A practical manual for Vissim COM programming in Matlab – 1st edition for Vissim 5.40

1 AUTHOR:

Tamas Tettamanti
Budapest University of Technology and Eco…

25 PUBLICATIONS **72** CITATIONS

SEE PROFILE

A practical manual for Vissim COM programming in Matlab

Tamás Tettamanti

1. Vissim traffic simulation via COM interface programming

1. The purpose of Vissim-COM programming

Vissim is a microscopic road traffic simulator based on the individual behavior of the vehicles. The goal of the microscopic modeling approach is the accurate description of the traffic dynamics. Thus, the simulated traffic network may be analyzed in detail. The simulator uses the so-called psycho-physical driver behavior model developed originally by Wiedemann (1974). Vissim is widely used for diverse problems by traffic engineers in practice as well as by researchers for developments related to road traffic. Vissim offers a user friendly graphical interface (GUI) through of which one can design the geometry of any type of road networks and set up simulations in a simple way. However, for several problems the GUI is not satisfying. This is the case, for example, when the user aims to access and manipulate Vissim objects during the simulation dynamically. For this end, an additional interface is offered based on the COM which is a technology to enable interprocess communication between software (Box, 1998). The Vissim COM interface defines a hierarchical model in which the functions and parameters of the simulator originally provided by the GUI can be manipulated by programming. It can be programmed in any type of languages which is able to handle COM objects (e.g. C++, Visual Basic, Java, etc.). Through Vissim COM the user is able to manipulate the attributes of most of the internal objects dynamically.

2. The basic steps of Vissim-COM programming

The following steps formulate a general synthesis for the realization of any adaptive control logic through Vissim-COM interface:

1. One generates the overall traffic network through Vissim GUI (geometry, signal heads, detectors, vehicle inputs, etc.).
2. After choosing a programming language which allows COM interface programming, one creates the COM Client.
3. Programming of the simulation via Vissim COM with specific commands, e.g.
   - simulation setting (multiple and automated runs),
   - vehicle behavior,
   - evaluation during simulation run (online),
   - traffic-responsive signal control logic.
4. Simulation running form COM program.

To understand the Vissim-COM concept, see the figure below which depicts a part of the Vissim-COM object model. The Vissim-COM is based on a strict object hierarchy with two kinds of object types:

- collections (array, list): store individual objects; the collection names in the Vissim-COM object model are always in plural, e.g. "Links".
- containers: store a single object; the container names are always in singular, e.g. "Link".

The head of this structure is the main Vissim object. Only one main object can be defined and all other objects belong to the main object. To understand the object model, consider the following example, which represents the access to a given road link:

1. Below the main object "Vissim" you can find the "Net" object, which compass all network functionalities.
2. Collections are situated below object "Net".
3. Collection "Links" contains all the links of the network (previously defined vie Vissim GUI)
4. To access a given "Link" object, one needs to define "Vissim", "Net", and "Links" objects.
5. After accessing the given "Link", one may apply Vissim-COM methods (e.g. "GetSegmentResult"), as well as ask or set attributes (e.g. "NAME").

This example is presented now by Visual Basic Script language. This practically shows the access to "Link" 10 (after the apostrophe character you can read comments):

```
Set vis = CreateObject("Vissim.Vissim")        'define Vissim main object
Set vnet = vis.Net                             'define Net object
vis.LoadNet("D:\Example\test.inp")             'Load the traffic network
Set links = vnet.Links                         'define Links collection
Set link_10 = links.GetLinkByNumber(10)        'Ask Link 10 as an object
```
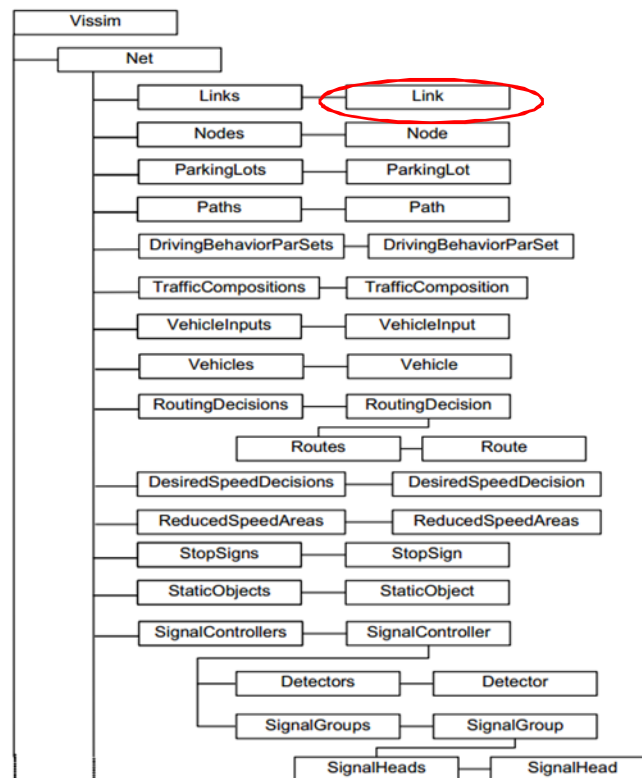

1. The Vissim-COM object model (PTV, 2012)

3. How to use Matlab for Vissim-COM programming?

In the following chapters the Vissim-COM programming is introduced by using Matlab Script language. For this the Vissim-COM interface manual (PTV, 2012) provides a great help. Although the examples of this official manual are written in Visual Basic, with a little programming knowledge, we can transform them in other programming languages

(therefore into Matlab environment) as well. The principle of COM programming is the same written in any language.

