railML® 3.1 Tutorial
Simple Example Step-by-Step
Part 1: Infrastructure

Revision History

Version	Date	Description	Author
1.0	22.12.2017	First final draft version	Christian Rahmig
1.1	22.01.2018	Incorporating corrections following the railML 3.1 Workshop on January 9-10, 2018	Christian Rahmig
1.2	12.11.2018	Updated version adapted to railML 3.1 (Release Candidate)	Christian Rahmig

Table of Content

1 Purpose of this document

The release of the new major version of the railway data exchange format, railML 3.1, is based on a completely new topology model. This new topology approach is derived from the RailTopoModel [6] that is being developed by the UIC led RailTopoModel Expert Group. In order to open up railML for a big variety of different use cases and to provide a very powerful data model for infrastructure managers as well as for railway undertakings, IT service providers and other stakeholders, the RailTopoModel basis is quite generic and complex. Consequently, railML 3.1 schema is very different from previous versions and understanding the new model requires more documentation.

Existing railML documentation comprises

- the railML Wiki [5] for information about schema application,
- the railML forum [3] for discussion with the railML user and developer community,
- the railML Trac ticket system [4] for recording and tracking all bugs and model enhancements, and
- the railML schema documentation for syntax explanations.

What has been missing so far is a "Hello World" example for implementation of railML v3. The Simple Example Tutorial shall fill this gap:

<Hello-railML3/>

This document shall help the potential users to understand the model by providing a step-by-step implementation of a simple example. In the end, the reader shall be able to decode all the lines of the railML 3.1 Simple Example file, so that he/she can adapt this knowledge to create railML files himself/herself[1].

[1] This document does not distinguish between male and female readers. Whenever one sex is used in the text, the other one is meant as well.

2 How to read the Tutorial

2.1 Prior Knowledge and related literature

This book is about writing and reading railML example files. Before learning railML, a user should possess basic knowledge about XML as well as railway infrastructure in general. For an introduction to these topics, please see for example the following resources:

XML

- Tutorialspoint [8]
- W3Schools [11]
- Webcurator [12]

Introduction into railway infrastructure

- The Railway Technical Website [7]

2.2 Syntax Guide

In the text, railML `<elements>` are put into XML specific brackets `<>`. railML `@attributes` can be recognized via the `@` symbol before the attribute name. Attribute `“values”` are framed by quotation marks “”. All railML syntax objects (`<element>@attribute=”value”`) are written in typewriter style in grey color.

Source code examples are written in grey boxes:

```
<railML sourcecode="example">
  ...
</railML>
```

In order to clarify that this document is not the legal reference for the railML 3.1 schema, but only a learning tutorial, you will find the following orange notice box at the beginning of each section or chapter:

! **The content of this document is a learning tutorial**. Please check current railML 3.1 syntax documentation at railML.org for latest binding information on elements, attributes and use.

At the end of each chapter or section you will find a green titled box that summarizes the main aspects and results:

What you may have learned
• Lessons • Learned

3 Step-by-step implementation

3.1 The railML File Skeleton

! **The content of this document is a learning tutorial**. Please check current railML 3.1 syntax documentation at railML.org for latest binding information on elements, attributes and use.

Every railML file has the same framework structure based on XML syntax. It looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<railML xmlns="https://www.railml.org/schemas/3.1"
        xmlns:gml="http://www.opengis.net/gml/3.2/"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="https://www.railml.org/schemas/3.1
https://www.railml.org/schemas/railml-3.1/railml3.xsd"
        version="3.1">

</railML>
```

The attribute `@version` defines the railML version, which is used in the railML file. In our case, the simple example is written in railML 3.1 that is the first official release of railML v3.

The simple example described in this document contains mainly infrastructure information. Thus, the infrastructure root element is added to the framework. This element acts like a container for all the different infrastructure elements and information:

```
<?xml version="1.0" encoding="UTF-8"?>
<railML xmlns="https://www.railml.org/schemas/3.1"
        xmlns:gml="http://www.opengis.net/gml/3.2/"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="https://www.railml.org/schemas/3.1
https://www.railml.org/schemas/railml-3.1/railml3.xsd"
        version="3.1">

  <infrastructure>
  </infrastructure>

</railML>
```

Additionally, the example also contains information that is relevant for infrastructure and other domains as well, e.g. positioning systems. These “cross domain” or “common” information shall be stored in a container `<common>` that is also added to the framework:

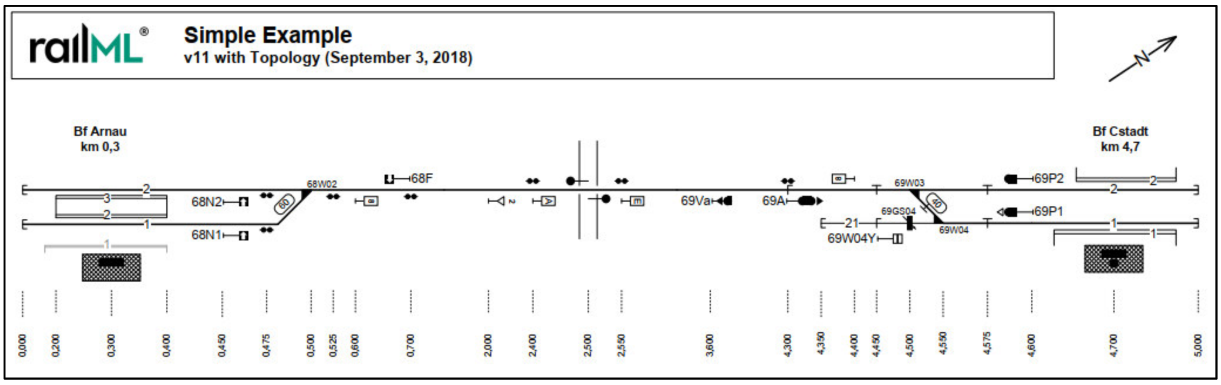
```
<?xml version="1.0" encoding="UTF-8"?>
<railML xmlns="https://www.railml.org/schemas/3.1"
        xmlns:gml="http://www.opengis.net/gml/3.2/"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="https://www.railml.org/schemas/3.1
https://www.railml.org/schemas/railml-3.1/railml3.xsd"
        version="3.1">

  <common>
  </common>
  <infrastructure>
  </infrastructure>

</railML>
```

In order to provide background information about the content of the railML file, it is strongly recommended to include a `<metadata>` element with children elements that define the source of the railML file, a generation timestamp and copyright information.

```
<?xml version="1.0" encoding="UTF-8"?>
<railML xmlns="https://www.railml.org/schemas/3.1"
        xmlns:gml="http://www.opengis.net/gml/3.2/"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="https://www.railml.org/schemas/3.1
https://www.railml.org/schemas/railml-3.1/railml3.xsd"
        version="3.1">

  <metadata>
    <dc:format>3.1</dc:format>
    <dc:identifier>2</dc:identifier>
    <dc:source>railML.org</dc:source>
    <dc:title xml:lang="en">Simple Example v11 railML 3.1</dc:title>
    <dc:language>en</dc:language>
    <dc:date>2018-11-05T14:31:00+01:00</dc:date>
    <dc:creator xml:lang="de">Christian Rahmig</dc:creator>
    <dc:rights>Copyright (c) railML.org e.V. Dresden/Germany. All Rights Reserved.
      For more information, visit https://www.railml.org/licence
      Content of this file: railML 3.1 Simple Example</dc:rights>
  </metadata>

  <common>
  </common>

  <infrastructure>
  </infrastructure>

</railML>
```

In the following sections we are going to have a closer look on the different components of the infrastructure starting with the most important one: railway track network topology.

But before we dive into the details of modelling let’s have a look at the Simple Example in all:



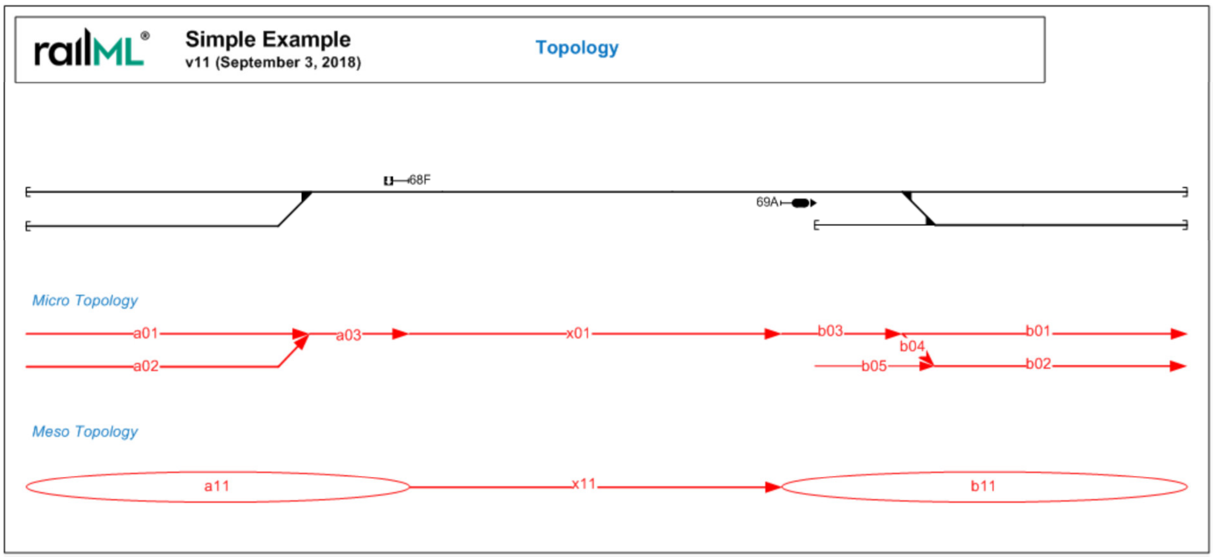
The simple example is completely fictitious. It contains a five kilometre long single track railway line between two stations (operational points). One station is equipped with ETCS signals and axle counters while the other station is equipped with conventional signalling infrastructure and track circuits. The railway line includes one level crossing and a related reduced speed section.

What you may have learned
• railML is an XML dialect and therefore follows the basic syntax pattern of XML. • railML references elements of DublinCore data exchange format in order to model file meta data. • For traceability it is strongly recommended to provide elementary meta information for every file including source and date of the data. • The railML data are structured in modules based on the railML schemes: infrastructure, timetable, rollingstock, interlocking and common. • railML Simple Example is a railML 3.1 example file that describes an artificial five kilometer long one-track railway line between two stations.

3.2 Topology

! **The content of this document is a learning tutorial**. Please check current railML 3.1 syntax documentation at railML.org for latest binding information on elements, attributes and use.

The structure of a railway track network is described by its topology. Topology defines how the tracks are connected with each other at switches and crossings and how the railway stations or operational points are connected with each other via railway lines. Since all trains run on rails (tracks) and all relevant infrastructure elements are somehow situated in a defined relation on or next to the track, topology provides the basis for location of static and dynamic elements in railways. Further, topology defines the possible ways of movement within the railway track network. The following figure shows how the railway track network is reduced to a microscopic and a more abstract mesoscopic topology model:



3.2.1 Microscopic Topology

The **micro topology** contains nine `<netElement>` objects, which are connected with each other by eleven `<netRelation>` objects. Altogether, the microscopic topology level network contains 20 `<networkResource>` objects. The railML file shall look like this:

```
<infrastructure id="is_01">
  <topology>
    <netElements>
      <netElement id="ne_a01">
        <relation ref="nr_a01a02"/>
        <relation ref="nr_a01a03"/>
      </netElement>
      <netElement id="ne_a02">
        <relation ref="nr_a01a02"/>
        <relation ref="nr_a02a03"/>
      </netElement>
      <netElement id="ne_a03">
        <relation ref="nr_a01a03"/>
        <relation ref="nr_a02a03"/>
        <relation ref="nr_a03x01"/>
      </netElement>
```

```
        <netElement id="ne_b01">
          <relation ref="nr_b01b03"/>
          <relation ref="nr_b01b04"/>
        </netElement>
        <netElement id="ne_b02">
          <relation ref="nr_b02b04"/>
          <relation ref="nr_b02b05"/>
        </netElement>
        <netElement id="ne_b03">
          <relation ref="nr_b01b03"/>
          <relation ref="nr_b03b04"/>
          <relation ref="nr_x01b03"/>
        </netElement>
        <netElement id="ne_b04">
          <relation ref="nr_b01b04"/>
          <relation ref="nr_b02b04"/>
          <relation ref="nr_b03b04"/>
          <relation ref="nr_b04b05"/>
        </netElement>
        <netElement id="ne_b05">
          <relation ref="nr_b02b05"/>
          <relation ref="nr_b04b05"/>
        </netElement>
        <netElement id="ne_x01">
          <relation ref="nr_a03x01"/>
          <relation ref="nr_x01b03"/>
        </netElement>
      </netElements>

