COLLEGE OF ENGINEERING
Systems & Industrial Engineering

RESEARCH, DISCOVERY & INNOVATION
Transportation Research Institute

Multi Modal Intelligent Traffic Signal System
Simulation Deployment – User Manual

Revision 0.0

(Initial Release)

July 10, 2020

THE UNIVERSITY OF ARIZONA

Table of Contents

1. Purpose of Document

This document is an instruction guide for deploying Multi-Modal Intelligent Traffic Signal System (MMITSS) applications in the simulation platform. The document contains the detailed configuration and usage instruction for each of the MMITSS software components in the docker container.

2. Simulation Model

PTV Vissim (a microscopic simulation software) is used to simulate all modes of traffic and analyses the performance of MMITSS software components. Vissim can be utilized to create realistic and accurate in every detail to test different traffic scenarios. A sample Vissim model of Daisy Mountain - Gavilan Peak intersection at Anthem, Arizona is shown in figure1.

Figure 1: Vissim model of Daisy Mountain - Gavilan Peak intersection

2.1. Configuring Connected Vehicle:

A new vehicle type (Connected vehicle) can be created using external driver model dll. In section 3, the method of setting up driver model dll is discussed. To create a connected vehicle say Transit, a new vehicle type required to be defined (figure2). After that, the directory of the diver model dll file has to define under the External Driver Model as shown in the figure 2. Finally this new vehicle type is required to assign under Base Data->Vehicle Classes.

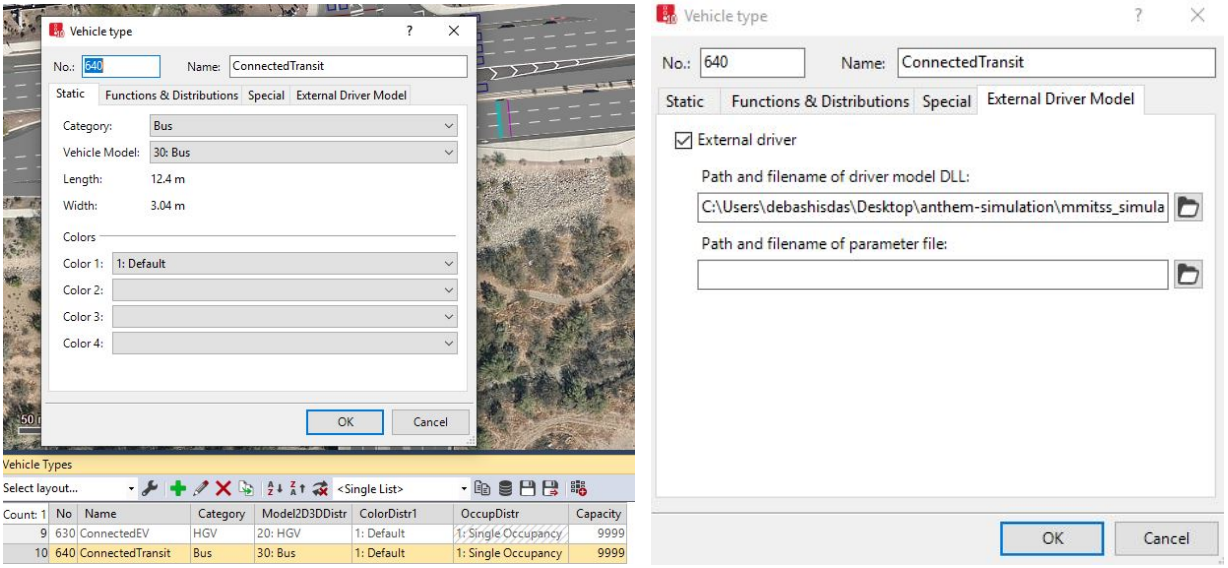

Figure 2: Defining Connected Vehicle Type

2.2. Signal Controller:

To establish communication between the MMITSS software components signal controller, it is required to configure the signal controller. Different agencies use different type of signal controllers. In Tucson and Anthem use Econolite ASC/3 signal controller, Portland use MaxTime signal controller to control the traffic flow at each intersection.

