

---

# **VSimRTI: Vehicle-2-X Simulation Runtime Infrastructure**

---

## **User Documentation**

---

Michalis Adamidis	Manuel Bock	Roland Cristea	Sebastian Dunkel
Jan Henning	Adrian Herrmann	Jiajun Hu	Karl Hübner
Sven Erik Jeroschewski	Franz Kage	Bernard Ladenthin	Nico Naumann
Robert Protzmann	Tobias Queck	Stefan Reichel	David Rieck
Stefan Rulewitz	Alexander Scheck	Erik Schleiff	Björn Schünemann
Julia Ullrich	Sascha-Gaetano Urso	Michael Weiner	

Version 18.0  
April 24, 2018

## Abstract

The Vehicle-2-X Simulation Runtime Infrastructure (**VSimRTI**) enables the preparation and execution of Vehicle-2-X (**V2X**) simulations. It is a flexible system which simulates traffic flow dynamically. **VSimRTI** couples different simulators and thus allows the simulation of the various aspects of intelligent transportation systems. The easy integration and exchangeability of simulators enables the utilization of the most relevant simulators for a realistic presentation of vehicle traffic, emissions, wireless communication and the execution of **V2X** applications.

## The developer alliance

The developer alliance consists of Fraunhofer-Institut für Offene Kommunikationssysteme (**FOKUS**), Daimler Center for Automotive Information Technology Innovations (**DCAITI**) and Automotive Services and Communication Technologies (**ASCT**).

## Contact information

### VSimRTI Mailing List (developer team)

 [vsimrti@fokus.fraunhofer.de](mailto:vsimrti@fokus.fraunhofer.de)

### VSimRTI: Smart Mobility Simulation

 <http://www.dcaiti.tu-berlin.de/research/simulation>

### Fraunhofer FOKUS: ASCT Competence Center

 <https://www.fokus.fraunhofer.de/asct>

### DCAITI

 <http://www.dcaiti.tu-berlin.de>



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Overview	1
1.2	Download and install	1
1.3	Update	2
1.4	License	2
1.5	Concept	3
1.5.1	Federates and Ambassadors	3
1.5.2	Scenario configuration	3
<b>2</b>	<b>Run simulations</b>	<b>4</b>
2.1	Run a single simulation via CLI	4
2.1.1	VSimRTIEmbeddedStarter	5
2.2	Results	7
2.2.1	Logging	7
2.3	Run a simulation set via CLI	8
<b>3</b>	<b>Simulators</b>	<b>9</b>
3.1	Kinds of simulators	9
3.1.1	Network simulators	9
3.1.2	Traffic simulators	10
3.1.3	Application simulators	10
3.1.4	VSimRTI communication simulators	10
3.1.5	Environment simulators	11
3.1.6	Electricity simulators	11
3.2	OMNeT++	12
3.2.1	omnetpp-ambassador folder structure	12
3.2.2	Installation	13
3.2.3	Installation in Docker environment	19
3.2.4	Configuration	15
3.3	ns-3	16
3.3.1	ns3-ambassador folder structure	17
3.3.2	Installation	17
3.3.3	Installation in Docker environment	19
3.3.4	Configuration	20
3.4	SUMO	22
3.4.1	sumo-ambassador folder structure	22
3.4.2	Installation	22
3.4.3	Configuration	23
3.4.4	Using the SUMO GUI with VSimRTI	23
3.4.5	Routes and Vehicle Types	23
3.4.6	Further information	24
3.4.7	VSimRTI specific information	24
3.5	VSimRTI Application Simulator	25
3.5.1	applicationNT-ambassador folder structure	25
3.5.2	Installation	25
3.5.3	Libraries	25
3.5.4	Basic functionality	25
3.5.5	Gather information in an application	26

3.5.6	Equipped units	26
3.5.7	Technical details	26
3.5.8	Configuration	26
3.5.9	Development of applications	27
3.5.10	Debugging of applications	29
3.5.11	Internal ETSI applications	30
3.5.12	Sending Cooperative Awareness Messages (CAM)	31
3.5.13	Access SUMO TraCI from applications	32
3.5.14	Navigation	32
3.5.15	Mapping 3	33
3.6	VSimRTI Cellular Simulator	36
3.6.1	cell2-ambassador folder structure	36
3.6.2	Installation	37
3.6.3	Configuration	37
3.6.4	Operation	40
3.7	VSimRTI Simple Network Simulator	41
3.7.1	sns-ambassador folder structure	41
3.7.2	Installation	41
3.7.3	Configuration	41
3.7.4	Operation	42
3.8	VSimRTI Eventserver	44
3.8.1	eventserver-ambassador folder structure	44
3.8.2	Installation	44
3.8.3	Configuration	44
3.9	VSimRTI Battery Simulator	45
3.9.1	battery-ambassador folder structure	45
3.9.2	Installation	45
3.9.3	Introduction	45
3.9.4	Vehicle model	45
3.9.5	Battery model	45
3.9.6	Environment model	46
3.9.7	Sample configuration	46
<b>4</b>	<b>Tutorial Tiergarten</b>	<b>47</b>
4.1	Mapping Configuration	48
4.2	Inter-Vehicle Communication	49
4.3	Intra-Vehicle Communication	53
4.3.1	The traffic light application	54
4.4	Interpretation of simulation results	55
<b>5</b>	<b>Tutorial Barnim</b>	<b>56</b>
5.1	Overview	56
5.2	Mapping configuration	57
5.3	VSimRTI configuration	58
5.4	Applications	58
5.4.1	DEN-Message handling	59
5.4.2	Cellular Communication	59
5.5	Conclusion	60
<b>6</b>	<b>Tutorial Traffic Lights</b>	<b>61</b>
6.1	Use scenario-convert to export traffic lights	61
6.2	Determine the traffic lights that should be equipped with applications	62
6.3	Configure the mapping for traffic lights	62
6.4	The Traffic Light API	63
<b>7</b>	<b>Tutorial LuST</b>	<b>64</b>
7.1	Execute the LuST scenario with VSimRTI	65

7.2	Current limitations	65
<b>8</b>	<b>Create a new scenario</b>	<b>66</b>
8.1	Road network data	66
8.1.1	Using OpenStreetMap data	66
8.2	Vehicles and Routes	68
8.3	VSimRTI	69
<b>9</b>	<b>Visualizers</b>	<b>72</b>
9.1	Kinds of Visualizers	72
9.2	VSimRTI WebSocket Visualizer	73
9.3	VSimRTI File Visualizer	74
9.3.1	Configuring the File Visualizer	74
9.4	VSimRTI Integrated Test and Evaluation Framework (ITEF)	78
<b>10</b>	<b>Run simulation series</b>	<b>80</b>
10.1	Simulation Runner	80
10.1.1	Usage	80
10.1.2	Configuration	81
10.1.3	Parametrization	83
10.1.4	Additional Information	85
<b>11</b>	<b>Additional tools</b>	<b>86</b>
11.1	scenario-convert	86
<b>12</b>	<b>VSimRTI configuration</b>	<b>90</b>
12.1	Overview	90
12.2	Federates configuration	90
12.2.1	Federate priorities	91
12.3	Host configuration	91
<b>13</b>	<b>Additional information</b>	<b>92</b>
13.1	PATH configuration	92
<b>14</b>	<b>List of Acronyms</b>	<b>94</b>
<b>A</b>	<b>VSimRTI deployment</b>	<b>96</b>
A.1	VSimRTI Folder Structure	96
A.2	File listings	98
<b>B</b>	<b>Example scenario Barnim</b>	<b>108</b>
B.1	Folder Structure	108
B.2	File listings	109
<b>C</b>	<b>Example scenario Tiergarten</b>	<b>123</b>
C.1	Folder Structure	123
C.2	File listings	124
<b>D</b>	<b>Package Additional Examples</b>	<b>126</b>
<b>E</b>	<b>Configuration Schemata</b>	<b>128</b>
<b>F</b>	<b>References</b>	<b>134</b>

# 1 Introduction

## 1.1 Overview

This documentation is part of the current release of [VSimRTI](#) and aims at supporting users in their first steps with the software. We try to explain most of [VSimRTI](#)'s features and configurations by providing two tutorials named [Barnim](#) (see chapter 5) and [Tiergarten](#) (see chapter 4).

If you need more information regarding [VSimRTI](#) or a specific configuration, we ask you to take a look at the [Doxygen](#) documentation provided alongside the current release. Documentation is available for all [JSON](#) style configs. Apart from that the appendix provides information for the respective [XML](#) style configurations.

If you have any questions or need further support, please feel free to contact our support team via our [mailing list](#). Please attach relevant log files and scenario files to your inquiry. We look forward to receiving your feedback concerning further improvements of the [VSimRTI](#) documentation as well as the installation and configuration process.

### **VSimRTI at a glance**

[VSimRTI](#) was built to provide users with the flexibility to perform various [V2X](#) simulations with their own choice of simulators. In order to provide the flexibility to exchange a simulator without changing the underlying infrastructure, [VSimRTI](#) offers several interfaces for the integration of different simulators, e.g. for network, traffic, and environment simulations. For the synchronization and the communication purposes, the implemented infrastructure uses common concepts defined in the Institute of Electrical and Electronics Engineers ([IEEE](#)) standard for modelling and simulation ([M&S](#)) High-Level Architecture ([HLA](#)). Thus, the runtime infrastructure [VSimRTI](#) allows a flexible combination of time-discrete simulators for [V2X](#) simulations. Based on the (possibly differing) requirements of a specific scenario, arbitrary simulators can be added to [VSimRTI](#) and are executed together.

[VSimRTI](#) is written in the programming language Java and deployed as Java Archive ([JAR](#)) files. Consequently, a compatible Java Runtime Environment ([JRE](#)) for your operating system must be installed to execute [VSimRTI](#). Currently, Java version 8 is required to run [VSimRTI](#).

## 1.2 Download and install

- Download `vsimrty-bin-18.0.zip` from the [DCAITI website](#) .
- The package `vsimrty-bin-18.0.zip` has to be extracted to an arbitrary path. This installation path is referenced as `vsimrty` throughout the entire document.

## 1.3 Update

In order to update [VSimRTI](#) to a new version, please perform the following steps manually:

- Backup your personal simulation scenarios from [VSimRTI](#)'s scenarios directory.
- Remove your old [VSimRTI](#) installation completely.
- Install [VSimRTI](#) according to the previous section.
- Copy your simulation scenarios back to [VSimRTI](#)'s scenarios directory.
- Migrate your scenarios to individual changes in the new [VSimRTI](#) version according to the [vsimrti-conversion-guide-18.0.pdf](#) , which can be found on the [DCAITI website](#) .

## 1.4 License

The licensing mechanism was introduced for a better release control. It improves user support whilst helping our [developer team](#) to deactivate outdated releases which cannot be maintained anymore. Every user needs to be registered at the license server to obtain a license.

1. Running the `firstStart` script in the [VSimRTI](#) root folder causes the file `vsimrti/systeminfo.txt` to be generated. This file is also generated when [VSimRTI](#) is run without a valid license file or an invalid user.
2. The created file contains basic information in plain text about the machine on which [VSimRTI](#) was executed and is used to identify the user. The system info needs to be sent to the [VSimRTI mailing list](#). A staff member registers the new user at the [VSimRTI](#) license server as soon as possible, usually within one workday. When your license is active, an email containing your user id will be sent to you.
3. The following information is stored to identify your machine:
  - central processing unit ([CPU](#)) model id
  - [CPU](#) cores
  - [CPU](#) architecture
  - Media Access Control ([MAC](#)) addresses
  - operating system name
  - operating system version
  - random-access memory ([RAM](#)) size
  - sockets
  - hashed user id
  - [VSimRTI](#) version

4. Your license will be activated directly after confirmation by the [VSimRTI](#) team. On the next [VSimRTI](#) run with an available Internet connection to the [VSimRTI](#) license server a valid license is generated and a local copy is stored in the [VSimRTI](#) folder.
5. The local license files `vsimrti/vsimrti-license.lcs` and `vsimrti/vsimrti.dat` are valid for the next 14 days. In this period no Internet connection is needed for the execution of [VSimRTI](#). Every time when an Internet connection to the license server can be established, the local license file will be renewed for another 14 days.

**i Notice:** You should not backup the license files. If you have a registered account and a valid license is present on the license server, you will always receive a new license file.

**i Notice:** If you encounter any problems with corrupt license files, you can also use the `firstStart` script to reset the local license files and retrieve a new copy during the next run.

**i Notice:** If the `firstStart` script does not result in a `vsimrti/systeminfo.txt` file, please check if you have sufficient rights on your machine. You can also try to start the `vsimrti` script instead, by calling `vsimrti.sh -u <YOUR-EMAIL-ADDRESS> -s Barnim`. This command will always fail, due to the missing license, however, it may generate the required `systemInfo.txt`.

## 1.5 Concept

In contrast to existing fixed simulator couplings [VSimRTI](#) allows easy integration and exchangeability of simulators. Depending on your specific requirements, the high flexibility of [VSimRTI](#) enables you to couple the most appropriate simulators for a realistic presentation of vehicle traffic, emissions, wireless communication, and the execution of [V2X](#) applications.

### 1.5.1 Federates and Ambassadors

[VSimRTI](#) uses the federate-ambassador concept inspired by the concept of the [HLA](#). Using this concept, it is possible to couple different simulation systems with a remote control interface. Attaching an additional simulator only requires implementation of the appropriate ambassador interface and the possibility to execute its specified commands.

### 1.5.2 Scenario configuration

Any scenario needs to be configured with a configuration file. The specific path to this file is `vsimrti/scenarios/<scenarioName>/vsimrti/vsimrti_config.xml`.



## 2 Run simulations

**Notice:** Install the required simulators for the specific scenario and have a valid license (see section 1.4).

### 2.1 Run a single simulation via CLI

To run a single simulation via Command Line Interface (CLI) calling the **VSimRTI** start script with the following command line arguments. The **VSimRTI** start script supports following arguments.

- c, --configuration The primary **VSimRTI** configuration file which is scenario dependent and located in the according scenario folder. This file transitively includes other necessary configuration files. Usually you will use the file `vsimrti/scenarios/<scenarioName>/vsimrti/vsimrti_config.xml`.
- s, --scenario If the **VSimRTI** configuration file of your scenario is located in the default scenario directory of **VSimRTI** (i.e. in `vsimrti/scenarios/<scenarioName>/vsimrti/vsimrti_config.xml`), this option can be used instead of the -c option by passing only the scenario name (-s <scenarioName>).
- u, --userid The id of the **VSimRTI** user which was assigned with the authentication for the **VSimRTI** release area. For better release control and support of **VSimRTI** users, a user id is needed to start a simulation. The user id is connected to a dedicated **VSimRTI** license.
- w, --watchdog-interval The interval of the internal alive check (in seconds) which is used by **VSimRTI** to detect execution stalls. This parameter is not mandatory and it is also possible to turn off the watchdog (-w 0) for debug sessions.
- o, --loggingLevelOverride Override all specified logging-levels. This option is useful for debugging simulations. For example logging every possible event would be done with -o TRACE.
- g, --performance-GUI Opens a GUI Panel which provides several charts regarding the current simulation run, such as the progression of the Real Time Factor or memory consumption of the simulation.
- b, --realtimeBrake With this parameter, the simulation will be slowed down to a desired Real Time Factor, if possible. When simulations already run slower than real time, this factor will have no effect. For example, use -b 1 to execute the simulation in real time.
- v, --start-visualizer Opens a page in your default browser which visualizes all vehicle movements of the simulation on a map. This option only works, if your scenario has configured the websocket visualizer (see section 9.2).

## GNU/Linux

Listing 2.1: VSimRTI GNU/Linux start command.

---

```
./vsimrti.sh -c ./scenarios/<scenario name>/vsimrti/vsimrti_config.xml -u userid
```

---

## Microsoft Windows

Listing 2.2: VSimRTI Microsoft Windows start command.

---

```
vsimrti.bat -c .\scenarios\<scenario name>\vsimrti\vsimrti_config.xml -u userid
```

---

### 2.1.1 VSimRTIEmbeddedStarter

The VSimRTIEmbeddedStarter is a customizable Java start program. You can also use this program as template to embed VSimRTI in your Java application. The VSimRTIEmbeddedStarter is much more powerful than the start scripts. E.g. the VSimRTIEmbeddedStarter can search in configuration files for the given scenario id. The embedded starter has a user friendly CLI and many options.

Listing 2.3: VSimRTI VSimRTIEmbeddedStarter help command.

---

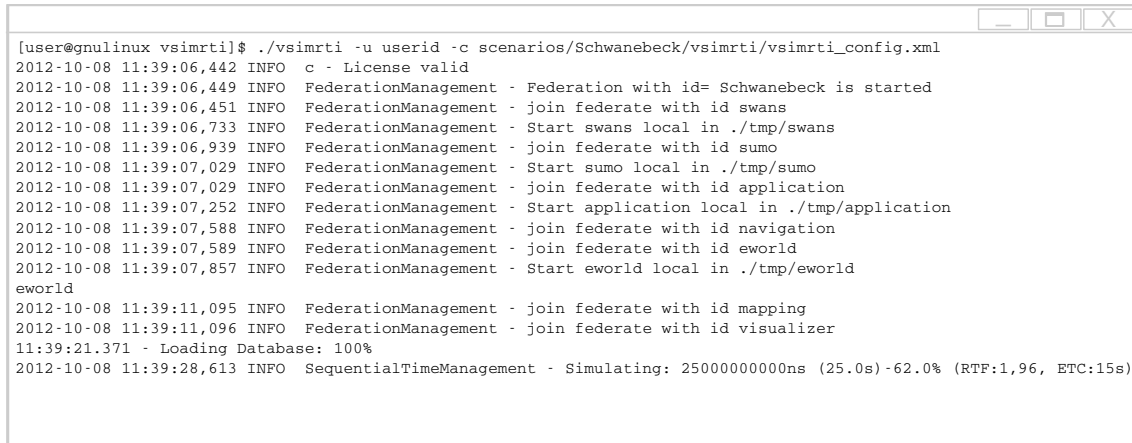
```
usage: java -jar VsimrtiEmbeddedStarter.jar
-b,--realtimeBrake <REALTIMEFACTOR> Set value for real time brake.
-c,--config <PATH> Path to Scenario configuration file
                          (vsimrti_config.xml). Can be used instead
                          of "-s" parameter. (mandatory).
--classpath Class paths. Default is "." (working
                          directory).
-d,--defaults-file <PATH> Path to VSimRTI configuration file
                          (default: etc/defaults.xml)
-e,--externalWatchDog <PORTNUMBER> Specific external watchdog port number
-g,--performance-GUI Opens performance GUI
-h,--hosts <PATH> Path to host configuration file (default:
                  etc/hosts.json)
--help Shows this help information.
-l,--logger <PATH> Path to logback configuration file
                  (default: etc/logback.xml)
--logback-path <PATH> Use specific logback.xml
-o,--loggingLevelOverride <LOGLEVEL> Overrides the log level to new value (e.g.
                  DEBUG)
-p,--process-builder Start VSimRTI using ProcessBuilder.
--pipe Use pipe redirect (only for
--process-builder
--print-jar-files Print the found jar files.
--print-parameter Print the parameter before start.
-s,--scenario <NAME> The name of the VSimRTI scenario. Can be
                  used instead of "-c" parameter.
                  (mandatory)
--simrun Use the VSimRTI simulation runner.
--strict-configuration-path The starter pass-through the configuration
                  path as it is.
-u,--user <USERID> Your UserId (mandatory). Please read the
                  user manual if you do not have a UserId
                  yet.
-v,--start-visualizer Starts the web socket visualizer.
--verbose Print information at start.
--version Print the version information and exits.
-w,--watchdog-interval <SECONDS> Kill VSimRTI process after n seconds if a
                  federate is not responding. 0 disables the
                  watchdog. (default: 30)
```

---

Listing 2.4: VSimRTI VSimRTIEmbeddedStarter start command.

```
java -jar VsimrtiEmbeddedStarter-18.0.jar -u userid -n scenarioId
```

While VSimRTI is running, it prints some information on the command line:



```
[user@gnulinux vsimrti]$ ./vsimrti -u userid -c scenarios/Schwanebeck/vsimrti/vsimrti_config.xml
2012-10-08 11:39:06,442 INFO c - License valid
2012-10-08 11:39:06,449 INFO FederationManagement - Federation with id= Schwanebeck is started
2012-10-08 11:39:06,451 INFO FederationManagement - join federate with id swans
2012-10-08 11:39:06,733 INFO FederationManagement - Start swans local in ./tmp/swans
2012-10-08 11:39:06,939 INFO FederationManagement - join federate with id sumo
2012-10-08 11:39:07,029 INFO FederationManagement - Start sumo local in ./tmp/sumo
2012-10-08 11:39:07,029 INFO FederationManagement - join federate with id application
2012-10-08 11:39:07,252 INFO FederationManagement - Start application local in ./tmp/application
2012-10-08 11:39:07,588 INFO FederationManagement - join federate with id navigation
2012-10-08 11:39:07,589 INFO FederationManagement - join federate with id eworld
2012-10-08 11:39:07,857 INFO FederationManagement - Start eworld local in ./tmp/eworld
eworld
2012-10-08 11:39:11,095 INFO FederationManagement - join federate with id mapping
2012-10-08 11:39:11,096 INFO FederationManagement - join federate with id visualizer
11:39:21.371 - Loading Database: 100%
2012-10-08 11:39:28,613 INFO SequentialTimeManagement - Simulating: 25000000000ns (25.0s) - 62.0% (RTF:1,96, ETC:15s)
```

Figure 2.1: Output of VSimRTI while running

The current simulation progress is shown in the following format.

- current wallclock time
- log level
- unit name
- current simulation time in [ns] and [s]
- progress in %
- Real Time Factor (RTF)
- Estimated Time to Completion (ETC)

The **RTF** is the ratio of simulated time to simulation duration in wallclock time, e.g. a real time factor greater than 1 means, the simulation is running faster than real time. Both **RTF** and **ETC** are calculated based on the performance of the last five seconds of the simulation and should only give a rough overview, how long a simulation can take. Depending on the simulation setup, the values can differ heavily between start and end of a simulation.

## Gather results

To get results from a simulation, the combined simulators have to be properly configured or a federate has to be integrated that generates global results. For detailed information and visualization possibilities, e.g. the FileVisualizer for detailed simulation data.

## 2.2 Results

VSimRTI generates logfiles for each simulation run. Logs are generated for the ambassadors of each coupled federate respectively simulator and for VSimRTI itself.

The logfiles are stored in the folder `vsimrti/logs/log-<timestamp>`. For each simulation run a new folder is created.

In addition to standard logging output for each federate there is a `statistics.csv` file which contains detailed information about sent and received VSimRTI messages. This file can be used to trace a simulation run.

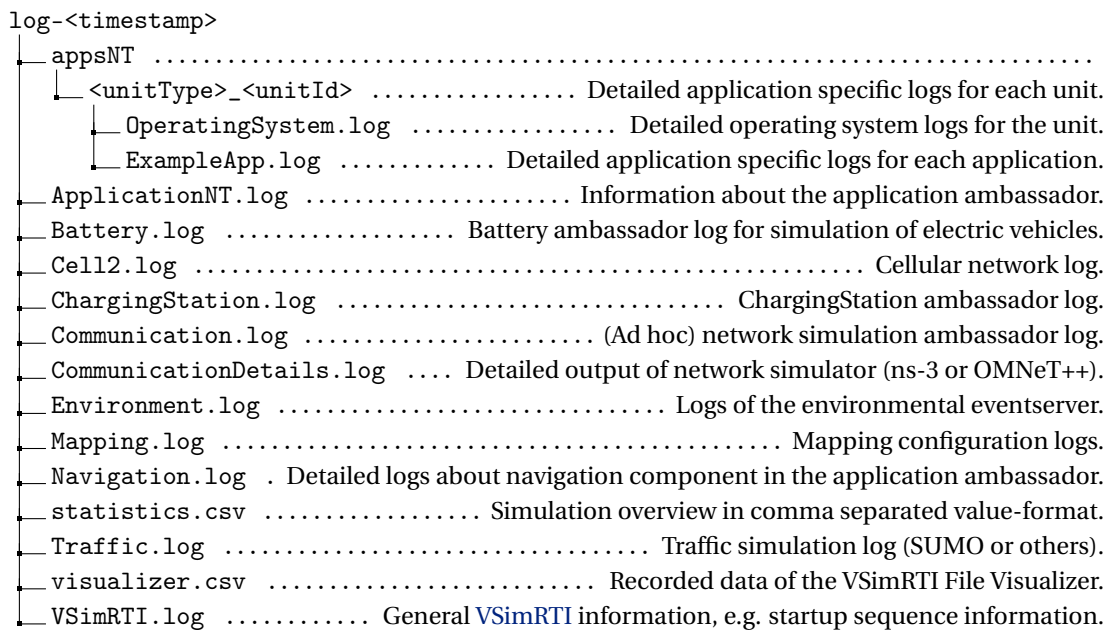


Figure 2.2: VSimRTI log folder structure

### 2.2.1 Logging

The main configuration file is `vsimrti/etc/logback.xml`. In this file, each output can be configured in great detail. This file can be adjusted to your needs, e.g. you can set up a more detailed logging for communication components but set a less verbose output for VSimRTI internal messages or traffic simulation depending on your simulation focus.

VSimRTI uses logback as logging framework and it is suggested to use logback for the other simulators as well. Logback offers a lot of parameters to adapt the output to your needs. Please refer to (<http://logback.qos.ch/manual/layouts.html#ClassicPatternLayout>) for a detailed overview of parameters you can use in the `logback.xml` file.

Please note that you can adjust the output to your needs by setting different log levels (ERROR, INFO, DEBUG etc.) for each component in the file under `vsimrti/etc/logback.xml`. This might also influence the simulation performance because of a possibly high amount of data to be logged.

## Federate specific logging

Depending on the simulation purpose, further configuration possibilities for federate specific logging may be of interest.

For instance, `OMNeT++` exhibits an elaborated logging concept. The `omnetpp.ini` in the scenario folder includes options to adjust the logging levels. The outputs of this federate are written into `CommunicationDetails.log`.

## 2.3 Run a simulation set via CLI

**i Notice:** The following options are only available with a commercial license of VSimRTI.

For further information on licenses, please refer to our [mailing list](#).

A common objective when running simulations is to assess the impact of different parameter settings for a specific scenario on the results of the simulation. To this end, the `simulation-runner` is a tool to apply different configurations to a scenario, after which a series of simulations is executed via **CLI** by calling the `simulation-runner start` script. Please refer to [10.1](#).

## 3 Simulators

**VSIMRTI** couples different simulators and can't be run alone. Therefore, it requires pre-installed simulators. In this chapter we give an overview of common simulators already supported as well as basic configuration help. For further information and configuration options, please see the javadoc documentation provided on the [VSIMRTI website](#) . To run other simulators than the provided ones, an additional component has to be written, which couples the simulator to **VSIMRTI** . If you have any questions or need further support, please feel free to contact our support team via our [mailing list](#) .

### 3.1 Kinds of simulators

The figure 3.1 gives an overview of the currently available simulators for **VSIMRTI**

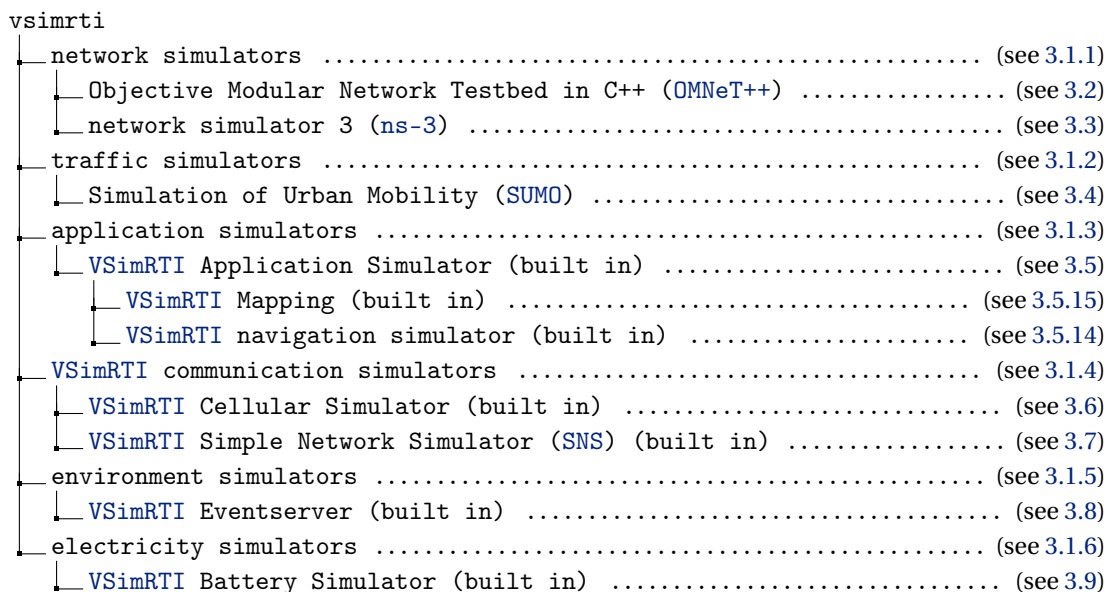


Figure 3.1: VSIMRTI simulator structure

#### 3.1.1 Network simulators

A network simulator is a piece of software modeling the behavior of a network by calculating the interactions between all participating entities. For **VSIMRTI** and, more specifically, the communication in a **V2X** environment this refers to simulations of the wireless transmissions taking place between the various simulated entities. The provided simulators are prepared to simulate a communication stack basing on the [IEEE 802.11p](#) network interfaces with IP and UDP on top. However, according to their model base,

alternative set ups are possible. Currently, VSimRTI supports the commonly known network simulators OMNeT++ and ns-3, and the built in communication simulator SNS.

### 3.1.2 Traffic simulators

A traffic simulator is a software modeling the movements of users in a traffic system. Users can mean cars, pedestrians, trains, ships, planes etc. Traffic simulators can be discriminated by various measures, of which one is the used scope:

**Microscopic models** Simulate each car individually and through the interaction of multiple cars also traffic flows. They are commonly used for situations such as a traffic crossing or an on-ramp situation.

**Macroscopic models** Models of a traffic flow without the modelling of individual cars. Instead the traffic flow is computed using models derived from fluid dynamics or gas-kinetics. By this the simulation is computationally far less expensive, so more cars and wider areas can be simulated. An example would be the prediction of traffic jams.

**Mesoscopic models** Try to bridge the gap between macroscopic and microscopic simulation using individual vehicles that are being actuated by macroscopic control variables.

**Sub-Microscopic models** Used to simulate a car in as much detail as possible. The prediction of the behavior is the most precise of all models, but also computationally the most expensive.

The currently supported traffic simulator SUMO is a microscopic traffic simulator.

### 3.1.3 Application simulators

An application simulator is an important component enabling the simulation and creation of V2X applications. It follows the European Telecommunications Standards Institute (ETSI) standards for Vehicle-to-Vehicle (V2V) communication. These standards contain message definitions like Decentralized Environmental Notification Messages (DENM) and Cooperative Awareness Messages (CAM) and also a general application container design.

### 3.1.4 VSimRTI communication simulators

For different reasons of convenience or extended analysis capabilities, VSimRTI ships with two additional communication simulators:

- The Cellular Simulator is a special case of network simulator to simulate the communication taking place within a cellular network, such as Universal Mobile Telecommunications System (UMTS) and Long Term Evolution (LTE).
- The Simple Network Simulator is written in Java and already tightly integrated in VSimRTI. It offers a lightweight and fast solution for simulation of ad hoc communication.

The provided VSimRTI communication simulators can be used as an alternative (SNS) or addition (Cell2) to classic network simulators.

### 3.1.5 Environment simulators

This kind of simulator simulates certain environmental events such as rain, fog, snow, etc..

### 3.1.6 Electricity simulators

In connection with [VSimRTI](#), electricity simulators enable investigations in the field of electric mobility. For this purpose, [VSimRTI](#) ships with the built in [VSimRTI](#) Battery Simulator.



## 3.2 OMNeT++

OMNeT++ itself is solely the simulation platform for discrete-event systems. Even though it is primarily targeted at simulating computer networks and distributed systems, it cannot be used without any extensions for wireless communication. For this kind of simulations, external model frameworks have to be included. Currently there are two prominent model frameworks which cover whole model suites for according focus of wireless research. These are the Mobility Framework and the OMNeT++ - Model library for wired, wireless and mobile networks (INET) Framework. As INET provides all models necessary for simulating Vehicle-2-X communication, it is selected for the integration to VSimRTI.

For more information on the INET extension you should look closer on the website <https://inet.omnetpp.org>.

Software information	
Developer(s)	OMNeT++ Community and OpenSim Ltd.
Written in	C++
Operating system	Windows (mingw), Linux
License	open source for academic use
Website	<a href="http://www.omnetpp.org/">http://www.omnetpp.org/</a> <a href="https://inet.omnetpp.org">https://inet.omnetpp.org</a>
Supported version(s)	OMNeT++ 4.6 INET 3.0
Dependencies	libprotobuf 3.3.0
Installation	via released installation script

Table 3.1: Software information: OMNeT++

### 3.2.1 omnetpp-ambassador folder structure

The omnetpp.ini file has to be located in the omnetpp folder of the scenario configuration.

```

<scenarioName>
├── omnetpp .....
│   ├── omnetpp_config.json ..... Ambassador configuration file
│   └── omnetpp.ini ..... OMNeT++ federate configuration file

```

Figure 3.2: omnetpp-ambassador folder structure

### 3.2.2 Installation

The `VSimRTI` all-in-one package comes with an installation script for the bash-shell, which automates the task of the `OMNeT++/INET` installation.

#### Prerequisites

The `OMNeT++/INET` federate furthermore depends upon:

- `libprotobuf 3.3.0`


Please make sure these libraries are installed before running the installation script.

#### Installation shell script



Figure 3.3: OMNeT++ folder structure

Please execute the following steps for installing the `OMNeT++/INET` federate:

1. Follow the link and download the source code of `OMNeT++ 4.6` (`omnetpp-4.6-src.tgz`):  
 <https://omnetpp.org/omnetpp/download/30-omnet-releases/2290-omnet-4-6-source-ide-tgz>

2. Make the installation script executable:

```
chmod 755 omnet_installer.sh
```

3. Make sure all dependencies are met by your system.
4. Run the script to install `OMNeT++/INET`:

```
./omnet_installer.sh -s /path/to/omnetpp-4.6-src.tgz
```

### 3.2.3 Installation in Docker environment

**Notice:** This is an experimental feature. Please refer to our [mailing list](#) if you experience any problems.

This guide gives instructions to execute the OMNeT++ federate inside a docker container. If you already installed OMNeT++ on your machine following the steps before, you can skip this section.

Docker<sup>1</sup> is a new approach to execute software. More precisely, it "wraps software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, and system libraries". As a result, the software is executed within a container and its execution does not rely on the environment the container is running in.

In context of VSimRTI, this approach allows to execute OMNeT++ within a docker container. The user does not need to manually install OMNeT++ and can even run OMNeT++ on Windows hosts.

1. Install Docker  $\geq$  1.13 on your machine.
2. To get everything work, please make sure to execute the following steps depending on your operating system:
  - Windows - In the settings, share the drive where VSimRTI is installed on. You may need to restart docker in the reset tab.
  - Linux - Make sure your user account belongs to the unix-group "docker". You may need to restart your machine.