One of the main advantages of using Matlab, is the simplicity of the Matlab Script language. Another very important aspect is that, the Matlab (as a mathematical software package for practical purposes) has a lot of built-in functions. For example, optimization tasks can be solved with the help of simple Matlab commands, statistical functions can be called freely or simple matrix usage can be achieved. With the functions provided by Matlab a lot of time and energy can be saved compared to other programming languages. Therefore, if yoa are programming the Vissim traffic simulator via COM, but you also want to perform special operations (e.g. optimization), you should choose the Matlab Script as the basic language for programming Vissim-COM.

An important technical information is that before creating a Vissim-COM program, you must register the Vissim as a COM-server in your operating system (so that other applications can access Vissim-COM objects). You can do the registration after the installation of Vissim by running the "RegVissimAsCOMSvr.vbs" Visual Basic Script file in the /Exe folder of the installer folder.

.

2. Creating Vissim-COM server in Matlab

User may create a script file (extension ".m") in Matlab with the "File/New/Script" menu or with the white paper icon located in the toolbar (see figure below).
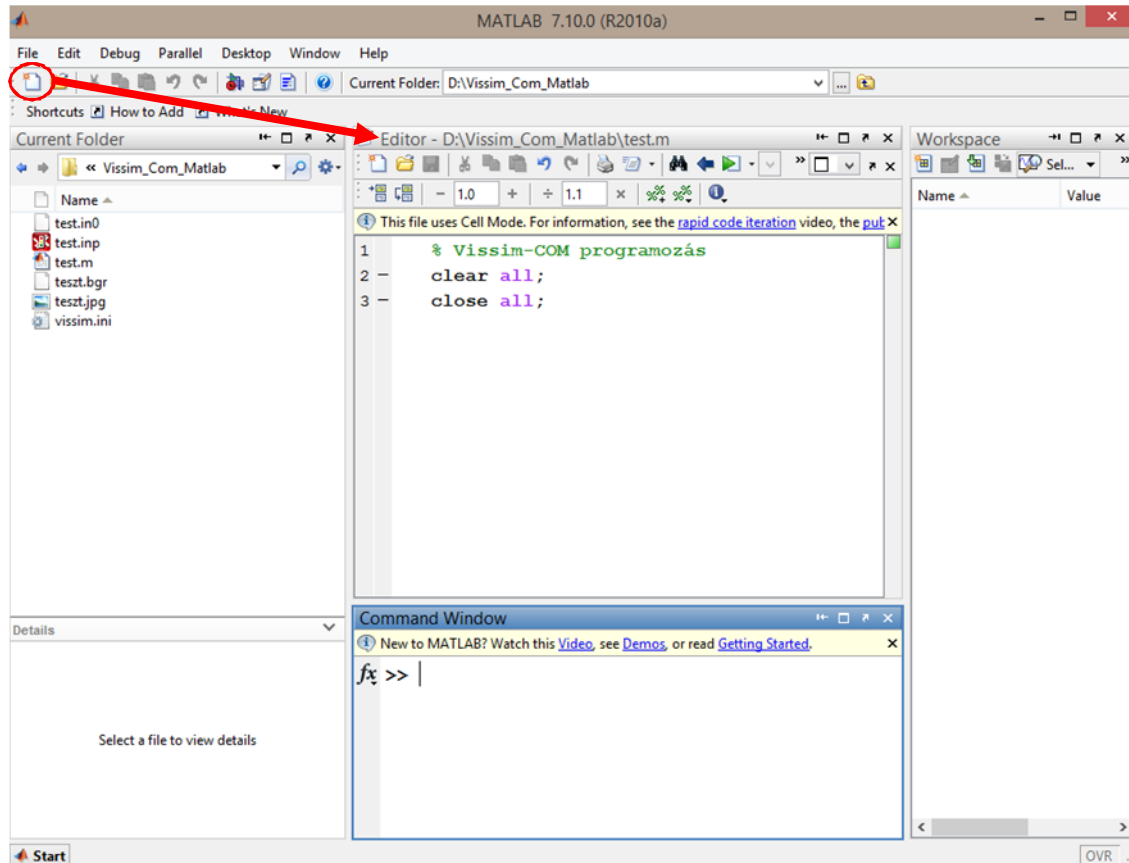
2. Creating Matlab Script file (extension ".m")

In the Matlab Script code you can write comments after the % sign.

It is useful to start ".m" file with two basic commands:

clear all;

close all;

The first deletes the contents of the Matlab workspace, i.e. the currently used variables and their values. Delete is very useful to avoid errors, e.g. the remained values of the variables in the previous executing may cause confusion. The second command closes all of the opened Matlab windows (e.g. diagrams) in one step.