2.2.1. Configuring Econolite ASC/3 signal controller:

Following steps are required to configure Econolite ASC/3 signal controller:

(a) At first define the IP address of the computer on which Vissim software. Mrp container and Vissim computer must be in the same subnet. For example- if the intersection container host IP address is: 10.254.56.XX then Vissim computer IP address will be 10.254.56.YY

(b) Next step is to define the controller IP, subnet mask, default gateway IP, server IP, NTCIP backup time and NTCIP UDP Port in the Vissim ASC/3 contoller. To do so, from the Vissim go to each signal controller and click “Parameters” option (figure3). Then go to the Comm-Pg1 option (MM 1-4) under configuration and define the parameters (figure4).

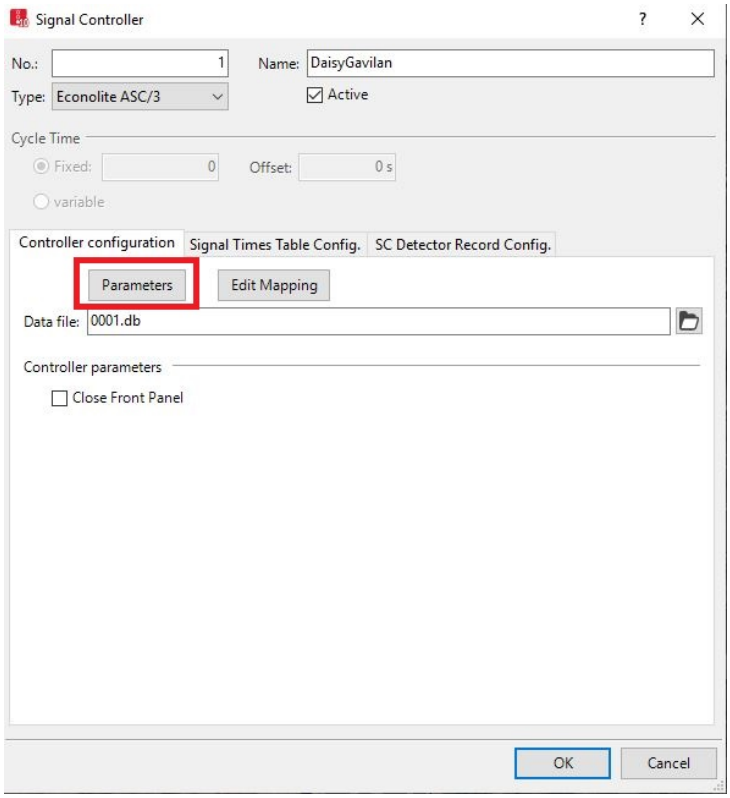

Figure 3: Prepare to setup the parameters

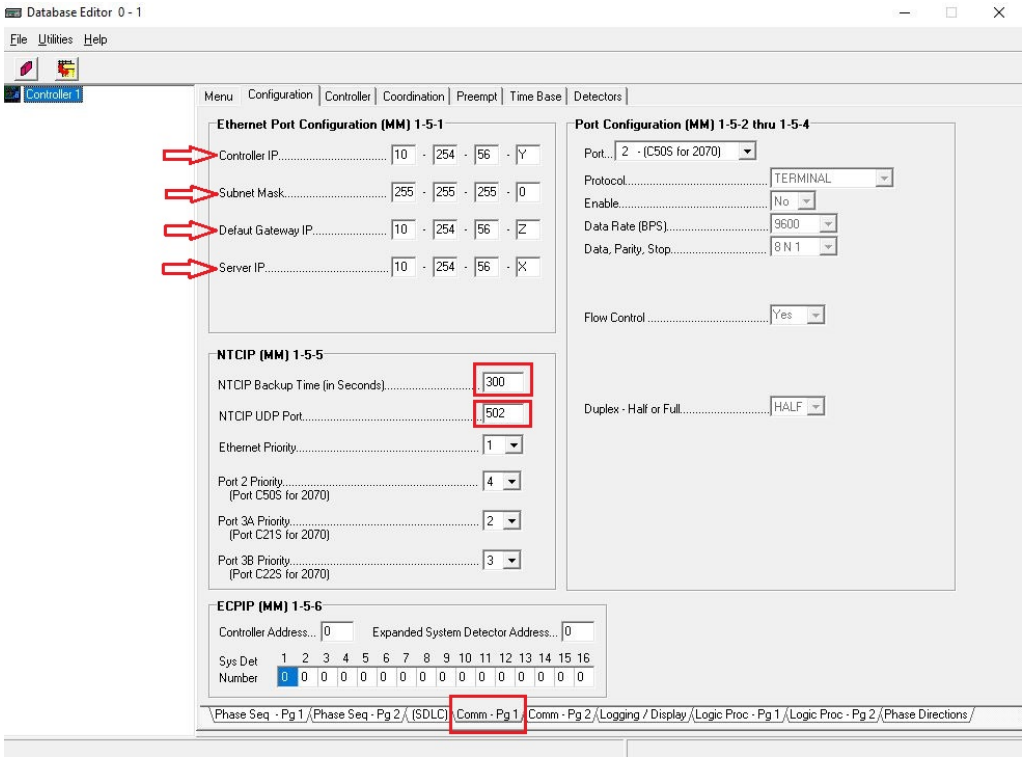

Figure 4: Define the parameters

2.2.2. Configuring MaxTime signal controller:

Following steps are required to configure MaxTime Signal controller:

(a) Start MaxTime signal controller by defining unique Web Port, PTV Vissim TCP port, PTV Vissim id for each intersection (figure 5).

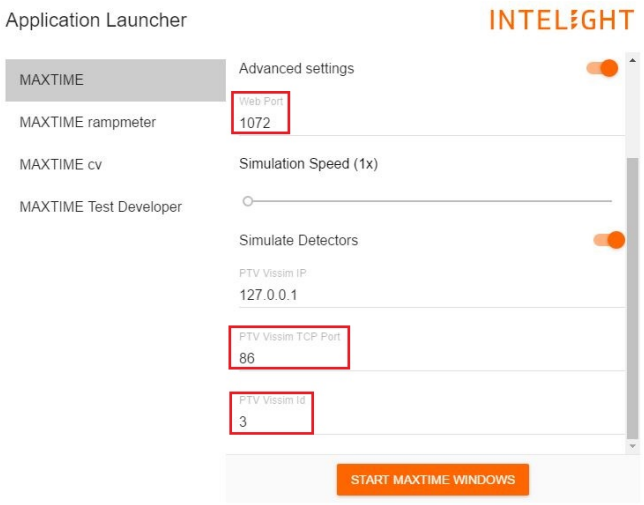

Figure 5: Starting MaxTime signal Controller

(b) Then controller IP address has to define from peer configuration (figure 6).

