪supported
#platerec_regions=['us','cn','kr']
# minimal confidence for actually detecting a plate
platerec_min_dscore=0.1
# minimal confidence for the translated text
platerec_min_score=0.2
```

This is an example config that uses OpenALPR:

```
models = yolo,alpr

[alpr]
alpr_service=open_alpr
alpr_key=SECRET

# For an explanation of params, see http://doc.openalpr.com/api/?api=cloudapi
openalpr_recognize_vehicle=1
```

(continues on next page)

(continued from previous page)

```
openalpr_country=us
openalpr_state=ca
# openalpr returns percents, but we convert to between 0 and 1
openalpr_min_confidence=0.3
```

Leave `alpr_use_after_detection_only` to the default values.

How license plate recognition will work

- To save on API calls, the code will only invoke remote APIs if a vehicle is detected
- This also means you MUST specify yolo along with alpr

2.9.2 How to use face recognition

Face Recognition uses this library. Before you try and use face recognition, please make sure you did a `sudo -H pip3 install face_recognition` The reason this is not automatically done during setup is that it installs a lot of dependencies that takes time (including dlib) and not everyone wants it.

Face recognition limitations

Don't expect magic with overhead cameras. This library requires a reasonable face orientation (works for front facing, or somewhat side facing poses) and does not work for full profiles or completely overhead faces. Take a look at the accuracy wiki of this library to know more about its limitations. Also note that I found *cnn* mode is much more accurage than *hog* mode. However, *cnn* comes with a speed and memory tradeoff.

Configuring face recognition

- Make sure you have images of people you want to recognize in `/var/lib/zmeventnotification/known_faces`
- Only one image per person
- For example, you may have the following image setup:

```
/var/lib/zmeventnotification/known_faces
    + david_gilmour.jpg
    + ramanujan.jpg
    + bruce_lee.jpg
```

- When face recognition is triggered, it will load each of these files and if there are faces in them, will load them and compare them to the alarmed image

known faces images

- Only put in one image per person
- Make sure the face is recognizable
- crop it to around 400 pixels width (doesn't seem to need bigger images, but experiment. Larger the image, the larger the memory requirements)

Yo, it can't recognize faces

- Look at debug logs.
  - If it says "no faces loaded" that means your known images don't have recognizable faces
  - If it says "no faces found" that means your alarmed image doesn't have a face that is recognizable
  - Read comments about `num_jitters`, `model`, `upsample_times` in `objectconfig.ini`
- Experiment. Read the accuracy wiki link I posted in the previous section

2.10 Performance comparison

DNNs perform very well on a GPU. My ZM server doesn't have a GPU. On a Intel Xeon 3.16GHz 4Core machine:

With BLAS installed, here are my performance stats: All tests are with a 600px wide image

- Face Detection with CNN:

```
[|--> model:face init took: 1.901829s]
[|--> model:face detection took: 4.218463s] (Fyi, this varies, from 4.x - 6.xs)
```

- Face Detection with HOG:

```
[|--> model:face init took: 1.866364s]
[|--> model:face detection took: 0.263436s]
```

- YoloV3 object detection (with full yolov3 weights)

```
[|--> model:yolo init took: 1.9e-05s]
[|--> model:yolo detection took: 2.487402s]
```

As always, if you are trying to figure out how this works, do this in 3 steps:

2.10.1 Manually testing if detection is working well

You can manually invoke the detection module to check if it works ok:

```
./sudo -u www-data /usr/bin/detect.py --config /etc/zm/objectconfig.ini  --eventid
↪<eid> --monitorid <mid>
```

The `--monitorid <mid>` is optional and is the monitor ID. If you do specify it, it will pick up the right mask to apply (if it is in your config)

STEP 1: Make sure the scripts(s) work

- Run the python script manually to see if it works (refer to sections above on how to run them manually)
- `./detect_wrapper.sh <eid> <mid>` –> make sure it downloads a proper image for that eid. Make sure it correctly invokes detect.py If not, fix it. (`<mid>` is optional and is used to apply a crop mask if specified)
- Make sure the `image_path` you've chosen in the config file is WRITABLE by www-data (or apache) before you move to step 2

STEP 2: run zmeventnotification in MANUAL mode

- `sudo zmdc.pl stop zmeventnotification.pl`

- change console_logs to yes in `zmeventnotification.ini`
- `sudo -u www-data ./zmeventnotification.pl --config ./zmeventnotification.ini`
- Force an alarm, look at logs

STEP 3: integrate with the actual daemon - You should know how to do this already

CHAPTER 3

Breaking Changes

3.1 Version 4.4 onwards

- If you are using picture messaging, then the URL format has changed. Please REMOVE `&username=<user>&password=<passwd>` from the URL and put them into the `picture_portal_username` and `picture_portal_password` fields respectively

3.2 Version 4.1 onwards

- Hook versions will now always be `<ES version>.x`, so in this case `4.1.x`
- Hooks have now migrated to using a proper python ZM logger module so it better integrates with ZM logging
- To view detection logs, you now need to follow the standard ZM logging process. See *Logging* documentation for more details)
- You no longer have to manually install python requirements, the setup process should automatically install them
- If you are using MQTT and your `MQTT::Simple` library was installed a while ago, you may need to update it. A new `login` method was added to that library on Dec 2018 which is required (ref)

3.3 Version 3.9 onwards

- Hooks now add ALPR, so you need to run *sudo -H pip install -r requirements.txt* again
- See modified objectconfig.ini if you want to add ALPR. Currently works with platerecognizer.com, so you will need an API key. See hooks docs for more info

3.4 Version 3.7 onwards

- There were some significant changes to ZM (will be part of 1.34), which includes migration to Bcrypt for passwords. Changes were made to support Bcrypt, which means you will have to add additional libraries. See the installation guide.

3.5 version 3.3 onwards

- Please use `yes` or `no` instead of `1` and `0` in `zmeventnotification.ini` to maintain consistency with `objectconfig.ini`
- In `zmeventnotification.ini`, `store_frame_in_zm` is now `hook_pass_image_path`

3.6 version 3.2 onwards

- Changes in paths for everything. - event server config file now defaults to `/etc/zm`
- hook config now defaults to `/etc/zm`
- Push token file now defaults to `/var/lib/zmeventnotification/push`
- all object detection data files default to `/var/lib/zmeventnotification`
- **If you are migrating from a previous version:**
  - Make a copy of your `/etc/zmeventnotification.ini` and `/var/detect/objectconfig.ini` (if you are using hooks)
  - Run `sudo -H ./install.sh` again inside the repo, let it set up all the files
  - Compare your old config files to the news ones at `/etc/zm` and make necessary changes
  - Make sure everything works well
  - You can now delete the old `/var/detect` folder as well as `/etc/zmeventnotification.ini`
  - Run zmNinja again to make sure its token is registered in the new tokens file (in `/var/lib/zmeeventnotification/push/tokens.txt`)

CHAPTER 4

FAQ

4.1 Machine Learning! Mmm..Machine Learning!

Easy. You will first have to read this document to correctly install this server along with zoneminder. Once it works well, you can explore how to enable Machine Learning based object detection that can be used along with ZoneMinder alarms. If you already have this server figured out, you can skip directly to the machine learning part (*Machine Learning Hooks*)

4.2 What is it?

A WSS (Secure Web Sockets) and/or MQTT based event notification server that broadcasts new events to any authenticated listeners. (As of 0.6, it also includes a non secure websocket option, if that's how you want to run it)

4.3 Why do we need it?

- The only way ZoneMinder sends out event notifications via event filters - this is too slow
- People developing extensions to work with ZoneMinder for Home Automation needs will benefit from a clean interface
- Receivers don't poll. They keep a web socket open and when there are events, they get a notification
- Supports WebSockets, MQTT and Apple/Android push notification transports
- Offers an authentication layer
- Allows you to integrate custom hooks that can decide if an alarm needs to be sent out or not (an example of how this can be used for person detection is provided)

4.4 Is this officially developed by ZM developers?

No. I developed it for zmNinja, but you can use it with your own consumer.

4.5 How can I use this with Node-Red or Home Assistant?

As of version 1.1, the event server also supports MQTT (Contributed by @vajonam). zmeventnotification server can be configured to broadcast on a topic called `/zoneminder/<monitor-id>` which can then be consumed by Home Assistant or Node-Red.

To enable this, set `enable = 1` under the `[mqtt]` section and specify the `server` to broadcast to.

You will also need to install the following module for this work

```
perl -MCPAN -e "install Net::MQTT::Simple"
```

The MQTT::Simple module is known to work only with Mosquitto as of 10 Jun 2019. It does not work correctly with the RabbitMQ MQTT plugin. The easiest workaround if you have an unsupported MQTT system is to install Mosquitto on the Zoneminder system itself and bridge that to RabbitMQ. You can bind Mosquitto to 127.0.0.1 and disable authentication to keep it simple. The eventserver.pl is then configured to send events to the local Mosquitto. This is an example known working bridge set up (on Ubuntu, for example, this is put into /etc/mosquitto/conf.d/local.conf):

