TraSMAPy

Release 1.0.0

João de Jesus Costa, Ana Inês Oliveira de Barros, João António C

Jun 16, 2023

INTRODUCTION:

TraSMAPy is a high-level object-oriented python API focused on both simplicities of usage by non-developers and feature completeness for the SUMO traffic simulator. The API is intended to be used by both software developers and engineers without an informatics background. Users only require minimal knowledge of the Python programming language to use the API, making it ideal for engineers of non-informatics-related fields.

The open-source code is available on GitHub and licensed under an MIT license.

CHAPTER

ONE

INSTALLING TRASMAPY

You can install TraSMAPy using the following command:

```
pip install "git+https://github.com/JoaoCostaIFG/TraSMAPy.git"
```

This will install TraSMAPy and all its dependencies.

1.1 Virtual environment

Optionally, you can create a virtual environment and install TraSMAPy inside it:

```
python3 -m venv venv
source venv/bin/activate
pip install "git+https://github.com/JoaoCostaIFG/TraSMAPy.git"
```

Don't forget to activate the virtual environment before using TraSMAPy, and to add the *venv* directory to your *.gitignore* file.

1.2 Development mode

If you want to contribute to TraSMAPy, you can clone the repository, and install it in development mode:

```
git clone "https://github.com/JoaoCostaIFG/TraSMAPy.git"
cd TraSMAPy
python3 -m venv .venv
source .venv/bin/activate
pip install -e .
```

CHAPTER

TWO

QUICKSTART WITH TRASMAPY

2.1 Generating our first network

After installing TraSMAPy, you need a network to work with. The *netgenerate* utility from SUMO can be used to generate a simple network. The following command will generate a random network:

```
netgenerate --rand -o rand.net.xml
```

The file *rand.net.xml* containes our network. Now we need a sumo configuration file to run the simulation. You can put the following in the a *rand.sumocfg* file:

```
<configuration>
    <input>
        <net-file value="rand.net.xml"/>
    </input>
</configuration>
```

Note that you can also consult the official SUMO documentation for more information about importing, building, and customizing networks.

2.2 Using TraSMAPy for the first time

With this, we are ready to write our runner script and use TraSMAPy. Create a *runner.py* file with the following content:

```
#!/usr/bin/env python

from trasmapy import TraSMAPy

def run(traSMAPy: TraSMAPy):
    """execute the TraCI control loop"""
    while traSMAPy.minExpectedNumber > 0:
        traSMAPy.doSimulationStep()

    traSMAPy.closeSimulation()


if __name__ == "__main__":
    traSMAPy = TraSMAPy("rand.sumocfg")
    run(traSMAPy)
```

Running this python script will open the sumo-gui. You'll notice that if you start the simulation (by pressing the play button), the simulation will end immediately. This is because we have not added any vehicles to the simulation and are we only ticking the simulation until there are no vehicles.

2.3 Adding vehicles to the simulation

We can add vehicles to the simulation by chaging our *runner.py* file like so:

```
def run(traSMAPy: TraSMAPy):
    for i in range(100):
        traSMAPy.users.createVehicle(f"v{i}")

    """execute the TraCI control loop"""
    while traSMAPy.minExpectedNumber > 0:
        traSMAPy.doSimulationStep()

    traSMAPy.closeSimulation()
```

This will spawn 100 vehicles in the simulation. If you run the simulation again, you'll notice that the vehicles will move around the network. You can also use the TraSMAPy API to control the vehicles. For example, you can change the speed of all vehicles like so:

```
def run(traSMAPy: TraSMAPy):
    for i in range(100):
        v = traSMAPy.users.createVehicle(f"v{i}")
        v.speed = 10

    """execute the TraCI control loop"""
    while traSMAPy.minExpectedNumber > 0:
        for vehicle in traSMAPy.users.getVehicles():
            vehicle.setSpeed(10)
        traSMAPy.doSimulationStep()

    traSMAPy.closeSimulation()
```

As you can see, vehicle, just like everything in TraSMAPy, are objects. This abstracts away the complexity of the TraCI API and makes it easier to use.

2.4 Examining the network

TraSMAPy also provides an API to examine the network. For example, you can get a sum of all CO2Emissions in all edges in the network for each simulation tick like so:

```
def run(traSMAPy: TraSMAPy):
    """execute the TraCI control loop"""
    for i in range(100):
        v = traSMAPy.users.createVehicle(f"v{i}")
        v.speed = 10

    while traSMAPy.minExpectedNumber > 0:
```

(continues on next page)

(continued from previous page)

```
        traSMAPy.doSimulationStep()

        edges = traSMAPy.network.edges
        co2Emissions = 0
        for edge in edges:
            co2Emissions += edge.CO2Emissions
        print(co2Emissions)

    traSMAPy.closeSimulation()
```

You'll probably notice that this makes the simulation run very slowly. This is because you are iterating all network edges for each simulation tick.

2.5 Introduction to queries

TraSMAPy provides a query API to make it easier to query the network and aggregate statistics. For this, there are two query mecanisms available: Python functions, and the Pyflwor query language. The Pyflwor query language is a query language that is inspired by the XQuery language, and is probably the easiest way to make simple queries. Let's convert the previous example to a Pyflwor query:

```
def run(traSMAPy: TraSMAPy):
    """execute the TraCI control loop"""
    for i in range(100):
        v = traSMAPy.users.createVehicle(f"v{i}")
        v.speed = 10

    while traSMAPy.minExpectedNumber > 0:
        traSMAPy.doSimulationStep()

        print(traSMAPy.query("return sum(<network/edges/CO2Emissions>)"))

    traSMAPy.closeSimulation()
```

Since we are interested in collecting this statistic for each simulation tick, we can register the query to be executed every simulation tick. This can be done by using the *registerQuery* method of the *TraSMAPy* class. Let's register the previous query (note that you need to provide a name for registered queries):

```
def run(traSMAPy: TraSMAPy):
    """execute the TraCI control loop"""
    for i in range(100):
        v = traSMAPy.users.createVehicle(f"v{i}")
        v.speed = 10

    traSMAPy.registerQuery("Total CO2 Emissions", "return sum(<network/edges/
↪CO2Emissions>)")

    while traSMAPy.minExpectedNumber > 0:
        traSMAPy.doSimulationStep()

        print(traSMAPy.collectedStatistics)
```

(continues on next page)

(continued from previous page)

```
    traSMAPy.closeSimulation()
```

As you can see, the *collectedStatistics* attribute of the *TraSMAPy* class contains all the statistics collected by the registered queries, organized by tick and name.

