

# FM Support

---

## Index

---

1. [Setup sherpas](#)
2. [Access Config Editor](#)
3. [Send updates to master fm](#)
4. [Setup auto parking feature](#)
5. [Setup battery swap trips](#)
6. [Setup optimal dispatch config](#)
7. [Generate api keys](#)
8. [Setup plugin conveyor](#)
9. [Setup plugin summon buttons](#)
10. [Flash summon buttons](#)
11. [Setup plugin IES](#)
12. [Restart FM](#)
13. [Debug FM](#)
14. [Access Postgres DB](#)
15. [Some docker commands](#)
16. [Update FM with master FM credentials](#)
17. [Setup optimal dispatch config](#)
18. [Clean up disk space in FM server](#)
19. [Create self signed certs for FM](#)
20. [Use trusted CA for FM](#)
21. [Run FM simulator](#)
22. [Where to get the logs?](#)
23. [Forgot password for frontend\\_user: admin ?](#)
24. [Add new fleet](#)
25. [Switch between multiple maps corresponding to the same fleet](#)
26. [Simulate summon\\_button/conveyor with postman](#)
27. [Setup run on host service](#)

## Setup sherpas

1. Copy fm cert file(fm\_rev\_proxy\_cert.pem) from <static\_dir>/certs to sherpa's /opt/ati/config directory
2. Add this patch to /opt/ati/config/config.toml in the mule

```
[fleet]
api_key = <api_key>
chassis_number = <chassis_number>
ip="<fm_ip_address>"
port="443"
fm_cert_file="/app/config/fm_rev_proxy_cert.pem"
```

## Access Config Editor

1. The config editor should be accessible at <https://<fm\_ip>/config\_editor>
2. Credentials for login in to config editor can be obtained from docker-compose\_v<fm\_version>.yaml file (would be available in the static folder on the FM server).

```
ME_CONFIG_BASICAUTH_USERNAME: ****
ME_CONFIG_BASICAUTH_PASSWORD: ****
```

## Send updates to master fm

1. Use config editor, select the database fm\_config, select the collection master\_fm, click on the document to edit it
2. If sanajaya.atimotors.com is accessible, you don't have to follow the remaining steps(steps 4-6) a. Change the below mentioned parameters in the document, save the same. And [Restart FM](#). The remaining steps need not be followed

```
api_key: '<api_key generated for the customer>'
send_updates: true
```

3. If sanajaya.atimotors.com is not accessible but you are be able to ssh to FM server via another machine which has access to sanjaya.atimotors.com, then a reverse tunnel can be setup to access sanjaya.atimotors.com
4. To setup reverse tunnel, get mfm\_rev\_tunnel.tar from the downloads section on the dashboard and copy the same to the machine which has access sanjaya.atimotors.com(pingable) and has ssh access to the FM server, do the following

```
tar -xvf mfm_rev_tunnel.tar ## This is for Linux, something similar has to
be done for other os
cd mfm_rev_tunnel
bash mfm_rev_tunnel.sh <user@fm_server_ip> <client_name>
```

5. ssh into the FM server, set GatewayPorts to yes in /etc/ssh/sshd\_config (This will require sudo access) and restart the ssh service, reboot the machine for the this to take effect.

```
sudo systemctl restart ssh
```

6. Follow step 2, Change the below mentioned parameters in the document, save the same

```
mfm_ip: '<fm_server_ip>'
mfm_ip: '9010'
http_scheme: 'http'
ws_scheme: 'ws'
api_key: '<api_key generated for the customer>'
send_updates: true
```

### 7. [Restart FM](#)

## Setup auto parking feature

1. Use the config editor, select the database fm\_config, select the collection conditional\_trips, click on the document to edit it
2. Edit auto park params, save the same

```
auto_park: {
  book: true,
  threshold: 600, ### Threshold in seconds after which sherpa should be
sent to parking station if ## found idle
  priority: 1 ## trip priority to be given to auto park trips
}
```

### 3. [Restart FM](#)

## Setup battery swap trips

1. Use the config editor, select the database fm\_config, select the collection conditional\_trips, click on the document to edit it
2. Edit battery\_swap params, save the same

```
battery_swap: {
  book: true,
  max_trips: 2, ### max number sherpas that can do battery swap trips
simultaneously
  threshold: 15, ### Threshold battery level
  priority: 10 ### trip priority to be given to battery_swap trips
}
```