Creating new COM server (other name ActiveX) is possible with the use of the Matlab command "actxserver":

vis=actxserver('Vissim.vissim')

For detailed information about a Matlab command use the Command Window and write the "help" before the command e.g.
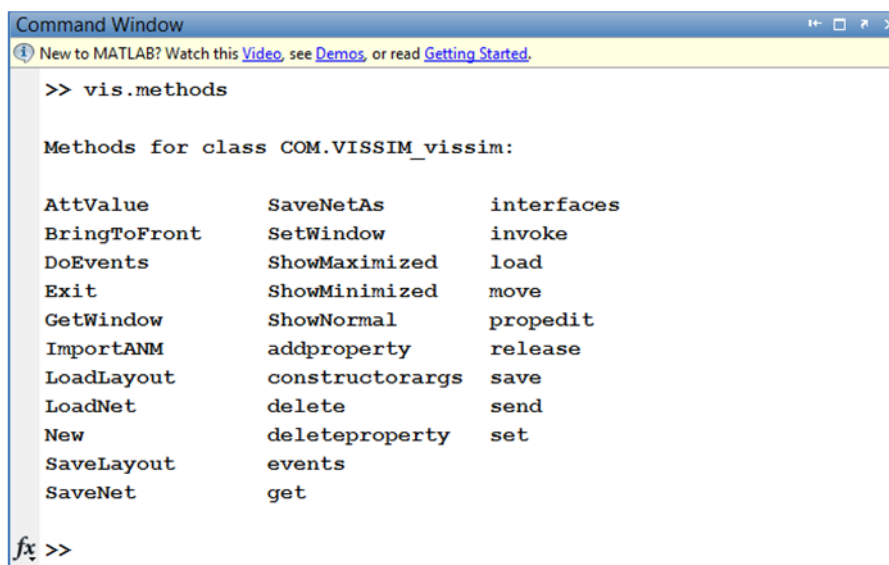
help actxserver

3. Vissim-COM methods

Object methods created via the Vissim-COM server are also accessible in the Command Window. The list of the objects can be found in the Vissim-COM Interface Manual (PTV, 2012). We can access the method list of each object if we type the object's name and the „methods" command with a dot between them into the Command Window:

{Vissim-COM object name}.methods

The method above can only be used if the object written between the curly braces was defined beforehand.

For example take a look at the following figure, which can be used for the object "vis" (main object predefined in the previous chapter), and shows the list of all methods.
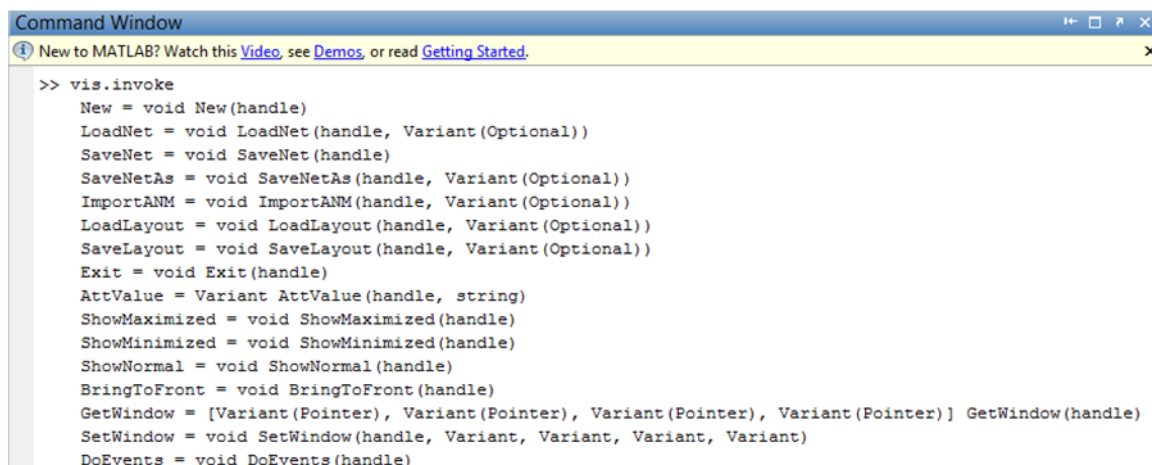
3. Getting the method list of the object "vis" in Matlab

From the method list above the "invoke" is shown below as an example.

4. The answer of Matlab Command Window to the "invoke" command of a Vissim-COM object

As can be seen, the list shows the available methods with the return value types and arguments. The „variant' is a variable type which involves several types. "void" means that the method does not have any return value.

Naturally, with this procedure the methods concerning any other Vissim-COM object can be listed analogously.

4. Loading of Vissim network

In case of Vissim-COM programming you have to create the simulation network and its elements on the graphic interface of the Vissim. You get a project file that has a .inp extension, and a "Layout" file with .ini extension. Then, you can infuse them from COM program with "Loadnet" and "LoadLayout" methods.

