



MAGMA

Magma Install Guide

February 2019



Facebook has made an effort to ensure that this documentation is accurate but does not make any representation or warranty regarding accuracy or completeness. Facebook reserves the right to update and otherwise modify the information included in this document without notice. If you find information that is incorrect or incomplete, we would appreciate your comments and suggestions.

Table of Contents

Magma Install Guide

- Prerequisites..... 4
- Running Magma On a Laptop 4
- Registering the network and gateways 7
- Troubleshooting on a Virtual Machines..... 12
- Staging the eNodeB with TR-069 support 13
- Staging the eNodeB without TR-069 support 14
- Provisioning the Magma Gateway 14

Prerequisites

You need the following devices before you begin:

- Computer
- Wired internet connection
- eNodeB
- CPE such as smartphone or tablet
- SIM card with known key values

You need to have the following applications installed on your computer.

- Virtual box, version 5.2 or higher
- Vagrant, version 2 or higher
- Ansible, version 2.7 or higher

To provision the Magma Gateway you need to have access to the Magma Network Management System (NMS). Please use the Magma Provision Guide for reference.

Note: You should not use a VPN during installation as it might affect the connection between VMs.

Running Magma on a Laptop

Clone the magma repository from github

1. [Generate SSH key](#)
2. [Add the generated SSH key to the github account](#)
3. Clone the repository to the laptop through SSH with the following command:

```
user@user-mbp:~$ git clone git@github.com:facebookexternal/magma.git
```

Bring up Orchestrator, FEG, and Gateway Virtual machines

- On a laptop computer, bring up Orchestrator/Datastore, FeG, and Gateway VMs with the following commands

```
user@user-mbp:~/magma/orc8r/cloud$ vagrant up cloud datastore
user@user-mbp:~/magma/lte/gateway$ vagrant up magma
user@user-mbp:~/magma/feg/gateway$ vagrant up feg
```

- If the virtual machine provider cannot be found, add “—provider=virtualbox” flag to the end of each command to force Vagrant to use VirtualBox.

Orchestrator VM

- Build and start service in Orchestrator VM with the commands:

```
user@user-mbp:~/magma/orc8r/cloud$ vagrant ssh cloud
vagrant@magma-cloud:~/magma/orc8r/cloud$ make run
```

- Check that all Orchestrator services are running correctly with the command:

```
vagrant@magma-cloud:~$ sudo service magma@* status
```

- After a successful make run, some certificates will be generated on the laptop under magma/.cache/tests_certs. They will be used later to access the Swagger API hosted by the obsidian (the REST endpoint service) service in Orchestrator.
- If certificates are not there, they could be generated with the following commands:

```
vagrant@magma-cloud:~/magma/orc8r/cloud$ make create_admin_oper
```

- If certificates are there, but you faced some issue accessing the swagger API with the certificate, use the following command to make Orchestrator recognize the certificate:

```
vagrant@magma-cloud:~/magma/orc8r/cloud$ make restore_admin_operator
```

All Orchestrator services should now be compiled and running and the local certificate provisioned.

Gateway VM

- Build and start service in the Gateway VM with the commands:

```
user@user-mbp:~/magma/lte/gateway$ vagrant ssh magma
vagrant@magma-dev:~/magma/lte/gateway$ make run
```

- Check that all Gateway services are running correctly with the command:

```
vagrant@magma-dev:~$ sudo service magma@* status
```

- Get the Hardware ID of the gateway which will be used later with the commands:

```
vagrant@magma-dev:~$ magtivate
vagrant@magma-dev:~$ show_gateway_info.py
```

- On the host machine, use the following command to make the Gateway VM use local Orchestrator VM:

```
user@user-mbp:~/magma/lte/gateway$ fab use_local_cloud
```

All the access gateway services should now be compiled and running, and the access gateway should be attempting to connect to the orchestrator. The access gateway will start streaming configurations after the network and gateway are provisioned. See provisioning guide for additional details.