      <netRelations>
        <netRelation id="nr_a01a02" positionOnA="1" positionOnB="1" navigability="None">
          <elementA ref="ne_a01"/>
          <elementB ref="ne_a02"/>
        </netRelation>
        <netRelation id="nr_a01a03" positionOnA="1" positionOnB="0" navigability="Both">
          <elementA ref="ne_a01"/>
          <elementB ref="ne_a03"/>
        </netRelation>
        <netRelation id="nr_a02a03" positionOnA="1" positionOnB="0" navigability="Both">
          <elementA ref="ne_a02"/>
          <elementB ref="ne_a03"/>
        </netRelation>
        <netRelation id="nr_b01b03" positionOnA="0" positionOnB="1" navigability="Both">
          <elementA ref="ne_b01"/>
          <elementB ref="ne_b03"/>
        </netRelation>
        <netRelation id="nr_b01b04" positionOnA="0" positionOnB="0" navigability="None">
          <elementA ref="ne_b01"/>
          <elementB ref="ne_b04"/>
        </netRelation>
        <netRelation id="nr_b02b04" positionOnA="0" positionOnB="1" navigability="Both">
          <elementA ref="ne_b02"/>
          <elementB ref="ne_b04"/>
        </netRelation>
        <netRelation id="nr_b02b05" positionOnA="0" positionOnB="1" navigability="Both">
          <elementA ref="ne_b02"/>
          <elementB ref="ne_b05"/>
        </netRelation>
        <netRelation id="nr_b03b04" positionOnA="1" positionOnB="0" navigability="Both">
          <elementA ref="ne_b03"/>
          <elementB ref="ne_b04"/>
        </netRelation>
        <netRelation id="nr_b04b05" positionOnA="1" positionOnB="1" navigability="None">
          <elementA ref="ne_b04"/>
          <elementB ref="ne_b05"/>
        </netRelation>
        <netRelation id="nr_a03x01" positionOnA="1" positionOnB="0" navigability="Both">
          <elementA ref="ne_a03"/>
          <elementB ref="ne_x01"/>
        </netRelation>
        <netRelation id="nr_x01b03" positionOnA="1" positionOnB="0" navigability="Both">
          <elementA ref="ne_x01"/>
          <elementB ref="ne_b03"/>
        </netRelation>
      </netRelations>

      <networks>
        <network id="nw01">
          <level id="lv0" descriptionLevel="Micro">
            <networkResource ref="ne_a01"/>
            <networkResource ref="ne_a02"/>
            <networkResource ref="ne_a03"/>
            <networkResource ref="ne_b01"/>
            <networkResource ref="ne_b02"/>
            <networkResource ref="ne_b03"/>
            <networkResource ref="ne_b04"/>
            <networkResource ref="ne_b05"/>
            <networkResource ref="ne_x01"/>
            <networkResource ref="nr_a01a02"/>
            <networkResource ref="nr_a01a03"/>
            <networkResource ref="nr_a02a03"/>
            <networkResource ref="nr_b01b03"/>
            <networkResource ref="nr_b01b04"/>
            <networkResource ref="nr_b02b04"/>
            <networkResource ref="nr_b02b05"/>
            <networkResource ref="nr_b03b04"/>
            <networkResource ref="nr_b04b05"/>
            <networkResource ref="nr_a03x01"/>
            <networkResource ref="nr_x01b03"/>
          </level>
        </network>
      </networks>
    </topology>
  </infrastructure>
```

3.2.2 Mesoscopic Topology

The **mesoscopic / macroscopic topology** implements the line view of the railway network. All tracks within an operational point are grouped together. The same is done for all the tracks that belong to one section of line between the operational points. This means for our simple example: the macro topology contains three `<netElement>` objects, which are connected with each other by two `<netRelation>` objects. Altogether, the mesoscopic topology level network contains five `<networkResource>` objects. The aggregation between microscopic and mesoscopic topology level is realized via element collections. In particular, the mesoscopic `<netElement>` object references the microscopic `<netElement>` objects as element parts. The railML syntax for resulting mesoscopic topology shall look like this:

```
  <infrastructure id="is_01">
    <topology>
      <netElements>
        ...
        <netElement id="ne_a11">
          <relation ref="nr_a11x11"/>
          <elementCollectionUnordered id="ne_a11_ecu01">
            <elementPart ref="ne_a01"/>
            <elementPart ref="ne_a02"/>
            <elementPart ref="ne_a03"/>
          </elementCollectionUnordered>
        </netElement>
        <netElement id="ne_b11">
          <relation ref="nr_x11b11"/>
          <elementCollectionUnordered id="ne_b11_ecu01">
            <elementPart ref="ne_b01"/>
            <elementPart ref="ne_b02"/>
            <elementPart ref="ne_b03"/>
            <elementPart ref="ne_b04"/>
            <elementPart ref="ne_b05"/>
          </elementCollectionUnordered>
        </netElement>
        <netElement id="ne_x11">
          <relation ref="nr_a11x11"/>
          <relation ref="nr_x11b11"/>
          <elementCollectionOrdered id="ne_x11_ecu01">
            <elementPart ref="ne_x01"/>
          </elementCollectionOrdered>
        </netElement>
      </netElements>

      <netRelations>
        ...
        <netRelation id="nr_a11x11" positionOnA="0" positionOnB="0" navigability="Both">
          <elementA ref="ne_a11"/>
          <elementB ref="ne_x11"/>
        </netRelation>
        <netRelation id="nr_x11b11" positionOnA="1" positionOnB="0" navigability="Both">
          <elementA ref="ne_x11"/>
          <elementB ref="ne_b11"/>
        </netRelation>
      </netRelations>

      <networks>
        <network id="nw01">
          ...
          <level id="lv1" descriptionLevel="Meso">
            <networkResource ref="ne_a11"/>
            <networkResource ref="ne_b11"/>
            <networkResource ref="ne_x11"/>
            <networkResource ref="nr_a11x11"/>
            <networkResource ref="nr_x11b11"/>
          </level>
        </network>
      </networks>
    </topology>
  </infrastructure>
```

Having finished modelling the railway track network topology, the next step is to define a linear referencing system that can be used to provide a spatial dimension to the topology elements.

What you may have learned
• Railway topology defines how tracks are connected with each other, and how railway lines connect operational points, thus providing the basis for location of both static and dynamic railway elements. It also implicitly defines the movement possibilities for railway vehicles. • A topology model may be illustrated on a microscopic, mesoscopic or macroscopic level. On the mesoscopic and macroscopic levels, all tracks and other elements belonging to the same operational point are grouped together, while they are individually shown on the microscopic level.

- All topology aspects are defined under the umbrella element `<topology>`, with its `<netElement>`, `<netRelation>` and `<network>` elements.
- A mesoscopic `<netElement>` objects may reference microscopic `<netElement>` objects, thus connecting the two levels.

3.3 Positioning Systems

! **The content of this document is a learning tutorial**. Please check current railML 3.1 syntax documentation at railML.org for latest binding information on elements, attributes and use.

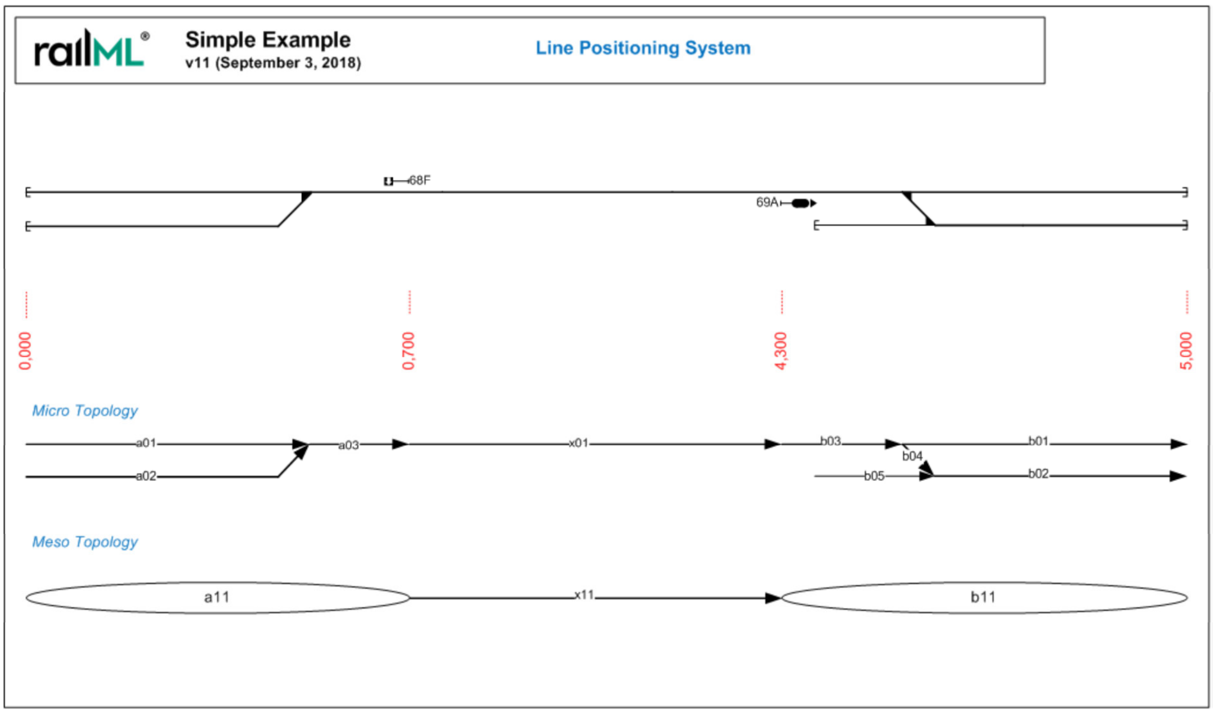
A positioning system provides a coordinate reference system for spatial identification of objects. Almost all components of railway infrastructure have such a spatial identification, either in a global context with their GPS coordinates or in a railway line context with a relative position along a railway track or line. Apart from infrastructure elements, railway vehicles and trains may have a (dynamic) spatial identification, too. Although not being modelled with this first version of railML v3, the location of these moving objects is defined on the basis of the same coordinate reference systems. Thus, positioning systems are not specific for infrastructure and consequently shall be modelled in the `<common>` domain:

```
<common id="co_01">
  <positioning>
    ...
  </positioning>
</common>
```

3.3.1 Associated Positioning System

The railway network topology defined by the `<netElement>` objects need to be linked with a positioning reference system. In particular, every `<netElement>` object must have a child element `<associatedPositioningSystem>`. It is used to locate infrastructure elements within the topology network.

For the mesoscopic topology, it is recommended to use the classic railway line positioning system consisting of a line identifier and a kilometre value. The Simple Example railway line has a length of five kilometres. As a first step, we define the mileages for the borders of the operational points. The location of the entrance signal has been chosen to define these borders. Consequently, the operational point defined by the mesoscopic non-linear `<netElement>` “a11” spans from mileage 0.000 kilometre to mileage 0.700 kilometre. At the other end of the line, the operational point defined by the mesoscopic non-linear `<netElement>` “b11” covers the mileage 4.300 kilometre to 5.000 kilometre. The railway line between the two operational points is modelled as linear `<netElement>` “x11” starting in kilometre 0.700 and ending in kilometre 4.300. The following figure depicts the relevant kilometre values.



The railML implementation foresees a separation between the definition of the coordinate reference system and the coordinates themselves. At first, we define the linear coordinate reference system defined by the classic railway line positioning system: The railway line has the name “6869” and it ranges from kilometre 0.0 to kilometre 5.0.

```
  <common id="co_01">
    ...
    <positioning>
      <linearPositioningSystems>
        <linearPositioningSystem id="lps01" units="m" startMeasure="0.0"
endMeasure="5000.0" linearReferencingMethod="absolute">
          <name name="6869"/>
          <isValid from="2018-01-01" to="2018-12-31"/>
        </linearPositioningSystem>
      </linearPositioningSystems>
    </positioning>
  </common>
```

At second, we list the coordinates linked with the borders of our mesoscopic topology `<netElement>` objects by using the child element `<associatedPositioningSystem>`. Each coordinate then references the linear positioning system that has been defined before.

```
  <infrastructure id="is_01">
    <topology>
      <netElements>
        ...
        <netElement id="ne_a11">
          <relation ref="nr_a11x11"/>
          <elementCollectionUnordered id="ne_a11_ecu01">
            <elementPart ref="ne_a01"/>
            <elementPart ref="ne_a02"/>
            <elementPart ref="ne_a03"/>
          </elementCollectionUnordered>
          <associatedPositioningSystem id="ne_a11_aps01">
            <intrinsicCoordinate id="ne_a11_aps01_ic01" intrinsicCoord="0">
              <linearCoordinate measure="0.0" positioningSystemRef="lps01"/>
              <linearCoordinate measure="700.0" positioningSystemRef="lps01"/>
            </intrinsicCoordinate>
            <isValid from="2018-01-01" to="2018-12-31" />
         </associatedPositioningSystem>
        </netElement>
        <netElement id="ne_b11">
          <relation ref="nr_x11b11"/>
          <elementCollectionUnordered id="ne_b11_ecu01">
            <elementPart ref="ne_b01"/>
            <elementPart ref="ne_b02"/>
            <elementPart ref="ne_b03"/>
            <elementPart ref="ne_b04"/>
            <elementPart ref="ne_b05"/>
          </elementCollectionUnordered>
          <associatedPositioningSystem id="ne_b11_aps01">
            <intrinsicCoordinate id="ne_b11_aps01_ic01" intrinsicCoord="0">
              <linearCoordinate measure="4300.0" positioningSystemRef="lps01"/>
              <linearCoordinate measure="5000.0" positioningSystemRef="lps01"/>
            </intrinsicCoordinate>
            <isValid from="2018-01-01" to="2018-12-31" />
          </associatedPositioningSystem>
        </netElement>
        <netElement id="ne_x11">
          <relation ref="nr_a11x11"/>
          <relation ref="nr_x11b11"/>
          <elementCollectionUnordered id="ne_x11_ecu01">
            <elementPart ref="ne_x01"/>
          </elementCollectionUnordered>
          <associatedPositioningSystem id="ne_x11_aps01" validFrom="2017-01-01"
validTo="2017-12-31">
            <intrinsicCoordinate id="ne_x11_aps01_ic01" intrinsicCoord="0">
              <linearCoordinate measure="0.700" positioningSystemRef="lps01"/>
            </intrinsicCoordinate>
            <intrinsicCoordinate id="ne_x11_aps01_ic02" intrinsicCoord="1">
              <linearCoordinate measure="4.300" positioningSystemRef="lps01"/>
            </intrinsicCoordinate>
          </associatedPositioningSystem>
        </netElement>
      </netElements>
      ...
    </topology>
  </infrastructure>
```

Please note:

1. The `<linearPositioningSystem>` like any other positioning system is defined in the `<common>` domain, and referenced from elements in `<infrastructure>` domain
2. The `<associatedPositioningSystem>` is only connected to `<netElement>` objects. The connecting `<netRelation>` objects are dimensionless and therefore have neither a length nor coordinates.
3. A `<netElement>` may have more than one `<associatedPositioningSystem>`. They are either types of linear or geometric positioning systems and need to be defined under the `<positioning>` element in the `<common>` container.

3.3.2 Linear Positioning System

A linear positioning system is a positioning system where a “line of reference” together with a single number allows defining a location within a railway network (cp. [10]). The “line of reference” is represented with a line or track number together with a start mileage and an end mileage.

In the Simple Example the line reference system providing the kilometre values along the line is modelled as linear positioning system.

3.3.3 Geometric Positioning System

A schematic positioning system defines a schematic, geographic or geodetic coordinate reference system, which is used to position <netElement> instances or other railway infrastructure element instances (cp. [10]). A geometric positioning system can often be defined via its EPSG code (cf. [8]).

There are no geometric coordinate systems being used within the Simple Example, because the described line from Bf Arnau to Bf Cstadt is purely artificial and not located anywhere on the earth surface.

3.3.4 Length of Topology elements

Extending the pure topology concept of RailTopoModel, railML 3.1 allows to define `<netElement>` elements with a certain length. This extension is based on the assumption that railML topology is used for describing either a track or a line network. In any case, it describes something that has a spatial dimension. Consequently, linear `<netElement>` instances shall have a length of their physical implementation.

This means for the Simple Example: `<netElement>` objects that model tracks on microscopic level should have the physical length of the tracks (in meters). The related source code looks like this:

```
    <topology>
      <netElements>
        <netElement id="ne_a01" length="500.0">
          ...
        </netElement>
        <netElement id="ne_a02" length="500.0">
          ...
        </netElement>
        <netElement id="ne_a03" length="200.0">
          ...
        </netElement>
        <netElement id="ne_b01" length="500.0">
          ...
        </netElement>
        <netElement id="ne_b02" length="450.0">
          ...
        </netElement>
        <netElement id="ne_b03" length="200.0">
          ...
        </netElement>
        <netElement id="ne_b04" length="50.0">
          ...
        </netElement>
        <netElement id="ne_b05" length="200.0">
```

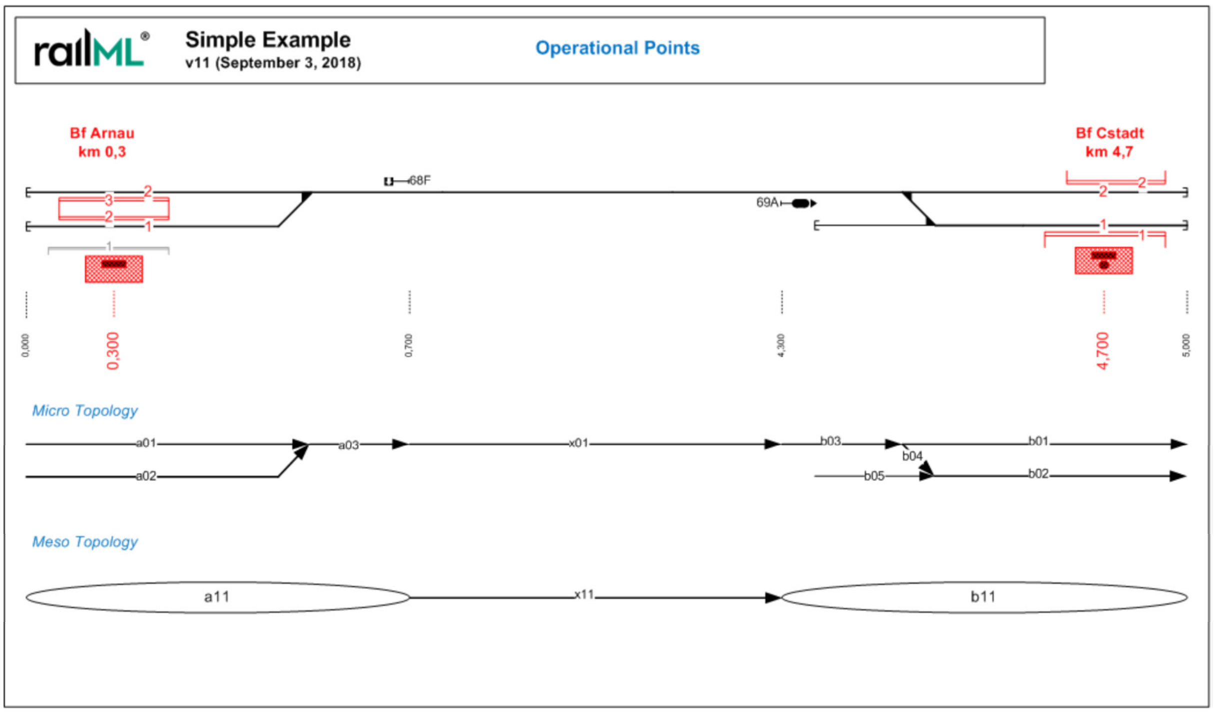
```
          ...
        </netElement>
        <netElement id="ne_x01" length="3600.0">
          ...
        </netElement>
        ...
      </netElements>
      ...
    </topology>
```

What you may have learned
• Positioning systems are relevant for all domains with spatial reference. Therefore, they are grouped together with `<positioning>` in the `<common>` domain. • Every `<netElement>` object has a child element `<associatedPositioningSystem>` which links intrinsic (topology) and other (linear and geometric) positioning systems. • For the mesoscopic topology, positioning is defined by a line identifier and a kilometer value. In the simple example, these values are the identifier “6869” and the kilometer range 0.000 to 5.000 • When the positioning system is defined, the `<associatedPositioningSystem>` is used to link the borders of the mesoscopic topology `<netElement>` objects. • Topology `<netElement>` objects can have an attribute `@length` if they build the basis for physical linear elements, e.g. railway tracks.

3.4 Operational Points

! **The content of this document is a learning tutorial**. Please check current railML 3.1 syntax documentation at railML.org for latest binding information on elements, attributes and use.

The Simple Example contains two operational control points in form of stations. Their names are "Bf Arnau" and "Bf Cstadt" and they are located in the beginning and in the end of the railway line. The center of "Bf Arnau" is located at kilometer 0.300 and the center of "Bf Cstadt" is located at kilometer 4.700 as can be seen in the following figure.



In order to locate the `<operationalPoint>` elements within the topology network (`<netElement>` "a11" and `<netElement>` "b11") the child element `<spotLocation>` is being used. Considering the mesoscopic topology network described in section 0 station "Bf Arnau" is located on the `<netElement>` "a11" and station "Bf Cstadt" is located on the `<netElement>` "b11". Since both referenced `<netElement>` objects are non-linear objects, their intrinsic position equals "0". The `<spotLocation>` element further implements the linking between the (topologic) intrinsic location and the railway line coordinates mentioned above.

Please note: An `<operationalPoint>` is modelled as part of the `<functionalInfrastructure>`. Thus, it is independent from the underlying topology that it references.

Both operational points have also designators, which act like external identifiers in a given register. In particular, "Bf Arnau" has the code "OAR" in the register named "RL100" and "Bf Cstadt" is designated with the RL100 code "OCS". The specification of the operational aspects of the operational points is given in the child element `<opOperations>`. With the child element `<name>` different names for the operational point (e.g. in different languages) can be defined. The child element `<opEquipment>` can be used to explicitly list infrastructure elements that belong to this operational point, e.g. platform edges or signals. The `<platform>` model is described in detail in section 0. The resulting railML code snippet looks like this:

```
  <infrastructure id="is_01">
    ...
    <functionalInfrastructure>
      <operationalPoints>
        <operationalPoint id="opp01">
          <name name="Bf Arnau" language="de"/>
          <name name="Adamov" language="cz"/>
          <spotLocation id="opp01_sloc01" netElementRef="ne_a11"
applicationDirection="both">
            <linearCoordinate positioningSystemRef="lps01" measure="300.0"/>
          </spotLocation>
          <designator register="RL100" entry="OAR"/>
          <opEquipment>
            <ownsPlatform ref="plf01"/>
          </opEquipment>
          <opOperations>
            <opOperation operationalType="station" trafficType="passenger"/>
          </opOperations>
        </operationalPoint>
        <operationalPoint id="opp02">
          <name name="Bf Cstadt" language="de"/>
          <name name="Bouzov" language="cz"/>
          <spotLocation id="opp02_sloc01" netElementRef="ne_b11"
applicationDirection="both">
            <linearCoordinate positioningSystemRef="lps01" measure="4700.0"/>
          </spotLocation>
          <designator register="RL100" entry="OCS"/>
          <opEquipment>
            <ownsPlatform ref="plf02"/>
            <ownsPlatform ref="plf03"/>
          </opEquipment>
          <opOperations>
            <opOperation operationalType="station" trafficType="passenger"/>
          </opOperations>
        </operationalPoint>
      </operationalPoints>
    </functionalInfrastructure>
  </infrastructure>
```

What you may have learned

- Operational points are usually considered macroscopic railway elements.
- `<operationalPoint>` elements within the topology network are located by their child elements `<spotLocation>` or `<linearLocation>`.
- When a referenced `<netElement>` object is non-linear, the intrinsic position equals “0”.
- Operational points usually have (external) designators given by an existing register, e.g. the “DIDOK” register.
- Further operational point specifications are given by the child element `<opOperations>`.
- The child element `<name>` allows for further name definitions for operational points; `<name>` can be repeated any number of times.
- The child element `<opEquipment>` provides a container for collecting references to infrastructure elements that belong to this operational point.

3.5 Railway Line

! **The content of this document is a learning tutorial**. Please check current railML 3.1 syntax documentation at railML.org for latest binding information on elements, attributes and use.

The Simple Example contains one railway line, which connects the two operational points defined in the section before. This railway line is specified as being a main line and its line category according to the EU regulation EN 15528 is set to “other:CE”. Further, the two operation control points are referenced by the line. Like the operational point, a <line> can have several names, which are given in the child element <name>.

```
  <infrastructure id="is_01">
    <functionalInfrastructure>
      <lines>
        <line id="lin01" lineCategory="other:CE" lineType="mainLine"
infrastructureManagerRef="im_01">
          <name name="Malý příklad železniční tratě" language="cz"/>
          <name name="Kleine Beispielstrecke" language="de"/>
          <name name="Simple Example railway line" language="en"/>
          <name name="Petit Exemple Ligne Ferroviaire" language="fr"/>
          <name name="Små eksempel på jernbanelinjen" language="no"/>
          <beginsInOP ref="opp01"/>
          <endsInOP ref="opp02"/>
        </line>
      </lines>
      ...
    </functionalInfrastructure>
  </infrastructure>
```

The railway line is owned by the infrastructure manager referenced via the attribute `@infrastructureManagerRef`. The infrastructure manager is a typical organizational unit in railways and can be found in the `<common>` section. The attribute `<infrastructureManager>@code` points to a value in the railML codelist *InfrastructureManagers.xml* where many national and regional infrastructure managers are listed. If you cannot find the infrastructure manager that is of interest for you in the list, please contact railML.org so that it can be easily added.

```
  <common id="co_01">
    <organizationalUnits>
      <infrastructureManager id="im_01" code="SZDC"/>
    </organizationalUnits>
  </common>
```

The railway line is connected with the line positioning reference system via a `<linearLocation>` element. Since the line spans over three mesoscopic `<netElement>` objects, the `<linearLocation>` contains three `<associatedElement>` child elements, which provide the reference to the topology layer. Please note, that in the specific example the anchor locations on the `<netElement>` objects are not given by the “intrinsic coordinates”, but by linear coordinates of positioning system “lps01” referenced by the child elements `<linearCoordinateBegin>` and `<linearCoordinateEnd>`. All three `<netElement>` objects are fully covered by the `<line>`. The resulting description of the railway line looks like this:

```
      <lines>
```

```
        <line id="lin01" lineCategory="other:CE" lineType="mainLine"
infrastructureManagerRef="im_01">
          ...
          <linearLocation id="lin01_lloc01" applicationDirection="both">
            <associatedNetElement netElementRef="ne_a11" keepsOrientation="true">
              <linearCoordinateBegin positioningSystemRef="lps01" measure="0.0"/>
              <linearCoordinateEnd positioningSystemRef="lps01" measure="700.0"/>
            </associatedNetElement>
            <associatedNetElement netElementRef="ne_x11" keepsOrientation="true">
              <linearCoordinateBegin positioningSystemRef="lps01" measure="700.0"/>
              <linearCoordinateEnd positioningSystemRef="lps01" measure="4300.0"/>
            </associatedNetElement>
            <associatedNetElement netElementRef="ne_b11" keepsOrientation="true">
              <linearCoordinateBegin positioningSystemRef="lps01" measure="4300.0"/>
              <linearCoordinateEnd positioningSystemRef="lps01" measure="5000.0"/>
            </associatedNetElement>
          </linearLocation>
          ...
        </line>
      </lines>
```

If you want to make use of the intrinsic coordinates instead, you may leave out the linear positioning coordinates, because due to the linking of both coordinate systems in the `<topology>`, one coordinate can be derived from the other one by calculation. The `<netElement>` instances “a11” and “b11” are non-linear and therefore have only an intrinsic coordinate “0”. The intrinsic coordinates of linear `<netElement>` “x11” range from “0” to “1”. The resulting source code contains the same information about the location of the `<line>` like the previous code snippet:

```
      <lines>
        <line id="lin01" lineCategory="other:CE" lineType="mainLine"
infrastructureManagerRef="im_01">
          ...
          <linearLocation id="lin01_lloc01" applicationDirection="both">
            <associatedElement netElementRef="ne_a11" intrinsicCoordBegin="0"
intrinsicCoordEnd="0" keepsOrientation="true"/>
            <associatedElement netElementRef="ne_x11" intrinsicCoordBegin="0"
intrinsicCoordEnd="1" keepsOrientation="true"/>
            <associatedElement netElementRef="ne_b11" intrinsicCoordBegin="0"
intrinsicCoordEnd="0" keepsOrientation="true"/>
          </linearLocation>
          ...
        </line>
      </lines>
```

The child element `<lineLayout>` can be used to describe the general layout of the line, e.g. characterising it as a single track line. The child element `<linePerfomance>` is a container element to collect performance parameters of the railway line, e.g. the maximum line speed. Further, you may reference `<loadingGauge>` elements that define the allowed loading gauge on the railway line. The maximum allowed axle load or meterload is another line performance parameter that is not modelled in the Simple Example. The resulting code snippet for the `<line>` looks like this:

```
      <lines>
        <line id="lin01" lineCategory="other:CE" lineType="mainLine"
infrastructureManagerRef="im_01">
          <name name="Malý příklad železniční tratě" language="cz"/>
          <name name="Kleine Beispielstrecke" language="de"/>
          <name name="Simple Example railway line" language="en"/>
          <name name="Petit Exemple Ligne Ferroviaire" language="fr"/>
          <name name="Små eksempel på jernbanelinjen" language="no"/>
          <linearLocation id="lin01_lloc01" applicationDirection="both">
            <associatedNetElement netElementRef="ne_a11" keepsOrientation="true">
              <linearCoordinateBegin positioningSystemRef="lps01" measure="0.0"/>
              <linearCoordinateEnd positioningSystemRef="lps01" measure="700.0"/>
            </associatedNetElement>
            <associatedNetElement netElementRef="ne_x11" keepsOrientation="true">
              <linearCoordinateBegin positioningSystemRef="lps01" measure="700.0"/>
              <linearCoordinateEnd positioningSystemRef="lps01" measure="4300.0"/>
            </associatedNetElement>
            <associatedNetElement netElementRef="ne_b11" keepsOrientation="true">
              <linearCoordinateBegin positioningSystemRef="lps01" measure="4300.0"/>
              <linearCoordinateEnd positioningSystemRef="lps01" measure="5000.0"/>
            </associatedNetElement>
          </linearLocation>
          <beginsInOP ref="opp01"/>
          <endsInOP ref="opp02"/>
          <lineLayout numberOfTracks="single"/>
          <linePerformance usablePlatformLength="200" maxSpeed="80">
            <allowedLoadingGauge ref="log01"/>
          </linePerformance>
        </line>
      </lines>
```

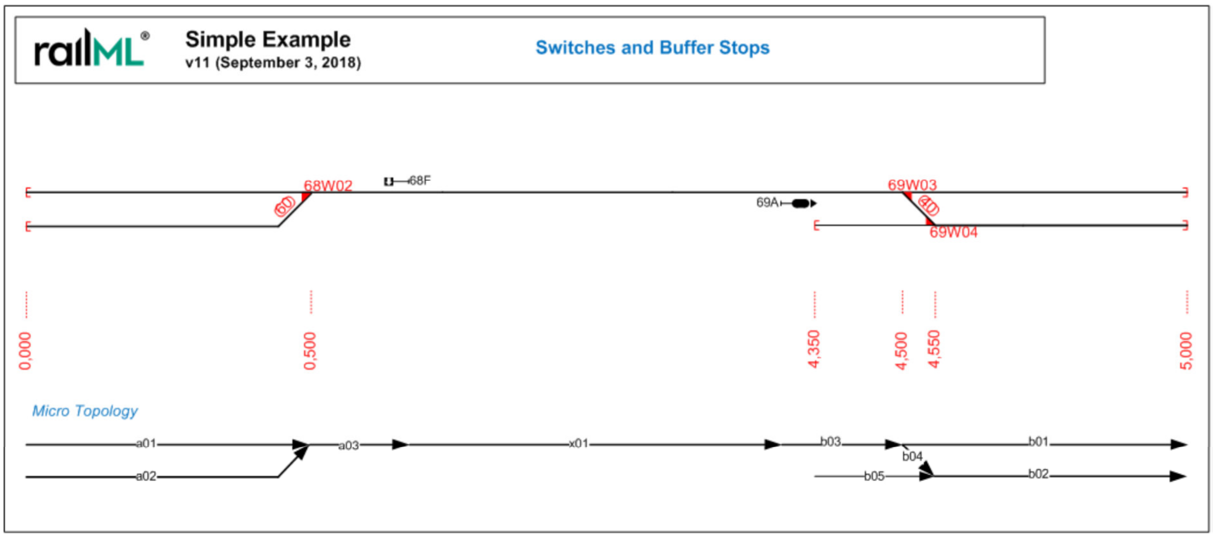
What you may have learned

- Railway lines connect operational points, and are usually considered macroscopic railway elements.
- The `<linearLocation>` element connects the railway line with the line positioning reference system, classically known as “mileage”
- When a line spans over a given number of mesoscopic `<netElement>` objects, the `<linearLocation>` element contains an equal number of `<associatedElement>` child elements.
- For locating an infrastructure element on the topology defined by the `<netElement>` objects, you may use either intrinsic coordinates or linear coordinates (if they have been defined for all the relevant `<netElement>` instances).
- Railway lines are owned by an infrastructure manager. The infrastructure manager is referenced via the attribute `@infrastructureManagerRef`.
- The attribute `<infrastructureManager>@code` in the `<common>` schema points to a value in the railML codelist InfrastructureManagers.xml, where infrastructure managers are listed.
- Further line parameters can be found in child elements `<lineLayout>` and `<linePerformance>`

3.6 Switches and BufferStops

! **The content of this document is a learning tutorial**. Please check current railML 3.1 syntax documentation at railML.org for latest binding information on elements, attributes and use.

The Simple Example contains three switches: one in station “Bf Arnau” and two more in station “Bf Cstadt”. Their locations in form of line coordinates as well as their names are marked in the following figure:



In the following code snippet three `<switch>` elements have been listed as part of the `<functionalInfrastructure>`. Besides their names and line positions further information are given, e.g. about the allowed speed on the branches of the switches. Please note that each switch branch (`<leftBranch>` and `<rightBranch>`) references a `<netRelation>` element and thus builds the connection between the functional switch model and the track network topology. Consequently, the `<switch>` element alone cannot be used for deriving the railway track network, but has to be interpreted together with the underlying topology layer.

```
    <functionalInfrastructure>
      ...
      <switches>
        <switch id="swi01" continueCourse="right" branchCourse="left" defaultCourse="right"
type="ordinarySwitch">
          <name name="68W02"/>
          <spotLocation id="swi01_sloc01" netElementRef="ne_a03"
applicationDirection="reverse" pos="0.0">
            <linearCoordinate positioningSystemRef="lps01" measure="500.0"/>
          </spotLocation>
          <leftBranch netRelationRef="nr_a02a03" speedBranching="60" speedJoining="60"
radius="-500"/>
          <rightBranch netRelationRef="nr_a01a03" speedBranching="80" speedJoining="80"
radius="0"/>
        </switch>
        <switch id="swi02" continueCourse="left" branchCourse="right" defaultCourse="left"
type="ordinarySwitch">
          <name name="69W03"/>
          <spotLocation id="swi02_sloc01" netElementRef="ne_b03"
applicationDirection="normal" pos="200.0">
            <linearCoordinate positioningSystemRef="lps01" measure="4.500"/>
          </spotLocation>
          <leftBranch netRelationRef="nr_b01b03" speedBranching="80" speedJoining="80"
radius="0"/>
          <rightBranch netRelationRef="nr_b03b04" speedBranching="40" speedJoining="40"
radius="300"/>
        </switch>
        <switch id="swi03" continueCourse="right" branchCourse="left" defaultCourse="left"
type="ordinarySwitch">
          <name name="69W04"/>
          <spotLocation id="swi03_sloc01" netElementRef="ne_b02"
applicationDirection="normal" pos="0.0">
            <linearCoordinate positioningSystemRef="lps01" measure="4.550"/>
          </spotLocation>
          <leftBranch netRelationRef="nr_b02b05" speedBranching="60" speedJoining="60"
radius="0"/>
          <rightBranch netRelationRef="nr_b02b04" speedBranching="40" speedJoining="40"
radius="300"/>
        </switch>
      </switches>
      ...
    </functionalInfrastructure>
```

Please note:

1. In order to provide information about the allowed speed at switches and crossings, please use the attributes `<*branch>@speedBranching` and `<*branch>@speedJoining`. Don't define these speeds with extra (line) speed changes.
2. The switch's track continue course (attribute `@continueCourse`), that describes the mainly used switch branch, does not have to be identical with the switch's default course (attribute `@defaultCourse`), that defines the "standby position" of the switch.

The Simple Example further contains five buffer stops as part of the `<functionalInfrastructure>`. Four of them are located at the ends of the railway line "6869" in kilometre 0.000 and kilometre 5.000. They are of type "fixedBufferStop". The fifth `<bufferStop>` is located on a siding track that is not part of the railway line "6869" and therefore does not have a railway line coordinate. The following code snippet summarizes the buffer stop model.

```
    <functionalInfrastructure>
      <bufferStops>
        <bufferStop id="bus01" type="fixedBufferStop">
          <spotLocation id="bus01_sloc01" netElementRef="ne_a01"
applicationDirection="reverse" pos="0.0">
            <linearCoordinate positioningSystemRef="lps01" measure="0.0"/>
          </spotLocation>
        </bufferStop>
        <bufferStop id="bus02" type="fixedBufferStop">
          <spotLocation id="bus02_sloc01" netElementRef="ne_a02"
applicationDirection="reverse" pos="0.0">
            <linearCoordinate positioningSystemRef="lps01" measure="0.0"/>
          </spotLocation>
        </bufferStop>
        <bufferStop id="bus03" type="fixedBufferStop">
          <spotLocation id="bus03_sloc01" netElementRef="ne_b01"
applicationDirection="normal" pos="500.0">
            <linearCoordinate positioningSystemRef="lps01" measure="5000.0"/>
          </spotLocation>
        </bufferStop>
        <bufferStop id="bus04" type="fixedBufferStop">
```

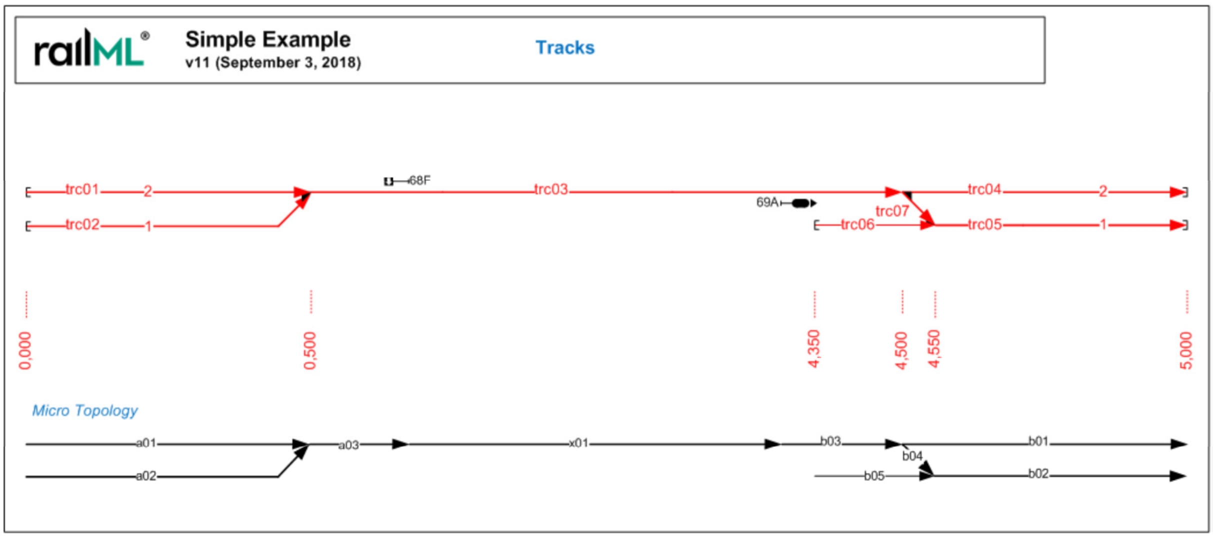
```
          <spotLocation id="bus04_sloc01" netElementRef="ne_b02"
applicationDirection="normal" pos="450.0">
            <linearCoordinate positioningSystemRef="lps01" measure="5000.0"/>
          </spotLocation>
        </bufferStop>
        <bufferStop id="bus05" type="sleeperCross">
          <spotLocation id="bus05_sloc01" netElementRef="ne_b05"
applicationDirection="reverse" pos="0.0">
          </spotLocation>
        </bufferStop>
      </bufferStops>
      ...
    </functionalInfrastructure>
```

What you may have learned
• Switches and buffer stops are usually located on the microscopic topology level. • `<switch>` elements and `<bufferStop>` elements are listed as part of the `<functionalInfrastructure>`. • Every switch branch references a `<netRelation>` element, thus connecting the functional model and the track network topology.

3.7 Tracks

! **The content of this document is a learning tutorial**. Please check current railML 3.1 syntax documentation at railML.org for latest binding information on elements, attributes and use.

While Operational points and Lines can be considered being mesoscopic or macroscopic railway infrastructure elements, switches, buffer stops and tracks form the microscopic level. A track is defined as a section of rails between two switches/crossings or between a switch/crossing and a buffer stop. In the Simple Example seven tracks can be found. While six of them are identical to microscopic topology `<netElement>` instances, the seventh track spans over three `<netElement>` objects from the switch at line kilometre 0.500 until the next switch at line kilometre 4.500.



Like the `<line>`, the `<track>` is modelled as a part of the functional railway infrastructure and thus located within the topology network. In particular, a `<track>` shall reference at least one `<netElement>` on microscopic level using a `<linearLocation>`. As already indicated, a `<track>` may also span over several `<netElement>` objects like `<track>` "trc03". Further, each `<track>` is oriented by explicitly listing a `<trackBegin>` and a `<trackEnd>` element. These child elements are used to reference "track nodes", which are delimiting elements of the functional infrastructure. Following the above mentioned definition of a track, `<bufferStop>`, `<switch>` and `<crossing>` elements can be referenced.

A `<track>` may have a number of additional parameters: The child element `<length>` provides the (physical or operationally available) length of the track in meters. The attribute `@type` is used to distinguish between the different track types including "`mainTrack`", "`secondaryTrack`", "`sidingTrack`" and "`connectingTrack`". The attribute `@mainDirection` defines the main direction of railway operation on the track. The following code snippet puts the Simple Example railway track network into railML code.