### 3. [Restart FM](#)

## Setup optimal dispatch config

1. Optimal dispatch logic tries to allocate the pending trips with the best sherpa available. Choice of best sherpa is made with the paramter \$Z\$

$$Z = (\eta)^a / (\text{priority})^b$$

$$\text{priority} = p_1 / p_2$$

where,  
 eta - expected time of arrival computed for the sherpa to reach the first station of the trip booked,  
 priority - measure of how long a trip has been pending,  
 p1 - Time since booking of current trip,  
 p2 - Minimum of time since booking across all the pending trips,  
 a - eta power factor ,  $0 < a < 1$ ,  
 b - priority power factor ,  $0 < b < 1$ ,

2. Use the config editor, select the database fm\_config, select the collection optimal\_dispatch, click on the document to edit it
3. **Maximise number of trips done:** To get maximum number of trips done in a given time frame eta\_power\_factor can be set to 1, priority\_power\_factor can be set to 0. This will make the optimal dispatch logic to lean towards trips that can be started faster. The trip booking order will not be followed.

```
method: 'hungarian',
prioritise_waiting_stations: true,
eta_power_factor: 0.9999 ## 0.00001-0.99999,
priority_power_factor: 0.0001 ## 0.00001-0.99999,
max_trips_to_consider: 5,
```

4. **Fair scheduling:** To configure optimal dispatch logic to take trips in the order they were booked eta\_power\_factor can be set to 0, priority\_power\_factor can be set to 1.

```
method: 'hungarian',
prioritise_waiting_stations: true,
eta_power_factor: 0.0001 ## 0.00001-0.99999,
priority_power_factor: 0.9999 ## 0.00001-0.99999,
max_trips_to_consider: 5,
```

5. **Custom configuration:** There is no ideal combination of eta\_power\_factor, priority\_power\_factor. They should be chosen according to the frequency of trip bookings, route length between the stations to maximise the throughput.
6. For good takt time, eta power factor should be higher, for fair scheduling priority power factor should be set higher.
7. To reduce computation load due to optimal dispatch, max\_trips\_to\_consider has been set to 5. Optimal dispatch logic will consider only the first <max\_trips\_to\_consider> number of trips.

Default is set to 5. This can be increased to <number\_of\_sherpas per fleet> in case there are more than 5 sherpas

```
[optimal_dispatch]  
max_trips_to_consider=<number_of_sherpas per fleet>
```

## Generate api keys

1. To generate api key with hardware id (sherpa or other smart devices)

```
docker exec -it fleet_manager bash  
apihw <hardware_id>
```

2. To generate api key for n smart devices

```
docker exec -it fleet_manager bash  
apind <number of devices>
```

## Setup plugin conveyor

1. Use the config editor, select the database plugin\_config, select the collection plugin\_conveyor, click on the document to edit it
2. Set activate\_plugin to true

```
activate_plugin: true
```

3. Modify max\_tote\_per\_trip if needed. This is the maximum number of totes that the sherpa can carry per trip
4. [Restart FM](#)
5. Conveyors need to be flashed along with fm\_server\_ip, cert\_file and the right api key.

## Setup plugin summon buttons

1. Use the config editor, select the database plugin\_config, select the collection plugin\_summon\_button, click on the document to edit it
2. Set activate\_plugin to true

```
activate_plugin: true
```

3. [Restart FM](#)
4. [Flash summon buttons](#)
5. What different colours in summon button LED mean?

```
"blinking red" - Not connected to FM  
"white" - Connected to FM but no unfinished trips  
"rotating yellow" - Trip booked on button press, waiting for the trip to start  
"blinking green" - Waiting at station(Trip enroute)  
"rotating green" - Trip enroute
```

## Flash summon buttons

1. Connect summon button to your laptop via USB to flash firmware
2. Download fm\_cert\_file to your laptop. It should be available in FM server  
`<fm_static_dir>/certs/fm_rev_proxy_cert.pem`
3. Download FlashTool\_SB.tar from downloads section on the dashboard and run the same

```
tar -xvf FlashTool_SB.tar (This is for Linux, use similar commands to extract files in other os)  
cd FlashTool_SB  
sudo bash ./install.sh  
sudo bash ./flashtool_8mb.sh
```

4. Upon flashing, reconnect the summon button usb.
5. Press and hold the summon button until LED on the summon button turns blue, and connect to summon button via wifi. For instance you would see something like Summon\_192049 in the available/known wifi networks. Upon successful connection to summon button wifi, you will see a summon button UI. The password for connecting to summon\_button network : 12345678
6. Press configure WiFi, choose the preferred network and add the wifi password for the same, save it. Wait until summon button led turns from yellow to blinking red .
7. Repeat step 4 and continue with the steps below
8. Now press configure device, add FM plugin url to HOST. PLUGIN\_PORT by default would be 443.

```
wss://<FM_IP>:<PLUGIN_PORT>/plugin/ws/api/v1/summon_button
```

