

# **railML® 3.1 Tutorial**

## **Simple Example Step-by-Step**

### **Part 2: Interlocking**

#### **Revision History**

Version	Date	Description	Name
0.1	13.05.2018	Initial Version	Jörg von Lingen
1.0	10.11.2018	Adaptation to railML3.1 (Release Candidate)	Jörg von Lingen
1.1	10.12.2018	Revision according to release of railML3.1 schema	Jörg von Lingen
1.2	11.12.2018	Replace "point" by "switch"	Jörg von Lingen
1.3	19.02.2019	Revision after schema refactoring of railML3.1	Jörg von Lingen

## Table of Content

<b>1</b>	<b><i>Interlocking schema of railML3</i></b>	<b>4</b>
1.1	Overview	4
1.2	Modelling Patterns	5
1.3	Common Types	5
1.4	Simple Example	6
1.5	Syntax Guide	7
1.6	Open Issues	7
<b>2</b>	<b><i>Infrastructure Manager dependent Data</i></b>	<b>8</b>
2.1	GenericIM	8
2.2	Signal Aspects	8
2.3	Reset of TVD Sections	11
2.4	Route Types	12
2.5	Level Crossing Types	13
2.6	Element Grouping	14
2.7	Detector Types	15
<b>3</b>	<b><i>Track and signalling components</i></b>	<b>15</b>
3.1	AssetsForIL	15
3.2	TVD Sections	16
3.3	Moveable Elements	17
3.3.1	SwitchIL	19
3.3.2	DerailerIL	20
3.3.3	MovableCrossing	21
3.3.4	Special Examples	22
3.3.4.1	Simple Crossing	22
3.3.4.2	Double Slip Switch	26
3.3.4.3	Single Slip Switch	28
3.3.4.4	Coupled Switchs	30
3.3.4.5	Dependency with Derailer	31
3.4	LevelCrossingIL	32
3.4.1	Level crossing activation	33
3.4.2	Level crossing deactivation	34
3.4.3	Complete Level Crossing	34
3.5	SignallIL	35
3.6	Logical Devices	37
3.6.1	KeyLockIL	37
3.6.2	Key	39
3.6.3	GenericDetector	39
3.7	ATP Devices	40
3.8	Restricted Areas	40

3.8.1	WorkZone .....	40
3.8.2	LocalOperationArea .....	41
3.8.3	ShuntingArea .....	42
3.8.4	PermissionZone .....	43
<b>4</b>	<b>Route .....</b>	<b>43</b>
4.1	Definition .....	43
4.2	State Space .....	46
4.2.1	AssetAndState .....	46
4.2.2	AssetAndGivenState .....	46
4.3	RouteEntry .....	47
4.4	Route activation .....	48
4.5	PartialRoute .....	48
4.5.1	RouteReleaseGroupAhead .....	49
4.5.2	RouteReleaseGroupRear .....	49
4.6	RouteExit .....	50
4.6.1	DangerPoint .....	50
4.6.2	Overlap .....	51
4.6.3	OverlapRelease .....	52
4.7	Complete Route .....	53
4.8	RouteRelation .....	54
4.9	Control table .....	55
4.10	Conflicting Routes .....	55
4.11	Combined Routes .....	57
4.12	DestinationPoint .....	57
<b>5</b>	<b>SignalBox (Interlocking) .....</b>	<b>58</b>
5.1	ExtentOfControl .....	59
5.2	SignalPlan .....	60
5.3	Configuration .....	66
<b>6</b>	<b>Interfaces .....</b>	<b>67</b>
6.1	Generic I/O-Interface .....	67
6.2	Sample Interface for NOR Level Crossing .....	68
6.3	Interlocking Interface .....	69
<b>7</b>	<b>SystemAssets .....</b>	<b>71</b>
7.1	PowerSupplyIL .....	71
<b>8</b>	<b>Controllers .....</b>	<b>72</b>
<b>9</b>	<b>References .....</b>	<b>73</b>

# 1 Interlocking schema of railML3

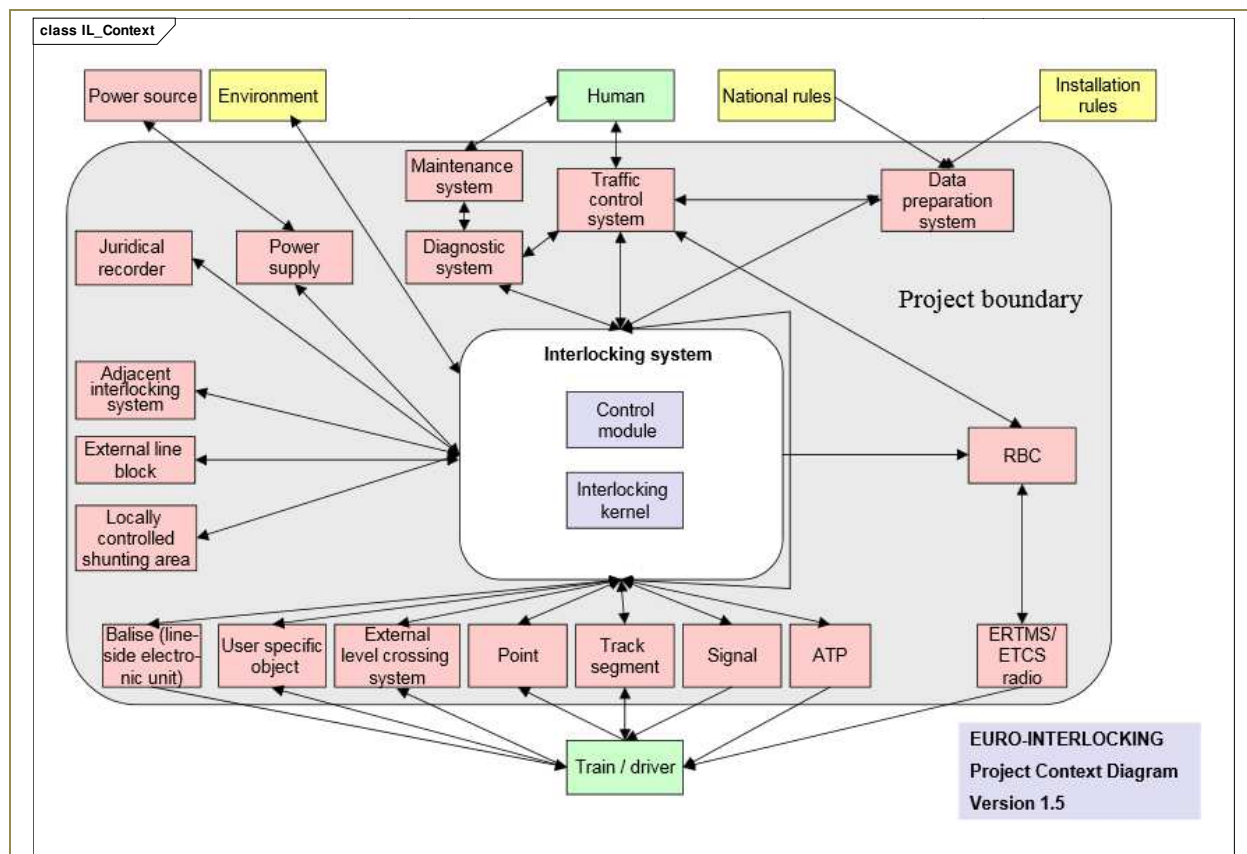
## 1.1 Overview

The railML interlocking subschema contains definitions of data describing railway signalling and the use of interlocking systems. Interlockings in strict sense are control systems using movable elements, signals, detectors, and other components in combinations and sequences that hinder collision and derailment of trains. Although the entire subschema is called “interlocking” the term “interlocking” is mainly used in this document for the equipment of the interlocking logic.

An interlocking description consists mainly of:

- Track and trackside signalling components,
- Route control logic,
- Interfaces,
- Control system, and
- Controllers.

Each of these categories are described in more detail in sections below.



## 1.2 Modelling Patterns

According to the modelling patterns agreed for railML the domain `<interlocking>` is divided in the views

- `<AssetsForIL>`
- `<Controllers>`
- `<SignalBoxes>`
- `<GenericIMs>`

These views have containers, except `<AssetsForIL>`, to collect data for objects of same type, i.e. such container cannot be empty if present but can take unlimited number of objects. Similar as the views and containers the majority of elements and attributes for the objects are optional, i.e. they shall be used only if the information is known. The XML feature of default values is not used within railML.

## 1.3 Common Types

The interlocking schema is modelled in UML with Enterprise Architect (EA). The model is designed in that way the tool can directly generate the XSD files. Thus no dependency on third party tools exists except EA.

The XSD generated from UML produces just one global element for the root class in the domain `<interlocking>`. All other classes/types are used with local elements only. This is known as the **Venetian Blind design** pattern. Because the majority of elements are not mandatory in the schema the factoring of railML into separate files containing only a small extract from the whole tree is possible. However, any references may be formally not checkable by the schema validator in such cases. The particular use of elements and attributes is to be defined in associated use case descriptions.

Cross-referencing between objects is done through ID→IDREF constructs that underlines the importance of unique identifiers. Alternatively the use of UUID for references is also possible. Despite the option to select an arbitrary amount of elements for an XML snippet the data of one project shall be stored within one XML file in normal case having the root `<railML>` and the domain `<interlocking>`. There might be other domains present in the same file but this does not affect the following description. At least the `<infrastructure>` domain is needed for reference targets from `<interlocking>` elements.

Within the schema the base types as defined in the common module (common3.xsd) are used. In addition two basic types were created for use in interlocking subschema – `EntityIL` and `EntityILref`. They give the elements in the subschema a standard set of features. The first type `EntityIL` gives every derived type the attribute `@id` and the child element `<designator>` with the attributes `@entry` and `@register`. The common type `<designator>` is used in railML to allow the attribution of the typical object identifiers of railway world. These designators often use a naming code as defined in the associated register. They can be in normal language but are not having different expressions for one particular object that is dependent on the language. In fact these are a kind of “human readable id”. The mandatory attributes `@entry` and `@register` take the particular identifying name and the name of the associated register used. In case there is no officially used register the project-specific one shall be named with a leading underscore. The `@id`

attribute is the “machine readable id” of the item. Although it can make use of the coded name it is not limited to. However, it has to follow the syntax of ID in XML-context.

In addition the type `EntityIL` allows the possibility of user-defined extensions with `@anyAttribute` and anchors for `<anyElements>`. The latter one requires a user-defined namespace with the associated schema provided, i.e. `processContents="strict"`. The second type `EntityILref` is the counterpart providing just the attribute `@ref` to point to the `@id` of another item plus the possibility for `@anyAttribute`.

Having the possibility for `@anyAttribute` at almost every element gives one the opportunity to add specially agreed information as bespoke features without violating the railML schema. For additions that are more extensive, the anchors for `<anyElements>` are provided. Last but not least some of the enumerations in the domain `<interlocking>` offer the use of “other:...” entries.

There is only one additional rule with `any`-extensions in railML. They shall not be used for information, which can be represented with the features of railML.

## 1.4 Simple Example

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

The learning of the features of the freshly developed railML schema is best with an example using them. This example has to be simple to allow easy manually checks and reviews. With this purpose in mind, a small railway network was designed comprising major railway components. From the graphical version the infrastructure was deducted as XML file. This was the basis for building up a similar XML file with the interlocking features.

The drawing as used for the infrastructure was enhanced by the yellow boxes containing the important ID from the infrastructure elements. The enhancement was purely for convenience as all these ID were needed to use the correct references. Although rather cryptic values could be used as ID a “speaking” style was preferred to not complicate manual work with the example. In later use with software applications the ID could be even UUID or similar.



## 2 Infrastructure Manager dependent Data

### 2.1 GenericIM

The very first step for setting up an interlocking file is the definition of the generic types. Although virtually depicted in the bottom of the schema these generic types come first as they are referenced by the more detailed data later on. The container `<specificIMs>` takes the data of the individual `<specificIM>`. It shall be noted although the type is named `GenericIM` the instantiation as element is named `<specificIM>`.

Each infrastructure manager (IM) has a couple of rather specific items, which have impact on interlocking issues and functions. In order to achieve a common exchange format it is vital to define such specific items within a fixed generic structure still allowing the individual characteristics. This is the purpose of the generic types. They are providing a mean of common classification of functions being rather individual for any IM. However, these types do not specify the operational rules of that IM.

Beside the `<designator>` and `@id` it comprises a reference to the related set of assets in `<ownsSetsOfAssets>`.

```
<specificIM id="BaneNor">
  <designator register="„SimpleRegister“ entry="BaneNor (JBV)"/>
  <ownsSetsOfAssets ref="ass_simpex_v0.9"/>
  .....
</specificIM>
```

The object `<specificIM>` is filled with the data of items specific for this IM. Such specific items are typical items like signal aspects or route handling.

### 2.2 Signal Aspects

Each IM has clearly his own set of signal aspects he uses for controlling train traffic. They are defined in `<hasAspect>`. Beside these individual characteristics, there are some common principles, which are considered here. At first the wide range of signal aspects can be categorised in several groups for the description of their meaning – the so-called `@genericAspect`. There are the following possibilities on the list:

- **“closed”** – This is used for any aspect with the meaning “Stop here”.
- **“callOn”** – This is used for any auxiliary aspect with the meaning “Pass at reduced speed with clear visibility over the route ahead” because the signal cannot be cleared normally. In most cases such aspect is used with a special call-on route.
- **“caution”** – This is used for an announcing aspect/slave aspect with the meaning “expect Stop” at next signal.
- **“warning”** – This is used for an announcing aspect/slave aspect with the meaning “expect any kind of proceed” at next signal.
- **“proceed”** – This is used for any aspect indicating the allowance to continue running without any speed restrictions, i.e. proceed with line speed. However, such aspect can be combined on a signal with a speed indicator restricting the allowed speed against the main aspect.
- **“limitedProceed”** – This is used for any aspect indicating the allowance to continue running with restricted speed. This is typically used for diverging routes or ones with reduced

braking distance. In addition this main aspect might be combined with a speed indicator restricting or relaxing the allowed speed against the main aspect.

- **“combinedProceed”** – This is used for any proceed aspect where the master and the slave aspect is combined within one single aspect like this were common in OSShD<sup>1</sup> networks. Of course, this applies only to proceed aspects as with the signal closed no slave aspect is given.
- **“supplementary”** – These are any additional signal aspects which are combined with the main aspect without causing a restriction or giving pure information. Such combination shall be supervised by the interlocking and a failure will affect the main aspect as well. A good example is an additional indicator announcing the change onto the wrong track, i.e. line track normally used in the opposite direction.
- **“restriction”** – This aspect gives an additional restriction to the main aspect. A failure of such aspect will affect the main aspect of the signal. An example would be a speed indicator restricting the main proceed aspect.
- **“informative”** – In contrast to supplementary aspects they are giving pure information without any consequences neither to the main aspect nor the train traffic. A failure of this aspect would not affect the main aspect. An example for an informative aspect is any aspect from a direction indicator. It can be also a speed indication if it is relaxing the speed information of the main aspect.

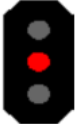


In addition to the generic meaning, the aspect shall have a specific naming in **<designator>** and **@id** to refer to it.

The list of three main aspects for IM “BaneNor” would look like this in railML:

```
<hasAspect id="sig_closed_20" genericAspect="closed">
  <designator register="_SimpleRegister" entry="Signal 20A/B «Stopp" />
</hasAspect>
<hasAspect id="sig_reducproceed_21" genericAspect="limitedProceed">
  <designator register="_SimpleRegister" entry="Signal 21 «Kjør med redusert hastighet" />
</hasAspect>
<hasAspect id="sig_fullproceed_22" genericAspect="proceed">
  <designator register="_SimpleRegister" entry="Signal 22 «Kjør" />
</hasAspect>
```

Just for illustration an extract from the operator’s manual (togframføringsforskriften) is given here including the optical appearance which is not included in **@genericAspect**. The mapping between the aspect and the activated lamps might be done with another **genericType** but this is currently not yet implemented.

<sup>1</sup> Organization for Cooperation of Railways as the equivalent of the International Union of Railways (UIC)




<i>Signal</i>	<i>Signalnummer og signalnavn</i>	<i>Signalbetydning</i>
Et rødt fast lys. Eksempel: 	Signal 20B «Stopp»	Stopp senest ved innkjørtogveiens slutt.
Et grønt fast lys. Eksempel: 	Signal 21 «Kjør med redusert hastighet»	Tog kan kjøre ut fra eller passere stasjonen med redusert hastighet i henhold til bestemmelsene i kapittel 6.
To grønne faste lys på loddrett linje. Eksempel: 	Signal 22 «Kjør»	Tog kan kjøre ut fra eller passere stasjonen med største tillatte hastighet.

The related distant signal aspects would look like this in railML:

```

<hasAspect id="sig_caution_23" genericAspect="caution">
  <designator register="_SimpleRegister" entry="Signal 23 «Forvent stopp»"/>
</hasAspect>
<hasAspect id="sig_warning_24" genericAspect="warning" >
  <designator register="_SimpleRegister"
    entry="Signal 24 «Forvent kjør med redusert hastighet»"/>
</hasAspect>
<hasAspect id="sig_warning_25" genericAspect="warning">
  <designator register="_SimpleRegister" entry="Signal 25 «Forvent kjør»"/>
</hasAspect>

```

<i>Signal</i>	<i>Signalnummer og signalnavn</i>	<i>Signalbetydning</i>
Et gult blinkende lys. Eksempel: 	Signal 23 «Forvent stopp»	Toget skal redusere hastigheten slik at toget kan stoppe foran hovedsignalet. Tilhørende hovedsignal viser signal 20A eller 20B «Stopp».
Et gult og et grønt blinkende lys på loddrett linje. Eksempel: 	Signal 24 «Forvent kjøør med redusert hastighet»	Toget skal redusere hastigheten. Tilhørende hovedsignal viser signal 21 «Kjør med redusert hastighet».
Et grønt blinkende lys. Eksempel: 	Signal 25 «Forvent kjøør»	Toget kan fortsette med største tillatte hastighet. Tilhørende hovedsignal viser signal 22 «Kjør».

## 2.3 Reset of TVD Sections

Railway tracks are divided in logical sections by technical means for detecting the vacancy of the related section. The typical technologies are track circuits or axle counter. They deliver mainly the status information “vacant”, “occupied” or “failed”. In case of a failure the particular sections needs to be reset, i.e. declared clear of any railway vehicle. The individual IM have various strategies for such a reset. In consequence the interlocking can attribute a particular TVD section with the necessary commands. The functional details are specific of the equipment manufacture and not needed here. Thus, a generic classification shall be sufficient. The list of TVD reset strategies in `<hasTVDresetStrategy>` allows the following possibilities:

- “`unconditionalReset`” – The section will be reset on command without any conditions, i.e. after the reset command was accepted by the interlocking system the TVD section is set to clear/vacant status.
- “`conditionalReset`” – The section will only be reset, if some conditions are fulfilled. In case of several variants one can include the condition information in the `<designator>` of the reset strategy. Such conditions can be that the TVD section must be failed or the last counting action of delimiting detection points was “in”.
- “`sweeprunWithoutConfirmation`” – Especially for axle counter sections a sweep run (*de: Bügelfahrt*) can be requested. Although the TVD section reset takes place, with receipt of the operator command to the axle counter unit the interlocking keeps it logically failed until a correct occupation and clearance sequence from a passing train is notified from the axle counter. In this case the section is reset immediately without any further operator action.