3. Switch to the location of the Dockerfile in `vsimrti/bin/fed/omnetpp`

4. Follow the link and download the source code of OMNeT++ 4.6 (`omnetpp-4.6-src.tgz`) and place the downloaded file into the `vsimrti/bin/fed/omnetpp` directory:

 <https://omnetpp.org/omnetpp/download/30-omnet-releases/2290-omnet-4-6-source-ide-tgz>

5. Execute the following command on command line:

```
docker build -t omnetpp-federate:18.0 .
```

This could take a while to finish.

6. Enter the name of the docker image `etc/defaults.xml` in the `omnetpp`-section into the tag `dockerImage`:

---

```
<federate class="...">
  <id>omnetpp</id>
  ...
  <dockerImage>omnetpp-federate:18.0</dockerImage>
  ...
</federate>
```

---

You can test the installation of your docker image with the Tiergarten scenario, by activating `omnetpp` in the `vsimrti_config.xml`.

---

<sup>1</sup><https://www.docker.com/>

### 3.2.4 Configuration

The whole OMNeT++ specific configuration is done via the `omnetpp.ini` file. It covers static parts for the simulator coupling as the specific VSimRTI Event Scheduler and the ScenarioManager. Furthermore, logging configurations and the typical parameters for the communication layers (MAC, PHY and Radio Channel) are addressed. The communication parameters are different for vehicles and RSUs. Please refer to the OMNeT++ documentation on the [OMNeT++ homepage](#) for further information about the structure of the `omnetpp.ini` file.

## 3.3 ns-3

The **ns-3** is a discrete-event network simulator, that was started as an open-source project by a team led by **Tom Henderson** from the **University of Washington** in July 2006 and made its first release in June 2008. **ns-3** is targeted primarily towards research and educational use and thereby, also offers a vital community of developers and maintainers. It was developed as a replacement for the popular network simulator 2 (**ns-2**) and mainly focuses upon improving the core architecture, software integration, models, and educational components for real-world network devices and protocols (regardless, **ns-2** still remains in active use and will continued to be maintained in the near future). It simulates both unicast and multicast protocols and is used extensively in research on mobile ad-hoc networks


Like other network simulators, **ns-3** has a relatively steep learning curve, especially compared to GUI-based simulators. If you have no prior experience with **ns-3**, we recommend to familiarize yourself with the **ns-3** simulation environment and the **ns-3** simulation basics first. The **ns-3** documentation can be found under:

 <http://www.nsnam.org/documentation>

To take your first steps with **ns-3**, you might want to download<sup>2</sup> the latest version, build a copy of **ns-3** (it uses the Python-based build-system **waf**) and take a look at the examples, that are shipped within most of the **ns-3** source code repositories and packages. You might also examine the simulation output and try to adjust it.

Typically, a **ns-3** user will work under a Linux environment. For those running under Windows, there do exist environments which simulate the Linux environment to various degrees. The **ns-3** project has in the past (but not presently) supported the **Cygwin** environment for these users (see <http://www.cygwin.com> for details on downloading). **MiniGW** is presently not officially supported. According to the **ns-3** installation guide, the officially supported platforms include (please note, that plenty of other distributions are likely to work with **ns-3** regardless):

- Fedora
- Ubuntu
- OS X Mountain Lion, Snow Leopard, ...
- FreeBSD
- Cygwin

 **Important:** As stated above, **ns-3** is primarily developed on and for GNU/Linux platforms. Since Windows is such a widely used platform and Cygwin is not a perfect emulation of any Linux, we highly recommended for non-Linux users to consider the installation of a Linux virtual machine with a virtual machine environment, such as VMware<sup>3</sup> or VirtualBox<sup>4</sup>.

<sup>2</sup>Please note, that downloading **ns-3** via tarball is simpler than the Mercurial process since all of the pieces are pre-packaged for you.

<sup>3</sup>[http://www.nsnam.org/wiki/index.php/HOWTO\\_use\\_VMware\\_to\\_set\\_up\\_virtual\\_networks\\_\(Windows\)](http://www.nsnam.org/wiki/index.php/HOWTO_use_VMware_to_set_up_virtual_networks_(Windows))

<sup>4</sup>[http://www.nsnam.org/wiki/index.php/HOWTO\\_use\\_VirtualBox\\_to\\_run\\_simulations\\_on\\_Windows\\_machines](http://www.nsnam.org/wiki/index.php/HOWTO_use_VirtualBox_to_run_simulations_on_Windows_machines)

Software information	
Developer(s)	Tom Henderson, Mathieu Lacage, George Riley, Mitch Watrous, Gustavo Carneiro, Tommaso Pecorella and others
Written in	C++ (core) and Python (bindings)
Operating system	GNU/Linux FreeBSD Mac OS X
License	free software: <a href="#">GNU GPLv2</a>
Website	<a href="http://www.nsnam.org/">http://www.nsnam.org/</a>
Supported version(s)	3.25
Dependencies	libprotobuf 3.3.0 libxml2 libsqlite3
Deployed in VSimRTI all-in-one	no (and need a patch to link)

Table 3.2: Software information: ns-3

### 3.3.1 ns3-ambassador folder structure



Figure 3.4: ns3-ambassador folder structure

### 3.3.2 Installation

VSimRTI offers support for the current stable release of [ns-3](#) (3.25), that was released in March 2016. Older versions of [ns-3](#) (prior to 3.25) are not supported. However, also for newer versions we cannot guarantee the correct operation. The coupling to VSimRTI is designed in a manner of minimal code changes on the [ns-3](#) simulator itself to keep the update capabilities for future versions.

#### Prerequisites

For GNU/Linux platforms, the minimal requirements to run basic simulations are a **gcc** or **clang** compiler and a **Python** interpreter. At least you need the following packages to be installed:

Listing 3.1: ns-3: Minimum requirements

```

gcc
g++
python
python-dev

```

For a complete list of required packages for different distributions, please refer to the [ns-3](#) installation guide:


 <http://www.nsnam.org/wiki/index.php/Installation>

For the connection to [VSimRTI](#), the ns-3 federate furthermore depends upon

- libxml2
- libsqlite3
- libprotobuf 3.3.0

Please make sure these libraries are installed before running the installation script.

### Run the installation script

 **Important:** ns-3 requires several packages to be installed on your computer. You will need to ensure, that all required libraries are present on your system before proceeding. You may need superuser permissions to install packages.

To ease the installation of ns-3 for VSimRTI, the installation process has been delegated to an installation script, that can be found in the associated ns-3 federate folder.



Figure 3.5: ns3-ambassador federate folder structure

The ns-3 installation script accomplishes the following tasks:

1. Download ns-3 tarball from the official sources
2. Download the ns-3 federate for VSimRTI.
3. Apply a patch to ns-3 in order to make it work with VSimRTI.
4. Add the ns-3 federate to the waf build system.
5. Configure and build the patched ns-3 with the ns-3 federate.

In order to start the simulation, the following steps need to be performed:

1. Set up the `confWifi.xml`-file in the scenario folder (see section 3.3.4). An example `confWifi.xml`-file is shipped with the Tiergarten scenario.
2. For reasonable result logging, the logger-configuration in `vsimrti/etc/logback.xml` has to be adapted to support the ns-3 ambassador and federate.
3. At last ns-3 has to be activated in the `vsimrti_config.xml` and the simulation can be started.

### 3.3.3 Installation in Docker environment

**1 Notice:** This is an experimental feature. Please refer to our [mailing list](#) if you experience any problems.

This guide gives instructions to execute the `ns-3` federate inside a docker container. If you already installed `ns-3` on your machine following the steps before, you can skip this section.

Docker<sup>5</sup> is a new approach to execute software. More precisely, it "wraps software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, and system libraries". As a result, the software is executed within a container and its execution does not rely on the environment the container is running in.

In context of `VSimRTI`, this approach allows to execute `ns-3` within a docker container. The user does not need to manually install `ns-3` and can even run `ns-3` on Windows hosts.

1. Install Docker  $\geq$  1.13 on your machine.
2. To get everything work, please make sure to execute the following steps depending on your operating system:
  - Windows - In the settings, share the drive where `VSimRTI` is installed on. You may need to restart docker in the reset tab.
  - Linux - Make sure your user account belongs to the unix-group "docker". You may need to restart your machine.
3. Switch to the location of the Dockerfile in `vsimrti/bin/fed/ns3`
4. Execute the following command on command line:  
`docker build -t ns3-federate:18.0 .`  
 This could take a while to finish.
5. Enter the name of the docker image `etc/defaults.xml` in the `ns3`-section into the tag `dockerImage`:

---

```
<federate class="...">
  <id>ns3</id>
  ...
  <dockerImage>ns3-federate:18.0</dockerImage>
  ...
</federate>
```

---

You can test the installation of your docker image with the Tiergarten scenario, by activating `ns3` in the `vsimrti_config.xml`.

---

<sup>5</sup><https://www.docker.com/>



### 3.3.4 Configuration

The whole ns-3 specific configuration is done via the `confWifi.xml` and `configTechnologies.xml` files.

Listing 3.2: `confWifi.xml`

---

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wifi>

  <!-- IPv4 address generator -->
  <ipConfiguration>
    <ip address="192.168.0.0" mask="255.255.0.0"/>
  </ipConfiguration>
  <!-- Calculate a propagation delay -->
  <propagationDelayModel>
    <delay model="ns3::NonePropagationDelayModel"/>
  </propagationDelayModel>
  <!-- Modelize the propagation loss through a transmission medium -->
  <propagationLossModel>
    <loss model="ns3::FriisPropagationLossModel"/>
  </propagationLossModel>
  <wifiConfiguration>
    <!-- Create non QoS-enabled MAC layers -->
    <wifiMac property="type" value="ns3::AdhocWifiMac"/>
    <!-- Wifi PHY mode -->
    <wifiManager property="phyMode" value="OfdmRate54Mbps"/>
    <!-- Wifi manager -->
    <wifiManager property="type" value="ns3::ConstantRateWifiManager"/>
    <!-- The energy of a received signal should be higher than this threshold (dbm) to allow ↵
         ↵ the PHY layer to detect the signal -->
    <wifiPhy property="EnergyDetectionThreshold" value="-81.0"/>
    <!-- The energy of a received signal should be higher than this threshold (dbm) to allow ↵
         ↵ the PHY layer to declare CCA BUSY state -->
    <wifiPhy property="CcaModelThreshold" value="-99.0"/>
    <!-- Transmission gain (dB) -->
    <wifiPhy property="TxGain" value="0.0"/>
    <!-- Reception gain (dB) -->
    <wifiPhy property="RxGain" value="0.0"/>
    <!-- Number of transmission power levels available between TxPowerStart and TxPowerEnd ↵
         ↵ included -->
    <wifiPhy property="TxPowerLevels" value="1"/>
    <!-- Maximum available transmission level (dbm) -->
    <wifiPhy property="TxPowerEnd" value="17.0"/>
    <!-- Minimum available transmission level (dbm) -->
    <wifiPhy property="TxPowerStart" value="17.0"/>
    <!-- Loss (dB) in the Signal-to-Noise-Ratio due to non-idealities in the receiver -->
    <wifiPhy property="RxNoiseFigure" value="0.0"/>
    <!-- Channel center frequency = Channel starting frequency + 5 MHz * (nch - 1) -->
    <wifiPhy property="ChannelNumber" value="1"/>
  </wifiConfiguration>
</wifi>
```

---

The IP configuration information includes the network address and network mask. The ns-3 propagation module defines two generic interfaces, namely **PropagationLossModel** and **PropagationDelayModel**, for the modelling of propagation loss respectively propagation delay.

In the default `confWifi.xml`, the Wi-Fi device uses the ns-3 standard propagation delay model `ns3::ConstantSpeedPropagationDelayModel` and the ns-3 standard propagation loss model `ns3::FriisPropagationLossModel`. Radio propagation models in ns-3 can easily be exchanged with

the `ns-3` class registration system (see the `ns-3` documentation for details). The Wi-Fi configuration includes additional parameters, like sending power and antenna gain.

Listing 3.3: `configTechnologies.xml`

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ns3Configuration>
  <installers>
    <installer type="ns3::WifiVehicleInstaller" name="WifiVehicle" file="confWifi.xml" ↵
      ↵ default="true" />
    <installer type="ns3::MobilityModelInstaller" name="Mobility" default="true" />
  </installers>
</ns3Configuration>
```

The configuration manager of the `ns-3` federate defines, which installer should be loaded for the Wi-Fi device (referring to the `confWifi.xml`) and the mobility model. Usually, you don't need to take any changes and simply use the default configuration file, that ships with the `ns-3` federate.

## 3.4 SUMO

SUMO is an highly portable, microscopic and continuous road traffic simulation package. It is designed to handle large road networks faster than real-time. Each vehicle has an own route and is simulated individually.

Software information	
Developer(s)	German Aerospace Center
Written in	C++
Operating system	GNU/Linux and Microsoft Windows
License	free software: <a href="#">GNU GPLv2</a> since 0.31.0: <a href="#">EPL-2.0</a>
Website	<a href="http://sumo.dlr.de/">http://sumo.dlr.de/</a>
Supported version(s)	0.30.0 ... 0.32.0
Deployed in VSimRTI all-in-one	no

Table 3.3: Software information: SUMO

**⚠ Important:** Please note that VSimRTI only supports SUMO 0.30.0 or higher.

### 3.4.1 sumo-ambassador folder structure

This ambassador can be configured with a configuration file. The specific path is `vsimrti/scenarios/<scenarioName>/sumo/sumo_config.json`

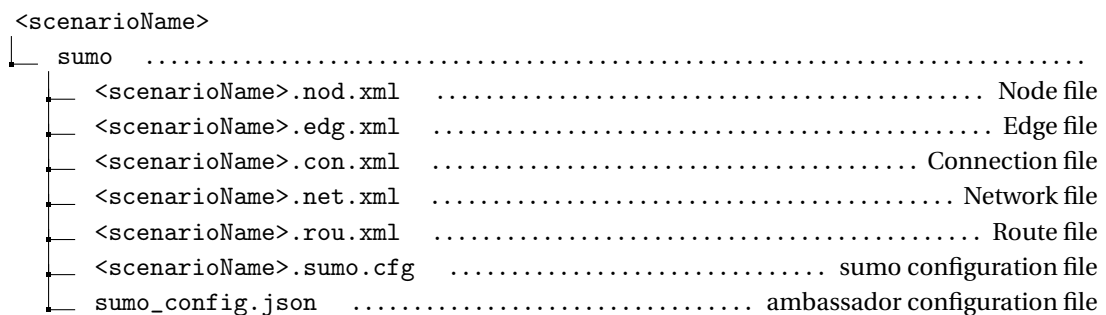


Figure 3.6: sumo-ambassadorfolder structure

### 3.4.2 Installation

For Microsoft Windows operating systems download and extract `sumo_winbin.zip` from the [SUMO](#) website. For GNU/Linux operating systems download and extract the provided GNU/Linux binaries or build sumo by yourself. Please refer to the [SUMO](#) Wiki for further information.

In order to run **SUMO** with **VSimRTI** you need to make the **SUMO** binaries available system wide by adding the **SUMO** binary folder to your `PATH` environment variable (see section 13.1).

### 3.4.3 Configuration

**i Notice:** The documentation for the **VSimRTI** specific component is freely available on the [DCAITI website](#), explaining all available options.

The **SUMO** configuration consists of sumo specific config files and the sumo-ambassador configuration file. `vsimrti/scenarios/<scenarioName>/sumo`. Your main configuration file name must end with the suffix `*.cfg`. The **SUMO** Wiki covers more information and also a tutorial about the different configuration files. Route-, network- and **SUMO** configuration files have been generated (using `scenario-convert` or by yourself). The last step is the creation of a sumo-ambassador configuration file.

In contrast to standard traffic simulations using **SUMO** only, **VSimRTI** handles the **SUMO** files in a slight different way, i.e. the route file for a simulation is created at runtime to enable dynamic routing. To get correct simulation results, we recommend using the `scenario-convert` tool to create the **SUMO** files out of OpenStreetMap data (see section 11.1).

### 3.4.4 Using the SUMO GUI with VSimRTI

It is also possible to use the Graphical User Interface (**GUI**) of **SUMO** in order to visualize and interact with the simulation. To achieve this, **VSimRTI** can be configured that it starts the **GUI** process of **SUMO** as the federate rather than the command line interface.

In order to use the **SUMO GUI** with **VSimRTI**, please open the file `vsimrti/etc/defaults.xml` and replace the entry `com.dcaiti.vsimrti.fed.sumo.ambassador.SumoAmbassador` with `com.dcaiti.vsimrti.fed.sumo.ambassador.SumoGUIAmbassador`.

Afterwards **VSimRTI** can be executed as usual. However, the traffic simulation has to be triggered manually within the **GUI** window of **SUMO**. Additionally, the field `Delay` within the **GUI** can be used to slow down the simulation. For further information about the graphical interface please consider the **SUMO** documentation.

**i Notice:** Keep in mind to launch **VSimRTI** with the argument `-w 0` in order to disable the watchdog timer. Otherwise it would shutdown **VSimRTI** if you pause the simulation in the **SUMO GUI**.

### 3.4.5 Routes and Vehicle Types

For the simulation the traffic simulator **SUMO** is started as a background process. In preparation, **VSimRTI** automatically generates all files needed, such as routes and vehicle types definition. In particular, **VSimRTI** generates a `*.rou.xml` which contains all vehicle types from the mapping3 configuration and all routes from the scenario database. This also means, that any route referenced by the mapping3 configuration must be available in the scenario database. Additionally, the `*.rou.xml` in the scenario configuration of **SUMO** will always be overwritten by **VSimRTI** at the start of the simulation.

If you want to use **SUMO** specific parameters for the vehicle types (e.g. to define the car following model or the emission class), you can predefine them in the file `<scenarioName>.rou.xml` of your scenario. All parameters you define their will be used when the route file is generated for the simulation (except all parameters which are defined in the mapping3 configuration such as `accel`, `decel`, `tau`, etc.). The Tiergarten scenario provides such example file.

### 3.4.6 Further information

**i Notice:** We recommend to use the 64 bit version of **SUMO** if you want to simulate scenarios with a big traffic network.

### 3.4.7 VSimRTI specific information

**i Notice:** The `SumoAmbassador` broadcasts `VehicleMovements` in fixed time steps and also sends a `TrafficLightUpdate` message, when the `tl`-state change was initiated in the previous step. For dynamic change of vehicle behavior during simulation, **SUMO** subscribes to the messages `VehicleTypesInitMessage`, `VehiclePathsInitMessage`, `ChangeSpeed`, `SlowDown`, `PropagateNewRoute`, `ChangeStaticRoute` and `ChangeTrafficLightsState`.

**⚠ Important:** Even though the `SumoAmbassador` is able to receive `ChangeRoute` messages which will result in using the internal Re-route feature of **SUMO**, it is strongly recommended to use the navigation functionality of the `ApplicationNTAmbassador` by using the provided API.

## 3.5 VSimRTI Application Simulator

The application simulator is a sophisticated simulator to provide simulation units (e.g. vehicles, road side units, traffic lights, charging stations) an interface to run their logic. The latest application simulator has been developed from scratch and is called `ApplicationSimulatorNT`.

### 3.5.1 applicationNT-ambassador folder structure

This ambassador can be configured with a configuration file. The specific path is `vsimrti/scenarios/<scenarioName>/applicationNT/applicationNT_config.json`

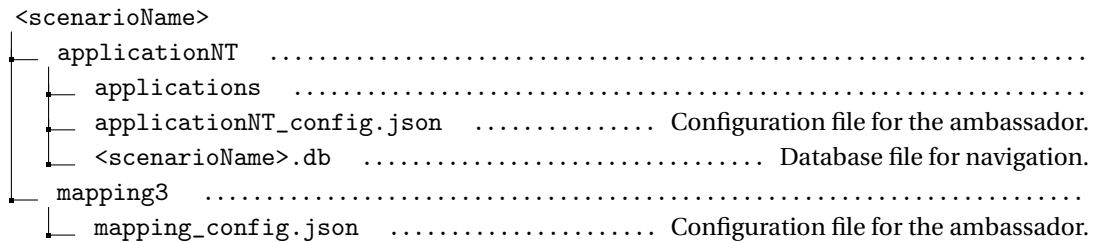


Figure 3.7: applicationNT-ambassador folder structure

### 3.5.2 Installation

This simulator does not need to be installed. It is delivered as part of the VSimRTI-installation package.

### 3.5.3 Libraries

To develop applications you need to reference to jar files you will find in the following directories:

- `vsimrti/bin/ambassadors/applicationNT-ambassador-18.0.jar`
- `vsimrti/bin/ambassadors/lib/*.jar`
- `vsimrti/bin/lib/*.jar`
- `vsimrti/bin/core/vsimrti-18.0.jar`

### 3.5.4 Basic functionality

Each logical unit is recognized as a separate simulation unit. The application simulator can load different applications for each simulation unit. For each application there is a general class and an extension for the particular type of the simulation unit (e.g. `Vehicle` extends `SimulationUnit`).

This application may request information from an object and wants to control this. For each simulation object, an operating system will be created. For each application, there is an operating system and an extension of the operating system for the particular type (e.g. `VehicleApplication` extends `AbstractApplication<VehicleOperatingSystem>`).

### 3.5.5 Gather information in an application

To gather information in an application from a simulation unit, there exist typically two different ways.

#### React on changes

Applications will be notified on changes. All applications must implement methods that are called before and after a change. Typically, these methods are named `before*` and `after*`. Please note that not all changes of information notify applications through this mechanism. Some information must be evaluated by sampling. As an example, the method `getStateOfEnvironmentSensor` could not notify the application by changing its state.

#### Sample information

The sampling of information at specific times can be done by requesting future events.

#### Self events

An event is a notification with information at a future point in the simulation. With a self-event, applications can wake up themselves at certain time steps. An event can be requested at any simulation point to trigger an event execution.


### 3.5.6 Equipped units

All units are generally classified into two types, units that have application logic or not. Equipped vehicles can also use a predefined ETSI application.

### 3.5.7 Technical details

- The simulator does not use concurrent processing.
- The logs are based on the application name.

### 3.5.8 Configuration

 **Notice:** The documentation for the VSimRTI specific component is freely available on the [DCAITI website](#), explaining all available options.

Most of the actual configuration of the applications is done by the mapping-simulator. The mapping-simulator is responsible for spawning the entities of the simulation and therefore also decide which application to map onto which entity.

To deploy an application you have to put it into the appropriate subdirectory for the simulated unit.

### 3.5.9 Development of applications

#### Overview

Developing custom applications in [VSimRTI](#) is rather easy. The best way to learn is by looking at the source code of actual applications. For this purpose, we provide the source code of all tutorial applications and further examples. You can find the source code on the [DCAITI website](#) .

For an easy understanding of the application API, the following questions and their answers should help:

- **What is required to get my own application run in VSimRTI?**

In [VSimRTI](#) it is very easy to build your own application. First, it needs to inherit from the `AbstractApplication` class (see following section). Secondly, the application must be mapped to a vehicle (or RSU, or traffic light, ...) via the mapping configuration (see section [3.5.15](#)). Finally, the application must be compiled as a Jar-File and placed into the application directory of your scenario.

- **How can I access vehicle functions from within the application, such as sending V2X messages?**

Every applications has access to the `OperatingSystem` of the underlying unit which allows to change its state or to initiate actions, such as sending messages to other vehicles.

- **How can I react on events during the simulation, such as receiving V2X messages?**

For each application you decide, which events the application should listening to. For example, if your application needs to react upon incoming V2X messages, it simply implements the `CommunicationApplication` interface. In the following section you can find all available interfaces applications can implement.

#### Create a 'Hello world' application using eclipse

Create a new project in eclipse (File → New → Java Project). Name your application `HelloWorldApp` . Confirm the creation of the project. Right click on the project and choose "Properties" to open the project specific configuration options. In the dialog window select "Java Build Path" and click on the Libraries tab. Choose "Add external JARs" and navigate to your `vsimrti` installation folder. Choose the following JARs and add them to your project:

- `vsimrti/bin/ambassadors/applicationNT-ambassador-18.0.jar`
- `vsimrti/bin/ambassadors/lib/*.jar`
- `vsimrti/bin/lib/*.jar`
- `vsimrti/bin/core/vsimrti-18.0.jar`

Right click on the "Hello world" Project and choose (New → Package). Name the package something like `com.mycompany.vsimrti.applications` . Right click on the package and chose (New → Class). Name the class `HelloWorldApp` , extend from the Class `[...]applicationNT.ambassador.simulationUnit.applications.AbstractApplication<OS>` and confirm. Afterwards, you need to specify the type of operating system `<OS>` your application is



running on by parameterizing the extended `AbstractApplication<OS>` by one of the following classes<sup>6</sup>:

- `VehicleOperatingSystem` - for applications mapped to normal vehicles.
- `ElectricVehicleOperatingSystem` - for applications for vehicles with electro mobility features.
- `RoadSideUnitOperatingSystem` - for applications mapped to Road Side Units (RSU)s.
- `TrafficLightOperatingSystem` - for applications mapped to traffic lights.
- `ChargingStationOperatingSystem` - for applications mapped to charging stations.

Furthermore, your application can implement the following interfaces<sup>7</sup> in order to get informed on specific events:

- `VehicleApplication` - get informed when information about the vehicles state is updated.
- `ElectricVehicleApplication` - get informed on electric vehicle specific events.
- `CommunicationApplication` - react on incoming V2X messages.
- `VSimRTIApplication` - get informed on VSimRTI internal events.
- `TrafficLightApplication` - get noticed when the traffic light program is changed.
- `ChargingStationApplication` - react on state changes of the charging station.

You can now continue and make modifications. To deploy the application you have to export the project (File → Export → Java/Jar File) and put the JAR in the folder `vsimrti/scenarios/<scenarioName>/applicationNT/applications`.

**Notice:** You can download the example Application `HelloWorldApp`, which is part of the `AdditionalExamples` package, from the website.

In the following picture you can find the example application `HelloWorldApp` opened in eclipse.

<sup>6</sup>See package `com.dcaiti.vsimrti.fed.applicationNT.ambassador.simulationUnit.operatingSystem.*`

<sup>7</sup>See package `com.dcaiti.vsimrti.fed.applicationNT.ambassador.simulationUnit.applicationInterfaces.*`

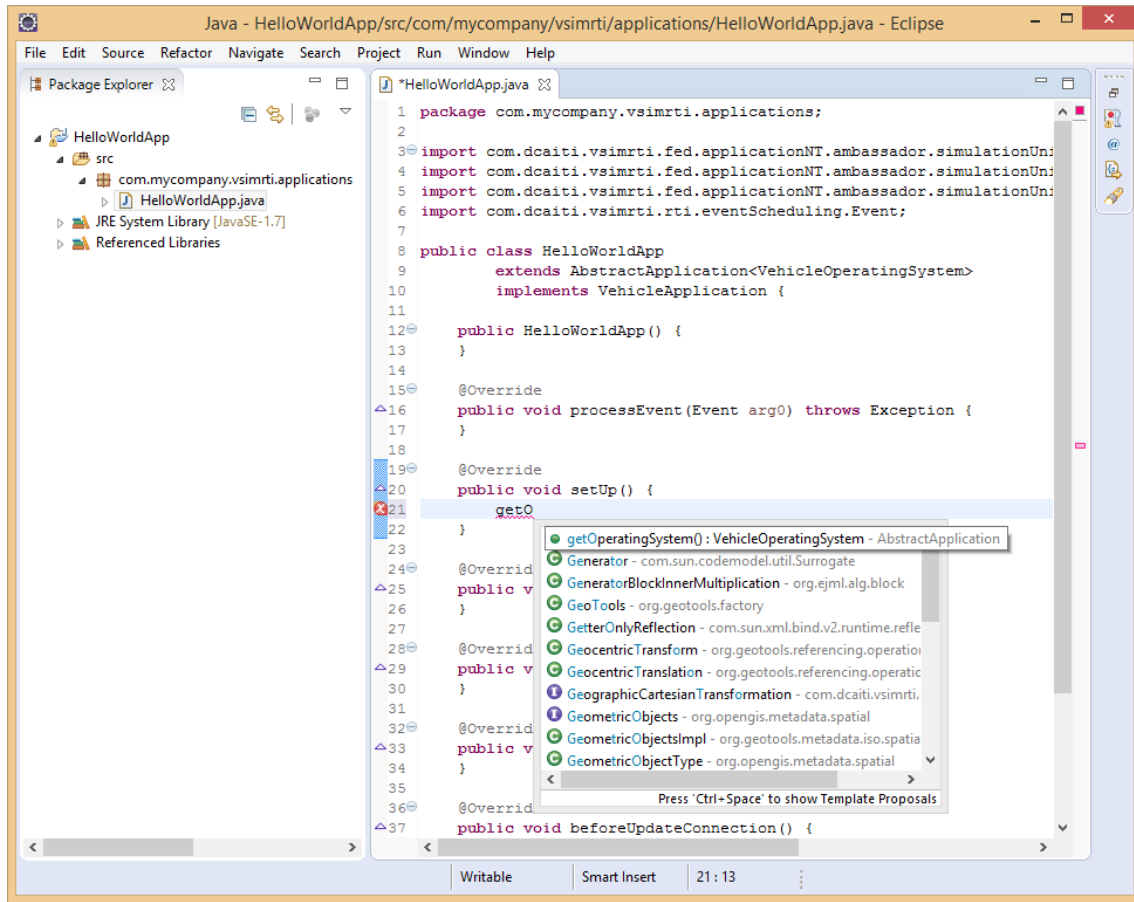


Figure 3.8: HelloWorldApp

### 3.5.10 Debugging of applications

To debug an application, remote debugging needs to be used. The following steps need to be performed in order to debug the application:

1. Open the application in your IDE
2. Modify your `vsimrti.sh` or `vsimrti.bat`, depending on if you are on windows or a Unix-like system. Examples can be found below.
3. Add a new debug profile for remote debugging, make sure port 4100 (or whichever was provided in the call to VSimRTI)
4. Launch VSimRTI with the argument `-w 0` to disable the watchdog timer will interfere with debugging

Listing 3.4: "vsimrty.sh on Unix-like systems"

---

```
#!/bin/bash
set -e

source common.sh
javaMemorySizeXms=""
javaMemorySizeXmx=""
javaProxySettings="-Djava.net.useSystemProxies=true"
javaDebugSettings="-agentlib:jdwp=transport=dt_socket,server=y,suspend=y,address=4100"

cmd="java ${javaProxySettings} ${javaMemorySizeXms} ${javaDebugSettings} ${javaMemorySizeXmx} ↵
↳ -cp .:${core}:${libs}:${ambassadors}:${ambassador_libs}:./etc -Djava.library.path=./lib ↵
↳ com.dcaiti.vsimrty.start.XMLConfigRunner $*"
$cmd
```

---

Listing 3.5: "vsimrty.bat on windows systems"

---

```
@ECHO OFF
SetLocal EnableDelayedExpansion

call common.bat

set javaMemorySizeXms=
set javaMemorySizeXmx=

set javaProxySettings=-Djava.net.useSystemProxies=true

set javaDebugSettings=-Xdebug -Xrunjdwp:transport=dt_socket\,server=y\,suspend=y\,address=4100

java %javaProxySettings% %javaDebugSettings% %javaMemorySizeXms% %javaMemorySizeXmx% -cp !libs ↵
↳ ! -Djava.library.path=./lib com.dcaiti.vsimrty.start.XMLConfigRunner %*

EndLocal

exit /b %errorlevel%
```

---

### 3.5.11 Internal ETSI applications

We provide [ETSI](#) conform applications which implement specific CAM generation rules. You can find them in the package

- `com.dcaiti.vsimrty.fed.applicationNT.etsiApplications.impl` .

#### ETSI Application (Vehicle)

For example, to provide ETSI functionality for your vehicle, you can load the ETSI application `com.dcaiti.vsimrty.fed.applicationNT.etsiApplications.impl.Vehicle` .

This application generates ETSI data for its simulation unit (e.g. heading, position, speed and time for vehicles). According to its configuration, the application then sends out CAMs to other vehicles in range. Note that the messages are only send when the time lies between the configured minimum and maximum interval.

**Default configuration** Without explicit configuration for the [ETSI](#) a default configuration is loaded. For the [ETSI](#) interceptor have a look at the class `CEtsi`.

**Notice:** For more information about this component, please have a look at the javadoc (doxygen) documentation.

Currently, the default configuration for the ETSI application looks like this:

```
{
  "maxStartOffset": 1000000000,
  "minInterval": 100000000,
  "maxInterval": 1000000000,
  "positionChange": 4,
  "headingChange": 4,
  "velocityChange": 0.5
}
```

In general, the numeric values should be interpreted as time delta. E.g., if the [ETSI](#) values change by the given amount, a new CAM is sent. The only exceptions are the minimum and maximum intervals. If the time lies over the maximum interval, a CAM is sent even if the difference between the [ETSI](#) values isn't high enough. If the difference is high enough but the minimum interval isn't surpassed, no message will be sent.

### 3.5.12 Sending Cooperative Awareness Messages (CAM)

The following section will show how CAMs are implemented in VSimRTI and how you can alter them. CAMs consist of four parts:

- Header with generic information
- `MessageBody`
- `ServiceList`
- `TaggedValue` list

First of all generic information like protocol version, creation time stamp are transmitted. Normally this data set follows a network beacon, already containing data like position and speed. Nevertheless this functionality must be implemented in the network layer, that means in the network simulator. At the moment this is not supported and is therefore compensated in the next message part, the message body.

The body can contain either RSU or Vehicle awareness data. In the first case, only the RSU type is submitted, but this probably changes with a new CAM specification. In the second case, we provide data like vehicle width, length, position, speed, type and heading. However, the specification is not completely implemented, especially acceleration data, as well as light and brake status are missing. The third part of the CAM specification, the service list, is also not implemented. This will probably change, when a list of services is defined by the ETSI.

Last but not least a message can contain optional data. This is fully implemented and is used for our traffic light CAM messages, which provide the traffic light status.

The CAM sending interval can be configured, more information can be found in the configuration section of the application simulator.

### 3.5.13 Access SUMO TraCI from applications

If SUMO is used as a traffic simulator and a special functionality is required, the `sendSumoTraciByteArrayMessage` function in the `OperatingSystem` can be used.

The function expects a string (a unique identifier which will be assigned to the response) and a byte array (representing the complete Traffic Control Interface (TraCI) request including the header). The message identifier can be an empty string.

In all cases the command will trigger a response. The application can receive the response from the method `onSumoTraciByteArrayMessageResponse`. This method will receive a `SumoTraciByteArrayMessageResponse` object. This response contains the specified identifier. The application must handle the identification process of the response itself.

**i Notice:** A response is delivered to every application. To prevent unintentional delivery each application request should use an immutable universally unique identifier (UUID).

**⚠ Important:** Be careful when using this interface and the TraCI commands. The commands are delivered to TraCI without any prior checks.

**i Notice:** You can find the example application `TestSumoTraciByteArrayMessageApp` in the additional examples bundle on the [DCAITI website](#).