```
<functionalInfrastructure>
  ...
  <tracks>
    <track id="trc01" type="mainTrack" >
      <name name="2"/>
```

```
          <linearLocation id="trc01_lloc01" applicationDirection="both">
            <associatedNetElement netElementRef="ne_a01" keepsOrientation="true"
fromPos="0.0" toPos="500.0">
            </associatedNetElement>
          </linearLocation>
          <trackBegin ref="bus01"/>
          <trackEnd ref="swi01"/>
          <length value="500.0" type="physical"/>
        </track>
        <track id="trc02" type="secondaryTrack">
          <name name="1"/>
          <linearLocation id="trc02_lloc01" applicationDirection="both">
            <associatedNetElement netElementRef="ne_a02" keepsOrientation="true"
fromPos="0.0" toPos="500.0">
            </associatedNetElement>
          </linearLocation>
          <trackBegin ref="bus02"/>
          <trackEnd ref="swi01"/>
          <length value="500.0" type="physical"/>
        </track>
        <track id="trc03" type="mainTrack" mainDirection="both">
          <linearLocation id="trc03_lloc01" applicationDirection="both">
            <associatedNetElement netElementRef="ne_a03" keepsOrientation="true"
sequence="1" fromPos="0.0" toPos="200.0">
            </associatedNetElement>
            <associatedNetElement netElementRef="ne_x01" keepsOrientation="true"
sequence="2" fromPos="0.0" toPos="3600.0">
            </associatedNetElement>
            <associatedNetElement netElementRef="ne_b03" keepsOrientation="true"
sequence="3" fromPos="0.0" toPos="200.0">
            </associatedNetElement>
          </linearLocation>
          <trackBegin ref="swi01"/>
          <trackEnd ref="swi02"/>
          <length value="4000.0" type="physical"/>
        </track>
        <track id="trc04" type="mainTrack">
          <name name="2"/>
          <linearLocation id="trc04_lloc01" applicationDirection="both">
            <associatedNetElement netElementRef="ne_b01" keepsOrientation="true"
fromPos="0.0" toPos="500.0">
            </associatedNetElement>
          </linearLocation>
          <trackBegin ref="swi02"/>
          <trackEnd ref="bus03"/>
          <length type="physical" value="500.0"/>
        </track>
        <track id="trc05" type="secondaryTrack">
          <name name="1"/>
          <linearLocation id="trc05_lloc01" applicationDirection="both">
            <associatedNetElement netElementRef="ne_b02" keepsOrientation="true"
fromPos="0.0" toPos="450.0">
            </associatedNetElement>
          </linearLocation>
          <trackBegin ref="swi03"/>
          <trackEnd ref="bus04"/>
          <length value="450.0" type="physical"/>
        </track>
        <track id="trc06" type="sidingTrack">
          <linearLocation id="trc06_lloc01" applicationDirection="both">
            <associatedNetElement netElementRef="ne_b05" keepsOrientation="true"
fromPos="0.0" toPos="200.0">
            </associatedNetElement>
          </linearLocation>
          <trackBegin ref="bus05"/>
          <trackEnd ref="swi03"/>
          <length value="200.0" type="physical"/>
        </track>
        <track id="trc07" type="connectingTrack">
          <linearLocation id="trc07_lloc01" applicationDirection="both">
            <associatedNetElement netElementRef="ne_b04" keepsOrientation="true"
fromPos="0.0" toPos="50.0">
            </associatedNetElement>
          </linearLocation>
          <trackBegin ref="swi02"/>
          <trackEnd ref="swi03"/>
          <length value="50.0" type="physical"/>
        </track>
      </tracks>
          ...
    </functionalInfrastructure>
```

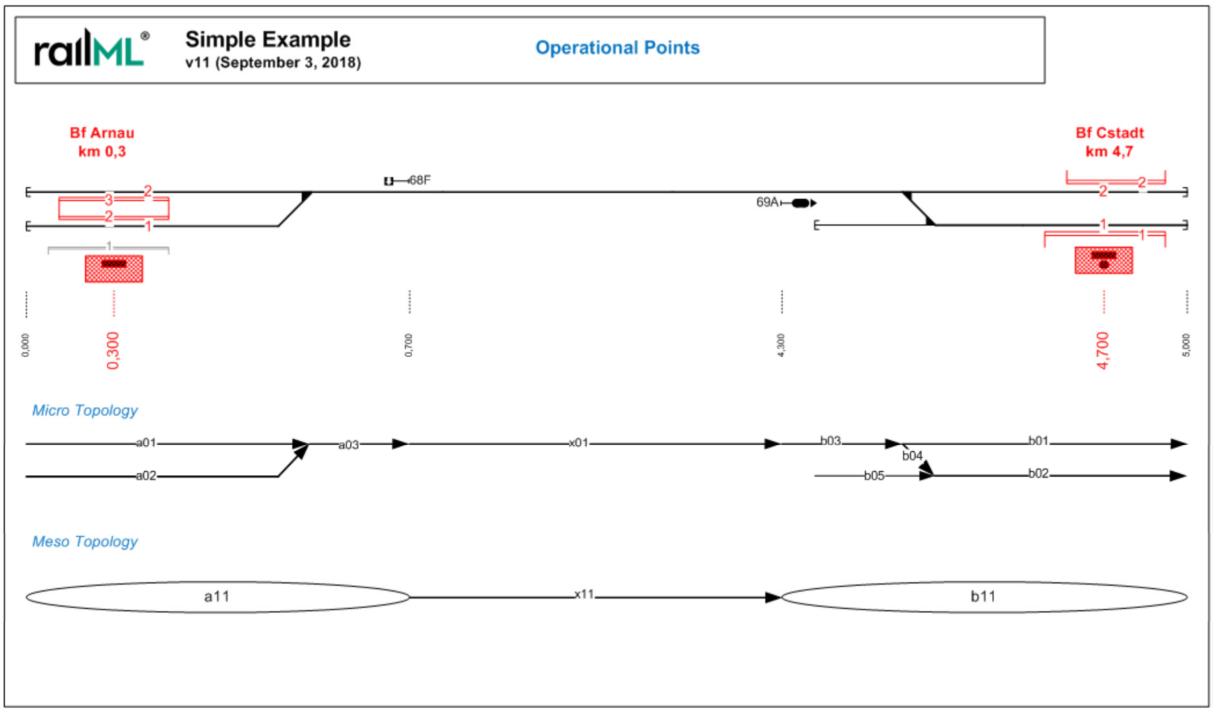
Please note: `<track>` objects together with `<switch>`, `<crossing>` and `<bufferStop>` elements don't form another topology layer within the railML data model. All topology aspects are defined under the umbrella element `<topology>` with its `<netElement>`, `<netRelation>` and `<network>` elements. The functional infrastructure element `<track>` shall be used as a base element for defining operational parameters on track level of the railway network.

What you may have learned
• A track is defined as a section of rails between two switches/crossing or between a switch/crossing and buffer stop. • The `<track>` element is part of the functional infrastructure and thus located on top of the topology network. • A `<track>` always references at least one `<netElement>` on microscopic level using `<linearLocation>`, though it can span over several `<netElement>` objects. • `<track>` orientation is defined by listing a `<trackBegin>` and `<trackEnd>` element. • Additional attributes and child elements for the `<track>` element may have a number of different parameters, among others `<length>`, `@type` and `@mainDirection`.

3.8 Platforms and Platform Edges

! **The content of this document is a learning tutorial**. Please check current railML 3.1 syntax documentation at railML.org for latest binding information on elements, attributes and use.

In section 0 the implementation of the OperationalPoint concept has been presented. While operational points describe the operation aspects of a station on a meso- or macroscopic level, the platforms and platform edges are the relevant elements on microscopic topology level. Platform edges are located along railway tracks and define the locations where passenger trains stop for passenger transfer. Knowing which platform edges belong to which platform is important for estimating passenger transfer time. In the Simple Example, both operational points have two platform edges: In “Bf Arnau” both platform edges are situated at one platform, while “Bf Cstadt” has two platforms each one having one platform edge with different lengths.



In railML 3.1, platforms and platform edges are both modelled using the `<platform>` element. Thus, the implementation of the Simple Example in railML results in seven `<platform>` elements. They are all listed in the container element `<platforms>`. The `<platform>` element may reference an arbitrary number of other `<platform>` elements that represent platform edges via the child element `<ownsPlatformEdge ref=””>`. A platform edge may have several (usable) lengths depending on the direction the train is travelling. This can be modelled via the repeatable child element `<length>` and included attributes for specifying the length information. The width of the platform is handled in the same way using the child element `<width>`.

```
    <functionalInfrastructure>
      ...
      <platforms>
        <platform id="plf01">
          <spotLocation id="plf01_sloc01" netElementRef="ne_a11"
applicationDirection="both">
```