- **“sweepRunWithConfirmation”** – Here a sweep run is also requested. However, after the notification of the correct occupation and clearance sequence the operator needs to confirm the result by command before the section is reset within the interlocking.
- **“procedure”** – This is a placeholder for any other procedure to reset a TVD section.

The definition here in `<hasTVDresetStrategy>` is rather straight forward, as one would give an `@id` and possibly a `<designator>` related to the used value from `GenericResetStrategyList`.

```
<hasTVDresetStrategy id="rst_uc" resetStrategy="unconditionalReset">
  <designator register="_SimpleRegister" entry="unconditional reset"/>
</hasTVDresetStrategy>
<hasTVDresetStrategy id="rst_cd" resetStrategy="conditionalReset">
  <designator register="_SimpleRegister" entry="conditional reset"/>
</hasTVDresetStrategy>
<hasTVDresetStrategy id="rst_swr_noconf" resetStrategy="sweepRunWithoutConfirmation">
  <designator register="_SimpleRegister" entry="sweeprun without confirmation"/>
</hasTVDresetStrategy>
<hasTVDresetStrategy id="rst_swr_conf" resetStrategy="sweepRunWithConfirmation">
  <designator register="_SimpleRegister" entry="sweeprun with confirmation"/>
</hasTVDresetStrategy>
```

## 2.4 Route Types

An interlocking has to handle different route types whose characteristics are defined by the IM. The details are given in the IM's rule book but are not subject of the data exchange. The generic classification allows the interlocking to attribute the route with a particular set of commands and a set of features, e.g. possible occupation within the running path.

The list of generic route types offers the following possibilities:

- **“callOn”** – This is a special route type for situations, where the interlocking cannot fully secure the route for a train but safety is established by operator rules.
- **“nonElectrified”** – This is the route type for securing the path of the trains in the standard way but allows the running path to be without electrification, i.e. it shall be used for diesel trains only.
- **“normal”** – This is the route type for securing the path of the trains in the standard way.
- **“occupied”** – The route leads to an already occupied TVD section, i.e. the destination tracks must be occupied.
- **“siding”** – This is a normal route onto the open line, which is destined for entering a key locked siding on that line.
- **“shunting”** – Such routes are typically used for splitting or merging formations of railway vehicles with a reduced amount of safety as the train movement is performed on low speed under responsibility of the driver. Parts of the running path can be occupied.
- **“tunnel”** – The route leads to a tunnel section, which has restrictions w.r.t. vehicle acceptance, i.e. the vehicles shall be equipped with emergency brake cancellation.
- **“other:... ”** – The route is of another type. This is the optional extension of the list. Each entry needs to start with the string “other:” and shall have at least two letters in addition.

The extract from the example shows the definition of a standard main route type and a route type for shunting purpose. The selection of route type is an indication for the interlocking for the related commands for route setting and release.

```
<hasRouteType id="rt_main" genericRouteType="normal">
  <designator register="_SimpleRegister" entry="normal main route for trains"/>
</hasRouteType>
<hasRouteType id="rt_shunt" genericRouteType="shunting" >
  <designator register="_SimpleRegister" entry="shunting route"/>
</hasRouteType>
```

## 2.5 Level Crossing Types

An IM uses normally a couple of standard types of level crossing equipment. They may have more individual interfaces but the major difference for interlocking is depending on the way of control. In the generic type description the following information can be included:

- **@isControlType** – This describes the way the interlocking controls the level crossing in relation to routes.
  - **"autonomous"** – The level crossing is acting autonomous from interlocking in terms of activation and deactivation. It gives only feedback of its state to the interlocking. This is normally the way of control for level crossings on the open block line not in proximity of stations or operational control points.
  - **"halfControlled"** – The level crossing is at least activated from interlocking only for one direction. Deactivation for this direction may be also commanded from interlocking. The remaining functions are autonomous. This applies normally for level crossings on open block line but in proximity of a station. Then the exit direction from the station is controlled from the interlocking.
  - **"fullControlled"** – The level crossing is activated and deactivated only from interlocking independently of the direction. This is the case for level crossings within stations. The level crossing is activated by route setting and deactivated when the underlying TVD section is released from route.
- **@allowsLocalOperation** – This flag indicates whether the level crossing has means for being operated locally independent from the interlocking.
- **@hasBarrier** – This flag indicates whether the level crossing has physical barriers to block road traffic. This may influence the way the interlocking is operating the level crossing and the amount of feedbacks.
- **@hasTrafficWarning** – This flag indicates whether the level crossing has means to indicate its status to road traffic, e.g. traffic warning lights. This may influence the way the interlocking is operating the level crossing and the amount of feedbacks.

The extract shows a level crossing equipped with barriers and traffic signals. It can be operated from onsite and is not activated or deactivated independent from the interlocking.

```
<hasLevelCrossingType id="lxt01" controlType="fullControlled" allowsLocalOperation="true"
  hasBarrier="true" hasTrafficWarning="true" >
  <designator register="_SimpleRegister" entry="both sided controlled LX"/>
</hasLevelCrossingType>
```

## 2.6 Element Grouping

Dependent on the operating rules of an infrastructure manager there are often groups of elements known to the interlocking that can be used to perform one operation to all members of the group at once. This grouping can have one of the following basic classifications:

- “automaticRouteSetting” – Group of signals for activating and deactivating the ARS mode for all elements within the group.
- “automaticTrainRouting” – Group of signals for activating and deactivating the ATR mode for all elements within the group.
- “callOn” – Group of signals for which the activation of call-on aspect is allowed simultaneously. This is used if no particular routes are required for call-on aspect.
- “catenary” – Group of elements which are related to one electrical section of catenary. Although the catenary switch status is normally not commanded from the interlocking, its status may be indicated and considered in interlocking. The related elements shall be reference to TVD sections as representation of track sections, switches etc.
- “emergencyStop” – Group of signals which can be commanded to stop with one special command.
- “elementBlocking” – Group of same type elements that can be blocked or unblocked from operation.
- “pointHeating” – Group of switches for which the point heating devices are controlled together.
- “pointStaggering” – Group of switches for which the point staggering is activated or deactivated together. The point staggering allows the automatic switching of switches if requested by any route.
- “other:...” – The element group is of another type. This is the optional extension of the list. Each entry needs to start with the string “other:” and shall have at least two letters in addition.

The example shows the definition of a signal emergency stop group.

```
<hasElementGroupType id="estop01" elementGroupType="emergencyStop">
  <designator register="_SimpleRegister" entry="Emergency Stop 01"/>
</hasElementGroupType>
```

The use of such an emergency stop group within `<signalBox>` is shown here. The group contains all main signals of station ARN.

```
<implementsElementGroup id="estopARN">
  <designator register="_SimpleRegister" entry="Stop ARN"/>
  <groupType ref="estop01" />
  <refersToMember ref="mb_sig01" />
  <refersToMember ref="mb_sig02" />
  <refersToMember ref="mb_sig03" />
</implementsElementGroup>
```

## 2.7 Detector Types

Infrastructure managers use different types of detectors to monitor for unwanted or dangerous situations related to the interlocking operation. The outputs of such detectors are fed into the **SignalBox** (interlocking) or the **Controller** to react accordingly. Dependent on the purpose there can be the following base types classified:

- **“avalanche”** – The detector detects avalanches, which may endanger the railway traffic.
- **“cranks”** – The detector detects the presence of cranks for switch actuators at their normal location, e.g. in a special cabinet at the stationmaster.
- **“derailment”** – The detector detects any derailed railway vehicle. It is often used in rear of tunnels or bridges to reduce the damages by derailed vehicles.
- **“doors”** – The detector monitors the entry doors of equipment rooms.
- **“fire”** – The detector detects fire or smoke in equipment cabinets or rooms.
- **“flatWheel”** – The detector detects any flat wheel of a railway vehicle.
- **“gas”** – The detector detects the excessive concentration of a particular gas in the vicinity.
- **“hotWheelBox”** – The detector detects any hot axle box of railway vehicles.
- **“intrusion”** – The detector monitors the access doors to any equipment cabinet or room in order to detect unauthorised access.
- **“landSlide”** – The detector detects landslides or rockfalls, which may endanger the railway traffic.
- **“loadingGauge”** – The detector detects any railway vehicle exceeding the loading gauge due to replaced goods or similar.
- **“weighing”** – The detector checks the axle load of any railway vehicle against a pre-set limit.
- **“other:... ”** – The detector is of another type. This is the optional extension of the list. Each entry needs to start with the string “other:” and shall have at least two letters in addition.

## 3 Track and signalling components

### 3.1 AssetsForIL

The physical features of tracks and trackside components (“track assets”) are described in the infrastructure subschema, and these object’s identities are referenced from the interlocking schema. Examples of track assets are: signals, switches, and TVD sections. Whereas the infrastructure subschema focus on the characteristics of the hardware the interlocking subschema concentrate on the more logical characteristics relevant for interlocking issues. Thus the interlocking subschema adds more information to some infrastructure elements.

Within the entire railML schema there are no different namespaces used. Subsequently some elements (types) in interlocking subschema had to be renamed in order to avoid collision with already existing elements in infrastructure subschema. In such cases just the suffix “IL” was added to the name, e.g. **Signal** vs. **SignalIL**.

The element of `<AssetsForIL>` is not only a collection of track assets but including some interlocking system components ("system assets") and functional combinations like routes or restricted areas. Examples of system assets are anything not strictly related to track position like radio block centre, communication systems, power supply systems and diagnostic systems. The various kinds of assets have individual containers, which each hold an unlimited number of objects.

There can be only one collection of assets within one interlocking data set. Different states of network evolution would cause different objects due to requirements of uniqueness concerning ID.

## 3.2 TVD Sections

The track vacancy detection sections are a fundamental feature to monitor train positions and to ensure safety of railway traffic. Although different technologies are used for the detection itself the main features are similar for each of them. The proving of track vacancy is limited to a defined part in the track network. The limits are beside physical ones detection points or insulated rail joints (physical or virtual).

Each object `<tvdSection>` defines the characteristics of such a track section with the following details.

- `@isBerthingTrack` – This flag marks a section where trains normally would halt and change direction. Typically, an Interlocking assures that trains progress from section to section in an ordered sequence (aka. two/three phase release). This check would fail when a train changes direction. If this attribute is true, the interlocking does not carry out this check for this section.
- `@residualRouteCancellationDelay` – In case there was an unusual train run with not sequential track occupation the automatic release of the track section is not performed. The operator has to initiate this release by command. The timer is a safety precaution to make the presence of vehicles running into this section unlikely.
- `@partialRouteReleaseDelay` – In most of electronic interlocking track sections are automatically released after correct sequential occupation and clearance by the train. In order to reduce the possibility of false sequences, i.e. previous section clear before next one is occupied; a timer is introduced until the release process is performed.
- `@technology` – It marks the technology type of this TVD section. The possible values can be
  - `"axleCounter"` – The TVD section is formed by axel counter detection points.
  - `"trackCircuit"` – The TVD section is using track circuit equipment with insulated rail joints or virtual ones as delimiters.
  - `"other:..."` – The TVD section is made of another technology. This is the optional extension of the list. Each entry needs to start with the string "other:" and shall have at least two letters in addition.
- `@frequency` – This is the frequency in Hertz of a track circuit. For DC track circuits this shall be set to zero. Omitting this value shall be interpreted as not applicable or unknown.
- `<hasDemarcatingBufferstop>` – This is the reference to `<bufferStop>` the physical track ends like buffer stop as a limit of the TVD section.

- `<hasDemarcatingTraindetector>` – This is the reference to the limiting points between track sections used for train detection `<trainDetectionElement>`, e.g. axle counter point, insulated rail joint.
- `<hasResetStrategy>` – This is the reference to one of the defined reset strategies. The information is used for determining the related operator commands, which the interlocking shall handle for this TVD section.
- `<hasExitSignal>` – This is the reference to a signal which protects the exit from this TVD section. It shall be used in case `@isBerthingTrack` is set to `“true”`.

Within the simple example there are different kind of TVD sections. Here is a platform track with physical track end at one side and an axle counter detection point at its other side. It is obviously a berthing track in this case and it has an exit signal. The sweep-run without confirmation is chosen as section reset strategy.

```
<tvdSection isBerthingTrack="true" id="A01T" partialRouteReleaseDelay="PT1S"
  residualRouteCancellationDelay="PT90S" technology="axleCounter">
  <designator register="_SimpleRegister" entry="Arnau A01"/>
  <hasDemarcatingBufferstop ref="bus01" />
  <hasExitSignal ref="mb_sig02" />
  <hasDemarcatingTraindetector ref="tde01" />
  <hasResetStrategy ref="rst_swr_noconf" />
</tvdSection>
```

The next extract shows a TVD section with more than two ends because it represents a switch. The section limits are established by axle counter detection points. Therefore, the reset strategy and release timers are similar to the first extract. The relation to tracks in infrastructure includes all three legs of the switch because the tracks are defined until the virtual connection point of the `netElements`. This may be the crossing of tangents of both branches named point of intersection (de: *Weichenmitte*).

```
<tvdSection isBerthingTrack="false" id="A68W02T" partialRouteReleaseDelay="PT1S"
  residualRouteCancellationDelay="PT90S" technology="axleCounter">
  <designator register="_SimpleRegister" entry="Arnau pt68W02"/>
  <hasDemarcatingTraindetector ref="tde01"/>
  <hasDemarcatingTraindetector ref="tde02"/>
  <hasDemarcatingTraindetector ref="tde03"/>
  <hasResetStrategy ref="rst_swr_noconf" />
</tvdSection>
```

The last extract shows a TVD section using insulated rail joints for limitation. Therefore, the conditional reset and different release timer were chosen.

```
<tvdSection isBerthingTrack="false" id="B03T" partialRouteReleaseDelay="PT4S"
  residualRouteCancellationDelay="PT90S" technology="trackCircuit">
  <designator register="_SimpleRegister" entry="entry B03"/>
  <hasDemarcatingTraindetector ref="tde05"/>
  <hasDemarcatingTraindetector ref="tde06"/>
  <hasResetStrategy ref="rst_cd"/>
</tvdSection>
```

### 3.3 Moveable Elements

The class of moveable elements is an abstract, which is enriched by the attributes and elements of the particular type of moveable element like switch, derailer or crossing. In general movable

elements are assets physically driven by the interlocking, i.e. the position of the element is actuated by interlocking operation. The movable elements are already defined in infrastructure. Thus a reference to the related infrastructure element is a common part of all moveable elements.

Other common elements and attributes for each moveable element are:

- **<refersTo>** – This is the reference to the track asset from the infrastructure part, i.e. the particular **<switchIS>**, **<derailerIS>** or **<crossing>**.
- **<hasGaugeClearanceMarker>** – This is the reference to a related infrastructure element **<GaugeClearanceMarker>**. However, the position of the physical marker does not give direct information, which branch of the movable element is affected.
- **<hasTvdSection>** – This is the reference to the TVD section the movable element is part of. It can be only one TVD section related to a movable element whereas a TVD section might comprise several movable elements and track sections.
- **<connectedToPowerSupply>** – This is the reference to the power supply used to energise the switch actuators. The relation is needed for the interlocking to control the number of switch actuators that are activated simultaneously without overloading the power supply.
- **<relatedMovableElement>** – This is the reference to another movable element from the asset list which forms a functional pair with this one. The use is sensible mainly for switches and in few cases for derailleurs. The relation shall be given for both of the related elements. The kind of relation is derived from the physical features **@type** in **<switchIS>**. From the possible combinations the following functions can be concluded:
  - “ordinarySwitch/insideCurvedSwitch/outsideCurvedSwitch” without use of **<relatedMovableElement>** – This is just simple switch with two branches of diverging tracks and without functional speciality.
  - “ordinarySwitch/insideCurvedSwitch/outsideCurvedSwitch” with reference to a switch as **<relatedMovableElement>** – This is a pair of simple switches that are always operated together with one command thus keeping them in synchronised position under normal circumstances. This is feature, called “coupled switches”, is used for example on a crossover (connection between two parallel tracks).
  - Any value of **@type** with reference to a derailer as **<relatedMovableElement>** – This is used to ensure the dependency between the switch and derailer, e.g. that the switch is only going into the position towards the derailer when the derailer is in “passablePosition”. In such case the use of **<hasPositionRestriction>** is required.
  - “singleSwitchCrossing” – works on the same principle as a double slip, but provides for only one switching possibility. Trains approaching on one of the two crossing tracks can either continue over the crossing, or switch tracks to the other line. However, trains from the other track can only continue over the crossing, and cannot switch tracks. With this type the **<relatedMovableElement>** shall refer to the other half of the single slip switch. Due to the modelling of single slip switches within the interlocking the use of **<hasPositionRestriction>** is required.
  - “doubleSwitchCrossing” – A narrow-angled diagonal flat crossing of two lines combined with four pairs of switches in such a way as to allow vehicles to change from

one straight track to the other, as well as going straight across. A train approaching the arrangement may leave by either of the two tracks on the opposite side of the crossing. To reach the third possible exit, the train must change tracks on the slip and then reverse. With this type the `<relatedMovableElement>` shall refer to the other half of the double slip switch.

- `@typicalThrowTime` – This value gives the time the movable element normally takes to move from one end position the other. The information is used when calculating the time needed to setup routes or to release them again.
- `@maxThrowTime` – This value gives the time the interlocking should expect the moveable element has reached its end-position. Latest after expiration of this timer the engine power is switched off and a failure is raised if the moveable element does not report end-position.
- `@returnsToPreferredPosition` – The automatic normalisation flag is closely related to the preferred position. It defines whether the interlocking shall return the movable element to the defined preferred position after use, i.e. after route release.
- `@isKeyLocked` – This flag is used to identify that the switch is clamped either mechanically or by any electric or electronic means. The interlocking shall never attempt to throw a clamped switch.
- `@numberOfBladeSwitchActuators` – This is the number of switch actuators mounted to operate the blades of the switch or the derailer.
- `@numberOfFrogSwitchActuators` – This is the number of switch actuators mounted to operate the swinging frog<sup>2</sup> nose of the switch or crossing.

A special issue is the common element `<hasGaugeClearanceMarker>`. The gauge clearance marker is a sign or concrete slab that limits where a train can go whilst staying clear of a train on the other track. Although the interlocking is unaware of these markers, it should be taken in for the sake of modelling because this determines whether the TVD provides gauge clearance. The same information is more precisely given by `<hasFoulingTrainDetectors>`.