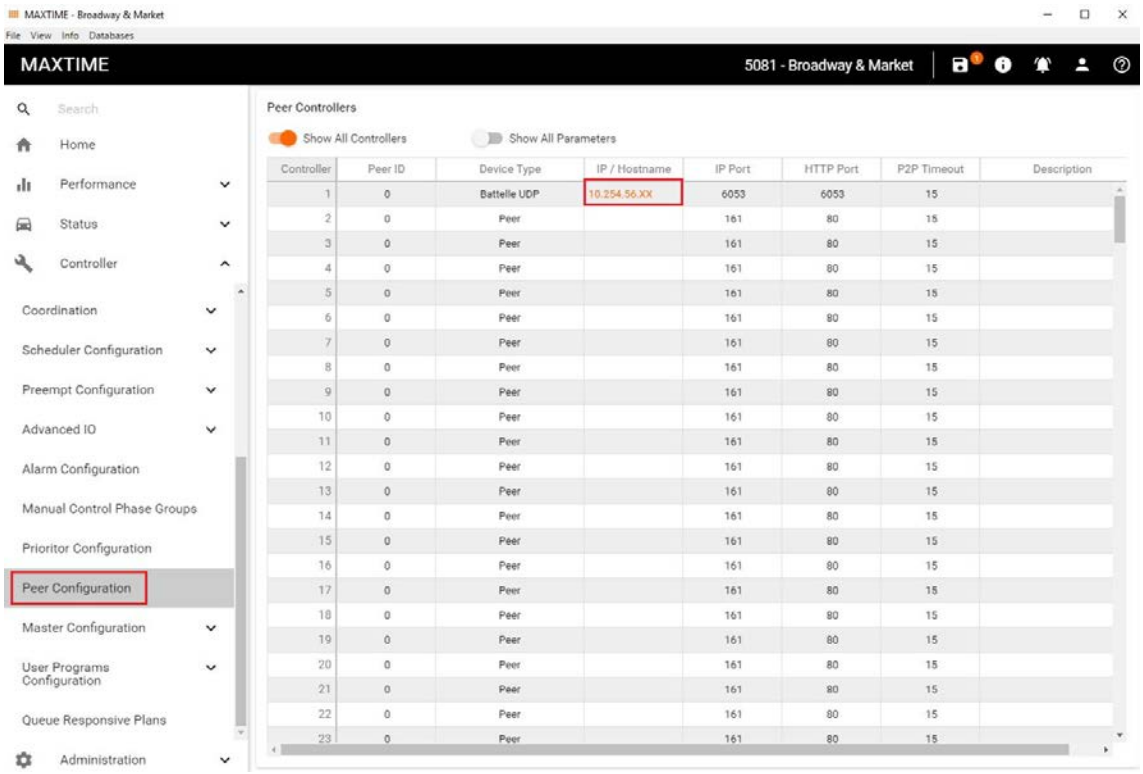

Figure 6: Defining the IP address in MaxTime Signal Controller

3. VISSIM Driver Model DLL

Built upon the original sample driver-model provided with VISSIM distribution, a custom driver-model has been developed to simulate connected vehicles that interact with MMITSS applications. Four variants of this driver-model are built which are the following:

1. M_DriverModelPassenger.dll (type = “passenger”)
2. M_DriverModelEmergency.dll (type = “emergency”)
3. M_DriverModelTransit.dll (type = “transit”)
4. M_DriverModelTruck.dll (type = “truck”)

At every time instant in the VISSIM simulation, vehicles using any of above driver-models send a UDP packet containing a Basic Safety Message (BSM) like JSON formatted information related to the vehicle’s identification, type, size, current location and motion to a network node defined in the configuration file. An example of JSON string developed by the driver model is as follows:


```
{
   "MsgType": BSM,
   "BasicVehicle": {
      "heading_Degree": 359.1,
      "position": {
         "elevation_Meter": 587,
         "latitude_DecimalDegree": 33.843455,
         "longitude_DecimalDegree": -112.135159
      },
      "secMark_Second": 2600,
      "size": {
         "length_cm": 400,
         "width_cm": 300
      },
      "speed_MeterPerSecond": 12.5,
      "temporaryID": 5822735,
      "type": "transit"
   },
}
```

Functionally all four driver models are identical except that the value in the *type* field in the JSON string is different for each driver-model. These driver-models are tested in VISSIM 10 and VISSIM 2020.

To use any of these driver-models, a directory named “mmitss_simulation” needs to be placed in the same directory where the “*.inpx” file of the simulation exists. This directory is provided in the MMITSS simulation distribution package. The “mmitss_simulation” directory contains a configuration file named “mmitss_driver_model_config.json” and a subdirectory named “driver_models” where all four prebuilt driver-models are stored. All four driver-models use the same configuration file and it must be made sure that the relative location of the configuration file from the “*.inpx” file is not altered. The configuration file allows to configure the network node where the UDP packets of JSON strings need to be sent. In addition, it is required to input the GPS coordinates of the origin (0,0) of the simulation model in the configuration file. This information is required to correctly transform the vehicle’s current location from the VISSIM’s local coordinate system to the GPS coordinate system. An example of the content of the “mmitss_driver_model_config.json” is as follows:

```
{
   "msg_distributor_ip": "127.0.0.1",
   "msg_distributor_port": 5000,
   "vissim_origin_position": {
      "elevation_Meter": 540.11,
      "latitude" : {
         "Degree" : 33.0,
         "Minute" : 50.0,
         "Second" : 35.4
      },
      "longitude" : {
         "Degree" : -112.0,
         "Minute" : -8.0,
         "Second" : -6.1
      }
   }
}
```

The IP address and UDP port of the receiver network node can be specified in the “msg-distributor-ip" and “msg-distributor-port" fields respectively. To interface with other MMITSS applications, this network-node must be the *message-distributor* component of MMITSS simulation-tools.

4. Simulation-Tools

To interface simulation models with other MMITSS core components, two additional components are developed: *message-distributor* and *priority-request-generator-server*. The *message-distributor* component is responsible for receiving and distributing messages from VISSIM simulation model and other MMITSS components to the *priority-request-generator-server* component or to appropriate intersection containers, whereas the *priority-request-generator-server* component is responsible for formulating the Signal Request Messages (SRMs) based on the vehicle information received from the *message-distributor* component.