```
bind_address 127.0.0.1
allow_anonymous true
connection bridge-zm2things
address 10.10.1.20:1883
bridge_protocol_version mqttv311
remote_clientid bridge-zm2things
remote_username zm
remote_password my_mqtt_zm_password
try_private false
topic # out 0
```

Set the address, remote_username and remote_password for Mosquitto to use on the RabbitMQ. Note that this is a one way bridge, so there is only a topic # out 0. try_private false is needed to avoid a similar error to using MQTT::Simple.

4.6 Disabling security

While I don't recommend either, several users seem to be interested in the following

- To run the eventserver on Websockets and not Secure Websockets, use `enable = 0` in the `[ssl]` section of the configuration file.
- To disable ZM Auth checking (be careful, anyone can get all your data INCLUDING passwords for ZoneMinder monitors if you open it up to the Internet) use `enable = 0` in the `[auth]` section of the configuration file.

4.7 How do I safely upgrade zmeventnotification to new versions?

4.7.1 STEP 1: get the latest code

Download the latest version & change dir to it:

```
git clone https://github.com/pliablepixels/zmeventnotification.git
cd zmeventnotification/
```

4.7.2 STEP 2: stop the current ES

```
sudo zmdc.pl stop zmeventnotification.pl
```

4.7.3 STEP 3: Make a backup of your config files

Before you execute the next step you may want to create a backup of your existing `zmeventnotification.ini` and `objectconfig.ini` config files. The script will prompt you to overwrite. If you say 'Y' then your old configs will be overwritten. Note that old configs are backed up using suffixes like `~1`, `~2` etc. but it is always good to backup on your own.

4.7.4 STEP 4: Execute the install script

NOTE : By default `install.sh` moves the ES script to `/usr/bin`. If your ZM install is elsewhere, like `/usr/local/bin` please modify the `TARGET_BIN` variable in `install.sh` before executing it.

```
sudo ./install.sh
```

Follow prompts. Note that just copying the ES perl file to `/usr/bin` is not sufficient. You also have to install the updated machine learning hook files if you are using them. That is why `install.sh` is better. If you are updating, make sure not to overwrite your config files (but please read breaking changes to see if any config files have changed). Note that the install script makes a backup of your old config files using `~n` suffixes where `n` is the backup number. However, never hurts to make your own backup first.

4.7.5 STEP 5: Start the new updated server

```
sudo zmdc.pl start zmeventnotification.pl
```

Make sure you look at the logs to make sure its started properly

4.8 Configuring the notification server

4.8.1 Understanding zmeventnotification configuration

Starting v1.0, @synthead reworked the configuration (brilliantly) as follows:

- If you just run `zmeventnotification.pl` it will try and load `/etc/zm/zmeventnotification.ini`. If it doesn't find it, it will use internal defaults
- If you want to override this with another configuration file, use `zmeventnotification.pl --config /path/to/your/config/filename.ini`.
- Its always a good idea to validate you config settings. For example:

```
sudo /usr/bin/zmeventnotification.pl --check-config

03/31/2018 16:52:23.231955 zmeventnotification[29790].INF [using config file: /etc/zm/
↪zmeventnotification.ini]
Configuration (read /etc/zm/zmeventnotification.ini):

Port ........................... 9000
Address ........................ XX.XX.XX.XX
Event check interval .......... 5
Monitor reload interval ....... 300

Auth enabled ................... true
Auth timeout ................... 20

Use FCM ........................ true
FCM API key .................... (defined)
Token file ..................... /var/lib/zmeventnotification/push/tokens.txt

SSL enabled .................... true
SSL cert file .................. /etc/zm/apache2/ssl/zoneminder.crt
SSL key file ................... /etc/zm/apache2/ssl/zoneminder.key

console_logs ................... false
Read alarm cause ............... true
Tag alarm event id ............. false
Use custom notification sound . false