### 3.3.1 SwitchIL

The most common movable element in railway networks is a switch<sup>3</sup>. It is an assembly of rails, blades and of auxiliaries, certain ones being movable, which effect the tangential branching of tracks and allows running over one track or another. In order to avoid ambiguity the related type is called `SwitchIL` in railML interlocking subschema, whereas its counterpart in infrastructure is called `SwitchIS`. The switch is commonly used for interlockings to form also special types of switches like single/double slip switches which hardware is derived from a simple crossing.

The instantiation extends the abstract type of `MovableElement` by some additional elements and attributes.

<sup>2</sup> Dependent on the language variant there are several terms in use: common crossing, frog. railML will use the term "frog" to avoid ambiguity.

<sup>3</sup> Dependent on the language variant there are several terms in use: point, turnout, switch. railML will use the term „switch“.

- **<refersTo>** – This is the reference to the track asset **<switchIS>** from the infrastructure part.
- **<hasFoulingTrainDetectors>** – This is the reference to the train detectors delimiting the TVD section of this switch which are too close and cannot guarantee a clear gauge of the set track. A switch may also have a fouling train detector at its tip if the distance to the tip of blades is too short, i.e. a vehicle can move into the switch as the blades are moving.
- **<branchLeft>** – This is the reference to the track in infrastructure representing the left branch of the switch.
- **<branchRight>** – This is the reference to the track in infrastructure representing the right branch of the switch.
- **<hasPositionRestriction>** – The element is needed especially for single slip switches but can be used for other pairings as well. It expresses the enforced positions of the pair the interlocking must respect.
  - **<relatedElementInPosition>** – This is the reference to other movable element of the pair and its required position expressed as **AssetAndState**.
  - **@restrictedPosition** – This is the position of the movable element that can be steered into only if the other one is in the required position or steered into it at the same time in case of coupled switches.
- **@preferredPosition** – This marks the switch position that the switch shall have when not in use of a route. The possible positions can be “right” and “left” where right means the train would drive onto the right side looking from the tip of switch.

The extract gives the example of a simple switch for the interlocking part. It has a preferred position but it will not be automatically returned into this position. The throw timers are given in seconds.

```
<switchIL returnsToPreferredPosition="false" id="pt_swi01" isKeyLocked="false"
  maxThrowTime="PT10S" typicalThrowTime="PT6S" preferredPosition="right">
  <designator register="_SimpleRegister" entry="68W02"/>
  <refersTo ref="swi01"/>
  <branchLeft ref="trc02"/>
  <branchRight ref="trc01"/>
</switchIL>
```

The more special cases of coupled switches, single/double slip switch or the dependency with a derailer are considered in the special examples of chapter 3.3.4.

### 3.3.2 DerailerIL

Another movable element is the derailer. In order to avoid ambiguity with the infrastructure element the related type is called **DerailerIL** in railML interlocking subschema. The derailer is a device which, when placed on the rail, derails the wheels of a vehicle, and serves to protect a converging line. There are also versions made of one or two stock rail(s) with a movable blade, also called “trap switch”, causing the same effect. Independently of its physical appearance it is handled by the interlocking very similar to a switch. The instantiation extends the abstract type of **MovableElement** by some additional elements and attributes.

- **<refersTo>** – This is the reference to the track asset **<derailerIS>** from the infrastructure part.

- **@preferredPosition** – This marks the position, which the derailer shall have in normal case, i.e. when not in use of a route. The possible positions can be
  - **“derailingPosition”** – The derailer is engaged and no vehicle can pass it without getting purposely guided off the running rails.
  - **“passablePosition”** – The derailer is retreated and can be passed by any vehicle.

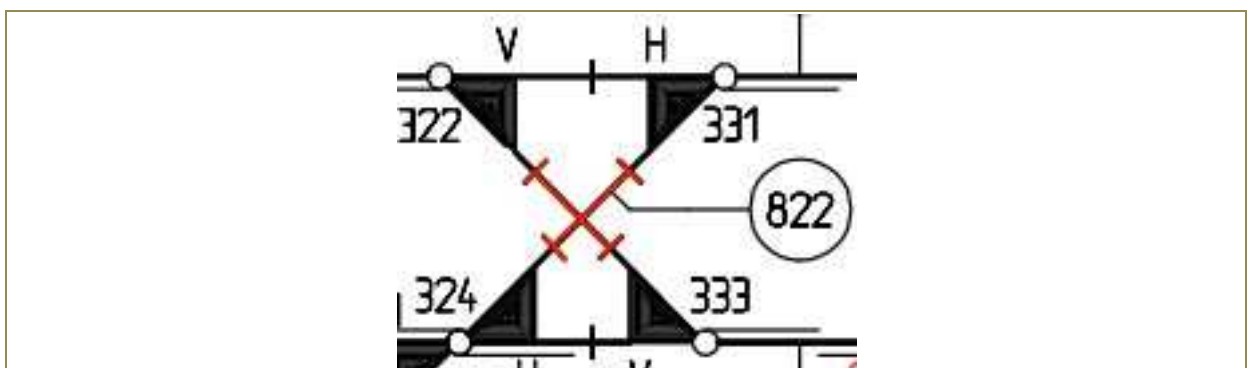
The example shows a derailer with similar features as a simple switch. The preferred position is the derailing one, which is automatically assumed after use. In addition the relation to the switch **“pt\_swi03”** is marked causing an interdependence of the positions. The position restriction itself is defined with the related switch.

```
<derailerIL returnsToPreferredPosition="true" id="dr_der01" isKeyLocked="false"
  maxThrowTime="PT10S" typicalThrowTime="PT6S" preferredPosition="derailingPosition">
  <designator register="_SimpleRegister" entry="69GS04"/>
  <refersTo ref="der01"/>
  <relatedMovableElement ref="pt_swi03" />
</derailerIL>
```

### 3.3.3 MovableCrossing

A crossing is described as element of the track network, where two tracks are crossing at the same height without having the possibility to change between the two tracks. Such elements are often called “diamond crossing” or “simple crossing” because of their shape. They shall not be mixed up with single or double slip switches, which are a combination of switches with a crossing within one element.

In majority of cases such crossings are just passive, i.e. there is nothing moving on its trackwork. Thus the “movable” in the element name seems to be incorrect. However, modern interlockings needs to assign a particular position to a crossing whether this is physically needed or not. Subsequently the crossing as represented in the interlocking can have a position. In case of switch actuators connected to the crossing for changing its position the attribute **@numberOfFrogSwitchActuators** shall be used to identify their number. Otherwise the value “0” shall be set or the attribute not used at all.



The picture shows the typical representation of a crossing in a schematic track plan. Therefore its configuration is always considered diagonal when describing its features.

Note: The above picture shows the often-used combination of four switches with a diamond crossing to interconnect two parallel tracks. This complete arrangement of four switches and the

crossing is called “scissor crossing” or “scissor crossover”.

Refer also <https://dccwiki.com/Crossing> [7].

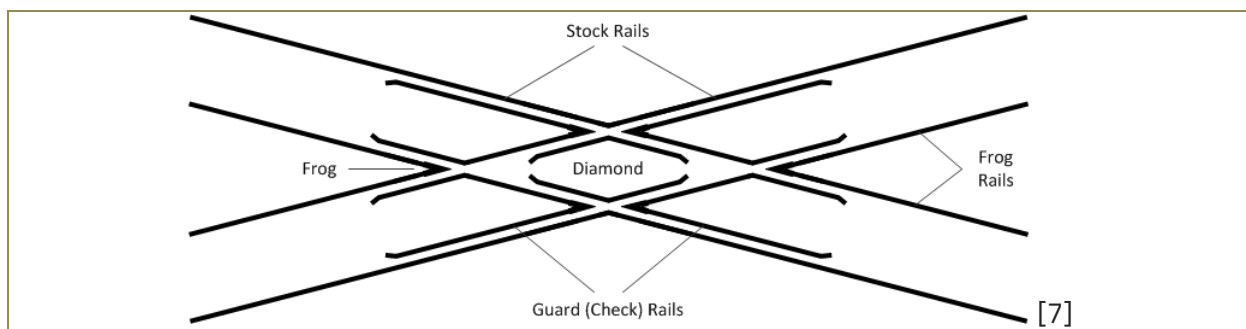
With the presumption of the crossing can have a position for the interlocking system, virtual or in reality, it is modelled as movable element. The instantiation extends the abstract type of **MovableElement** by some additional elements and attributes.

- **<refersTo>** – This is the reference to the track asset **<crossing>** from the infrastructure part.
- **<branchUpLeft>** – This the reference to the underlying track element in infrastructure forming its upper left branch.
- **<branchUpRight>** – This the reference to the underlying track element in infrastructure forming its upper right branch.
- **<branchDownLeft>** – This the reference to the underlying track element in infrastructure forming its lower left branch.
- **<branchDownRight>** – This the reference to the underlying track element in infrastructure forming its lower right branch.
- **<hasFoulingTrainDetectors>** – This is the reference to the train detectors delimiting the TVD section of this crossing, which are too close and cannot guarantee a clear gauge of the set track.
- **@preferredPosition** – This marks the switch position, which the crossing shall have when not in use of a route. The possible positions can be “downleft-rightup” and “upleft-rightdown” where the combination between up and down indicates the branches of the crossing a train would drive on.

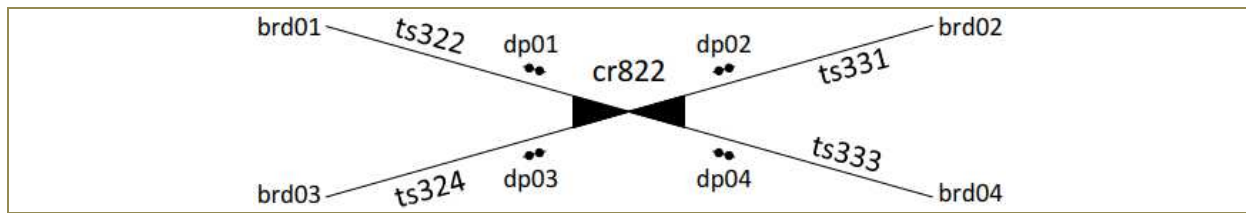
### 3.3.4 Special Examples

#### 3.3.4.1 Simple Crossing

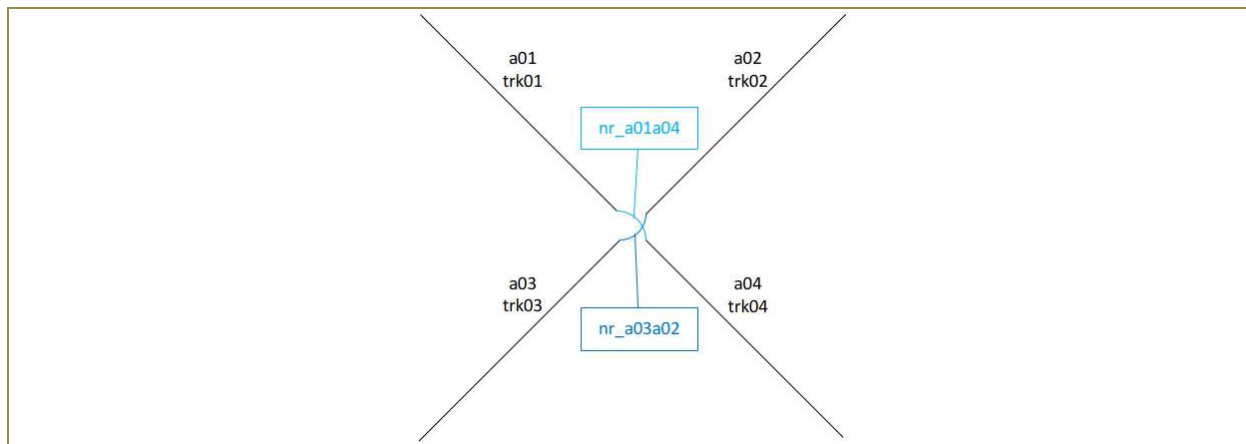
Because the simple example as shown in chapter 1.4 does not contain any special trackwork elements like crossings or slip switches there is the need for special examples. It shall be noted that the representation in infrastructure is given here just as base for the interlocking element example. However, it is not yet approved by infrastructure community and may be subject to alterations.



The physical appearance of the simple crossing with its components is shown above. Based on the extract from a real plan in chapter 3.3.3 the track plan just for a crossing with neighbouring tracks is shown in the figure below.



The related representation in infrastructure topology could be sketched like in the figure below. Only the `<netElements>` and `<netRelations>` are drawn.



Transferring this topology into railML we need first the four `<netElements>`.

```
<netElements>
  <netElement id="ne_a01">
    <name name="branch322-822" language="en"/>
    <associatedPositioningSystem id="pos01">
      <intrinsicCoordinate intrinsicCoord="0.0" id="a01.0" />
      <intrinsicCoordinate intrinsicCoord="1.0" id="a01.1" />
    </associatedPositioningSystem>
  </netElement>
  <netElement id="ne_a02">
    <name name="branch331-822" language="en"/>
    <associatedPositioningSystem id="pos02">
      <intrinsicCoordinate intrinsicCoord="0.0" id="a02.0" />
      <intrinsicCoordinate intrinsicCoord="1.0" id="a02.1" />
    </associatedPositioningSystem>
  </netElement>
  <netElement id="ne_a03">
    <name name="branch324-822" language="en"/>
    <associatedPositioningSystem id="pos03">
      <intrinsicCoordinate intrinsicCoord="0.0" id="a03.0" />
      <intrinsicCoordinate intrinsicCoord="1.0" id="a03.1" />
    </associatedPositioningSystem>
  </netElement>
  <netElement id="ne_a04">
    <name name="branch333-822" language="en"/>
    <associatedPositioningSystem id="pos04">
      <intrinsicCoordinate intrinsicCoord="0.0" id="a04.0" />
      <intrinsicCoordinate intrinsicCoord="1.0" id="a04.1" />
    </associatedPositioningSystem>
  </netElement>
</netElements>
```

Although the figure above shows only the two navigable relations in railML we shall list all possible combinations in the `<netRelations>`.

```

<netRelations>
  <netRelation id="nr_a01a02" navigability="None" positionOnA="1" positionOnB="0">
    <name name="rel322-822-331" language="en"/>
    <elementA ref="ne_a01" />
    <elementB ref="ne_a02" />
  </netRelation>
  <netRelation id="nr_a01a03" navigability="None" positionOnA="1" positionOnB="1">
    <name name="rel322-822-324" language="en"/>
    <elementA ref="ne_a01" />
    <elementB ref="ne_a03" />
  </netRelation>
  <netRelation id="nr_a01a04" navigability="Both" positionOnA="1" positionOnB="0">
    <name name="rel322-822-333" language="en"/>
    <elementA ref="ne_a01" />
    <elementB ref="ne_a04" />
  </netRelation>
  <netRelation id="nr_a02a04" navigability="None" positionOnA="0" positionOnB="0">
    <name name="rel331-822-333" language="en"/>
    <elementA ref="ne_a02" />
    <elementB ref="ne_a04" />
  </netRelation>
  <netRelation id="nr_a03a02" navigability="Both" positionOnA="1" positionOnB="0">
    <name name="rel324-822-331" language="en"/>
    <elementA ref="ne_a03" />
    <elementB ref="ne_a02" />
  </netRelation>
  <netRelation id="nr_a03a04" navigability="None" positionOnA="1" positionOnB="0">
    <name name="rel324-822-333" language="en"/>
    <elementA ref="ne_a03" />
    <elementB ref="ne_a04" />
  </netRelation>
</netRelations>

```

Finally the **<topology>** element requires the collection of these information within a **<network>**.

```

<networks>
  <network id="netw_01">
    <level id="netlv_01">
      <networkResource ref="ne_a01"/>
      <networkResource ref="ne_a02"/>
      <networkResource ref="ne_a03"/>
      <networkResource ref="ne_a04"/>
      <networkResource ref="nr_a01a02"/>
      <networkResource ref="nr_a01a03"/>
      <networkResource ref="nr_a01a04"/>
      <networkResource ref="nr_a02a04"/>
      <networkResource ref="nr_a03a02"/>
      <networkResource ref="nr_a03a04"/>
    </level>
  </network>
</networks>

```

In the infrastructure the various objects needs to be defined in **<functionalInfrastructure>**. Just for the sake of limiting the viewed area of the example the **<borders>** are needed.

```

<borders>
  <border type="area" id="brd01">
    <spotLocation id="sl_brd01" netElementRef="ne_a01" pos="0.0"
      applicationDirection="both"/>
  </border>
  <border type="area" id="brd02">
    <spotLocation id="sl_brd02" netElementRef="ne_a02" pos="1.0"
      applicationDirection="both"/>
  </border>

```

```

<border type="area" id="brd03">
  <spotLocation id="sl_brd03" netElementRef="ne_a03" pos="0.0"
    applicationDirection="both"/>
</border>
<border type="area" id="brd04">
  <spotLocation id="sl_brd04" netElementRef="ne_a04" pos="1.0"
    applicationDirection="both"/>
</border>
</borders>

```

As the interlocking needs also information to the neighbouring relations of track elements `<tracks>` are needed. In infrastructure a `<track>` can be a part of just one `<netElement>` or span over several `<netElements>`. For simplicity the four tracks are comprising the complete `<netElement>` from the `<border>` to the `<crossing>`. It would be also possible to use other limits like the `<trainDetectionElements>`.

```

<tracks>
  <track type="mainTrack" id="trk01">
    <linearLocation id="ltrk01" applicationDirection="both">
      <associatedNetElement keepsOrientation="true" netElementRef="ne_a01" />
    </linearLocation>
    <trackBegin ref="brd01" />
    <trackEnd ref="crx01" />
    <length type="operational" value="100" />
  </track>
  <track type="mainTrack" id="trk02">
    <linearLocation id="ltrk02" applicationDirection="both">
      <associatedNetElement keepsOrientation="true" netElementRef="ne_a02" />
    </linearLocation>
    <trackBegin ref="crx01" />
    <trackEnd ref="brd02" />
    <length type="operational" value="100" />
  </track>
  <track type="mainTrack" id="trk03">
    <linearLocation id="ltrk03" applicationDirection="both">
      <associatedNetElement keepsOrientation="true" netElementRef="ne_a03" />
    </linearLocation>
    <trackBegin ref="brd03" />
    <trackEnd ref="crx01" />
    <length type="operational" value="100" />
  </track>
  <track type="mainTrack" id="trk04">
    <linearLocation id="ltrk04" applicationDirection="both">
      <associatedNetElement keepsOrientation="true" netElementRef="ne_a04" />
    </linearLocation>
    <trackBegin ref="crx01" />
    <trackEnd ref="brd04" />
    <length type="operational" value="100" />
  </track>
</tracks>

```

For the definition of `<TvdSections>` in interlocking the `<trainDetectionElements>` needs to be listed. We assume axle counter detection points naming them according the related sections.

```

<trainDetectionElements>
  <trainDetectionElement id="ax_dp01" type="axleCounter">
    <name name="ax322_822" language="en"/>
    <spotLocation netElementRef="ne_a01" id="ax_dpp01" pos="0.5"
      applicationDirection="both"/>
  </trainDetectionElement>
  <trainDetectionElement id="ax_dp02" type="axleCounter">
    <name name="ax822_331" language="en"/>
  </trainDetectionElement>
</trainDetectionElements>

```

```

    <spotLocation netElementRef="ne_a02" id="ax_dpp02" pos="0.5"
      applicationDirection="both"/>
  </trainDetectionElement>
  <trainDetectionElement id="ax_dp03" type="axleCounter">
    <name name="ax324_822" language="en"/>
    <spotLocation netElementRef="ne_a03" id="ax_dpp03" pos="0.5"
      applicationDirection="both"/>
  </trainDetectionElement>
  <trainDetectionElement id="ax_dp04" type="axleCounter">
    <name name="ax822_333" language="en"/>
    <spotLocation netElementRef="ne_a04" id="ax_dpp04" pos="0.5"
      applicationDirection="both"/>
  </trainDetectionElement>
</trainDetectionElements>

```

The `<crossing>` itself looks rather simple as infrastructure element. It is mainly defined by the reference to the `<netRelations>` of the straight branches.

```

<crossing id="crx01">
  <name name="cr822" language="en"/>
  <external id="cr822_1" ref="nr_a01a04" />
  <external id="cr822_2" ref="nr_a03a02" />
</crossing>

```

In the interlocking part the `<assetsForIL>` shall contain the interlocking known elements. For the example only the `<TvdSection>` of the crossing and the crossing itself is required. Beside the limiting train detectors, the relation to the infrastructure `<tracks>` is needed.

```

<tvdSection residualRouteCancellationDelay="PT90S" partialRouteReleaseDelay="PT60S"
  isBerthingTrack="false" id="tvd_01">
  <designator register="_SimpleRegister" entry="ts_cr822"/>
  <hasDemarcatingTraindetector ref="ax_dp01" />
  <hasDemarcatingTraindetector ref="ax_dp02" />
  <hasDemarcatingTraindetector ref="ax_dp03" />
  <hasDemarcatingTraindetector ref="ax_dp04" />
</tvdSection>

```

The crossing is described as movable element with references to the `<TvdSection>` and the references to the adjacent `<tracks>`.