We can also take advantage of the *registerQuery* method to register a query that doesn't run every simulation tick, thus having a smaller performance hit. For example, we can register a query that runs every 10 simulation ticks:

```
def run(traSMAPy: TraSMAPy):
    """execute the TraCI control loop"""
    for i in range(100):
        v = traSMAPy.users.createVehicle(f"v{i}")
        v.speed = 10

    traSMAPy.registerQuery("Total CO2 Emissions", "return sum(<network/edges/
↪CO2Emissions>)", tickInterval=10)

    while traSMAPy.minExpectedNumber > 0:
        traSMAPy.doSimulationStep()

        print(traSMAPy.collectedStatistics)

    traSMAPy.closeSimulation()
```

2.6 The next steps

This is just a small introduction to TraSMAPy. For more information, please refer to the API reference, and to the examples on the TraSMAPy repository.

CHAPTER

THREE

API REFERENCE

This page contains auto-generated API reference documentation[1].

3.1 trasmapy

3.1.1 Submodules

trasmapy.TraSMAPy

Module Contents

Classes

TraSMAPy

Attributes

tools

trasmapy.TraSMAPy.**tools**

class trasmapy.TraSMAPy.**TraSMAPy**(*sumoCfg: str*, *useGui: bool = True*, *log: bool = False*)

property network: trasmapy.network._Network.Network

property users: trasmapy.users._Users.Users

property publicServices: trasmapy.publicservices._PublicServices.PublicServices

property control: trasmapy.control._Control.Control

property step: int

[1] Created with sphinx-autoapi

property stepLength: float

The length of one simulation step (s).

property time: float

The current simulation time (s).

property minExpectedNumber: int

The minimum number of vehicles expected to be in the simulation.

property collectedStatistics: dict[int, dict]

The accumulated statistics of the queries.

query(*query: Union[str, Callable]*) → dict

Run a query once and get its current result.

registerQuery(*queryName: str*, *query: Union[str, Callable]*, *tickInterval: int = 1*) → None

Register query to be run every tick (by default). The tickInterval param can be customized to change the frequency of the statistics collection. Results are accumulated and can be obtained through the collectedStatistics property.

doSimulationStep() → None

closeSimulation() → None

_genQueryMap() → dict

_startSimulation(*sumoCfg: str*, *useGui: bool*, *log: bool*) → None

trasmapy._IdentifiedObject

Module Contents

Classes

IdentifiedObject

class trasmapy._IdentifiedObject.**IdentifiedObject**(*id: str*)

property id: str

__repr__() → str

Return repr(self).

trasmapy._Query

Module Contents

Classes

Query

class trasmapy._Query.**Query**(*queryFunc: Callable*, *tickInterval: int = 1*)

tick() → bool

Ticks the counter. Returns True if it is time to call the query.

__call__(**args: Any*, ***kwds: Any*) → Any

trasmapy._Regulator

Module Contents

Classes

_BorgSingleton
Regulator

class trasmapy._Regulator.**_BorgSingleton**

Bases: object

_shared_borg_state

class trasmapy._Regulator.**Regulator**

Bases: *_BorgSingleton*

trasmapy._SimUpdatable

Module Contents

Classes

SimUpdatable

class trasmapy._SimUpdatable.**SimUpdatable**

abstract _doSimulationStep(**args*, *step: int*, *time: float*) → None

3.1.2 Package Contents

Classes

TraSMAPy

class trasmapy.**TraSMAPy**(*sumoCfg: str*, *useGui: bool = True*, *log: bool = False*)

property network: trasmapy.network._Network.Network

property users: trasmapy.users._Users.Users

property publicServices: trasmapy.publicservices._PublicServices.PublicServices

property control: trasmapy.control._Control.Control

property step: int

property stepLength: float

The length of one simulation step (s).

property time: float

The current simulation time (s).

property minExpectedNumber: int

The minimum number of vehicles expected to be in the simulation.

property collectedStatistics: dict[int, dict]

The accumulated statistics of the queries.

query(*query: Union[str, Callable]*) → dict

Run a query once and get its current result.

registerQuery(*queryName: str*, *query: Union[str, Callable]*, *tickInterval: int = 1*) → None

Register query to be run every tick (by default). The tickInterval param can be customized to change the frequency of the statistics collection. Results are accumulated and can be obtained through the collectedStatistics property.

doSimulationStep() → None

closeSimulation() → None

_genQueryMap() → dict

_startSimulation(*sumoCfg: str*, *useGui: bool*, *log: bool*) → None

3.2 SignalColor

3.2.1 Module Contents

Classes

SignalColor	Generic enumeration.

class SignalColor.**SignalColor**

Bases: enum.Enum

Generic enumeration.

Derive from this class to define new enumerations.