Hook ........................... '/usr/bin/person_detect_wrapper.sh'
Use Hook Description........... true
```

4.8.2 What is the hook section ?

The `hook` section allows you to invoke a custom script when an alarm is triggered by ZM.

`hook_script` points to the script that is invoked when an alarm occurs

If the script returns success (exit value of 0) then the notification server will send out an alarm notification. If not, it will not send a notification to its listeners. This is useful to implement any custom logic you may want to perform that decides whether this event is worth sending a notification for.

Related to `hook` we also have a `hook_description` attribute. When set to 1, the text returned by the hook script will overwrite the alarm text that is notified.

We also have a `skip_monitors` attribute. This is a comma separated list of monitors. When alarms occur in those monitors, hooks will not be called and the ES will directly send out notifications (if enabled in ES). This is useful when you don't want to invoke hooks for certain monitors as they may be expensive (especially if you are doing object detection)

Finally, `keep_frame_match_type` is really used when you enable "bestmatch". It prefixes an `[a]` or `[s]` to tell you if object detection succeeded in the alarmed or snapshot frame.

Here is an example: (Note: just an example, please don't ask me for support for person detection)

- You will find a sample `detect_wrapper.sh` hook in the `hook` directory. This script is invoked by the notification server when an event occurs.
- This script in turn invokes a python OpenCV based script that grabs an image with maximum score from the current event so far and runs a fast person detection routine.

- It returns the value "person detected" if a person is found and none if not
- The wrapper script then checks for this value and exits with either 0 (send alarm) or 1 (don't send alarm)
- the notification server then sends out a ": person detected" notification to the clients listening

Those who want to know more: - Read the detailed notes here - Read this for an explanation of how this works

4.9 Troubleshooting common situations

4.9.1 LetsEncrypt certificates cannot be found when running as a web user

When the notification server is run in web user mode (example `sudo -u www-data`), the event notification server complains that it cannot find the certificate. The error is something like this:

```
zmeventnotification[10090].ERR [main:547] [Failed starting server: SSL_cert_file /etc/
↪letsencrypt/live/mysite.net-0001/fullchain.pem does not exist at /usr/share/perl5/
↪vendor_perl/IO/Socket/SSL.pm line 402.]
```

The problem is read permissions, starting at the root level. Typically doing `chown -R www-data:www-data /etc/letsencrypt` solves this issue

4.9.2 Picture notifications don't show images

Starting v2.0, I support images in alarms. However, there are several conditions to be met:

- You can't use self signed certs
- The IP/hostname needs to be publicly accessible (Apple/Google servers render the image)
- You need patches to ZM unless you are using a package that is later than Oct 11, 2018. Please read the notes in the INI file
- A good way to isolate if its a URL problem or something else is replace the `picture_url` with a knows HTTPS url like this

Before you report issues, please make sure you have been diligent in testing - Try with a public URL as indicated above. This is important. - In your issue, post debug logs of zmeventnotification so I can see what message it is sending

4.9.3 Secure mode just doesn't work (WSS) - WS works

Try to put in your event server IP in the `address` token in `[network]` section of `zmeventnotification.ini`

4.9.4 I'm not receiving push notifications in zmNinja

This almost always happens when zmNinja is not able to reach the server. Before you contact me, please perform the following steps and send me the output:

1. Stop the event server. `sudo zmdc.pl stop zmeventnotification.pl`
2. Do a `ps -aef | grep zmevent` and make sure no stale processes are running
3. Edit your `/etc/zm/zmeventnotification.ini` and make sure `console_logs = yes` to get console debug logs

4. Run the server manually by doing `sudo -u www-data /usr/bin/zmeventnotification.pl` (replace with `www-data` with `apache` depending on your OS)
5. You should now see logs on the commandline like so that shows the server is running:

```
018-12-20,08:31:32 About to start listening to socket
12/20/2018 08:31:32.606198 zmeventnotification[12460].INF [main:582] [About to start␣
↪listening to socket]
2018-12-20,08:31:32 Secure WS(WSS) is enabled...
12/20/2018 08:31:32.656834 zmeventnotification[12460].INF [main:582] [Secure WS(WSS)␣
↪is enabled...]
2018-12-20,08:31:32 Web Socket Event Server listening on port 9000
12/20/2018 08:31:32.696406 zmeventnotification[12460].INF [main:582] [Web Socket␣
↪Event Server listening on port 9000]
```

6. Now start zmNinja. You should see event server logs like this:

```
2018-12-20,08:32:43 Raw incoming message: {"event":"push","data":{"type":"token",
↪"platform":"ios","token":"cVuLzCBsEn4:APA91bHYuO3hVJqTIMsm0IRNQEYAUa<deleted>
↪GYBwNdwRfKyZV0","monlist":"1,2,4,5,6,7,11","intlist":"45,60,0,0,0,45,45","state":
↪"enabled"}}
```

If you don't see these logs on the event server, zmNinja is not able to connect to the event server. This may be because of several reasons: a) Your event server IP/DNS is not reachable from your phone b) If you are using SSL, your certificates are invalid (try disabling SSL first - both on the event server and on zmNinja) c) Your zmNinja configuration is wrong (the most common error I see is the server has SSL disabled, but zmNinja is configured to use `wss://` instead of `ws://`)

7. Assuming the above worked, go to zmNinja logs in the app. Somewhere in the logs, you should see a line similar to:

```
Dec 20, 2018 05:50:41 AM DEBUG Real-time event: {"type":"","version":"2.4","status":
↪"Success","reason":"","event":"auth"}
```

This indicates that the event server successfully authenticated the app. If you see step 6 work but not step 7, you might have provided incorrect credentials (and in that case, you'll see an error message)

8. Finally, after all of the above succeeds, do a `cat /var/lib/zmeventnotification/push/tokens.txt` to make sure the device token that zmNinja sent is stored (desktop apps don't have a device token). If you are using zmNinja on a mobile app, and you don't see an entry in `tokens.txt` you have a problem. Debug.
9. *Always* send me logs of both zmNinja and zmeventnotification - I need them to understand what is going on. Don't send me one line. You may think you are sending what is relevant, but you are not. One line logs are mostly useless.
10. Some other notes:

- If you are not using machine learning hooks, make sure you comment out the `hook_script` line in `[hook]`. If you have not setup the scripts correctly, if will fail and not send a push.
- If you don't see an entry in `tokens.txt` (typically in `/var/lib/zmeventnotification/push`) then your phone is not registered to get push. Kill zmNinja, start the app, make sure the event server receives the registration and check `tokens.txt`
- Sometimes, Google's FCM server goes down, or Apple's APNS server goes down for a while. Things automagically work in 24 hrs.
- Kill the app. Then empty the contents of `tokens.txt` in the event server (don't delete it). Then restart the event server. Start the app again. If you don't see a new registration token, you have a connection problem
- I'd strongly recommend you run the event server in "manual mode" and stop daemon mode while debugging.

4.9.5 I'm getting multiple notifications for the same event

99.9% of times, its because you have multiple copies of the eventserver running and you don't know it. Maybe you were manually testing it, and forgot to quit it and terminated the window. Do `sudo zmdc.pl stop zmeventnotification.pl` and then `ps -aef | grep zme`, kill everything, and start again. Monitor the logs to see how many times a message is sent out.

The other 0.1% is at times Google's FCM servers send out multiple notifications. Why? I don't know. But it sorts itself out very quickly, and if you think this must be the reason, I'll wager that you are actually in the 99.9% lot and haven't checked properly.

4.9.6 The server runs fine when manually executed, but fails when run in daemon mode (started by zmdc.pl)

- Make sure the file where you store tokens (`/var/lib/zmeventnotification/push/tokens.txt or whatever you have used`) is not RW Root only. It needs to be RW `www-data` for Ubuntu/Debian or `apache` for Fedora/CentOS. You also need to make sure the directory is accessible. Something like `chown -R www-data:www-data /var/lib/zmeventnotification/push`
- Make sure your certificates are readable by `www-data` for Ubuntu/Debian, or `apache` for Fedora/CentOS (thanks to @jagee).
- Make sure the *path* to the certificates are readable by `www-data` for Ubuntu/Debian, or `apache` for Fedora/CentOS

4.9.7 When you run zmeventnotifiation.pl manually, you get an error saying 'port already in use' or 'cannot bind to port' or something like that

The chances are very high that you have another copy of `zmeventnotification.pl` running. You might have run it in daemon mode. Try `sudo zmdc.pl stop zmeventnotification.pl`. Also do `ps -aef | grep zmeventnotification` to check if another copy is not running and if you do find one running, you'll have to kill it before you can start it from command line again.

4.9.8 Running hooks manually detects the objects I want but fails to detect via ES (daemon mode)

There may be multiple reasons, but a common one is of timing. When the ES invokes the hook, it is invoked almost immediately upon event detection. In some cases, ZoneMinder still has not had time to create an alarmed frame, or the right snapshot frame. So what happens is that when the ES invokes the hook, it runs detection on a different image from the one you run later when invoked manually. Try adding a `wait = 5` to `objectconfig.ini` to that monitor section and see if it helps

4.9.9 Great Krypton! I just upgraded ZoneMinder and I'm not getting push anymore!

Make sure your eventserver is running: `sudo zmdc.pl status zmeventnotification.pl`

4.10 How do I disable secure (WSS) mode?

As it turns out many folks run ZM inside the LAN only and don't want to deal with certificates. Fair enough. For that situation, edit zmeventnotification.pl and use `enable = 0` in the `[ssl]` section of the configuration file. **Remember to ensure that your EventServer URL in zmNinja does NOT use wss too - change it to ws**

4.11 Debugging and reporting problems

STOP. Before you shoot me an email, **please** make sure you have read the *common problems* and have followed *every step* of the *install guide* and in sequence. I can't emphasize how important it is.

There could be several reasons why you may not be receiving notifications:

- Your event server is not running
- Your app is not able to reach the server
- You have enabled SSL but the certificate is invalid
- The event server is rejecting the connections

Here is how to debug and report:

- Enable Debug logs in zmNinja (Setting->Developer Options->Enable Debug Log)
- SSH into your zoneminder server
- Stop the zmeventnotification doing `sudo zmdc.pl stop zmeventnotification.pl`
- Make sure there are no stale processes running of zmeventnotification by doing `ps -aef | grep zmeventnotification` and making sure it doesn't show existing processes (ignore the one that says `grep <something>`)
- Make sure ES debug logs are on. - Enable ZM debug logs for both ES (and hooks if you use them) as described in *Logging*. Note that ES debug logs are different from hooks debug logs. You need to enable both if you use them.
- Start a terminal and start zmeventnotification manually from command line like so `sudo -u www-data /usr/bin/zmeventnotification.pl`
- Start another terminal and tail logs like so `tail -f /var/log/zm/zmeventnotification.log /var/log/zm/zmesdetect_m*.log`. If you are NOT using hooks, simply do `tail -f /var/log/zm/zmeventnotification.log`
- Make sure you see logs like this in the logs window like so: (this example shows logs from both ES and hooks)

