

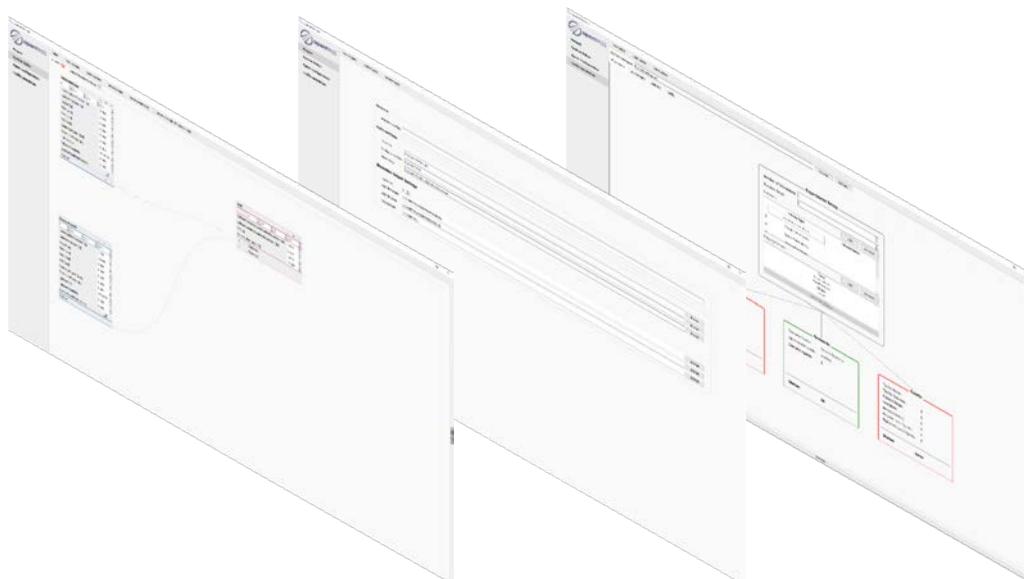


Tutorial: How to configurate a simulation with openPASS

What is openPASS ?

The growth of advanced driver assistance systems and partially automated driving functions demand a virtual simulation tool, which assess these systems and their effects. Especially with regard to the safety in traffic, it is inevitable to establish a universal simulation software to evaluate and analyse the effectiveness of safety systems. OpenPASS (Open Platform for Assessment of Safety Systems) serves this purpose: it is aimed at providing a modular and, thus, extendible simulation platform for simulating complex traffic scenarios and for testing safety systems.

This tutorial is a guide on how to configure a simulation using the OpenPASS graphical user interface (GUI). In particular, it is presented how to go through the required steps until a simulation can be run successfully. To enhance the guiding, there are going to be examples in every section. It is important to mention that openPASS is still in progress and some plugins and options have not been fully implemented yet.



List of contents

- 1. Introduction..... 3
- 2. Master Configuration 3
- 3. System Editor 4
- 4. Agent Configuration 6
- 5. Scenario-based Simulation Configuration 8
 - 5.1. Environment setup 9
 - 5.2. Scenario setup 9
 - 5.3. Traffic setup..... 10

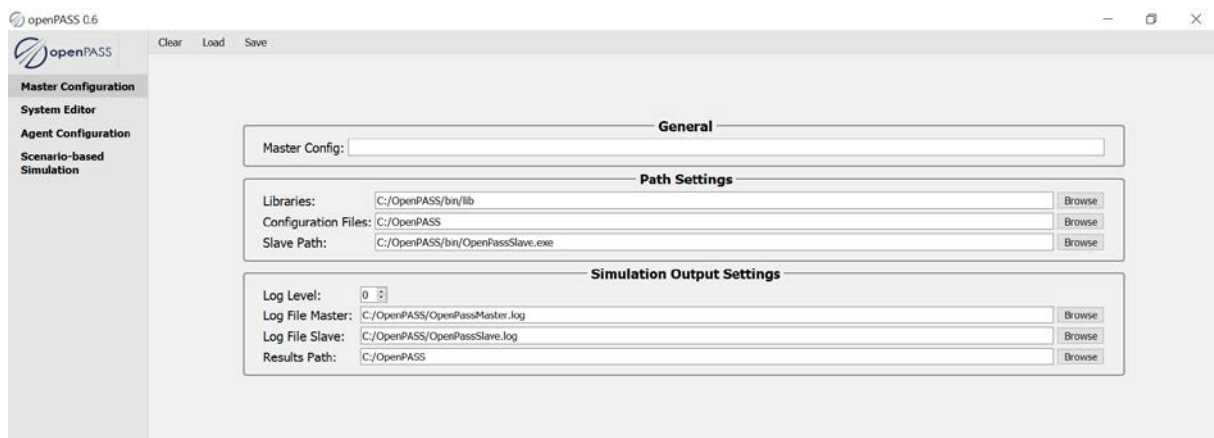
1. Introduction

The main task of the OpenPASS GUI is to write (mostly) all configuration files required by the OpenPASS simulation slave, i.e. the instance which executes the simulation. To date, the configuration files that are editable by the OpenPASS GUI are the **masterConfig**, the **systemConfig**, the **ProfilesCatalog** and the **slaveConfig**. Additionally, there are OpenDRIVE and OpenSCENARIO files as well as pedestrian and vehicle model catalogues for which no GUI editor has been designed, yet.