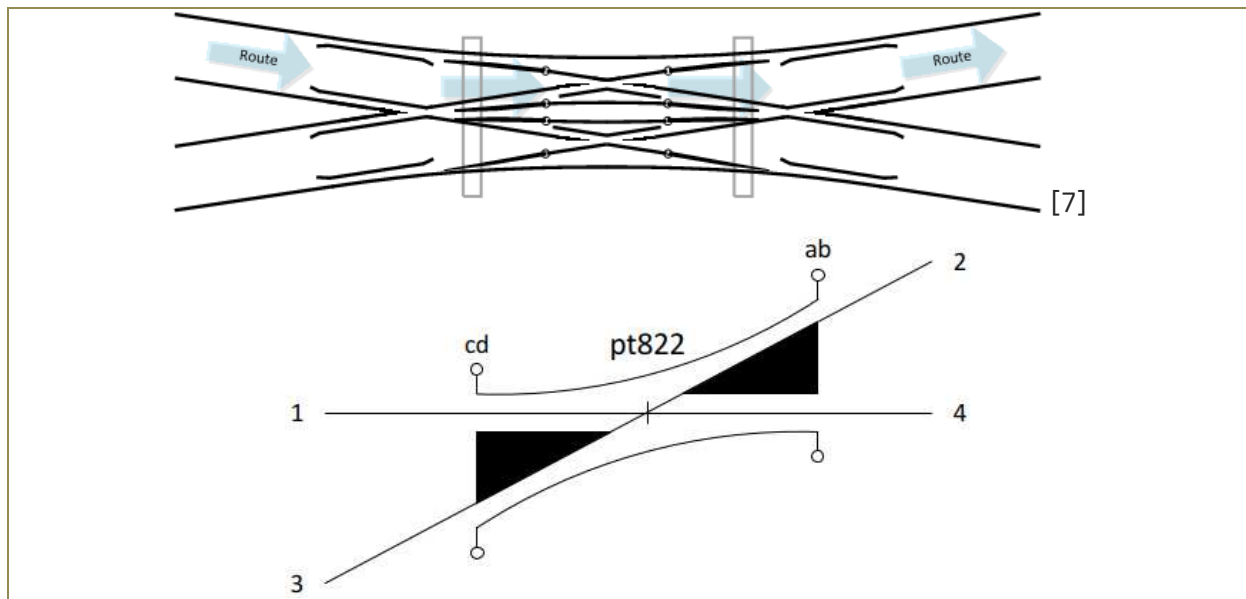
```

<movableCrossing maxThrowTime="PT10S" id="mov_01" numberOfFrogSwitchActuators="0">
  <designator register="_SimpleRegister" entry="cr822"/>
  <refersTo ref="crx01" />
  <hasTvdSection ref="tvd_01" />
  <branchUpLeft ref="trk01" />
  <branchUpRight ref="trk02" />
  <branchDownLeft ref="trk03" />
  <branchDownRight ref="trk04" />
</movableCrossing>

```

### 3.3.4.2 Double Slip Switch

Naturally the next evolution step from a simple crossing to a slip switch would be the single slip switch. However, this is a more complicated element for an interlocking than a double slip switch. Therefore the above example is extended to a double slip switch. For common understanding the physical appearance and the track layout of such a double slip switch is shown here. For an interlocking it does not matter whether the blades are located inside or outside of the diamond.



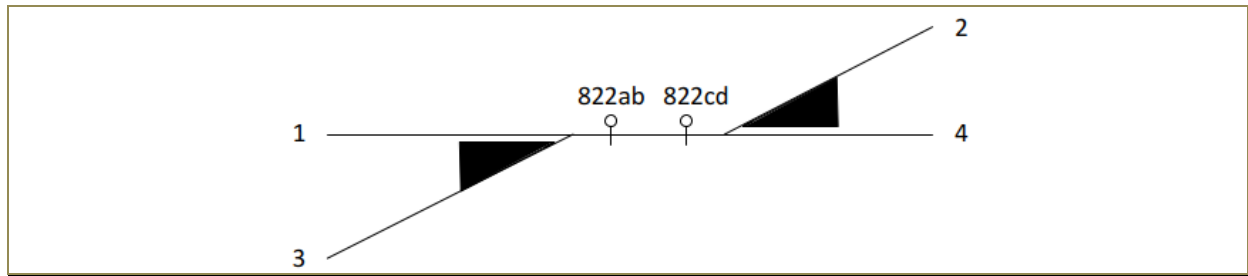
Considering the XML data of the simple crossing in chapter 3.3.4.1 just the necessary changes are highlighted here. It starts with the `<netRelations>` as the double slip switch has four possible paths to navigate. Subsequently the relations “nr\_a01a02” and “nr\_a03a04” have to be revised, as they are now navigable in both directions.

```
<netRelation id="nr_a01a02" navigability="Both" positionOnA="1" positionOnB="0">
  <name name="rel322-822-331" language="en"/>
  <elementA ref="ne_a01" />
  <elementB ref="ne_a02" />
</netRelation>
<netRelation id="nr_a03a04" navigability="Both" positionOnA="1" positionOnB="0">
  <name name="rel324-822-333" language="en"/>
  <elementA ref="ne_a03" />
  <elementB ref="ne_a04" />
</netRelation>
```

The `<crossing>` element in infrastructure is changed to `<switchIS>` and extended by the switch type and the references to the turning branches. The name is updated to reflect a switch instead of a crossing.

```
<switchIS id="dcrx01" type="doubleSwitchCrossing">
  <name name="pt822" language="en"/>
  <straightBranch netRelationRef="nr_a01a04" />
  <straightBranch netRelationRef="nr_a03a02" />
  <turningBranch netRelationRef="nr_a01a02" />
  <turningBranch netRelationRef="nr_a03a04" />
</switchIS>
```

The biggest changes are necessary in the interlocking part as a slip switch is typically translated for the interlocking into two single switches arranged tip to tip as shown in the figure above and the alternative diagram below. Although this allows direct relation to the related switch actuators, the virtual replacement of both switches has to be noted. Thus the right side (“ab” in the example) is placed on the left side for the interlocking as the position of the ab-side blades will control the path towards “trk01” (upper left) and “trk03” (lower left). The functional model (alternative diagram) as used within an interlocking is shown below.



The model results in the following combinations:

from track	to track	switch positions
1	2	822ab-right / 822cd-left
1	4	822ab-right / 822cd-right
3	2	822ab-left / 822cd-left
3	4	822ab-left / 822cd-right

With the splitting of one track element into two interlocking elements the relation to the other half is required. This is realised with `<relatedMovableElement>` and `@switchPointType`.

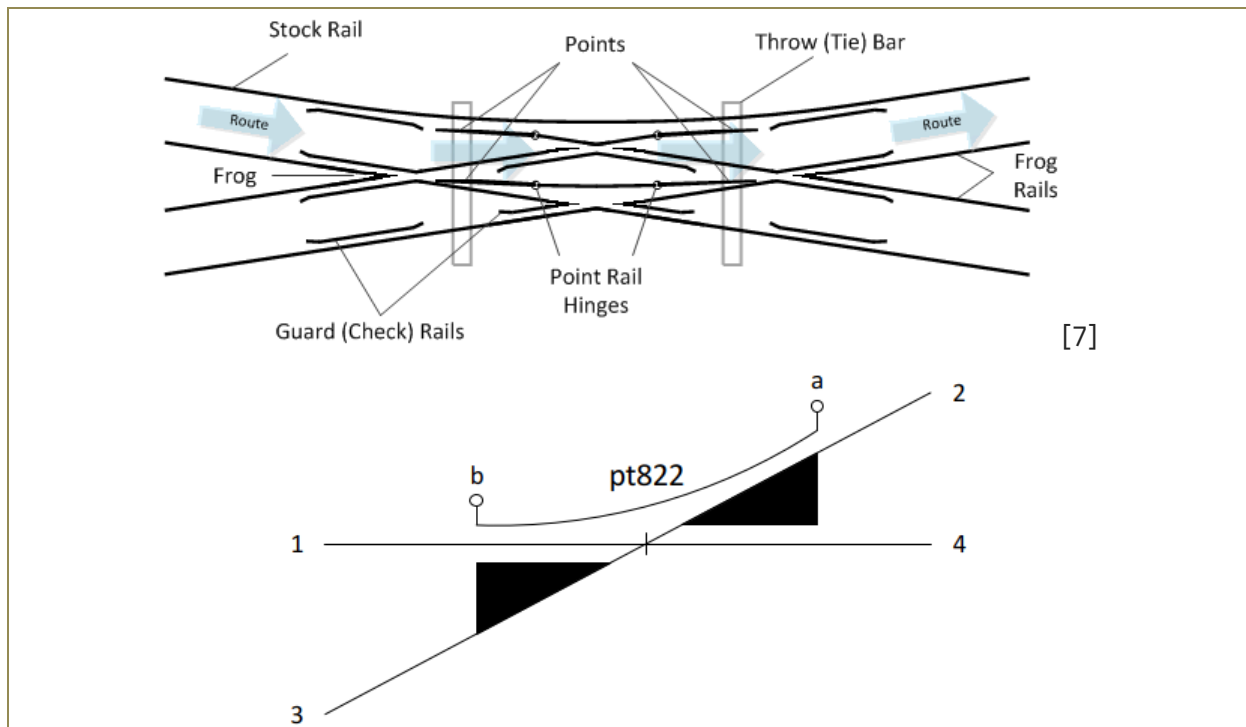
```

<switchesIL>
<!--
pt822ab is the point machine physically on the right hand side but virtually left sided
-->
  <switchIL maxThrowTime="PT10S" id="mov_04" numberOfFrogSwitchActuators="0"
    numberOfBladeSwitchActuators="1">
    <designator register="_SimpleRegister" entry="pt822ab"/>
    <refersTo ref="dcrx01" />
    <hasTvdSection ref="tvd_01" />
    <relatedMovableElement ref="mov_05" />
    <branchLeft ref="trk03" />
    <branchRight ref="trk01" />
  </switchIL>
<!--
pt822cd is the point machine physically on the left hand side but virtually right sided
-->
  <switchIL maxThrowTime="PT10S" id="mov_05" numberOfFrogSwitchActuators="0"
    numberOfBladeSwitchActuators="1">
    <designator register="_SimpleRegister" entry="pt822cd"/>
    <refersTo ref="dcrx01" />
    <hasTvdSection ref="tvd_01" />
    <relatedMovableElement ref="mov_04" />
    <branchLeft ref="trk02" />
    <branchRight ref="trk04" />
  </switchIL>
</switchesIL>

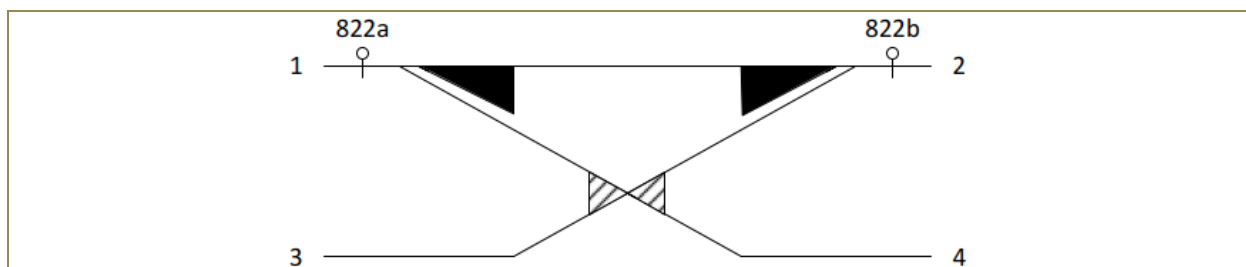
```

### 3.3.4.3 Single Slip Switch

In terms of the `<topology>` and the `<functionalInfrastructure>` the single slip switch is just halfway from simple crossing to double slip switch as there is only one turning branch. For common understanding the physical appearance and the track layout of such a single slip switch is shown here.



The functional model (alternative diagram) as used within an interlocking is shown below.



The model results in the following combinations:

from track	to track	switch positions
1	2	822a-left / 822b-right
1	4	822a-right (822b-right)
3	2	822b-left (822a-left)

Considering the XML data of the double slip switch in chapter 3.3.4.2 just the necessary changes are highlighted here. It starts with the `<netRelations>` as the single slip switch has only three possible paths to navigate. Subsequently the relation "nr\_a03a04" has to be revised, as it is not navigable in this example.

```
<netRelation id="nr_a03a04" navigability="None" positionOnA="1" positionOnB="0">
  <name name="rel324-822-333" language="en"/>
  <elementA ref="ne_a03" />
  <elementB ref="ne_a04" />
</netRelation>
```

The `<switchIS>` element in infrastructure is revised by the removal of the second turning branch and the switch type.

```
<switchIS id="scrx01" type="singleSwitchCrossing">
```

```

<name name="pt822" language="en"/>
<straightBranch netRelationRef="nr_a01a04" />
<straightBranch netRelationRef="nr_a03a02" />
<turningBranch netRelationRef="nr_a01a02" />
</switchIS>

```

For an interlocking the single slip switch is a special case of the double slip switch because the model of two single switches require a position restriction. As visible from the combination table the situation of a-side in right position and the b-side in left position shall never occur. If the turning branch is on the other side of the single slip switch the condition has to be inverted.

```

<switchesIL>
<!--
pt822a is the point machine physically on the right hand side but virtually left sided
-->
  <switchIL maxThrowTime="PT10S" id="mov_02" numberOfFrogSwitchActuators="0"
    numberOfBladeSwitchActuators="1" >
    <designator register="_SimpleRegister" entry="pt822a"/>
    <refersTo ref="scrx01" />
    <hasTvdSection ref="tvd_01" />
    <relatedMovableElement ref="mov_03" />
    <branchLeft ref="trk03" />
    <branchRight ref="trk01" />
    <hasPositionRestriction restrictedPosition="right">
      <relatedSwitchInPosition id="res822a" inPosition="right">
        <refersToSwitch ref="mov_03" />
      </relatedSwitchInPosition>
    </hasPositionRestriction>
  </switchIL>
<!--
pt822b is the point machine physically on the left hand side but virtually right sided
-->
  <switchIL maxThrowTime="PT10S" id="mov_03" numberOfFrogSwitchActuators="0"
    numberOfBladeSwitchActuators="1" >
    <designator register="_SimpleRegister" entry="pt822b"/>
    <refersTo ref="scrx01" />
    <hasTvdSection ref="tvd_01" />
    <relatedMovableElement ref="mov_02" />
    <branchLeft ref="trk02" />
    <branchRight ref="trk04" />
    <hasPositionRestriction restrictedPosition="left">
      <relatedSwitchInPosition id="res822b" inPosition="left">
        <refersToSwitch ref="mov_02" />
      </relatedSwitchInPosition>
    </hasPositionRestriction>
  </switchIL>
</switchesIL>

```

### 3.3.4.4 Coupled Switches

Coupled switches are mainly configured at crossovers between parallel tracks. They are always operated at once. Thus both of them are synchronised in branching or straight position. The interlocking just need to know the relation itself. The synchronised combination of both switch positions can be derived from the topology.

In the example of coupled switches the relation between both switches is marked by using the element `<relatedMovableElement>`. In addition it shall be ensured that any other attributes of these switches are not contradictory, e.g. `@returnsToPreferredPosition`.

```

<switchIL returnsToPreferredPosition="true" id="pt_swi02" isKeyLocked="false"
  maxThrowTime="PT10S" typicalThrowTime="PT6S" preferredPosition="left">
  <designator register="_SimpleRegister" entry="69W03"/>
  <refersTo ref="swi02" />
  <relatedMovableElement ref="pt_swi03" />
  <branchLeft ref="trc04" />
  <branchRight ref="trc07" />
</switchIL>
<switchIL returnsToPreferredPosition="true" id="pt_swi03" isKeyLocked="false"
  maxThrowTime="PT10S" typicalThrowTime="PT6S" preferredPosition="left">
  <designator register="_SimpleRegister" entry="69W04"/>
  <refersTo ref="swi03" />
  <relatedMovableElement ref="pt_swi02" />
  <branchLeft ref="trc06" />
  <branchRight ref="trc07" />
</switchIL>

```

### 3.3.4.5 Dependency with Derailer

Some operators might configure a dependency between a derailer and the switch leading towards the derailer. The normal sequence will be that the derailer has to be switched to passable position before the related switch can be switched in the position towards the derailer. After use the switch has to be switched first into the protecting position before the derailer can be switched into the derailing position. This shall reduce the risk of unintended derailing during shunting movements.