FeG VM

- Build and start service in the federation gateway (FeG) VM with the commands:

```
user@user-mbp:~/magma/feg/gateway$ vagrant ssh feg
vagrant@magma-dev:~/magma/feg/gateway$ make run
```

- Check that all FeG services are running correctly with the command:

```
vagrant@magma-feg-dev:~$ sudo service magma@* status
```

- Get the Hardware ID of the FeG which will be used later with the commands

```
vagrant@magma-feg-dev:~$ magtivate
vagrant@magma-feg-dev:~$ show_gateway_info.py
```

All the federation gateway features should now be running and the federation gateway should be attempting to connect to the orchestrator. Proceed to the next section.

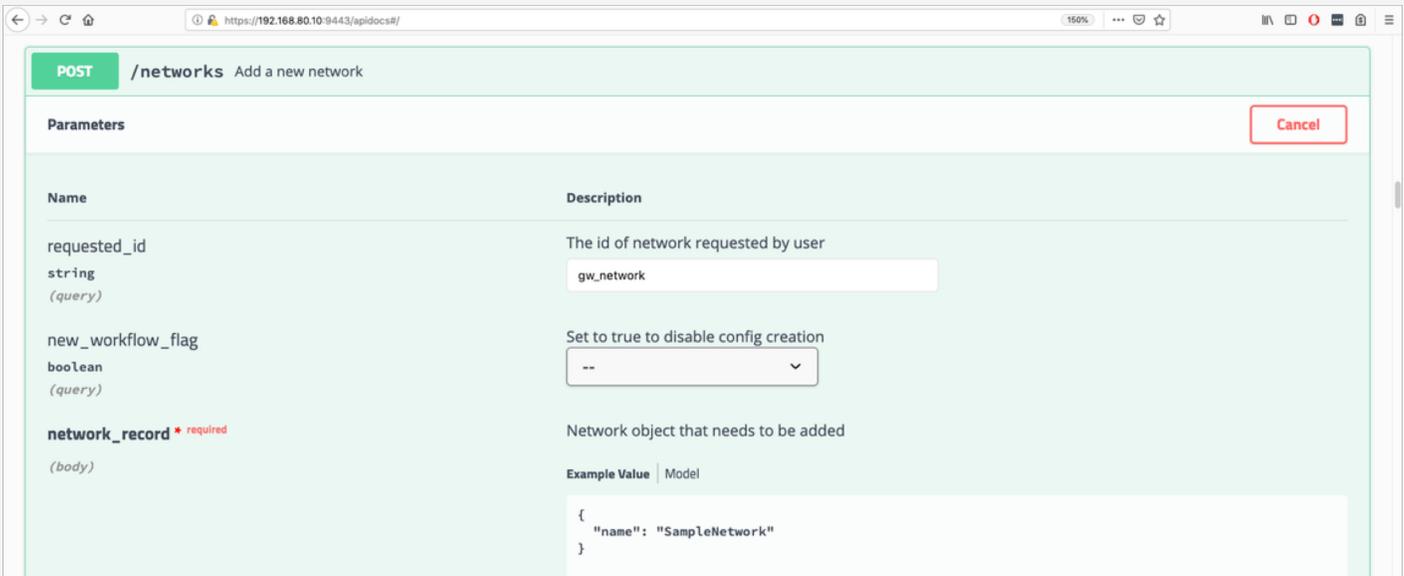
Registering the Network and Gateways

The certificate authenticates the browser as admin operator. You can use curl with the cert as well. Magma REST API Authentication & Authorization is based on the Operator Identity, which is tied to a client certificate signed by the Magma Certifier Service. So, to use the API, a client must present the certificate. The following steps provide instructions how to import this certificate into Firefox:

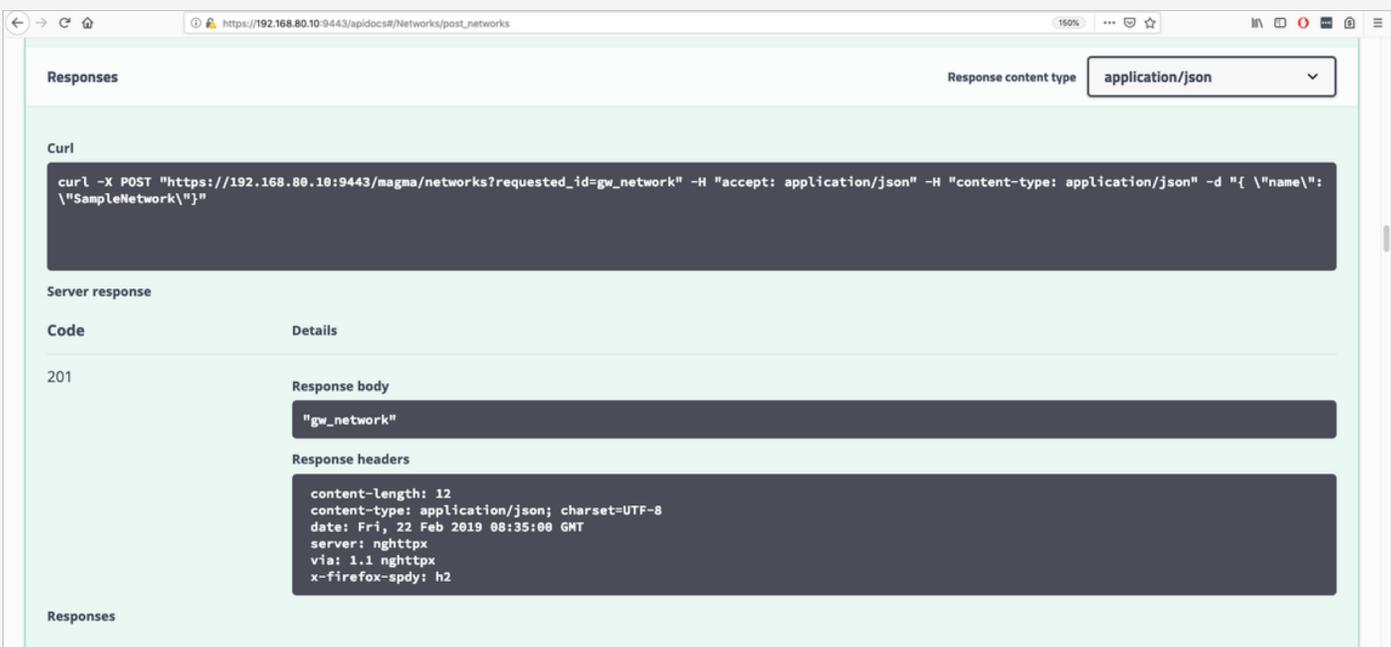
1. Import the client-side certificate **admin_operator.pfx** found under `~/magma/.cache/tests_certs` on your laptop to your browser (Firefox is recommended for this procedure):
 - a. Open Firefox, go to **about:preferences#privacy**.
 - b. Scroll down and select **View Certificates**.
 - c. Select **Import**.
 - d. Navigate to `.../magma/.cache/test_certs` directory/folder and select **admin_operator.pfx**
2. Go to <https://192.168.80.10:9443/apidocs#/> to access Swagger API. Swagger API is for setting configs, and Orchestrator will pass configs to corresponding VMs. You will be prompted to select a certificate. Select the certificate which is imported in step 2.
 - a. Since this is a private certificate generated when setting up the Orchestrator, the browser might warn you that the website uses an invalid certificate. It is safe to ignore the warning and proceed.
 - b. If a wrong cert was selected, and you could not see the prompt again even after refreshing the page, you should clear the history of browser so it would forget which cert was selected and asked you again.

3. In Swagger API, register the network and the gateway:

- a. Create a network for the gateway with a name such as **gw_network**. Do not start the name with a number.



- b. Verify that the request is successful by checking the response from the server. You should see a 200s. This also applies to all following steps. You should always check the response whenever sending requests.



- c. Set the network configuration parameters: such as cellular, dns, etc. You can use the default configuration unless you want to modify something.