RED_LIGHT = r

red light - vehicles must stop

YELLOW_LIGHT = y

yellow light - vehicles start to decelerate if far away, otherwise they shall pass

GREEEN_LIGHT_NO_PRIORITY = g

green light, no priority - vehicle may pass the junction if there is not a vehicle using a higher priorised stream, otherwise they let it pass. They always decelerate on approach until they are within the configured visibility distance

GREEEN_LIGHT_PRIORITY = G

green light, priority - vehicle may pass the junction

GREEN_RIGHT_TURN = s

green right-turn arrow requires stopping - vehicles may pass the junction if no vehicle uses a higher priorised foe stream. They always stop before passing. This is only generated for junction type traffic_light_right_on_red.

ORANGE_LIGHT = u

red + yellow light - indicates upcoming green light. However, vehicles may not pass yet.

BROWN_LIGHT = o

off, blinking - signal is switched off, blinking indicates that vehicles have to yield

BLUE_LIGHT = O

off, no signal - signal is switched off, vehicles have the right of way

3.3 _TLPhase

3.3.1 Module Contents

Classes

TLPhase

class _TLPhase.**TLPhase**(*duration: int*, *colors: list[trasmapy.control.SignalColor.SignalColor]*, *minDur: int = -1*, *maxDur: int = -1*, *next=tuple()*, *name: str = ''*)

Bases: `traci._trafficlight.Phase`

classmethod tlPhase(*phase: traci._trafficlight.Phase*)

setState(*colors: list[trasmapy.control.SignalColor.SignalColor]*)

3.4 _TLProgram

3.4.1 Module Contents

Classes

TLProgram

class _TLProgram.**TLProgram**(*id: str*, *progType: int*, *currentPhaseIndex: int*, *phases: list[trasmapy.control._TLPhase.TLPhase] = []*, *parameters={}*)

property programId: str

Returns the id of the program.

property typeP: int

Returns the type of the program.

property currentPhaseIndex: int

Returns the index of the current phase.

property phases: list[trasmapy.control._TLPhase.TLPhase]

Returns the list of phases.

property parameters

Returns the a dictionary of parameters.

classmethod tlProg(*prog: traci._trafficlight.Logic*)

__repr__()

Return repr(self).

3.5 _Link

3.5.1 Module Contents

Classes

Link

class _Link.**Link**(*incomingId: str*, *outgoingId: str*, *viaLaneId: str*)

__repr__()

Return repr(self).

3.6 _TrafficLight

3.6.1 Module Contents

Classes

TrafficLight

class _TrafficLight.**TrafficLight**(*id: str*)

Bases: *trasmapy._IdentifiedObject.IdentifiedObject*

property state: list[trasmapy.control.SignalColor.SignalColor]

Returns the named traffic lights state.

property phaseIndex: int

Returns the index of the current phase in the currrent program.

property phaseDuration: float

Returns a default total duration of the active phase (s).

property phaseName: str

Returns the name of the current phase in the current program.

property nextSwitchTime: float

Returns the absolute simulation time at which the traffic light is schedule to switch to the next phase (s).

property timeTillNextSwitch: float

Returns the time left for the next switch (s).

property controlledLinkIds: dict[int, list[trasmapy.control._Link.Link]]

Returns a dictionary of links controlled by the traffic light, where the key is the tls link index of the connection.

property controlledLaneIds: list[str]

Returns the list of lanes which are controlled by the named traffic light. Returns at least one entry for every element of the phase state (signal index).

property programSet: list[trasmapy.control._TLProgram.TLProgram]

Returns the list of programs of the traffic light. Each progam is encoded as a TrafficLogic object.

property programId: str

"Returns the id of the current program.

property program: trasmapy.control._TLProgram.TLProgram

"Returns the current program.

getProgram(*programId: str*) → trasmapy.control._TLProgram.TLProgram

Returns the program with the given id.

getBlockingVehiclesIds(*linkIndex: int*) → list[str]

Returns the ids of vehicles that occupy the subsequent rail signal block.

getRivalVehiclesIds(*linkIndex: int*) → list[str]

Returns the ids of vehicles that are approaching the same rail signal block.

getPriorityVehiclesIds(*linkIndex: int*) → list[str]

Returns the ids of vehicles that are approaching the same rail signal block with higher priority.

setRedYellowGreenState(*colors: list[trasmapy.control.SignalColor.SignalColor]*)

Sets the phase definition. Accepts a list of SignalColors that represent light definitions. After this call, the program of the traffic light will be set to online, and the state will be maintained until the next call of setRedYellowGreenState() or until setting another program with setProgram()

turnOff()

Turns off the traffic light.

isPhaseInProgram(*programId: str*, *phaseIndex: int*) → bool

Returns true if the program with the given Id contains a phase with at the given index.

3.7 Toll

3.7.1 Module Contents

Classes

Toll

class Toll.**Toll**(*id: str*, *detectors: list[trasmapy.network._Detector.Detector]*)

Bases: *trasmapy._IdentifiedObject.IdentifiedObject*

property detectors: list[trasmapy.network._Detector.Detector]

abstract roadPricingScheme(*detectedVehicles*)

3.8 _Control

3.8.1 Module Contents

Classes

Control

class _Control.**Control**

Bases: *trasmapy._SimUpdatable.SimUpdatable*

property trafficlights: list[trasmapy.control._TrafficLight.TrafficLight]

property tolls: list[trasmapy.control.Toll.Toll]

getTrafficLight(*id: str*) → trasmapy.control._TrafficLight.TrafficLight

registerToll(*toll: trasmapy.control.Toll.Toll*) → None

getToll(*id: str*) → trasmapy.control.Toll.Toll

Returns the registered Toll with the given ID or raises KeyError if none is found.