```
pp@homeserver:~/fiddle/zmeventnotification$ tail -f /var/log/zm/zmeventnotification.
↪log /var/log/zm/zmesdetect_m*.log
==> /var/log/zm/zmeventnotification.log <==
10/06/2019 06:48:29.200008 zmeventnotification[13694].INF [main:557] [Invoking hook:'/
↪usr/bin/detect_wrapper.sh' 33989 2 "DoorBell" " front" "/var/cache/zoneminder/
↪events/2/2019-10-06/33989"]
10/06/2019 06:48:34.013490 zmeventnotification[29913].INF [main:557] [New event 33990␣
↪reported for Monitor:10 (Name:FrontLawn)  front steps]
10/06/2019 06:48:34.020958 zmeventnotification[13728].INF [main:557] [Forking␣
↪process:13728 to handle 1 alarms]
10/06/2019 06:48:34.021347 zmeventnotification[13728].INF [main:557] [processAlarms:␣
↪EID:33990 Monitor:FrontLawn (id):10 cause: front steps]
10/06/2019 06:48:34.237147 zmeventnotification[13728].INF [main:557] [Adding event␣
↪path:/var/cache/zoneminder/events/10/2019-10-06/33990 to hook for image storage]
```

(continues on next page)

(continued from previous page)

```
10/06/2019 06:48:34.237418 zmeventnotification[13728].INF [main:557] [Invoking hook:'/
↪usr/bin/detect_wrapper.sh' 33990 10 "FrontLawn" " front steps" "/var/cache/
↪zoneminder/events/10/2019-10-06/33990"]
10/06/2019 06:48:46.529693 zmeventnotification[13728].INF [main:557] [For Monitor:10␣
↪event:33990, hook script returned with text: exit:1]
10/06/2019 06:48:46.529896 zmeventnotification[13728].INF [main:557] [Ending␣
↪process:13728 to handle alarms]
10/06/2019 06:48:47.640414 zmeventnotification[13694].INF [main:557] [For Monitor:2␣
↪event:33989, hook script returned with text: exit:1]
10/06/2019 06:48:47.640668 zmeventnotification[13694].INF [main:557] [Ending␣
↪process:13694 to handle alarms]