While using the methods, you can give the Vissim files with their direct access path that shows the destination of the files with the letter of the driver and name of the containing folders, i.e.

vis.LoadNet('D:\Vissim_Com_Matlab\test.inp');

vis.LoadLayout('D:\Vissim_Com_Matlab\vissim.ini');

There is a possibility to give a relative access path, and that is the better solution. You only have to use the "pwd" command from the Matlab, and it shows the access path of the current folder (see figure below).
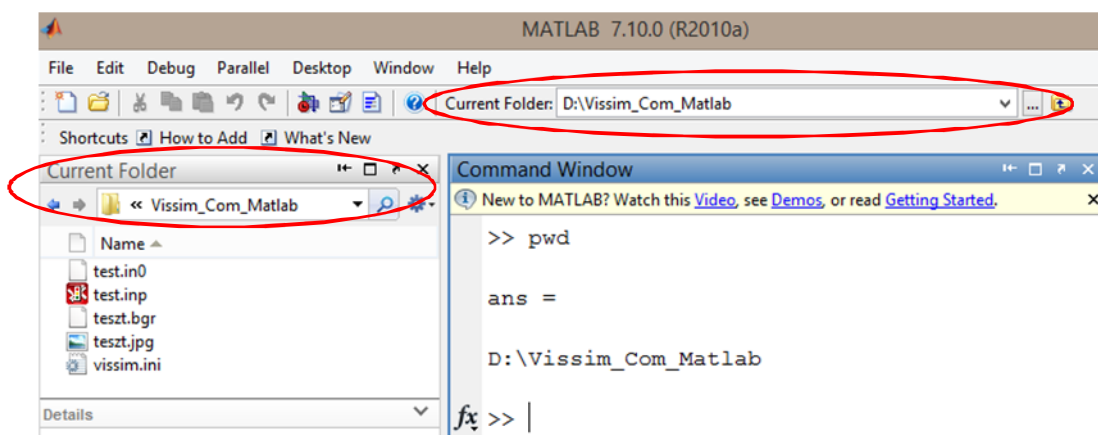
5. Using the "pwd" command in Matlab command line

You can load the network with relative access path as follows:

access_path=pwd;

vis.LoadNet([access_path '\test.inp']);

vis.LoadLayout([access_path '\vissim.ini']);

Using the relative access path is very useful if we wish to run the Vissim project on different computers. You only have to copy the project folder to the current computer and open the Matlab Script file from there. Thus, there is no need to refresh the whole path of the Vissim project folder before running the code.

5. General simulation adjustments in Vissim-COM program

Hereinafter we introduce the setting of object properties and attributes. For the sake of clarity, we describe the simulation adjustments as a specific example, but of course the method is the same with other objects as well.

For simulation settings first you have to define the "Simulation" object that can be found under the main object in the hierarchy-model of Vissim-COM (see below). We can do that via the previously defined main object called "vis":

sim=vis.Simulation;
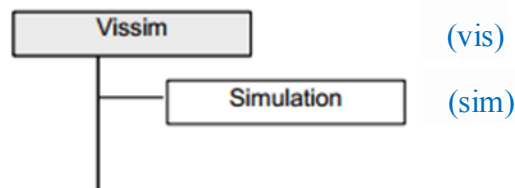
6. "Simulation" is below the main object "Vissim" in hierarchy (PTV, 2012); between the round brackets you can read the name of the object used in the sample code

1. Object properties
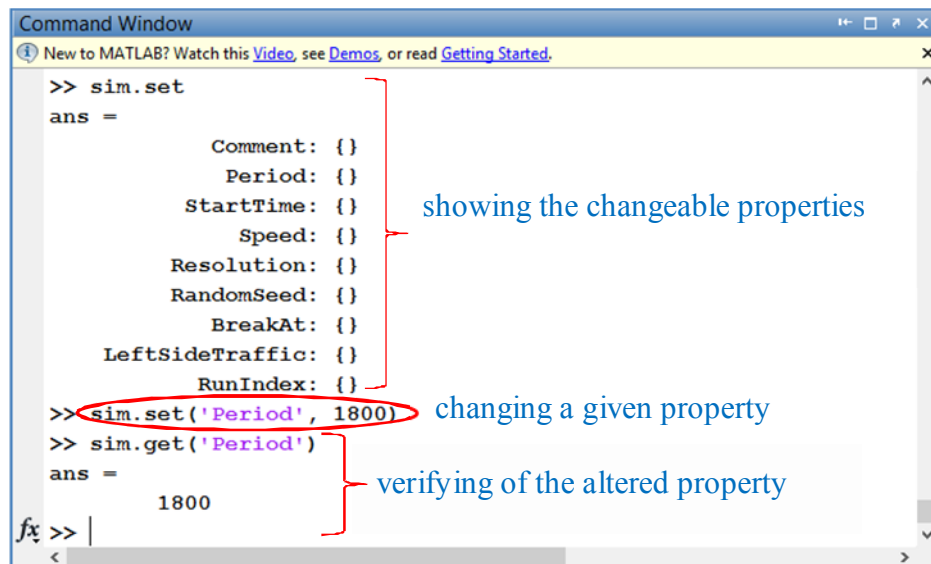
Every Vissim-COM object has properties (Property). We can query the properties of the objects with "get" method. In case of "Simulation" object they can be seen on the figure below.

7. Query of the "Simulation" object with "get" command

We can change the properties of the objects with the "set" method. Again, as an example regarding the object "Simulation", see the following figure.

8. Set of the "Simulation" object with "set" command

Of course, editing properties of the objects happens the same in Matlab Script file as in the command line.

An example to set the length of the simulation in Matlab Script file::

period_time=3600;

sim.set('Period', period_time);

As another example, we mention the "Simulation Resolution" property. This represents how many times the Vissim traffic model runs in a second during the simulation. We can change it with the following code:

step_time=3;

sim.set('Resolution', step_time);

2. Object attributes

Objects have so-called attributes ("Attribute") as well. To reach them we have to use the "AttValue" method.