```
            <linearCoordinate measure="300.0" positioningSystemRef="lps01"/>
          </spotLocation>
          <ownsPlatformEdge ref="ple01"/>
          <ownsPlatformEdge ref="ple02"/>
        </platform>
        <platform id="plf02">
          <spotLocation id="plf02_sloc01" netElementRef="ne_b11"
applicationDirection="both">
            <linearCoordinate measure="4700.0" positioningSystemRef="lps01"/>
          </spotLocation>
          <ownsPlatformEdge ref="ple03"/>
        </platform>
        <platform id="plf03">
          <spotLocation id="plf03_sloc01" netElementRef="ne_b11"
applicationDirection="both">
            <linearCoordinate measure="4700.0" positioningSystemRef="lps01"/>
          </spotLocation>
          <ownsPlatformEdge ref="ple04"/>
        </platform>
        <platform id="ple01" height="550">
          <name name="Gleis 3" language="de"/>
          <linearLocation id="ple01_lloc01" applicationDirection="both">
            <associatedNetElement netElementRef="ne_a01" keepsOrientation="true"
fromPos="200.0" toPos="400.0">
              <linearCoordinateBegin measure="200.0" positioningSystemRef="lps01"
lateralDistance="1.7" lateralSide="right"/>
              <linearCoordinateEnd measure="400.0" positioningSystemRef="lps01"
lateralDistance="1.7" lateralSide="right"/>
            </associatedNetElement>
          </linearLocation>
          <length type="physical" value="200.00" validForDirection="both"/>
        </platform>
        <platform id="ple02" height="550">
          <name name="Gleis 2" language="de"/>
          <linearLocation id="ple02_lloc01" applicationDirection="both">
            <associatedNetElement netElementRef="ne_a02" keepsOrientation="true"
fromPos="200.0" toPos="400.0">
              <linearCoordinateBegin measure="200.0" positioningSystemRef="lps01"
lateralDistance="1.7" lateralSide="left"/>
              <linearCoordinateEnd measure="400.0" positioningSystemRef="lps01"
lateralDistance="1.7" lateralSide="left"/>
            </associatedNetElement>
          </linearLocation>
          <length type="physical" value="200.00" validForDirection="both"/>
        </platform>
        <platform id="ple03" height="550">
          <name name="Gleis 2" language="de"/>
          <linearLocation id="ple03_lloc01" applicationDirection="both">
            <associatedNetElement netElementRef="ne_b01" keepsOrientation="true"
fromPos="150.0" toPos="350.0">
              <linearCoordinateBegin measure="4650.0" positioningSystemRef="lps01"/>
              <linearCoordinateEnd measure="4850.0" positioningSystemRef="lps01"/>
            </associatedNetElement>
          </linearLocation>
          <length type="physical" value="200.00" validForDirection="both"/>
        </platform>
        <platform id="ple04" height="380">
          <name name="Gleis 1" language="de"/>
          <linearLocation id="ple04_lloc01" applicationDirection="both">
            <associatedNetElement netElementRef="ne_b02" keepsOrientation="true"
fromPos="100.0" toPos="350.0">
              <linearCoordinateBegin measure="4650.0" positioningSystemRef="lps01"/>
              <linearCoordinateEnd measure="4900.0" positioningSystemRef="lps01"/>
            </associatedNetElement>
          </linearLocation>
          <length type="physical" value="250.00" validForDirection="both"/>
        </platform>
      </platforms>
      ...
    </functionalInfrastructure>
```

Please note:

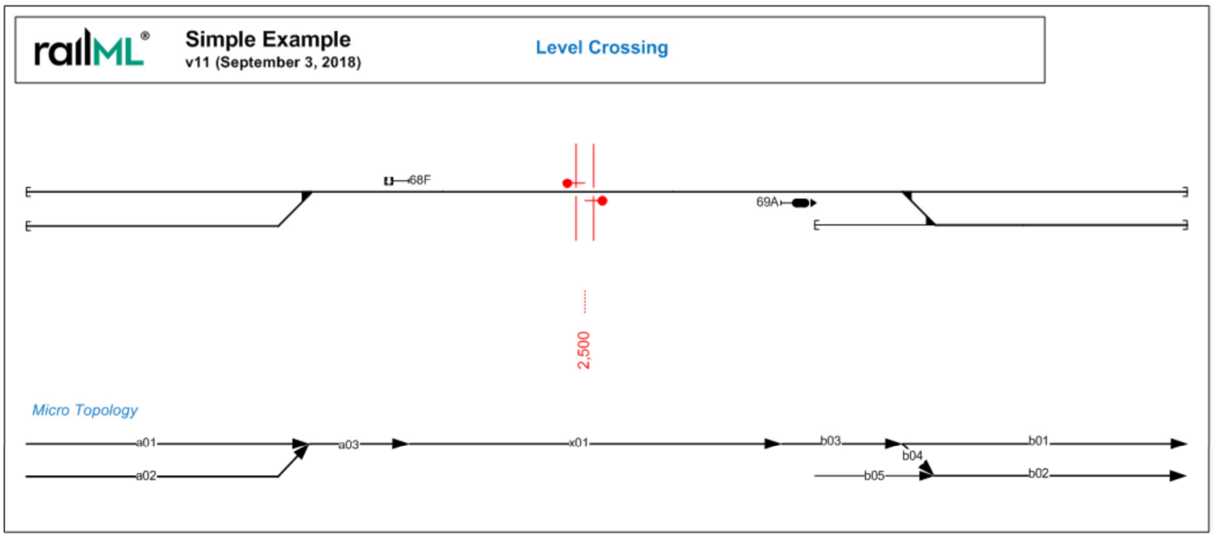
1. The `<platform>` element is modelled as "entity". Therefore, it may be placed on the railway topology layer in several ways using `<spotLocation>`, `<linearLocation>` or `<areaLocation>`. In the given example the `<platform>` elements representing platforms are located as spot elements on the mesoscopic level `<netElement>` objects "a11" and "b11". This translates into: "The station 'Bf Arnau' has one platform and the station 'Bf Cstadt' has two platforms." The `<platform>` elements representing platform edges have been placed as linear elements on the microscopic level `<netElement>` objects "a01", "a02", "b01" and "b02". In general, it is recommended to locate platform edges on track level topology (microscopic) while platforms may be located also at more aggregated levels.
2. To distinguish between platforms and platform edges, it is further recommended to define the `<width>` only for platforms. Platform edges shall be interpreted as interfaces between train and platform and therefore shall not have a physical width.

What you may have learned
• Platforms and platform edges are relevant elements on the microscopic topology level. • All platforms have one or more platform edges, which may vary in length. `<platform>` elements reference other `<platform>` elements representing platform edges via the child element `<ownsPlatformEdge ref="">`. • Both, platforms and platform edges are implemented through the `<platform>` elements. They are listed under the container element `<platforms>.` • Platform edges can have several (usable) lengths depending on the direction the train is travelling. Therefore, the child element `<length>` can be repeated.

3.9 Level Crossing

The content of this document is a learning tutorial. Please check current railML 3.1 syntax documentation at railML.org for latest binding information on elements, attributes and use.

The Simple Example includes a level crossing just in the middle of the railway line at line kilometer 2.500. As depicted in the following figure, the level crossing is technically secured by half-barriers.



In railML 3.1 the level crossing is modelled as a component of the functional railway infrastructure. Like any other of these elements the `<levelCrossing>` can be placed on one or more `<netElement>` objects forming the topology layer of the railway network. In the following code snippet the `<levelCrossing>` has been placed on the microscopic topology level `<netElement>` "x01".

```
    <functionalInfrastructure>
      ...
      <levelCrossings>
        <levelCrossing id="lcr01" activation="infrastructureAutomatic">
          <name name="LX Arnau Cstadt"/>
          <spotLocation id="lcr01_sloc01" netElementRef="ne_x01"
applicationDirection="both" pos="1800.0">
            <linearCoordinate positioningSystemRef="lps01" measure="2500.00"/>
          </spotLocation>
          <protection barriers="singleHalfBarrier" lights="none" acoustic="none"/>
        </levelCrossing>
      </levelCrossings>
      ...
    </functionalInfrastructure>
```

The technical protection system of the level crossing is given with the child element `<protection>` and enclosed attributes specifying a technical protection with barriers, lights and acoustic signals. Further, the activation of the level crossing protection is described with the attribute `@activation`. The attributes `@obstacleDetection`, `@opensOnDemand` and `@supervision` provide additional information about the technical implementation of the level crossing, which are not included in the Simple Example.

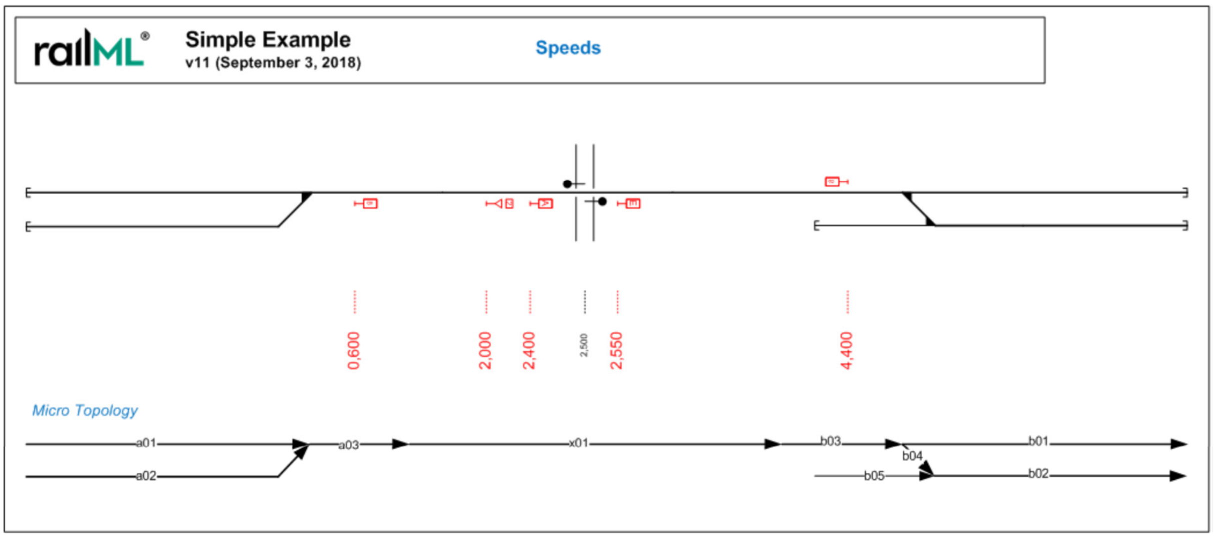
Please note: The attribute `<levelCrossing><protection>@barriers` is an “open” attribute. This means, that it is recommended to choose one of the already defined attribute values (“`singleFullBarrier`”, “`singleHalfBarrier`”, “`doubleHalfBarrier`”), but you are free to add your own attribute value here, too. However, these own attributes must not describe a state that is already defined by the existing enumeration values and it shall be used only temporarily (until it will be added to the enumeration data type of the railML standard).

What you may have learned
• Level crossings are modelled as components of the functional infrastructure. Thus, the `<levelCrossing>` can be placed on one or more `<netElement>` objects. • Technical protection systems of the level crossings are given by the child element `<protection>`. It is recommended to use a predefined enumeration values to specify the level crossing protection system (barriers, lights, acoustic signals).

3.10 Speeds

! **The content of this document is a learning tutorial**. Please check current railML 3.1 syntax documentation at railML.org for latest binding information on elements, attributes and use.

The **maximum permitted speed** at a railway line is defined by both, the direction specific line speed and temporary speed restriction zones. In the Simple Example the line speed is given with 80 km/h. For the driving direction from Operational Point "Bf Arnau" to Operational Point "Bf Cstadt" there is another (temporary) speed restriction at the level crossing. Due to e.g. insufficient line of sight or other reasons the maximum permitted speed there is reduced to 20 km/h. As depicted in the following figure, this temporary speed reduction is only valid for one driving direction, but not for the reverse one.



In general, the information about the maximum permitted speed is independent from a specific speed signal/panel. Therefore, the railML v3 implementation contains the functional infrastructure element `<speedSection>`. A `<speedSection>` is used to reference a maximum permitted speed with a part of the railway network that is travelled in a certain direction. In many cases, the begin and the end of a `<speedSection>` are marked with speed signals. The boolean attribute `@isSignalized` can be used to indicate this situation independently from the `<signal>` elements that are described in the next section.

At first, the direction specific line speed is implemented based on two `<speedSection>` elements that are located on the microscopic topology level:

```
  <functionalInfrastructure>
    ...
    <speeds>
      <speedSection id="sps01" vMax="80" isTemporary="false" isSignalized="true">
        <linearLocation id="sps01_lloc01" applicationDirection="normal">
          <associatedNetElement netElementRef="ne_a03" keepsOrientation="true">
            <linearCoordinateBegin positioningSystemRef="lps01" measure="600.0"/>
            <linearCoordinateEnd positioningSystemRef="lps01" measure="700.0"/>
          </associatedNetElement>
          <associatedNetElement netElementRef="ne_x01" keepsOrientation="true">
            <linearCoordinateBegin positioningSystemRef="lps01" measure="700.0"/>
            <linearCoordinateEnd positioningSystemRef="lps01" measure="4300.0"/>
          </associatedNetElement>
          <associatedNetElement netElementRef="ne_b03" keepsOrientation="true">
            <linearCoordinateBegin positioningSystemRef="lps01" measure="4300.0"/>
            <linearCoordinateEnd positioningSystemRef="lps01" measure="4500.0"/>
          </associatedNetElement>
          <associatedNetElement netElementRef="ne_b01" keepsOrientation="true">
            <linearCoordinateBegin positioningSystemRef="lps01" measure="4500.0"/>
            <linearCoordinateEnd positioningSystemRef="lps01" measure="5000.0"/>
          </associatedNetElement>
        </linearLocation>
      </speedSection>
      <speedSection id="sps02" vMax="80" isTemporary="false" isSignalized="true">
        <linearLocation id="sps02_lloc01" applicationDirection="reverse">
          <associatedNetElement netElementRef="ne_b03" keepsOrientation="true">
            <linearCoordinateBegin positioningSystemRef="lps01" measure="4400.0"/>
            <linearCoordinateEnd positioningSystemRef="lps01" measure="4300.0"/>
          </associatedNetElement>
          <associatedNetElement netElementRef="ne_x01" keepsOrientation="true">
            <linearCoordinateBegin positioningSystemRef="lps01" measure="4300.0"/>
            <linearCoordinateEnd positioningSystemRef="lps01" measure="700.0"/>
          </associatedNetElement>
          <associatedNetElement netElementRef="ne_a03" keepsOrientation="true">
            <linearCoordinateBegin positioningSystemRef="lps01" measure="700.0"/>
            <linearCoordinateEnd positioningSystemRef="lps01" measure="500.0"/>
          </associatedNetElement>
          <associatedNetElement netElementRef="ne_a01" keepsOrientation="true">
            <linearCoordinateBegin positioningSystemRef="lps01" measure="500.0"/>
            <linearCoordinateEnd positioningSystemRef="lps01" measure="0.0"/>
          </associatedNetElement>
        </linearLocation>
      </speedSection>
      ...
    </speeds>
    ...
  </functionalInfrastructure>
```

At next, the temporary speed restriction zone at the level crossing is defined:

```
  <functionalInfrastructure>
    ...
    <speeds>
      ...
      <speedSection id="sps03" vMax="20" isTemporary="true" isSignalized="true">
        <isValid from="2018-12-15" to="2018-12-22"/>
        <linearLocation id="sps03_lloc01" applicationDirection="normal">
          <associatedNetElement netElementRef="ne_x01" keepsOrientation="true">
            <linearCoordinateBegin positioningSystemRef="lps01" measure="2400.0"/>
            <linearCoordinateEnd positioningSystemRef="lps01" measure="2550.0"/>
          </associatedNetElement>
        </linearLocation>
      </speedSection>
    </speeds>
    ...
  </functionalInfrastructure>
```

Please note: the branching speed at switches and switch crossings shall not be defined via `<speedSection>` elements, but directly in the `<switch>` element and its `<*Branch>` children (see section 3.6)

What you may have learned

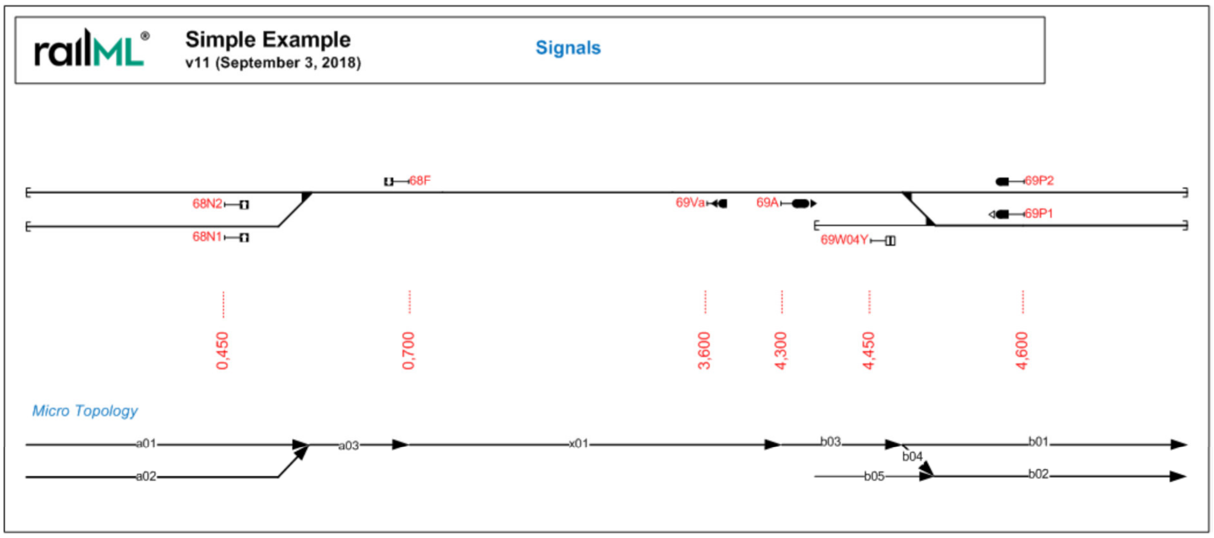
- Maximum permitted speed at a railway line is defined by both direction specific line speed and temporary speed restriction zones. In railML, information about speed limits is modelled independently from specific signals/panels.
- `<speedSection>` instances are functional infrastructure elements.
- `<speedSection>` references a maximum permitted speed on a defined part of the railway network in a certain direction of travel ("application direction").

3.11 Signalling Infrastructure

! **The content of this document is a learning tutorial**. Please check current railML 3.1 syntax documentation at railML.org for latest binding information on elements, attributes and use.

3.11.1 Signals

In the last step the signaling elements are placed at the tracks of the railway line. As depicted in the following figure, there are different kinds of signals installed in the different Operational Points: "Bf Arnau" is already equipped with ETCS and ETCS marker boards have been placed at the locations of former entry and exit signals. In "Bf Cstadt" conventional main and distant signals can be found. There is also a shunting signal at the siding track in "Bf Cstadt".

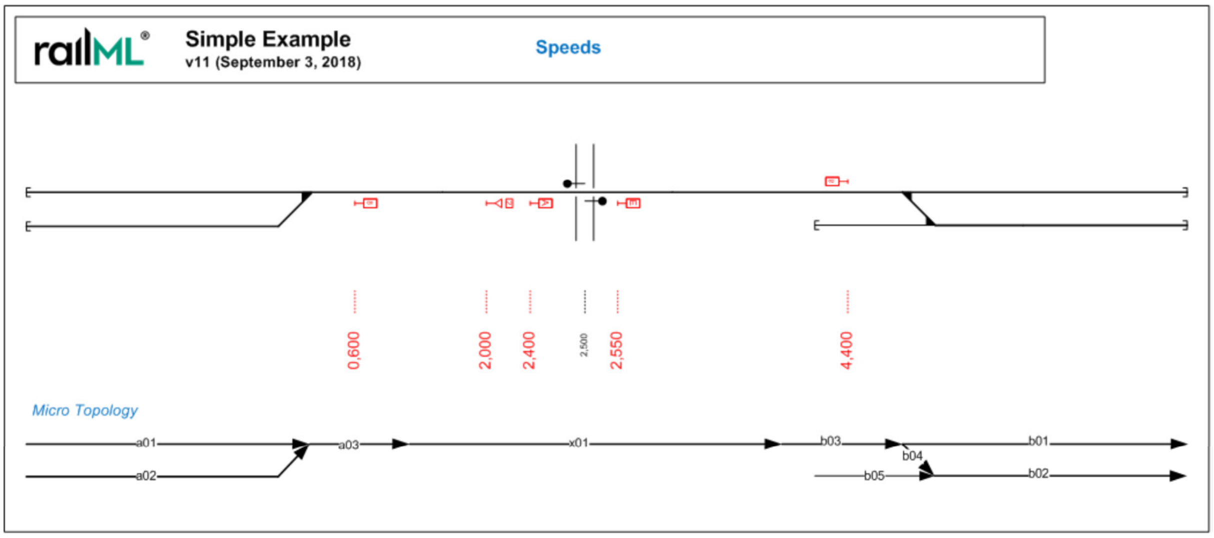


Modelling these eight signals results in the following code snippet:

```
    <functionalInfrastructure>
      ...
      <signals>
        <signal id="sig01" type="main" switchable="false">
          <name name="68N2"/>
          <spotLocation id="sig01_sloc01" netElementRef="ne_a01"
applicationDirection="normal" pos="450.0">
            <linearCoordinate positioningSystemRef="lps01" measure="450.0"
lateralDistance="2.2" lateralSide="right"/>
          </spotLocation>
          <isEtcsSignal/>
        </signal>
        <signal id="sig02" type="main" switchable="false">
          <name name="68N1"/>
          <spotLocation id="sig02_sloc01" netElementRef="ne_a02"
applicationDirection="normal" pos="450.0">
            <linearCoordinate positioningSystemRef="lps01" measure="450.0"
lateralDistance="2.2" lateralSide="right"/>
          </spotLocation>
          <isEtcsSignal/>
        </signal>
        <signal id="sig03" type="main" switchable="false">
          <name name="68F"/>
          <spotLocation id="sig03_sloc01" netElementRef="ne_a03"
applicationDirection="reverse" pos="200.0">
            <linearCoordinate positioningSystemRef="lps01" measure="700.0"
lateralDistance="2.2" lateralSide="left"/>
          </spotLocation>
          <isEtcsSignal/>
        </signal>
        <signal id="sig04" type="main" switchable="true">
          <name name="69A"/>
          <spotLocation id="sig04_sloc01" netElementRef="ne_b03"
applicationDirection="normal" pos="0.0">
            <linearCoordinate positioningSystemRef="lps01" measure="4300.0"
lateralDistance="2.2" lateralSide="right"/>
          </spotLocation>
        </signal>
        <signal id="sig05" type="main" switchable="true">
          <name name="69P2"/>
          <spotLocation id="sig05_sloc01" netElementRef="ne_b01"
applicationDirection="reverse" pos="100.0">
            <linearCoordinate positioningSystemRef="lps01" measure="4600.0"
lateralDistance="2.2" lateralSide="left"/>
          </spotLocation>
        </signal>
        <signal id="sig06" type="main" switchable="true">
          <name name="69P1"/>
          <spotLocation id="sig06_sloc01" netElementRef="ne_b02"
applicationDirection="reverse" pos="50.0">
            <linearCoordinate positioningSystemRef="lps01" measure="4600.0"
lateralDistance="2.2" lateralSide="left"/>
          </spotLocation>
        </signal>
        <signal id="sig07" type="distant" switchable="true">
          <name name="69Va"/>
          <spotLocation id="sig07_sloc01" netElementRef="ne_x01"
applicationDirection="normal" pos="2900.0">
            <linearCoordinate positioningSystemRef="lps01" measure="3600.0"
lateralDistance="2.2" lateralSide="right"/>
          </spotLocation>
        </signal>
        <signal id="sig08" type="shunting" switchable="true">
          <name name="69W04Y"/>
          <spotLocation id="sig08_sloc01" netElementRef="ne_b05"
applicationDirection="normal" pos="100.0">
          </spotLocation>
        </signal>
        ...
      </signals>
      ...
    </functionalInfrastructure>
```

Please note: Although the ETCS marker boards are panels and can only show one signal aspect without changing, they are modelled as `<signal>` elements. The attribute `@switchable=”false”` indicates that the described signal is in fact a panel.

The **speed signals** are physical markers at the points where the maximum permitted speed changes, e.g. at the beginning or at the end of a `<speedSection>`. In the Simple Example all these speed changes are signalized by speed signals/panels as depicted in the following figure:



The resulting code snippet for the five speed related signals/panels looks like this:

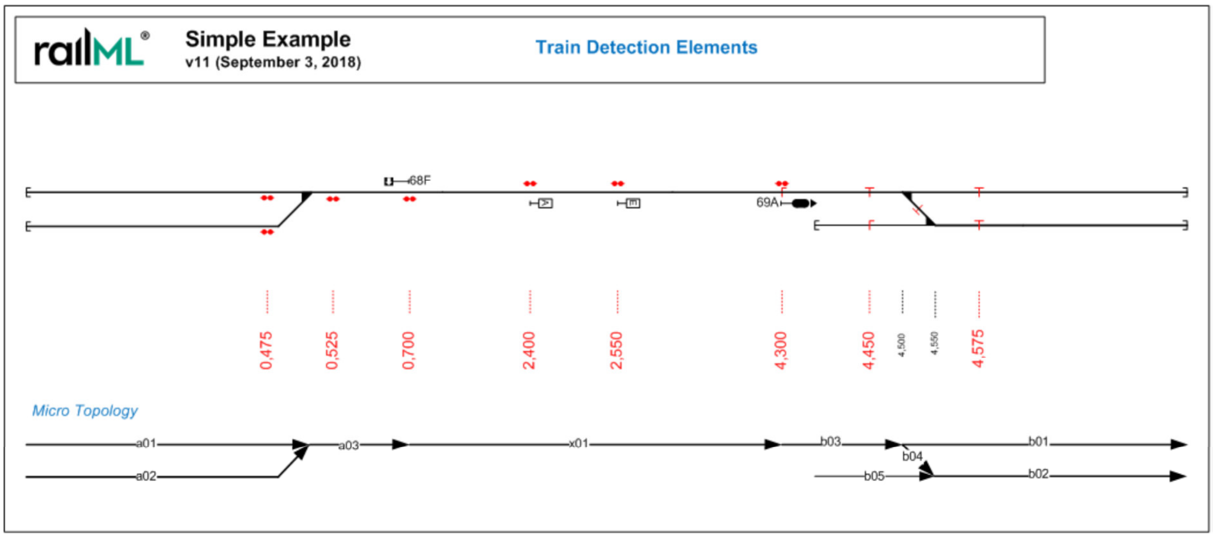
```
    <functionalInfrastructure>
      ...
      <signals>
        ...
        <signal id="sig09" type="speed" switchable="false">
          <spotLocation id="sig09_sloc01" netElementRef="ne_a03" 
applicationDirection="normal" pos="100.0">
            <linearCoordinate positioningSystemRef="lps01" measure="600.0" 
lateralDistance="2.2" lateralSide="right"/>
          </spotLocation>
          <isSpeedSignal signalKind="execution" trainRelation="headOfTrain">
            <refersToBeginOfSpeedSection ref="sps01"/>
          </isSpeedSignal>
        </signal>
        <signal id="sig10" type="speed" switchable="false">
          <spotLocation id="sig10_sloc01" netElementRef="ne_b03" 
applicationDirection="reverse" pos="100.0">
            <linearCoordinate positioningSystemRef="lps01" measure="4400.0" 
lateralDistance="2.2" lateralSide="left"/>
          </spotLocation>
          <isSpeedSignal signalKind="execution" trainRelation="headOfTrain">
            <refersToBeginOfSpeedSection ref="sps02"/>
          </isSpeedSignal>
        </signal>
        <signal id="sig11" type="speed" switchable="false">
          <spotLocation id="sig11_sloc01" netElementRef="ne_x01" 
applicationDirection="normal" pos="1300.0">
            <linearCoordinate positioningSystemRef="lps01" measure="2000.0" 
lateralDistance="2.2" lateralSide="right"/>
          </spotLocation>
          <isSpeedSignal signalKind="announcement" trainRelation="headOfTrain">
            <refersToBeginOfSpeedSection ref="sps03"/>
          </isSpeedSignal>
        </signal>
        <signal id="sig12" type="speed" switchable="false">
          <spotLocation id="sig12_sloc01" netElementRef="ne_x01" 
applicationDirection="normal" pos="1700.0">
            <linearCoordinate positioningSystemRef="lps01" measure="2400.0" 
lateralDistance="2.2" lateralSide="right"/>
          </spotLocation>
          <isSpeedSignal signalKind="execution" trainRelation="headOfTrain">
```