_doSimulationStep(**args*, *step: int*, *time: float*) → None

3.9 _PublicServices

3.9.1 Module Contents

Classes

PublicServices

class _PublicServices.**PublicServices**(*users: trasmapy.users._Users.Users*)

Bases: *trasmapy._SimUpdatable.SimUpdatable*

property fleets: dict[str, trasmapy.publicservices._Fleet.Fleet]

createFleet(*fleetId: str*, *fleetRoute: Union[trasmapy.users._Route.Route, None]*, *vehicleType: trasmapy.users._VehicleType.VehicleType*, *fleetStops: list[trasmapy.users.ScheduledStop.ScheduledStop]*, *period: float*, *start: float = 0*, *end: float = INVALID_DOUBLE_VALUE*) → trasmapy.publicservices._Fleet.Fleet

Create a fleet. If the fleetRoute is None, a Route is calculated from the given fleetStops. If the fleetRoute is None, the list of FleetStops can't be empty.

getFleet(*fleetId: str*) → trasmapy.publicservices._Fleet.Fleet

_doSimulationStep(**args*, *step: int*, *time: float*) → None

3.10 _Fleet

3.10.1 Module Contents

Classes

Fleet

class _Fleet.**Fleet**(*fleetId: str*, *fleetRoute: trasmapy.users._Route.Route*, *vehicleType: trasmapy.users._VehicleType.VehicleType*, *fleetStops: list[trasmapy.users.ScheduledStop.ScheduledStop]*, *period: float*, *start: float = 0*, *end: float = INVALID_DOUBLE_VALUE*)

Bases: *trasmapy._IdentifiedObject.IdentifiedObject*, *trasmapy._SimUpdatable.SimUpdatable*

property vehicleType: trasmapy.users._VehicleType.VehicleType

property route: trasmapy.users._Route.Route

property fleetStops: list[trasmapy.users.ScheduledStop.ScheduledStop]

property end: float

property period: float

property start: float

property lastSpawnTime: float

Last spawn vehicle simulation time. -1 means no spawn yet.

property nextSpawnTime: float

Simulation time of the next vehicle spawn. Note that due to update rates, the spawn might occur later than this time. -1 means first spawn.

property spawnedVehiclesIds: list[str]

Ids of all vehicles spawned until the current simulation step.

property vehicles: list[trasmapy.users._Vehicle.Vehicle]

The vehicles that are currently present in the simulation.

_doSimulationStep(**args*, *step: int*, *time: float*) → None

_spawnVehicle(*time: float*, *users: trasmapy.users._Users.Users*)

3.11 RemoveReason

3.11.1 Module Contents

Classes

RemoveReason	Enum where members are also (and must be) ints

class RemoveReason.**RemoveReason**

Bases: enum.IntEnum

Enum where members are also (and must be) ints

TELEPORT = 0

Vehicle started teleport

PARKING = 1

Vehicle removed while parking

ARRIVED = 2

Vehicle arrived

VAPORIZED = 3

Vehicle was vaporized

TELEPORT_ARRIVED = 4

Vehicle finished route during teleport

3.12 ScheduledStop

3.12.1 Module Contents

Classes

ScheduledStop	

class ScheduledStop.**ScheduledStop**(*stop: trasmapy.network._Stop.Stop*, *duration: Union[float, str] = 0.0*, *until: Union[float, str] = INVALID_DOUBLE_VALUE*, *stopParams: list[trasmapy.users.StopType.StopType] = []*)

property stop: trasmapy.network._Stop.Stop

property stopParams: list[trasmapy.users.StopType.StopType]

property stopTypes: list[trasmapy.users.StopType.StopType]

property duration: float

property until: float

hasDuration() → bool

hasUntilTime() → bool

shiftUntilTime(*timeReference: float*) → None

Changes the until time of the ScheduledStop to start counting after the given timeReference. Useful when establishing public transport schedules: the until time of transports after the first should start counting on the moment it has departed.

_timeStr2Sec(*timeStr: str*) → float

_checkParamsValidaty()

3.13 VehicleClass

3.13.1 Module Contents

Classes

VehicleClass	Generic enumeration.

class VehicleClass.**VehicleClass**

Bases: enum.Enum

Generic enumeration.

Derive from this class to define new enumerations.

IGNORING = ignoring

PRIVATE = private

EMERGENCY = emergency

AUTHORITY = authority

ARMY = army

VIP = vip

PEDESTRIAN = pedestrian

PASSENGER = passenger

HOV = hov

TAXI = taxi

BUS = bus

COACH = coach

DELIVERY = delivery

TRUCK = truck

TRAILER = trailer

MOTORCYCLE = motorcycle

MOPED = moped

BICYCLE = bicycle

EVEHICLE = evehicle

TRAM = tram

RAIL_URBAN = rail_urban

RAIL = rail

RAIL_ELECTRIC = rail_electric

RAIL_FAST = rail_fast

SHIP = ship

CUSTOM1 = custom1

CUSTOM2 = custom2

3.14 MoveReason

3.14.1 Module Contents

Classes

MoveReason	Enum where members are also (and must be) ints

class MoveReason.**MoveReason**

Bases: enum.IntEnum

Enum where members are also (and must be) ints

AUTOMATIC = 0

Infer reason from move distance.

TELEPORT = 1

Vehicle teleports to another location

NORMAL = 2

vehicle moved normally

3.15 _VehicleStop

3.15.1 Module Contents

Classes

VehicleStop

class _VehicleStop.**VehicleStop**(*stopData: traci._vehicle.StopData*)

property stop: str

property duration: float

property until: float

property arrival: float

property intendedArrival: float

property depart: float

property stopTypes: list[trasmapy.users.StopType.StopType]

hasArrived() → bool

hasDeparted() → bool

__repr__()

Return repr(self).