The screenshot shows a REST client interface for the endpoint `/networks/{network_id}/configs/cellular`. The method is `POST`. The parameters section includes:

- network_id** (required, string, path): `gw_network`
- config** (required, body): A JSON object representing the cellular configuration. An example value is shown in a code block:

```

{
  "epc": {
    "default_rule_id": "default_rule_1",
    "lte_auth_amf": "gAA=",
    "lte_auth_op": "EREREREREREREREREQ==",
    "mcc": "001",
    "mnc": "01",
    "network_services": [
      "metering",
      "dpi",
      "policy_enforcement"
    ],
    "relay_enabled": false,
    "sub_profiles": {
      "additionalProp1": {
        "max_dl_bit_rate": 20000000,
        "max_ul_bit_rate": 100000000
      }
    }
  }
}

```

- d. Create a gateway under the network using the hardware ID you got using `show_gateway_info.py`. For other fields, enter:

```

"key_type": "ECHO"
"key": null

```

The screenshot shows a REST client interface for the endpoint `/networks/{network_id}/gateways`. The method is `POST`. The parameters section includes:

- network_id** (required, string, path): `gw_network`
- requested_id** (string, query): `gw_test`
- new_workflow_flag** (boolean, query): `--`
- Gateway Record** (required, body): A JSON object representing the gateway record. An example value is shown in a code block:

```

{
  "hw_id": {
    "id": "abca7138-e284-4287-93d0-0c94ce01e3f8"
  },
  "key": {
    "key": null,
    "key_type": "ECHO"
  },
  "name": "Gateway Test"
}

```

- e. Set the gateway configuration parameters: such as configs, configs/cellular, etc.
You can use the default configuration unless you want to modify something.

POST /networks/{network_id}/gateways/{gateway_id}/configs Create Gateway Configs

Parameters Cancel

Name	Description
network_id * required string (path)	Network ID <input type="text" value="gw_network"/>
gateway_id * required string (path)	Gateway ID <input type="text" value="gw_test"/>
GatewayConfigs * required (body)	Gateway Configs Example Value Model <pre>{ "autoupgrade_enabled": true, "autoupgrade_poll_interval": 300, "checkin_interval": 60, "checkin_timeout": 10, "dynamic_services": [], "feature_flags": { "newfeature1": true, "newfeature2": false }, "tier": "default" }</pre>

4. Repeat these steps to register the FeG. FeG is just like another gateway which should be registered under a different network.
5. One additional thing needs to be done when registering the FeG is setting the federation config. In the “served_network_ids” field, set it to the name of the network which hosts the gateway.

POST /networks/{network_id}/configs/federation Create Network FeG Configs

Parameters Cancel

Name	Description
network_id * required string (path)	Network ID <input type="text" value="feg_network"/>
config * required (body)	New config Example Value Model <pre>} }, "served_network_ids": ["gw_network"], "swx": { "server": { "address": "foo.bar.com:5555", "dest_host": "magma-fedgw.magma.com", "dest_realm": "magma.com", "host": "fedgw" } } }</pre>

POST /networks/{network_id}/gateways/{gateway_id}/configs/federation Create Gateway FeG Configs

Parameters Cancel

Name	Description
network_id * required string (path)	Network ID <input type="text" value="feg_network"/>
gateway_id * required string (path)	Gateway ID <input type="text" value="feg_test"/>
config * required (body)	New config Example Value Model <pre> } }, "served_network_ids": ["gw_network"] </pre>

6. Verify that both the FeG and the Gateway are setup correctly:
 - a. Check /var/log/syslog of FeG VM, you should see **Checkin Successful!**
 - b. Check /var/log/syslog of gateway VM, you should see **Checkin Successful!**

```
Jul 13 04:39:02 magma-dev magmad[23931]: [2018-07-13 04:39:02,708 INFO root] Checkin Successful!
```

All the Magma Services should now be provisioned and communicating with each other.

Troubleshooting on Virtual Machines

- Read `/var/log/syslog` on VMs to look for issues. All Magma services are logged there.
- All services are stateless, so you can try to restart all services.
 - For Orchestrator VM, use the command:

```
vagrant@magma-cloud:~/magma/orc8r/cloud$ make restart
```

- For FeG VM, use the command:

```
vagrant@magma-dev:~/magma/feg/gateway$ make restart
```

- For Gateway VM, use the command:

```
vagrant@magma-dev:~$ sudo service magma@* stop  
vagrant@magma-dev:~$ sudo service magma@magmad start
```