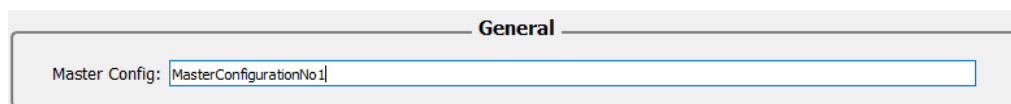
In general, each of the aforementioned files is edited by a single GUI plugin listed in the sidebar of the OpenPASS window. The necessary steps for the file's configuration will be outlined in the following sections based on examples. The aim is to make the user familiar with each GUI plugin and its usage. Please note, that we **do not aim** at explaining all the details of the options and settings of each configuration file and their consequences for the simulation core. Hence, please consult the corresponding documentation of the simulation framework for such information.

2. Master Configuration

Before the simulation adjustments begins, the user is obligated to load or create a “Master Configuration” (masterConfig). A Master Configuration in openPASS can be understood as a project. It is a XML-File which get inscribed the path settings and simulation settings after you click **Save** .

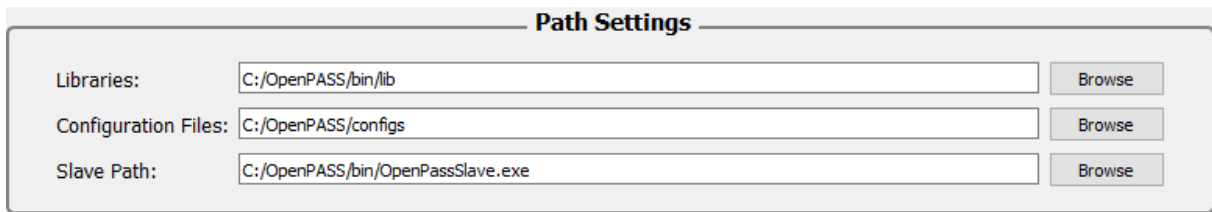


a) First segment: **General**



In this segment you are able to name the Master Configuration. In our example it is called „MasterConfiguratioNo1”

b) The next segment: **Path Settings**



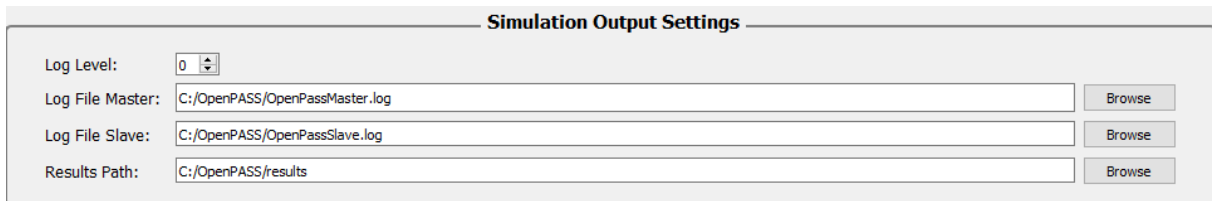
Path Settings	
Libraries:	<input type="text" value="C:/OpenPASS/bin/lib"/> <input type="button" value="Browse"/>
Configuration Files:	<input type="text" value="C:/OpenPASS/configs"/> <input type="button" value="Browse"/>
Slave Path:	<input type="text" value="C:/OpenPASS/bin/OpenPassSlave.exe"/> <input type="button" value="Browse"/>

The next step is path settings. These will change depending on where your openPASS.exe is located. In the screen shot above the openPASS.exe is located at C:/OpenPASS. For easier use of this tutorial it is recommended to save the Demo Folder in C:/ and name it OpenPASS.

On to the settings. As you can see three paths need to be set. The library comes with openPASS. There are plans to remove the option for the user to set the library path, but at this moment there is still the option to change it, although this is not recommended.

The Slave Path references the OpenPassSlave.exe, the file to execute the slave. If you are using the provided Demo, there is no need for you to change it. The only path you need to set is the Configuration Files. In the Demo it will be located at [directory of openPASS.exe]/configs, so in this case it would be C:/OpenPASS/configs.

c) **Simulation Output Settings**



Simulation Output Settings	
Log Level:	<input type="text" value="0"/> <input type="button" value="Browse"/>
Log File Master:	<input type="text" value="C:/OpenPASS/OpenPassMaster.log"/> <input type="button" value="Browse"/>
Log File Slave:	<input type="text" value="C:/OpenPASS/OpenPassSlave.log"/> <input type="button" value="Browse"/>
Results Path:	<input type="text" value="C:/OpenPASS/results"/> <input type="button" value="Browse"/>

Next up is the Simulation Output Settings. There are three output files. First is the log file of the master. However, when simulation jobs are started by the GUI, the openPASS master is not executed and, hence, the master log is not important in this tutorial. Second is the log file created by the slave. In this log file you will find error messages, actions of the slave etc. depending on the log level. The Log level lets you choose, which type of messages are logged. "0" means that only errors are logged, whereas the highest log level of "5" leads to the most detailed description of which steps are executed by the slave. The results path specifies the folder in which the results of a successful simulation will be saved. **It is recommended to create a new folder in C:/OpenPASS called "results" and set it as the results path as in the picture above.**