3.16 _Users

3.16.1 Module Contents

Classes

Users

class _Users.**Users**

Bases: *trasmapy._SimUpdatable.SimUpdatable*

property vehicles: list[trasmapy.users._Vehicle.Vehicle]

Retrieves an object for each vehicle currently in the simulation. The API doesn't keep track of the liveness of the references returned from this method. As such, the values returned from this method should only be kept for one tick of the simulation (e.g., for querries).

property pendingVehicles: list[trasmapy.users._Vehicle.Vehicle]

Retrieves an object for each pending vehicle currently in the simulation. The API doesn't keep track of the liveness of the references returned from this method. As such, the values returned from this method should only be kept for one tick of the simulation (e.g., for querries).

property vehicleTypes: list[trasmapy.users._VehicleType.VehicleType]

getAllVehicleIds() → list[str]

getAllPendingVehicleIds() → list[str]

getAllVehicleTypeIds() → list[str]

getVehicleType(*vehicleTypeId: str*) → trasmapy.users._VehicleType.VehicleType

Retrieves an object for each vehicle type currently in the simulation.

getVehicle(*vehicleId: str*) → trasmapy.users._Vehicle.Vehicle

Retrieve a registered vehicle reference to a vehicle in the network. See createVehicle.

createVehicle(*vehicleId: str*, *route: Union[trasmapy.users._Route.Route, None] = None*, *vehicleType: trasmapy.users._VehicleType.VehicleType = VehicleType('DEFAULT_VEHTYPE')*, *personNumber: int = 0*, *personCapacity: int = 0*, *departTime: Union[str, float] = 'now'*) → trasmapy.users._Vehicle.Vehicle

Creates a registered vehicle and adds it to the network. Registered vehicles are vehicle objects whose liveness is checked (safe). If the route is None, the vehicle will be added to a random network edge. If the route consists of two disconnected edges, the vehicle will be treated like a <trip> and use the fastest route between the two edges. If depart time is the string 'now', the depart time is the same as the vehicle spawn. Negative values for depart time have special meanings:

-1: 'triggered' -2: 'containerTriggered'

getRoute(*routeId: str*) → trasmapy.users._Route.Route

createRouteFromIds(*routeId: str*, *edgesIds: list[str]*) → trasmapy.users._Route.Route

createRouteFromEdges(*routeId: str*, *edges: list[trasmapy.network._Edge.Edge]*) → trasmapy.users._Route.Route

_registerVehicle(*vehicleId*) → trasmapy.users._Vehicle.Vehicle

_doSimulationStep(**args*, *step: int*, *time: float*) → None

3.17 _VehicleType

3.17.1 Module Contents

Classes

VehicleType

class _VehicleType.**VehicleType**(*typeId: str*)

Bases: *trasmapy._IdentifiedObject.IdentifiedObject*, trasmapy.color._Colorable.Colorable

property length: float

Returns the length of the vehicles of this type (m).

property maxSpeed: float

Returns the maximum speed of the vehicles of this type (m/s).

property maxLateralSpeed: float

Returns the maximum lateral speed of the vehicles of this type (m/s).

property maxAcceleration: float

Returns the maximum acceleration of the vehicles of this type (m/s^2).

property maxDeceleration: float

Returns the maximum deceleration of the vehicles of this type (m/s^2).

property vehicleClass: trasmapy.users.VehicleClass.VehicleClass

property emissionClass: str

property shape: str

property minGap: float

Returns the offset (gap to front vehicle if halting) of vehicles of this type (m).

property minLateralGap: float

Returns the desired lateral gap of vehicles of this type at 50 km/h (m).

property width: float

Returns the width of vehicles of this type (m).

property height: float

Returns the height of vehicles of this type (m).

property personCapacity: float

Returns the total number of people that can ride in a vehicle of this type at the same time.

property scale: float

Returns the traffic scaling factor of vehicles of this type.

property color: trasmapy.color._Colorable.Color

duplicate(*cloneId: str*)

3.18 _Vehicle

3.18.1 Module Contents

Classes

Vehicle

class _Vehicle.**Vehicle**(*vehicleId: str*)

Bases: *trasmapy._IdentifiedObject.IdentifiedObject*, trasmapy.color._Colorable.Colorable

property vehicleClass: trasmapy.users.VehicleClass.VehicleClass

property vehicleType: trasmapy.users._VehicleType.VehicleType

property emissionClass: str

property shapeClass: str

property personCapacity: int

Returns the person capacity of the vehicle.

property personCount: int

Returns the number of people inside the vehicle.

property speed: float

Returns the speed of the vehicle within the last step (m/s). Error value: -2^30

property lateralSpeed: float

Returns the lateral speed of the vehicle within the last step (m/s). Error value: -2^30

property allowedSpeed: float

Returns the maximum allowed speed of the lane the vehicle is in (m/s).

property acceleration: float

Returns the acceleration in the previous time step (m/s^2).

property doRerouting: bool

Returns whether the vehicle is able to do automatic rerouting.

property edgeId: str

Returns the ID of the edge the vehicle was in the previous time step.

property laneId: str

Returns the ID of the lane the vehicle was in the previous time step.

property drivenDistance: float

Returns the distance the vehicle has already driven (m). Error value: -2^30

property CO2Emissions: float

Returns the vehicle's CO2 emissions during this time step (mg/s). To get the value for one step multiply with the step length. Error value: -2^30

property COEmissions: float

Returns the vehicle's CO emissions during this time step (mg/s). To get the value for one step multiply with the step length. Error value: -2^30

property HCEmissions: float

Returns the vehicle's HC emissions during this time step (mg/s). To get the value for one step multiply with the step length. Error value: -2^30

property PMxEmissions: float

Returns the vehicle's PMx emissions during this time step (mg/s). To get the value for one step multiply with the step length. Error value: -2^30

property NOxEmissions: float

Returns the vehicle's NOx emissions during this time step (mg/s). To get the value for one step multiply with the step length. Error value: -2^30

property fuelConsumption: float

Returns the vehicle's NOx emissions during this time step (ml/s). To get the value for one step multiply with the step length. Error value: -2^30

property electricityConsumption: float

Returns the vehicle's electricity consumption during this time step (Wh/s). To get the value for one step multiply with the step length. Error value: -2^30

property noiseEmission: float

Returns the noise generated by the vehicle (dBA). Error value: -2^30

property timeLoss: float

property color: trasmapy.color._Colorable.Color

property via: list[str]