4.1. Message-Distributor

As the name suggests, the *message-distributor* component is responsible for receiving messages from VISSIM simulation model or other MMITSS components and distributing them to applicable configured clients. The *message-distributor* also allows to configure and simulate the wireless range (in Meter) of each intersection in the corridor along with a probability of dropping message packets. The clients (IP address and UDP port) for supported messages can be configured in the configuration file of the *message-distributor*. An example of the configuration file is as follows:


```
{
   "package_drop_probability": 0,
   "raw_bsm_logging": true,
   "intersections": [
      {
         "name": "DaisyMountain_GavilanPeak",
         "ip_address": "xxx.xxx.xxx.xxx",
         "bsm_client_port": 5001,
         "srm_client_port": 20002,
         "dsrc_range_Meter": 500,
         "position": {
            "latitude_DecimalDegree": 33.842932,
            "longitude_DecimalDegree": -112.135186,
            "elevation_Meter": 539
         }
      },
      {
         "name": "DaisyMountain_DedicationTrail",
         "ip_address": "xxx.xxx.xxx.yyy",
```

```
        "bsm_client_port": 5001,
        "srm_client_port": 20002,
        "dsrc_range_Meter": 500,
        "position": {
            "latitude_DecimalDegree": 33.843239,
            "longitude_DecimalDegree": -112.131541,
            "elevation_Meter": 539
        }
    },
],
"bsm_clients":
{
    "transit":[
        {
            "ip_address": "xxx.xxx.xxx.zzz",
            "port":20022
        }
    ],
    "truck":[
        {
            "ip_address": "xxx.xxx.xxx.zzz",
            "port":20022
        }
    ],
    "emergency":[
        {
            "ip_address": "xxx.xxx.xxx.zzz",
            "port":20022
        }
    ],
```

```
    "passenger":[
    ]
  },
  "map_clients":
  [
    {
      "ip_address": "xxx.xxx.xxx.zzz",
      "port":20022
    }
  ],
  "ssm_clients":
  [
    {
      "ip_address": "xxx.xxx.xxx.zzz",
      "port":20022
    }
  ]
}
```


The *message-distributor* receives and distributes following messages in the described way:

1. *BSM-like information from VISSIM simulation model:*

   If the *vehicle type* in the received message is either “emergency”, “transit”, or “truck”, the message is forwarded to the priority-request-generator-server component (or more clients if required). In the configuration file above, the IP address xxx.xxx.xxx.zzz corresponds to the docker container (or physical machine) that hosts the *priority-request-generator-server*. In addition, regardless of the *vehicle type*, if the location described in the received message is within the wireless range (Meter) of any of the configured intersections, the message is distributed to that intersection’s BSM receiving client (defined by intersection_ip and bsm_client_port).

2. *SRMs from priority-request-generator-server*

   The *priority-request-generator-server* sends JSON strings of formulated SRMs to the *message-distributor*. If the location of vehicle (as per the received message) is within the wireless range of any (Meter) of any of configured intersections, the message is distributed to that intersection’s SRM receiving client (defined by intersection_ip and

srm_client_port). For MMITSS priority applications, the SRM receiving client is the *priority-request-server*.

3. *SSMs from priority-request-server of each intersection*

   The *priority-request-server* component of each intersection that receives SRMs, generate and broadcast SSMs as an acknowledgement of the receipt of SRM if it is eligible for the priority service. The *priority-request-server* of each intersection also forwards such SSMs to the *message-distributor,* which then forwards these SSMs to configured SSM receiving clients. For MMITSS priority applications, one SSM receiving client is the *priority-request-generator-server* component.

4. *MAPs from map-spat-broadcasters of each intersection*

   The *map-spat-broadcaster* component of each intersection broadcasts MAP message at a frequency of 1 Hz. It also forwards the MAP messages to the *message-distributor*, which then forwards these MAPs to configured MAP receiving clients. For MMITSS priority applications, one MAP receiving client is the *priority-request-generator-server* component.

4.2. Priority Request Generator Server:

5. Deployment – Docker Containers

3.1.Intersections

3.2. Simulation-Tools

3.2.1. Message-Distributor

3.2.2. Priority Request Generator Server