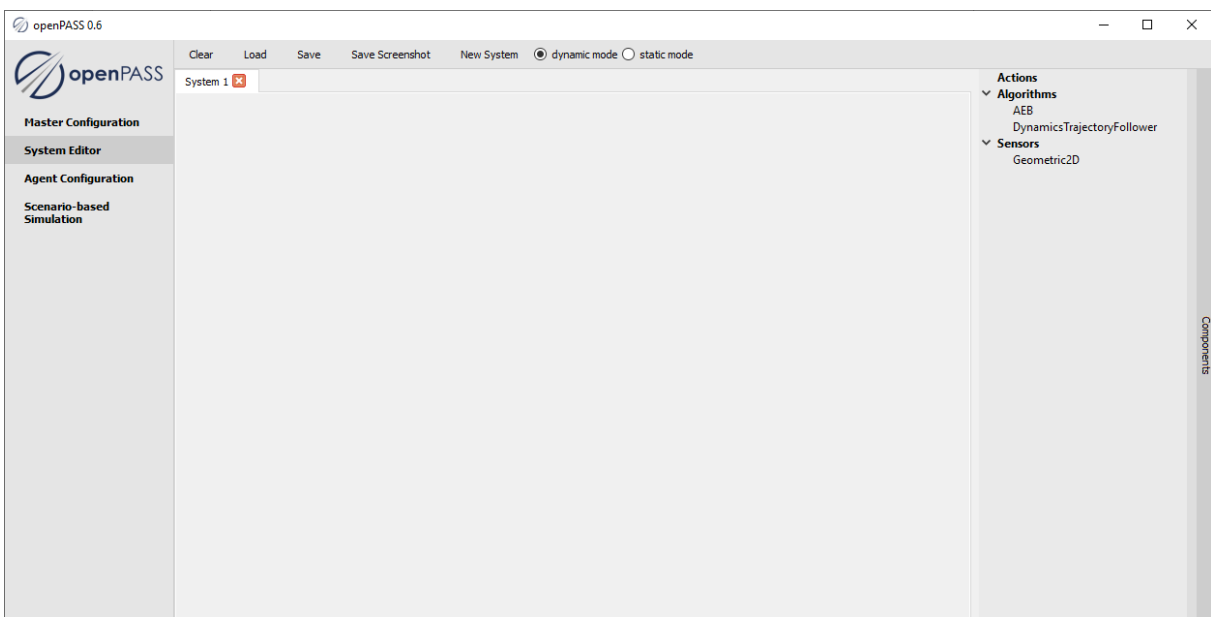
3. System Editor

The next point is building the system configuration (systemConfig) in the System Editor. This file is also a XML-file and specifies the systems used, in this case advanced driver assistance systems. The Editing of the systemconfig has two modes: the static and dynamic mode.

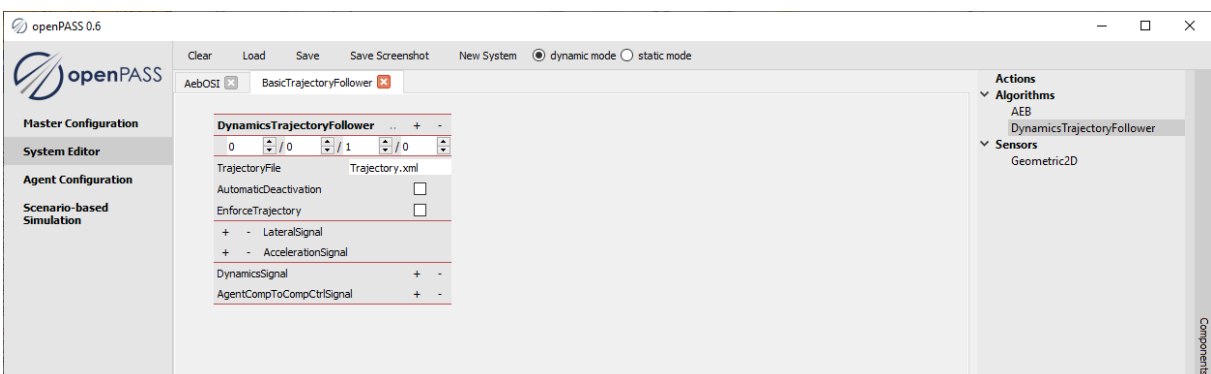
dynamic mode: The dynamic mode only includes the system itself. The advantage of the dynamic mode is, that you can statistically vary if the system will be built into the vehicle. The possibilities will be shown in the following chapter. The disadvantage is that the interaction between system and slave is hard coded in the systemConfigBlueprint and the slave itself. Therefore it is not possible to create new systems, which is why a second mode is introduced. The static mode.

static mode: The second mode requires the user to build a complete system which includes sensors, algorithms and actions. As the actions directly manipulate the simulated vehicle's parameters, which are the same no matter the system (i.e. gas pedal position, braking pedal position, steering wheel angle), there is no need to code the interaction between system and slave. The disadvantage is that at this point there is no support for supporting statistical inclusion of systems in the static mode.

The current demo only provides an example for the dynamic mode, so make sure you have the dynamic mode selected.