Returns the list of IDs via edges (edge it needs to pass through in the route) for the vehicle.

static _checkVehicleExistance(*method*)

setAcceleration(*newAccel: float*, *duration: float*) → None

Sets the vehicle acceleration (m/s^2) for the given amount of time.

rerouteByTravelTime() → None

Computes a new route to the current destination that minimizes travel time. The assumed values for each edge in the network can be customized in various ways. See Simulation/Routing#Travel-time_values_for_routing. Replaces the current route by the found.

rerouteByEffort() → None

Computes a new route using the vehicle's internal and the global edge effort information. Replaces the current route by the found.

isDead() → bool

isPending() → bool

_getStopState() → int

isStoppedAnyReason() → bool

Returns whether the vehicle's is stopped state for any reason (any stopped state)

isStopped() → bool

Returns whether the vehicle's stop state is: stopped

isParking() → bool

Returns whether the vehicle's stop state is: parking

isTriggered() → bool

Returns whether the vehicle's stop state is: triggered

isContainerTriggered() → bool

Returns whether the vehicle's stop state is: containerTriggered

isAtBusStop() → bool

Returns whether the vehicle's stop state is: atBusStop

isAtContainerStop() → bool

Returns whether the vehicle's stop state is: atContainerStop

isAtChargingStation() → bool

Returns whether the vehicle's stop state is: atChargingStation

isAtParkingArea() → bool

Returns whether the vehicle's stop state is: atParkingArea

getStops() → list[trasmapy.users._VehicleStop.VehicleStop]

stop(*scheduledStop: trasmapy.users.ScheduledStop.ScheduledStop*) → None

Stops the vehicle at the given location with the given schedule. Re-issuing a stop command with the same location allows changing the duration. Setting the duration to 0 cancels an existing stop. Note that it might not be possible for a vehicle to stop at a given place because of access restrictions.

stopFor(*stop: trasmapy.network._Stop.Stop*, *duration: float*, *stopParams: list[trasmapy.users.StopType.StopType] = []*) → None

Stops the vehicle at the given position in the given edge for the given duration (s). See documentation for the stop(...) method.

stopUntil(*stop: trasmapy.network._Stop.Stop*, *until: float*, *stopParams: list[trasmapy.users.StopType.StopType] = []*) → None

Stops the vehicle at the given position in the given edge until a given simulation time (s). See documentation for the stop(...) method.

resume() → None

Resumes the march of a stopped vehicle. Throws exception if the vehicle isn't stopped.

moveTo(*laneId: str*, *pos: float*, *reason: trasmapy.users.MoveReason.MoveReason = MoveReason.AUTOMATIC*) → None

Move a vehicle to a new position along its current route.

remove(*reason: trasmapy.users.RemoveReason.RemoveReason = RemoveReason.VAPORIZED*) → None

changeTargetEdge(*targedEdge: trasmapy.network._Edge.Edge*) → None

3.19 _Route

3.19.1 Module Contents

Classes

Route	

class _Route.**Route**(*routeId: str*)

Bases: *trasmapy._IdentifiedObject.IdentifiedObject*

property edgesIds: list[str]

3.20 StopType

3.20.1 Module Contents

Classes

StopType	Enum where members are also (and must be) ints

class StopType.**StopType**

Bases: enum.IntEnum

Enum where members are also (and must be) ints

DEFAULT = 0

Stops on the lane.

PARKING = 1

Whether the vehicle stops on the road or beside.

TRIGGERED = 2

Whether a person may end the stop.

CONTAINER_TRIGGERED = 4

BUS_STOP = 8

If given, containerStop, chargingStation, edge, lane, startPos and endPos are not allowed.

CONTAINER_STOP = 16

If given, busStop, chargingStation, edge, lane, startPos and endPos are not allowed.

CHARGING_STATION = 32

If given, busStop, containerStop, edge, lane, startPos and endPos are not allowed.

PARKING_AREA = 64

//sumo.dlr.de/docs/Simulation/ParkingArea.html#letting_vehicles_stop_at_a_parking_area.

Type
Stops at a parking area. See

Type
https

OVERHEAD_WIRE = 128

3.21 Color

3.21.1 Module Contents

Classes

Color

class Color.**Color**(*r: int*, *g: int*, *b: int*, *a: int = 255*)

property colorTuple: tuple[int, int, int]

property colorTupleA: tuple[int, int, int, int]

classmethod grayscale(*gray: int*, *a: int = 255*)

classmethod hsv(*h: float*, *s: float*, *v: float*, *a: int = 255*)

classmethod hls(*h: float*, *l: float*, *s: float*, *a: int = 255*)

classmethod yiq(*y: float*, *i: float*, *q: float*, *a: int = 255*)

__repr__() → str
Return repr(self).

3.22 _Colorable

3.22.1 Module Contents

Classes

Colorable

class _Colorable.**Colorable**

abstract property color: trasmapy.color.Color.Color

3.23 _Edge

3.23.1 Module Contents

Classes

Edge

class _Edge.**Edge**(*edgeId: str*, *laneList: list[trasmapy.network._Lane.Lane]*)

Bases: *trasmapy._IdentifiedObject.IdentifiedObject*

property streetName: str

Returns the street name of the edge.