### 3.5.14 Navigation

The navigation of vehicles is handled completely by the application simulator. Each vehicle is equipped with a navigation system which provides all required information and functions for navigational purposes:

- Retrieve the current position and heading of the vehicle.
- Get the target of the vehicle.
- Calculate various routes from the current position to an arbitrary target.
- Choose a suitable route out of existing ones from the current position to an arbitrary target.
- Switch onto a specific route.

In order to provide routing functionality, a map model based on Open Street Map data is used, which needs to be transformed before the simulation using `scenario-convert`. The map data including initial routes for vehicles is provided with the database file which needs to be located in `vsimrti/scenarios/<scenarioName>/applicationNT/<scenarioName>.db`

## Configuration

If the database needs to be located somewhere else, the path can be specified in

`vsimrti/scenarios/<scenarioName>/applicationNT/applicationNT_config.json`:

---

```
{
  ...
  "navigationConfiguration": {
    "databaseFile": "path/to/scenario.db"
  }
}
```

---

## Usage

The following snippet show, how the navigation system can be used within an application:

---

```
//get navigation module
INavigationModule navigationModule = getOperatingSystem().getNavigationModule();

//choose target position to current target
RoutingPosition targetPosition = new RoutingPosition(navigationModule.getTargetPosition());

//set routing parameters to fastest route search
RoutingParameters params = new RoutingParameters().costFunction(IRoutingCostFunction.Fastest);

//calculate routes
RoutingResponse response = navigationModule.calculateRoutes(targetPosition, params);

//switch to best route
if(response.getBestRoute() != null){
    boolean routeSwitched = navigationModule.switchRoute(response.getBestRoute());
    ...
}
```

---

### 3.5.15 Mapping 3

The mapping configuration is used to define how many vehicles of a specific type should drive in a simulation. You can choose between a deterministic mapping (producing the exact same sequence of mapped vehicles in every simulation run with regard to the given ratios at each point in time the simulation) and a stochastic mapping (resulting in a random order of mapped vehicles).

The new mapping ambassador was built to simplify the configuration of complex scenarios. It is not as tightly linked to the database as its predecessor and relies on [JSON](#) rather than Extensible Markup Language ([XML](#)) for configuration.

The configuration for the Mapping3-Federate is located in `vsimrti/scenarios/<scenarioName>/mapping3/mapping_config.json`.

In this file an object is described in JavaScript Object Notation ([JSON](#)), which will at runtime be parsed using the MappingFramework-Class.

**Notice:** The documentation for the VSimRTI specific component is freely available on the [DCAITI website](#), explaining all available options.

## The Mapping-Framework-Class

The main class for the Mapping3-Configuration is the MappingFramework-Class. It is divided into six parts: `prototypes`, `rsus`, `tls`, `vehicles`, `matrixMappers` and `config`.

In `prototypes` you can define models for other objects, which can later be reused in the other sections. This makes it possible to reuse the definition of certain vehicles or the combination of multiple applications, reducing redundancies.

The `rsus`-section offers the possibility to define instances of `RSU`'s. To reuse a prototype simply specify its name in the `name`-property. All relevant properties will then be automatically filled in. At least, you must define a position for every `RSU`.

In the `tls`-section traffic lights can be defined. The traffic lights themselves are already specified in the database of the scenario. Here applications can be mapped to them. Two different modes are offered for that: You can map the entities to individual traffic lights by specifying the name of the traffic light in the `tlName`-property. Alternatively you can define weights for the specified traffic lights, which will then be the base for a random distribution through all traffic lights. The weights do not have to add up to 100 or 1. Consequently all traffic lights will be mapped using the specified prototypes as soon as one weight differs from zero. So in case you don't want all traffic lights to have applications running on them you have to define one traffic light without any applications and add a weight to it.

The `vehicles`-section is the centerpiece of the configuration. Here the departures, routes and types of the vehicles are defined. It is divided into `VehicleSpawner`. These spawners are designed to produce a traffic stream with the given traffic density (in vehicles/hour). By specifying a maximum number of vehicles to be created you can also spawn individual vehicles. A `VehicleSpawner` offers the possibility to combine multiple vehicle types. By specifying weights for the different vehicle types the ratios between them can be defined. If no weights are defined they are assumed to be equal. Note: If at least one vehicle type has a weight defined this does not apply! In that case all types without a defined weight are ignored. Instead of specifying a route you can also choose two points (given in long/lat). If no radius for the point is specified the closest point to the given position will be chosen. Otherwise a point inside the radius will be selected randomly.

It is also possible to define and use OD-Matrices. To do so add a `ODMatrixMapper` to the `matrixMappers`-list. Each `MatrixMapper` consists of a list of points, the vehicle-types to be used and the actual flow-values between each of the points. It is possible to define multiple matrices. This way distinctively different compositions of the vehicle flows can be achieved. The `MatrixMapper` will be called before the actual execution of the simulation and will generate vehicle-spawners for the flow between each of the points. Thus the `MatrixMapper` is just a way to save work. If you want to use the extra-option that multiple `vehicleSpawners` offer you should use them.

In the `config` extra options are defined.

## Mapping and Identifier

Every traffic object in VSimRTI has a globally unique string identifier. These identifiers are used to identify a traffic object in VSimRTI as well as in different ambassadors. From user's aspect, these identifiers will be seen in the log files which are generated after a simulation. Therefore, it should be explained, which identifier belongs to which traffic object.

Traffic Object	VSimRTI Internal ID
Vehicle	veh_[seqNr]
RSU	rsu_[rsuName]
Traffic Light	tl_[groupId]

Table 3.4: Mapping aspects

- *seqNr* is the sequence number of simulated vehicles, which starts from zero.
- *rsuName* is the element name defined by the user for the attribute **RsuPosition**.
- *groupId* is the group id of the traffic light.

**Notice:** For more information about this component, please have a look at the javadoc (doxygen) documentation.



## 3.6 VSimRTI Cellular Simulator

The VSimRTI built-in Cellular Simulator enables the applications to use cellular network communication. The simulation of cellular communication in VSimRTI consists of two parts:

1. The Cellular Simulator itself and
2. The applications that can communicate over cellular networks in the Application Simulator

These changes are done in a generic way, making the cellular simulator exchangeable. Users interested in a different kind of simulation of cellular communication may use other simulators and develop ambassadors connecting them to VSimRTI.

The Cellular Simulator in the current state consists of three main modules:

1. Uplink Module
2. Geocaster Module
3. Downlink Module

The Geocaster module simulates a mandatory component for ITS communication. It is inspired by the several architectures from research projects as simTD or CONVERGE to enable ITS use cases over cellular networks. It mainly takes over the task of an addressing and routing component with geographic knowledge to support geo-addressing. However, it also supports regular topological addressing.

The Uplink and Downlink Module are responsible for the transmission simulation. They account for the aspects of transmission delays, packet losses and available data rates. In this context, Uplink and Downlink always refer to the direction towards respectively from the Geocaster. For instance, a transmission from an Internet-based server towards a vehicle would include an Uplink between the server and the Geocaster and a Downlink between the Geocaster and the vehicle. While the Uplink direction only allows point-to-point communication (Unicast), the Downlink supports point-to-point (Unicast) as well as point-to-multipoint (Multicast).

### 3.6.1 cell2-ambassador folder structure

The VSimRTI cellular simulator can be configured via three distinct configuration files, which can be found within the scenarios folder structure:



Figure 3.9: cell2-ambassador folder structure

The network and regions configuration files are referenced in the cellular ambassador configuration file.

### 3.6.2 Installation

This simulator does not need to be installed. It is delivered as part of the VSimRTI-installation package.

### 3.6.3 Configuration

We provide a cellular configuration file in the example scenarios of Tiergarten and Barnim. Please note that in the default configuration of this scenario the Cellular Simulator is deactivated. To activate the cellular simulator just add

---

```
<!-- Cell federate -->
<federate id="cell12" active="true" />
```

---

to the `vsimrti_config.xml` (or, if this line already exist, make sure 'active' is set to 'true'). With this done you now can deactivate all network simulators if only cellular simulation is intended in the investigated scenario. However, it is also possible to use multiple communication technologies (of ad hoc and cellular) in a hybrid scenario.

The central configuration for the cellular simulator in the file

`vsimrti/scenarios/<scenarioName>/cell12/cell12_config.json` could look like this:

---

```
{
  "debugGeocasting": false,
  "visualizeRegions": true,
  "networkConfigurationFile": "network.json",
  "regionConfigurationFile": "regions.json"
}
```

---

The `visualizeRegions`-option from the `cell12_config.json` is used to write a KML-file that visualizes the used cell regions. Google Earth can be used to display it.

The configuration for the global network in the cellular simulator in the file

`vsimrti/scenarios/<scenarioName>/cell12/network.json` could look like this:

---

```
{
  # global region has no area definition (is the default region)
  "globalRegion": {
    "uplink": {
      "delay": {
        "type": "ConstantDelay",
        "delay": 100,
        "prpl": 0,
        "retries": 2
      }
    },
    "capacity": 28000000
  },
  "downlink": {
    "unicast": {
      "delay": {
        "type": "ConstantDelay",
        "delay": 50,
        "prpl": 0,
        "retries": 2
      }
    },
    "multicast": {
      "delay": {
        "type": "ConstantDelay",
```

---

```

        "delay":      100,
        "prpl":      0
    },
    "usableCapacity": 0.6
},
"capacity":          42200000
}
}

```

Note that all bandwidths are given in bit per second and all delays in milliseconds. Also, every json configuration goes through a minifier, so comments are allowed in the configuration files. An example would be the comment before the `globalRegion` option.

## Delay regions

Additionally the user has the option to define regions with individual delays. This can be used to simulate areas with bad reception, crowded areas etc.

The regions should be stored in `vsimrti/scenarios/<scenarioName>/cell2/regions.json`. An example definition for a single region called *Ernst-Reuter-Platz* could look like this:

```

{
  "regions": [
    {
      "id": "Ernst-Reuter-Platz",
      "area": {
        "nw": { "lon": 13.3249, "lat": 52.5131 },
        "se": { "lon": 13.3273, "lat": 52.5125 }
      },
      "uplink": {
        "delay": {
          "type": "SimpleRandomDelay",
          "steps": 4,
          "minDelay": 50,
          "maxDelay": 200,
          "prpl": 0.8,
          "retries": 2
        },
        "capacity": 28000000
      },
      "downlink": {
        "unicast": {
          "delay": {
            "type": "SimpleRandomDelay",
            "steps": 3,
            "minDelay": 100,
            "maxDelay": 200,
            "retries": 2
          }
        },
        "multicast": {
          "delay": {
            "type": "SimpleRandomDelay",
            "steps": 3,
            "minDelay": 120,
            "maxDelay": 220,
            "prpl": 0.8
          },
          "usableCapacity": 0.6
        }
      },
      "capacity": 42200000
    }
  ]
}

```

```

    }
  ]
}

```

Note that *nw* represents the upper-left (north-west) point of the rectangle and *se* the lower-right (south-east). For further information about the possible options, please refer to the [VSimRTI API documentation](#).

The actual configuration of the uplink and the downlink modules for each region exhibits the identical format as configuration of the `globalRegion` in the `network.json`.

When no regions are found, or if a node (a vehicle) is not within a specified region, the `globalRegion` defined in the `network.json`-File will be used as the delay model.

### Transmission simulation

One of the most important feature of the cellular simulator is an estimation of the delay experienced through the transport over the cellular network.

The cellular simulator offers various modes to estimate the delay of the transmissions. The type of estimation is specified with by `delayType` for the uplink and downlink for each region.

- `delay.type = 'ConstantDelay'`: The message is transmitted with the latency being exactly equal to `delay`.
- `delay.type = 'SimpleRandomDelay'`: The latency can assume different (randomly generated and uniformly distributed) values between `minDelay` and `maxDelay`. The number of different values is determined by `steps`.
- `delay.type = 'GammaRandomDelay'`: A gamma distribution is used to estimate the latency, with  $\alpha = 2$  and  $\beta = 2$ . The minimum delay `minDelay` is added to the result. The curve is fitted so that the maximum likelihood for the delay is exactly equal to `expDelay`.
- `delay.type = 'GammaSpeedDelay'`: This mode closely resembles the `GammaRandomDelay`. Additionally, a penalty for the velocity with which the node is moving is calculated. This penalty is then added to the original delay. The `GammaRandomDelay` and the `GammaSpeedDelay` are derived from a measurement campaign during a research project at the DCAITI.

The two different modes for the downlink are *unicast* and *multicast* which are configured separately. Multicast aims to simulate the features of Multimedia Broadcast Multicast Service ([MBMS](#)). The main difference in terms of the transmission for unicast and multicast is the handling of undeliverable messages. For unicast, the options *prpl* and *retries* are used. Pr is short for packet retransmit and denotes the probability for a failed delivery and a subsequent retransmit. The maximum number of retries for the retransmit is configured through the *retries*-option. The probability of a successful retransmit is recalculated on every try.

In case of multicast the *prpl* is used as packet loss rate. The value is factored into the delay calculation. In contrast to the unicast, just one transmission attempt is made for multicast.

### 3.6.4 Operation

Beside the transmission simulation, the Addressing and Routing is the other important aspect of the Cellular Simulator. This task is enabled by the Geocaster.

The Geocaster evaluates the message headers for cellular messages, which are created by the communicating applications in the Application Simulator.

It supports the following addressing and casting schemes.

**CellTopocast** is the *normal* unicast, where the Geocaster simply resolves the single receiver via the IPResolver. Hence, the CellTopocast directly routes the message further. Currently, Topocast doesn't allow broadcast or anycast addresses, but any transmission protocols (tcp, udp).

**CellGeoUnicast** addresses every node in the destination area individually. In this way it takes a geographic address and results in a loop to generate multiple unicasts.

**CellGeoBroadcast**, which is basically MBMS, uses one broadcast to all nodes in the destined regions. The MBMS uses the different transmission mode of multicast in the downlink. CellGeoUnicast as well as CellGeoBroadcast require broadcast, but don't allow tcp (as ack for broadcasts is denied).

## 3.7 VSimRTI Simple Network Simulator

As well as the network simulators OMNeT++ and ns-3, the VSimRTI built-in Simple Network Simulator (SNS) aims at simulating ad hoc communication for V2X scenarios. In contrast to the other simulators, SNS is already bundled with VSimRTI and can be used directly out of the box, and due to Java language, in an operating system independent way.

SNS implements abstract communication models to support the following important communication principles for V2X applications:

**TopoCast** currently implemented as Singlehop Topocast, to communicate either to all or an individual destination node in the direct communication range of the sender.

**GeoCast** to communicate, similar to Georouting, to nodes in a certain geographic destination area, with the options of unicast (to address one node), broadcast (to address all nodes) and anycast (to address the first one of all reachable nodes).

SNS does not implement a full featured IEEE 802.11p communication stack, as the other simulators could do. It does not consider particular detailed properties of the communication, e.g. signal fading models, PHY Layer reception, MAC Layer scheduling, detailed Geo Routing protocols etc. Due to this, SNS is an interesting alternative several reasons and suited for users aiming for quick and easy use without the need of detailed models.

However, when detailed communication characteristics need to be considered, the other simulators are the better choice.

### 3.7.1 sns-ambassador folder structure

The SNS can be configured via the sns\_config.json which can be found within the scenarios folder structure:

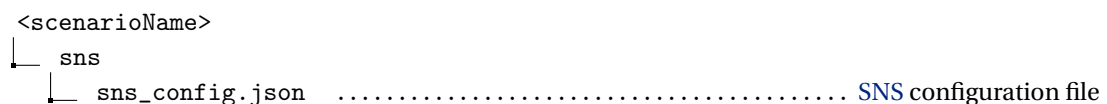


Figure 3.10: sns-ambassador folder structure

### 3.7.2 Installation

This simulator does not need to be installed. It is delivered as part of the VSimRTI-installation package.

### 3.7.3 Configuration

The SNS configuration basically consists of the two components for **singlehop** (direct Topocast) and **multihop** (Georouting) transmission. Each of these transmission modes base on the delay configuration, which is already known from the Cell2 Simulator (3.6).

The singlehop component additionally includes the singlehop **radius**, which actually is the direct communication range of the sender - meaning that all nodes within this distance can successfully receive the according message. In the context of the fundamental Freespace Model, which is implemented by the classic network simulators **OMNeT++** and **ns-3** this value results from the carrier frequency, transmission power, receiver sensitivity and antenna properties (gains) etc. Additionally, the delay supports the value for packet retransmission and packet loss (**prpl**), which allows the simulation of packet errors within the communication range. However, in contrast to fading models as the Rayleigh or Rice Fading, the simulated packet errors in **SNS** are distributed uniformly over the whole communication range and do not, for instance, increase with higher distances (which would be more realistic).

The multihop component only consists of the delay configuration including the value for **prpl** and according transmission retries. In this case, the destined communication area is determined by the dissemination area in the **GeocastDestinationAddress**.

An example definition for **SNS**, with relatively short delays in the order of microseconds or low milliseconds for direct singlehop and slightly higher delays for multihop could look like this:

---

```
{
  "version": "0.16.0",

  "singlehop": {
    "_singlehop radius in m": 0,
    "radius": 509.4,
    "_delay config according to vsimrti-communication": 0,
    "delay": {
      "type": "SimpleRandomDelay",
      "steps": 5,
      "minDelay": 0.4,
      "maxDelay": 2.4,
      "prpl": 0,
      "retries": 0
    }
  },

  "multihop": {
    "_delay config according to vsimrti-communication": 0,
    "delay": {
      "type": "GammaRandomDelay",
      "minDelay": 10,
      "expDelay": 30,
      "prpl": 0,
      "retries": 2
    }
  }
}
```

---

### 3.7.4 Operation

The individual steps of operation for the two modes of Topocast (direct singlehop) and Geocast (georouting with multihop) transmission are as following.

#### Simulation of direct singlehop transmission

1. get transmission radius from configuration (to simulate tx power, carrier frequency and antenna properties etc.)

2. determine potential receiver nodes in the sending radius (without original sender)
3. calculate equal delay for all nodes
4. simulate successful message transmission (via prpl) for each node

### **Simulation of geocast routing transmission**

1. get destination area from geocast header in message (to simulate georouting)
2. determine potential receiver nodes in the destination area (including original sender due to re-broadcasting in geocast)
3. calculate specific delay for each node
4. simulate successful message transmission (with packet loss and possible retransmissions via prpl) for each node



## 3.8 VSimRTI Eventserver

### 3.8.1 eventserver-ambassador folder structure

This ambassador can be configured with a configuration file. The specific path is  
vsimrti/scenarios/<scenarioName>/eventserver/eventserver\_config.json

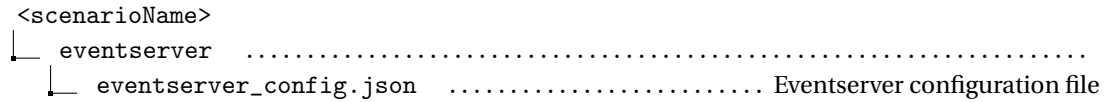


Figure 3.11: eventserver-ambassador folder structure

### 3.8.2 Installation

This simulator does not need to be installed. It is delivered as part of the VSimRTI-installation package.

### 3.8.3 Configuration

**Notice:** The documentation for the VSimRTI specific component is freely available on the [DCAITI website](#) , explaining all available options.

The root node of the configuration is a list of environment events. Each event require the type of the event, a rectangle area, a strength and the time window.

## 3.9 VSimRTI Battery Simulator

### 3.9.1 battery-ambassador folder structure

This ambassador can be configured with a configuration file. The specific path is `vsimrti/scenarios/<scenarioName>/battery/battery_config.json`

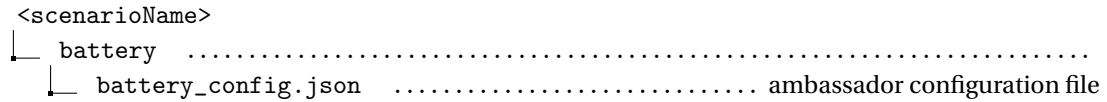


Figure 3.12: battery-ambassador folder structure

### 3.9.2 Installation

This simulator does not need to be installed. It is delivered as part of the VSimRTI-installation package.

### 3.9.3 Introduction

The VSimRTI Battery Simulator is used to simulate electric vehicles. It takes environment, vehicle characteristics and the battery itself into account. The main feature of the battery ambassador is that it can utilize dynamic class loading to use a battery model provided by the user, depending on the given needs. This also holds true for the environment model and the vehicle model. However, simple models for vehicle, environment and the battery are provided and will be briefly explained in the following sections.

### 3.9.4 Vehicle model

The vehicle model holds the general properties of a vehicle. Examples would be the weight of the vehicle and the voltage at which the electric engine operates. However, as with the other models, the provided class for the vehicle model directly affects what can be configured here. If other properties are needed for the vehicle, this is the right place to put them. To implement an own vehicle, the class `AVehicleModel` has to be extended.

### 3.9.5 Battery model

The battery model defines the battery used by the vehicle itself. Useful properties could be the number of cells and their capacity. As with the vehicle, this class can be freely defined and use configuration values placed in the `batteryModelConfig`-area. To implement an own battery model, the class `ABatteryModel` needs to be extended.

### 3.9.6 Environment model

Normally environmental factors like rolling resistance of the given underground or air drag go into this section. At the current state, just a minimal environment model is bundled with the battery ambassador, mainly to show what is possible. To implement a custom environment model, `AEnvironmentModel` has to be extended.

### 3.9.7 Sample configuration

```
{
  "vehicleModelClass": "com.dcaiti.vsimrti.fed.battery.sim.models.vehicle.ElectricSmart",
  "vehicleModelConfig": {
    "Mass": 1060,
    "ReferenceArea": 1.95,
    "DragCoefficient": 0.375,
    "TankToWheelEfficiency": 0.7,
    "ElectricMotorOperatingVoltage": 350,
    "ConsumerOperatingVoltage": 12
  },
  "batteryModelClass": "com.dcaiti.vsimrti.fed.battery.sim.models.battery.VerySimpleBatteryModel",
  "batteryModelConfig": {
    "NumberOfCells": 93,
    "CellVoltage": 4.3,
    "CellCapacityInAh": 50,
    "eff_Summer": 0.8448,
    "RechargingType": 2,
    "MinSOC": 0.30,
    "MaxSOC": 0.70
  },
  "environmentModelClass": "com.dcaiti.vsimrti.fed.battery.sim.models.environment.DefaultEnvironment",
  "environmentModelConfig": {
    "Gravity": 9.81,
    "FluidDensity": 1.293,
    "RollingResistanceCoefficient": 0.01
  },
  "consumers": [ { "Radio": 10 }, { "HeadLight": 100 } ]
}
```

This listing shows how the vehicle, environment and battery model classes for the bundled models are configured. Additionally, an arbitrary number of consumers can be configured that draw additional power from the battery. In this case, headlights and a radio are pre-defined.

## 4 Tutorial Tiergarten

This tutorial aims to provide a general overview of the VSimRTI application concept and shows two examples that involve ad hoc communication via IEEE 802.11p between different participants and message passing from one application to another that run on the same vehicle. Additionally, a short introduction will be given on Traffic Lights, with a more in-depth description following in Chapter 6. The tutorial is split up into five main parts:

1. How to equip a vehicle with one or more applications. This process is called *mapping*.
2. An example application that shows how to implement communication between different vehicles via a wireless ad hoc network.
3. The next part of the tutorial shows how to accomplish message passing between two applications that run on the same vehicle.
4. An overview of the traffic light application and a summary about what can be done with the traffic light from within the application.
5. The last part of the tutorial shows how to retrieve the results of a simulation run.

The scenario itself consists of three vehicles that drive down a road in consecutive order and pass a Road Side Unit (RSU) that emits messages in a fixed interval. The vehicles will receive the messages from the RSU as soon as they are in communication range. After completing this tutorial the reader should be

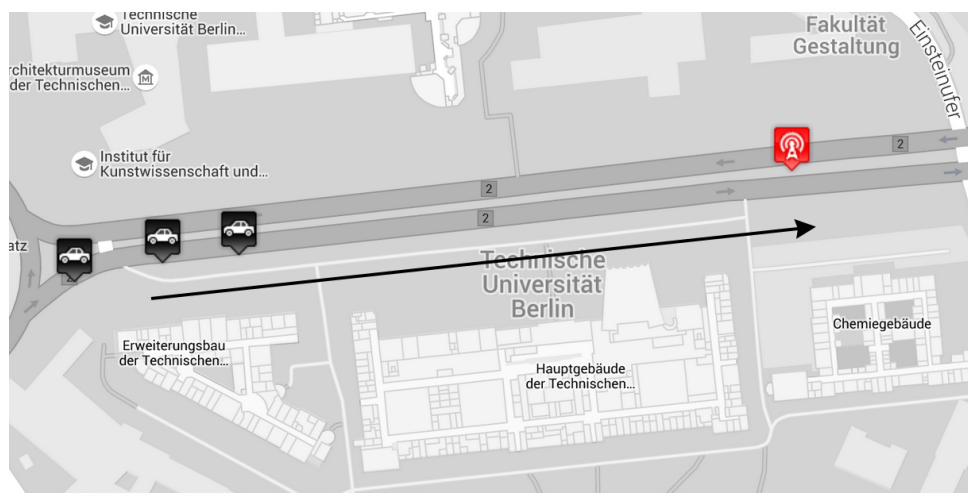


Figure 4.1: Overview of Tiergarten tutorial scenario

able to deploy own applications according to his needs and make use of ad hoc communication among vehicles and intra-vehicle communication among applications on the same vehicle. The VSimRTI cell simulator that is used to simulate cellular network communication will be covered in tutorial 2.

## 4.1 Mapping Configuration

In order to use applications they have to be assigned (*mapped* in VSimRTI terminology) to a simulation entity. In this tutorial, we will assign applications to a RSU that is placed along the road (the red symbol in picture 4.1) and to the vehicles. In order to do this the following steps are necessary:

1. Navigate to the `mapping3` folder of the tiergarten tutorial scenario.
2. Edit the `mapping_config.json` to let it point to the correct classes for your application, define prototypes and to add entities to the simulation.

The mapping is already configured correctly, so we will continue with a description of the important parts of the mapping configuration.

### Prototype Section

This section contains the properties of the simulated entities, including their applications. You can, for example, configure the maximum speed and the length of a vehicle. Normally, the default values are fine here. In order to map one or more applications you have to fill in the complete class identifier including the package in the `applications` array. In this tutorial, we have mapped two applications to the vehicles and one application to the RSU.

Mapping for the vehicles:

```
"applications": ["com.dcaiti.vsimrti.app.tutorials.tiergarten.TiergartenVehicle",  
"com.dcaiti.vsimrti.app.tutorials.tiergarten.TiergartenVehicleSlave"]
```

Mapping for the RSU:

```
"applications": ["com.dcaiti.vsimrti.app.tutorials.tiergarten.TiergartenRSU"]
```

In order for VSimRTI to be able to locate these classes, the resulting `.jar` archive should be placed in the `applicationNT` folder of your scenario. For our tutorial we have packaged the needed classes `TiergartenVehicle`, `TiergartenVehicleSlave` and `TiergartenRSU` into one `.jar` file. However, this isn't a strict requirement. What these classes actually do and how they are implemented will be covered in the next two parts of this tutorial.

### Vehicle and RSU sections

This part of the mapping is used to actually bring entities like vehicles and RSUs into the simulation. For example, we placed the RSU that is equipped with the `TiergartenRSU` application at the specific geo coordinate with `latitude = 52.5130607` and `longitude = 13.328910`. Usually, this and the mapping of its application is all that is necessary to make use of an RSU. For this tutorial we go with just one RSU but analogous to adding multiple vehicles we could also add more of them if the need arises.

Spawning vehicles is a little bit more complex than adding a RSU. Vehicles are configured as groups from which we have three in this tutorial scenario. Exemplary, we will describe one of these groups in more detail:

---

```

"vehicles": [
  {
    "startingTime": 1.0,
    "route": 0,
    "maxNumberVehicles": 1,
    "types": [{"name": "PKW"}]
  },
  ...
]

```

---

Listing 4.1: Example Vehicle Group

- `startingTime`: Defines at which second the vehicle is spawned, in this case one second after the beginning of the simulation.
- `route`: Describes which route these vehicle group should use. For now it is sufficient to know that there is one route with the id 0 that leads the vehicles near the RSU that is already defined in this tutorial scenario.
- `maxNumberVehicles`: Used to determine how many vehicles should be in that group. Here, we go with just one single vehicle.
- `types`: This array maps to the defined PKW (german for passenger car) prototype from the `proto-types`-section of the mapping.

## 4.2 Inter-Vehicle Communication

This second part describes the `TiergartenVehicle` and `TiergartenRSU` applications which are used for inter-vehicle communication in more detail. As a coarse overview, the `TiergartenRSU` application sends out a defined message at a fixed interval. These messages are received and written to the log file by the `TiergartenVehicle` application. First, we start off with the class definition and will run through the methods we have to implement in order to get the communication working.

### Class definition

---

```

public class TiergartenVehicle extends AbstractApplication<VehicleOperatingSystem> implements ✓
    ↘ VehicleApplication, CommunicationApplication

```

---

Listing 4.2: TiergartenVehicle: Class definition for a vehicle application

In general, every vehicle application will extend the `AbstractApplication` abstract class with the `VehicleOperatingSystem` type parameter. The `VehicleOperatingSystem` is the main way to interact with the underlying vehicle and gives a wide array of functionality to the user. Also, depending on the needs of the application, several interfaces have to be implemented. For our scenario we implement `VehicleApplication` to denote that the application is intended to run on a vehicle and `CommunicationApplication` to be able to use the actual communication facilities we need for this tutorial. In general, it makes sense to let the IDE generate the bodies of the methods that need implementation and fill them with functionality if needed or letting them empty otherwise.

## Application initialization

During the initialization procedure of communicating applications, the communication modules (WLAN-Module or CellModule) need to be activated. For example, activating the WLAN module can be achieved with the following code snippet:

```
@Override
public void setup() {
    getLog().infoSimTime(this, "Initialize application");
    getOperatingSystem().getAdHocModule().enable(new AdHocModuleConfiguration()
        .addRadio().channel(AdHocChannel.CCH).power(50).create()
    );
    getLog().infoSimTime(this, "Activated AdHoc Module");
}
```

Listing 4.3: TiergartenVehicle activate WLAN module

Since we want to communicate over ad hoc Wifi we have to turn on the Wifi-module of the car. Here you can specify the mode of operation. The first argument says if the vehicle should be able to receive messages, how strong the Wifi antenna is (50mW in this case) and which ad hoc channel to use.

**Note:** The honoring of these arguments depends on the underlying network simulator. Currently, [ns-3](#) and [OMNeT++](#) make use of these arguments.

## Receiving the V2X messages

In order to act upon received messages from other simulation entities, the `receiveV2XMessage` from the `CommunicationApplication` interface is used. In our tutorial scenario we don't do something special upon message receiving and just write the name of the sender into the log file.

```
@Override
public void receiveV2XMessage(ReceivedV2XMessage receivedV2XMessage) {
    getLog().infoSimTime(this, "Received V2X Message from {}",
        receivedV2XMessage.getMessage().getRouting().getSourceAddressContainer().getSourceName());
}
```

Listing 4.4: TiergartenVehicle: Class definition for a vehicle application

This is basically all it takes to receive messages from another simulation entity. Normally, the application author uses an own message class that derives from `V2XMessage` and casts it to his specific class in order to handle messages easier.

## Sending the V2X messages

Since the receiving application is now set up we move on to the sending side of this tutorial. The sending takes place from the RSU via a broadcast, so every vehicle in transmission range will receive the message sent by it. [Picture 4.2](#) shows the communication range of the RSU which is dependent from the settings in the communication simulator. The actual sending takes place in the `TiergartenRSU` application which is equipped to the RSU via the mapping configuration.

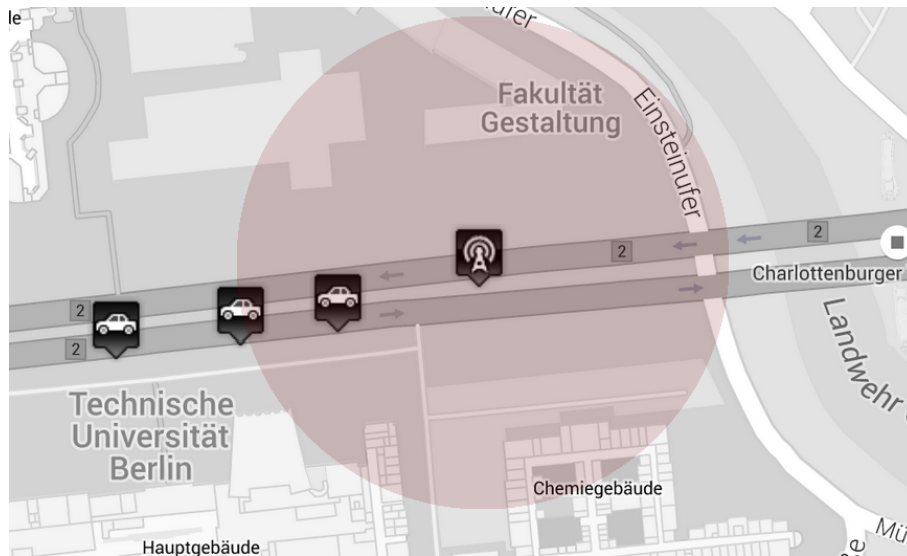


Figure 4.2: Example of RSU communication range

### ApplicationNT event model

In order to send messages at a fixed interval we make use of the event based model of the VSimRTI application simulator. A high level description in what we need to do in order to send messages at a specific interval can be summarized as follows:

1. Line up an event every X seconds. For this tutorial an interval of two seconds was chosen.
2. Once the event gets processed in the `processEvent()`-method the actual sending of the message will be triggered.
3. All information necessary for the sending will be assembled in `sendAdHocBroadcast()` and the message gets actually sent out.

### Application Setup

```

@Override
public void setUp() {
    getLog().infoSimTime(this, "Initialize application");
    getOperatingSystem().getAdHocModule().enable(new AdHocModuleConfiguration()
        .addRadio().channel(AdHocChannel.CCH).power(50).create()
    );
    getLog().infoSimTime(this, "Activated WLAN Module");
    sample();
}

```

Listing 4.5: TiergartenRSU: Setup for the RSU application

The setup for the RSU is the same as for the vehicle where we activate the Wifi module. Additionally, the `sample()` method gets called which is responsible for the events.



## Event scheduling

The next step is to line up the event at the interval we desire. The method we use for that, `sample()`, looks like this:

---

```
public void sample() {
    final Event event = new Event(getOperatingSystem().getSimulationTime() + TIME_INTERVAL, ↵
        ↵ this);
    getOperatingSystem().getEventManager().addEvent(event);
    getLog().infoSimTime(this, "Sending out AdHoc broadcast");
    sendAdHocBroadcast();
}

```

---

Listing 4.6: TiergartenRSU: Event sampling

Here, an `Event` is created which will be received at the current time in simulation plus the defined interval which is set to two seconds. The second argument for the event creation is a reference to the originating class instance. For our tutorial, the `this` reference is sufficient here. After we created the event, it is added to the event queue via `addEvent()` which will result in a call to `processEvent()` at the given interval.