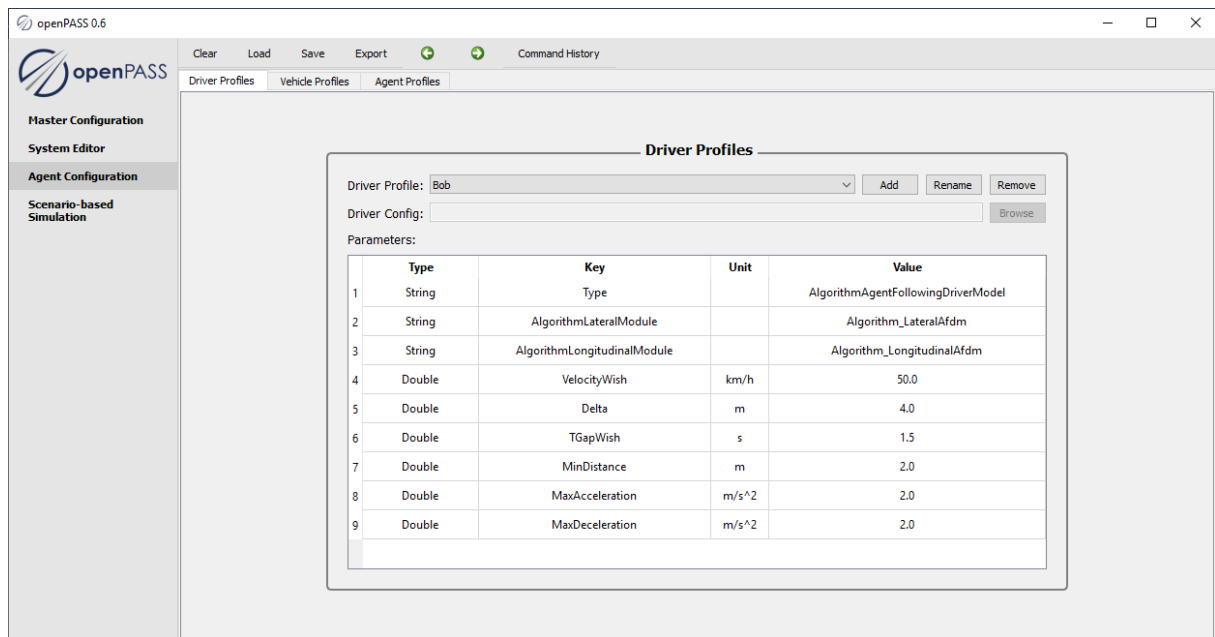
On to the system config itself. The picture displays the window to create a system config. On the right you will see available components, in this case the algorithms “AEB” and “DynamicsTrajectoryFollower” and a sensor “Geometric2D”. You can simply drag and drop the components into the blank window. Afterwards you can add connections by clicking the + of an input and the + of an output. Inputs have the ± on the left side while outputs have the ± on the right side. Clicking the – will delete all connections of the corresponding signal. The demo provides a systemconfig which we will be working with in this tutorial. Simply go to load system and choose “systemConfig.xml”. Now the window should look like this





4. Agent Configuration

Agents are the main components of the simulation and are stored in the ProfilesCatalogue. An agent consists of a driver component and a vehicle component. The driver component is called “Driver Profile” and the vehicle component is called “Vehicle Profile”, which includes vehicle and system. As you need a driver profile and vehicle profile to create an agent profile, we will start with creating a driver profile.

Go to the Driver profile tab and click **Add**. Now we need to pick a name for the driver, let’s call him “Bob”. Your window should now look like this:



There are three strings specifying the type of driver profile, the lateral and longitudinal module. Note, that these options should not be changed in the current released as there is only one type of each module. It is intended, however, to provide further flexibility in future releases. Below the three strings, there are parameters that you are allowed to vary. You can specify the speed at which Bob will drive, the timeGap he wants between his own vehicle and the one ahead, the minimum distance before he gets uncomfortable etc.

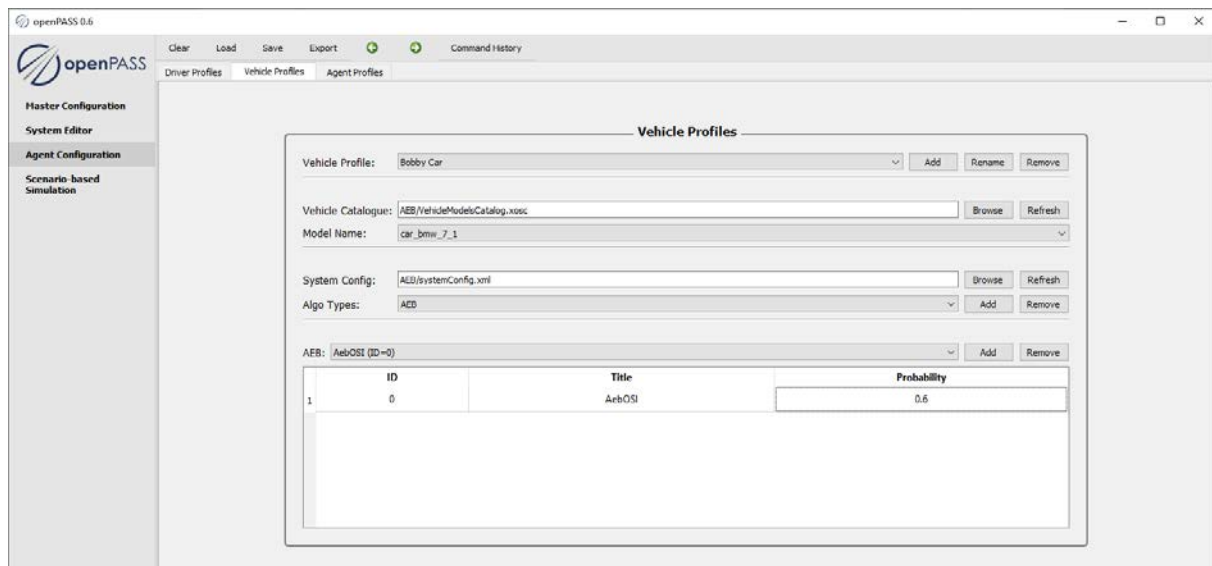
The Agent Configuration plugin offers an undo and redo button  . It also provides a function to display every action made by clicking **Command History**. Both features are implemented to let the user keep track of his actions and change them if necessary.



Hint: In this plugin the Save and Load-buttons are only meant for storing the GUI’s current state. For historical reasons, the file format for the ProfilesCatalog.xml, required by the simulation slave, is different. In order to generate the required file for the simulation slave, use the export-button.