The example shows the relation between switch “swi03” and derailer “der01” by using the element `<relatedMovableElement>`.

```

<switchesIL>
  <switchIL returnsToPreferredPosition="true" id="pt_swi03" isKeyLocked="false"
    maxThrowTime="PT10S" typicalThrowTime="PT6S" preferredPosition="left">
    <designator register="_SimpleRegister" entry="69W04"/>
    <refersTo ref="swi03" />
    <relatedMovableElement ref="dr_der01" />
    <branchLeft ref="trc06" />
    <branchRight ref="trc07" />
    <hasPositionRestriction restrictedPosition="left">
      <relatedDerailerInPosition inPosition="passablePosition">
        <refersToDerailer ref="dr_der01" />
      </relatedDerailerInPosition>
    </hasPositionRestriction>
  </switchIL>
</switchesIL>
<derailersIL>
  <derailerIL returnsToPreferredPosition="true" id="dr_der01" isKeyLocked="false"
    maxThrowTime="PT10S" typicalThrowTime="PT6S" preferredPosition="derailingPosition">
    <designator register="_SimpleRegister" entry="69GS04"/>
    <refersTo ref="der01"/>
    <relatedMovableElement ref="pt_swi03" />
    <hasPositionRestriction restrictedPosition="derailingPosition">
      <relatedSwitchInPosition inPosition="right">
        <refersToSwitch ref="pt_swi03" />
      </relatedSwitchInPosition>
    </hasPositionRestriction>
  </derailerIL>
</derailersIL>

```

### 3.4 LevelCrossingIL

At places where rail and road traffic cross at the same level the safety often has to be established by technical means. The related equipment is modelled in `<levelCrossingIL>`. In order to avoid ambiguity with the infrastructure element the related type is called `LevelCrossingIL` in railML interlocking subschema.

The features of the particular level crossing can be detailed as below even not all information may be really “known” to the interlocking, i.e. in case of autonomous level crossing.

- `@constantWarningTime` – This is the time the level crossing is activated in advance of the approaching train based on the activation position and the train speed.
- `@maximumClosedTime` – This is the time a level crossing is accepted by the interlocking as constantly closed before an alarm is raised. The value is needed by some IM because of human behaviour. A level crossing shall be considered insecure after elapsing this time as road traffic might start to negotiate the closed level crossing after an eventless waiting time.
- `@minimumOpenTime` – This is the time the level crossing shall be at least open until it is allowed to close again in order to ensure a certain amount of road traffic flow across the level crossing.
- `@preferredPosition` – In most cases the preferred position of a level crossing is “open”, i.e. allowing road traffic to pass freely. However, there might be cases when the level crossing is normally “closed” and opened for road traffic only on command.
- `@requiresStopBeforeUnprotectedLevelCrossing` – Flag to define whether any train needs to stop in front of the level crossing in case it is unprotected. Only afterwards it can proceed according the value in `@unprotectedSpeed`.
- `@unprotectedSpeed` – This is the speed limit in km/h a train is allowed to pass the level crossing when out of order, i.e. not activated for the approaching train.
- `@typicalTimeToClose` – This is the time the interlocking shall expect required between commanding the level crossing and the notification of the other position. After elapse of the timer an alarm might be raised to indicate the problem.  
In general it is assumed that closing and opening do take the same amount of time.
- `<isLevelCrossingType>` – This is the reference to the generic type of the level crossing defining at least the way of control from the interlocking.
- `<refersTo>` – This the reference to the physical level crossing `<levelCrossingIS>` in the infrastructure.
- `<interface>` – This is the description of the physical interface between the interlocking and the level crossing. It mainly contains the list of commands and messages exchanged on this interface. For details refer chapter 6 below.
- `<deactivatedBy>` – This describes the deactivation of the level crossing. For details refer chapter 3.4.2 below.
- `<activationCondition>` – This describes the conditions for activating the level crossing. For details refer chapter 3.4.1 below.
- `<hasTvdSection>` – This is the mandatory reference list to all TVD sections located directly at the level crossing.

### 3.4.1 Level crossing activation

A level crossing is activated, i.e. requested to close for road traffic, upon train approach. There is a range of conditions to conclude a train is approaching the level crossing. In first case this happens when the train crosses a detection point, i.e. an insulated track joint, axle counter or treadle. These approach detectors are commonly referred to as Approach Starting (AS). Otherwise the activation might be also dependent of a particular route set and the related signal aspects. There may be even combinations of it.

- **@andOr** – This element defines how the conditions for activation shall be logically combined. However, combinations that are more complex are not possible.
  - **“AND”** – The activation conditions will be combined all together, i.e. the level crossing will be activated when all of the named conditions apply.
  - **“OR”** – The activation conditions will be combined alternatively, i.e. the level crossing will be activated when at least one of the named conditions apply.
  - **“XOR”** – The activation conditions will be combined alternating exclusively, i.e. the level crossing will be activated when only one of the named conditions applies.
- **<delayedBySwitchPosition>** – The activation may be delayed by the position of a switch.
  - **@delay** – The time between the switch detected in the required position and the possible activation of the level crossing.
  - **@inPosition** – This is the position the switch shall have for activation.
  - **<refersToSwitch>** – This is the reference to the related moveable element.
- **<aspectRelatedDelay>** – The activation of the level crossing may be delayed by a given duration, if a signal shows a given aspect.
  - **@delay** – The time between the signal is detected in required aspect and the possible activation of the level crossing. This delay depends on the signalled speed of the approaching train hence on signal aspect.
  - **<refersToSignal>** – This is the reference to the related signal in the interlocking part.
  - **<showsAspect>** – This is the related aspect the signal shall show for activation.
- **<signalDelayTime>** – The delay time that maintains a signal at stop for a given duration after the level crossing was triggered.
  - **@delay** – The time that has to elapse for the signal to change from stop to any proceed aspect after the level crossing was activated.
  - **<hasDelayedSignal>** – This is the reference to the related signal in the interlocking part.
- **<activatedBy>** – The activation of the level crossing may be delayed depending on occupation status from the related train detection device. This activation condition is needed in any case independent of the logical combination of the other conditions.
  - **@delay** – The time between the device has detected an occupation and the possible activation of the level crossing.

- **<refersTo>** – This is the reference to the related train detection device. This device can be an axle counter, track section, track joint or treadle.

### 3.4.2 Level crossing deactivation

The level crossing is automatically deactivated after the train has passed and no activation trigger is present. The passage is detected by TVD section or particular train detection devices. There are usually two deactivation items per level crossing one per each direction.

- **@delay** – The time between the train detection has notified the clearing after train passage and the possible deactivation of the level crossing.
- **<tvdDetectorRef>** – This is the reference to the related train detection device. This device can be an axle counter, track section, track joint or treadle.

### 3.4.3 Complete Level Crossing

An example of completely described level crossing is shown in the extract below:

```
<levelCrossingIL constantWarningTime="PT15S" id="lx_lcr01" maximumClosedTime="PT20S"
  minimumOpenTime="PT10S" unprotectedSpeed="20.0" typicalTimeToClose="PT10S"
  preferredPosition="open">
  <designator register="_SimpleRegister" entry="levelCrossing01"/>
  <isLevelCrossingType ref="lxt01"/>
  <refersTo ref="lcr01"/>
  <deactivatedBy delay="PT5S" id="lcr01_off1">
    <designator register="_SimpleRegister" entry="lx01_off1"/>
    <tvdDetectorRef ref="tde11"/>
  </deactivatedBy>
  <deactivatedBy delay="PT5S" id="lcr01_off2">
    <designator register="_SimpleRegister" entry="lx01_off2"/>
    <tvdDetectorRef ref="tde12"/>
  </deactivatedBy>
  <activationCondition andOr="XOR">
    <delayBySwitchPosition delay="PT2S" id="swi01_d1" inPosition="left">
      <designator register="_SimpleRegister" entry="pt_delay"/>
      <refersToSwitch ref="swi01"/>
    </delayBySwitchPosition>
    <aspectRelatedDelay delay="PT15S" id="sig02_21">
      <designator register="_SimpleRegister" entry="sigaspect_delay"/>
      <refersToSignal ref="mb_sig02"/>
      <showsAspect ref="sig_reducproceed_21"/>
    </aspectRelatedDelay>
    <signalDelayTime delay="P1D" id="sig02_d1">
      <designator register="_SimpleRegister" entry="si_delay"/>
      <hasDelayedSignal ref="mb_sig02"/>
    </signalDelayTime>
    <activatedBy delay="P1D" id="td02_act">
      <designator register="_SimpleRegister" entry="td_act"/>
      <refersTo ref="tde02"/>
    </activatedBy>
  </activationCondition>
  <hasTvdSection ref="LX2.5T"/>
</levelCrossingIL>
```

### 3.5 Signall

Signals are used to transmit information from trackside to the train and its driver. They are protecting the train movements. Depending on the signal type they give allowance for movements or prohibit them and indicate acceptable speed for the movement for tracks in advance of it. The physical characteristics are included in the infrastructure part. Thus, the more functional features are part of the interlocking subschema. In order to avoid ambiguity with the infrastructure element the related type is called **SignalIL** in railML interlocking subschema.

- **@isVirtual** – There are two ways of having a virtual signal. First a virtual signal can be a dummy-signal that is a software object in the interlocking but has no physical trackside presence. Such virtual signals can be useful for modelling speed steps; there need not be a physical signal but the interlocking enforces a different speed at the position of the virtual signal. Such signals are set as virtual in infrastructure **SignalIS @isVirtual**, as well. The other way round, stand-alone boards (e.g. marker boards of ETCS) that are not wired to the interlocking shall be labelled virtual in interlocking subschema, if the interlocking is using them with different aspects. They are virtually switched by the interlocking. Although they have a physical representation on track side, the transmitted information to the driver is not included in this representation. However, they also affect the train movements.
- **@releaseSpeed** – Release speed in km/h from controlled braking curve. A release speed is a speed limit under which the train is allowed to run towards this signal, when the target speed is zero. This release speed is independent of any associated overlap or danger point, i.e. the minimum speed in case of different possibilities.
- **@malfunctionSpeed** – This constant indicates the maximum speed in km/h with which a train may travel past a failed signal. The malfunctioning signal cannot be opened with normal proceed aspect.
- **@approachSpeed** – The maximum speed in km/h with which a train can approach the signal independent of the aspect it is shown. This matches the Ka of the previous (=upstream) signal or speed sign. This is suitable for defining the line speed profile.
- **@passingSpeed** – Maximum speed in km/h beyond the signal independent of the aspect it is shown. This is suitable for defining the line speed profile.
- **@releaseDelayTimer** – This is the time to elapse between receiving the revocation command and before route release.
- **@function** – This is the function of the signal in the interlocking context. Just specifying the main purpose like main, distant or shunting signal only is not right information for the interlocking. A more detailed differentiation including the main purpose is necessary to define the handling of the particular signal in general, i.e. not only considering the use within routes. There are the following functions possible:
  - **“barrage”** – The barrage signal is a special protection signal. In most cases it is not really a main signal. Examples are the extra protection of level crossings or the case of the destination (berthing) track in a station, when it is divided in two parts, to protect the first train from the following one entering this track.
  - **“block”** – The block signal is used on open line at the start of a block route.

- **“entry”** – The entry signal is the main signal protecting the entrance of a station from the open line.
- **“exit”** – The exit signal is the start of a route from within a station onto the open line.
- **“distant”** – The distant signal is announcing the actual aspect of the related main signal. It is positioned in the normal braking distance in rear of the related main signal.
- **“intermediate”** – The intermediate signal is a main signal within a station neither used for entry nor exit routes.
- **“intermediateStop”** – This is a special intermediate signal that is used for marking the stopping place. It is used with long berthing tracks in station where the platform is not close to the exit signal.
- **“junction”** – The junction signal is used within several routes that start not at a main signal as the common main signal. This signal is typically used for exits from a yard where the individual tracks have shunting signals only.
- **“main”** – The main signal is a normal signal for train traffic protection which is neither used as block, entry, exit nor intermediate signal. This is the more general function in case no specific one can be used.
- **“repeater”** – The repeater signal just repeats the aspect of the related signal. Dependent on the IM the related signal can be of different type like main, distant or shunting signal.
- **“shunting”** – The shunting signal is mainly used in routes for shunting purpose or as indicator for local operation modus. Sometimes the shunting signal is included as intermediate signal in normal train routes.
- **“other:... ”** – The signal function is none of the already defined ones. This is the optional extension of the list. Each entry needs to start with the string “other:” and shall have at least two letters in addition.
- **@callOnAspectTime** – This is the time value for setting the duration of an active call-on aspect at this signal.
- **@sightDistance** – This is the distance in metres the signal is visible in advance by the train driver. The value may affect the reaction times on changing aspects in operation simulation.
- **<refersTo>** – This is the reference to the physical signal **<signalIS>** in the infrastructure.
- **<protectsBlockExit>** – This is the reference to train detection device at the border between station and open line, when the signal is protecting the exit from open line, i.e. the block exit or station entry.

It has to be noted that the data given in **<signalIL>** do only reflect characteristics of the signal in general but are not bound to a particular route. The dynamic signal information related to a particular route is defined in **SignalPlan**.

The first extract shows a marker board which is used as virtual exit signal from the station.

```
<signalIL id="mb_sig01" isVirtual="true" releaseSpeed="0" malfunctionSpeed="0"
  approachSpeed="0" passingSpeed="0" releaseDelay="PT5S" function="exit">
  <designator register="_SimpleRegister" entry="Arnau 68N2"/>
  <refersTo ref="sig01" />
</signalIL>
```

This extract shows a conventional light signal, which is not virtual and located at the station entry.

```
<signalIL id="ls_sig04" isVirtual="false" releaseSpeed="0" malfunctionSpeed="0"
  approachSpeed="0" passingSpeed="0" releaseDelay="PT5S" function="entry"
  callOnAspectTime="PT90S">
  <designator register="_SimpleRegister" entry="Cstadt 69A"/>
  <refersTo ref="sig04" />
</signalIL>
```

## 3.6 Logical Devices

There can be various logical devices, which provide their state as input to the interlocking. Depending on their use it is also possible to send particular information from the interlocking to the logical device in order to provoke a specific reaction. It is an abstract class extended by the particular instances of it like `KeyLockIL` or `GenericDetector`. The common attributes and elements are:

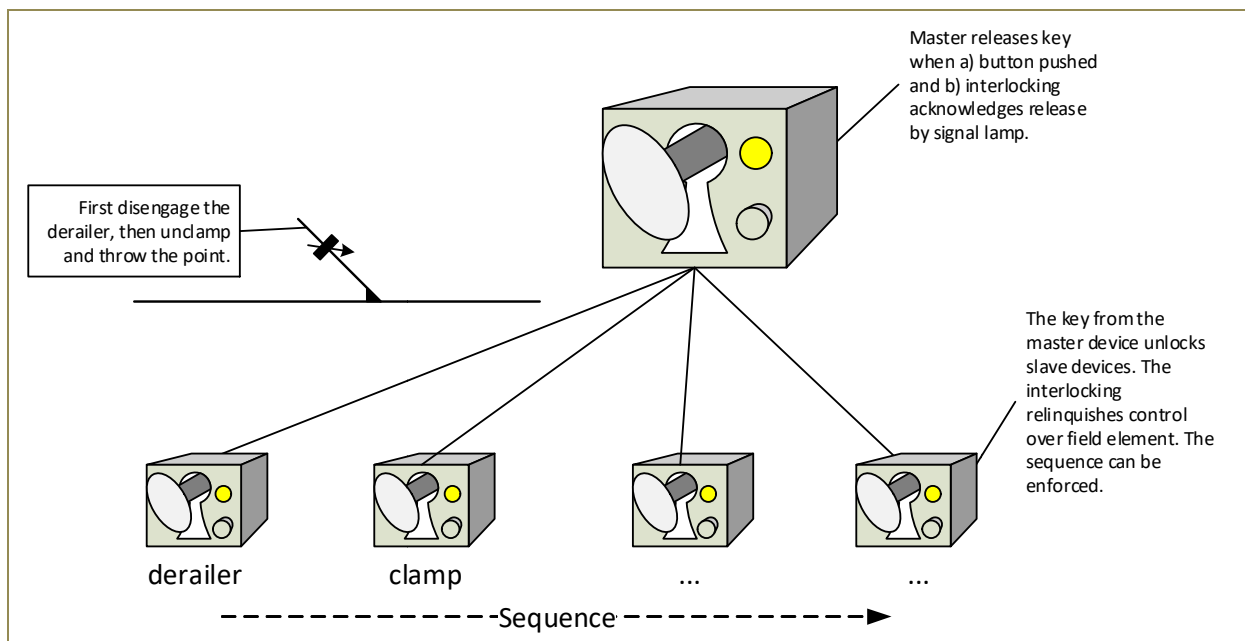
- `@description` – The string element takes a more detailed description of the logic. This is for information of the user only.
- `<takesControlOf>` – This is the reference to a movable element which can be controlled with the lock. In case of a detector, it is used as reference to any track asset, which is related to the logical device.
- `<hasInterface>` – The reference to the physical interface of the logical device to the interlocking.
- `<refersTo>` – The reference to the physical device like `<keyLockIS>` in the infrastructure. This element is optional as some detectors may not have a corresponding element in the infrastructure because they are not located at the track.

The particular instantiation extends the abstract type of `logicalDevice` by some additional elements and attributes.

There are further logical devices conceivable that **are yet not implemented**: tunnel gates, bascule bridges or water barriers.

### 3.6.1 KeyLockIL

A key lock (*de: Schlüsselschalter*) is a mechanical device often electrically controlled from the interlocking. It is situated near the track and can have two basic positions. One of them is the key securely locked in and the other with the key released for removal. The key is a simple mean of authorisation transfer as its use allows the manual operation of a moveable element. The lock with control from the interlocking is called master lock, whereas the other locks being operable by the released key are the slave locks.



A key lock is used to request local control of a (group of) track assets from the interlocking without the need of expensive machines and control stuff. Commonly, railway staff requests local control from the interlocking via this device. Once granted, the key can be removed upon which the (group of) track asset is no longer under interlocking control. In reverse, the interlocking takes back control when the key is inserted and the staff acknowledged relinquishing control.

Note that the lock is a track asset defined in infrastructure namespace. The interlocking reads the state of the lock and returns permission to remove the key, i.e. `levelOfControl=fullControl`.

A combined lock has a master lock that controls a set of slave locks that can be operated with the key from the master lock but are not connected to the interlocking. Slave locks may have to be released in a well-defined sequence.

The master lock is a key release element that is operated directly from the interlocking. The key is electrically locked and released on operator command or trigger event from the interlocking. These locks are typically used for movable elements in route path, which are not operated directly from the interlocking like siding switches on open line or tunnel gates or bascule bridges.

The key lock is defined by the following attributes and elements:

- **@hasAutomaticKeyRelease** – The flag specifies whether the key shall be released (authorised) automatically when a triggering event occurs. This is used for key locks on open line for automatically release the key when the related TVD section is occupied, i.e. when the train arrives at the entry to the siding.
- **@hasAutomaticKeyLock** – The flag specifies whether the key shall be locked automatically when it is returned into the lock, i.e. only one time use is permitted.
- **@keyRequestTime** – The timer value for automatic revocation of the key request, i.e. the indication to the controller for key release request is automatically removed if not answered (key release commanded) in-between.
- **@keyAuthoriseTime** – The timer value for automatic revocation of the key release, i.e. the key is automatically relocked if not taken out in-between.