## Message sending

After we made sure the method that does the actual sending of the message is called at the specified interval, we take a closer look at the `sendAdHocBroadcast()` method we defined in the `TiergartenRSU` class:

---

```
private void sendAdHocBroadcast() {
    final TopocastDestinationAddress tda = TopocastDestinationAddress.getBroadcastSingleHop ↵
        ↵ ();
    final DestinationAddressContainer dac = DestinationAddressContainer. ↵
        ↵ createTopocastDestinationAddressAdHoc(tda);
    final MessageRouting routing = new MessageRouting(dac, getOperatingSystem(). ↵
        ↵ generateSourceAddressContainer());
    final InterVehicleMsg message = new InterVehicleMsg(routing, getOperatingSystem(). ↵
        ↵ getPosition());
    getOperatingSystem().getAdHocModule().sendV2XMessage(message);
}

```

---

Listing 4.7: TiergartenRSU: Event sampling

We step through this method line by line and explain what each statement does:

1. The destination address for this method is constructed in the first line. We use the helper-method `getBroadcastSingleHop()` to denote that we want to send a single hop broadcast.
2. After that, we wrap the destination address in a container that is created via `createTopocastDestinationAddressAdHoc()`. Since we use ad hoc wifi for this tutorial this method will suffice.
3. The next step is to create the routing for the message. The routing basically ties together a destination address and a source address. We use the destination address we created in the first two lines and use the default source address that can be created with the `generateSourceAddressContainer()` method.
4. The last step in crafting the message is to create an instance of our custom message class `InterVehicleMsg`. The actual payload is the position of the sender, other payloads are possible depending on the requirements of the application.

5. The message is sent out using the `sendV2XMessage()`-method that is provided by the operating system.

These steps conclude the sending of the message. The procedure is the same for sending messages from a vehicle instead of a RSU.

## 4.3 Intra-Vehicle Communication

This part of the tutorial describes the steps necessary for letting two applications to communicate with each other on the same vehicle. Here, we use the `TiergartenVehicle` application as the sender and `TiergartenVehicleSlave` as the receiver. In general, the approach is similar to the sending of a V2X message and also makes use of the event system.

### Message sending

First, we start off with the sending side. The event code should look familiar:

---

```
@Override
public void afterUpdateVehicleInfo() {
    final List<? extends Application> applications = getOperatingSystem().getApplications();
    final IntraVehicleMsg message = new IntraVehicleMsg(getOperatingSystem().getId(),
        random.nextInt(MAX_ID));

    for (Application application : applications) {
        final Event event = new Event(getOperatingSystem().getSimulationTime() + 1, application,
            message);
        this.getOperatingSystem().getEventManager().addEvent(event);
    }
}
```

---

Listing 4.8: TiergartenVehicle: Sending intra application message

One noteworthy thing is that we use the `afterUpdateVehicleInfo()`-method to line up a new event. This method is called automatically if the vehicle info (for example, speed or heading) gets updated, which usually takes place at every time step of the simulation.

The general approach works like this:

1. Get a list of all applications that run on the vehicle using `getApplications()`.
2. Create a new message. Here we use our own custom message called `IntraVehicleMsg` which takes an randomly generated id and the name of the vehicle as payload. Again, this is for tutorial purposes and could be anything.
3. After that we iterate over every applications that runs on the vehicle, in this case two: `TiergartenVehicle` and `TiergartenVehicleSlave`.
4. Then, an event is constructed for each app running on the vehicle and added to the event queue the same way as in the inter-vehicle example.

## Receiving the message

The actual receiving of the message takes place in the `TiergartenVehicleSlave` application. Since the intra vehicle messages are basically treated as an event, the code is straight forward here:

```
@Override
public void processEvent(Event event) throws Exception {
    Object resource = event.getResource();
    if (resource != null && resource instanceof IntraVehicleMsg) {
        final IntraVehicleMsg message = (IntraVehicleMsg) resource;
        if (message.getOrigin().equals(getOperatingSystem().getId())) {
            getLog().infoSimTime(this, "Received message from another application");
        }
    }
}
```

Listing 4.9: TiergartenVehicleSlave: Receiving intra application message

The general concept here is that the payload is wrapped as an object in the actual event. The procedure here is as follows:

1. Retrieve the resource from the event using `getResource()`
2. Check if there is actually some kind of payload
3. Make sure the payload is of the expected type
4. Cast the payload to an actual class and do something with it.

### 4.3.1 The traffic light application

As can be seen in the mapping configuration there is an additional prototype defined for traffic lights:

```
{
    "applications":["com.dcaiti.vsimrti.app.tutorials.tiergarten.trafficLight.<
        ↵ TrafficLightApp"],
    "name":"TrafficLight"
}
```

Listing 4.10: Traffic Light Prototype

These prototype is used to map the referenced application onto two specific traffic lights, as shown in the following listing.

```
tls": [
    {
        "tlName": "27011311",
        "name": "TrafficLight"
    },
    {
        "tlName": "252864801",
        "name": "TrafficLight"
    }
]
```

Listing 4.11: Traffic Light Mapping

The use case shown for tutorial purposes here is simple: The two traffic lights referenced in the mapping default to *always red*. Their application then waits for a V2X message with a “*secret*” payload in it (the `GreenWaveMsg`) and switches the traffic light program to *always green* upon receiving that message. The application mapped onto the traffic light then switches back to its previous program after 20 simulation time steps have passed. An in-depth explanation of traffic light functionality can be found in Chapter 6.

## 4.4 Interpretation of simulation results

The last part of the tutorial describes how to retrieve the actual simulation results. For this tutorial, the results are quite simple and simply show the arrival of the Inter- and IntraVehicle messages. So after executing the simulation using the following command:

---

```
./vsimrti.sh -u <user> -s Tiergarten
```

---

Listing 4.12: Running the simulation with the Tiergarten scenario (Unix machine)

---

```
vsimrti.bat -u <user> -s Tiergarten
```

---

Listing 4.13: Running the simulation with the Tiergarten scenario (Windows machine)

Afterwards, in the log directory of `VSimRTI` a new folder should be created containing all log files of the simulation run. Within, the sub-folder `appsNT` contains the log files for each simulation unit and its application. For example, for the vehicles we end up with two log files: `TiergartenVehicle.log` and `TiergartenVehicleSlave.log`.

This following snippet shows the receiving of the V2X messages that were sent by the RSU:

---

```
INFO - Initialize application (at simulation time 6.000,000,000 s)
INFO - Activated AdHoc Module (at simulation time 6.000,000,000 s)
INFO - Received V2X Message from rsu_0 (at simulation time 18.000,400,000 s)
INFO - Received V2X Message from rsu_0 (at simulation time 20.000,400,000 s)
```

---

Listing 4.14: Snippet from `TiergartenVehicle.log`

Next, we see the contents of the `IntraVehicleMessage` that originates from another app on the vehicle:

---

```
INFO - Initialize application (at simulation time 2.000,000,000 s)
INFO - Received message from another application: IntraVehicleMsg{origin='veh_0', id=631} ...
```

---

Listing 4.15: Snippet from `TiergartenVehicleSlave.log`

The following log was generated by the RSU application and logs the sending of each ad hoc broadcast:

---

```
INFO - Initialize application (at simulation time 0.000,000,000 s)
INFO - Activated WLAN Module (at simulation time 0.000,000,000 s)
INFO - Sending out AdHoc broadcast (at simulation time 0.000,000,000 s)
INFO - Sending out AdHoc broadcast (at simulation time 2.000,000,000 s)
```

---

Listing 4.16: Snippet from `TiergartenRSU.log`

This concludes the first tutorial and hopefully gave an idea on how to use the `VSimRTI` application simulator to send out and receiving messages to other simulation entities or inside the same vehicle.

The `OperatingSystem.log` files do not contain specific application output and is mainly used for debugging purposes and won't be discussed in this tutorial.

## 5 Tutorial Barnim

The following tutorial shows some of the more advanced features of VSimRTI and will take a closer look at different *federates*. The term *federates* describes the High Level Architecture and basically denotes a dedicated simulator with whom the simulation infrastructure can communicate. The following topics will be covered in this tutorial:

- How to use and react to DENMs *Decentralized Environmental Notification Message*.
- Enabling and using a different communication simulator on the basis of the VSimRTI cell simulator.
- The integration of electric mobility aspects into a simulation.

**Note:** The topics from the last tutorial, especially how to map applications and the basics of how to implement them are not covered in this tutorial. If you are not yet familiar with these concepts you should have a look at the preceding tutorials.

### 5.1 Overview

The scenario used for this tutorial is more complex than the Tiergarten tutorial; thus, we will provide you a short explanation. Picture 5.1 shows an overview of the Barnim scenario, exhibiting an icy road section along the highway.

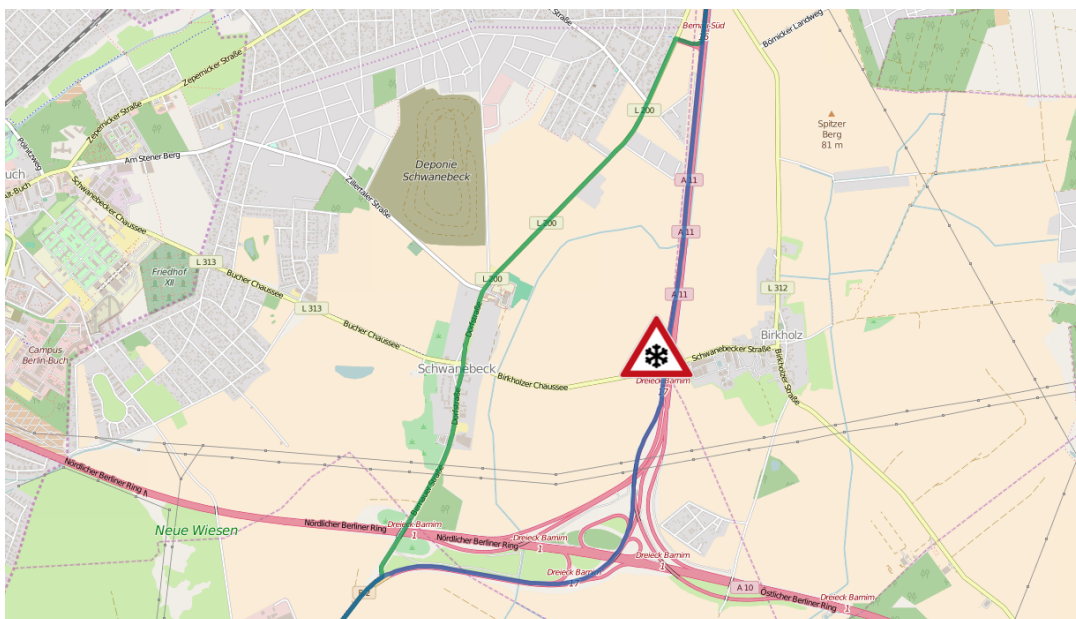


Figure 5.1: Overview of Barnim tutorial scenario

In this scenario, several cars drive on the blue route and are forced to slow down in a specific section due to icy conditions. The rest of the scenario can be described as follows:

1. A car (**Car A**) which is equipped with ad hoc communication (WiFi) capabilities detects an environmental hazard - in this case an icy section of the road.
2. **Car A** now sends out a DENM which reaches other cars in its (relatively small) dissemination area. With multi hop routing, the DENM message is transported upstream towards other vehicles.
3. Cars that do not have any form of communication equipment are not warned and drive towards the icy part of the road. Since they have careful and responsible drivers they slow down to avoid accidents.
4. Cars that are equipped with the appropriate communication equipment are able to receive the DENM, which induces them to use a different route (green) which is safer and faster due to the lack of ice on it.
5. Last but not least, the **WeatherServer** (technically implemented as an RSU) propagates information over the cellular network and could therefore be located virtually everywhere.

We also have some electric vehicles here, mainly to demonstrate how the VSimRTI battery ambassador works. Later in this tutorial, we will show you how to access statistics such as the state of charge of the battery from within an application.

## 5.2 Mapping configuration

This section gives a short explanation of the mapping we use in this scenario. First of all we use five different types of entities. One RSU which acts as the WeatherServer and four types of cars, each of them loaded with different applications. As usual, the configuration takes place in `mapping3/mapping_config.json` in your scenario folder.

1. Normal vehicles which cannot communicate shall demonstrate how to use electric vehicles in a scenario (60% of all vehicles). They are only equipped with the `com.dcaiti.vsimrti.app.tutorials.barnim.SlowDownApp`-application which induces them to reduce their speed as soon as their onboard sensors detect hazardous conditions. After leaving the hazardous area, they will resume by increasing their speed again.
2. Vehicles equipped with ad hoc wifi. They might not be able to receive the DENM due to range limitations and drive into the icy section nonetheless. They are equipped with the `com.dcaiti.vsimrti.app.tutorials.barnim.WeatherWarningApp`-application and the `com.dcaiti.vsimrti.app.tutorials.barnim.SlowDownApp`-application (20% of all vehicles).
3. Cellular communication enabled vehicles which are able to communicate with the WeatherServer. These vehicles are equipped with the `com.dcaiti.vsimrti.app.tutorials.barnim.WeatherWarningAppCell`-application, which is a specialized form of the normal weather warning application that can make use of cellular communication. They are also equipped with the `com.dcaiti.vsimrti.app.tutorials.barnim.SlowDownApp`-application (10% of all vehicles).

4. Electric vehicles cannot communicate since they are only equipped with the `com.dcaiti.vsimrti.app.tutorials.barnim.SlowDownApp`-application. Apart from that, they also incorporate an electrical battery which empties during driving (10% of all vehicles).
5. The WeatherServer is only equipped with cellular equipment. Despite the greater distance it is able to warn vehicles that can also make use of cellular communication.

## 5.3 VSimRTI configuration

Since several applications are mapped, we need to make sure that the corresponding simulators are enabled. This is done in the main VSimRTI configuration file that resides in `vsimrti/vsimrti_config.xml` in your scenario. The last part of that configuration file is used to enable and disable certain simulators according to the needs of the user.

Per default, any simulation of cellular communication is disabled. In order to enable communication via cellular networks in this scenario, you need to enable the `cell2` simulator by setting the field `active` to `true`:

```
<!-- Cellular network simulators -->  
<federate id="cell2" active="true"/>
```

Listing 5.1: VSimRTI configuration for barnim example scenario

## 5.4 Applications

This section deals with the specific details of the `WeatherWarningApplication`. We will cover in particular how to enable and use cellular communication, how to react to DEN-Messages and how the user can achieve a re-routing of vehicles.

### Overview

Here we will explain the application logic and the main ideas behind it in general terms and will go into the implementation details in the following sections. All of the subsequent points stay the same regardless of what form of communication is used.

- If the sensor picks up an event, the vehicle sends out a DEN-message (via cellular communication or ad hoc) with the position that was recorded in order to warn other vehicles.
- The application also contains code to react upon DENMs received from other vehicles. Depending on the configured behavior of the application, either a slow down or a re-routing is performed.

Most of the things that are happening here like re-routing, interval-handling and sending out ad hoc messages were already covered in the first tutorial and will not be discussed in detail in this tutorial.

### 5.4.1 DEN-Message handling

#### Receiving

First of all, we look at the receiving side of the DENM here. Since the message is a normal V2X message it is received through the `receiveV2XMessage()`-method which is part of the application interface.

```
if (!(msg instanceof DENM)) {
    getLog().info("Ignoring message of type: {}", msg.getClassName());
    return;
}
```

Listing 5.2: Testing whether or not message is a DENM

Analogous to a normal message, we check with `instanceof`, if it is of a type that we are interested in, in this case DENM. Afterwards, we perform a potential route change if not already done.

#### Sending

In case the sensor detects an environmental hazard the vehicle sends out a DEN-message to warn other vehicles. If `WeatherWarningAppCell` is mapped, cellular communication is used, in case of `WeatherWarningApp` ad hoc WiFi is used as described in the Tiergarten tutorial.

### 5.4.2 Cellular Communication

We will now take a closer look at how the communication via the VSimRTI cellular simulator works. The application we point to in the mapping is a specialized form of the normal `WeatherWarningApplication` and can be found under its full identifier `com.dcaiti.vsimrti.app.tutorials.barnim.WeatherWarningAppCell`. It overloads the `useCellNetwork`-method which returns a constant `true` in order to indicate that we want to use cellular communication inside the base application. The reason for this overloading is to provide a distinct class to which we can point in the mapping. The actual logic that is required to send messages via cell is done in the base `WeatherWarning` application, so the overloading just signals that we want to use cellular communication.

The method where the actual sending takes place is called `reactOnEnvironmentData()`. The following code snippet shows the steps necessary to send a DEN-message via the cellular network:

```
GeoCircle dest = new GeoCircle(v_longLat, 3000);
GeocastDestinationAddress gda = GeocastDestinationAddress.createBroadcast(dest);
final DestinationAddressContainer dac;

if (useCellNetwork()) {
    dac = DestinationAddressContainer.createGeoUnicastDestinationAddressCell(gda);
} else {
    dac = DestinationAddressContainer.createGeocastDestinationAddressAdHoc(gda);
}

MessageRouting mr = new MessageRouting(dac, getOperatingSystem().generateSourceAddressContainer ↵
    ↵ ());
DENM denm = new DENM(mr, getOperatingSystem().getSimulationTime(), v_longLat, roadId, type, ↵
    ↵ strength, newSpeed, 0.0f);
getOperatingSystem().getAdHocModule().sendV2XMessage(denm);
```

Listing 5.3: Example for sending over cellular networks



The sending is the same as in the Tiergarten tutorial, except that we decide via an `if`-statement whether or not cellular communication or ad hoc WiFi is used. Note the two different methods - one is ending with `Ce11` the other with `AdHoc`, depending on what we want to use in the application.

## 5.5 Conclusion

This concludes the description of the VSimRTI tutorial scenarios Tiergarten and Barnim. The following list sums up the things we have covered in this tutorial series:

- Getting a general idea on how to configure VSimrti, especially how to enable and disable different, coupled simulators.
- How to map different applications to a vehicle or a road side unit
- Achieving inter- and intra-application communication...
- ... via cellular networks or AdHoc wifi
- Receiving and handling of messages
- Basic interaction with the `OperatingSystem` by means of changing a route
- How to find and process the results of a simulation run with the help of generated log files.

## 6 Tutorial Traffic Lights

In this brief tutorial we will discuss how to interact with traffic lights via VSimRTI. Usually the position and the lanes they control are coming from the *OpenStreetmap*-data, which we will also assume during this tutorial. Note that this tutorial is intended to be used with the **Tiergarten** scenario described in the first tutorial in Chapter 4, so creating a new scenario for is not necessary here and the Traffic Light data is already exported.

At first we cover the general steps necessary to enable traffic lights and eventually “map” applications onto them. Each step is described in more detail later on.

1. Tell `scenario-convert` to export the traffic light information into the database
2. Determine the node ids of the traffic lights which should have applications on them
3. Adjust the mapping configuration of VSimRTI accordingly and assign the desired application(s) onto one or more traffic lights.

Additional information on how to use traffic lights, how they are modeled within sumo and how their phases work can be found in the sumo wiki under

[http://sumo.dlr.de/wiki/Simulation/Traffic\\_Lights](http://sumo.dlr.de/wiki/Simulation/Traffic_Lights)

### 6.1 Use scenario-convert to export traffic lights

Although the Tiergarten scenario which we are focusing on in this tutorial already has the traffic lights exported from the `.osm` file, this step is important if you desire to set up an own scenario. For the Tiergarten example, the `scenario-convert` command line looks like this:

---

```
scenario-convert-18.0.jar --osm2sumo -d tiergarten.db -i tiergarten.osm -l -n
```

---

Listing 6.1: Export of traffic lights using `scenario-convert`

Note the `-l` switch here that actually exports the traffic light information from the `.osm` file to the `tiergarten.db` database. If this command line switch is omitted the traffic lights will not be available in the scenario. More information on how traffic lights are handled by Open Streetmap can be found in their wiki: [http://wiki.openstreetmap.org/wiki/Tag:highway%3Dtraffic\\_signals](http://wiki.openstreetmap.org/wiki/Tag:highway%3Dtraffic_signals)

## 6.2 Determine the traffic lights that should be equipped with applications

The next step is to decide which traffic lights should have applications on them. Since their ID(s) have to be referenced in the mapping, we will have to determine them first. There are several ways to do this, e.g. taking them directly from the database using a tool like *sqlitebrowser* or extract them directly with a text editor from the *.net.xml* file that gets generated by *scenario-convert* for usage with *sumo*. However, here we will focus on a graphical approach using *sumo-gui* since it is the easiest way to determine the exact traffic lights we want to map applications too.

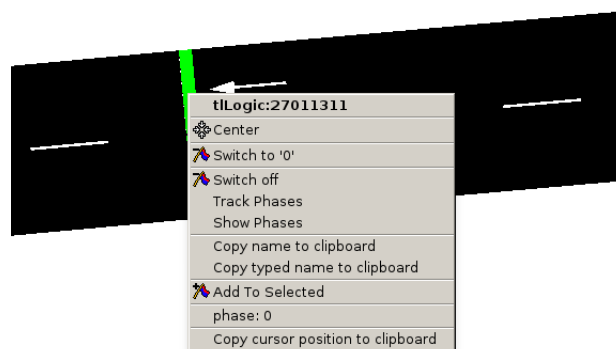


Figure 6.1: Locating traffic lights in sumo gui

Figure 6.1 shows how to locate a specific traffic light, or more precise, a traffic light group. The process is as simple as scrolling to the desired traffic light, right click on it and write down the number behind **tlLogic** in the drop down menu or use the *copy to clipboard* feature of the *sumo* gui. This id is then used in the next section which explains how to actually map applications to a traffic light.

## 6.3 Configure the mapping for traffic lights

Now that we have the ids of the traffic lights we can modify the mapping to assign application(s) to them. Note that the tutorial scenario already has applications assigned to two arbitrary traffic lights, so the mapping file bundled with it can also be used as a starting point here. In general, the mapping looks the same as for vehicles or RSUs. First, we define a prototype that is used for all subsequent traffic lights:

```

1 {
2   "prototypes": [
3     {
4       "applications": ["com.vsimrti.applications.additional.samples.misc.↵
                    ↵ TrafficLightExample"],
5       "name": "TrafficLight"
6     }
7   ]
8 }
```

Listing 6.2: Prototype for traffic light

The listing 6.2 shows an additional prototype called *TrafficLight* that gets associated with the *TrafficLightExample* application. Note that the application that should run on the traffic light has to extend `AbstractApplication<TrafficLightOperatingSystem>` in order to work. The next step is to use this prototype and assign it to a traffic light we found in the section before using the sumo gui:

```
1 "tls": [
2   {
3     "tlName": "27011311",
4     "name": "TrafficLight"
5   },
6   {
7     "tlName": "252864801",
8     "name": "TrafficLight"
9   }
10 ]
```

Listing 6.3: Example mapping of traffic lights

Listing 6.3 shows as an example how this prototype is assigned via the *name* attribute to specific traffic lights. The important aspect here is that *tlName* refers to the id of the traffic light we found via the sumo gui in section 6.2 which must exactly match each other.

## 6.4 The Traffic Light API

In order to access the traffic light functionality from within an application it must extend the `AbstractApplication<TrafficLightOperatingSystem>` abstract class. There are two main functionalities provided by it, reachable from the application operating system that can be acquired by using the `getOperatingSystem()` method:

- Change the currently active traffic light program via `setProgramById()`
- Adjust the remaining time of the currently active phase via `setPhaseRemainingDuration()`

However, adding a new traffic light program from within the application at runtime using the *VSimRTI* application interface is currently not possible and has to be done beforehand via sumo as described here: [http://sumo.dlr.de/wiki/Simulation/Traffic\\_Lights](http://sumo.dlr.de/wiki/Simulation/Traffic_Lights). The two main ways of doing this is to use an `additional-file` or the `tls_csv2SUMO.py` tool bundled with sumo. However, it is possible to add a new program by using the *TraCI* interface as described here: [http://sumo.dlr.de/wiki/TraCI/Change\\_Traffic\\_Lights\\_State](http://sumo.dlr.de/wiki/TraCI/Change_Traffic_Lights_State).

## 7 Tutorial LuST

The Luxembourg SUMO Traffic (**LuST**) scenario is a traffic simulation scenario which aims to provide realistic traffic patterns of a common mid-size European city during an average day [4]. This traffic simulation scenario for **SUMO** has been developed by Lara Codecá et al. and is publicly available via GitHub<sup>1</sup>. The following tutorial shows, how the **LuST** scenario can be integrated into **VSimRTI** in order to assess your own mobile application at large scale.

**i Notice:** Also, you can use this scenario as a blueprint for integrating your own prepared SUMO scenarios into VSimRTI.

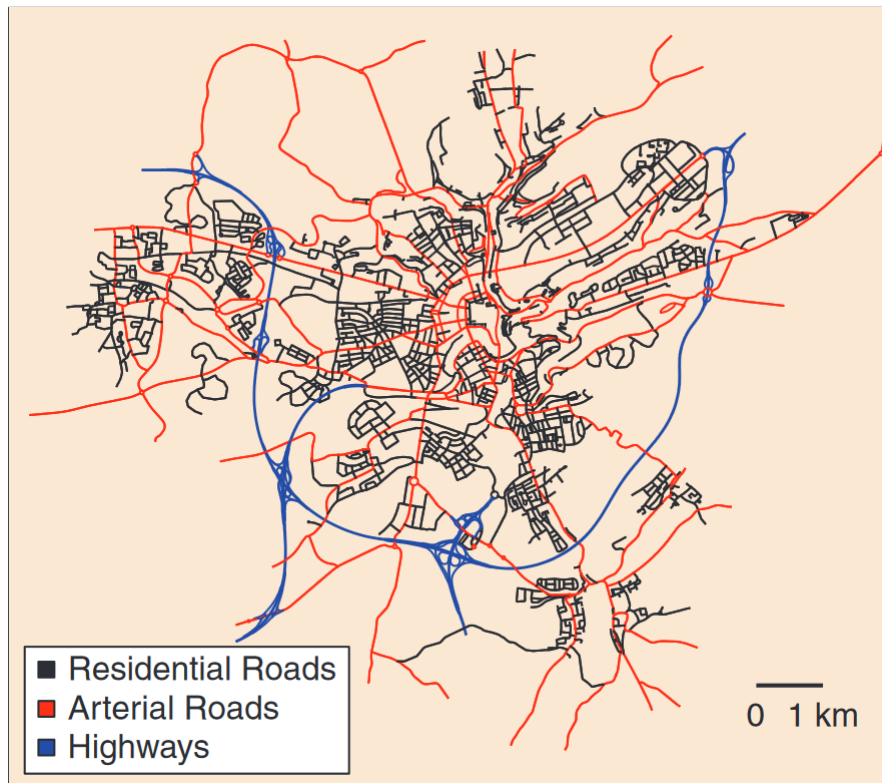


Figure 7.1: Overview of the simulation site of the LuST scenario [4]

- Area: **156 km<sup>2</sup>**
- Total length of all roads: **930 km**
- Total length of highways: **89 km**
- Number of intersections: **4.473**
- Number of traffic lights: **203**
- Number of vehicles during 24h of simulation time: **215.526**

<sup>1</sup><https://github.com/lcodeca/LuSTScenario>

## 7.1 Execute the LuST scenario with VSimRTI

You can find a prepared VSimRTI scenario for the LuST scenario in the pre-installed scenario directory. However, to get this scenario working, you need to execute the following steps:

1. In the subdirectory `sumo` of the LuST scenario directory, you can find several scripts which will help you to download the actual SUMO scenario from the official sources on GitHub. Depending on which operating system you use and which version control system client you have installed, you can use one of the provided batch or shell script files. If you do not have a git or svn client installed, you can also manually download and unzip the SUMO scenario from <https://github.com/lcodeca/LuSTScenario>. Please note that the SUMO scenario must be placed inside the `sumo` subdirectory of the scenario.
2. In order to execute VSimRTI with the scenario, you need to pass the given `defaults.xml` to the executable. On Windows, this can be done by executing VSimRTI as follows:

---

```
vsimrti.bat -u <user-id> -s LuST -d scenarios/LuST/defaults.xml -w 0
```

---

Please note that the execution of this scenario might take a long time, due to the massive amount of vehicles simulated in this scenario. If you want to assess your mobility applications in combination with further simulators (e.g. Cell2), the simulation might slow down even more.

Finally, you can map your own applications onto vehicles using the prepared `mapping_config.json`. Please note that applications can only be mapped for each vehicle type (see limitations below). Furthermore, you can activate any communication simulator, such as Cell2 or SNS, to integrate communication simulation into your assessment.

## 7.2 Current limitations

The integration of the LuST scenario is currently a work in progress and therefore not all features are available. The following limitations currently exist:

- **No mapping of applications on single vehicles.** It is currently not possible to map applications on single vehicles. Instead, only vehicle types can be mapped with them. Of course, the applications are running solely on each simulated vehicle which belongs to the respective vehicle type the application is mapped on.
- **No definition of own vehicle spawner.** Currently, all simulated vehicles are spawned by the SUMO traffic scenario itself. There is no way to add additional vehicles via the `mapping_config.json`.
- **No routing capabilities in applications.** Applications cannot use the navigation module of the vehicle. Furthermore, detailed information about the current route or the road position (upcoming nodes, type of the current road) is also not available to the applications. This is due to the missing scenario-database, which is currently not possible to create from the LuST road network.
- **No valid support of traffic light applications.** The mapping of applications onto traffic lights has not yet been tested.
- There may be more limitations we are currently not aware of.

## 8 Create a new scenario

This chapter is intended to act as a checklist for all tasks, needed to create a new scenario. Basically, a new scenario means that the simulation is performed in a different road network where vehicles have different routes. Moreover, the number of vehicles can also vary.

### 8.1 Road network data

The first step always is to prepare a road network the simulation should take place in. While in general it is not relevant where the network data is coming from, we only provide ready to go import functions for OpenStreetMap (OSM) data which is freely available and comprehensive. Also if data in the target area is not up to date or needs to be adapted to the specific needs this can be done without problems.

In case there are ready to go scenarios for supported traffic simulators like SUMO we also try to support that. Nevertheless, we cannot guarantee to support all tool specific functions, so we really recommend using OpenStreetMap data as source.

If you already have data from other sources or want to use another data provider, additional steps might be necessary. If you have any questions or need further support, please feel free to contact our support team via our [mailing list](#).

#### 8.1.1 Using OpenStreetMap data

##### Preparing the data source

For this approach, simply download the \*.osm file of the desired region using the [openstreetmap.org](https://openstreetmap.org) website or any other tool that supports \*.osm file export (e.g. [josm](https://josm.openstreetmap.de/)<sup>1</sup> or [merkaartor](http://merkaartor.be/)<sup>2</sup>) and save it. Be aware that we currently do not support the binary version \*.pbp or any compressed versions.

Make sure that the file only contains complete ways. This means the file should not contain ways which reference nodes that are outside the target areas bounds. Usually this only has to be checked if the downloaded data was manually cropped with tools like [osmosis](https://wiki.openstreetmap.org/wiki/Osmosis)<sup>3</sup>. Such tools usually also have an option that will remove such ways (for osmosis this is `completeWays=yes`).

It is best practice to not let the source file contain roads clearly not usable (like bicycle or pedestrian ways). This is to not clutter the simulation with unnecessary data resulting in a serious slow down for some aspects of the simulation like routing. There are several ways to filter OpenStreetMap files, the most convenient way is included in the tool we provide to process the source file, we will get back to that

---

<sup>1</sup><https://josm.openstreetmap.de/>

<sup>2</sup><http://merkaartor.be/>

<sup>3</sup><http://wiki.openstreetmap.org/wiki/Osmosis>

in a minute. If you need more control over the filter process we recommend to use `osmosis` or one of the editors already mentioned.

## Processing the data

`VSimRTI` uses a database for internal storage of map and route data. Therefore it is necessary to convert OpenStreetMap data into the `VSimRTI` specific database format. While converting we filter out unnecessary information about the road network and prepare the data in a way, that allows us to provide routing functionality and create a common base for other components in the simulation like the traffic simulator.

To be able to do this we provide the tool `scenario-convert` which can be found in `vsimrti/bin/tools`. As already mentioned this tool also provides a basic filtering for OpenStreetMap input. The usual command to use in this case is:

---

```
java -jar scenario-convert-18.0.jar --osm2db -i path/to/source.osm -o
```

---

In this command `-o` provides the default filtering which filters out every street below type `tertiary`. Executing this will generate you a new file with file ending `.db` which will be named the same as your `*.osm` file. So for the given command it would generate a `source.db` in the directory `source.osm` is found. Additionally a second `*.osm` file with the suffix `_cleaned` will be created which is the filtered source data actually used in the import process. With `-d` you can provide a custom file name for the database alongside a path where it should be placed. If you provide a name please do so without the file extension, the correct one will be added automatically.

Now you should have a brand new database which contains the road network. In the next step you need to export the data in a format that can be understood by the traffic simulator to be used and still matches the information in the database. To do so export that data from the database like this:

---

```
java -jar scenario-convert-18.0.jar --db2sumo -d path/to/database.db -n
```

---

The example execution should have created a number of `SUMO` specific network files (nodes, edges and connections). The parameter `-n` automatically executes `SUMO`'s `netconvert` command after the export to create the needed netfile (requires `SUMO` to be in the systems path). If you want to have traffic lights generated by `netconvert`, you need to add the option `--export-traffic-lights` to the command above.

Now you have processed the road network to be in formats usable by `VSimRTI` and its connected simulators.

Regarding files, the output of these two steps should be the following:

- `<scenarioName>.db` - scenario database (including road network)
- `<scenarioName>.nod.xml` - SUMO node file
- `<scenarioName>.edg.xml` - SUMO edge file
- `<scenarioName>.con.xml` - SUMO connection file
- `<scenarioName>.net.xml` - SUMO network file (after executing `netconvert`)
- `<scenarioName>.sumo.cfg` - SUMO configuration file



## 8.2 Vehicles and Routes

VSimRTI contains a navigation component which supports the usage of different routes, vehicles can be mapped to. In order to make the routes accessible for the simulation in VSimRTI, they have to be available in the scenario database. There are two ways to achieve this:

- Generate routes with scenario-convert

or

- Import an existing SUMO route file

### Generate routes

scenario-convert offers possibilities to generate routes for the simulation. With the following command you can generate routes from one geo-point (latitude,longitude) to another:

---

```
java -jar scenario-convert-18.0.jar -d path/to/database.db -g --route-begin-latlon LAT,LON --route-end-latlon LAT,LON
```

---

If geo-points are given as input, the route generation might fail due to unmatchable points. In this case, you may use the following command in order to calculate routes from one street node to another:

---

```
java -jar scenario-convert-18.0.jar -d path/to/database.db -g --route-begin-node-id NODE_ID --route-end-node-id NODE_ID
```

---

The required node ids can be determined from the source OSM file or the generated SUMO network file. For more than one route between the given points/nodes, the parameter `-number-of-routes NUMBER` can be used. The calculated routes are directly written into the database. If you need to export them into a SUMO readable format, you can use the command:

---

```
java -jar scenario-convert-18.0.jar --db2sumo -d path/to/database.db -r <scenarioname>.rou.xml
```

---

### Import routes from SUMO route file

If you want to use the tools provided by SUMO, e.g. `duarouter`<sup>4</sup>, to calculate a set of routes for the simulation, you need to import the resulting routes into the scenario-database of VSimRTI. Note, if you use such tools, please always use the network file generated by scenario-convert. The route file (\*.rou.xml) generated by those tools can be directly imported into the database by using scenario-convert. For this purpose, you can use the following command:

---

```
java -jar scenario-convert-18.0.jar --sumo2db -r routefile.rou.xml -d path/to/database
```

---

This command will read the routes in the route file and write them in the supplied database file. The database file needs to be the same you used to export the network file. If the route file also contained vehicle definitions these will be converted into a mapping3 configuration file. You will need to adapt this file later on to achieve a meaningful mapping between vehicles and applications. For more information regarding this please refer to chapter 3.5.15.