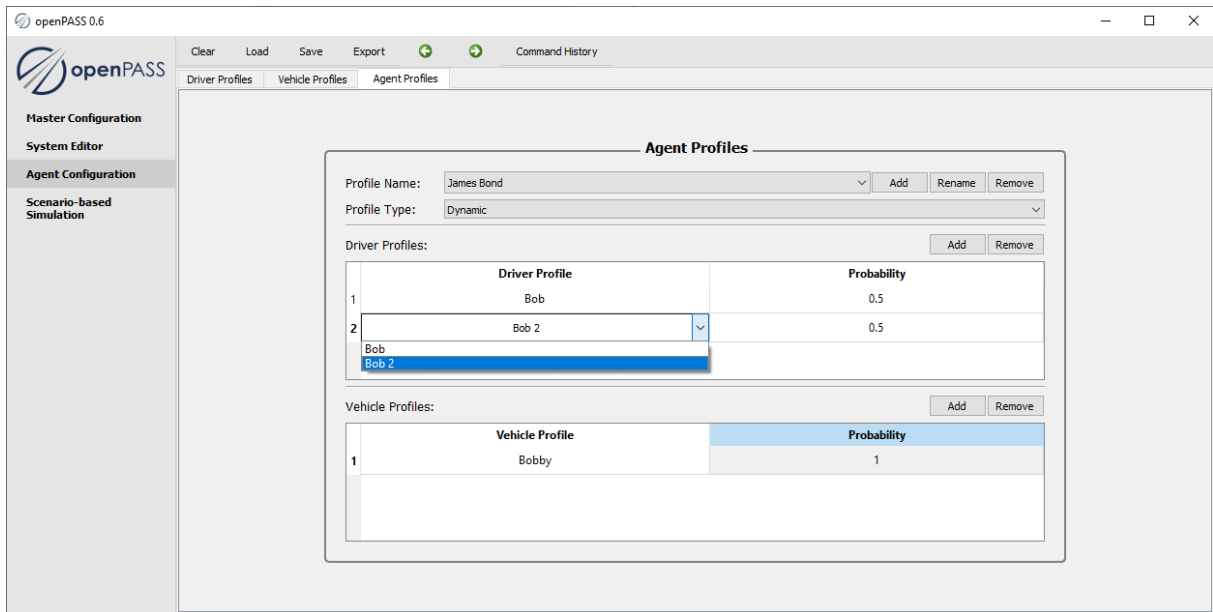
Next up we're going to build the vehicle Profile. Simply go to the tab Vehicle Profiles, click on **Add** and give it a name. We are going to name it "Bobby Car". Now you have to reference a vehicle catalog. Click on browse and choose "VehicleModelsCatalog.xosc". If you can't find it, go back to the Master Configuration and confirm you have linked the right folder for Configuration files. As Bobby cars do not have driver assistance systems yet, we are going with a BMW 7_1.

Afterwards we will include a system into Bob's car. To do that, you have to click on **Browse** and either choose systemConfig.xml or, if you recreated it, you may also choose your own systemConfig. Now choose an algorithm type (Algo Type) you want to include in the vehicle. After adding an algorithm type (Algo Type), you can add a specific system (defined in "systemConfig.xml") of that type to the vehicle profile and assign to it a probability to be equipped. In our example, we have chosen a system of type "AEB" and added the specific system "AebOSI" (ID=0) to the vehicle with a probability of 0.6. Note, that the difference probability (0.4) determines the probability that no system of type "AEB" is built into the vehicle.



Last but not least, we are going to build a Agent Profile. To build a complete Agent Profile, you have to enter a Driver Profile and a Vehicle Profile. OpenPASS offers the user to add probabilities to every driver and vehicle profile to enable a distribution here as well.

First up we are going to name the Agent. Let's go with the obvious and name it "James Bond". The Profile type specifies the mode chosen in the system editor, so in this case set it to Dynamic. Afterwards add a Driver Profile and Vehicle Profile. Simply click on **Add** and they appear. If you created more than one Driver or Vehicle Profile, you are able to select a Profile by double clicking on the cell with the Profiles in it. In this case the probabilities have to add up to 1 again.