property travelTime: float

Returns the estimated travel time for the last time step on the edge (s).

property CO2Emissions: float

Sum of CO2 emissions on this edge during this time step (mg).

property COEmissions: float

Sum of CO emissions on this edge during this time step (mg).

property HCEmissions: float

Sum of HC emissions on this edge during this time step (mg).

property PMxEmissions: float

Sum of PMx emissions on this edge during this time step (mg).

property NOxEmissions: float

Sum of NOx emissions on this edge during this time step (mg).

property fuelConsumption: float

Sum of fuel consumption on this edge during this time step (ml).

property electricityConsumption: float

Sum of electricity consumption on this edge during this time step (kWh).

property vehicleCount: int

The number of vehicles on this edge within the last time step.

property vehicleMeanSpeed: float

Returns the mean speed of vehicles that were on this edge within the last simulation step (m/s).

property vehicleIds: list[str]

Returns the list of ids of vehicles that were on the edge in the last simulation step. The order is from rightmost to leftmost lane and downstream for each lane.

property occupancy: float

Returns the percentage of time the edge was occupied by a vehicle (%).

property vehicleMeanLength: float

Returns the mean length of the vehicles on the edge in the last time step (m).

property vehicleWaitingTime: float

Returns the sum of the waiting times for all vehicles on the edge (s).

property vehicleHaltCount: int

Returns the total number of halting vehicles for the last time step on the edge. A speed of less than 0.1 m/s is considered a halt.

property lanes: list[trasmapy.network._Lane.Lane]

property stops: list[trasmapy.network._Stop.Stop]

getLane(*laneId*) → trasmapy.network._Lane.Lane

getAdaptedTravelTime(*time: float*) → float

Returns the edge travel time for the given time as stored in the global container. If no such value exists, -1 is returned.

setAdaptedTravelTime(*travelTime: float*, *beginTime: float = None*, *endTime: float = None*) → None

getEffort(*time: float*) → float

Returns the edge effort for the given time as stored in the global container. If no such value exists, -1 is returned.

setEffort(*travelTime: float*, *beginTime: float = None*, *endTime: float = None*) → None

Inserts the information about the effort of the named edge valid from begin time to end time into the global edge weights container.

setMaxSpeed(*maxSpeed: float*) → None

Sets the maximum speed for the vehicles in this edge (for all lanes) to the given value.

limitMaxSpeed(*maxSpeed: float*) → None

Limits the maximum speed for the vehicles in this edge to the given value. Only affects lanes with higher maximum vehicle speeds than the given value.

setAllowed(*allowedVehicleClasses: list[trasmapy.users.VehicleClass.VehicleClass]*) → None

Set the classes of vehicles allowed to move on this edge.

setDisallowed(*disallowedVehicleClasses: list[trasmapy.users.VehicleClass.VehicleClass]*) → None

Set the classes of vehicles disallowed to move on this edge.

allowAll() → None

Allow all vehicle classes to move on this edge.

forbidAll() → None

Forbid all vehicle classes to move on this edge.

3.24 _Detector

3.24.1 Module Contents

Classes

Detector

class _Detector.**Detector**(*detectorId: str*)

Bases: *trasmapy._IdentifiedObject.IdentifiedObject*, *trasmapy._SimUpdatable.SimUpdatable*

property timeSinceLastDetection: float

Returns how many seconds elapsed since the last detection.

property laneId: str

Returns the ID of the lane where the detector is placed.

property position: float

Returns the position of the detection on its containing lane.

listen(*listener*)

Hooks into the detector. The given function will be called with the IDs of the detected vehicles there's a detection.

_doSimulationStep(**args*, *step: int*, *time: float*) → None

3.25 _BusStop

3.25.1 Module Contents

Classes

BusStop

class _BusStop.**BusStop**(*busStopId: str*)

Bases: trasmapy.network._StopLocation.StopLocation

property name: str

property startPos: float

property endPos: float

property vehicleIds: list[str]

property personIds: list[str]

stopType :trasmapy.users.StopType.StopType

3.26 _Lane

3.26.1 Module Contents

Classes

Lane

class _Lane.**Lane**(*laneId: str*, *stopList: list[trasmapy.network._Stop.Stop]*)

Bases: *trasmapy._IdentifiedObject.IdentifiedObject*

property parentEdge

property stops: list[trasmapy.network._Stop.Stop]

property linkCount: int

Returns the number of links outgoing from this lane.

property length: float

Returns the length of the named lane (m).

property width: float

Returns the width of the named lane (m).

property CO2Emissions: float

Sum of CO2 emissions on this lane in mg during this time step (mg).

property COEmissions: float

Sum of CO emissions on this lane in mg during this time step (mg).

property HCEmissions: float

Sum of HC emissions on this lane in mg during this time step (mg).

property PMxEmissions: float

Sum of PMx emissions on this lane in mg during this time step (mg).

property NOxEmissions: float

Sum of NOx emissions on this lane in mg during this time step (mg).

property fuelConsumption: float

Sum of fuel consumption on this lane in ml during this time step (ml).

property noiseEmissions: float

Sum of noise generated on this lane (dBA).

property electricityConsumption: float

Sum of electricity consumption on this edge during this time step (kWh).

property vehicleCount: int

The number of vehicles on this lane within the last time step.

property vehicleMeanSpeed: float

Returns the mean speed of vehicles that were on this lane within the last simulation step (m/s)

property vehicleIds: list[str]

Returns the list of ids of vehicles that were on this lane in the last simulation step.

property occupancy: float

Returns the total lengths of vehicles on this lane during the last simulation step divided by the length of this lane (%).

property vehicleMeanLength: float

Returns the mean length of the vehicles which were on this lane in the last step (m).

property vehicleWaitingTime: float

Returns the sum of the waiting times for all vehicles on the lane (s).

property travelTime: float

Returns the estimated travel time for the last time step on the given lane (s).

property vehicleHaltCount: int

Returns the total number of halting vehicles for the last time step on the given lane. A speed of less than 0.1 m/s is considered a halt.

property maxSpeed: float

Returns the maximum speed allowed on this lane (m/s).