---

<sup>4</sup><http://sumo.dlr.de/wiki/DUAROUTER>

## 8.3 VSimRTI

Now you have everything you need to create a configuration set that **VSimRTI** will recognize as a scenario. A scenario is generally a folder structure that reflects the different components usually found in a simulation. The example tutorials of Barnim and Tiergarten also follow this specific folder structure.

The complete set of components is depicted in the Listing 8.1.

```
<scenarioName>
├── applicationNT
├── battery
├── eventserver
├── cell2
├── mapping3
├── ns3
├── omnetpp
├── sources
├── sns
├── sumo
├── visualizer
└── vsimrti
```

Figure 8.1: Scenario: folder structure for complete component set

However, a subset of all components already allows reasonable simulations. The most important components are mentioned in the following part.

### Applications and Mapping (**applicationNT**, **mapping3**)

Usually you want the simulated vehicles to be equipped with some kind of applications that influence the vehicles behavior. To do that you copy the jar files of your applications to the folder `application/applicationNT`. To further configure the behavior of the application simulator have a look at chapter 3.5.

Having the applications in place you will have to copy your mapping file to the folder `mapping3`. If you don't have one by now have a look at the configuration for the default simulations provided with **VSimRTI**.

For further information on controlling the mapping see chapter 3.5.15.

### Navigation (**applicationNT**)

The generated database file `<scenarioName>.db` needs to be moved into the `applicationNT` folder. As long as you only have one database file in this folder you don't need to do anything more. **VSimRTI** will recognize the file automatically.

For further info on navigation in the application simulator have a look at chapter 3.5.14.

### Traffic Simulator (sumo)

The generated files for the used traffic simulator go into the folder named after that simulator e.g. `SUMO`. In the folder should be a configuration file that needs to be adapted, to point to the correct simulator specific network file, e.g. `<scenarioName>.sumo.cfg`.


For further info on this type of simulators have a look at chapters 3.4.

### Communication Simulator (cell2, ns3, omnetpp, sns)

There is an extensive default configuration provided in the simulators folder in the tutorials of Barnim and Tiergarten for all communication simulators.

Depending on the simulator you will need to tell the simulator the geographic extend of the simulation area. You can find that data in the traffic simulators network file, e.g. `SUMOs *.net.xml` contains this information in the `convBoundary` attribute of the `location` tag.

- For `OMNeT++`, it concerns the values of `constraintArea` in the `omnetpp.ini`.
- For the `CELL` simulator, the expansions do not need to be configured directly. However, the areas of the configured regions (in `regions.json`) have to be in line with the scenario location.
- The `SNS` also comes without an additional expansion definition.

 **Important:** Larger values for the spatial expansion of the simulation area are also usually possible for the communication simulators. However, much larger this might decrease the performance. **Smaller values lead to wrong behavior and will crash the simulation!**

For further information on the communication simulators, have a look at Chapters 3.2, 3.3, 3.6, and 3.7.

### VSimRTI config

Having done that you are now ready to adapt the `vsimrti/vsimrti_config.xml` file to your needs. This usually contains of setting the scenario id (recommended to use the scenarios folder name), adapting the end time to cover everything you want to analyze.

An important setting is the transformation configuration, as this will control how geographic coordinates will be converted to cartesians and back. Having a correct setting here is crucial to get correct results that map to real world coordinates so the simulation results can be visualized in some way. The center coordinate will be used to determine the correct `UTMZone`, the `cartesianOffset` can be determined by having a look at the traffic simulators network file, e.g. `SUMOs *.net.xml` contains this information in the `netOffset` attribute of the `location` tag.

Within the `IPResolverConfig` tag the address resolution scheme is specified. The subnets for all node types are described in JSON format.

Last but not least make sure, that the correct simulators are activated and unused deactivated in the last section of the configuration.

To simulate a generated scenario, copy the now ready scenario into the `vsimrti/scenarios` folder.

### **Further components**

Several components (as the battery configuration or the visualizer configuration) are generally scenario independent and can be copied from the existing tutorial scenarios Barnim and Tiergarten.

The event server component (to simulate an icy road etc.) obviously also depends on the scenario location. The position of the event should be in line with and be located in the area of the scenario.

# 9 Visualizers

## 9.1 Kinds of Visualizers

VSimRTI provides several ways of data evaluation by so called visualizers. To visualize a scenario, different visualizers can connect to a running simulation. Figure 9.1 gives an overview about the current coupled visualizers. More than one visualizer of the same type can be defined.

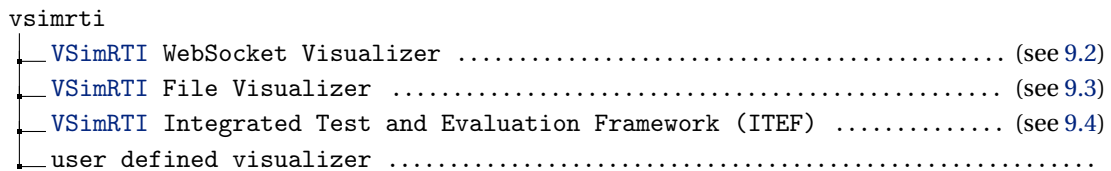


Figure 9.1: VSimRTI visualizer structure

The visualization ambassador provides several ways to visualize and/or trace the activities that happen during the simulation. The here explained visualization federate is responsible for providing data (in form of subscribed messages) to the visualization component. The kind of visualization is completely dependent on the component itself.

The VSimRTI File Visualizer aims to collect and format data for further processing after simulation. To get an instant impression of a simulation, the **WebSocket Visualizer** shows vehicle movements on a Google Map.

The visualizers are integrated into the simulation with a visualization ambassador.

## 9.2 VSimRTI WebSocket Visualizer

To get a simple and instant impression of a simulation or to get an idea of how fast it runs or where a simulation is located, the WebSocket Visualizer was created.

It shows a Google Map with markers, indicating the positions of all vehicles, updated as the simulation progresses.

To start the visualization, simply open the visualizer.html in your browser. The WebSocket Visualizer is enabled by default.

As soon, as the simulation is running, you can see the vehicle markers moving around.

You can also start the visualization at each simulation run, using the command line parameter `-v`.

**⚠ Important:** By default, the WebSocket Visualizer does not work on Microsoft Edge. UWP (Universal Windows Platform) apps on Windows 10 do not have direct network access but are subject to a network isolation for security reasons, preventing localhost loopback by default. While Microsoft Edge itself does allow localhost access, it treats localhost as an Internet site, which leads to restrictions e.g. for IPC over IP. To prevent this, an exception for Edge must be added to the network isolation via the following command in an elevated command prompt:

---

```
CheckNetIsolation LoopbackExempt -a -n="Microsoft.MicrosoftEdge_8wekyb3d8bbwe"
```

---

## 9.3 VSimRTI File Visualizer

The File Visualizer is a tool which gives you the opportunity to log specific VSimRTI message types.

### 9.3.1 Configuring the File Visualizer

- The main configuration file is located at  
vsimrti/scenarios/<scenarioName>/visualizer/visualizer\_config.xml

#### message record

- Each message record is derived from a certain message type and composed of several entries, which are separated by Element *separator*. A more detailed overview about the visualizable message types in VSimRTI is given in the next chapter Results. The configuration of the file visualizer is explained at the example of the VehicleMovements message.
- Attribute *id* indicates the message type, namely the class name of the message.
- Message has also an attribute *enabled*.
- The element *entries* defines the format and content of the finally visualized message record.
- The element *entries* is composed of several sub-elements *entry*, which correspond to columns of a message record and the sequence of the columns is the same as that of sub-elements entry.

#### entry

Basically, there are totally three types of entry: Just look at an example:

---

```
<?xml version="1.0" encoding="UTF-8" ?>
<message id="VehicleMovements">
  <entries>
    <entry>"MOVE_VEHICLE"</entry>
    <entry>TimeInSec</entry>
    <entry>Updated:Id</entry>
    <entry>Updated:ApplicationNr</entry>
    <entry>Updated:Position.Latitude</entry>
    <entry>Updated:Position.Longitude</entry>
  </entries>
</message>
```

---

Listing 9.1: Specific Configuration for message.

#### Constant

Every quoted entry is defined as a constant. The content inside the quotation will be directly visualized into each corresponding message record.

---

```
<entry>"MOVE_VEHICLE"</entry>
```

---

Listing 9.2: An example for constant type entry.

## Basic Method

The other two types of entry originate from the *getXXX()* methods of a certain object. For an entry, the root object for method invoking is the corresponding message class, here *VehicleMovements*. If a null object is returned before the last method of cascaded methods is invoked, then "null" will be written to the corresponding field. If a null object is returned by iteration method, then all fields involving this iteration method will be set "null".

## Iteration

---

```
<entry>Updated:Id</entry>
```

---

Listing 9.3: An example for method type entry with iteration.

The first part of this example is *Updated*, which means to invoke the *getUpdated* method of class *VehicleMovements*. Then a list of *VehicleInfo* objects is returned. Then *;Id* remains. The semicolon is an operator for iteration, which means for each of the *VehicleInfo* objects in the returned list invoking *getId* method.

## Cascading

---

```
<entry>Updated:Position.Latitude</entry>
```

---

Listing 9.4: An example for method type entry with iteration and cascading.

In this example, there is a dot operation. It is a cascade operation. Here *getPosition* method of *VehicleInfo* class is called and a *GlobalPosition* object is returned. Then we continuously invoke the *getLatitude* method of this *GlobalPosition* object.

## Extended Method

All the methods involved above are the basic methods. There are also some functionality, which we can't extract from these existing methods. So an extended method set is offered to meet these requirements and also as an extension point in the future.

### simple extended method

---

```
<entry>TimeInSec</entry>
```

---

Listing 9.5: An example for simple extended method type entry.

With existing methods of *VehicleMovements* and its super class *Message*, we can't get the timestamp of a message in second. (only *Message.getTime*, which returns a time in ns, is available). Here, *getTimeInSec* is a method extension for *Message* class. The extended method set will be listed later.

### assisting extended method

---

```
<entry>Updated:Type</entry>
```

---

Listing 9.6: An example for assisting extended method type entry.



There are also methods both in the simple and extended method set. Taking *getType* method of *VehicleInfo* as an example. If the type of a *VehicleInfo* object has not been set, this method will return null. But we can take advantage of *AddedVehiclesMapping* messages to get its type. Thus, an extended *getType* will be invoked if *VehicleInfo.getType* returns null.

## Iteration

Basically, the method type of entry definition supports cascaded iteration as follows:

```
<entry>List1:List2:Id</entry>
```

Listing 9.7: An example for cascaded iteration.

What we haven't met yet, is that, if in the definition of entries, several different iterating operations exists, for example:

```
<entry>Senders:Id</entry>
<entry>Receivers:Id</entry>
```

Listing 9.8: An example for multi-level iteration.

*getSenders()* and *getReceivers()* are two different iterations. In this case, a product of Ids in both list will be generated. The result may look like this:

```
sender1 , receiver1
sender1 , receiver2
sender2 , receiver1
sender2 , receiver2
```

Listing 9.9: Visualising result of the above listing.

Note: the longest matched prefix will be considered as the same iterating operation, which means, they are in the same level of iteration structure.

## Method Sets

Method	Type	Info
Message.getTimeInSec()	long	
Message.getTimeInMS()	long	
ReceiveV2Xmessage.getType()	String	
SendV2Xmessage.getType()	String	
VehicleInfo.getApplicationNr()	int	
VehicleInfo.getType()	UnitType	@Deprecated
VehicleInfo.getTypeInOrdinal()	int	@Deprecated

Table 9.1: A list of all extended methods

---

<b>Method</b>	<b>Type</b>
<code>VehicleMovements.getUpdated()</code>	<code>List&lt;VehicleInfo&gt;</code>
<code>VehicleInfo.getId()</code>	<code>String</code>
<code>VehicleInfo.getPosition()</code>	<code>GlobalPosition</code>
<code>GlobalPosition.getLatitude()</code>	<code>double</code>
<code>Message.getTime()</code>	<code>long</code>
<code>VehicleInfo.getType()</code>	<code>UnitType</code>

---

Table 9.2: A list of some basic methods

## 9.4 VSimRTI Integrated Test and Evaluation Framework (ITEF)

**Notice:** ITEF is only available with a commercial license of VSimRTI. For further information on licences, please refer to our [mailing list](#).

The Integrated Test and Evaluation Framework (ITEF) is a webtool for planning and evaluating vehicular communication scenarios. It is suited for field operational tests as well as simulations.

ITEF also offers a variety of visualization features, making a comparison of different vehicles or between equipped and unequipped vehicles easy. It is structured into 4 screens, whereas the following 3 screens are intended for the visualization.

The Replay screen (see Figure 9.2) is intended to be used for an initial overview of the test run. The main feature is the display of the vehicle movements on a map, while the player can be used to playback the movement situation. In this manner, the ITEF allows a location and time dependent evaluation of simulation test runs.

The Evaluate screen (see Figure 9.3) allows the detailed investigation of the correlations in a test run. The main feature of this screen is to display the behavior summarized over the whole run. The structure of this screen with is similar to the Replay screen. However, the focus here is on the detailed (statistical) summary of evaluated metrics.

Finally, the Statistics screen (see Figure 9.4) provides statistical evaluations of the test and simulation run. Currently, statistics on Vehicle Speed, Travel Time, Travel Distance and several more are supported.

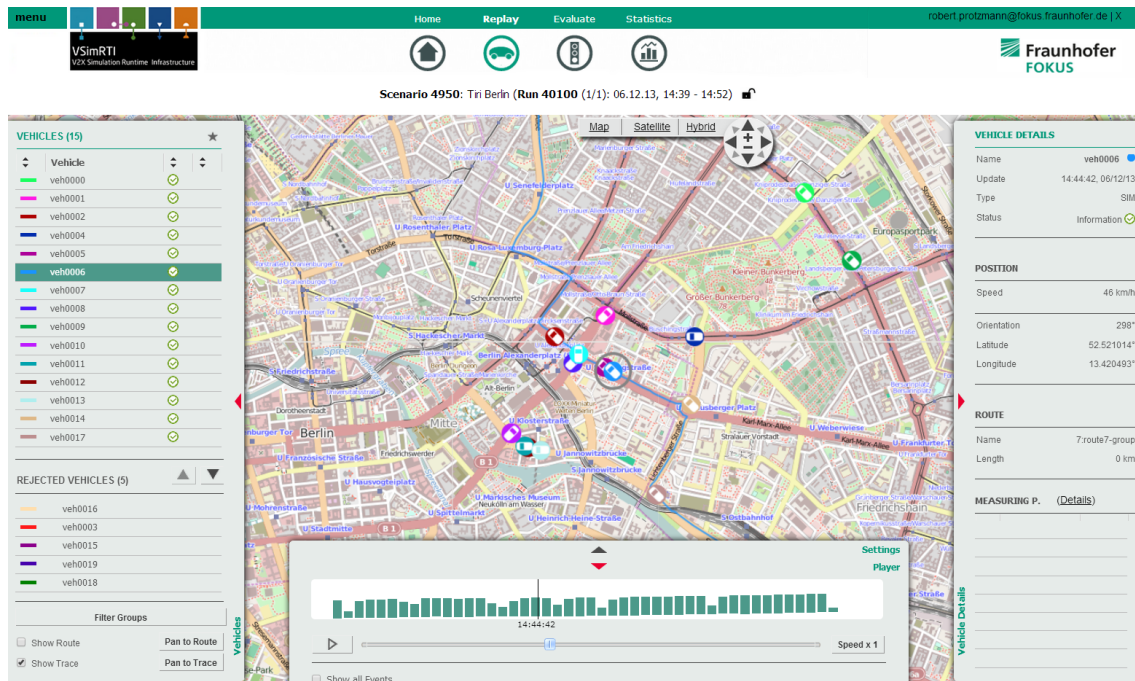


Figure 9.2: ITEF Replay Screen

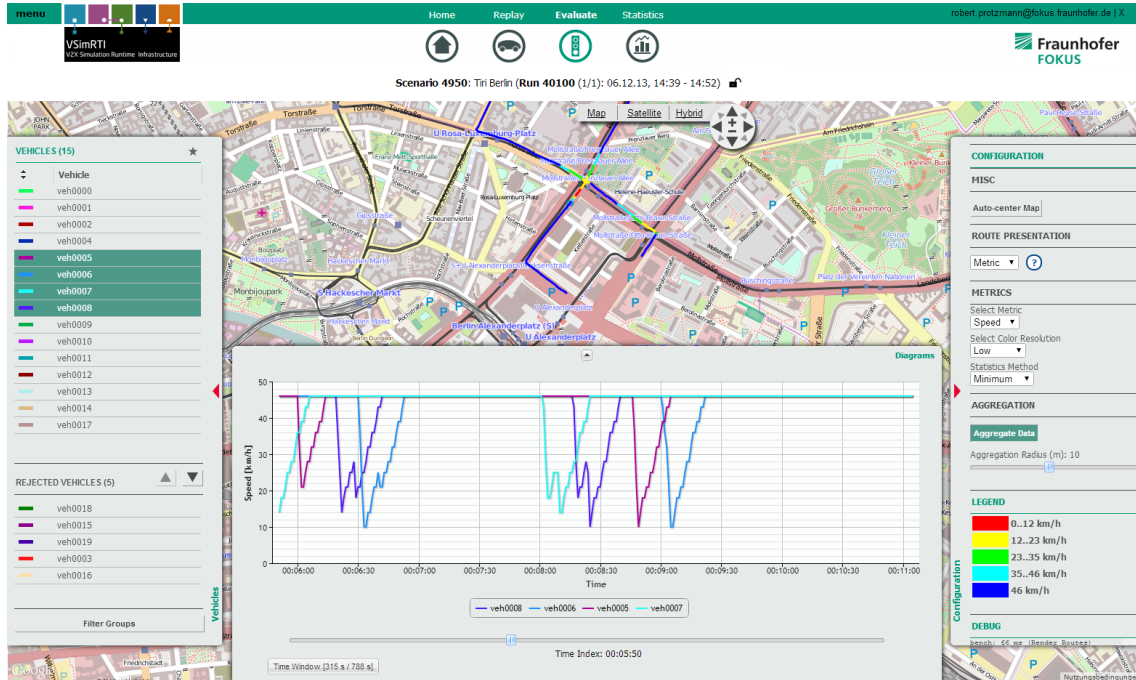


Figure 9.3: ITEF Evaluate Screen



Figure 9.4: ITEF Statistics Screen

# 10 Run simulation series

## 10.1 Simulation Runner

**i Notice:** The simulation runner is only available with a commercial license of VSimRTI.

For further information on licences, please refer to our [mailing list](#).

The simulation runner is a tool for automatic simulation parametrization and execution. Accordingly, a comfortable way exists to configure the execution of multiple simulations, e.g. of simulation series including several runs where only few parameters are changed in each run. With the simulation runner, a simulation series can be defined, for example, where the [V2X](#) penetration rate is changed in each simulation run. As a result, VSimRTI starts all simulation runs automatically according to the defined parameters.

### 10.1.1 Usage

The simulation runner is started as follows:

#### GNU/Linux

---

```
./simrun.sh -c /scenarios/<scenario name>/vsimrti/simrun_config.xml -p ↵  
↳ numberofparallelsimulations
```

---

Listing 10.1: VSimRTI simulation runner start command for GNU/Linux.

#### GNU/Linux

---

```
java -jar VsimrtiEmbeddedStarter-x.x.x.jar -s -u userid --strict-configuration-path -c ↵  
↳ scenarios/<scenario name>/vsimrti/simrun_config.xml
```

---

Listing 10.2: VSimRTI VSimRTIEmbeddedStarter simulation runner start command for GNU/Linux.

#### Microsoft Windows

---

```
simrun.bat -c \scenarios\<scenario name>\vsimrti\simrun_config.xml -p ↵  
↳ numberofparallelsimulations
```

---

Listing 10.3: VSimRTI simulation runner start command for Microsoft Windows.

#### Microsoft Windows

---

```
java -jar VsimrtiEmbeddedStarter-x.x.x.jar -u userid -s --strict-configuration-path -c ↵  
↳ scenarios\<scenario name>\vsimrti\simrun_config.xml
```

---

Listing 10.4: VSimRTI VSimRTIEmbeddedStarter simulation runner start command for Microsoft Windows.

## Configuration file

The configuration file `vsimrti/scenarios/<scenarioName>/vsimrti/simrun_config.xml` contains the parameterization information.

### 10.1.2 Configuration

The example in listing 10.5 shows a complete configuration. Using this configuration the simulation-runner would try to run a scenario called Barnim while adapting the mapping (see 3.5.15), the configuration file of SNS, and VSimRTI (see 8.3) configuration files. The actual simulation is triggered by generating an adapted scenario folder and calling the same executable the user would normally trigger himself.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <configuration
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:noNamespaceSchemaLocation="http://www.dcaiti.tu-berlin.de/research/simulation/download/
   ↳ get/scenarios/scenarioname/vsimrti/simrun_config.xsd">
5   <!-- basic configuration -->
6   <vsimrti location="/path/to/vsimrti_folder" executable="vsimrti.sh" username="user_id" />
7   <scenario name="Barnim" config="scenarios/Barnim/vsimrti/vsimrti_config.xml" persistent="
   ↳ false" repetitions="1">
8     <!-- argument -o TRACE -->
9     <!-- argument -w 0 -->
10  </scenario>
11
12  <!-- define connected values for controlled changes -->
13  <dimension name="PenetrationRates">
14    <parameter name="V2XVehiclePercentage" file="mapping3/mapping_config.json" fileFormat="
   ↳ json" item="vehicles[0].types[0].weight" type="ValueList">
15      <value>0.0</value>
16      <value>0.5</value>
17      <value>0.75</value>
18    </parameter>
19    <parameter name="ClassicVehiclePercentage" file="mapping3/mapping_config.json"
   ↳ fileFormat="json" item="vehicles[0].types[1].weight" type="ValueList">
20      <value>1</value>
21      <value>0.5</value>
22      <value>0.25</value>
23    </parameter>
24    <parameter name="Simulationtime" file="vsimrti/vsimrti_config.xml" fileFormat="xml" item=
   ↳="//simulation/endtime" type="ValueList">
25      <value>100</value>
26      <value>100</value>
27      <value>100</value>
28    </parameter>
29  </dimension>
30
31  <!-- define values for automatically permuted simulations -->
32  <parameter name="SinglehopRadius" file="sns/sns_config.json" fileFormat="json" item="
   ↳ singlehop.radius" type="ValueList">
33    <value>500</value>
34    <value>600</value>
35  </parameter>
36 </configuration>

```

Listing 10.5: Simulation Runner configuration example

The configuration contains three distinct parts of configuration. The system specific definition, the scenario definition and the parametrization. While the first two parts will be explained as part of this section, the parametrization will be explained in it's own section.

## system specific definition

---

```

1 <vsimrti location="/path/to/vsimrti_folder" executable="vsimrti.sh"
2   username="user_id" parallelSimulations="1" />

```

---

Listing 10.6: example of system specific definition in configuration file

The system specific part of the configuration (see listing 10.6) is the only part of the configuration that can be overwritten using a second configuration file. It contains the following values:

- `location` of the [VSimRTI](#) installation to use (can be relative or absolute).
- `executable` used to start the simulation. This value is optional and will automatically be set to the default `*.bat` or `*.sh` file when omitted.
- `username` needed to check the license against. This is the user name given to you when registering your system with us.
- `parallelSimulations` defines how many simulations are started in parallel to speed up things. This value is optional and defaults to 1. Be aware that you should only use this if you have multiple cores available. This might also coincide with the `threads` option in the [VSimRTI](#) configuration.

## scenario definition

---

```

1 <scenario name="Barnim" config="scenarios/Barnim/vsimrti/vsimrti_config.xml" persistent="false"
   ↳ " repetitions="1">
2   <!--argument-->-o TRACE</argument-->
3   <!--argument-->-w 0</argument-->
4 </scenario>

```

---

Listing 10.7: example of scenario definition in configuration file

The scenario definition (see listing 10.7) contains everything needed to identify the scenario to execute along with parameters that needs to be passed to the [VSimRTI](#) executable. It contains the following values:

- `name` of the simulation to run. This is expected to be the same as the scenarios folder name and is used to automatically generate the path pointing to the scenarios `vsimrti_config.xml` in a default case. It can be omitted if the `config` option is set.
- `config` is an optional value containing the concrete path to the scenarios `vsimrti_config.xml`. This can be used if the scenario is not placed in the default scenarios folder (which is discouraged) and overwrites the path generated by the `name` attribute. Either `name` or `config` have to be defined!
- `persistent` defines if the generated scenario folders (containing the adapted configuration files for every single simulation to be executed) will be kept after simulation (value `true`) or deleted (value `false`).
- `repetitions` is an optional value containing the number of times each simulation should be repeated. Setting this to a value other than 1 only makes sense if some part of the simulation follows stochastic models (like non deterministic vehicle spawning).

The configuration can also contain a number of additional arguments that are passed to the executable without any changes, separated by spaces.

### 10.1.3 Parametrization

The heart of this tool is the parametrization of simulations. Using this, one can define values in the default configuration that are adapted between simulation runs. How many simulation runs are performed is defined by the number of changes configured enriched with the information about simulation repetitions.

For the example in listing 10.5 it is expected that the mapping file to be changed has one `vehicles` definition spawning multiple cars with a weighted `type` distribution defining first the equipped and then the unequipped vehicles.

#### Parameters

---

```

1 <parameter name="V2XVehiclePercentage" file="mapping3/mapping_config.json"
2   fileFormat="json" item="vehicles[0].types[0].weight" type="ValueList">
3   <value>0</value>
4   <value>50</value>
5   <value>75</value>
6 </parameter>
```

---

Listing 10.8: example of a parametrization for a single value

Each value that should be changed in a run is defined by a `parameter` element identified by a name (see listing 10.8). The base value is the `file` which should be changed (relative to the scenario folder). Currently it is needed to define what `fileFormat` is expected from that file, which has impact on the syntax of the `item` definition which denotes what part of this file should be changed (this will be explained in a bit). The final value is the `type` which denotes how the value change behaves. The child elements depend on this definition and will also be explained in a bit.

`fileFormat` can be one of `xml` or `json`. The `item` syntax is as followed:

- `xml`: contains an XPath<sup>1</sup> expression
- `json`: contains an array-style definition of the target value. The value in listing 10.8 would change line 13 in listing 10.9. (In the first entry of `vehicles` the attribute `weight` of the `types` first entry).

---

```

1 {
2   "prototypes": [{"name": "PKW"}],
3   "vehicles": [
4     {
5       "startingTime": 5.0,
6       "targetDensity": 1200,
7       "maxNumberVehicles": 250,
8       "route": "1",
9       "types": [
10        {
11          "applications": ["com.dcaiti.vsimrti.app.WeatherWarningApp.WeatherWarningApp"],
12          "name": "PKW",
13          "weight": 0.2
14        },
15        {
16          "name": "PKW",
17          "weight": 0.8
18        }
19      ]
20    }
21  ]
22 }
```

---

<sup>1</sup><https://en.wikipedia.org/wiki/XPath>



```

19     ]
20   }
21 ]
22 }

```

Listing 10.9: mapping file expected for example

`type` can currently only have two entries:

- `ValueList`: This expects a list of values as child elements of the parameter. Each value will be used for at least one permutation.
- `IntegerGenerator`: This automatically generates integer values to write as values. The generated numbers can be configured by adding these attributes to the parameter element:
  - `offset` denoting a minimal number where generation should start (this will be the first value), default is 0.
  - `step` denoting the number that will be added to the previous value to generate the new one, default is 1.

In contrast to `ValueList` this can create an infinite number of values.

### Permutation of parameters

When multiple such parameter elements are defined, a permutation for each specific value definition is generated. Lets say defined are parameters A and B and each parameter has values a and b. The resulting permutations would be:

```

1   A=a, B=a
2   A=a, B=b
3   A=b, B=a
4   A=b, B=b

```

### Dimensions

Sometimes it is wanted to group some value changes. This can be necessary when changed values need to sum up to a specific value or when specific (named) output files need to be defined. This can be done by enclosing the affected parameters into a dimension definition. Doing this the values of each parameter are connected by their index. For this to work the number of values for each parameter have to be the same. The example in listing 10.5 utilizes this function to make sure the vehicle percentages sum up to 100. The generated permutations for the dimension enclosed parameters are:

```

1   V2XVehiclePercentage=0, ClassicVehiclePercentage=100
2   V2XVehiclePercentage=50, ClassicVehiclePercentage=50
3   V2XVehiclePercentage=75, ClassicVehiclePercentage=25

```

When additionally parameters are defined which are not enclosed in the dimension tag or another dimension tag is defined, then the permutations will be extended even further. The full permutation for listing 10.5 is as follows:

---

```
1 PenetrationRates(V2XVehiclePercentage=0, ClassicVehiclePercentage=100), SinglehopRadius=500
2 PenetrationRates(V2XVehiclePercentage=0, ClassicVehiclePercentage=100), SinglehopRadius=600
3 PenetrationRates(V2XVehiclePercentage=50, ClassicVehiclePercentage=50), SinglehopRadius=500
4 PenetrationRates(V2XVehiclePercentage=50, ClassicVehiclePercentage=50), SinglehopRadius=600
5 PenetrationRates(V2XVehiclePercentage=75, ClassicVehiclePercentage=25), SinglehopRadius=500
6 PenetrationRates(V2XVehiclePercentage=75, ClassicVehiclePercentage=25), SinglehopRadius=600
```

---

### 10.1.4 Additional Information

These are some side effects to remember when working with this tool.

**Ports:** The Simulation Runner supports automatic assigning of free ports for federates. This means that all federates configured in the `defaults.xml` will get a free port configured by default. This enables multiple simulations to be run simultaneously as long as the federates are started by `VSimRTI`. If some federates are **not** started through `VSimRTI` but are already running, this will not work.

**Paths:** Relative paths of the files to be modified will be expanded with the deployment directory of the current simulation run to an absolute one.

**Adaptations:** All values will be modified in copies of the original scenario. The copies will be placed in the `simrunner` folder in the `VSimRTI` base directory and will be (if not deactivated by configuration) deleted upon completion of the simulation.

# 11 Additional tools

## 11.1 scenario-convert

scenario-convert is a tool used to work with a scenarios database by importing and exporting data from external sources like OpenStreetMap [SUMO](#) etc.

This database is the basis for all map-related tasks, which can be performed by [VSimRTI](#) (e.g. navigation, route calculation...).

Based on a [VSimRTI](#) database, scenario-convert can export the data to [SUMO](#) input formats, which then can be used in the simulation. To enable dynamic rerouting of vehicles, scenario-convert generates, exports and imports route data from and to [SUMO](#). This way, one can choose, whether to use generated routes (all possible routes between a point A and B), use existing routes and import them via scenario-convert or build own routes via the route generation tools supplied with the standard [SUMO](#) installation.

### Workflow

An example workflow to create a valid map database would be as follows:

- get OpenStreetMap file of the desired region
- import OpenStreetMap file and export to [SUMO](#) using scenario-convert
- create routes using existing tools or manually in sumo
- import created routes back into database using scenario-convert
- start simulation using the database, which now contains all relevant map data and route information

The example scenario-convert call for this workflow would be as follows in listing 1:

```
1 scenarioconvert --osm2sumo -i <osmFile> -d <dbFile> -n --export-traffic-lights --generate-  
  ↘ routes --route-begin-latlon 50.429997,8.6567009 --route-end-latlon 50.429024,8.6642044
```

Listing 11.1: Exemplary call of scenario-convert.jar.

(import OpenStreetMap file to database, generate routes by given coordinates, generate network with sumo-netconvert and export to [SUMO](#) configuration files).

## Usage

The following listing 2 provides an overview about the scenario-convert functions.

```

1  Usage: scenario-convert [OPERATION] -d <DATABASEFILE> [OPTIONS]*
2
3  Examples (options in [] are optional):
4  1. Import an osm file and write data into database
5     scenario-convert --osm2db -i <OSMFILE> [-o] [-d <DATABASE>]
6  2. Export db content to sumo-readable files
7     scenario-convert --db2sumo -d <DATABASE> [-s <SUMOPREFIX>] [-n]
8  3. Reimport a sumo routefile into a database
9     scenario-convert --sumo2db -d <DATABASE> -r <ROUTEFILE>
10 4. Combine steps 1 and 2
11    scenario-convert --osm2sumo -d <DATABASE> -i <OSMFILE> [-s <SUMOPREFIX>]
12 5. Export db content to Shapefile format
13    scenario-convert --db2shp -d <DATABASE>
14 6. Import an srtm file and write elevation data to nodes of an existing database
15    scenario-convert --srtm2db -i <SRTMFILE> -d <DATABASE>
16
17 -----
18
19 CONFIGURATION FILE:
20 -c, --config-file FILE           Optional, refers to a configuration file which
21                                  contains all parameters in JSON format. Single
22                                  options can be set by the following parameters.
23
24 OPERATIONS:
25 --osm2db -i [-o] [-d]           Imports an OSM file creating a new database.
26 --srtm2db -i [-d]              Imports an SRTM file creating a new database.
27 --db2sumo -d [-s] [-n] [-l]    Exports the network to SUMO node and edge files.
28 --sumo2db [-i] [-r -d]         Imports a SUMO Network/Routefile. Be aware that
29                                  you have to re-export an imported network!
30 --osm2sumo                     Combination of osm2db and db2sumo.
31 --db2shp -d                   Exports the network into Shapefile format.
32 --osm2shp -d                   Combination of osm2db and db2shp.
33 --srtm2db -i -d               Imports an SRTM file and writes elevation data
34                                  to nodes.
35
36 --update -d                   Updates the given database to the current scheme.
37
38
39 CONFIGURATION OPTIONS:
40 -i, --input-file FILE          Defines an input file to use in an import.
41                                  File type is dependend on the import operation.
42                                  (OSM File/Database file/SUMO net file)
43 -d, --database FILE           When importing: file to import to. Either omit
44                                  the file extension or set it to '.db'.
45                                  For export operations: file to load from.
46                                  For update operations: file to update.
47 -s, --sumo-prefix SUMOPREFIX Prefix for the generated sumo files
48                                  (uses database name when not defined).
49 --import-zone                 UTM zone of location for projecting coords in
50                                  default format (e.g. 32n).
51 --import-lon/--import-lat     center longitude/latitude of imported region
52                                  to project coordinates.
53                                  ATTENTION: Zone and coordinate parameters are
54                                  mutually exclusive alternatives!
55 -r, --route-file FILE         Sumo *.rou.xml file location
56 --export-routes                Export existing route data to *.rou.xml
57                                  (only in combination with db2sumo).
58 -f, --force                   Force overwrite of existing files
59                                  (do not increment file names).
60
61
62 INTERNAL ROUTE GENERATION
63 (If you use this, a route file containing these routes will be generated too)

```

```

64 -g, --generate-routes      Generate routes based on the given options.
65                             Please see below for possible options,
66                             You will at least need start and end coordinates
67 Start of route:
68 --route-begin-lat         Latitude of point, needs longitude defined!
69 --route-begin-lon         Longitude of point, needs latitude defined!
70 --route-begin-latlon      Combined value in format [latitude],[longitude]
71                             (this is an alternative to separate definition!)
72 --route-begin-node-id     OSM node id as start point
73                             (this is an alternative to the options above!)
74 End of route:
75 --route-end-lat           Latitude of point, needs longitude defined!
76 --route-end-lon           Longitude of point, needs latitude defined!
77 --route-end-latlon        Combined value in format [latitude],[longitude]
78                             (this is an alternative to separate definition!)
79 --route-end-node-id       OSM node id as start point
80                             (this is an alternative to the options above!)
81
82 --number-of-routes INT    Optional, limits the maximum number of generated
83                             routes. Default is 1.
84
85 ADDITIONAL TOOLS
86 -o, --execute-osmosis     Execute osmosis to apply automatic filters to
87                             the given osm file (only for osm2xxx).
88 -b, --import-buildings    Write osm building information in database
89 --ignore-turn-restrictions Ignore all defined turn restrictions on OSM import.
90 --disable-graph-cleanup   Turns off the removal of unconnected parts from the
91                             main traffic network graph. Since several components of
92                             VSimRTI require one main graph without disconnected ways
93                             and nodes, this option should be used only if the
94                             cleanup procedure is faulty.
95 -n, --execute-netconvert  Automatically start sumo netconvert after export
96                             to create netfile (only with xxx2sumo)
97 -l, --export-traffic-lights export traffic light information for nodes
98 --osm-speeds-file          Define a property file which contains speed information
99                             which are used to set the speed for OSM ways without a
100                             max speed tag. (only with osm2xxx)
101 --osm-speeds-overwrite     If set to true, the maxspeed tags of ways are ignored and
102                             replaced by either default values, or by speed information
103                             defined via the --osm-speeds-file option. (only with osm2xxx)
104 -l, --export-traffic-lights export traffic light information for nodes
105 -v, --version               Print current version number

```

Listing 11.2: Overview about scenario-convert options.

The following listing 3 shows a JSON example file which can be used with the `-c` option of `scenario-convert`.

