

# 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 and hashed passwords](#)
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](#)

### 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](https://<fm_ip>/config_editor)>
2. Credentials for login to config editor can be obtained from docker-compose\_v<fm\_version>.yml file (would be available in the static folder on the FM server)

### Send updates to master fm

1. Check whether FM server has access to [sanjaya.atimotors.com](http://sanjaya.atimotors.com) by doing a ping

```
ping sanjaya.atimotors.com
```

2. Use config editor, select the database fm\_config, select the collection master\_fm, click on the document to edit it
3. If sanajaya.atimotors.com is accessible a. Change the below mentioned parameters in the document, save the same

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

4. If you are 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
5. To setup reverse tunnel, get mfm\_rev\_tunnel.tar from 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>
```

6. ssh into the FM server, set GatewayPorts to yes in /etc/ssh/sshd\_config (This will require sudo access) and restart the ssh service

```
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'
```

## 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,
    max_trips: 2, ### max number sherpas that can do auto parked trips
    simultaneously
    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 parameter \$Z\$

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

$\text{priority} = p_1 / p_2$

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: 1.0,
priority_power_factor: 0.0,
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: 0.0,
priority_power_factor: 1.0,
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 can be 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 and hashed passwords

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](#)

## Flash summon buttons

1. Connect summon button to your laptop via USB to flash firmware
2. 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
```

3. Upon flashing, reconnect the summon button usb.

4. 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.
5. 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 .
6. Repeat step 4 and continue with the steps below
7. Now press configure device, add FM plugin url to HOST. PLUGIN\_PORT by default would be 8002

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

8. Set wifi type: WPA/WPA2
9. Set Mode to WiFi-Only
10. Set HEARTBEAT to disable
11. Set APIKEY and save.

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

12. Press restart device in summon button UI.

## Setup plugin IES

Will be added soon

## Restart FM

1. 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
```

## Debug FM

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

```
docker exec -it fle et_manager bash  
inspect  
rqi
```

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
```

## 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

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