9. Set wifi type: WPA/WPA2
10. Set Mode to WiFi-Only

11. Set HEARTBEAT to disable

12. Set APIKEY and save.

```
X-API-Key:<api_key_generated_with_summon_button_id>
```

13. Upload fm\_rev\_proxy\_cert.pem

14. Press restart device in summon button UI.

## Setup plugin IES

Will be added soon

## Restart FM

1. If there were code changes/updates, run the following command from the static directory

```
docker-compose -p fm -f docker_compose_v<fm_version> down  
docker-compose -p fm -f docker_compose_v<fm_version> up
```

2. If there were only config changes, restart FM from the dashboard maintenance page

## Debug FM

1. Check if there were any queue build ups. The output would show queue build ups if any.

```
docker exec -it fleet_manager bash  
inspect ## This would list all historical queue build ups  
rqi ## This would show if there is any queue build up at present
```

2. Check for occurrences of rq errors (rqe) in fleet\_manager.log, the output might lead to the issue

```
rqe
```

3. If you are unable to login to FM, Check the docker logs - this should be run outside docker. There might be some errors in the init scripts.

```
docker logs fleet_manager  
docker logs fleet_db  
# Also check logs/fm.out inside fleet manager container
```

## Access Postgres DB

1. Get a dump and copy it to the host machine, db\_names can be ati\_fleet, plugin\_conveyor, plugin\_summon\_button, plugin\_ies etc

```
docker exec -it fleet_db bash
pg_dump -U postgres <db_name> > /home/<db_name>.dump
exit
docker cp fleet_db:/home/<db_name>.dump .
```

2. Access db inside FM

```
docker exec -it fleet_manager bash
psql $FM_DATABASE_URI
```

3. Access db inside fm\_plugins

```
docker exec -it fm_plugins bash
psql $PLUGIN_DATABASE_URI
```

## Some docker commands

1. Useful docker commands (run outside container)

```
docker stats
docker system df
docker image prune
docker rmi <image_name>
docker stop <container_name>
docker rm <container_name>
```

## Update FM with master FM credentials

1. Run the script mentioned below, you would need login credentials to sanjaya.atimotors.com to complete the update

```
bash ./scripts/update_with_master_fm_cred.sh
```

2. There will be prompts to help you pull the images



```
Sanjaya Username: ### Enter master fm username ###  
Sanjaya Password: ### Enter master fm password ###  
FM version: ### Enter fm version like fm_dev, FM_v3.2 ###
```

### 3. Restart FM

## Clean up disk space in FM server

1. Set backup config in FM config editor appropriately
2. Use the config editor, select the database fm\_config, select the collection data\_backup, click on the document to edit it
3. Edit keep\_size\_mb, FM will try to restrict the data inside static/data\_backup folder to keep\_size\_mb only. The contents in the data backup folder will sorted and deleted based on their time of creation, older data will be deleted first. The default is set to 1000MB
4. Use can also set prune\_unused\_images to true or false based on whether you want to clean up old docker images. Set prune\_images\_used\_until\_h accordingly, all the images that were unused in the last prune\_images\_used\_until\_h hours will be deleted.

## Create self signed certs for FM

**This step needs to be done only if the certs were not setup already**

1. Run the following command to get all the ips of the server

```
ifconfig | grep inet | awk '{print $2}' | egrep '[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+'
```

2. Create ssl certs (This has to be done only for the first time). ip\_1, ip\_2...ip\_n etc are ips of the FM server obtained in the previous step.

```
docker exec -it fleet_manager bash  
create_certs "127.0.0.1,<ip_1>,<ip2>,...,<ip_n>"
```

3. Restart FM (using docker-compose command)

## Use trusted CA for FM

1. Create certificate bundle. The attached link can help you in creating cert bundle <https://www.ssldragon.com/blog/what-is-a-ca-bundle/>
2. Rename the certificate bundle as fm\_rev\_proxy\_cert.pem
3. Rename key file as fm\_rev\_proxy\_key.pem

4. Copy the renamed files(key, cert) to fm static/certs directory
5. [Restart FM](#) (using docker-compose command)
6. Set fm\_cert\_file in sherpa config.toml as below

```
fm_cert_file="/etc/ssl/certs/ca-certificates.crt"
```

## RUN FM Simulator

---

**FM simulator creates proxy for all the sherpa, Make sure real/physical sherpas are not connected to FM, Config changes need FM restart to take effect**

1. Use the config editor, select the database fm\_config, select the collection simulator, click on the document to edit it
2. Set simulate to true.

```
simulate: true
```

3. If you need to simulate visa/traffic gates as well, set visa\_handling to true. **There is a caveat, FM simulator would only request/release/simulate transit type visas, so full fledged visa simulation cannot be performed with fm simulator**

```
visa_handling: true
```