- `<acceptsKey>` – This is the reference to a key.
- `<hasTvdSection>` – This is the reference to a TVD section the lock might be related to. This is especially relevant for sidings with key lock on the open line where the key is released on occupation of the related TVD section in combination with a special route type.
- `<hasSlaveLock>` – This is the reference to any dependent lock that can be used once this master lock is released.

### 3.6.2 Key

Although the key itself is not a logical device it is handled here due to its relation to key locks. The element mainly needs just the `<designator>` and `@id` to describe a key that can be referred by any lock as suitable to operate it. In a conventional system this means a hardware key that is related to physical lock.

- `@isPhysical` – The Boolean value gives indication whether the key is of physical type for use in a mechanical lock.

### 3.6.3 GenericDetector

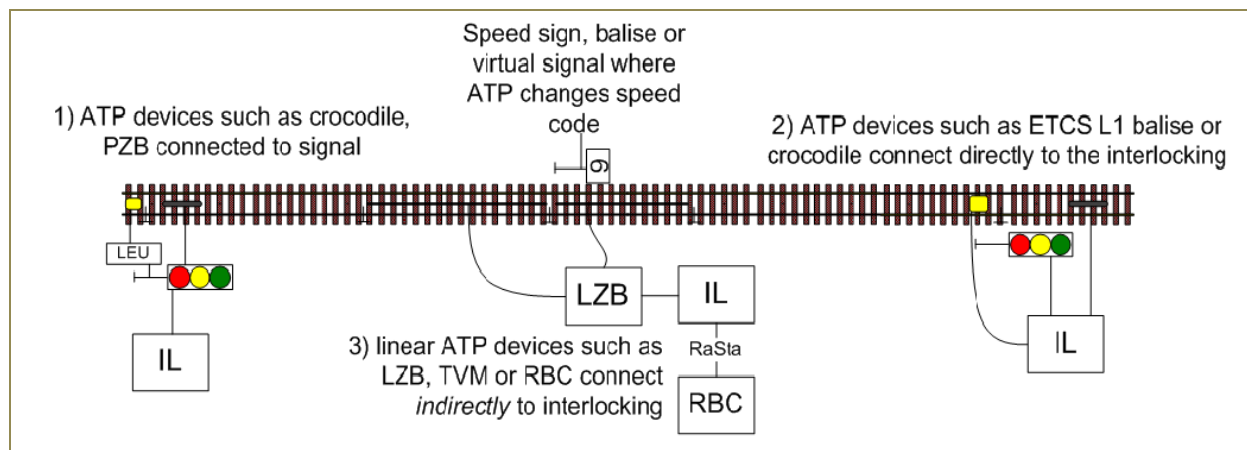
Detectors are devices that check for exceeding a particular characteristic and providing an output to the interlocking. Depending on the function it may influence the route signalling. The base types of such detectors are defined in the IM dependent section 2.7.

Independent of the particular detector type it can be described by the following attributes:

- `@affectsRouteSignalling` – The Boolean value gives indication whether this detector is safety critical and its output influences the train routes.
- `@allowsSingleOverride` – The Boolean value defines whether the particular command for single override shall be provided by the interlocking for this detector.
- `@allowsPermanentOverride` – The Boolean value defines whether the particular commands for enabling and disabling of override shall be provided by the interlocking for this detector.
- `@hasTriggeredSelfTest` – The Boolean value defines whether the interlocking needs to trigger a self-test of this detector in order to check its correct working.
- `@selfTestToleranceTime` – The duration gives the time the self-test needs to run and any outputs from the detector during it shall be tolerated.
- `@selfTestInterval` – The duration value gives the time interval at which the interlocking needs to trigger the self-test of the detector. It has to be used in combination with `@triggeredSelfTest`.
- `<detectorType>` – This is the reference to the particular type of the detector. For details refer to chapter 2.7.

### 3.7 ATP Devices

Systems for Automatic Train Protection (ATP) can be intermittently or continuously. The latter ones are systems like LZB, TVM or ETCS, which comprise a control unit independent of the interlocking. This control unit is steering the track devices of the system. The intermittent systems have devices directly controlled from interlocking module or indirectly via a Lineside Electronic Unit (LEU). Such LEU derives the steering information from the control circuits the interlocking uses for signals or other elements. The principles of ATP devices are shown in the figure below.



- `<atpType>` – reference not yet implemented!
- `<atpDevice>` – reference not yet implemented!
- `<exitSignal>` – reference not yet implemented!
- `<entrySignal>` – reference not yet implemented!

### 3.8 Restricted Areas

The `RestrictedArea` is an abstract class, which is instantiated for any kind of special controlled area within the network. It will be then enriched by the particular classes of the area. The elements available for each instantiation are:

- `<isLimitedBy>` – This is the reference to track assets forming the limits of the defined area. The references shall be made preferable to interlocking elements.

#### 3.8.1 WorkZone

The work zone is an area that separates a part of the network for special purpose. The assets of the area are not available for normal operation or train traffic. The activation and deactivation is controlled by special means in order to ensure the safety of any workers in that zone. Subsequently such work zone cannot be revoked without proper action and consent from the outside staff.

A work zone is mainly activated for the protection of working gang from train traffic. There is normally no intention for any train movement inside it. If there are any movable elements released for local operation this will be for support of the working process (maintenance) only. The limits of the zone are defined by the end of TVD sections, i.e. axle detection points or insulated rail joints.

- **<activationLock>** – This is the reference to a lock or several of them which are used to activate the work zone and ensure its deactivation only after the workers did return the key/token into the lock.
- **<switchInPosition>** – The reference to any switch which shall be in a particular position for this work zone together with the protection side and the level of enforcement. For details refer to chapter 4.2.2 below.
- **<derailerInPosition>** – The reference to any derailer which shall be in a particular position for this work zone together with the protection side and the level of enforcement. For details refer to chapter 4.2.2 below.
- **<crossingInPosition>** – The reference to any movable crossing which shall be in a particular position for this work zone together with the protection side and the level of enforcement. For details refer to chapter 4.2.2 below.
- **<detectorInState>** – The reference to any detector which shall be in a particular state for this work zone together with the protection side and the level of enforcement. For details refer to chapter 4.2.2 below.
- **<signalWithAspect>** – The reference to any signal which shall show a particular aspect for this work zone together with the protection side and the level of enforcement. For details refer to chapter 4.2.2 below.
- **<keyLockInState>** – The reference to any key lock which shall be in a particular state for this work zone together with the protection side and the level of enforcement. For details refer to chapter 4.2.2 below.
- **<levelCrossingInState>** – The reference to any level crossing which shall be in a particular state for this work zone together with the protection side and the level of enforcement. For details refer to chapter 4.2.2 below.
  - **@protectingSide** – This marks whether the element is protecting the area from “inside”, “outside” or “none”.
- **<releasedForLocalOperation>** – This is the reference to any movable elements within the work zone which can be locally operated by keys/buttons located nearby.
- List of buttons for local operable assets – **not yet implemented**

### 3.8.2 LocalOperationArea

The local operation area brings the assets in a special mode where they can be operated freely from on-site devices, e.g. button panel near a switch. These assets are not available for any normal operation by the interlocking operator. The activation is done from the interlocking operator giving authorisation for the mode. The return of operational control might be done by command from the interlocking operator or a device on-site (special deactivation button). Local operation areas are mainly used for shunting purpose without the use of any route. The traffic

safety within this area is solely dependent on the on-site staff. The active status of a local operation area is indicated to the railway staff by special signals or special signal aspects. The area limits are defined by the end of TVD sections, i.e. by axle detection points or insulated rail joints.

- `<deactivationKey>` – This is the reference to a button device located outside within the area used by the shunter to return control of the area to the signalman.
- `<switchInPosition>` – The reference to any switch which shall be in a particular position for this local operation area together with the protection side and the level of enforcement. For details refer to chapter 4.2.2 below.
- `<derailerInPosition>` – The reference to any derailer which shall be in a particular position for this local operation area together with the protection side and the level of enforcement. For details refer to chapter 4.2.2 below.
- `<crossingInPosition>` – The reference to any movable crossing which shall be in a particular position for this local operation area together with the protection side and the level of enforcement. For details refer to chapter 4.2.2 below.
- `<detectorInState>` – The reference to any detector which shall be in a particular state for this local operation area together with the protection side and the level of enforcement. For details refer to chapter 4.2.2 below.
- `<signalWithAspect>` – The reference to any signal which shall show a particular aspect for this local operation area together with the protection side and the level of enforcement. For details refer to chapter 4.2.2 below.
- `<keyLockInState>` – The reference to any key lock which shall be in a particular state for this local operation area together with the protection side and the level of enforcement. For details refer to chapter 4.2.2 below.
- `<levelCrossingInState>` – The reference to any level crossing which shall be in a particular state for this local operation area together with the protection side and the level of enforcement. For details refer to chapter 4.2.2 below.
  - `@protectingSide` – This marks whether the element is protecting the area from “inside”, “outside” or “none”.
- `<releasedForLocalOperation>` – This is the reference to any movable element within the local operation area which can be locally operated by keys/buttons located nearby.
- list of buttons/panels for local operable assets – not yet implemented
- list of signals that may be start or end of a main/shunting route despite the active area – not yet implemented

### 3.8.3 ShuntingArea

In contrast to local operation area some operators need to define a more simple type of shunting zone for various purpose, e.g. for shunting of ETCS trains. Within ETCS a shunting area is used to limit an area for train movements without the need of MA. However, the train movement might be controlled by shunting routes or so. Subsequently the definition of such areas might not be known to an interlocking but it is needed inside RBC data.

The list of assets inside the area can be any net elements like TVD sections or balises. At least the list of balises permissible to pass in shunting mode is announced by the RBC to the train.

The limits of such an area are not necessarily train detection devices (end of TVD sections). However, the list shall refer to physical assets on the track like end of TVD sections, balises or fixed signals (boards).

The example shows the definition of an ETCS shunting zone including the entire station of Cstadt.

```
<shuntingZone id="rae01">
  <designator register="_SimpleRegister" entry="Cstadt RA1"/>
  <isLimitedBy ref="bus01"/>
  <isLimitedBy ref="bus02"/>
  <isLimitedBy ref="mb_sig03"/>
</shuntingZone>
```

### 3.8.4 PermissionZone

In order to have clear command path only one operator/controller is exclusively allowed to operate assets by the interlocking. However, the permission to operate assets can be shifted to another predefined operator/controller. Subsequently zones have to be defined, describing the set of assets for which the permission is transferred. In the simple case the permission can only be changed for the entire station or interlocking. Then no particular definition of affected assets might be necessary.

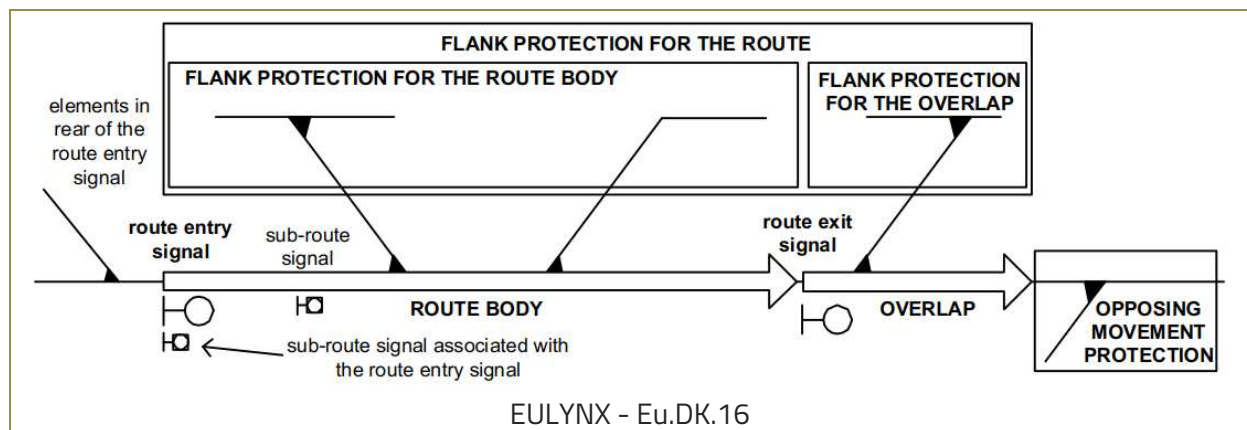
Although a permission zone might appear as another type of **RestrictedArea** on the first glance the way of defining the zone is different.

- **<canBeControlledBy>** – This is the reference to any controller, i.e. operator/train dispatcher place, which can use the zone for manual commands.
- **<controlledElement>** - This is the reference to any element which belong to the area and will have the same operating permissions.

## 4 Route

### 4.1 Definition

A route is a central element to ensure traffic safety in railway networks. The basic definition is a predetermined path for a traffic movement. This path starts from a starting point (usually a signal) and ends at a destination point (usually a signal), univocally identified by the list of switches and their position included in the path. In addition there may be elements outside the (running) path of the route. These elements are overlap, flank protection, TVD sections, intermediate signals, movable elements, level crossings etc. Although they are naturally important to route setting, they are defined in separate elements, which are not necessarily sub-elements of the route itself. See the route control relations, control table, route activation and signal plan section below.



A route in its purest form consists only of an entry and exit signal. If there are any switches on the route, their position is associated with the route. The interlocking need only be aware of “atomic” routes from one signal to the next. From a mathematical and modelling point of view, there’s no need to distinguish short “atomic” routes or “elementary” routes from concatenated paths or itineraries, however long.

Beside the physical track elements there are several elements in the `<assetsForIL>` container, which may be needed for the complete definition of a route within the interlocking logic. These are:

- `<route>` – This is the basic definition of the route itself.
- `PartialRoute`: `<routeReleaseGroupAhead>`, `<routeReleaseGroupRear>` – These are the definitions of route parts especially for route release control. For details refer to chapter 4.5.
- `<conflictingRoute>` – This is the list of other routes conflicting with it. For details refer to chapter 4.10.
- `<routeRelation>` – This is the list of additional conditions required for route setting and supervision. For details refer to chapter 4.8.
- `<combinedRoute>` – This is the list of atomic routes combined to a long route activated by a single command. For details refer to chapter 4.11.
- `<overlap>` – This is the extension of the route beyond the route destination for safety purpose. For details refer chapters 4.6.2 and 4.6.3.
- `<dangerPoint>` – This is the point beyond the route destination comprising a danger to the train if it would pass the destination. For details refer chapter 4.6.1.

When a route is requested, the interlocking sets and locks the objects that make the route safe. The interlocking will look up the objects associated with this route and set them accordingly.

The description of a route starts with the following information:

- `@locksAutomatically` – This optional flag indicates whether the route is automatically and immediately re-established and locked after it was cleared by previous train.
- `@processingDelay` – The delay time between the moment the interlocking receives the route call and the moment the interlocking reports back that the route is locked, i.e. the processing time for setting that route. There may be systems when the final route lock is only

applied when a train is approaching the route. In that case this time is the minimum required before the final route lock can be applied.

- `@proceedAspectDelay` – The time the opening of the start signal is delayed for operational reasons, i.e. to slow down an approaching train.
- `@signalClosureDelay` – The time the closure of start signal is delayed after the train has passed it, i.e. did sequentially occupy the first route track.
- `@approachReleaseDelay` – The delay time between the moment the interlocking receives the request to release an already approached (definitely locked) route and the interlocking really releases all associated elements from the route.
- `<handlesRouteType>` – This is the reference to the predefined route type applicable.
- `<routeActivationSection>` – This is the description of activation conditions for the route. For details refer chapter 4.4 below.
- `<facingSwitchInPosition>` – This is the list of the switches and their position traversed from their top to define the running path of the route. For details refer chapter 4.2.1 below.
- `<routeEntry>` – This defines the start of the route path, which is in most cases a signal. For details refer chapter 4.3 below.
- `<hasTvdSection>` – The list of references to the TVD sections within the running path of this route. It is preferred to have an ordered list beginning at `<routeEntry>`, however this cannot be enforced by the schema.
- `<hasReleaseGroup>` – This defines the parts of the route which are released automatically after being cleared by the train. For details refer chapter 4.5 below.
- `<switchInDepartureTrack>` – This is the list of switches and their position which are requested by the route. For details refer chapter 4.2.1 below.
- `<routeExit>` – This defines the destination of the route path including any danger point or overlap related to the route end. For details refer chapter 4.6 below.
- `<additionalRelation>` – The list of references to relations to elements with their position or state, which are relevant for the route. For details refer chapter 4.8 below.

Note that information about the route path, which is derivable by using the entry and exit objects together with the railway network, is left out of this definition. This includes TVD sections in the train path and movable objects in the train path.

The skeleton of a route definition is shown in the extract. It is a main route, which is not automatically set after first use. The time for route locking is given as 1 sec, which can only be achieved, if there are no movable elements to be switched. The remaining elements are detailed below.

```
<route id="rt_sig02_sig04" locksAutomatically="false" processingDelay="PT1S" >
  <designator register="_SimpleRegister" entry="Route_68N1_69A"/>
  <handlesRouteType ref="rt_main"/>
  <routeActivationSection ... />
  <facingSwitchInPosition ... />
  <hasTvdSection ... />
  <routeEntry ... />
  <hasReleaseGroup ... />
  <routeExit ... />
  <additionalRelation ref="rtr01"/>
</route>
```

## 4.2 State Space

### 4.2.1 AssetAndState

The railway network owns assets that can be associated with a state. This is expressed as a set of tuples, for instance, (signal<sub>i</sub> / aspect<sub>k</sub>). Other functional classes can make use of these tuples. The using of the state space is twofold:

1. direct use of (asset, state) tuples with class **AssetAndState** when just the state of the asset is needed
2. indirect use of (asset, state) tuples with class **AssetAndGivenState** when additional the level of enforcement and its proving is needed because it is not obvious from the context.

Because of the construct of an abstract class the instances of **AssetAndState** can use specific enumerations with the related states. The possible values in these enumerations are self-explanatory. In addition it can have the following attribute:

- **@isNegated** – This Boolean value shall be used to express the inversion, i.e. all other possible states or positions are meant except the given one.

### 4.2.2 AssetAndGivenState

For a clear definition of a **Route** it is necessary to have not only information about assets needed but also their state they shall assume during its use in the route. This is realised with elements like **<facingSwitchInPosition>** using the abstract class **AssetAndState** or **<requiredSwitchPosition>** using the class **AssetAndGivenState**. The latter one is similar build as **AssetAndState** but including the information as child element **<relatedAssetAndState>**.

The class **AssetAndState** is an abstract tuple of references to the asset and its related state. Depending on the particular instance the naming of these elements are varying.

- **<refersTo...>** – This is the reference to the particular asset in **AssetsForIL**.
- **<inPosition>**, **<inState>**, **<showsAspect>** – This is the related position/state/aspect of the asset like “left” for a switch.