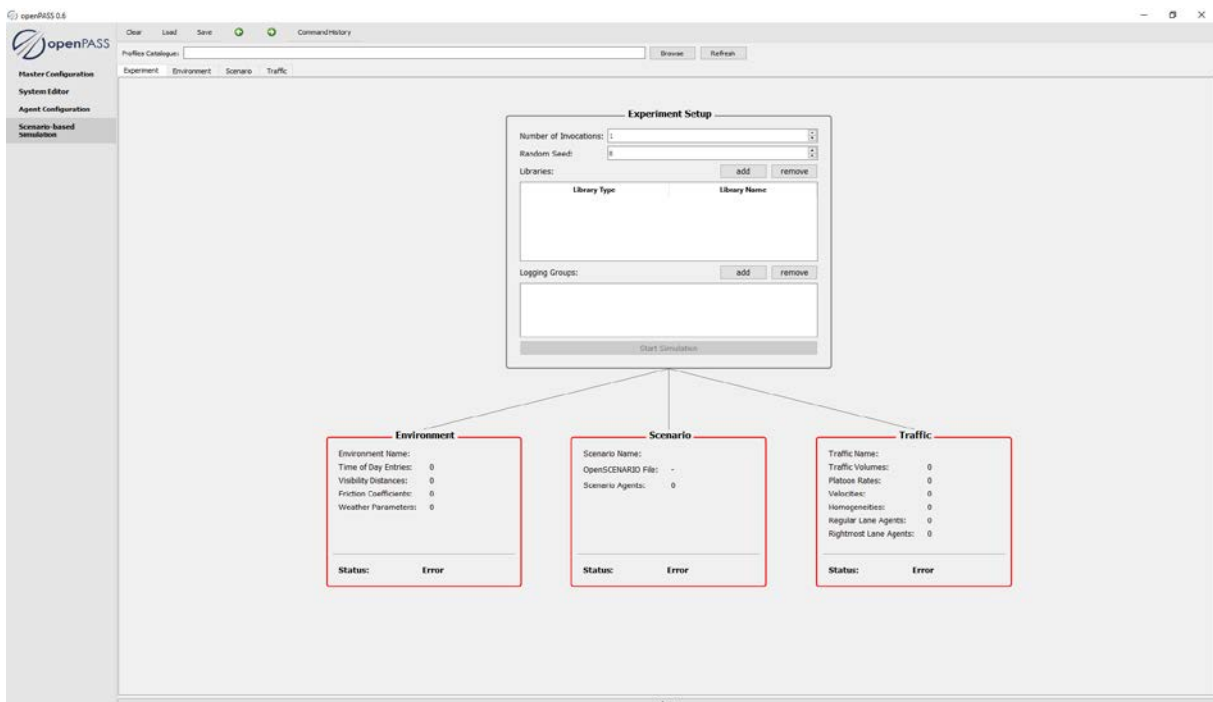


5. Scenario-based Simulation Configuration

Now that we completed the Agent-related setup, we have to set up the Scenario-based Simulation Configuration, which represents the slaveConfig file. This plugin consists of three parts:

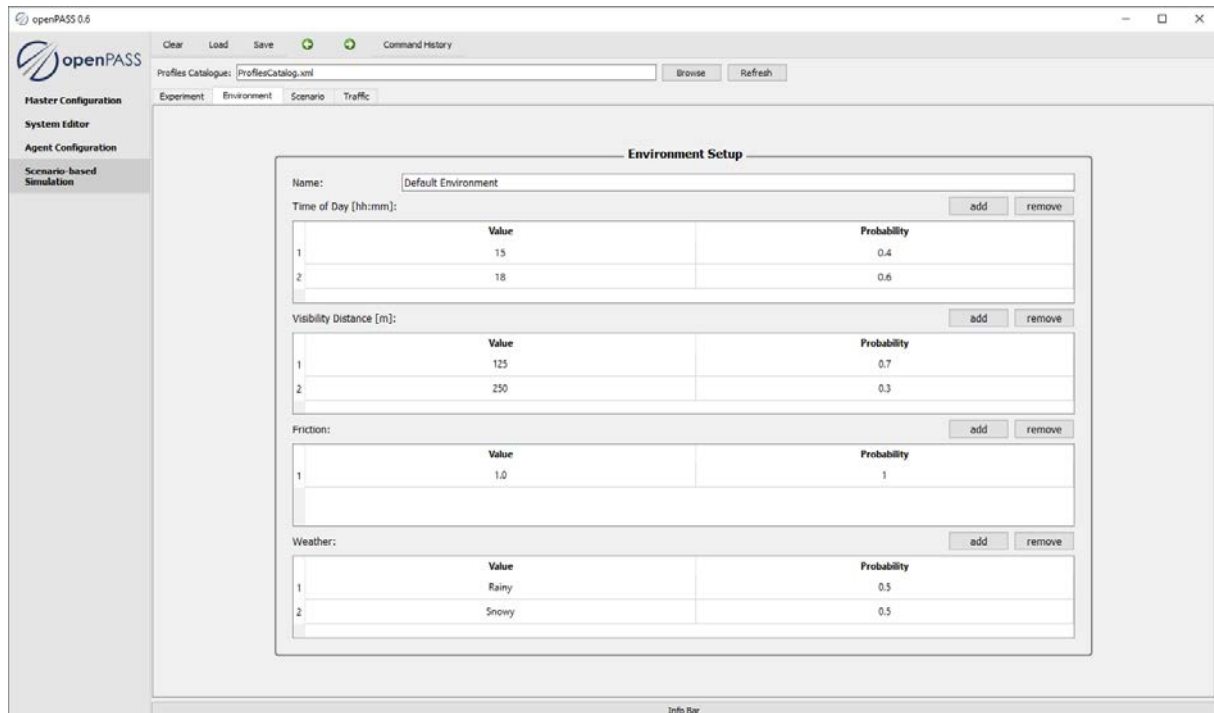
- Environment
- Scenario
- Traffic

In Addition to the mentioned undo and redo feature this plugin is also equipped with a Info bar at the bottom of the window. You can find details to the current window by clicking on it.



5.1. Environment setup

In this tab you are able to create, save and load an environment setup by clicking the corresponding buttons. Let's load the provided "slaveConfig.xml" as an example by clicking . By doing this, the Profile Catalogue uses the given "ProfilesCatalog.xml"- File. The window is now supposed to look like this:

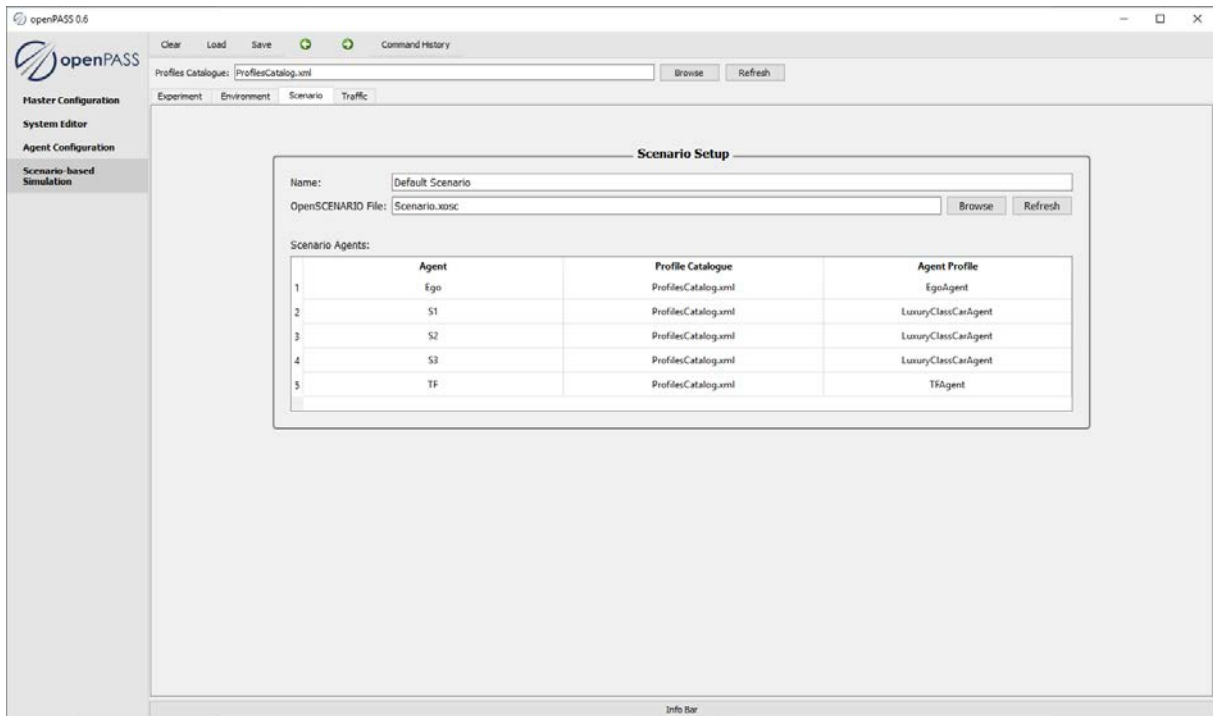


In this tab the are four Parameters listed:

- 1) Time of Day: openPASS allows you to set a Time of Day. The value has to be set in hh:mm format. (currently not used)
- 2) Visibility Distance: the distance the agent can see in meters.
- 3) Friction: friction coefficient of the road. This value is unitless. (currently not used)
- 4) Weather: weather as string. (currently not used)

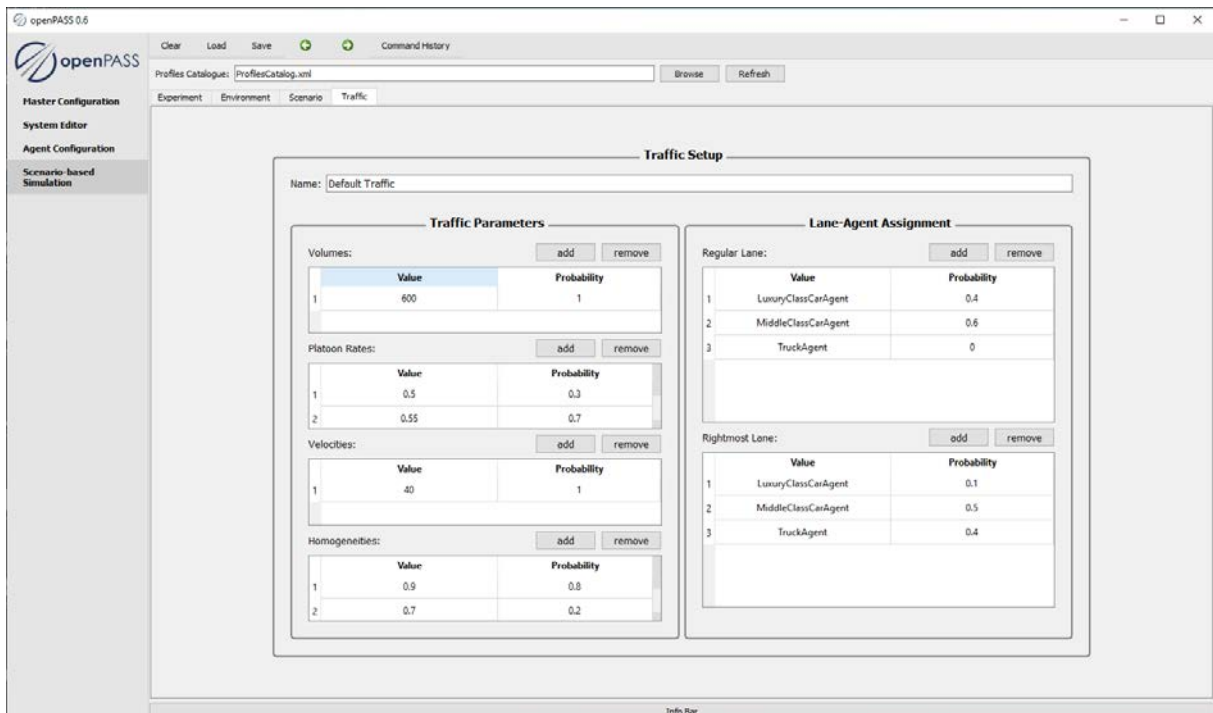
5.2. Scenario setup

In the scenario setup you load the scenario file. It describes the initial setup of the scenario. The initial position and dynamics of ego and scenario agents are listed here. The XML-File that we had loaded in the previous chapter called "slaveConfig.xml" sets default values and uses the Open Scenario File "Scenario.xosc". By clicking you can choose another Open Scenario File.