```

1 {
2   "operating_mode":      "osm2db",
3
4   "input_path":          "path/to/input.osm",
5   "database_path":       "path/to/database.db",
6
7   "sumo_prefix":         "SUMO_PREFIX",
8
9   "execute_netconvert":  true,
10  "export_traffic_lights": true,
11  "export_routes":        true,
12  "import_buildings":     false,
13  "force_overwrite":      false,
14  "disable_graph_cleanup": false,
15  "ignore_turn_restrictions": false,
16
17  "generate_routes":      true,
18  "number_of_routes":     2,

```

```

19     "route_begin_lat":      52.5,
20     "route_begin_lon":     13.4,
21     "route_end_lat":       52.2,
22     "route_end_lon":       13.1,
23
24     "osm_overwrite_speeds": false,
25     "osm_speeds_file":     "path/to/speeds.properties"
26 }

```

Listing 11.3: Example file for scenario-convert configuration with JSON

Furthermore, in listing 4 you can find a properties file which can be used during the import of OSM data in order to define speeds for ways, which do not have a maxspeeds-tag defined. For this purpose use the option `--osm-speeds-file <FILE>`. In the speed properties file, for each way type a speed value can be defined, according to the OSM highway key<sup>1</sup>.

```

1  # the unit the speed values are defined in [kmh, ms]
2  speed.unit = kmh
3
4  # the default speed for all way types which are not defined here
5  speed.default = 30
6
7  # autobahn
8  highway.motorway = 130
9  highway.motorway_link = 70
10
11 # bundesstrasse (germany)
12 highway.trunk = 70
13 highway.trunk_link = 65
14
15 # linking bigger town
16 highway.primary = 65
17 highway.primary_link = 60
18
19 # linking towns + villages
20 highway.secondary = 60
21 highway.secondary_link = 50
22
23 #streets without middle line separation
24 highway.tertiary = 50
25 highway.tertiary_link = 40
26 highway.residential = 30
27
28 #special roads
29 highway.living_street = 5
30 highway.service = 20
31
32 # unclassified roads
33 highway.unclassified = 30
34 highway.road = 20
35
36 # forest tracks
37 highway.track 15

```

Listing 11.4: Example file which can be used to configure speeds for ways during OSM import

<sup>1</sup><http://wiki.openstreetmap.org/wiki/Key:highway>

# 12 VSimRTI configuration

## 12.1 Overview

VSimRTI can be configured by adapting the files in the `/vsimrti/etc` directory. The following technical aspects of VSimRTI can be configured:

- `vsimrti/etc/defaults.xml` - Configuration of all federates which are coupled with VSimRTI.
- `vsimrti/etc/hosts.json` - Configuration of the machines the simulations are executed on.
- `vsimrti/etc/logback.xml` - Logging configuration of VSimRTI and its federates.

A full example of the default configuration files can be found in [A.2](#).

## 12.2 Federates configuration

This part of software can be configured with a configuration file. The specific path is `vsimrti/etc/defaults.xml`. This file is used for configuring all federates and simulators coupled with VSimRTI. For each federate you can define various aspects:

- What's the id and class of the federate?
- On which host is the federate running?
- How is the federate started?
- What is the name of the configuration file for the federate?
- For which messages is the federate subscribing?
- The priority of the federate when it comes to message distribution.

A detailed description of the various parameters can be found in the documentation available on the [DCAITI website](#).

### 12.2.1 Federate priorities

There might be cases where it is important that a certain federate receives messages with equal timestamps before or after the other ones. In the context of VSimRTI for example, it is often desirable that the Application Simulator receives the *vehicle movements*-message after the traffic simulator to avoid problematic behavior in the first simulator. The configuration file `defaults.xml` allows for a fine grained adjustment of the federate priorities.

For example, a federate with a certain priority looks like this: `<federate class="com.dcaiti.vsimrti.fed.sumo.ambassador.SumoAmbassador" priority="20">`. That means that the Sumo Ambassador has an priority of 20, where the following rules apply to the priorities:

- Priorities range from 0 to 100.
- 0 is the highest priority, e.g. the federate with this priority assigned will receive the message first.
- The default priority if none is given is 50.

Note that VSimRTI ships with sane default values for the priority and a modification is only necessary in rare cases.

## 12.3 Host configuration

This part of software can be configured with a configuration file. The specific path is `vsimrti/etc/hosts.json`

**Notice:** The documentation for the VSimRTI specific component is freely available on the [DCAITI website](#), explaining all available options.

This file is used for configuring simulation runs using multiple machines. Under normal circumstances it is not necessary to change this file.

In VSimRTI there are two kinds of hosts: local and remote hosts. All hosts have to be identified with a unique string that is valid for the remainder of the configuration to reference a defined host.

For a local host additionally the operating system and a name of a directory that is used to deploy needed federates is necessary. Values for operating systems are *"linux"* or *"windows"*.

The syntax for referenced paths has to be chosen according to operating system standards (e.g. `/vsimrti/temp` for GNU/Linux and `C:\vsimrti\temp` for Microsoft Windows). Note that also relative paths are possible.

For remote hosts SSH and SFTP is used to deploy and start federates. Therefore the address of the host as well as the port, user name, and password for an SSH connection must additionally be given.



# 13 Additional information

Here you find helpful information regarding VSimRTI and its configuration.

## 13.1 PATH configuration

PATH is an environment variable on operating systems, specifying a set of directories where executable programs are located.

### GNU/Linux

On GNU/Linux operating system you can edit you local `.bash_profile` in your home directory with your favorite CLI editor e.g. Vi IMproved ([VIM](#)) or Nano's ANOther editor ([NANO](#)).

---

```
1 # .bash_profile
2
3 # Get the aliases and functions
4 if [ -f ~/.bashrc ]; then
5     . ~/.bashrc
6 fi
7
8 # User specific environment and startup programs
9
10 PATH=$PATH:$HOME/bin:$HOME/sumo/
11
12 export PATH
```

---

Listing 13.1: Add the [SUMO](#) location to PATH on GNU/Linux.

On most GNU/Linux distributions you must log out and login to refresh the PATH variable. You can check your path with the shell command `echo $PATH`.

## Microsoft Windows

For Microsoft Windows please follow the instructions below.

1. Select “Computer” from the start menu. Choose “Properties” from the context menu.
2. Click “Advanced system settings” and switch to the “Advanced” tab in “System Properties”.
3. Click on “Environment Variables”.
4. Find “Path”, and click on it (choose it).
5. Click “Edit”.
6. Modify Path by adding the absolute location to the value for Path.  
(Don't forget a semicolon to separate).
7. Click “OK”, click “OK”, click “OK”, . . .

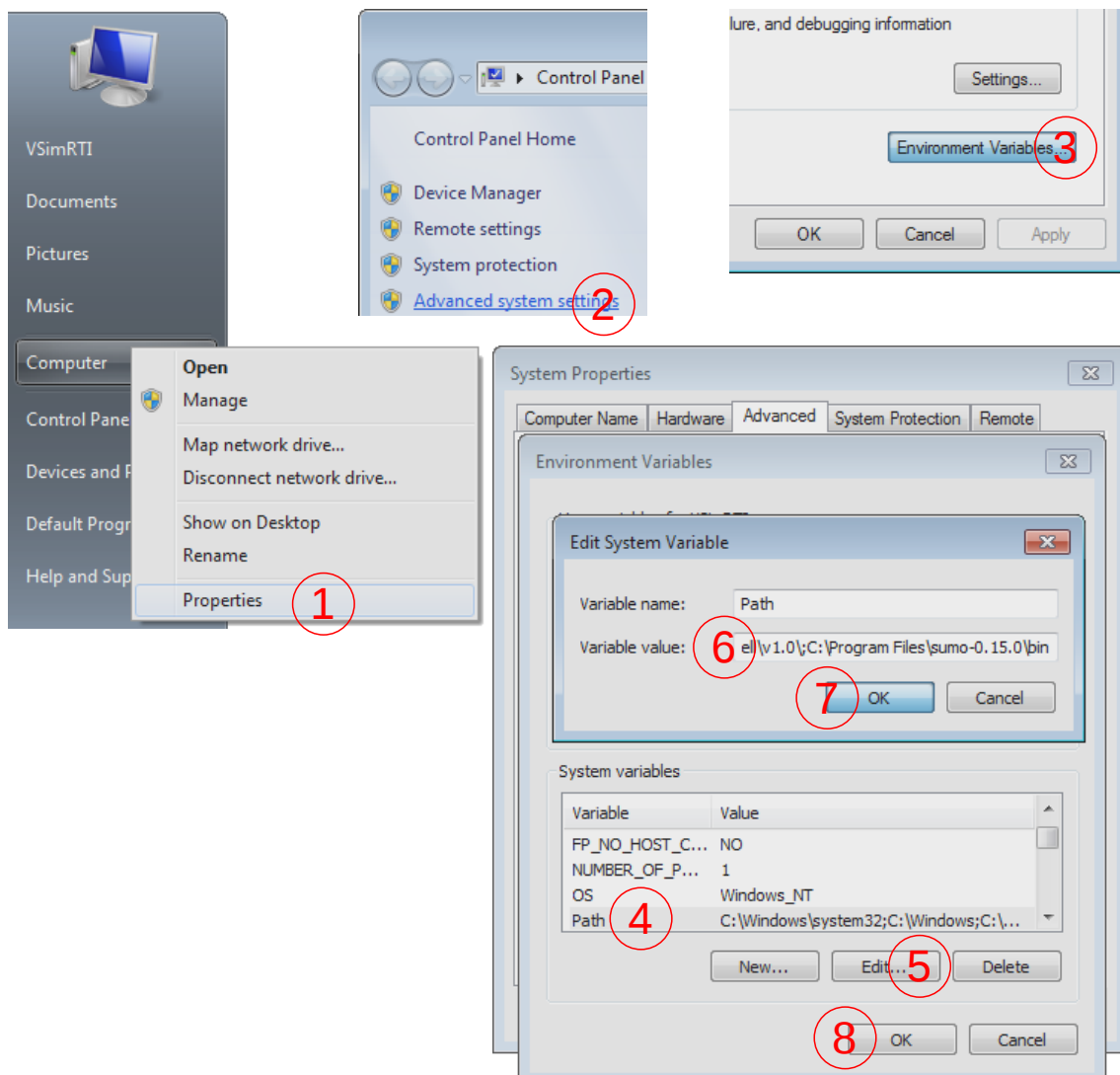


Figure 13.1: Edit PATH variable in Microsoft Windows

## 14 List of Acronyms

<b>ASCT</b>	Automotive Services and Communication Technologies .....	2
<b>CAM</b>	Cooperative Awareness Messages .....	10
<b>CLI</b>	Command Line Interface .....	4
<b>CPU</b>	central processing unit .....	2
<b>DCAITI</b>	Daimler Center for Automotive Information Technology Innovations .....	2
<b>DENM</b>	Decentralized Environmental Notification Messages .....	10
<b>ETC</b>	Estimated Time to Completion .....	6
<b>ETSI</b>	European Telecommunications Standards Institute .....	10
<b>EPL</b>	Eclipse Public License	
<b>FOKUS</b>	Fraunhofer-Institut für Offene Kommunikationssysteme .....	2
<b>GUI</b>	Graphical User Interface .....	23
<b>GNU</b>	GNU's Not Unix!	
<b>GPL</b>	GNU General Public License	
<b>HLA</b>	High-Level Architecture .....	1
<b>IEEE</b>	Institute of Electrical and Electronics Engineers .....	1
<b>INET</b>	OMNeT++ - Model library for wired, wireless and mobile networks .....	12
<b>ITEF</b>	Integrated Test and Evaluation Framework .....	78
<b>JAR</b>	Java Archive .....	1
<b>JRE</b>	Java Runtime Environment .....	1
<b>JSON</b>	JavaScript Object Notation .....	33
<b>LTE</b>	Long Term Evolution .....	10
<b>LuST</b>	Luxembourg SUMO Traffic .....	64
<b>M&amp;S</b>	modelling and simulation .....	1
<b>MAC</b>	Media Access Control .....	2
<b>MBMS</b>	Multimedia Broadcast Multicast Service .....	39
<b>NANO</b>	Nano's ANOther editor .....	92
<b>ns-2</b>	network simulator 2 .....	16
<b>ns-3</b>	network simulator 3 .....	9
<b>OMNeT++</b>	Objective Modular Network Testbed in C++ .....	9
<b>OSM</b>	Open Street Map	
<b>RAM</b>	random-access memory .....	2
<b>RSU</b>	Road Side Units .....	28
<b>RTF</b>	Real Time Factor .....	6
<b>SNS</b>	Simple Network Simulator .....	9
<b>SUMO</b>	Simulation of Urban Mobility .....	9
<b>TraCI</b>	Traffic Control Interface .....	32
<b>UMTS</b>	Universal Mobile Telecommunications System .....	10

<b>UUID</b>	universally unique identifier .....	32
<b>V2V</b>	Vehicle-to-Vehicle .....	10
<b>V2X</b>	Vehicle-2-X.....	2
<b>VIM</b>	Vi IMproved.....	92
<b>VSimRTI</b>	Vehicle-2-X Simulation Runtime Infrastructure .....	2
<b>XML</b>	Extensible Markup Language.....	33

# Appendix A

## VSimRTI deployment

You will find a collection of all relevant data.

### A.1 VSimRTI Folder Structure

The VSimRTI all-in-one package has a clear separation of binary files, general VSimRTI configuration files and simulation scenario specific configuration files in different folders. The `vsimrti/bin` folder contains all binary files needed for execution of VSimRTI. Normally, you do not have to make changes here.

General VSimRTI configuration files, which are used through all scenarios can be found in the `vsimrti/etc` folder.

These files contain information about how simulators interact, which remote hosts exists etc.. If you only want to perform simulations using the preconfigured simulators in the all-in-one package you do not need to make changes here. In the `vsimrti/scenarios` folder you can find the scenario specific configuration files, separated by component, e.g. traffic simulator or communication simulator.

This configuration can be changed in the file `vsimrti/etc/defaults.xml` (see A.1).

For normal usage of VSimRTI, this configuration does not need to be changed. Please edit this file only, if you know what you are doing, as unwanted side effects might occur.

The configuration of VSimRTI ambassadors is done by using configuration files in the folder `vsimrti/scenarios/<scenarioName>/<federateid>`.

To configure scenario specific VSimRTI options, e.g. to define which simulators should be active in a simulation scenario, you can adjust the `vsimrti/scenarios/<scenarioName>/vsimrti/vsimrti_config.xml` file located in `vsimrti/scenarios/<scenarioName>/vsimrti`. The overall folder structure is as follows:

```

vsimrti
├── bin ..... path containing binary files
├── etc ..... path containing general configuration files
│   ├── defaults.xml ..... (see A.1)
│   ├── hosts.json ..... (see A.2)
│   └── logback.xml ..... (see A.3)
├── lib ..... path containing system specific (non JAR) libraries
├── logs ..... path for all log files
├── scenarios ..... path for scenario specific configuration files
│   ├── Barnim ..... path containing Barnim example scenario
│   └── Tiergarten ..... path containing Tiergarten example scenario
├── tmp ..... path for temporary files during a simulation
├── simrun.sh ..... shell script to start a simulation series
├── simrun.bat ..... batch script to start a simulation series
├── systeminfo.txt ..... (see A.4)
├── vsimrti.sh ..... shell script to start a single simulation
├── vsimrti.bat ..... batch script to start a single simulation
├── vsimrti.dat ..... VSimRTI license file
└── vsimrti-license.lcs ..... VSimRTI license file

```

Figure A.1: VSimRTI folder structure

## A.2 File listings

### etc/defaults.xml

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- file version: 2013-04-13 -->
3 <configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:noNamespaceSchemaLocation="http://www.dcaiti.tu-berlin.de/research/
5     ↪ simulation/download/get/etc/defaults.xsd">
6     <federates>
7         <federate class="com.dcaiti.vsimrti.fed.omnetpp.ambassador.OmnetppAmbassador">
8             <id>omnetpp</id>
9             <deploy>true</deploy>
10            <start>true</start>
11            <host>local</host>
12            <port>4998</port>
13            <dockerImage></dockerImage>
14            <config>omnetpp_config.json</config>
15            <messages>
16                <message>AddedVehicle</message>
17                <message>AddedRsu</message>
18                <message>AddedTrafficLight</message>
19                <message>VehicleMovements</message>
20                <message>SendV2XMessage</message>
21                <message>ConfigureAdHocCommunication</message>
22            </messages>
23        </federate>
24        <federate class="com.dcaiti.vsimrti.fed.ns3.ambassador.Ns3Ambassador">
25            <id>ns3</id>
26            <deploy>true</deploy>
27            <start>true</start>
28            <host>local</host>
29            <port>5011</port>
30            <dockerImage></dockerImage>
31            <config>ns3_config.json</config>
32            <messages>
33                <message>AddedVehicle</message>
34                <message>AddedRsu</message>
35                <message>AddedTrafficLight</message>
36                <message>VehicleMovements</message>
37                <message>SendV2XMessage</message>
38                <message>ConfigureAdHocCommunication</message>
39            </messages>
40        </federate>
41        <federate class="com.dcaiti.vsimrti.fed.sns.ambassador.SnsAmbassador">
42            <id>sns</id>
43            <deploy>false</deploy>
44            <start>false</start>
45            <host>local</host>
46            <config>sns_config.json</config>
47            <messages>
48                <message>AddedVehicle</message>
49                <message>AddedRsu</message>
50                <message>AddedTrafficLight</message>
51                <message>VehicleMovements</message>
52                <message>SendV2XMessage</message>
53                <message>ConfigureAdHocCommunication</message>
54            </messages>
55        </federate>
56        <federate class="com.dcaiti.vsimrti.fed.sumo.ambassador.SumoAmbassador">
57            <id>sumo</id>
58            <deploy>true</deploy>
59            <start>true</start>
60            <host>local</host>
61            <port>0</port>
62            <config>sumo_config.json</config>

```

```

62         <messages>
63             <message>VehicleTypesInitMessage</message>
64             <message>VehiclePathsInitMessage</message>
65             <message>AddedVehicle</message>
66             <message>SetMaxSpeed</message>
67             <message>SlowDown</message>
68             <message>PropagateNewRoute</message>
69             <message>ChangeStaticRoute</message>
70             <message>ChangeStaticRouteByEdge</message>
71             <message>ChangeLane</message>
72             <message>ChangeTrafficLightsState</message>
73             <message>Stop</message>
74             <message>Resume</message>
75             <message>SumoTraciByteArrayMessage</message>
76             <message>EnableDistanceSensors</message>
77             <message>ChangeVehicleParameters</message>
78             <message>ChangeSpeed</message>
79             <message>VehicleControl</message>
80             <message>VehicleMovements</message>
81         </messages>
82     </federate>
83     <federate class="com.dcaiti.vsimrti.fed.applicationNT.ambassador."/ >
84         ↪ ApplicationNTAmbassador" >
85         <id>applicationNT</id>
86         <deploy>false</deploy>
87         <start>false</start>
88         <host>local</host>
89         <port>0</port>
90         <config>applicationNT_config.json</config>
91         <messages>
92             <message>AddedRsu</message>
93             <message>AddedChargingStation</message>
94             <message>AddedTrafficLight</message>
95             <message>AddedVehicle</message>
96             <message>ApplicationSpecificMessage</message>
97             <message>ChangeVehicleState</message>
98             <message>ChargingStationUpdate</message>
99             <message>ChargingRejected</message>
100            <message>VehicleElectricInformationMessage</message>
101            <message>PropagateNewRoute</message>
102            <message>ReceiveV2XMessage</message>
103            <message>ReceiveFullV2XMessage</message>
104            <message>AcknowledgeV2XMessage</message>
105            <message>SensorData</message>
106            <message>SumoTraciByteArrayMessageResponseContainer</message>
107            <message>UpdateTrafficLight</message>
108            <message>VehicleMovements</message>
109            <message>VehiclePathsInitMessage</message>
110            <message>VehicleTypesInitMessage</message>
111        </messages>
112    </federate>
113    <federate class="com.dcaiti.vsimrti.fed.eventserver.ambassador.EventserverAmbassador">
114        <id>eventserver</id>
115        <deploy>false</deploy>
116        <start>false</start>
117        <host>local</host>
118        <config>eventserver_config.json</config>
119        <messages>
120            <message>SensorRegistration</message>
121            <message>VehicleMovements</message>
122        </messages>
123    </federate>
124    <federate class="com.dcaiti.vsimrti.fed.mapping3.MappingAmbassador">
125        <id>mapping3</id>
126        <deploy>false</deploy>
127        <start>false</start>
128        <host>local</host>

```



```

128         <config>mapping_config.json</config>
129     <messages>
130         <message>VehiclePathsInitMessage</message>
131         <message>ScenarioTrafficLights</message>
132         <message>ScenarioAddedVehicle</message>
133     </messages>
134 </federate>
135 <federate class="com.dcaiti.vsimrti.fed.cell2.ambassador.Cell2Ambassador">
136     <id>cell2</id>
137     <deploy>>false</deploy>
138     <start>>false</start>
139     <host>local</host>
140     <config>cell2_config.json</config>
141     <messages>
142         <message>AddedVehicle</message>
143         <message>AddedRsu</message>
144         <message>AddedTrafficLight</message>
145         <message>AddedChargingStation</message>
146         <message>VehicleMovements</message>
147         <message>SendV2XMessage</message>
148         <message>ConfigureCellCommunication</message>
149     </messages>
150 </federate>
151 <federate class="com.dcaiti.vsimrti.fed.battery.ambassador.BatteryAmbassador">
152     <id>battery</id>
153     <deploy>>false</deploy>
154     <start>>false</start>
155     <host>local</host>
156     <config>battery_config.json</config>
157     <messages>
158         <message>AddedVehicle</message>
159         <message>VehicleMovements</message>
160         <message>ChargingStarted</message>
161         <message>ChargingStopped</message>
162     </messages>
163 </federate>
164 <federate class="com.dcaiti.vsimrti.fed.chargingstation.ambassador.↵
    ↵ ChargingStationAmbassador">
165     <id>chargingstation</id>
166     <deploy>>false</deploy>
167     <start>>false</start>
168     <host>local</host>
169     <config>chargingstation_config.json</config>
170     <messages>
171         <message>AddedChargingStation</message>
172         <message>ChargeRequest</message>
173         <message>StopChargeRequest</message>
174         <message>ChargingStationUpdate</message>
175     </messages>
176 </federate>
177 <federate class="com.dcaiti.vsimrti.fed.visual.VisualizationAmbassador">
178     <deploy>>false</deploy>
179     <start>>false</start>
180     <id>visualizer</id>
181     <host>local</host>
182     <port>5099</port>
183     <update>1</update>
184     <config>visualizer_config.xml</config>
185     <messages>
186         <message>VehicleTypesInitMessage</message>
187         <message>VehiclePathsInitMessage</message>
188         <message>AddedVehicle</message>
189         <message>AddedRsu</message>
190         <message>AddedTrafficLight</message>
191         <message>AddedChargingStation</message>
192         <message>VehicleMovements</message>
193         <message>SensorData</message>

```

```

194         <message>VehicleElectricInformationMessage</message>
195         <message>UpdateTrafficLight</message>
196         <message>ChargingStationUpdate</message>
197         <message>SetMaxSpeed</message>
198         <message>SlowDown</message>
199         <message>ChangeRoute</message>
200         <message>ChangeStaticRoute</message>
201         <message>ChangeLane</message>
202         <message>ChangeTrafficLightsState</message>
203         <message>ApplicationItefMessage</message>
204         <message>SendV2XMessage</message>
205         <message>ReceiveV2XMessage</message>
206         <message>DeleteV2XMessage</message>
207     </messages>
208 </federate>
209 <federate class="com.dcaiti.vsimrti.fed.phabmacs.ambassador.PhabmacsAmbassador">
210     <id>phabmacs</id>
211     <deploy>true</deploy>
212     <start>true</start>
213     <host>local</host>
214     <port>50423</port>
215     <config>phabmacs_config.json</config>
216     <javaMemorySizeXms>100</javaMemorySizeXms>
217     <javaMemorySizeXmx>4086</javaMemorySizeXmx>
218
219     <!-- Add custom classpath entries to avoid copying jar files manually -->
220     <!-- Base dir is "tools/VsimrtiEmbeddedStarter/tmp/phabmacs" -->
221
222     <!--
223     <javaClasspathEntries>
224         <classpath>../../../../federates/phabmacs/phabmacs-federate/target/classes</classpath>
225         ↪ classpath
226     </javaClasspathEntries>
227     -->
228
229     <!-- Add custom JavaArgument for debugging -->
230     <!-- <customJavaArgument>-Xdebug -Xrunjdw:transport=dt_socket|,server=y|,suspend=n|
231         ↪ |,address=8086</customJavaArgument>-->
232
233 <messages>
234     <message>VehicleTypesInitMessage</message>
235     <message>VehiclePathsInitMessage</message>
236     <message>AddedVehicle</message>
237     <message>AddedRsu</message>
238     <message>ChangeSpeed</message>
239     <message>SetAcceleration</message>
240     <message>SetMaxSpeed</message>
241     <message>SlowDown</message>
242     <message>ChangeStaticRoute</message>
243     <message>ChangeStaticRouteByEdge</message>
244     <message>ChangeLane</message>
245     <message>Stop</message>
246     <message>Resume</message>
247     <message>ChangeTrafficLightsState</message>
248     <message>PropagateNewRoute</message>
249     <message>SetActuators</message>
250     <message>SendV2XMessage</message>
251     <message>ReceiveV2XMessage</message>
252     <message>CurrentEvents</message>
253     <message>VehicleControl</message>
254     <message>VehicleMovements</message>
255 </messages>
256 </federate>
257 <federate class="com.dcaiti.vsimrti.fed.phasca.ambassador.PhascaAmbassador">
258     <id>phasca</id>
259     <deploy>true</deploy>
260     <start>false</start>
261     <host>local</host>

```

```
259         <config>phasca_config.json</config>
260     <messages>
261         <message>AddedVehicle</message>
262     </messages>
263 </federate>
264 </federates>
265 </configuration>
```

---

Listing A.1: VSimRTI: file listing: etc/defaults.xml

**etc/hosts.json**


---

```

1 {
2   "localHosts" : [
3     {
4       "id" : "local",
5       "workingDirectory" : "./tmp",
6       "address" : "localhost",
7       "operatingSystem" : "linux"
8     }
9   ],
10  "remoteHosts" : [
11    {
12      "id" : "remote",
13      "workingDirectory" : "/home/vsimrti/test/tmp",
14      "address" : "192.168.0.1",
15      "operatingSystem" : "linux",
16      "user" : "username",
17      "password" : "password",
18      "port" : 22
19    }
20  ]
21 }

```

---

Listing A.2: VSimRTI: file listing: etc/hosts.json

**etc/logback.xml**


---

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- file version: 2013-04-05 -->
3 <configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:noNamespaceSchemaLocation="http://www.dcaiti.tu-berlin.de/research/
5     ↘ simulation/download/get/etc/logback.xsd">
6     <!-- timestamp key="timestamp" datePattern="yyyy-MM-dd-HH:mm:ss" /-->
7     <!-- ##### APPENDER ##### -->
8     <!-- Console logger -->
9
10    <if condition='isDefined("logDirectory")'>
11      <then>
12        <property name="logDirectory" value="{logDirectory}"/>
13        <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
14          <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
15            <pattern>%date %-5level %logger{0} - %msg%n</pattern>
16          </encoder>
17        </appender>
18        <!-- Logger for Time Advances -->
19        <appender name="STDOUT-TimeAdvance" class="ch.qos.logback.core.ConsoleAppender">
20          <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
21            <!-- Difference to previous logger: no %n at line end... -->
22            <pattern>%date{HH:mm:ss} - %msg</pattern>
23          </encoder>
24        </appender>
25        <!-- Logger for DBloading progress -->
26        <appender name="STDOUT-DbLoading" class="ch.qos.logback.core.ConsoleAppender">
27          <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
28            <!-- Difference to previous logger: no %n at line end... -->
29            <pattern>%date{HH:mm:ss.SSS} - %msg</pattern>
30          </encoder>
31        </appender>
32        <!-- Logger for plain messages -->
33        <appender name="STDOUT-MessageOnly" class="ch.qos.logback.core.ConsoleAppender">
34          <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
35            <pattern>%msg%n</pattern>

```

---

```

36         </encoder>
37     </appender>
38     <!-- File Appender for different components -->
39     <appender name="VsimrtiLog" class="ch.qos.logback.core.FileAppender">
40         <file>${logDirectory}/VSimRTI.log</file>
41         <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
42             <pattern>%date %-5level %logger{0} - %msg%n</pattern>
43         </encoder>
44     </appender>
45     <appender name="TrafficLog" class="ch.qos.logback.core.FileAppender">
46         <file>${logDirectory}/Traffic.log</file>
47         <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
48             <pattern>%date %-5level %logger{0} - %msg%n</pattern>
49         </encoder>
50         <append>false</append>
51     </appender>
52     <appender name="CommunicationLog" class="ch.qos.logback.core.FileAppender">
53         <file>${logDirectory}/Communication.log</file>
54         <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
55             <pattern>%date %-5level %logger{0} - %msg%n</pattern>
56         </encoder>
57     </appender>
58     <appender name="CommunicationDetailsLog" class="ch.qos.logback.core.FileAppender">
59         <file>${logDirectory}/CommunicationDetails.log</file>
60         <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
61             <pattern>%msg%n</pattern>
62         </encoder>
63         <append>false</append>
64     </appender>
65     <appender name="ApplicationNTLog" class="ch.qos.logback.core.FileAppender">
66         <file>${logDirectory}/ApplicationNT.log</file>
67         <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
68             <pattern>%date %-5level %logger{0} - %msg%n</pattern>
69         </encoder>
70     </appender>
71
72     <appender name="ApplicationNtLogDelegation" class="ch.qos.logback.classic.sift.↳
↳ SiftingAppender">
73         <discriminator>
74             <key>path</key>
75             <defaultValue>unknown</defaultValue>
76         </discriminator>
77         <sift>
78             <appender name="FILE-${unitId}" class="ch.qos.logback.core.FileAppender">
79                 <file>${logDirectory}/appsNT/${path}.log</file>
80                 <layout class="ch.qos.logback.classic.PatternLayout">
81                     <pattern>%date %-5level - %msg%n</pattern>
82                 </layout>
83             </appender>
84         </sift>
85     </appender>
86
87     <appender name="GeneralPurposeAmbassador" class="ch.qos.logback.classic.sift.↳
↳ SiftingAppender">
88         <discriminator>
89             <key>loggerId</key>
90             <defaultValue>unknown</defaultValue>
91         </discriminator>
92         <sift>
93             <appender name="FILE-${loggerId}" class="ch.qos.logback.core.FileAppender"↳
↳ >
94                 <file>${logDirectory}/loggerId-${loggerId}.log</file>
95                 <layout class="ch.qos.logback.classic.PatternLayout">
96                     <pattern>%date %-5level %logger{0} - %msg%n</pattern>
97                 </layout>
98             </appender>
99         </sift>

```

```

100     </appender>
101
102     <appender name="StatisticsLog" class="ch.qos.logback.core.FileAppender">
103         <file>${logDirectory}/statistics.csv</file>
104         <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
105             <pattern>%date %-5level %logger{0} [%thread] - %msg%n</pattern>
106         </encoder>
107     </appender>
108     <appender name="MappingLog" class="ch.qos.logback.core.FileAppender">
109         <file>${logDirectory}/Mapping.log</file>
110         <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
111             <pattern>%date %-5level %logger{0} - %msg%n</pattern>
112         </encoder>
113     </appender>
114     <logger name="dbLoadingProgress" additivity="false" level="INFO">
115         <appender-ref ref="STDOUT-DbLoading"/>
116     </logger>
117     <appender name="NavigationLog" class="ch.qos.logback.core.FileAppender">
118         <file>${logDirectory}/Navigation.log</file>
119         <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
120             <pattern>%date %-5level %logger{0} - %msg%n</pattern>
121         </encoder>
122     </appender>
123     <appender name="EnvironmentLog" class="ch.qos.logback.core.FileAppender">
124         <file>${logDirectory}/Environment.log</file>
125         <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
126             <pattern>%date %-5level %logger{0} - %msg%n</pattern>
127         </encoder>
128     </appender>
129     <appender name="Cell2Log" class="ch.qos.logback.core.FileAppender">
130         <file>${logDirectory}/Cell2.log</file>
131         <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
132             <pattern>%date %-5level %logger{0} - %msg%n</pattern>
133         </encoder>
134     </appender>
135     <appender name="BatteryLog" class="ch.qos.logback.core.FileAppender">
136         <file>${logDirectory}/Battery.log</file>
137         <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
138             <pattern>%date %-5level %logger{0} - %msg%n</pattern>
139         </encoder>
140     </appender>
141     <appender name="ChargingStationLog" class="ch.qos.logback.core.FileAppender">
142         <file>${logDirectory}/ChargingStation.log</file>
143         <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
144             <pattern>%date %-5level %logger{0} - %msg%n</pattern>
145         </encoder>
146     </appender>
147
148     <!-- ##### LOGGER ##### -->
149     <!-- Creating the reference between certain VSimRTI classes and logback
150          appenders -->
151     <logger name="VSimRTIStarter" additivity="false" level="INFO">
152         <appender-ref ref="VsimrtiLog"/>
153     </logger>
154     <logger name="com.dcaiti.vsimrti.docker" additivity="false" level="INFO">
155         <appender-ref ref="VsimrtiLog"/>
156     </logger>
157     <logger name="com.dcaiti.vsimrti.fed.sumo" additivity="false" level="INFO">
158         <appender-ref ref="TrafficLog"/>
159     </logger>
160     <logger name="com.dcaiti.vsimrti.fed.omnetpp" additivity="false" level="INFO">
161         <appender-ref ref="CommunicationLog"/>
162     </logger>
163     <logger name="com.dcaiti.vsimrti.fed.ns3" additivity="false" level="INFO">
164         <appender-ref ref="CommunicationLog"/>
165     </logger>
166     <logger name="com.dcaiti.vsimrti.fed.sns" additivity="false" level="INFO">

```