There are several instantiations of the abstract class **AssetAndState**. These are:

- **CrossingAndPosition** – This combines a **MovableCrossing** with a possible position.
- **DeraillerAndPosition** – This combines a **DeraillerIL** with a possible position.
- **LockAndState** – This combines a **LockIL** with a possible state.
- **LevelCrossingAndState** – This combines a **LevelCrossingIL** with a possible state.
- **SectionAndVacancy** – This combines a **TvdSection** with a possible vacancy state.
- **SignalAndAspect** – This combines a **SignalIL** with reference to an aspect as defined in **GenericTypes**.
- **SwichPointAndPosition** – This combines a **SwitchIL** with a possible position.

The class **AssetAndGivenState** adds information on the level of enforcement and the way of proving it.

- **@mustOrShould** – This allows the definition how strong the state is required.
  - **“should”** – The state is the preferred one and should be used if possible (recommendation).
  - **“must”** – The state must be used in any case (obligation).
- **@proving** – This describes the way the state is proven.
  - **“staffAcknowledged”** – The asset is proven by staff to have the required state. This is acknowledged by manual input action (command) to the interlocking.
  - **“continuously”** – The interlocking checks the asset state continuously as long as the asset is required.
  - **“oneOff”** – The interlocking checks the asset state only once when request for the asset is made.
- **@isNegated** – This Boolean value shall be used to express the inversion, i.e. all other possible states or positions are meant except the given one.

The extract shows the sample definition of a particular switch position as used to describe the route path.

```
<facingSwitchInPosition id="rp_pt_swi01_li" inPosition="left">  
  <designator register="_SimpleRegister" entry="pt01 in left"/>  
  <refersToSwitch ref="pt_swi01"/>  
</facingSwitchInPosition>
```

The extended variant is shown here. The named track section must be occupied and is checked continuously for it.

```
<requiredSectionState mustOrShould="must" proving="continuously">  
  <relatedSectionAndVacancy inState="occupied">  
    <refersToSection ref="A02T" />  
  </relatedSectionAndVacancy>  
</requiredSectionState>
```

## 4.3 RouteEntry

The start of a route is described in **<routeEntry>**. The element contains the following information:

- **<refersTo>** – This is the reference to the start element of the route. In most cases this is a signal. Preferable the reference shall be made to the element with the interlocking attributes instead of the pure infrastructure element.
- **<nonReplacement>** – This is the reference to any track section in the route path which does not cause signal replacement on occupation, i.e. start signal is kept open until occupation of the next section. There can be more than one such section within one route.

The extract shows the route start from the virtual exit signal where the occupation of the first track, the switch A68W02 will not cause signal replacement.

```
<routeEntry id="rts_68N1">
```

```
<designator register="_SimpleRegister" entry="Start 68N1"/>
<refersTo ref="mb_sig02"/>
<nonReplacement ref="A68W02T"/>
</routeEntry>
```

## 4.4 Route activation

The route activation stands for two different scenarios. Commonly, the signalman or control system calls a route, which the interlocking then prepares. The route is activated (=locked) either after a given delay but more generally after an activation section turns occupied. This indicates an approaching train and the route shall not be released without special precautions. In the second scenario the automatic route setting is triggered as defined in route activation, i.e. on occupation of the referred track section the interlocking is triggered to set the related route. Usually the definition of the activation data is independently of the scenario, which it is used for.

A route may refer to a route activation section, which contains:

- **@delayForLock** – This is the time between the moment the approach section turns from vacant to occupied and the moment the interlocking locks the route.
- **@automaticReleaseDelay** – This is time between the moment that the route is locked because the approach section turned occupied, and the possible release of the route. This delay for automatic delay would typically be used when an approach train stops in an approach section but fails to enter the route.
- **<activationSection>** – This is the reference to the TVD section activating the route when this section turns from vacant to occupied. The activation means setting the route or lock it due to the approaching train.

The example shows the route is activated when the section in rear of the route entry is occupied.

```
<routeActivationSection id="rt_act01" delayForLock="PT2S" automaticReleaseDelay="PT5S">
  <designator register="_SimpleRegister" entry="activation Route_68N1_69A"/>
  <activationSection ref="A02T"/>
</routeActivationSection>
```

## 4.5 PartialRoute

The elements of a route are automatically released after use by a train. This release can be for the whole route at once or parts of it. The trigger for release is the clearing of TVD sections after sequential occupation. Thus the TVD section are of interest for release groups as all other elements are indirectly included in such groups. If one or more TVD section is released in a group, the characteristics are detailed as **PartialRoute**. However, if the interlocking in general releases each individual TVD section after use the release groups can be omitted.

The instantiation extends the abstract type of **PartialRoute** by some additional elements and attributes.

The abstract class contains

- **@delay** – Duration after which the interlocking releases the partial route. Starts counting from the moment that all the conditions for release are fulfilled. This delay is engineered in

static data. If not defined, the interlocking releases the group without delay.

If the route has only one route release group then the set of TVD sections in the route is released en bloc with the delay given here.

- **@typicalDelay** – Duration after which the partial route is typically released. Use this delay for simulation purposes. Starts counting from the moment that the interlocking has received all conditions for the release. E.g. TVD sections in the group have been vacated, timers expired.
- **<hasTvdSection>** – The reference to the TVD section contained in the release group. There can be more than one section referred to.

#### 4.5.1 RouteReleaseGroupAhead

The route sections are typically a list of TVD sections to be released as a unit ahead of a stationary train. This is mainly applicable for ETCS operation and shall be defined in destination areas like berthing tracks inside stations. It shall be used in conjunction with train initiated overlap release.

There is one additional attribute:

- **@isAutomatic** – This Boolean value indicates whether this section is automatically released when the train reports being stationary.

#### 4.5.2 RouteReleaseGroupRear

Release groups are typically a list of TVD sections to be released as a unit in rear of the passing train. The TVD sections of one group may cover only a part of a route. These release groups are necessary if several elements within the route are to be released together, i.e. each one is cleared by the passing train. A good example is a pair of coupled switches. They are kept in the route until both of them are cleared.

The **Route** lists each of its release groups in **<hasReleaseGroup>**. However, this is only a reference to the instantiations of **PartialRoute** for a particular **RouteReleaseGroup** as defined in **<knowsPartialRoute>**. The release groups are only referenced in the route definition as shown here.

```
<hasReleaseGroup ref="prt02"/>
<hasReleaseGroup ref="prt03"/>
```

The related details of the release groups are defined in the known assets.

```
<routeReleaseGroupRear delay="PT1S" id="prt02" typicalDelay="PT2S">
  <designator register="_SimpleRegister" entry="tm_A02T"/>
  <hasTvdSection ref="A02T"/>
</routeReleaseGroupRear>
<routeReleaseGroupRear delay="PT10S" id="prt03" typicalDelay="PT7S">
  <designator register="_SimpleRegister" entry="tm_A68W02T"/>
  <hasTvdSection ref="A68W02T"/>
</routeReleaseGroupRear>
```

This extract show the combination of two sections within one release group.

```
<hasReleaseGroup ref="prt02-03"/>
...
<routeReleaseGroupRear delay="PT1S" id="prt02-03" typicalDelay="PT2S">
```

```
<designator register="_SimpleRegister" entry="tm_A02T+A68W02T"/>
<hasTvdSection ref="A02T"/>
<hasTvdSection ref="A68W02T"/>
</routeReleaseGroupRear>
```

## 4.6 RouteExit

The end of a route path as its destination is defined in `<RouteExit>`. This element refers onto the asset, which marks this destination. It may be enhanced by information about necessary safety precautions in advance of the destination signal.

- `<refersTo>` – This is the reference to the physical end of the route path. This is in most cases a signal but can also be a buffer stop or just a train detection device when continuing into a track area outside the interlocking responsibility ("terra incognita"). Preferable the reference shall be made to the element with the interlocking attributes instead of the pure infrastructure element.
- `<hasDangerPoint>` – This is the reference to the danger point in advance of the destination signal. For details refer chapter 4.6.1 below.
- `<hasOverlap>` – This is the reference to the path in advance of the destination signal as safety precaution. For details refer chapter 4.6.2 below.

The extract shows the skeleton of a `<RouteExit>`.

```
<routeExit id="rtd_69A">
  <designator register="_SimpleRegister" entry="Dest 69A"/>
  <refersTo ref="ls_sig04"/>
  <hasDangerPoint ref="dp01" />
  <hasOverlap ref="ov01" />
</routeExit>
```

### 4.6.1 DangerPoint

A route exit may refer to a danger point. The danger point defines the position in advance of the destination signal up to where a train is likely to be safe. The danger point is not necessarily related to a train detection element.

The element can take the following information:

- `@distance` – This is the distance in metres from the destination signal to the physical danger point.
- `@releaseSpeed` – The release speed in km/h for release from controlled braking curve associated with the danger point. A release speed is a speed limit under which the train is allowed to run in the vicinity of the EoA (End of Authority: here `routeExit` or `DestinationPoint`), when the target speed is zero. One release speed can be associated with the Danger Point.
- `<lastSupervisedSectionBeforeDP>` – This is the reference to last TVD section which is completely before the danger point. It is used when the danger point is situated at the end of a TVD section.

- `<situatedAtTrackAsset>` – This is the reference to the track asset the danger point is located. It is used when the danger point is situated in the middle of a TVD section or beyond any TVD section in case of not supervised tracks.

```
<dangerPoint id="dp01" distance="300.0" releaseSpeed="0.0">
  <designator register="_SimpleRegister" entry="DPe69P2"/>
  <lastSupervisedSectionBeforeDP ref="B69W03T"/>
  <situatedAtTrackAsset ref="B01T"/>
</dangerPoint>
```

## 4.6.2 Overlap

Overlap is a defined number of track sections in advance of a stop signal, or a stopping point in a continuous signalling system, which must be kept clear to avoid the risk of collision should the train inadvertently run past the signal or the stopping point.

Many IMs require overlap in advance of active routes to protect from overshoot. One or more sections in advance of the destination signal are locked out from use by other routes. The overlap is delimited by train detectors. Facing switches within the overlap are locked. Trailing switches in the overlap are may be not locked. Note that there is no need to explicitly identify the switches in the overlap because they can be derived from the begin and endpoints of the overlap. There may be several overlaps defined per destination signal.

The overlap can be released if the RBC deems that an approaching train is slow enough such that overshoot is unlikely.

Trains other than the one for which the route-overlap is locked may be attributed a permitted speed in the overlap. If the value is set to “0” it is not possible to set a route through the overlap. Compare this variable with the release speed that applies to the train that is being released beyond the danger point.

The overlap element can have the following information:

- `@overlapValidityTime` – The overlap validity time is the time that the train assumes the overlap to be locked. This is the ETCS validity time  $T_{OL}$ .
- `@speedInOverlap` – Maximum speed in the overlap in km/h for other trains than the one using the related route.
- `@releaseSpeed` – Maximum speed in km/h for release from controlled braking curve when running towards the overlap. A release speed is a speed limit under which the train is allowed to run in the vicinity of the EoA (End of Authority: here `routeExit` or `DestinationPoint`), when the target speed is zero. One release speed can be associated with the Overlap.
- `@length` – The physical length of the overlap in metres can be given here as alternative to a particular track asset as limitation.
- `<activeForApproachRoute>` – This is the reference to the related route using this overlap.
- `<relatedToTrackAsset>` – This is the reference to the related track asset the overlap is connected to. This can be e.g. the destination signal of one or several routes.

- `<requiresAssetInPosition>` – This is the definition of any movable element located within the overlap which is required to be in a certain position. For details refer to chapter 4.2.2 above.
- `<hasTvdSection>` – The list of references to the TVD sections within the path of this overlap. It is preferred to have an ordered list beginning at `<routeExit>`, however this cannot be enforced.
- `<isLimitedBy>` – This is the reference to train detection element limiting the overlap. It can be more than one limiting element. It shall be used in conjunction to `@length`.
- `<overlapRelease>` – This defines the way of automatic release of overlap after use. For details refer chapter 4.6.3 below.

### 4.6.3 OverlapRelease

The automatic release of an overlap after use is detailed in `<overlapRelease>`. The interlocking typically releases the overlap when the train approaching the destination signal has plausibly stopped. This is assumed a pre-set time after the train has either occupied or vacated a trigger section. Typically, the overlap release timer starts when the train occupies the section ending at the destination signal. Alternatively, the interlocking may release the overlap immediately or with a short delay after the train positively confirms to have stopped, e.g. by means of an ETCS position report.

The information for overlap release is described as follows:

- `<releaseTriggerSection>` – This is the reference to the TVD section which is used as trigger for the overlap release.
- `<overlapReleaseTimer>` – This details the timing for the overlap release.
- `@timer` – The time that has to be elapsed after the trigger and before the overlap is physically released.
- `@overlapReleaseCondition` – This defines the trigger condition for the release timer to start.
  - `"trainInitiatedTrigger"` – The release timer is triggered when the train has reported its halt.
  - `"startTimerAfterVacating"` – The release timer is triggered when the trigger section becomes clear after sequential occupation. Typically this refers to the section before the last one, i.e. the train has completely entered the last route section.
  - `"startTimerUponOccupation"` – The release timer is triggered when the trigger section becomes sequentially occupied. Typically this refers to the last section of the route.

The extract shows an overlap from signal 69A to the end of switch 69W03 inclusively. It is released after 60 s of occupation of the last track section in rear of signal 69A. The position of switch 69W03 in left is only check upon establishing the overlap.

```
<overlap id="ov01" overlapValidityTime="PT60S" overlapSpeed="0.0">
  <designator register="_SimpleRegister" entry="Overlap 69A-P2"/>
  <activeForApproachRoute ref="rt_sig02_sig04"/>
  <requiresSwitchInPosition mustOrShould="should" proving="oneOff">
```

```

    <relatedSwitchAndPosition inPosition="left">
      <refersToSwitch ref="pt_swi02" />
    </relatedSwitchAndPosition>
  </requiresSwitchInPosition>
  <hasTvdSection ref="B03T"/>
  <hasTvdSection ref="B69W03T"/>
  <isLimitedBy ref="tde07"/>
  <overlapRelease id="ov01_rl">
    <designator register="_SimpleRegister" entry="ov01 Release"/>
    <releaseTriggerSection ref="X02T"/>
    <overlapReleaseTimer timerValue="PT60S"
      overlapReleaseCondition="startTimerUponOccupation" />
  </overlapRelease>
</overlap>

```

## 4.7 Complete Route

Finally this extract shows the complete definition of a route. The details were already explained above.

```

<route id="rt_sig02_sig04" locksAutomatically="false" processingDelay="PT1S" >
  <designator register="_SimpleRegister" entry="Route_68N1_69A"/>
  <handlesRouteType ref="rt_main"/>
  <routeActivationSection id="rt_act01" delayForLock="PT2S" automaticReleaseDelay="PT5S">
    <designator register="_SimpleRegister" entry="activation Route_68N1_69A"/>
    <activationSection ref="A02T"/>
  </routeActivationSection>
  <facingSwitchInPosition id="rp_pt_swi01_li" inPosition="left">
    <designator register="_SimpleRegister" entry="pt01 in left"/>
    <refersToSwitch ref="pt_swi01"/>
  </facingSwitchInPosition>
  <hasTvdSection ref="A68W02T"/>
  <hasTvdSection ref="X01T"/>
  <hasTvdSection ref="LX2.5T"/>
  <hasTvdSection ref="X02T"/>
  <routeEntry id="rts_68N1">
    <designator register="_SimpleRegister" entry="Start 68N1"/>
    <refersTo ref="mb_sig02"/>
    <nonReplacement ref="A68W02T"/>
  </routeEntry>
  <hasReleaseGroup ref="prt02"/>
  <hasReleaseGroup ref="prt03"/>
  <hasReleaseGroup ref="prt04"/>
  <hasReleaseGroup ref="prt05"/>
  <hasReleaseGroup ref="prt06"/>
  <routeExit id="rtd_69A">
    <designator register="_SimpleRegister" entry="Dest 69A"/>
    <refersTo ref="ls_sig04"/>
    <hasDangerPoint ref="dp01" />
    <hasOverlap ref="ov01" />
  </routeExit>
  <additionalRelation ref="rtr01"/>
</route>
...
<overlap id="ov01" overlapValidityTime="PT60S" overlapSpeed="0.0">
  <designator register="_SimpleRegister" entry="Overlap 69A-P2"/>
  <activeForApproachRoute ref="rt_sig02_sig04"/>
  <requiresSwitchInPosition mustOrShould="should" proving="oneOff">
    <relatedSwitchAndPosition inPosition="left">
      <refersToSwitch ref="pt_swi02" />
    </relatedSwitchAndPosition>
  </requiresSwitchInPosition>

```

```

</requiresSwitchInPosition>
<hasTvdSection ref="B03T"/>
<hasTvdSection ref="B69W03T"/>
<isLimitedBy ref="tde07"/>
<overlapRelease id="ov01_rl">
  <designator register="_SimpleRegister" entry="ov01 Release"/>
  <releaseTriggerSection ref="X02T"/>
  <overlapReleaseTimer timerValue="PT60S"
    overlapReleaseCondition="startTimerUponOccupation" />
</overlapRelease>
</overlap>
...
<dangerPoint id="dp01" distance="300.0" releaseSpeed="0.0">
  <designator register="_SimpleRegister" entry="DPe69P2"/>
  <lastSupervisedSectionBeforeDP ref="B69W03T"/>
  <situatedAtTrackAsset ref="B01T"/>
</dangerPoint>

```

## 4.8 RouteRelation

A route relation states the conditions that must be fulfilled for the route start signal to be open. The relation consists of the affected route and any track asset with its related state. Subsequently the signal may only open when the relation is fulfilled. One possible use is for definition of specific flank protection conditions.

Note that these relations may well be captured in a control table, **once it is implemented in the schema**. Therefore, the use is optional.

The relationship is described in `<routeRelation>` for the various types of assets. For details refer chapter 4.2.2 above.

- `<requiredSwitchPosition>` – This is the reference to the related switch, its position and the level of enforcement.
- `<requiredDeraillerPosition>` – This is the reference to the related derailer, its position and the level of enforcement.
- `<requiredCrossingPosition>` – This is the reference to the related movable crossing, its position and the level of enforcement.
- `<requiredDetectorState>` – This is the reference to the related detector, its state and the level of enforcement.
- `<requiredSignalAspect>` – This is the reference to the related signal, the aspect it shows and the level of enforcement.
- `<requiredSectionState>` – This is the reference to the related TVD section, its state and the level of enforcement.
- `<requiredKeyLockState>` – This is the reference to the related key lock, its state and the level of enforcement.
- `<requiredLevelCrossingState>` – This is the reference to the related level crossing, its state and the level of enforcement.

The extract shows the relationship between the occupation status of the start track A02T and the route from start signal 68N1 to destination signal 69A.

```
<routeRelation id="rtr01">
  <requiredSectionState mustOrShould="must" proving="continuously">
    <relatedSectionAndVacancy inState="occupied">
      <refersToSection ref="A02T" />
    </relatedSectionAndVacancy>
  </requiredSectionState>
</routeRelation>
```

## 4.9 Control table

not yet implemented!