==> /var/log/zm/zmesdetect_m10.log <==
10/06/19 06:48:42 zmesdetect_m10[13732] DBG detect.py:344 [No match found in /var/lib/
↪zmeventnotification/images/33990-alarm.jpg using model:yolo]
10/06/19 06:48:42 zmesdetect_m10[13732] DBG detect.py:189 [Using model: yolo with /
↪var/lib/zmeventnotification/images/33990-snapshot.jpg]
10/06/19 06:48:46 zmesdetect_m10[13732] DBG detect.py:194 [|--> model:yolo detection␣
↪took: 3.541227s]
```

- If you are debugging problems with receiving push notifications on zmNinja mobile, then replicate the following scenario:
  - Run the event server in manual mode as described above
  - Kill zmNinja
  - Start zmNinja
  - At this point, in the `zmeventnotification` logs you should

    registration messages (refer to logs example above). If you don't you've either not configured zmNinja to use the eventserver, or it can't reach the eventserver (very common problem)
  - Next up, make sure you are not running zmNinja in the foreground

    (move it to background or kill it). When zmNinja is in the foreground, it uses websockets to get notifications
  - Force an alarm like I described above. If you don't see logs in

    `zmeventnotification` saying "Sending notification over FCM" then the eventserver, for some reason, does not have your app token. Inspect `tokens.txt` (typically in `/etc/zm/`) to make sure an entry for your phone exists
  - If you see that message, but your mobile phone is not receiving a

    push notification: - Make sure you haven't disable push notifications on your phone (lots

      of people do this by mistake and wonder why)
    - Make sure you haven't muted notifications (again, lots of people...)
    - Sometimes, the push servers of Apple and Google stop forwarding

      messages for a day or two. I have no idea why. Give it a day or two?
    - Open up zmNinja, go right to logs and send it to me
    - If you have issues, please send me a copy of your zmeventnotification

      logs generated above from Terminal-Log, as well as zmNinja debug logs

4.12 Brickbats

Why not just supply the username and password in the URL as a resource? It's over TLS

Yup its encrypted but it may show up in the history of a browser you tried it from (if you are using a browser) Plus it may get passed along from the source when accessing another URL via the Referral header

So it's encrypted, but passing password is a bad idea. Why not some token?

- Well, now that ZM supports login tokens (starting 1.33), I'll get to supporting it.

Why WSS and not WS?

Not secure. Easy to snoop. Updated: As of 0.6, I've also added a non secure version - use `enable = 0` in the `[ssl]` section of the configuration file. As it turns out many folks don't expose ZM to the WAN and for that, I guess WS instead of WSS is ok.

Why ZM auth in addition to WSS?

WSS offers encryption. We also want to make sure connections are authorized. Reusing ZM authentication credentials is the easiest. You can change it to some other credential match (modify `validateZM` function)

CHAPTER 5

For Developers writing their own consumers

5.1 How do I talk to it?

- `{"JSON":"everywhere"}`
- Your client sends messages (authentication) over JSON
- The server sends auth success/failure over JSON back at you
- New events are reported as JSON objects as well
- By default the notification server runs on port 9000 (unless you change it)
- You need to open a secure web socket connection to that port from your client/consumer
- You then need to provide your authentication credentials (ZoneMinder username/password) within 20 seconds of opening the connection
- If you provide an incorrect authentication or no authentication, the server will close your connection
- As of today, there are 3 categories of message types your client (zmNinja or your own) can exchange with the server (event notification server)

1. auth (from client to server)
2. control (from client to server)
3. push (only applicable for zmNinja)
4. alarm (from server to client)

5.1.1 Authentication messages

To connect with the server you need to send the following JSON object (replace username/password) Note this payload is NOT encrypted. If you are not using SSL, it will be sent in clear.

Authentication messages can be sent multiple times. It is necessary that you send the first one within 20 seconds of opening a connection or the server will terminate your connection.

Client –> Server:

```
{"event":"auth","data":{"user":"<username>","password":"<password>"}}
```

Server –> Client: The server will send back one of the following responses

Authentication successful:

```
{"event":"auth", "type":"", "version":"0.2","status":"Success","reason":""}
```

Note that it also sends its version number for convenience

Incorrect credentials:

```
{"event":"auth", "type":"", "status":"Fail","reason":"BADAUTH"}
```

No authentication received in time limit:

```
{"event":"auth","type":"", "status":"Fail","reason":"NOAUTH"}
```

5.1.2 Control messages

Control messages manage the nature of notifications received/sent. As of today, Clients send control messages to the Server. In future this may be bi-directional

Control message to restrict monitor IDs for events as well as interval durations for reporting

A client can send a control message to restrict which monitor IDs it is interested in. When received, the server will only send it alarms for those specific monitor IDs. You can also specify the reporting interval for events.

Client–>Server:

```
{"event":"control","data":{"type":"filter","monlist":"1,2,4,5,6", "intlist":"0,0,3600,
↪60,0"}}
```

In this example, a client has requested to be notified of events only from monitor IDs 1,2,4,5 and 6 Furthermore it wants to be notified for each alarm for monitors 1,2,6. For monitor 4, it wants to be notified only if the time difference between the previous and current event is 1 hour or more (3600 seconds) while for monitor 5, it wants the time difference between the previous and current event to be 1 minute (60 seconds)

There is no response for this request, unless the payload did not have either monlist or intlist.

No monitorlist received:

```
{"event":"control","type":"filter", "status":"Fail","reason":"NOMONITORLIST"}
```

No interval received:

```
{"event":"control","type":"filter", "status":"Fail","reason":"NOINTERVALLIST"}
```

Note that if you don't want to specify intervals, send it a interval list comprising of comma separated 0's, one for each monitor in monitor list.

Control message to get Event Server version

A client can send a control message to request Event Server version

Client–>Server:

```
{"event":"control","data":{"type":"version"}}
```

Server–>Client:

```
{"event":"control", "type:":"version", "version":"0.2","status":"Success","reason":""}
```

5.1.3 Alarm notifications

Alarms are events sent from the Server to the Client

Server–>Client: Sample payload of 2 events being reported:

```
{"event":"alarm", "type":"", "status":"Success", "events":[{"EventId":"5060","Name":
↪"Garage","MonitorId":"1"},{"EventId":"5061","MonitorId":"5","Name":"Unfinished"}]}
```

5.1.4 Push Notifications (for both iOS and Android)

To make Push Notifications work, please make sure you read the section on enabling Push for the event server.

Concepts of Push and why it is only for zmNinja

Both Apple and Google ensure that a "trusted" application server can send push notifications to a specific app running in a device. If they did not require this, anyone could spam apps with messages. So in other words, a "Push" will be routed from a specific server to a specific app. Starting Jan 2018, I am hosting my trusted push server on Google's Firebase cloud. This eliminates the need for me to run my own server.

Registering Push token with the server

Client–>Server:

Registering an iOS device:

```
{"event":"push","data":{"type":"token","platform":"ios","token":"<device tokenid here>
↪", "state":"enabled"}}
```

Here is an example of registering an Android device:

```
{"event":"push","data":{"type":"token","platform":"android","token":"<device tokenid␣
↪here>", "state":"enabled"}}
```

For devices capable of receiving push notifications, but want to stop receiving push notifications over APNS/GCM and have it delivered over websockets instead, set the state to disabled

For example: Here is an example of registering an Android device, which disables push notifications over GCM:

```
{"event":"push","data":{"type":"token","platform":"android","token":"<device tokenid␣
↪here>", "state":"disabled"}}
```

What happens here is if there is a new event to report, the Event Server will send it over websockets. This means if the app is running (foreground or background in Android, foreground in iOS) it will receive this notification over the open websocket. Note that in iOS this means you won't receive notifications when the app is not running in the foreground. We went over why, remember?

Server–>Client: If its successful, there is no response. However, if Push is disabled it will send back

```
{"event":"push", "type":"", "status":"Fail", "reason": "PUSHDISABLED"}
```

Badge reset

Only applies to iOS. Android push notifications don't have a concept of badge notifications, as it turns out.

In push notifications, the server owns the responsibility for badge count (unlike local notifications). So a client can request the server to reset its badge count so the next push notification starts from the value provided.

Client–>Server:

```
{"event":"push", "data":{"type":"badge", "badge":"0"}}
```

In this example, the client requests the server to reset the badge count to 0. Note that you can use any other number. The next time the server sends a push via APNS, it will use this value. 0 makes the badge go away.

5.1.5 Testing from command line

If you are writing your own consumer/client it helps to test the event server commands from command line. The event server uses Secure/WebSockers so you can't just HTTP to it using tools like `curl`. You'll need to use a websocket client. While there are examples on the net on how to use `curl` for websockets, I've found it much simpler to use wscat like so:

```
wscat -c wss://myzmeventnotification.domain:9000 -n
connected (press CTRL+C to quit)
> {"event":"auth","data":{"user":"admin","password":"xxxx"}}
< {"reason":"","status":"Success","type":"","event":"auth","version":"0.93"}
```

In the example above, I used `wscat` to connect to my event server and then sent it a JSON login message which it accepted and acknowledged.

CHAPTER 6

Writing your own detection plugin

It is super simple to create your own plugin.

Your plugin needs to be created as a class that detect.py can import.

The best file to start from is hog.py (simplest)

In general, your plugin needs to follow the following structure:

```
class YourPluginName:

        # input
        def __init__ (self, param1, ...,paramN):

        # expected output
        # none


        def get_classes(self):
                # classes is a list of objects your plugin detects
                # example ['item1', 'item2', 'item3']

        # expected output
        # list of class names


        def detect (self, image):
                # image passed will be the image to detect
                # format is that returned by cv2.imread

        # expected output
        # list of (rects, labels, confidence)
        # where:
        #       rects = list of (x1,y1,x2,y2) bounding box of object detected
        #               x1,y1 = left, top coordinates
        #               x2,y2 = right, bottom coordinates
```

(continues on next page)

(continued from previous page)

```
        #       labels = list of object names
        #       conidence = string number between 1 and 0 for confidence
```

Github Repository

Installation of the Event Server (ES) How to install the Event Notification Server

Machine Learning Hooks How to configure the machine learning hooks *after* you install the Event Server

Breaking Changes Breaking changes. Always read this if you are upgrading (for example, lots changed with 3.x and 3.2)

FAQ FAQ covering common scenarios/issues

For Developers writing their own consumers If you want to use the Event Notification Server to make your own app/client

Writing your own detection plugin If you want to add your own algoritm to the machine learning hooks (Needs to be expanded)

zmNinja Documentation Documentation for zmNinja