Syntax of the usage of the "AttValue" method in the case of readout ("get") and for change ("set") is as follows:

sim.get('AttValue', {'attribute'});

sim.set('AttValue', {'attribute'}, { adjustable value});

Those attributes that can be written between the braces can be found in Vissim-COM Interface Manual document (PTV, 2012) where you can find a detailed attribute table to each objects. As an example below we can see the attribute table of "Simulation" object, where "R" (readable) concerns the readability and "W" (writeable) concerns the writability of the attribute.

9

R	W	Attribute	Description
✓		ELAPSEDTIME	Simulated time seconds since start of simulation [s]
✓		ISRUNNING	Simulation is running (True/False)
✓	✓	LEFTSIDETRAFFIC	0 for right-side traffic , ≠ 0 for left-side traffic
✓	✓	NUMRUNS	Number of simulation runs for a multi-run process

9. Part of attribute table of the "Simulation" object (PTV, 2012)

A part of these attributes covers the properties of the objects mentioned above, so they can be used with "AttValue" method too, e.g. the following commands are completely equivalent:

sim.set('Period', period_time);   ⇔   sim.set('AttValue', 'Period', period_time);

The remaining part of the attributes can only be handled with "AttValue" method i.e.

sim.get('AttValue', 'ElapsedTime');

sim.set('AttValue', 'NumRuns', 10);

6. Running a simulation

Using Vissim there are three ways to run simulation:

- "RunContinuous": continuous running,
- "RunSingleStep": running step-by-step, i.e. the time interval between steps will be simulated according to the "Simulation Resolution" setting,
- "RunMulti": multiple simulations in a row.

We point out the "RunSingleStep" method since this way makes easy to manipulate the simulation "online", i.e. during the simulation run (for example changing the traffic demands continuously).

"RunSingleStep" is suggested to use with "for" cycle. In the following example we run a simulation which shows the elapsed simulation time at each step („period_time" and „step_time" variables are defined previously).

```
for i=0:(period_time* step_time)
    sim.RunSingleStep;
    sim.get('AttValue', 'ElapsedTime');
end
```

While using "RunSingleStep", the "Simulation Speed" setting has no effect on the running speed of the simulation. In this case, the simulation runs step by step according to the "for" cycle, by running each „time step" on the maximum speed possible. Therefore, using the above method the simulation speed can be controlled by Matlab "pause" command (e.g. to slow down the simulation for visual observation). In the following example, a 500 ms long pause is inserted after each simulated time step:

```
for i=0:(period_time*step_time)
    sim.RunSingleStep;
    pause(0.5);
end
```

"SaveSnapshot" and the "LoadSnapshot" methods are also worth mentioning, especially when we would like to run simulations several times from a given starting point, e.g. in order to calibrate a simulation, or to set a „warm start" simulation. By applying "SaveSnapshot", such a snapshot (with „snp" file extension) is taken from the simulation that contains the states (positions of vehicles, traffic lights, etc.) of all simulation objects and conditions at the given moment. "LoadSnapshot" method can be used to load previously taken snapshots. Here follows an example Matlab Script code to it:

```
sim.SaveSnapshot([folder_path '\start.snp']);
sim.LoadSnapshot([folder_path '\start.snp']);
```

7. Traffic generation

Vissim-COM make possible to dynamically change the traffic demands, which is very useful in the following cases for example:

- to run several simulations with different traffic demands (possibly by "MultiRun" method),
- to generate varying traffic demand by following the traffic changes of a day (during the simulation run).

First of all, the "Net" object must be created, which is located below the main object in the Vissim-COM hierarchy model (see figure below). This can be achieved through the main object "vis" (already defined above):

vnet=vis.Net;
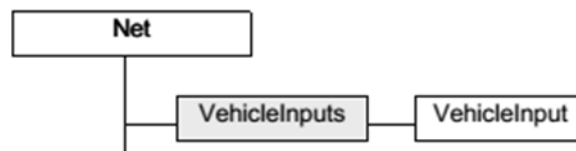
10. "VehicleInput" object in the Vissim-COM hierarchy (PTV, 2012)

Next the "VehicleInputs" collection has to be created which contains all vehicle inputs ("VehicleInput"), defined in Vissim GUI:

vehins=vnet.VehicleInputs;

Via the "VehicleInputs" collection any "VehicleInput" object becomes accessible by using the "GetVehicleInputByNumber" method, e.g.:

vehin_1=vehins.GetVehicleInputByNumber(1);

It must be mentioned that the numbers of the "VehicleInput" objects are unfortunately not visible in Vissim GUI. Therefore, we need to look for them in the Vissim project file (with ".inp" file extensions). As the project file is practically a text file, it can be opened by any text editor. By searching on the "Input", the "VehicleInput" objects can be found, and identified based on their "Link" (on which they are situated).
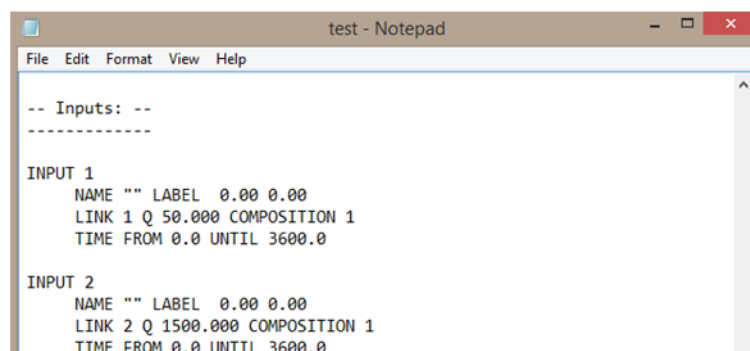
11. Extract from a Vissim project file (".inp") opened with text editor

The given "VehicleInput" object is easy to edit with the "AttValue" method (by using the attributes in figure below).

