

# MATSim Traffic Signals Contrib – User Guide

Development Version  
last updated May 6, 2014

Theresa Thunig Dominik Grether



---

## Note

The material found in this document was originally maintained as a web page. The html was automatically converted to latex, and incrementally cleaned up since then.

As a result, leftovers from those web pages can still be found in the document, including lots of links to web pages which could be replaced by internal links in the document, and links to web pages where the material should be inlined into the document.

---

Originally generated 2014-05-06  
from [matsim.org/node/732](http://matsim.org/node/732)  
but changed since then.

# Contents

---

<b>1</b>	<b>Tutorial</b>	<b>3</b>
1.1	Introduction . . . . .	3
1.1.1	Terminology . . . . .	3
1.1.2	Overview . . . . .	3
1.2	Configuration . . . . .	4
1.3	Input . . . . .	4
1.3.1	Intergreens . . . . .	4
1.4	Generating Input Data – Example . . . . .	4
1.4.1	Links . . . . .	6
	<b>Appendices</b>	<b>7</b>

# Tutorial

---

This tutorial is tested with MATSim 0.4.0 or higher.

## 1.1 Introduction

The MATSim contrib for traffic lights is able to simulate traffic signals microscopically. For fixed-time control a default implementation is provided. No programming is required to simulate fixed-time traffic signal control. Traffic-responsive signal control need custom implementation but may use the provided infrastructure.

### 1.1.1 Terminology

To help you to understand xml formats, configuration, and code better, here some translations...

#### **Real world terms**

Traffic light, traffic signal, i.e. a physical box standing somewhere on the transport network indicating driving a

Logical group of traffic lights, all lights display same color at the same time

Algorithm or control scheme that determines which colors are displayed by the different Signal Groups, e.g. fixed

Collection of Signal Groups that is controlled by the same Signal Control

### 1.1.2 Overview

#### Intergreens

MATSim's queue model implementation does not contain any collision detection. That is, a signal control can switch several conflicting approaches to green at the same time. The simulation does not report any error.

To prevent such faulty behavior, the intergreens component can be switched on. An xml file defines for each signal system the intergreen times of the signal groups, i.e. the minimal time period between the ending of the one and the beginning of the other signal groups green phase. This information is then used to validate the signal control while the simulation is executed and may result in a warning or an exception.

This is especially important if you implement a custom, traffic-responsive signal control strategy.

## 1.2 Configuration

If input files for signal systems are already available, simulation of traffic lights can be enabled via MATSim config options:

- Set the parameter `useSignalsystems` to `true` in the config module `scenario`
- Set at least three input file names in the config module `signalsystems`:
  - parameter name: `signalsystems` value: path to a file in the `signalSystems_v2.0.xsd` file format
  - parameter name: `signalgroups` value: path to a file in the `signalGroups_v2.0.xsd` file format
  - parameter name: `signalcontrol` value: path to a file in the `signalControl_v2.0.xsd` file format
  - parameter name: `ambertimes` (optional) value: path to a file in the `amberTimes_v1.0.xsd` file format
  - parameter name: `useIntergreenTimes` (optional) value: `true` or `false`
  - parameter name: `intergreentimes` (optional) value: path to a file in the `intergreenTimes_v1.0.xsd` file format
  - parameter name: `actionOnIntergreenViolation` (optional) value: `warn` or `exception`

## 1.3 Input Files – Examples

### 1.3.1 Intergreens

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<intergreenTimes xsi:schemaLocation="http://www.matsim.org/files/dtd http://www.
matsim.org/files/dtd/intergreenTimes_v1.0.xsd" xmlns:xsi="http://www.w3.org
/2001/XMLSchema-instance" xmlns="http://www.matsim.org/files/dtd">
  <signalSystem refId="23">
    <endingSignalGroup refId="1">
      <beginningSignalGroup refId="2" timeSeconds="5" />
      <beginningSignalGroup refId="3" timeSeconds="3" />
      <beginningSignalGroup refId="4" timeSeconds="3" />
    </endingSignalGroup>
  </signalSystem>