4. You can configure pre-defined routes, set book\_trips to true. This can be used if there are so many trips to be booked. Route names don't not matter, but the stations list(route) must be valid. In the example below route1 is a schedule trip, route2 is a normal trip. In the route1 definition, 10 - trip frequency(in seconds), the following timestamps represent start\_time and end\_time respectively.

```
### OPTIONAL ###
book_trips: true,
routes : {
  "route1": [ ["Station A", "Station B"], ["10", "2023-05-31 15:00:00",
"2023-05-31 16:00:00"]],
  "route2": [ ["Station B", "Station A"], ["-1", "", ""]]
}
```

5. By default, sherpas will start at random stations. But if you want sherpas to start at specific stations use initialize\_sherpas\_at parameter. Follow the below example. Make sure the key, value match

sherpa name, station name exactly

```
### OPTIONAL ###
initialize_sherpas_at: {
  "sample_sherpa" : "Station A"
}
```

#### 6. [Restart FM](#)

## Where to get the logs?

---

We store logs corresponding to two containers (fleet\_manager, fm\_plugins)

### 1. Get fleet\_manager logs

```
docker exec -it fleet_manager bash
cd logs
```

### 2. Get fm\_plugins logs

```
docker exec -it fm_plugins bash
cd plugin_logs
```

### 3. Description of some important log files

- a. uvicorn.log - Any error related to fastapi app(endpoints) will be present in uvicorn.log
- b. fleet\_manager.log - Any success/error in handlers/rq will be recorded in fleet\_manager.log
- c. visa.log - All the visa assignments/rejects will be present in visa.log
- d. plugin\_logs/plugin\_<plugin\_conveyor>/<plugin\_name>.log - would have logs corresponding to the plugin
- e. plugin\_logs/plugin\_main.log - Any error in fm\_plugins can be seen in plugin\_main.log

## Forgot password for frontend\_user: admin ?

- 1. Use the config editor, select the database frontend\_users, select the collection user\_details
- 2. There would be multiple documents corresponding to different users, delete the document with name as admin
- 3. [Restart FM](#)
- 4. On restart you should be able to login with the below credentials

```
username: admin  
password: 1234
```

## Add new fleet

### You would need ssh access to carry out this step

1. Create a folder in fleet\_manager server static directory if not already present

```
## this needs to be done on  
mkdir -p <fleet_manager_static_dir>/map
```

2. Copy the map files to the folder <fleet\_manager\_static\_dir>/map/.
3. Use configure page in fleet\_manager dashboard to add the fleet

## Switch between multiple maps corresponding to the same fleet

### These steps have to be carried out in the FM server

1. Create all\_maps/ folder. You name version as you like.

```
For example, if there are two versions of map for the same fleet  
mkdir -p <fleet_manager_static_dir>/all_maps/<version 1>  
mkdir -p <fleet_manager_static_dir>/all_maps/<version 2>
```

2. Copy different sets of map files to the folders created above
3. Press Maintenance button in the web page header
4. Use dashboard, switch to corresponding fleet and press update\_map button on the webpage header.  
A pop would help you to choose the required map version.
5. Press Start ops button in the web page header

## Simulate summon\_button/conveyor with postman

1. Create a new window for websocket connection in postman
2. Use the url given below based on plugin you are using

```
summon_button: ws://<fm_ip>:<plugin_port>/plugin/ws/api/v1/summon_button  
conveyor: ws://<fm_ip>:<plugin_port>/plugin/ws/api/v1/conveyor
```

3. Set api-key in headers

```
X-API-Key: <api_key>
```

4. Send a `tote_status` message on behalf of conveyor. Only the attributes `num_totes` matters. All other are info given for debugging

```
{"sensor_1": 0, "sensor_2": 0, "sensor_3": 0, "sensor_4": 0, "sensor_5": 0, "sensor_6": 0, "num_totes": 1, "num_totes_to_transfer": 0, "compact_time": 0, "last_compact_time": 420774, "last_egress_time": 420798, "last_egress_reading": 0, "gate_state": 0, "gate_ready_time": 3421937, "last_tick_time": 28964247, "type": "tote_status"}
```

5. Simulate `summon_button` press, send `button_pressed` message

```
{"type": "button_pressed"}
```

## Setup run on host service

**This will be used to run commands outside docker's context, setting up this will require sudo access**

1. Check the FM static dir, following files should exists

```
run_on_host.sh
run_on_host_updater.sh
install_run_on_host_service.sh
```

2. Give all perimissions to `install_run_on_host_service` script

```
sudo chmod ugo+rwX install_run_on_host_service.sh
```

3. Run `install_run_on_host_service.sh`

```
./install_run_on_host_service.sh
```

4. Ensure that the output shows that the `run_on_host` is successfully installed