R	W	Attribute	Description
✓		ID	Identifier number
✓	✓	NAME	Name
✓		LINK	Link number
✓	✓	TIMEFROM	Time interval start [s]
✓	✓	TIMEUNTIL	Time interval end [s]
✓	✓	TRAFFICCOMPOSITION	Traffic Composition number
✓	✓	VOLUME	Volume [Veh/h]

**12. The attribute table of the "VehicleInput" object (PTV, 2012)**

An example for the modification of traffic volume attribute:

vehin_1.set('AttValue', 'Volume', 600);

13

8. Traffic signal control, detectors

Traffic light control can be programmed via COM interface as well. However, it must be noted that the previously mentioned Visvap module (flowcharts programming) or Signal Controller API interface (on C++ language) are also applicable for traffic signal programming.

The traffic signal control within Vissim-COM object model is shown in the figure below.
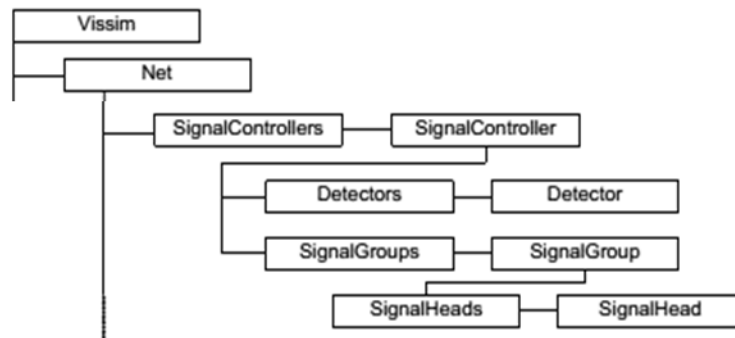
13. Components of traffic signal control within Vissim-COM object model (PTV, 2012)

Now, a simple example is provided to demonstrate traffic signal control via Vissim-COM.

A simple signalized intersection is given (shown in the figure below), where two one-way roads (a main road and a side street) meet. There are two signal groups working in the junction. By default, main road is operated with a constant green time signal. At the same time, the signal group of the side road only gets green time when the loop detector is activated. This is the so-called demand-actuated traffic signaling. The system checks the loop detector's availability in every 20 seconds. The demand-actuated stage has 20 seconds.
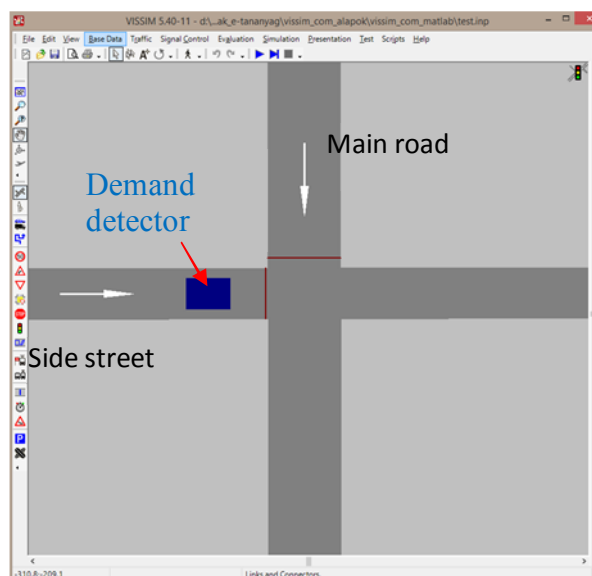
14. Simple intersection in Vissim with traffic demand actuated control

First we have to create the necessary elements in Vissim GUI:

- define signal control system in "Signal Control/Edit Controllers" menu, by choosing "Fix time" controller (later it will be operated as traffic-responsive controller from COM programe).
- create signal groups ("Signal Group") with a given signal plan (shown in the figure below),
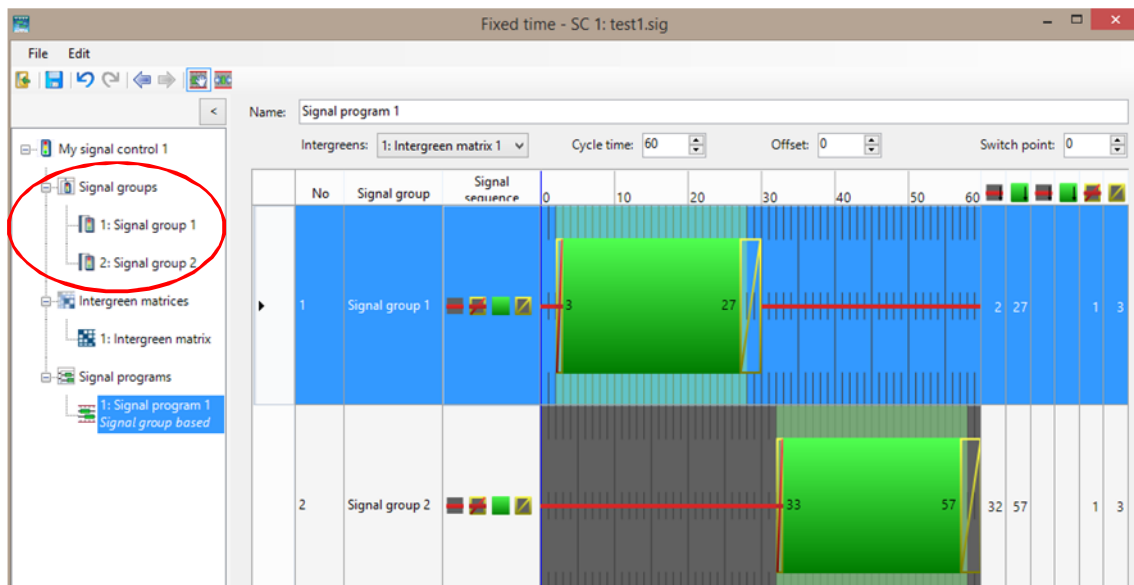
15. Create "Signal Group" in Vissim GUI with signal plan

- locate signal heads ("Signal Head") on the main and side roads
- locate demand detector on the side street ("Detector").

Then, the signal controller must be defined through Vissim-COM "SignalControllers" collection:

scs=vnet.SignalControllers;

sc=scs.GetSignalControllerByNumber(1);

Create signal group objects through "SignalGroups" collection:

sgs=sc.SignalGroups;

sg_1=sgs.GetSignalGroupByNumber(1);

sg_2=sgs.GetSignalGroupByNumber(2);

Additionally, define loop detector object for traffic demand sensing:

dets=sc.Detectors;

det_1=dets.GetDetectorByNumber(1);

The signals of the signal groups can be controlled by "State" attribute of "SignalGroup" object with the correct codes (see figure below), e.g. setting red signal for State 1:

sg_1.set('AttValue','State',1);

STATE	0 = Default (means: use the state of the external controller), 1 = Red, 2 = Redamber, 3 = Green, 4 = Amber, 5 = Off, 6 = Undefined, 7 = Flashing Amber, 8 = Flashing Red, 9 = Flashing Green, 10 = Flashing Redgreen, 11 = Greenamber, 12 = Off_red

16. "State" attribute codes of "SignalGroup" (PTV, 2012)

Status of the loop detector is requested also through the "AttValue" method by various attribute e.g.:

det_1.get('AttValue', 'Detection');

det_1.get('AttValue', 'Impulse');

det_1.get('AttValue', 'Occupancy');

det_1.get('AttValue', 'Presence');

In addition to the above, the traffic-responsive logic is created by "rem" command of Matlab (which gives back the remainder after a division of two numbers):

```
for i=0:( period_time*step_time)
  sim.RunSingleStep;
  if rem(i/step_time,20)==0                    % verifying at every 20 seconds
    demand=det_1.get('AttValue','Presence'); % verifying detector occupancy: 0/1
    if demand==1                               % demand -> demand-actuated stage
        sg_1.set('AttValue','State',1);   % main road red (1)
        sg_2.set('AttValue','State',3);   % side street green (3)
     else                                      % no demand on loop -> main road is green
        sg_1.set('AttValue','State',3);
        sg_2.set('AttValue','State',1);
    end
  end
end
```

For the sake of clarity, in the example above we neglected the intergreen times between the two phases and also we did not use transition signals (red-amber, amber). To create them further programming is necessary.

9. Evaluation while running

One of the biggest advantages of the Vissim-COM programming is the possibility of evaluation while running. Two examples are shown from the numerous evaluation options:

- evaluation of data collection points through "DataCollection" objects,
- measurement of parameters of road sections through "Link" object.

Data Collection Points can be used effectively with Vissim GUI. They can be positioned anywhere in the road network, furthermore they are suitable for measuring several parameters in the given cross section (e.g. acceleration, number of vehicles, occupation).

In order to apply the data collection points via the Vissim-COM interface, first the "Evaluation" object has to be made available as follows:

eval=vis.Evaluation;

We can reach the given data collection point through the "Data Collections" and by using the "GetDataCollectionByNumber" method:

datapoints=vnet.DataCollections;

datapoint_1=datapoints.GetDataCollectionByNumber(1);

For the evaluation of the data collection points via Vissim-COM (even while running) the "GetResult" method can be used with the appropriate parameters (see figure below).

Parameter	Description
ACCELERATION	Acceleration [m/s2] [ft/s2]. MIN, MAX, MEAN, FREQUENCIES
LENGTH	Vehicle length [m] [ft]. MIN, MAX, MEAN, FREQUENCIES
MOTORTEMPERATURE	Cooling water temperature [°C]. MIN, MAX, MEAN, FREQUENCIES
NVEHICLES	Number of vehicles. SUM
NPERSONS	Number of people. MIN, MAX, MEAN, SUM, FREQUENCIES
OCCUPANCYRATE	Occupancy rate [%]. SUM
QUEUEDELTIME	Total queue delay time [s]. MIN, MAX, MEAN, SUM, FREQUENCIES
SPEED	Speed [km/h] [mph]. MIN, MAX, MEAN, FREQUENCIES
TACHO	Total distance traveled in the network [m] [ft]. MIN, MAX, MEAN, FREQUENCIES

17. The "Data Collection Point" parameters which can be asked by the "GetResult" method (PTV, 2012)

The following code is an example for the "GetResult" method:

datapoint_1.GetResult('Speed', 'Mean', 0);

where the elements in brackets are:

- parameter to ask,

- function name,
- Vehicle Class, where "0" value includes all vehicle classes.