```
            <refersToBeginOfSpeedSection ref="sps03"/>
          </isSpeedSignal>
        </signal>
        <signal id="sig13" type="speed" switchable="false">
          <spotLocation id="sig13_sloc01" netElementRef="ne_x01"
applicationDirection="normal" pos="1850.0">
            <linearCoordinate positioningSystemRef="lps01" measure="2550.0"
lateralDistance="2.2" lateralSide="right"/>
          </spotLocation>
          <isSpeedSignal signalKind="execution" trainRelation="endOfTrain">
            <refersToEndOfSpeedSection ref="sps03"/>
          </isSpeedSignal>
        </signal>
      </signals>
      ...
    </functionalInfrastructure>
```

Please note: the actual speed information is not provided with the speed signal elements, but with the `<speedSection>` elements that have been already described in section 0. Each speed signal references the begin or the end of a related `<speedSection>` element via the child elements `<refersToBeginOfSpeedSection ref=””/>` and `<refersToEndOfSpeedSection ref=””/>`. Generally speaking, there must not be a speed signal that does not reference a related `<speedSection>` element.

3.11.2 Train Detection Elements

After having placed the signals, let’s have a look at the **train detection elements** that are interesting for track-side determination of railway vehicles. The Simple Example realizes two different systems for train detection: track circuits in “Bf Cstadt” and axle counting circuits in “Bf Arnau”. Both systems border on each other at line kilometer 4.300 where an axle counter is installed as well as an insulated joint. The following figure depicts the situation:



railML 3.1 allows the implementation of both, point based train detection objects (like axle counters) and linear train detection objects (like track circuits). Every `<trainDetectionElement>` should define its train detection system via the enumeration attribute `@type`. The most common values are:

"axleCounter", "axleCountingCircuit", "insulatedRailJoint" and "trackCircuit" The resulting code snippet for the Simple Example looks like this:

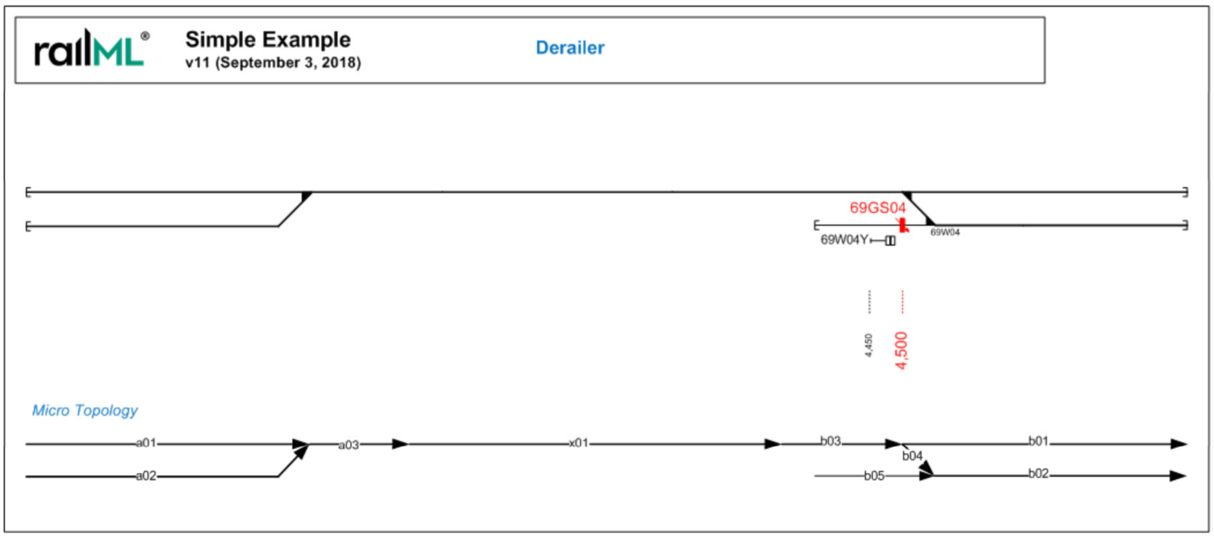
```
    <functionalInfrastructure>
      ...
      <trainDetectionElements>
        <trainDetectionElement id="tde01" type="axleCounter">
          <spotLocation id="tde01_sloc01" netElementRef="ne_a01"
applicationDirection="both" pos="475.0">
          </spotLocation>
        </trainDetectionElement>
        <trainDetectionElement id="tde02" type="axleCounter">
          <spotLocation id="tde02_sloc01" netElementRef="ne_a02"
applicationDirection="both" pos="475.0">
          </spotLocation>
        </trainDetectionElement>
        <trainDetectionElement id="tde03" type="axleCounter">
          <spotLocation id="tde03_sloc01" netElementRef="ne_a03"
applicationDirection="both" pos="25.0">
          </spotLocation>
        </trainDetectionElement>
        <trainDetectionElement id="tde04" type="axleCounter">
          <spotLocation id="tde04_sloc01" netElementRef="ne_a03"
applicationDirection="both" pos="200.0">
          </spotLocation>
        </trainDetectionElement>
        <trainDetectionElement id="tde05" type="axleCounter">
          <spotLocation id="tde05_sloc01" netElementRef="ne_b03"
applicationDirection="both" pos="0.0">
          </spotLocation>
        </trainDetectionElement>
        <trainDetectionElement id="tde06" type="insulatedRailJoint">
          <spotLocation id="tde06_sloc01" netElementRef="ne_b03"
applicationDirection="normal" pos="0.0">
          </spotLocation>
        </trainDetectionElement>
        <trainDetectionElement id="tde07" type="insulatedRailJoint">
          <spotLocation id="tde07_sloc01" netElementRef="ne_b03"
applicationDirection="both" pos="150.0">
          </spotLocation>
        </trainDetectionElement>
        <trainDetectionElement id="tde08" type="insulatedRailJoint">
          <spotLocation id="tde08_sloc01" netElementRef="ne_b01"
applicationDirection="both" pos="75.0">
          </spotLocation>
        </trainDetectionElement>
        <trainDetectionElement id="tde09" type="insulatedRailJoint">
          <spotLocation id="tde09_sloc01" netElementRef="ne_b04"
applicationDirection="both" pos="25.0">
          </spotLocation>
        </trainDetectionElement>
        <trainDetectionElement id="tde10" type="insulatedRailJoint">
          <spotLocation id="tde10_sloc01" netElementRef="ne_b02"
applicationDirection="both" pos="25.0">
          </spotLocation>
        </trainDetectionElement>
        <trainDetectionElement id="tde11" type="insulatedRailJoint" detectedObject="axle">
          <spotLocation id="tde11_sloc01" netElementRef="ne_b05"
applicationDirection="normal" pos="100.0">
          </spotLocation>
        </trainDetectionElement>
        <trainDetectionElement id="tde12" type="axleCounter">
          <spotLocation id="tde12_sloc01" netElementRef="ne_x01"
applicationDirection="both" pos="1700.0">
```

```
            <linearCoordinate positioningSystemRef="lps01" measure="2400.0"
lateralSide="left"/>
          </spotLocation>
        </trainDetectionElement>
        <trainDetectionElement id="tde13" type="axleCounter">
          <spotLocation id="tde13_sloc01" netElementRef="ne_x01"
applicationDirection="both">
            <linearCoordinate positioningSystemRef="lps01" measure="2550.0"
lateralSide="left"/>
          </spotLocation>
        </trainDetectionElement>
      </trainDetectionElements>
    </functionalInfrastructure>
```

Please note: The (logical) linking between train detection elements, signals and train protection elements is not part of the railML infrastructure scheme. These references are implemented in the railML interlocking schema (see tutorial part 2).

3.11.3 Derailer

The Simple Example also contains a **derailer**, which shall protect the train runs in “Bf Cstadt” from and to the southern track by derailing vehicles coming from the siding track before they can enter the loading gauge profile of the other track. The following figure depicts the situation in “Bf Cstadt”:



The `<derailer>` is modelled as functional infrastructure element that should be located on microscopic (track) topology level. The attribute `@applicationDirection` defines the `<netElement>` related direction of travel where the vehicle is derailed. The attribute @derailSide defines the direction to which the train running over this derailer will be derailed. The value of `@derailSide` thus directly depends on the `@applicationDirection`. From infrastructure point of view the `<derailer>` element has a limited functionality, but the element will become interesting together with the implementation of the interlocking scheme and related use cases.

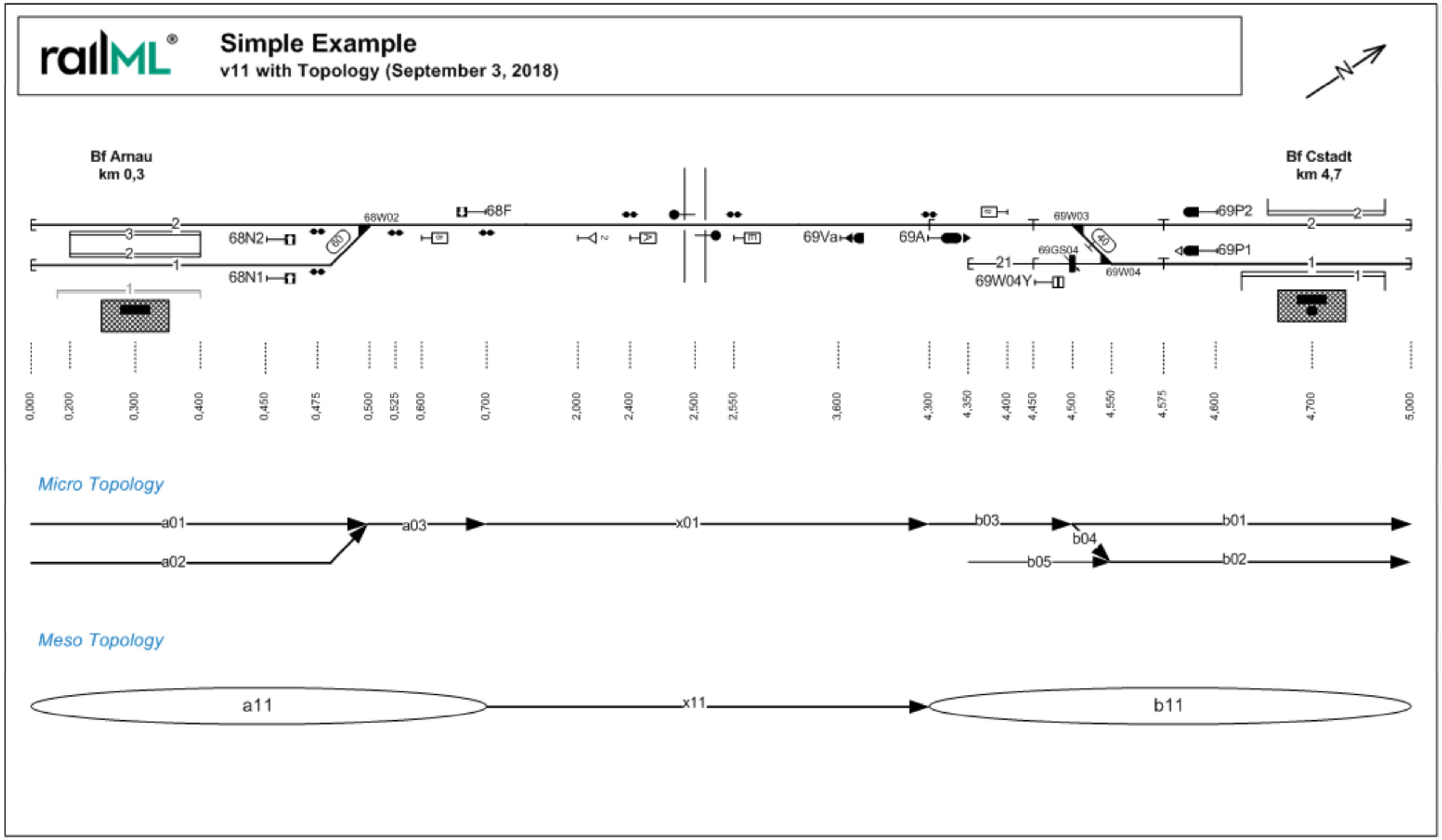
```
    <functionalInfrastructure>
      ...
      <derailers>
        <derailer id="der01" derailSide="right">
```

```
          <spotLocation id="der01_sloc01" netElementRef="ne_b05" 
applicationDirection="normal" pos="150.0">
          </spotLocation>
        </derailer>
      </derailers>
      ...
    </functionalInfrastructure>
```

What you may have learned
• Signals are modelled as `<signal>` elements. • `<signal>` also comprises marker boards and panels, which can show only one signal aspect ("panels"). • Speed signals are placed at the points where maximum permitted speed changes (begin or end of a `<speedSection>`. • The `<speedSection>` element also holds the actual speed limit information. Thus, `<signal>` elements which address speed information most often reference a related `<speedSection>` element. • `<trainDetectionElement>` models train detection elements. These can be both point based (e.g. axle counters) and linear (e.g. track circuits). • Train detection systems are defined through `<trainDetectionElement>@type`. Common values are "`axleCounter`", "`axleCountingCircuit`", "`insulatedRailJoint`" and "`trackCircuit`". • Derailers are modelled as `<derailer>` elements. `<derailer>` is part of the functional infrastructure, and should be located on the microscopic topology level.

4 Complete Example

The following figure depicts the complete picture of the Simple Example that has been implemented step-by-step in this application guide. The complete source code of the Simple Example is available as valid railML 3.1 file on the railML website [2]. There, you may also find further “national dialects” of the Simple Example where different national symbols have been used.


railML®
Simple Example
v11 with Topology (September 3, 2018)
N
Bf Arnau
km 0,3
Bf Cstadt
km 4,7
68W02
68F
68N2
68N1
69Va
69A
69W03
69GS04
69W04
69W04Y
69P2
69P1
21
0,000
0,200
0,300
0,400
0,450
0,475
0,500
0,525
0,600
0,700
2,000
2,400
2,500
2,550
3,600
4,300
4,350
4,400
4,450
4,500
4,550
4,575
4,600
4,700
5,000
Micro Topology
a01
a02
a03
x01
b03
b04
b05
b01
b02
Meso Topology
a11
x11
b11

5 References

[1] railML.org: *Website*. https://www.railml.org/en/; last access: 17.11.2017

[2] railML.org: *railML example data*. https://www.railml.org/en/user/exampledata.html; last access: 07.11.2018

[3] railML.org: *railML Forum*. http://forum.railml.org/; last access: 04.12.2017.

[4] railML.org: *Trac ticket system*. http://trac.railml.org/; last access: 04.12.2017.

[5] railML.org: *railML Wiki – Main Page*. http://wiki.railml.org/index.php?title=Main_Page; last access: 04.12.2017

[6] RTM Expert Group: *RailTopoModel*. http://www.railtopomodel.org/en/; last access: 27.11.2017

[7] The Railway Technical Website: *Infrastructure*. http://www.railway-technical.com/infrastructure/; last access: 27.11.2017

[8] Spatial Reference: *spatial reference list*. http://spatialreference.org/ref/epsg/; last access: 04.12.2017

[9] tutorialspoint: *XML Tutorial*. https://www.tutorialspoint.com/xml/index.htm; last access: 27.11.2017

[10] UIC: *UIC International Railway Standard IRS 30100 – RailTopoModel. Railway infrastructure topological model*. Version 1.0, 18.04.2016

[11] w3schools.com: *XML Tutorial*. https://www.w3schools.com/xml/default.asp; last access: 27.11.2017

[12] webucator: *XML Free Tutorial*. https://www.webucator.com/tutorial/learn-xml/index.cfm; last access: 27.11.2017