  <signalSystem refId="42">
    <endingSignalGroup refId="1">
      <beginningSignalGroup refId="2" timeSeconds="5" />
    </endingSignalGroup>
    <endingSignalGroup refId="2">
      <beginningSignalGroup refId="1" timeSeconds="3" />
    </endingSignalGroup>
  </signalSystem>
</intergreenTimes>
```

## 1.4 Generating Input Data – Example

There is a small example in order to help you getting started, the class is `CreateSimpleTrafficSignalScenario`.

The example creates the network attached to this page as pdf ([http://matsim.org/uploads/2012\\_ks\\_2010\\_network.pdf](http://matsim.org/uploads/2012_ks_2010_network.pdf)). The numbers show the link ids. Traffic lights to be created are shown as small icons.

The following highlights important lines of code method by method

## Code

Input files for the examples can be found in the folder `examples/tutorial/unsupported/example90TrafficLights` of the current MATSim release / head.

Example code can be found in the folder `src/main/java` in the package `tutorial.unsupported.example90TrafficLig`

### `CreateSimpleTrafficSignalScenario.run()`

This method sets up the config and writes the signals that are created in submethods to file.

As optional module, the signal systems module has to be switched on in the scenario first:

```
config.scenario().setUseSignalSystems(true);
```

Then, qsim as mobility simulation should be used:

```
config.controller().setMobsim("qsim");
```

```
config.addQSimConfigGroup(new QSimConfigGroup());
```

After the scenario is loaded, the top-level container for all signal related data is retrieved by:

```
SignalsData signalsData =
scenario.getScenarioElement(SignalsData.class);
```

### `CreateSimpleTrafficSignalScenario.createSignalSystemsAndGroups(..)`

This method creates the physics, i.e. locations of signals. Furthermore SignalGroups for the Signals are created.

First a SignalSystem has to be created and explicitly added to the container:

```
SignalSystemData sys =
systems.getFactory().createSignalSystemData(scenario.createId("3"));
systems.addSignalSystemData(sys);
```

Then on link 23 a Signal with Id 1 can be created:

```
SignalData signal =
systems.getFactory().createSignalData(scenario.createId("1"));
sys.addSignalData(signal);
signal.setLinkId(scenario.createId("23"));
```

The Signal must be added to the SignalSystem after creation.

This is continued until all Signals of the SignalSystem are created.

We then need some SignalGroup for the Signals. In this example each Signal has its own group. That is done by calling:

```
SignalUtils.createAndAddSignalGroups4Signals(groups, sys);
```

### `CreateSimpleTrafficSignalScenario.createSignalControl(..)`

This method adds a fixed-time traffic signal control on top of the SignalGroups created by the last method. The SignalGroups control the hardware, i.e. the Signals. MATSim Events are created for SignalGroups.

Each SignalSystem is equipped with a fixed-time control. Each SignalSystem is setup the same way: A cycle time of 120 sec is used. Each direction gets green for second 0 to 55 within the cycle. Offsets for green waves are set to 0 seconds.

The Code can be read as follows:

Create a fixed-time control and add it to the container:

```
SignalSystemControllerData controller =
control.getFactory().createSignalSystemControllerData(id); control.addSignalSystemC
```

Then, create a fixed-time control plan (a plan can be disabled or changed at a certain time of day) and add it to the container:

```
SignalPlanData plan =
control.getFactory().createSignalPlanData(scenario.createId("1")); controller.addSi
```

Each fixed-time control plan has specific attributes for cycle and synchronization offset:

```
plan.setCycleTime(cycle); plan.setOffset(0);
```

Create specific green times (onset = switch to green, offset = switch to amber/red) for all SignalGroups and add them to the signal control plan via

```
SignalGroupSettingsData settings1 =
control.getFactory().createSignalGroupSettingsData(scenario.createId("1")); plan.ad
```

## Visualization

If you want to visualize the created scenario try to get and run the OTFVis contribution to MATSim (use MATSim head or nightly build instead of 0.4.1), see OTFVis(this contribution is unsupported, please submit patches instead of mails).

If OTFVis runs successfully, you may be able to run

```
tutorial.unsupported.example90TrafficLights.VisSimpleTrafficSignalScenario
```

within an up to date (¿ 04.12.2012) checkout within eclipse.

Another option is to write the scenario to file via the example code and then start OTFVis with the created config file.

### 1.4.1 Links

- Technical Documentation (JavaDoc)
- Module Documentation and Publications
- OTFVis



# Appendices

---