The indispensable condition of measurement by data collection points is (even by using Vissim-COM) that the option of „Data collection" is flagged (and appropriately configured) in the „Evaluation/Files" menu in the Vissim GUI.

In order to ask traffic parameters of a given road stretch, the road link has to be assigned through collection "Links" by calling the "GetLinkByNumber" method:

links=vnet.Links;

link_1=links.GetLinkByNumber(1);

Then, the „GetSegmentResult" method can be used for the given "Link" object to ask the traffic parameters (see figure below) even during simulation run.

Attribute	Description
DENSITY	Average density (current unit selection)
DELAY	Average relative lost time [s/s]
NVEHICLES	Number of vehicles (cumulative value of VOLUME). VOLUME must be activated within the evaluation configuration.
SPEED	Average speed (current unit selection)
VOLUME	Average volume [veh/h]

18. "Link" parameters which can be asked by "GetSegmentResult" (PTV, 2012)

The following code is an example for asking the traffic volume on Link 1:

link_1.GetSegmentResult('Volume', 0, 0.0, 1, 0);

where the elements in the brackets are:

- parameter to retrieve,
- Vehicle Class,
- given point of the road section (in meters), where the measurement starts (the measurement ends at the end of the link); if it is 0.0 then we are measuring on the whole link,
- number of the lanes,
- setting for cumulative evaluation: yes (1) or no (0).

There are two critical points of the appropriate measurement of road sections by "GetSegmentResult":

- Previously, the "Link Evaluation" option must be flagged and set appropriately in the "Evaluation/Files" menu of Vissim GUI.
- Previously, the "Link Evaluation" option must be flagged in Vissim GUI in the context menu of the road sections to measure.

10. A complete sample code for Vissim-COM programming

A sample code for Vissim-COM programming (written in Matlab) is presented below based on the examples introduced in this manual.

```matlab
%% Vissim-COM programming - example code %%
clear all;
close all;
%% Create Vissim-COM server
vis=actxserver('VISSIM.vissim');
%% Loading the traffic network
access_path=pwd;
vis.LoadNet([access_path '\test.inp']);
vis.LoadLayout([access_path '\vissim.ini']);
%% Simulation settings
sim=vis.Simulation;
period_time=3600;
sim.set('Period', period_time);
step_time=3;
sim.set('Resolution', step_time);
%% Define the network object
vnet=vis.Net;
%% Setting the traffic demands of the network
vehins=vnet.VehicleInputs;
vehin_1=vehins.GetVehicleInputByNumber(1);
vehin_1.set('AttValue', 'Volume', 1500); % main road
vehin_2=vehins.GetVehicleInputByNumber(2);
vehin_2.set('AttValue', 'Volume', 100); % side street
%% The objects of the traffic signal control
scs=vnet.SignalControllers;
sc=scs.GetSignalControllerByNumber(1);
sgs=sc.SignalGroups;
sg_1=sgs.GetSignalGroupByNumber(1);
sg_2=sgs.GetSignalGroupByNumber(2);
dets=sc.Detectors;
det_1=dets.GetDetectorByNumber(1);
%% Access to Evaluation object
eval=vis.Evaluation;
%% Access to DataCollectionPoint object
datapoints=vnet.DataCollections;
datapoint_1=datapoints.GetDataCollectionByNumber(1);
%% Access to Link object
links=vnet.Links;
link_1=links.GetLinkByNumber(1);
%% Running the simulation
for i=0:(period_time*step_time)
    sim.RunSingleStep;
    if rem(i/step_time,20)==0        % verifying at every 20 seconds
        igeny=det_1.get('AttValue','Presence'); %get detector occupancy:0/1
        if igeny==1                  % demand -> demand-actuated stage
            sg_1.set('AttValue','State',1); % main road red (1)
            sg_2.set('AttValue','State',3); % side street green (3)
        else                         % no demand on loop -> main road is green
            sg_1.set('AttValue','State',3);
            sg_2.set('AttValue','State',1);
        end
    end
    datapoint_1.GetResult('Speed','Mean',0) %get avg speed from DataPoint 1
    link_1.GetSegmentResult('Volume',0,0.0,1,0) %get traffic flow on Link 1
end
%% Delete Vissim-COM server (also closes the Vissim GUI)
delete(vis);
```

19

11. Bibliography

Box D. Essential COM, Addison-Wesley, ISBN 0-201-63446-5, 1998

PTV, Vissim-COM Interface Manual 5.4, PTV Planung Transport Verkehr AG, Germany, 2012

Wiedemann R. Simulation des Straßenverkehrsflusses Schriftenreihe des Instituts für Verkehrswesen der Universität Karlsruhe, Heft 8, 1974