- Rebuild and restart all services.
- Re-provisioning VMs can help in some situations. Please read the YAML file or Ansible playbook to understand what the provisioning process does for each VM. Here are some examples where re-provisioning might help:
 - File structure change
 - Environment variables being messed up
 - Missing libraries / wrong libraries version
 - Issues caused by missing files or binaries

```
user@user-mbp:~/magma/orc8r/cloud$ vagrant provision cloud  
user@user-mbp:~/magma/orc8r/cloud$ vagrant provision datastore  
user@user-mbp:~/magma/lte/gateway$ vagrant provision magma  
user@user-mbp:~/magma/feg/gateway$ vagrant provision feg
```

- Destroy existing VMs and bring up new VMs with these commands:

```
user@user-mbp:~/magma/orc8r/cloud$ vagrant destroy cloud && vagrant up cloud  
user@user-mbp:~/magma/orc8r/cloud$ vagrant destroy datastore && vagrant up datastore &  
user@user-mbp:~/magma/lte/gateway$ vagrant destroy magma && vagrant up magma  
user@user-mbp:~/magma/feg/gateway$ vagrant destroy feg && vagrant up feg
```

- Use **tcpdump** to capture the communication between VMs

Staging the eNodeB with TR-069 support

Note: The gateway should already be set up for staging with the controller before you begin staging with the eNodeB.

Note: These instructions will only work for a device that uses TR-069 for configuration.

1. Setup the eNodeB and its peripherals according to the vendor instructions.
2. Power on the eNodeB and verify that the PWR light goes green on the eNodeB.
3. Connect to the MGMT port of the eNodeB to your computer.
4. In your web browser, open up the management portal of the device.
 - a. Find and edit the setting for the Management Server IP that the device uses to `baiomc.cloudapp.net:48080`
5. Connect the DATA or WAN port on the eNodeB directly to a second gateway. Do not connect the eNodeB directly to the internet.
 - a. The eNodeB and the first gateway should be connected to a network through the second gateway. The first gateway runs a DHCP server that services this eNodeB. So, no other DHCP servers can be connected to this eNodeB.
6. Verify that the eNodeB and the gateway are communicating. On the command line of the gateway:
 - a. Enter: `sudo enodebd_cli.py get_status`
The eNodeB should display `eNodeB connected: 1`
 - b. If the eNodeB does not connect to the gateway within 10 minutes, restart the eNodeB daemon (`enodebd`).
Enter: `sudo systemctl restart magma@enodebd.`
It may take up to 10 minutes for the gateway to configure the eNodeB the first time it is powered on.
7. If the eNodeB configuration is not completed in 10 minutes:
 - a. Check the progress of the `enodebd` service.
Enter: `sudo journalctl -f -u magma@enodebd.service`
 - b. Restart the `enodebd` service.
Enter: `sudo systemctl restart magma@enodebd`
 - c. Power cycle the eNodeB.

8. Confirm that the configuration has completed successfully:
 - a. Enter: `sudo enodebd_cli.py get_status`.
The eNodeB should display `eNodeB configured: 1`
9. Verify that the eNodeB is radiating:
 - a. On the gateway, enter: `sudo enodebd_cli.py get_status`.
The gateway should display `RF TX on: 1`.
 - i. Or on the controller, in the gateway summary box, verify that the **RF transmitter on** button is green.

Staging the eNodeB without TR-069 support

You can stage an eNodeB without TR-069 support as follows:

1. Connect the laptop to the MGMT port on the eNodeB.
2. In a web browser, connect to the management portal of the eNodeB device (this address is device dependent).
3. Configure the following parameters or their corresponding device specific variations through the management portal. All other parameters are optional.
4. Configure the PLMN settings.

Parameter	Value
MME Connection Settings	
MME IP	192.168.60.142
MME Port	36412
Performance Management Settings	
Performance Management Enable	TRUE
Performance Management Upload Interval	300 (seconds)
Performance Management Upload URL	http://192.168.60.142:36412/

Provisioning the Magma Gateway

Once you have the Gateway installed, you can connect to and use the Magma Network Management System (NMS) to monitor the system and verify settings, as well as provision and configure the network, gateway, and subscribers.



facebook
connectivity