_setParent(*parentEdge*) → None

limitMaxSpeed(*maxSpeed: float*) → None

Limits the maximum speed for the vehicles in this lane. Only changes the value if it needs to be lowered.

allowedVehicles() → list[trasmapy.users.VehicleClass.VehicleClass]

List of allowed vehicle classes on this lane.

disallowedVehicles() → list[trasmapy.users.VehicleClass.VehicleClass]

List of disallowed vehicle classes on this lane.

_setAllowed(*allowedVehicleClasses: list[str]*) → None

Set the classes of vehicles allowed to move on this lane.

_setDisallowed(*disallowedVehicleClasses: list[str]*) → None

Set the classes of vehicles disallowed to move on this lane.

setAllowed(*allowedVehicleClasses: list[trasmapy.users.VehicleClass.VehicleClass]*) → None

Set the classes of vehicles allowed to move on this lane.

setDisallowed(*disallowedVehicleClasses: list[trasmapy.users.VehicleClass.VehicleClass]*) → None

Set the classes of vehicles disallowed to move on this lane.

allowAll() → None

Allow all vehicle classes to move on this lane.

forbidAll() → None

Forbid all vehicle classes to move on this lane.

3.27 _Stop

3.27.1 Module Contents

Classes

Stop

class _Stop.**Stop**(*stopId: str*)

Bases: *trasmapy._IdentifiedObject.IdentifiedObject*

abstract property laneIndex: int

abstract property lane

abstract property startPos: float

abstract property endPos: float

stopType :trasmapy.users.StopType.StopType

3.28 _ParkingArea

3.28.1 Module Contents

Classes

ParkingArea

class _ParkingArea.**ParkingArea**(*parkingAreaId: str*)

Bases: trasmapy.network._StopLocation.StopLocation

property name: str

property startPos: float

property endPos: float

property vehicleIds: list[str]

stopType :trasmapy.users.StopType.StopType

3.29 _ChargingStation

3.29.1 Module Contents

Classes

ChargingStation

class _ChargingStation.**ChargingStation**(*chargingStationId: str*)

Bases: trasmapy.network._StopLocation.StopLocation

property name: str

property startPos: float

property endPos: float

property vehicleIds: list[str]

stopType :trasmapy.users.StopType.StopType

3.30 _Network

3.30.1 Module Contents

Classes

Network

class _Network.**Network**

Bases: *trasmapy._SimUpdatable.SimUpdatable*

property edges: list[trasmapy.network._Edge.Edge]

Returns a list of all edges in the network.

property lanes: list[trasmapy.network._Lane.Lane]

Returns a list of all edges in the network.

property stops: list[trasmapy.network._Stop.Stop]

Returns a list of all stops in the network.

_indexStops(*laneToStopMap*, *StopClass*, *traciModule*)

getEdge(*edgeId: str*) → trasmapy.network._Edge.Edge

Returns an object representing the edge with the given ID in the network. Raises KeyError if the given edge doesn't exist.

getLane(*laneId: str*) → trasmapy.network._Lane.Lane

Returns an object representing the lane with the given ID in the network. Raises KeyError if the given lane doesn't exist in any edge.

getStop(*stopId: str*) → trasmapy.network._Stop.Stop

Returns an object representing the lane with the given ID in the network. Raises KeyError if the given lane doesn't exist in any edge.

getDetector(*detectorId: str*) → trasmapy.network._Detector.Detector

Returns an object representing the inductionloop (E1) with the given ID in the network. Raises KeyError if the given inductionloop doesn't exist in any lane.

createLaneStop(*laneId: str*, *endPos: float = 0*, *startPos: float = INVALID_DOUBLE_VALUE*) → trasmapy.network._LaneStop.LaneStop

createEdgeStop(*edgeId: str*, *endPos: float = 0*, *startPos: float = INVALID_DOUBLE_VALUE*) → trasmapy.network._LaneStop.LaneStop

_doSimulationStep(**args*, *step: int*, *time: float*) → None

3.31 _LaneStop

3.31.1 Module Contents

Classes

LaneStop

class _LaneStop.**LaneStop**(*lane: trasmapy.network._Lane.Lane*, *endPos: float = 0*, *startPos: float = INVALID_DOUBLE_VALUE*)

Bases: trasmapy.network._Stop.Stop

property laneIndex: int

The lane index to stop at. Traci uses the pair (edge ID, lane index) for lane stops (instead of lane ID). There isn't a way to get the index of a lane in traci. As such, we use the index of the order at which lanes are found. If a user wants to, it is possible to define lanes' indexes, in the simulation's XML files, in an order that doesn't match their positions in the lane. If this happens, the lane index used for the stop can match with a different lane (in the same edge).

property lane: trasmapy.network._Lane.Lane

property startPos: float

property endPos: float

stopType :trasmapy.users.StopType.StopType

3.32 _StopLocation

3.32.1 Module Contents

Classes

StopLocation

class _StopLocation.**StopLocation**(*stopId: str*)

Bases: trasmapy.network._Stop.Stop

property parentLane

property laneIndex: int

The lane index is ignored on all stops beside LaneStops (StopType DEFAULT).

property lane

property stopTypes: trasmapy.users.StopType.StopType

abstract property name: str

abstract property vehicleIds: list[str]

stopType :trasmapy.users.StopType.StopType

_setParent(*parentLane*) → None

- genindex
- modindex

PYTHON MODULE INDEX

INDEX

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

Y