```

167         <appender-ref ref="CommunicationLog"/>
168     </logger>
169     <!-- NOTE: do not specify logging levels other than "INFO" use omnetpp.ini (OMNeT2
170         ↪ ++) for logging configuration -->
171     <logger name="com.dcaiti.vsimrti.fed.ns3.ambassador.Ns3AmbassadorOutput" ↪
172         ↪ additivity="false" level="INFO">
173         <appender-ref ref="CommunicationDetailsLog"/>
174     </logger>
175     <logger name="com.dcaiti.vsimrti.fed.omnetpp.ambassador.OmnetppAmbassadorOutput" ↪
176         ↪ additivity="false" level="INFO">
177         <appender-ref ref="CommunicationDetailsLog"/>
178     </logger>
179     <logger name="com.dcaiti.vsimrti.fed.applicationNT.ambassador.navigation" ↪
180         ↪ additivity="false" level="INFO">
181         <appender-ref ref="NavigationLog"/>
182     </logger>
183     <logger name="com.dcaiti.vsimrti.fed.applicationNT" additivity="false" level="INFO"↪
184         ↪ ">
185         <appender-ref ref="ApplicationNTLog"/>
186     </logger>
187     <logger name="ApplicationNtLogDelegate" additivity="false" level="INFO">
188         <appender-ref ref="ApplicationNtLogDelegation"/>
189     </logger>
190     <logger name="com.dcaiti.vsimrti.fed.eventserver" additivity="false" level="INFO">
191         <appender-ref ref="EnvironmentLog"/>
192     </logger>
193     <logger name="com.dcaiti.vsimrti.fed.mapping3" additivity="false" level="INFO">
194         <appender-ref ref="MappingLog"/>
195     </logger>
196     <logger name="com.dcaiti.vsimrti.lib.routing" additivity="false" level="INFO">
197         <appender-ref ref="NavigationLog"/>
198     </logger>
199     <logger name="net.sf.jsi" additivity="false" level="OFF">
200         <appender-ref ref="NavigationLog"/>
201     </logger>
202     <logger name="com.dcaiti.vsimrti.fed.cell2" additivity="false" level="INFO">
203         <appender-ref ref="Cell2Log"/>
204     </logger>
205     <logger name="com.dcaiti.vsimrti.fed.battery" additivity="false" level="INFO">
206         <appender-ref ref="BatteryLog"/>
207     </logger>
208     <logger name="com.dcaiti.vsimrti.fed.phabmacs" additivity="false" level="INFO">
209         <appender-ref ref="TrafficLog"/>
210     </logger>
211     <logger name="com.dcaiti.vsimrti.fed.chargingstation" additivity="false" level="↪
212         ↪ INFO">
213         <appender-ref ref="ChargingStationLog"/>
214     </logger>
215     <logger name="statistics" additivity="false" level="OFF">
216         <appender-ref ref="StatisticsLog"/>
217     </logger>
218     <logger name="com.dcaiti.vsimrti.rti.services.time" additivity="false" level="INFO"↪
219         ↪ ">
220         <appender-ref ref="STDOUT-TimeAdvance"/>
221     </logger>
222     <logger name="com.dcaiti.vsimrti.start" additivity="false" level="INFO">
223         <appender-ref ref="STDOUT-MessageOnly"/>
224     </logger>
225     <!-- All other stuff, which was not logged by other loggers before goes
226         ↪ to stdout and VSimRTI.log -->
227     <root level="INFO">
228         <appender-ref ref="STDOUT"/>
229         <appender-ref ref="VsimrtiLog"/>
230     </root>

```

```
227         </then>
228     </if>
229
230 </configuration>
```

Listing A.3: VSimRTI: file listing: etc/logback.xml

---

### systeminfo.txt

---

```
1 {"userid": "0",
2  "version": "1.0",
3  "arch": "i686",
4  "os": "Linux",
5  "osversion": "12.04",
6  "cpumodel": "atom(tm)cpun270@1.60ghz",
7  "sockets": 1,
8  "cores": 2,
9  "memory": 1024,
10 "type": "checkin",
11 "loginmd5": "7fe5b9b9305ce3ffcd18e32bf0f0b9d9",
12 "simulationuuid": "",
13 "macs": ["00:22:00:00:0e:00", "00:00:54:00:00:00"]}
```

Listing A.4: VSimRTI: file listing: systeminfo.txt



# Appendix B

## Example scenario Barnim

You will find a collection of all relevant data.

### B.1 Folder Structure

```
Barnim
├── applicationNT .....
│   ├── applications .....
│   └── Barnim.db ..... not listed (binary content)
├── battery .....
│   └── battery_config.json ..... (see B.1)
├── eventserver .....
│   └── eventserver_config.json ..... not listed (in this release)
├── cell2 .....
│   ├── cell2_config.json ..... (see B.2)
│   ├── network.json ..... (see B.3)
│   └── regions.json ..... (see B.4)
├── mapping3 .....
│   └── mapping_config.json ..... (see B.5)
├── sns .....
│   └── sns_config.json ..... (see B.6)
├── sources .....
│   └── Barnim_fixed.osm ..... partially listed (see B.7)
├── sumo .....
│   ├── Barnim.edg.xml ..... partially listed (see B.8)
│   ├── Barnim.net.xml ..... partially listed (see B.9)
│   ├── Barnim.nod.xml ..... partially listed (see B.10)
│   ├── Barnim.rou.xml ..... (see B.11)
│   ├── Barnim.sumo.cfg ..... (see B.12)
│   └── sumo_config.json ..... (see B.13)
├── visualizer .....
│   └── visualizer_config.xml ..... (see B.14)
├── vsimrti .....
│   ├── vsimrti_config.xml ..... (see B.15)
│   └── simrun_config.xml ..... (see B.16)
```

Figure B.1: Scenario Barnim: folder structure

## B.2 File listings

### battery/battery\_config.json

---

```

1 {
2   "vehicleModelClass": "com.dcaiti.vsimrti.fed.battery.sim.models.vehicle.ElectricSmart",
3   "vehicleModelConfig": {
4     "Mass": 1060,
5     "ReferenceArea": 1.95,
6     "DragCoefficient": 0.375,
7     "TankToWheelEfficiency": 0.7,
8     "ElectricMotorOperatingVoltage": 350,
9     "ConsumerOperatingVoltage": 12
10  },
11  "batteryModelClass": "com.dcaiti.vsimrti.fed.battery.sim.models.battery.↵
    ↵ VerySimpleBatteryModel",
12  "batteryModelConfig": {
13    "NumberOfCells": 93,
14    "CellVoltage": 4.3,
15    "CellCapacityInAh": 50,
16    "eff_Summer": 0.8448,
17    "RechargingType": 2,
18    "MinSOC": 0.30,
19    "MaxSOC": 0.70
20  },
21  "environmentModelClass": "com.dcaiti.vsimrti.fed.battery.sim.models.environment.↵
    ↵ DefaultEnvironment",
22  "environmentModelConfig": {
23    "Gravity": 9.81,
24    "FluidDensity": 1.293,
25    "RollingResistanceCoefficient": 0.01
26  },
27  "consumers": [ { "Radio": 10 }, { "HeadLight": 100 } ]
28 }

```

---

Listing B.1: Scenario Barnim: file listing: battery/battery\_config.json

### cell2/cell2\_config.json

---

```

1 {
2   "debugGeocasting": false,
3   "visualizeRegions": false,
4   "networkConfigurationFile": "network.json",
5   "regionConfigurationFile": "regions.json"
6 }

```

---

Listing B.2: Scenario Barnim: file listing: cell2/cell2\_config.json

### cell2/network.json

---

```

1 {
2   "globalRegion": {
3     "uplink": {
4       "delay": {
5         "type": "ConstantDelay",
6         "delay": 100,
7         "prpl": 0,
8         "retries": 2
9       },
10    "capacity": 28000000

```

---

```

11     },
12     "downlink": {
13         "unicast": {
14             "delay": {
15                 "type": "ConstantDelay",
16                 "delay": 100,
17                 "prpl": 0,
18                 "retries": 2
19             }
20         },
21         "multicast": {
22             "delay": {
23                 "type": "ConstantDelay",
24                 "delay": 100,
25                 "prpl": 0
26             },
27             "usableCapacity": 0.6
28         },
29         "capacity": 42200000
30     }
31 }
32 }

```

Listing B.3: Scenario Barnim: file listing: cell2/network.json

### cell2/regions.json

```

1 {
2     "regions": [
3     ]
4 }

```

Listing B.4: Scenario Barnim: file listing: cell2/regions.json

### mapping3/mapping\_config.json

```

1 {
2     "prototypes": [
3         {
4             "name": "PKW",
5             "accel": 2.6,
6             "decel": 4.5,
7             "length": 5.00,
8             "maxSpeed": 70.0,
9             "minGap": 2.5,
10            "sigma": 0.5,
11            "tau": 1
12        },
13        {
14            "name": "electricPKW",
15            "vehicleClass": "ElectricVehicle",
16            "accel": 2.6,
17            "decel": 4.5,
18            "length": 5.00,
19            "maxSpeed": 40.0,
20            "minGap": 2.5,
21            "sigma": 0.5,
22            "tau": 1
23        },
24        {
25            "applications": ["com.dcaiti.vsimrti.app.tutorials.barnim.WeatherServer"],

```

```

26         "name": "WeatherServer"
27     }
28 ],
29     "rsus": [
30         {
31             "lat": 52.65027421760045,
32             "lon": 13.545005321502686,
33             "name": "WeatherServer"
34         }
35     ],
36     "vehicles": [
37         {
38             "startingTime": 5.0,
39             "targetDensity": 1200,
40             "maxNumberVehicles": 120,
41             "route": "1",
42             "types": [
43                 {
44                     "applications": ["com.dcaiti.vsimrti.app.tutorials.barnim.WeatherWarningAppCell", "↵
45                                 ↵ , "com.dcaiti.vsimrti.app.tutorials.barnim.SlowDownApp"],
46                     "name": "PKW",
47                     "weight": 0.1
48                 },
49                 {
50                     "applications": ["com.dcaiti.vsimrti.app.tutorials.barnim.WeatherWarningApp", "↵
51                                 ↵ com.dcaiti.vsimrti.app.tutorials.barnim.SlowDownApp"],
52                     "name": "PKW",
53                     "weight": 0.2
54                 },
55                 {
56                     "applications": ["com.dcaiti.vsimrti.app.tutorials.barnim.SlowDownApp"],
57                     "name": "PKW",
58                     "weight": 0.6
59                 },
60                 {
61                     "applications": ["com.dcaiti.vsimrti.app.tutorials.barnim.SlowDownApp"],
62                     "name": "electricPKW",
63                     "weight": 0.1
64                 }
65             ]
66         }

```

Listing B.5: Scenario Barnim: file listing: mapping3/mapping\_config.json

**sns/sns\_config.json**

```

1 {
2   "version": "0.16.0",
3
4   "singlehop": {
5     "_singlehop radius in m": 0,
6     "radius": 509.4,
7     "_delay config according to vsimrti-communication": 0,
8     "delay": {
9       "type": "SimpleRandomDelay",
10      "steps": 5,
11      "minDelay": 0.4,
12      "maxDelay": 2.4,
13      "prpl": 0,
14      "retries": 0
15    }
16  },
17
18  "multihop": {
19    "_delay config according to vsimrti-communication": 0,
20    "delay": {
21      "type": "GammaRandomDelay",
22      "minDelay": 10,
23      "expDelay": 30,
24      "prpl": 0,
25      "retries": 2
26    }
27  }
28 }

```

Listing B.6: Scenario Barnim: file listing: sns/sns\_config.json

**sources/Barnim\_fixed.osm**

Note: This file is partially listed.

```

1 <?xml version='1.0' encoding='UTF-8'?>
2 <osm version='0.6' upload='true' generator='JOSM'>
3   <bounds minlat='52.59241' minlon='13.51215' maxlat='52.65827' maxlon='13.6227' origin='↵
4     ↵ CGImap 0.3.3 (11990 thorn-02.openstreetmap.org)' />
5   <node id='9671195' timestamp='2014-03-25T09:54:31Z' uid='43566' user='anbr' visible='true' ↵
6     ↵ version='4' changeset='2383615' lat='52.5798854' lon='13.6374307' />
7   <node id='9671197' timestamp='2014-03-25T09:54:31Z' uid='43566' user='anbr' visible='true' ↵
8     ↵ version='5' changeset='14033189' lat='52.5905574' lon='13.6219588' />
9   <node id='9671202' timestamp='2014-03-25T09:54:31Z' uid='429794' user='SennaHB' visible='↵
10    ↵ true' version='5' changeset='13985567' lat='52.6051075' lon='13.5993505' />
11   <node id='21508739' timestamp='2014-03-25T09:54:31Z' uid='43566' user='anbr' visible='true' ↵
12    ↵ version='6' changeset='17739489' lat='52.6639091' lon='13.5696135' />
13   <node id='21508746' timestamp='2014-03-25T09:54:31Z' uid='118278' user='Panketaler' visible='↵
14    ↵ 'true' version='12' changeset='19973396' lat='52.6240762' lon='13.5434165' />
15   <node id='21508747' timestamp='2014-03-25T09:54:31Z' uid='118278' user='Panketaler' visible='↵
16    ↵ 'true' version='27' changeset='4921151' lat='52.6297992' lon='13.5459023' />
17   <node id='21508748' timestamp='2014-03-25T09:54:31Z' uid='118278' user='Panketaler' visible='↵
18    ↵ 'true' version='4' changeset='4746323' lat='52.6392207' lon='13.5605381' />
19   <node id='21508756' timestamp='2014-03-25T09:54:31Z' uid='118278' user='Panketaler' visible='↵
20    ↵ 'true' version='6' changeset='19973383' lat='52.6266342' lon='13.5441771' />
21   <node id='21508757' timestamp='2014-03-25T09:54:31Z' uid='43566' user='anbr' visible='true' ↵
22    ↵ version='4' changeset='17739489' lat='52.662124' lon='13.569613' />
23   <node id='21508763' timestamp='2014-03-25T09:54:31Z' uid='429794' user='SennaHB' visible='↵
24    ↵ true' version='8' changeset='18626470' lat='52.614571' lon='13.5536105' />
25   <node id='26971235' timestamp='2014-03-25T09:54:31Z' uid='429794' user='SennaHB' visible='↵
26    ↵ true' version='9' changeset='16740262' lat='52.6126837' lon='13.5681426' />

```

```

15 <node id='26971236' timestamp='2014-03-25T09:54:31Z' uid='429794' user='SennaHB' visible='↵
    ↵ true' version='9' changeset='16740262' lat='52.6124118' lon='13.5701801' />
16 <node id='26971237' timestamp='2014-03-25T09:54:31Z' uid='579301' user='Lamarqe' visible='↵
    ↵ true' version='5' changeset='12440961' lat='52.6236544' lon='13.5757994' />
17 <node id='26971240' timestamp='2014-03-25T09:54:31Z' uid='429794' user='SennaHB' visible='↵
    ↵ true' version='5' changeset='13985567' lat='52.6137568' lon='13.564784' />
18 <node id='26971241' timestamp='2014-03-25T09:54:31Z' uid='90528' user='Peter Maiwald' ↵
    ↵ visible='true' version='4' changeset='8364243' lat='52.6135612' lon='13.565324' />
19 <node id='26971244' timestamp='2014-03-25T09:54:31Z' uid='429794' user='SennaHB' visible='↵
    ↵ true' version='6' changeset='13985567' lat='52.614347' lon='13.5636669' />
20 <node id='26971267' timestamp='2014-03-25T09:54:31Z' uid='429794' user='SennaHB' visible='↵
    ↵ true' version='9' changeset='18626470' lat='52.6141673' lon='13.5561542' />
21 <node id='26971272' timestamp='2014-03-25T09:54:31Z' uid='429794' user='SennaHB' visible='↵
    ↵ true' version='11' changeset='12771612' lat='52.6131516' lon='13.5672267' />
22 <node id='26971273' timestamp='2014-03-25T09:54:31Z' uid='429794' user='SennaHB' visible='↵
    ↵ true' version='14' changeset='18626470' lat='52.6150704' lon='13.5439418' />

```

Listing B.7: Scenario Barnim: file listing: sources/Barnim\_fixed.osm

## sumo/Barnim.edg.xml

Note: This file is partially listed.

```

1 <edges>
2   <edge id="23995140_260162539_260162554_260162539" from="260162539" to="260162554" numLanes=↵
    ↵ "1" speed="8.33" />
3   <edge id="23995140_260162539_260162558_260162539" from="260162539" to="260162558" numLanes=↵
    ↵ "1" speed="8.33" />
4   <edge id="116570245_1313885509_866614488_1313885509" from="1313885509" to="866613837" ↵
    ↵ numLanes="2" speed="22.22" />
5   <edge id="116570245_1313885509_866614488_866613837" from="866613837" to="2026362890" ↵
    ↵ numLanes="2" speed="22.22" />

```

Listing B.8: Scenario Barnim: file listing: sumo/Barnim.edg.xml

## sumo/Barnim.net.xml

Note: This file is partially listed.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <!-- generated on 02/15/17 17:22:13 by SUMO netconvert Version 0.28.0
4 <?xml version="1.0" encoding="UTF-8"?>
5
6 <configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ↵
    ↵ xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/wsd/netconvertConfiguration.xsd">
7
8   <input>
9     <node-files value="D:\dev\scenarios\Barnim-Startaufgabe\applicationNT\Barnim.nod.xml" ↵
    ↵ "/>
10    <edge-files value="D:\dev\scenarios\Barnim-Startaufgabe\applicationNT\Barnim.edg.xml" ↵
    ↵ "/>
11    <connection-files value="D:\dev\scenarios\Barnim-Startaufgabe\applicationNT\Barnim.con" ↵
    ↵ ".xml"/>
12  </input>
13
14  <output>
15    <output-file value="D:\dev\scenarios\Barnim-Startaufgabe\applicationNT\Barnim.net.xml" ↵
    ↵ "/>
16  </output>
17

```

```

18 </configuration>
19 -->
20
21 <net version="0.27" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ↵
    ↵ xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/net_file.xsd">
22
23 <location netOffset="-397957.77,-5826114.68" convBoundary="0.00,0.00,9905.80,10297.64" ↵
    ↵ origBoundary="397957.77,5826114.68,407863.57,5836412.32" projParameter="!"/>
24
25 <edge id=":1003778301_0" function="internal">
26 <lane id=":1003778301_0_0" index="0" speed="8.33" length="0.29" shape="↵
    ↵ 1221.68,7155.88,68.90 1221.42,7155.74,68.90"/>
27 </edge>
28 <edge id=":1003778301_1" function="internal">
29 <lane id=":1003778301_1_0" index="0" speed="8.33" length="0.31" shape="↵
    ↵ 1223.08,7152.88,68.90 1223.34,7153.03,68.90"/>
30 </edge>

```

Listing B.9: Scenario Barnim: file listing: sumo/Barnim.net.xml

## sumo/Barnim.nod.xml

Note: This file is partially listed.

```

1 <nodes>
2 <node id="866613617" x="401721.2894" y="5830383.5757" z="58.9788" type="priority" />
3 <node id="2632245436" x="399909.1546" y="5834705.7934" z="60.0000" type="priority" />
4 <node id="41240371" x="405205.4892" y="5835368.8690" z="69.9442" type="priority" />
5 <node id="268746480" x="402891.3857" y="5833194.2388" z="68.1866" type="priority" />

```

Listing B.10: Scenario Barnim: file listing: sumo/Barnim.nod.xml

## sumo/Barnim.rou.xml

```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <!-- generated on 04/15/14 14:26:30 by SUMO duarouter Version 0.19.0
4 <?xml version="1.0" encoding="UTF-8"?>
5
6 <configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ↵
    ↵ xsi:noNamespaceSchemaLocation="http://sumo-sim.org/xsd/duarouterConfiguration.xsd">
7
8 <input>
9 <net-file value=".\\Barnim.net.xml"/>
10 <trip-files value=".\\trips.xml"/>
11 </input>
12
13 <output>
14 <output-file value="Barnim.rou.xml"/>
15 </output>
16
17 </configuration>
18 -->
19
20 <routes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="↵
    ↵ http://sumo-sim.org/xsd/routes_file.xsd">
21 <!--vType model="SUMO_KRAUSS" id="PKW" /-->
22

```

```

23 <route id="1" edges="4067968_28830219_39933927_2450938914 4067968 ↵
    ↵ _28830219_39933927_60635629 4067968_28830219_39933927_267925251 4067968 ↵
    ↵ _28830219_39933927_599826230 62710154_39933927_39933921_39933927 73017982 ↵
    ↵ _39933921_2026362940_39933921 73017982_39933921_2026362940_2026362943 -3366 ↵
    ↵ _2026362940_1313885502_2026362940 -3366_2026362940_1313885502_2026362938 -3366 ↵
    ↵ _2026362940_1313885502_2364474851 -3366_2026362940_1313885502_2364474850 -3366 ↵
    ↵ _2026362940_1313885502_2026362934 -3366_2026362940_1313885502_2364474850 -3366 ↵
    ↵ _2026362940_1313885502_2026362930 -3366_2026362940_1313885502_2026362929 73017981 ↵
    ↵ _1313885502_866614526_1313885502 73017981_1313885502_866614526_2364474849 73017981 ↵
    ↵ _1313885502_866614526_1313885493 73017981_1313885502_866614526_2364474847 73017981 ↵
    ↵ _1313885502_866614526_1313885455 73017981_1313885502_866614526_1313885452 73017981 ↵
    ↵ _1313885502_866614526_866614218 73017981_1313885502_866614526_1313885474 73017981 ↵
    ↵ _1313885502_866614526_866614127 73017981_1313885502_866614526_866614379 73017981 ↵
    ↵ _1313885502_866614526_866614760 73017981_1313885502_866614526_866613761 73017981 ↵
    ↵ _1313885502_866614526_866614254 116570220_866614526_866613611_866614526 116570220 ↵
    ↵ _866614526_866613611_1313885383 116570220_866614526_866613611_2314374830 116570220 ↵
    ↵ _866614526_866613611_2026362918 73017984_866613611_866613693_866613611 73017984 ↵
    ↵ _866613611_866613693_2538574191 73017984_866613611_866613693_1313885521 73017984 ↵
    ↵ _866613611_866613693_1313885489 73017984_866613611_866613693_866614362 73017984 ↵
    ↵ _866613611_866613693_866614700 106057563_866613693_866614474_866613693 106057563 ↵
    ↵ _866613693_866614474_866614207 106057563_866613693_866614474_1313885413 106057563 ↵
    ↵ _866613693_866614474_1313885451 116570250_866614474_866613833_866614474 116570250 ↵
    ↵ _866614474_866613833_2026362855 116570250_866614474_866613833_1313885473 116570237 ↵
    ↵ _866613833_866613992_866613833 116570237_866613833_866613992_1313885484 116570237 ↵
    ↵ _866613833_866613992_2026362842 116570237_866613833_866613992_1313885386 116570237 ↵
    ↵ _866613833_866613992_1313885524 116570237_866613833_866613992_866614195 116570237 ↵
    ↵ _866613833_866613992_866614570 116570237_866613833_866613992_1313885441 116570237 ↵
    ↵ _866613833_866613992_866613497 165881196_866613992_866614320_866613992 165881196 ↵
    ↵ _866613992_866614320_866614684 165881196_866613992_866614320_1313885548 165881196 ↵
    ↵ _866613992_866614320_1313885531 165881196_866613992_866614320_2026362847 165881196 ↵
    ↵ _866613992_866614320_1313885511 165881196_866613992_866614320_2026362848 165881196 ↵
    ↵ _866613992_866614320_1313885515 165881196_866613992_866614320_1313885471 165881196 ↵
    ↵ _866613992_866614320_1313885435 165881196_866613992_866614320_1313885393 165881196 ↵
    ↵ _866613992_866614320_1313885492 165881196_866613992_866614320_1313885454 165881196 ↵
    ↵ _866613992_866614320_1313885415 232375390_866614320_866614320_866614320 232375390 ↵
    ↵ _866614323_866614062_866614323 248048346_866614062_866613506_866614062 248048344 ↵
    ↵ _866613506_866613555_866613506 245434998_866613555_1313885532_866613555 245434998 ↵
    ↵ _866613555_1313885532_2389986818 245434998_866613555_1313885532_866613905 116570222 ↵
    ↵ _1313885532_2026362814_1313885532 116570222_1313885532_2026362814_1313885431 ↵
    ↵ 116570222_1313885532_2026362814_2026362827 116570222 ↵
    ↵ _1313885532_2026362814_2026362821 192079314_2026362814_692755381_2026362814 ↵
    ↵ 192079314_2026362814_692755381_2510575567 182262835_692755381_49832298_692755381" />
24 <route id="2" edges="4067968_28830219_39933927_2450938914 4067968 ↵
    ↵ _28830219_39933927_60635629 4067968_28830219_39933927_267925251 4067968 ↵
    ↵ _28830219_39933927_599826230 5477467_39933927_335376891_39933927 5477467 ↵
    ↵ _39933927_335376891_39934065 5477467_39933927_335376891_267923737 5477467 ↵
    ↵ _39933927_335376891_267923739 5477467_39933927_335376891_267923741 166752304 ↵
    ↵ _335376891_39933917_335376891 166752304_39933917_75666594_39933917 166752304 ↵
    ↵ _39933917_75666594_564006580 166752304_75666594_75673057_75666594 166752304 ↵
    ↵ _75666594_75673057_76565701 166752304_75666594_75673057_739356847 166752304 ↵
    ↵ _75666594_75673057_21508748 166752304_75666594_75673057_739356839 166752304 ↵
    ↵ _75673057_303184519_75673057 166752304_75673057_303184519_739356830 166752304 ↵
    ↵ _303184519_1781963887_303184519 166752301_1781963887_73586268_1781963887 166752301 ↵
    ↵ _1781963887_73586268_2250683387 30602177_73586268_75679644_73586268 30602177 ↵
    ↵ _73586268_75679644_765965511 229068361_75679644_21508747_75679644 229068361 ↵
    ↵ _21508747_2621152986_21508747 229068361_21508747_2621152986_765965509 229068361 ↵
    ↵ _21508747_2621152986_2625982333 229068361_21508747_2621152986_2621153010 229068361 ↵
    ↵ _21508747_2621152986_2621153009 229068361_21508747_2621152986_2621153003 229068361 ↵
    ↵ _21508747_2621152986_2621152999 229068361_21508747_2621152986_73586264 229068361 ↵
    ↵ _21508747_2621152986_21508756 229068361_2621152986_2621152983_2621152986 111283478 ↵
    ↵ _2621152983_2621152974_2621152983 111283478_2621152983_2621152974_2621152977 ↵
    ↵ 111283478_2621152983_2621152974_2621152975 229068360 ↵
    ↵ _2621152974_2621152972_2621152974 229068360_2621152972_21508746_2621152972 229068360 ↵
    ↵ _2621152972_21508746_601059352 229068360_2621152972_21508746_1600991158 229068360 ↵
    ↵ _2621152972_21508746_1600991160 229068357_21508746_2510564806_21508746 229068357 ↵
    ↵ _21508746_2510564806_1600991159 229068357_21508746_2510564806_1600991155 229068357 ↵
    ↵ _21508746_2510564806_2377050402 229068357_21508746_2510564806_1600991157 229068357 ↵

```



```

25 ↪ _21508746_2510564806_661749240 229068357_21508746_2510564806_416065959 229068357 ↪
26 ↪ _21508746_2510564806_1600991154 229068357_21508746_2510564806_661749237 146875503 ↪
27 ↪ _2510564806_27031232_2510564806 146875503_2510564806_27031232_1600991153 146875503 ↪
28 ↪ _2510564806_27031232_1600991156 146875503_2510564806_27031232_1587700342 146875503 ↪
↪ _2510564806_27031232_2510564804 146871033_27031232_27031229_27031232 227901601 ↪
↪ _27031229_866613841_27031229 227901601_27031229_866613841_2526390859 227901601 ↪
↪ _27031229_866613841_866613799 227901601_27031229_866613841_866614008 227901601 ↪
↪ _27031229_866613841_1313885439 227901601_27031229_866613841_1313885527 227901601 ↪
↪ _27031229_866613841_1313885436 227901601_27031229_866613841_866613786 245432872 ↪
↪ _866613841_866614578_866613841 116570239_866614578_866613555_866614578 116570239 ↪
↪ _866614578_866613555_1313885456 245434998_866613555_1313885532_866613555 245434998 ↪
↪ _866613555_1313885532_2389986818 245434998_866613555_1313885532_866613905 116570222 ↪
↪ _1313885532_2026362814_1313885532 116570222_1313885532_2026362814_1313885431 ↪
↪ 116570222_1313885532_2026362814_2026362827 116570222 ↪
↪ _1313885532_2026362814_2026362821 192079314_2026362814_692755381_2026362814 ↪
↪ 192079314_2026362814_692755381_2510575567 182262835_692755381_49832298_692755381" />
25
26 <!--vehicle id="0" type="PKW" route="1" depart="1" repno="500" period="5" /> Standard route ↪
27 ↪ -->
28 <!--vehicle id="1" type="PKW" route="2" depart="3" repno="20" period="20" /> Ausweichroute ↪
29 ↪ -->
30 </routes>

```

Listing B.11: Scenario Barnim: file listing: sumo/Barnim.rou.xml

### sumo/Barnim.sumo.cfg

```

1 <configuration>
2   <input>
3     <net-file value="Barnim.net.xml" />
4     <route-files value="Barnim.rou.xml" />
5   </input>
6   <time>
7     <begin value="0" />
8     <end value="10000" />
9   </time>
10 </configuration>

```

Listing B.12: Scenario Barnim: file listing: sumo/Barnim.sumo.cfg

### sumo/sumo\_config.json

```

1 {
2   "sumoConfigurationFile" : "Barnim.sumo.cfg"
3 }

```

Listing B.13: Scenario Barnim: file listing: sumo/sumo\_config.json

**visualizer/visualizer\_config.xml**

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <!-- file version: 2013-11-26 -->
3  <configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:noNamespaceSchemaLocation="http://www.dcaiti.tu-berlin.de/research/
      ↳ simulation/download/get/scenarios/scenarioname/visualizer/
      ↳ visualizer_config.xsd">
5      <visualizer id="filevisualizer" enabled="true" update="5" loader="com.dcaiti.vsimrti.fed.
      ↳ visualizer.FileVisualizerConfig">
6          <filename>visualizer.csv</filename>
7          <directory>.</directory>
8          <separator>;</separator>
9          <messages>
10             <message id="VehicleMovements">
11                 <entries>
12                     <entry>"MOVE_VEHICLE"</entry>
13                     <entry>Time</entry>
14                     <entry>Updated:Name</entry>
15                     <entry>Updated:Position.Latitude</entry>
16                     <entry>Updated:Position.Longitude</entry>
17                     <entry>Updated:Speed</entry>
18                     <entry>Updated:Heading</entry>
19                 </entries>
20             </message>
21             <message id="ReceiveV2XMessage">
22                 <entries>
23                     <entry>"RCV_MESSAGE"</entry>
24                     <entry>Time</entry>
25                     <entry>MessageId</entry>
26                     <entry>ReceiverName</entry>
27                     <entry>Type</entry>
28                 </entries>
29             </message>
30             <message id="SendV2XMessage">
31                 <entries>
32                     <entry>"SEND_MESSAGE"</entry>
33                     <entry>Time</entry>
34                     <entry>MessageId</entry>
35                     <entry>SourceName</entry>
36                     <entry>Type</entry>
37                     <!-- <entry>Destination.Position.Latitude</entry> -->
38                     <!-- <entry>Destination.Position.Longitude</entry> -->
39                     <!-- <entry>Destination.Radius</entry> -->
40                 </entries>
41             </message>
42             <message id="AddedVehicle" enabled="true">
43                 <entries>
44                     <entry>"ADDED_VEHICLE"</entry>
45                     <entry>Time</entry>
46                     <entry>ApplicationVehicle.Name</entry>
47                     <entry>ApplicationVehicle.Applications</entry>
48                     <entry>ApplicationVehicle.VehicleType.Name</entry>
49                 </entries>
50             </message>
51             <message id="AddedTrafficLight">
52                 <entries>
53                     <entry>"ADDED_TRAFFICLIGHT"</entry>
54                     <entry>Time</entry>
55                     <entry>ApplicationTrafficLight.Name</entry>
56                     <entry>ApplicationTrafficLight.Applications</entry>
57                     <entry>TrafficLightGroup.FirstPosition.Latitude</entry>
58                     <entry>TrafficLightGroup.FirstPosition.Longitude</entry>
59                 </entries>
60             </message>
61             <message id="AddedRsu">
62                 <entries>

```

```

63         <entry>"ADDED_RSU"</entry>
64         <entry>Time</entry>
65         <entry>ApplicationRsu.Name</entry>
66         <entry>ApplicationRsu.Applications</entry>
67         <entry>ApplicationRsu.Position.Latitude</entry>
68         <entry>ApplicationRsu.Position.Longitude</entry>
69     </entries>
70 </message>
71 </messages>
72 </visualizer>
73
74 <visualizer id="eworld" enabled="false"
75     loader="com.dcaiti.vsimrti.fed.visualizer.eworldvisualizer.config.↵
76     ↵ EworldVisualizerConfig">
77     <synchronized>true</synchronized>
78     <host>local</host>
79     <port>50500</port>
80     <messages>
81         <message id="AddedTrafficLight"></message>
82         <message id="AddedRsu"></message>
83         <message id="AddedVehicle"></message>
84         <message id="VehicleMovements"></message>
85     </messages>
86 </visualizer>
87
88 <visualizer id="websocket" enabled="true" loader="com.dcaiti.vsimrti.fed.visualizer.↵
89     ↵ WebSocketVisualizerConfig">
90     <synchronized>true</synchronized>
91     <port>46587</port>
92     <messages>
93         <message id="VehicleMovements" enabled="true"></message>
94         <message id="ReceiveV2XMessage" enabled="true"></message>
95         <message id="SendV2XMessage" enabled="true"></message>
96         <message id="ReceiveCellMessage" enabled="false"></message>
97         <message id="SendCellMessage" enabled="false"></message>
98         <message id="AddedVehicle" enabled="true"></message>
99         <message id="AddedRsu" enabled="true"></message>
100        <message id="AddedTrafficLight" enabled="false"></message>
101        <message id="AddedChargingStation" enabled="false"></message>
102        <message id="ChargingStationUpdate" enabled="false"></message>
103    </messages>
104 </visualizer>
</configuration>

```

Listing B.14: Scenario Barnim: file listing: visualizer/visualizer\_config.xml

## vsimrti/vsimrti\_config.xml