5.3. Traffic setup

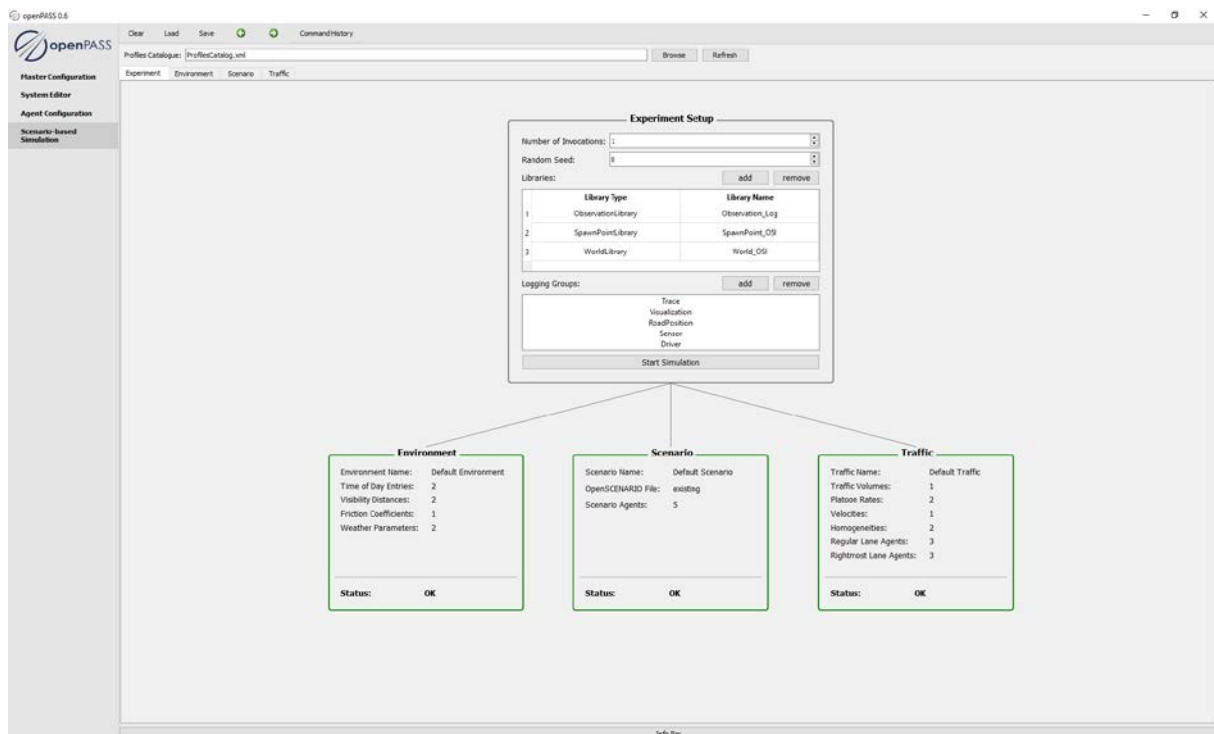
On to the Traffic setup. The traffic setup generates surrounding traffic and the setup we had loaded in 5.1. is setting default values to the parameters and assignments. Here again the probability has to be add up to 1 in every parameter.



There are 6 Parameters to specify.

- 1) Volumes: Amount of traffic in cars per hours per lane.
- 2) Platoon Rates: Chance of cars Platooning. Platooning will cause cars to spawn closer together.
- 3) Velocities: Average Velocity of right most lane in m/s.
- 4) Homogenities: Determines how much the average velocity drops per lane in km/h.
- 5) Regular Lane: Determines the agent profiles used for common agents in regular lanes.
- 6) Rightmost Lane: Determines the agent profiles used for common agents in the right most lane.

Now let's go back to the Experiment setup. All the boxes should be green as seen in the following picture.



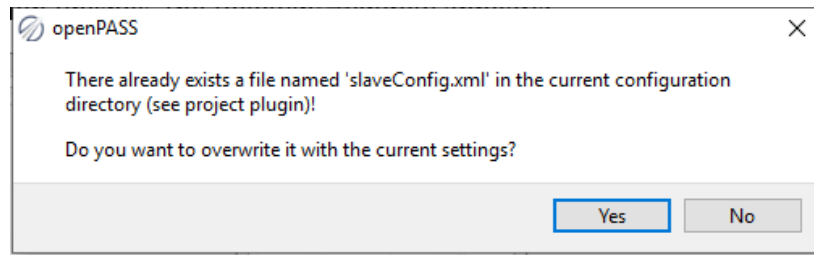
There are a couple of experiment setups you can change:

- 1) Number of Invocations: Number of invocation in the experiment. For each invocation probabilities are rerolled.
- 2) Random Seed: Random seed for the entire experiment. The seed must be within the bounds of unsigned integers.
- 3) Libraries: Name of the core module Libraries to use. If a name is not specified the default name is assumed.
- 4) Logging Groups: List of logging groups to be activated.

Now the simulation is ready to start. To start the Simulation press the Button in the experiment setup box

Start Simulation

Before the actual simulations begin, the following window appears:



This means that openPASS always need a file called “slaveConfig.xml” to run the simulation. If you want to keep the default settings provided by the Demo, you have to rename the slaveConfig-File after you have loaded it in chapter 5.1.

If you press the simulation is going to stop and the results folder stays empty.

If you press all settings in the Scenario-based Simulation plugin will be saved in the slaveConfig.xml file.

If the Simulation exits with code 0 you will find the results in the folder specified in chapter 2. Else check the OpenPassSlave.log for information why the Simulation crashed.