```

1  íž&lt;?xml version="1.0" encoding="UTF-8"?>
2  <configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xsi:noNamespaceSchemaLocation="https://www.dcaiti.tu-berlin.de/research/
      ↵ simulation/download/get/scenarios/scenarioname/vsimrti/vsimrti_config.
      ↵ xsd">
4      <simulation>
5          <!-- Scenario name -->
6          <id>Barnim</id>
7          <!-- Simulation time frame -->
8          <starttime>0</starttime>
9          <endtime>1200</endtime>
10         <!-- Seed for pseudo random number generator used by most of the federates -->
11         <randomSeed>268965854</randomSeed>
12         <!-- Projection between global and cartesian coordinates -->
13         <!-- centerCoordinates: roughly the center of simulation area -->
14         <!-- cartesianOffset: can be found in the generated network file for the traffic
      ↵ simulator, e.g. the .net.xml file for sumo -->
15         <WGS84UTMTransformConfig>
16         {
17             "centerCoordinates": {
18                 "longitude": 13.56,
19                 "latitude": 52.63
20             },
21             "cartesianOffset": {
22                 "x": -397957.77,
23                 "y": -5826114.68
24             }
25         }
26         </WGS84UTMTransformConfig>
27         <!-- Global IP management -->
28         <IPResolverConfig>
29         {
30             netMask: "255.255.0.0",
31             vehicleNet: "10.1.0.0",
32             rsuNet: "10.2.0.0",
33             tlNet: "10.3.0.0",
34             csNet: "10.4.0.0",
35             serverNet: "10.5.0.0"
36         }
37         </IPResolverConfig>
38         <threads>1</threads>
39     </simulation>
40     <federates>
41         <!-- Cellular network simulator -->
42         <federate id="cell12" active="false"/>
43         <!-- V2X (ad hoc) network simulators -->
44         <federate id="omnetpp" active="false"/>
45         <federate id="ns3" active="false"/>
46         <federate id="sns" active="true"/>
47         <!-- Traffic simulators -->
48         <federate id="sumo" active="true"/>
49         <!-- Application simulator -->
50         <federate id="applicationNT" active="true"/>
51         <!-- Environment simulator -->
52         <federate id="eventserver" active="true"/>
53         <!-- Electric mobility simulators -->
54         <federate id="battery" active="true"/>
55         <federate id="chargingstation" active="false"/>
56         <!-- Mapping -->
57         <federate id="mapping3" active="true"/>
58         <!-- Visualization -->
59         <federate id="visualizer" active="true"/>
60     </federates>
61 </configuration>

```

Listing B.15: Scenario Barnim: file listing: vsimrti/vsimrti\_config.xml

**vsimrti/simrun\_config.xml**

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- file version: 2014-08-27 -->
3 <!--
4 This configuration induces the execution of 6 simulations
5 (2 dimensions with 5 parameter for each dimension).
6 The simulations will be executed one after another (sequential mode).
7 following simulations would be triggered:
8
9 The parameter set
10 (V2XVehiclePercentage, CellVehiclePercentage, ClassicVehiclePercentage and Simulationtime)
11 nest to parameter
12 (SinglehopRadius).
13 They define the dimensions of the simulations matrix.
14
15 ===== 001. simulation =====
16 V2XVehiclePercentage=0
17 CellVehiclePercentage=0
18 ClassicVehiclePercentage=100
19 Simulationtime=100
20 SinglehopRadius=500
21
22 ===== 002. simulation =====
23 V2XVehiclePercentage=0
24 CellVehiclePercentage=0
25 ClassicVehiclePercentage=100
26 Simulationtime=100
27 SinglehopRadius=600
28
29 ===== 003. simulation =====
30 V2XVehiclePercentage=50
31 CellVehiclePercentage=0
32 ClassicVehiclePercentage=50
33 Simulationtime=100
34 SinglehopRadius=500
35
36 ===== 004. simulation =====
37 V2XVehiclePercentage=50
38 CellVehiclePercentage=0
39 ClassicVehiclePercentage=50
40 Simulationtime=100
41 SinglehopRadius=600
42
43 ===== 005. simulation =====
44 V2XVehiclePercentage=75
45 CellVehiclePercentage=0
46 ClassicVehiclePercentage=25
47 Simulationtime=100
48 SinglehopRadius=500
49
50 ===== 006. simulation =====
51 V2XVehiclePercentage=75
52 CellVehiclePercentage=0
53 ClassicVehiclePercentage=25
54 Simulationtime=100
55 SinglehopRadius=600
56
57 ===== 007. simulation =====
58 V2XVehiclePercentage=0
59 CellVehiclePercentage=50
```

```

60 ClassicVehiclePercentage=50
61 Simulationtime=100
62 SinglehopRadius=500
63
64 ===== 008. simulation =====
65 V2XVehiclePercentage=0
66 CellVehiclePercentage=50
67 ClassicVehiclePercentage=50
68 Simulationtime=100
69 SinglehopRadius=600
70
71 ===== 009. simulation =====
72 V2XVehiclePercentage=0
73 CellVehiclePercentage=75
74 ClassicVehiclePercentage=25
75 Simulationtime=100
76 SinglehopRadius=500
77
78 ===== 010. simulation =====
79 V2XVehiclePercentage=0
80 CellVehiclePercentage=75
81 ClassicVehiclePercentage=25
82 Simulationtime=100
83 SinglehopRadius=600
84
85 -->
86 <configuration
87     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
88     xsi:noNamespaceSchemaLocation="http://www.dcaiti.tu-berlin.de/research/simulation/download/↵
↵ get/scenarios/scenarioname/vsimrti/simrun_config.xsd">
89     <!-- basic configuration -->
90     <vsimrti location="/path/to/your/vsimrti_folder" executable="vsimrti.sh" username="↵
↵ your_user_id" />
91     <scenario name="Barnim" config="scenarios/Barnim/vsimrti/vsimrti_config.xml" persistent="↵
↵ false" repetitions="1" progressLogger="true">
92         <!-- argument -o TRACE -->
93         <!-- argument -w 0 -->
94     </scenario>
95
96     <!-- define connected values for controlled changes -->
97     <dimension name="SimulationRuns">
98         <parameter name="V2XVehiclePercentage" file="mapping3/mapping_config.json" fileFormat="↵
↵ json" item="vehicles[0].types[0].weight" type="ValueList">
99             <value>0</value>
100            <value>50</value>
101            <value>75</value>
102            <value>0</value>
103            <value>0</value>
104        </parameter>
105        <parameter name="CellVehiclePercentage" file="mapping3/mapping_config.json" fileFormat↵
↵ ="json" item="vehicles[0].types[0].weight" type="ValueList">
106            <value>0</value>
107            <value>0</value>
108            <value>0</value>
109            <value>50</value>
110            <value>75</value>
111        </parameter>
112        <parameter name="ClassicVehiclePercentage" file="mapping3/mapping_config.json" ↵
↵ fileFormat="json" item="vehicles[0].types[2].weight" type="ValueList">
113            <value>100</value>
114            <value>50</value>
115            <value>25</value>
116            <value>50</value>
117            <value>25</value>
118        </parameter>
119        <parameter name="Simulationtime" file="vsimrti/vsimrti_config.xml" fileFormat="xml" item↵
↵ ="//simulation/endtime" type="ValueList">

```

```
120     <value>100</value>
121     <value>100</value>
122     <value>100</value>
123         <value>100</value>
124         <value>100</value>
125     </parameter>
126 </dimension>
127
128 <!-- define values for automatically permuted simulations -->
129 <parameter name="SinglehopRadius" file="sns/sns_config.json" fileFormat="json" item="↵"
130     ↵ singlehop.radius" type="ValueList">
131     <value>500</value>
132     <value>600</value>
133 </parameter>
</configuration>
```

Listing B.16: Scenario Barnim: file listing: vsimrti/simrun\_config.xml

# Appendix C

## Example scenario Tiergarten

You will find a collection of all relevant data.

### C.1 Folder Structure

Tiergarten	
├─ applicationNT .....	
│   └─ applications .....	
│   └─ applicationNT_config.json .....	not listed (in this release)
│   └─ tiergarten.db .....	not listed (binary content)
├─ cell2 .....	
│   └─ cell2_config.json .....	not listed (in this release)
│   └─ network.json .....	not listed (in this release)
│   └─ regions.json .....	not listed (in this release)
├─ mapping3 .....	
│   └─ mapping_config.json .....	not listed (in this release)
├─ omnetpp .....	
│   └─ omnetpp.ini .....	(see C.1)
├─ ns3 .....	
│   └─ configTechnologies.xml .....	not listed (in this release)
│   └─ confWifi.xml .....	not listed (in this release)
├─ sns .....	
│   └─ sns_config.json .....	not listed (in this release)
├─ sources .....	
│   └─ tiergarten.osm .....	not listed (in this release)
├─ sumo .....	
│   └─ sumo_config.json .....	not listed (in this release)
│   └─ tiergarten.edg.xml .....	not listed (in this release)
│   └─ tiergarten.net.xml .....	not listed (in this release)
│   └─ tiergarten.nod.xml .....	not listed (in this release)
│   └─ tiergarten.rou.xml .....	not listed (in this release)
│   └─ tiergarten.sumo.cfg .....	not listed (in this release)
├─ visualizer .....	
│   └─ visualizer_config.xml .....	not listed (in this release)
├─ vsimrti .....	
│   └─ vsimrti_config.xml .....	not listed (in this release)

Figure C.1: Scenario Tiergarten: folder structure



## C.2 File listings

### omnetpp/omnetpp.ini

```

1 [General]
2
3 # SimulationParameters
4 # -----
5 # network *.ned-file and time-scale in nanoseconds
6 network = Simulation
7 simtime-scale = -9
8
9 # EventScheduler
10 # -----
11 # scheduler-class and debugging option for more verbose logging
12 scheduler-class = "VSimRTIEventScheduler"
13 vsimrtieventscheduler-debug = false
14 # connection settings, when omnetpp-federate is started manually
15 vsimrtieventscheduler-host = "localhost"
16 vsimrtieventscheduler-port = 4998
17
18 # RecordingModi
19 # -----
20 record-eventlog = false
21 cmdenv-express-mode = false
22 cmdenv-event-banners = false
23
24 # random numbers
25 # -----
26 num-rngs = 3
27 Simulation.mobility.rng-0 = 1
28 Simulation.wlan[*].mac.rng-0 = 2
29
30 # general logging output
31 # -----
32 *.mgmt.cmdenv-ev-output = true
33 *.mgmt.debug = false
34 *.veh[*]**.cmdenv-ev-output = false
35 *.rsu[*]**.cmdenv-ev-output = false
36
37
38 ##### application settings #####
39 Simulation.rsu[*].udpApp.maxProcDelay = 1e-3
40 Simulation.veh[*].udpApp.maxProcDelay = 1e-3
41
42
43 ##### UDP Settings #####
44
45 ##### network layer settings #####
46
47 ##### mac and phy settings #####
48 **.wlan0.opMode = "p"
49 **.wlan0.bitrate = 6Mbps
50
51 **.wlan0.mac.basicBitrate = 3Mbps # we have to set this explicitly because the default value
    ↳ of 1Mbps is not part of 802.11p
52 **.wlan0.mac.controlBitrate = 3Mbps
53 **.wlan0.mac.retryLimit = 7
54 **.wlan0.mac.maxQueueSize = 10
55 **.wlan0.mac.cwMinData = 15
56 **.wlan0.mac.cwMinMulticast = 31
57
58
59 **.wlan0.radio.bandName = "5.9 GHz" #this string actually selects the band from {2.4; 5; 5.9}
    ↳ (see "IEEE80211Band.h")
60 **.wlan0.radio.carrierFrequency = 5.9GHz

```


```
61 **.wlan0.radio.bandwidth = 10MHz
62 **.wlan0.radio.receiver.snirThreshold = 4dB
63 **.wlan0.radio.receiver.sensitivity = -81dBm
64
65
66 **.wlan1.opMode = "p"
67 **.wlan1.bitrate = 6Mbps
68 **.wlan1.mac.basicBitrate = 3Mbps # we have to set this explicitly because the default value
    ↪ of 1Mbps is not part of 802.11p
69 **.wlan1.mac.controlBitrate = 3Mbps
70 **.wlan1.mac.retryLimit = 7
71 **.wlan1.mac.maxQueueSize = 10
72 **.wlan1.mac.cwMinData = 15
73 **.wlan1.mac.cwMinMulticast = 31
74
75 **.wlan1.radio.bandName = "5.9 GHz" #this string actually selects the band from {2.4; 5; 5.9} ↪
    ↪ (see "IIEEE80211Band.h")
76 **.wlan1.radio.carrierFrequency = 5.9GHz
77 **.wlan1.radio.bandwidth = 10MHz
78 **.wlan1.radio.receiver.snirThreshold = 4dB
79 **.wlan1.radio.receiver.sensitivity = -81dBm
80
81 ##### radio medium settings #####
82 Simulation.radioMedium.radioModeFilter = true #use this filter for increased performance -> ↪
    ↪ does not compute transmissions to receivers whose radio is turned off
83 Simulation.radioMedium.listeningFilter = true #second filter that may improve performance
84 Simulation.radioMedium.backgroundNoise.power = -110dBm
85 Simulation.radioMedium.mediumLimitCache.carrierFrequency = 5.9GHz
86 Simulation.radioMedium.propagationType = "ConstantSpeedPropagation"
87 Simulation.radioMedium.pathLossType = "FreeSpacePathLoss"
88 Simulation.radioMedium.obstacleLossType = ""
89
90 ##### mobility #####
91 **.mobility.constraintAreaMinX = 0m
92 **.mobility.constraintAreaMinY = 0m
93 **.mobility.constraintAreaMinZ = 0m
94 **.mobility.constraintAreaMaxX = 5000m
95 **.mobility.constraintAreaMaxY = 5000m
96 **.mobility.constraintAreaMaxZ = 0m
```

Listing C.1: Scenario Tiergarten: file listing: omnetpp/omnetpp.ini

## **Appendix D**

### **Package Additional Examples**

This package shows different sample applications for several application spectrum. Here you can download the `AdditionalExamples-18.0-sources.zip` file:

 <https://www.dcaiti.tu-berlin.de/research/simulation/download/>

The following table describes these applications.

<b>Applications from the "Additional Examples" package</b>	
HelloWorldApp	This is a simple Hello World application. It prints a hello world string in an interval.
Interconnect	This simple application demonstrates an interconnection between applications which are running on same units.
IntervalSampling	This simple application demonstrates a sampling in a specific interval. When the simulation starts it initialise with a random generated offset (e.g. 150 ms). Afterwards a fixed interval will be invoked (e.g. 1 second). For example there are the following time steps: t = {0 ms, 150 ms, 1150 ms, 2150 ms, ...}
MultithreadSampling	This simple application demonstrates a concurrency task. In a scenario may exist many vehicles (e.g. veh_0, veh_1, veh_2, ...). This application demonstrate the following behavior: veh_0, veh_1, veh_2, ... start a thread veh_0, veh_1, veh_2, ... do some logic parallel to each other veh_0, veh_1, veh_2, ... join the thread
MyMessage	This application shows how to create a message.
RandomSampling	This application demonstrates a sampling in a random interval. One random simulation point will be chosen when the initialisation starts. For example: t = {0 ms, 300 ms, 967 ms, 1487 ms, ...}
TestSumoTraciMsg	This sample shows the interaction between an application and the <a href="#">TraCI</a> interface of <a href="#">SUMO</a> .
UpdateVehicleInfo	This application reacts on a vehicle info update. Mostly a vehicle info will be triggered from a traffic simulator (e.g. SUMO).
UserTaggedValueRoundtrip	If the control of every byte is not needed, the <code>DefaultObjectSerialization</code> can be used. This class converts an object into a byte field and obversely.
VSimRTIMessage	This simple application shows how to create a <code>VSimRTI</code> -message and sends it to all simulators. This application could also react on <code>ApplicationSpecificMessages</code> .
VehicleConfiguration	This simple application demonstrates a configurable application.

Table D.1: Applications from the "Hello world package"

# Appendix E

## Configuration Schemata

Some of the required JSON configuration files are automatically validated by VSimRTI and, therefore, must follow a specific schema<sup>1</sup>. If you have problems with VSimRTI accepting your configuration files, please make sure they follow the respective schema:

### JSON Schema for cell2/cell2\_config.json

---

```
1 {
2   "$schema": "http://json-schema.org/schema#",
3   "type": "object",
4   "properties": {
5     "bandwidthMeasurementInterval": { "type": "number", "minimum": 1 },
6     "bandwidthMeasurements": {
7       "type": "array",
8       "items": [
9         {
10          "type": "object",
11          "properties": {
12            "fromRegion": { "type": "string" },
13            "toRegion": { "type": "string" },
14            "transmissionMode": {
15              "type": "string",
16              "enum": [ "UplinkUnicast", "DownlinkUnicast", "DownlinkMulticast" ]
17            }
18          }
19        },
20        {
21          "type": "object",
22          "properties": {
23            "fromRegion": { "type": "string" },
24            "toRegion": { "type": "string" },
25            "transmissionMode": {
26              "type": "string",
27              "enum": [ "UplinkUnicast", "DownlinkUnicast", "DownlinkMulticast" ]
28            }
29          }
30        }
31      ]
32     },
33     "required": [ "fromRegion", "toRegion", "transmissionMode" ]
34   }
35 }
36 }
```

---

Listing E.1: JSON Schema for cell2\_config.json

---

<sup>1</sup>The schemata listed here follow the specification at <http://json-schema.org/documentation.html>

## JSON Schema for cell2/network\_config.json

```

1  {
2    "$schema": "http://json-schema.org/schema#",
3    "type": "object",
4    "properties": {
5      "globalRegion": {
6        "type": "object",
7        "properties": {
8          "uplink": {
9            "type": "object",
10           "properties": {
11             "delay": { "$ref": "#/definitions/delay" },
12             "capacity": { "$ref": "#/definitions/capacity" }
13           },
14           "required": ["delay", "capacity"]
15         },
16         "downlink": {
17           "type": "object",
18           "properties": {
19             "unicast": { "$ref": "#/definitions/unicast" },
20             "multicast": { "$ref": "#/definitions/multicast" },
21             "capacity": { "$ref": "#/definitions/capacity" }
22           },
23           "required": ["unicast", "multicast", "capacity"]
24         }
25       }
26     },
27     "required": ["globalRegion"],
28     "definitions": {
29       "delay": {
30         "oneOf": [
31           {
32             "type": "object",
33             "properties": {
34               "type": { "type": "string",
35                 "enum": ["GammaRandomDelay", "GammaSpeedDelay"] },
36               "minDelay": { "type": "number",
37                 "minimum": 0 },
38               "expDelay": { "type": "number",
39                 "minimum": 0 },
40               "prpl": { "type": "number",
41                 "minimum": 0 },
42               "retries": { "type": "number",
43                 "minimum": 0 }
44             },
45             "required": ["type", "minDelay", "expDelay", "prpl"]
46           },
47           {
48             "type": "object",
49             "properties": {
50               "type": { "type": "string",
51                 "enum": ["ConstantDelay"] },
52               "delay": { "type": "number",
53                 "minimum": 0 },
54               "prpl": { "type": "number",
55                 "minimum": 0 },
56               "retries": { "type": "number",
57                 "minimum": 0 }
58             },
59             "required": ["type", "delay", "prpl"]
60           }
61         ],
62         "type": "object",
63         "properties": {
64           "type": "object",
65           "properties": {

```

```

66         "type" : {"type" : "string",
67                 "enum" : ["SimpleRandomDelay"]},
68         "steps" : {"type" : "number",
69                  "minimum" : 0},
70         "minDelay" : {"type" : "number",
71                     "minimum" : 0},
72         "maxDelay" : {"type" : "number",
73                      "minimum" : 0},
74         "prpl" : {"type" : "number",
75                  "minimum" : 0},
76         "retries" : {"type" : "number",
77                    "minimum" : 0}
78     },
79     "required" : ["type", "steps", "minDelay", "maxDelay", "prpl"]
80 }
81 ]
82 },
83 "unicast" : {
84     "type" : "object",
85     "properties" : {
86         "delay" : {"$ref" : "#/definitions/delay"}
87     },
88     "required" : ["delay"]
89 },
90 "multicast" : {
91     "type" : "object",
92     "properties" : {
93         "delay" : {"$ref" : "#/definitions/delay"},
94         "usableCapacity" : {"type" : "number",
95                             "minimum" : 0.0,
96                             "maximum" : 1.0}
97     },
98     "required" : ["delay", "usableCapacity"]
99 },
100 "capacity" : {
101     "oneOf" : [
102         {
103             "type": "integer",
104             "minimum": 0
105         },
106         {
107             "type": "string",
108             "pattern" : "^unlimited$"
109         }
110     ]
111 }
112 }
113 }

```

Listing E.2: JSON Schema for network\_config.json

## JSON Schema for cell2/regions\_config.json

```

1  {
2    "$schema": "http://json-schema.org/schema#",
3    "type": "object",
4    "properties": {
5      "regions": {
6        "type": "array",
7        "items": [
8          {
9            "type": "object",
10           "properties": {
11             "polygon": {
12               "type": "object",
13               "coordinates": [
14                 {"$ref": "#/definitions/geopoint"}
15               ],
16               "required": ["coordinates"]
17             },
18             "area": {
19               "type": "object",
20               "properties": {
21                 "nw": {"$ref": "#/definitions/geopoint"},
22                 "se": {"$ref": "#/definitions/geopoint"}
23               },
24               "required": ["nw", "se"]
25             },
26             "id": {"type": "string"},
27             "uplink": {
28               "type": "object",
29               "properties": {
30                 "delay": {"$ref": "#/definitions/delay"},
31                 "capacity": {"$ref": "#/definitions/capacity"}
32               },
33               "required": ["delay", "capacity"]
34             },
35             "downlink": {
36               "type": "object",
37               "properties": {
38                 "unicast": {"$ref": "#/definitions/unicast"},
39                 "multicast": {"$ref": "#/definitions/multicast"},
40                 "capacity": {"$ref": "#/definitions/capacity"}
41               },
42               "required": ["unicast", "multicast", "capacity"]
43             }
44           },
45           "anyOf": [
46             {"required": ["area", "id", "uplink", "downlink"]},
47             {"required": ["polygon", "id", "uplink", "downlink"]}
48           ]
49         }
50       ]
51     }
52   },
53
54   "definitions": {
55     "delay": {
56       "oneOf": [
57         {
58           "type": "object",
59           "properties": {
60             "type": {"type": "string",
61               "enum": ["GammaRandomDelay", "GammaSpeedDelay"]},
62             "minDelay": {"type": "number",
63               "minimum": 0,
64               "exclusiveMinimum": true},
65             "expDelay": {"type": "number",

```



```

66         "minimum" : 0,
67         "exclusiveMinimum" : true},
68     "prpl" : {"type" : "number",
69             "minimum" : 0},
70     "retries" : {"type" : "number",
71                "minimum" : 0}
72 },
73 "required" : ["type", "minDelay", "expDelay", "prpl"]
74 },
75 {
76     "type" : "object",
77     "properties" : {
78         "type" : {"type" : "string",
79                 "enum" : ["ConstantDelay"]},
80         "delay" : {"type" : "number",
81                  "minimum" : 0,
82                  "exclusiveMinimum" : true},
83         "prpl" : {"type" : "number",
84                  "minimum" : 0},
85         "retries" : {"type" : "number",
86                     "minimum" : 0}
87     },
88     "required" : ["type", "delay", "prpl"]
89 },
90 {
91     "type" : "object",
92     "properties" : {
93         "type" : {"type" : "string",
94                 "enum" : ["SimpleRandomDelay"]},
95         "steps" : {"type" : "number",
96                  "minimum" : 0},
97         "minDelay" : {"type" : "number",
98                     "minimum" : 0,
99                     "exclusiveMinimum" : true},
100        "maxDelay" : {"type" : "number",
101                     "minimum" : 0,
102                     "exclusiveMinimum" : true},
103        "prpl" : {"type" : "number",
104                "minimum" : 0},
105        "retries" : {"type" : "number",
106                    "minimum" : 0}
107     },
108     "required" : ["type", "steps", "minDelay", "maxDelay", "prpl"]
109 }
110 ]
111 },
112 "unicast" : {
113     "type" : "object",
114     "properties" : {
115         "delay" : {"$ref" : "#/definitions/delay"}
116     },
117     "required" : ["delay"]
118 },
119 "multicast" : {
120     "type" : "object",
121     "properties" : {
122         "delay" : {"$ref" : "#/definitions/delay"},
123         "usableCapacity" : {"type" : "number",
124                             "minimum" : 0.0,
125                             "maximum" : 1.0}
126     },
127     "required" : ["delay", "usableCapacity"]
128 },
129 "geopoint" : {
130     "type" : "object",
131     "properties" : {
132         "lon" : {"type" : "number"},

```

```
133         "lat" : {"type" : "number"}
134     },
135     "required" : ["lon", "lat"]
136 },
137 "capacity" : {
138     "oneOf": [
139         {
140             "type": "integer",
141             "minimum": 0
142         },
143         {
144             "type": "string",
145             "pattern" : "^unlimited$"
146         }
147     ]
148 }
149 }
150 }
```

Listing E.3: JSON Schema for regions\_config.json

## F References

- [1] BENZ, Thomas ; SCHÜNEMANN, Björn ; KERNCHEN, Ralf ; KILLAT, Moritz ; RICHTER, Andreas: A Comprehensive Simulation Tool Set for Cooperative Systems. In: **Advanced Microsystems for Automotive Applications 2010 : Smart Systems for Green Cars and Safe Mobility** (2010), May, S. 411–422. ISBN 978–3–642–12647–5
- [2] BISSMEYER, Norbert ; SCHÜNEMANN, Björn ; RADUSCH, Ilja ; SCHMIDT, Christian: Simulation of attacks and corresponding driver behavior in vehicular ad hoc networks with VSimRTI. In: **Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques**. ICST, Brussels, Belgium, Belgium : ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011 (SIMUTools '11). – ISBN 978–1–936968–00–8, 162–167
- [3] CHUANG, David ; SCHÜNEMANN, Björn ; RIECK, David ; RADUSCH, Ilja: GRIND: An Generic Interface for Coupling Power Grid Simulators with Traffic, Communication and Application Simulation Tools. In: **SIMUL 2013, The Fifth International Conference on Advances in System Simulation**, 2013. – ISBN 978–1–61208–308–7, S. 174–177
- [4] CODECÁ, Lara ; FRANK, Raphaël ; FAYE, Sébastien ; ENGEL, Thomas: Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation. In: **IEEE Intelligent Transportation Systems Magazine** 9 (2017), Nr. 2, S. 52–63
- [5] HÜBNER, Karl ; CRISTEA, Roland ; RULEWITZ, Stefan ; RADUSCH, Ilja ; SCHÜNEMANN, Björn: Implementation of Cognitive Driver Models in Microscopic Traffic Simulations. In: **Proceedings of the 9th EAI International Conference on Simulation Tools and Techniques**. ICST, Brussels, Belgium, Belgium : ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016 (SIMUTOOLS'16). – ISBN 978–1–63190–120–1, 104–111
- [6] HÜBNER, Karl ; SCHÜNEMANN, Björn ; RADUSCH, Ilja: Sophisticated Route Calculation Approaches for Microscopic Traffic Simulations. In: **Proceedings of the 8th International Conference on Simulation Tools and Techniques**. ICST, Brussels, Belgium, Belgium : ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2015 (SIMUTools '15). – ISBN 978–1–63190–079–2, 147–154
- [7] HÜBNER, Karl ; SCHÜNEMANN, Börn ; SCHILLING, Tom ; RADUSCH, Ilja: On assessing road safety aspects of a cycling router application. In: **2017 15th International Conference on ITS Telecommunications (ITST)**, 2017, S. 1–8
- [8] KATSAROS, Konstantinos ; KERNCHEN, Ralf ; DIANATI, Mehrdad ; RIECK, David ; ZINOVIU, Charalambos: Application of vehicular communications for improving the efficiency of traffic in urban areas. In: **Wireless Communications and Mobile Computing** 11 (2011), Nr. 12, S. 1657–1667

- [9] LOBACH, S. ; RADUSCH, I.: Integration of Communication Security into Advanced Simulation Environments for ITS. In: **Vehicular Technology Conference (VTC Fall), 2011 IEEE**, 2011. – ISSN 1090–3038, S. 1–6
- [10] NAUMANN, Nico ; SCHÜNEMANN, Björn ; RADUSCH, Ilja: VSimRTI - Simulation Runtime Infrastructure for V2X Communication Scenarios. In: **Proceedings of the 16th World Congress and Exhibition on Intelligent Transport Systems and Services (ITS Stockholm 2009)**, ITS Stockholm 2009, September 2009
- [11] NAUMANN, Nico ; SCHÜNEMANN, Björn ; RADUSCH, Ilja ; MEINEL, Christoph: Improving V2X Simulation Performance with Optimistic Synchronization. In: **Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific**, 2009. – ISBN 978–1–4244–5338–2, S. 52–57
- [12] PROTZMANN, R. ; MAHLER, K. ; OLTMANN, K. ; RADUSCH, I.: Extending the V2X simulation environment VSimRTI with advanced communication models. In: **ITS Telecommunications (ITST), 2012 12th International Conference on**, 2012, S. 683–688
- [13] PROTZMANN, Robert ; MASSOW, Kay ; RADUSCH, Ilja: An Evaluation Environment and Methodology for Automotive Media Streaming Applications. In: **Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2014 Eighth International Conference on IEEE**, 2014, S. 297–304
- [14] PROTZMANN, Robert ; MASSOW, Kay ; RADUSCH, Ilja: On performance estimation of prefetching algorithms for streaming content in automotive environments. In: **Wireless On-demand Network Systems and Services (WONS), 2014 11th Annual Conference on IEEE**, 2014, S. 147–147
- [15] PROTZMANN, Robert ; SCHÜNEMANN, Björn ; RADUSCH, Ilja: The influences of communication models on the simulated effectiveness of V2X applications. In: **Vehicular Networking Conference (VNC), 2010 IEEE**, 2010. – ISBN 978–1–4244–9526–9, S. 102–109
- [16] PROTZMANN, Robert ; SCHÜNEMANN, Björn ; RADUSCH, Ilja: The influences of communication models on the simulated effectiveness of V2X applications. In: **Communications Magazine, IEEE** 49 (2011), november, Nr. 11, S. 149–155. <http://dx.doi.org/10.1109/MCOM.2011.6069722>. – DOI 10.1109/MCOM.2011.6069722. – ISSN 0163–6804
- [17] PROTZMANN, Robert ; SCHÜNEMANN, Björn ; RADUSCH, Ilja: Effects of Random Number Generators on V2X Communication Simulation. In: TAN, Gary (Hrsg.) ; YEO, GeeKin (Hrsg.) ; TURNER, StephenJohn (Hrsg.) ; TEO, YongMeng (Hrsg.): **AsiaSim 2013 Bd. 402**, Springer Berlin Heidelberg, 2013 (Communications in Computer and Information Science). – ISBN 978–3–642–45036–5, 200–211
- [18] PROTZMANN, Robert ; SCHÜNEMANN, Björn ; RADUSCH, Ilja: On Site-Specific Propagation Models for the Evaluation of V2X Applications. In: **Communication Technologies for Vehicles (Nets4Cars-Fall), 2014 7th International Workshop on IEEE**, 2014, S. 35–39
- [19] PROTZMANN, Robert ; SCHÜNEMANN, Björn ; RADUSCH, Ilja: A Sensitive Metric for the Assessment of Vehicular Communication Applications. In: **Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on IEEE**, 2014, S. 697–703
- [20] QUECK, Tobias ; SCHÜNEMANN, Björn ; RADUSCH, Ilja: Runtime Infrastructure for Simulating Vehicle-2-X Communication Scenarios. In: **VANET '08: Proceedings of the fifth ACM international workshop on Vehicular Inter-NETworking**. New York, NY, USA : ACM, 2008. – ISBN 978–1–60558–191–0, S. 78–79

- [21] QUECK, Tobias ; SCHÜNEMANN, Björn ; RADUSCH, Ilja ; MEINEL, Christoph: Realistic Simulation of V2X Communication Scenarios. In: **APSCC '08: Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference**. Washington, DC, USA : IEEE Computer Society, 2008. – ISBN 978–0–7695–3473–2, S. 1623–1627
- [22] RIECK, David ; SCHÜNEMANN, Björn ; RADUSCH, Ilja ; MEINEL, Christoph: Efficient traffic simulator coupling in a distributed V2X simulation environment. In: **SIMUTools '10: Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques**. ICST, Brussels, Belgium, Belgium : ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010. – ISBN 978–963–9799–87–5, S. 1–9
- [23] SCHÜNEMANN, Björn: V2X simulation runtime infrastructure VSimRTI: An assessment tool to design smart traffic management systems. In: **Computer Networks** 55 (2011), October, 3189–3198. <http://dx.doi.org/http://dx.doi.org/10.1016/j.comnet.2011.05.005>. – DOI <http://dx.doi.org/10.1016/j.comnet.2011.05.005>. – ISSN 1389–1286
- [24] SCHÜNEMANN, Björn ; MASSOW, Kay ; RADUSCH, Ilja: A Novel Approach for Realistic Emulation of Vehicle-2-X Communication Applications. In: **Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE**, 2008. – ISSN 1550–2252, S. 2709–2713
- [25] SCHÜNEMANN, Björn ; MASSOW, Kay ; RADUSCH, Ilja: Realistic Simulation of Vehicular Communication and Vehicle-2-X Applications. In: **Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops**. ICST, Brussels, Belgium, Belgium : ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008. – ISBN 978–963–9799–20–2, S. 1–9
- [26] SCHÜNEMANN, Björn ; RIECK, David ; RADUSCH, Ilja: Performance and scalability analyses of federation-based V2X simulation systems. In: **Ad Hoc Networking Workshop (Med-Hoc-Net), 2012 The 11th Annual Mediterranean**, 2012. – ISBN 978–1–4673–2038–2, S. 119–126
- [27] SCHÜNEMANN, Björn ; WEDEL, Jan W. ; RADUSCH, Ilja: V2X-Based Traffic Congestion Recognition and Avoidance. In: **Tamkang Journal of Science and Engineering** 13 (2010), March, Nr. 1, 63–70. <http://www2.tku.edu.tw/~tkjse/13-1/catalogy.htm>. – ISSN 1560–6686
- [28] WEDEL, Jan W. ; SCHÜNEMANN, Björn ; RADUSCH, Ilja: V2X-Based Traffic Congestion Recognition and Avoidance. In: **Parallel Architectures, Algorithms, and Networks, International Symposium on 0** (2009), S. 637–641. <http://dx.doi.org/http://doi.ieeecomputersociety.org/10.1109/I-SPAN.2009.71>. – DOI <http://doi.ieeecomputersociety.org/10.1109/I-SPAN.2009.71>. ISBN 978–0–7695–3908–9