Typically, the signaller calls the route after which the interlocking sets the elements in the state as required per control table. Control tables, colloquially referred to as route tables or interlocking tables, are sets of elements associated with a required state. If all elements are asserted in that state, the route is locked and made available to a particular train. In other words, if all track elements are in the correct state, the interlocking sets the entry signal to proceed. A route is composed of a number of track elements, and their respective status. Track elements are grouped in (1) approach locking, (2) en route elements and (3) off route elements.

1. When an approach section is occupied, the interlocking locks the route.

2. The en route elements are track elements that the train encounters. The status of these elements is given. The route is locked only when the elements report this status. The elements are ordered as the train encounters them. This allows graph traversal algorithms to compile the list.

3. Off route elements typically provide flank protection and overlap. The elements must be in the given position for the route to be locked. These elements are by nature not situated within the route, hence the name off route.

## 4.10 Conflicting Routes

In general the interlocking logic can identify and exclude any conflicts of route combinations which requires elements in contradicting way. However, there may be topologies and rules where such conflicts are not obvious from the physical data. Such conflicts shall be defined in particular.

The route conflict table identifies the routes that may never be simultaneously allocated, due to utilisation of common track elements. Conflicting routes are listed as part of the AssetsForIL container. The element `<knowsConflictingRoute>` is a tuple of references to both routes that cannot be used simultaneously plus a list of conflict reasons. In most cases there will be two entries for both views of the combination, which might be seen as duplicate information. However, it cannot be excluded that the combinations are not symmetrically. Therefore each conflict view shall be listed, especially the conflict reason may differ. The particular information is contained in:

- `<refersToRoute>` – This is the reference to the route that shall be established.
- `<conflictsWithRoute>` – This is the reference to the route that conflicts with the selected route. This can be more than one route. However, only the routes shall be listed that have the same combination of conflict reasons.

- **<reasonForConflict>** – This contains the pair of conflict reasons in **@origin** with the reference to the causing element in **@refersTo**. Both attributes are mandatory when listing a conflict. The list of possible conflict reasons is
  - **“ConflictingOverlap”** – This value is set in case of the overlap of either route causes the conflict.
  - **“ConflictingSwitchPosition”** – This value is used in case any movable element, not only switches, required by the route needs another position then the conflicting route.
  - **“ConflictingStatus”** – This value is used in case the status of any element like logical device or signal is not in accordance with the needs of the route because of the conflicting route.
  - **“OverlappingTVDsection”** – This value is used in case of any TVD section of the route is also used by the conflicting one.
  - **“ConflictingHeadProtection”** – This value is used in case any signal that shall provide flank/head protection to either route is used as route destination for the other route, which is excluded to be simultaneously.
  - **“other:...”** – There is another conflict reason. This is the optional extension of the list. Each entry needs to start with the string “other:” and shall have at least two letters in addition.
- **@refersTo** – This is the reference to the track asset causing the conflict. This is typically a **TVDsection**, **MovableElement**, **SignalIL** or **LogicalDevice**.

The example shows the two exit routes from ARN station, which are obviously exclusive to each other. For such cases normally the conflict does not need to be defined explicitly. However, the example shows that more than one conflict reason may be named and both “sides” of the conflict shall be defined. Although not in the simple case here but the conflict reasons may differ when looking from the other side. It shall be noted that for simplification only one overlapping TVD section is named in the example.

```
<conflictingRoutes>
  <conflictingRoute id="crt_01">
    <designator register="_SimpleRegister" entry="conflict rt1-rt2"/>
    <refersToRoute ref="rt_sig02_sig04"/>
    <conflictsWithRoute ref="rt_sig01_sig04"/>
    <reasonForConflict origin="conflictingSwitchPosition" refersTo="pt_swi01"/>
    <reasonForConflict origin="overlappingTVDsection" refersTo="X01T"/>
  </conflictingRoute>
  <conflictingRoute id="crt_02">
    <designator register="_SimpleRegister" entry="conflict rt2-rt1"/>
    <refersToRoute ref="rt_sig01_sig04"/>
    <conflictsWithRoute ref="rt_sig02_sig04"/>
    <reasonForConflict origin="conflictingSwitchPosition" refersTo="pt_swi01"/>
    <reasonForConflict origin="overlappingTVDsection" refersTo="X01T"/>
  </conflictingRoute>
</conflictingRoutes>
```

## 4.11 Combined Routes

In order to reduce operational effort it is possible to define route combinations that are established by one operator command. The routes of one combination have to be all of the same type and direction. The list of single routes in the combination shall form a continuous path from the `<comboEntry>` to the `<comboExit>`.

- `<comboEntry>` – This is the reference to the start element of first route in the combination.
- `<comboExit>` – This is the reference to the destination element of the last route in the combination.
- `<containsRoute>` – This is the reference to the single route contained in the combination. All route must have the same type. It is preferable that the list of contained routes in the right order from start to end. However, this cannot be enforced by railML.



The above shown sequence of two routes from Sig1 to Sig3 is used for the example of a combined route.

```
<combinedRoute id="crt01">
  <designator register="_SimpleRegister" entry="crt_sig01_sig03"/>
  <comboEntry ref="sig01"/>
  <comboExit ref="sig03"/>
  <containsRoute ref="rt_sig01_sig02"/>
  <containsRoute ref="rt_sig02_sig03"/>
</combinedRoute>
```

## 4.12 DestinationPoint

Especially for systems without explicit routes but limited running path handling like RBC of ETCS there may be definitive destination points related to a physical track asset without a particular route but with the definition of overlaps and/or danger points. Therefore it is possible to have a list of such destination points within `<AssetsForIL>`. The container `<knowsDestinationPoint>` has the same structure as `<RouteExit>`. Refer also chapter 4.6 above.

- `<refersTo>` – This is the reference to the physical end of the route path. This is can be any physical track asset from infrastructure marking the path end.
- `<hasDangerPoint>` – This is the reference to the danger point in advance of the destination point. For details refer chapter 4.6.1 above.
- `<hasOverlap>` – This is the reference to the path in advance of the destination point as safety precaution. For details refer chapter 4.6.2 above.

The example shows a destination point at signal 69A. It shall be noted the reference `"sig04"` does not refer to the `<signalIL>` element in the interlocking but the `<signalIS>` element in infrastructure.

```
<destinationPoint id="dstp_69A1">
  <designator register="_SimpleRegister" entry="Dest 69A"/>
  <refersTo ref="sig04"/>
</destinationPoint>
```

```
<hasDangerPoint ref="dp02" />
<hasOverlap ref="ov02" />
</destinationPoint>
```

## 5 SignalBox (Interlocking)

The logic for control of track assets and train traffic is housed in a physical compartment traditionally called signalbox. In order to avoid ambiguity one has to bear in mind the clear discrimination. The entire sub-scheme is called as “interlocking” but the term is not used here without addition of “scheme”. Whenever the pure term interlocking is used, it will refer to a particular interlocking logic located inside a signalbox.

The `<SignalBox>` combines the information of the related network in the interlocking. Therefore the description of is mainly a list of references to the related items needed for the interlocking logic.

- `<controlsSystemAsset>` – This is the reference to any system asset which is controlled by this interlocking. In addition it gives also the extent of control the interlocking has of the asset.
- `<controlsTrackAsset>` – This is the reference to any track asset which is controlled by this interlocking. In addition it gives also the extent of control the interlocking has of the asset.
- `<controlsRoute>` – This is the reference to any route the interlocking controls.
- `<controlsCombinedRoute>` – This is the reference to any route combination the interlocking controls. For details refer to chapter 4.11.
- `<controlsInterface>` – This the reference to any defined interfaces to other interlocking. For details refer to chapter 6.3.
- `<controlledBy>` – This is the reference to any controller which can control this interlocking.
- `<implementsRouteRelation>` – This is the reference to any defined route relations the interlocking shall handle. For details refer to chapter 4.8.
- `<implementsSignalplan>` – This contains the list of signal aspect relations the interlocking has to consider when steering any signals. For details refer to chapter 5.2.
- `<implementsElementGroup>` – This contains the list of element groups that are operated together with a common command. Additionally the type of group is given as reference to a definition in `<hasElementGroupType>`. The referred elements in this list must be all controlled by this interlocking to the same level of extent.
- `<hasPermissionZone>` – This is the reference to any permission zone within the area controlled by this interlocking.
- `<hasConflictingRoutes>` – This is the reference to any route pairing that cannot be used simultaneously. For details refer to chapter 4.10.
- `<hasConfiguration>` – This gives some general information about the particular interlocking. For details refer to chapter 5.3.

The example shows an interlocking with a simplified list of features. It gets status information from the power supply “ups01”. It fully controls the TVD section “A01T” and the switch “pt\_swi01”. It commands the virtual signal “mb\_sig01” without getting any status information back. This `@extentOfControl` is explained in chapter 5.1. This interlocking controls the routes

“rt\_sig02\_sig04” and “rt\_sig01\_sig04” and has a related signalplan for the routes. In addition it knows the group “estop01” to set the listed signals to stop and has some configuration information.

```
<signalBox id="ilx01">
  <designator register="_SimpleRegister" entry="ILX-ARN"/>
  <controlsSystemAsset extentOfControl="notificationOnly">
    <connectedSystemAsset ref="ups01"/>
  </controlsSystemAsset>
  <controlsTrackAsset extentOfControl="fullControl">
    <connectedTrackAsset ref="A01T"/>
  </controlsTrackAsset>
  ...
  <controlsTrackAsset extentOfControl="fullControl">
    <connectedTrackAsset ref="pt_swi01"/>
  </controlsTrackAsset>
  <controlsTrackAsset extentOfControl="steeringOnly">
    <connectedTrackAsset ref="mb_sig01"/>
  </controlsTrackAsset>
  ...
  <controlsRoute ref="rt_sig02_sig04"/>
  <controlsRoute ref="rt_sig01_sig04"/>
  <implementsSignalplan id="sipaAC">
    <designator register="_SimpleRegister" entry="signalplan ARN-CST"/>
    <aspectRelation expectingSpeed="0.0" passingSpeed="60.0" endSectionTime="PT30S"
      id="sip01">
      <designator register="_SimpleRegister" entry="aspects 68N1-69A"/>
      <masterAspect>
        <refersToSignal ref="ls_sig04"/>
        <showsAspect ref="sig_caution_23"/>
      </masterAspect>
      <slaveAspect>
        <refersToSignal ref="mb_sig02"/>
        <showsAspect ref="sig_reducproceed_21"/>
      </slaveAspect>
      <signalsSpeedProfile ref="sps01"/>
      <appliesToRoute ref="rt_sig02_sig04"/>
    </aspectRelation>
  </implementsSignalplan>
  <implementsElementGroup id="estopARN">
    <designator register="_SimpleRegister" entry="Stop ARN"/>
    <groupType ref="estop01" />
    <refersToMember ref="mb_sig01" />
    <refersToMember ref="mb_sig02" />
    <refersToMember ref="mb_sig03" />
  </implementsElementGroup>
  <hasConfiguration technologyType="electronic" model="Westrace 0815"
    signalSystem="ETCS L2" SWversion="WR31.0.12" />
</signalBox>
```

## 5.1 ExtentOfControl

The interlocking knows an amount of track assets and routes, however, not for each and every element it has the full steering authority. Some elements just notify their status to the interlocking, e.g. an autonomous working level crossing. In order to mark this level of control authority the attribute `@extentOfControl` is combined with each asset reference. It can have the following values:

- “**steeringOnly**” – The control is one sided only in command (steering) direction, i.e. there is no feedback on the command actions foreseen.
- “**fullControl**” – The control covers both directions – commanding and status messages as feedback.
- “**notificationOnly**” – The control is one sided only in message (notification) direction, i.e. only the status messages can be received but no command action is possible.
- “**none**” – There is no control whatsoever possible. This will be used for any assets where the physical presence is needed as information.

## 5.2 SignalPlan

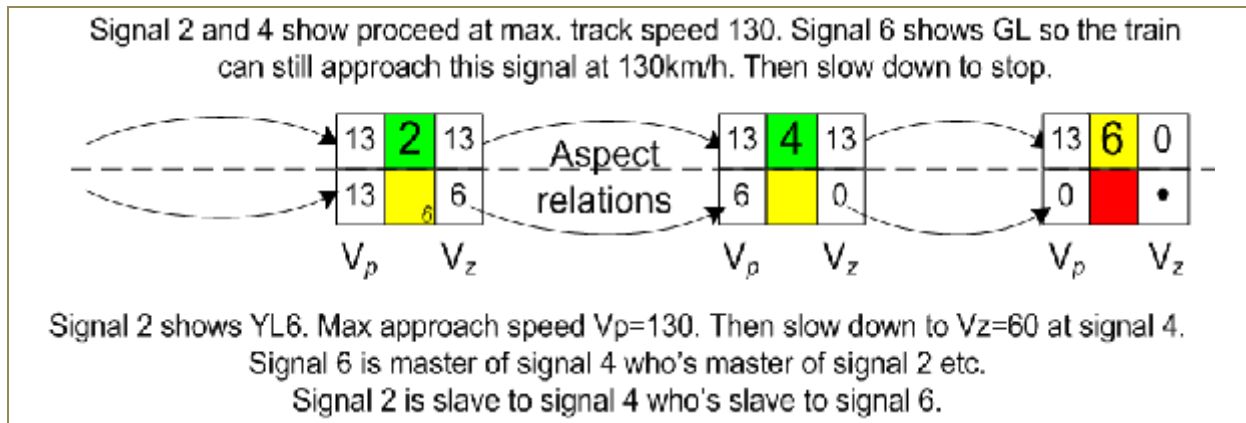
Signals protect the path as defined in `<knowsRoute>`. Depending on the particular signalling system used there is not a strict 1:1 relation between the signal aspect and the route as the aspect at the route entry may depend on the aspect of the next signal. Therefore the so-called signal plan is used with element `<implementsSignalplan>` as part of the interlocking features in `<signalBox>`. It consists of a set of aspect relations whereas the master is always the signal influencing the other. In case of signalling for the route itself, the master is the destination signal of the route and the slave is the start signal of the route. In case of signalling with stand-alone distant signals there is no master and the slave is the start signal of the route. However, with railML3.1 there can be currently no relation described between any main signal and distant signal or repeater.

Each aspect relation consists of:

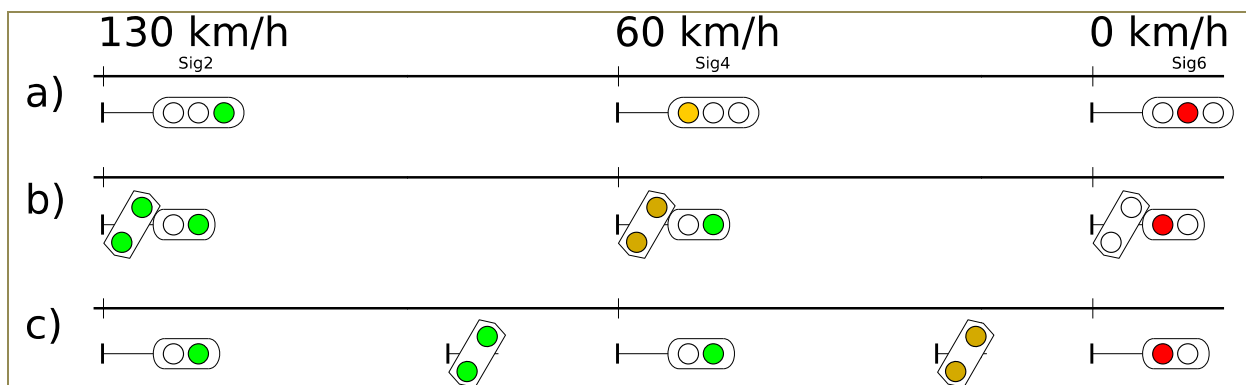
- `@expectingSpeed` – This is the maximum signalled speed in km/h at master signal at route exit (aka target speed).
- `@passingSpeed` – This is the speed in km/h signalled by the slave aspect, i.e. it is the speed that the train must respect when passing the slave signal at route entry.
- `@endSectionTime` – The end-section of a route is the section between the closed route exit signal and the previous slave signal. Commonly, the interlocking revokes (part of) the route when this timer expires.
- `<masterAspect>` – This is the status of the master signal (at route exit), i.e. the tuple of signal and aspects the signal is showing.
- `<slaveAspect>` – This is the status of the slave signal (at route entry) , i.e. the tuple of signal and aspects the signal is showing.
- `<distantAspect>` – This is the status of the distant signal (within the route or its entry) related to the signal with the `<masterAspect>`, i.e. the tuple of signal and aspects the signal is showing. This element shall be used for “old-style” signalling systems with separate distant signals or in case of repeaters within the route.
- `<signalsSpeedProfile>` – This is the reference to a `<speedSection>` in the infrastructure in case there are sections of the route with speed targets deviating from the signalled speed. The particular speed value and the section position can be derived from the infrastructure element.

- **<appliesToRoute>** – This is the reference to the route this aspect relation applies to. With knowledge of the route path the signals at route entry and exit can be identified.

Here are some illustrations for aspect relations and signalled speed.



The diagram below depicts the situation as described in the above signal plan example for the lower relation. In order to allow discussion of differences the signal aspect relations are shown for the typical variants of signalling systems, i.e. combined signalling (a), distant signal at the same mast (b) and stand-alone distant signal (c). The used colours and appearance of the signals in diagram are just symbolic and does not reflect a particular system.



The extract shows the railML representation of the above signal plan illustration for both relations. Although the physical appearance of the signal aspects according to the above variants a)...c) are different, the railML representation would only differ in the related aspect, if not considering the separate distant signals. The aspect of distant signals may not to be included here if the relation to the main signal is fix.

```
<implementsSignalplan id="sipaIL">
  <designator register="_SimpleRegister" entry="signalplan illustration"/>
  <aspectRelation expectingSpeed="130.0" passingSpeed="130.0" endSectionTime="PT30S"
    id="sip01">
    <designator register="_SimpleRegister" entry="aspects 2-4 upper"/>
    <masterAspect>
      <refersToSignal ref="sig4"/>
      <showsAspect ref="sig_fullproceed"/>
    </masterAspect>
    <slaveAspect>
      <refersToSignal ref="sig2"/>
      <showsAspect ref="sig_fullproceed"/>
    </slaveAspect>
  </aspectRelation>
</implementsSignalplan>
```

```

    <appliesToRoute ref="rt_sig02_sig04"/>
  </aspectRelation>
  <aspectRelation expectingSpeed="60.0" passingSpeed="130.0" endSectionTime="PT30S"
    id="sip02">
    <designator register="_SimpleRegister" entry="aspects 4-6 upper"/>
    <masterAspect>
      <refersToSignal ref="sig6"/>
      <showsAspect ref="sig_YL6"/>
    </masterAspect>
    <slaveAspect>
      <refersToSignal ref="sig4"/>
      <showsAspect ref="sig_fullproceed"/>
    </slaveAspect>
    <appliesToRoute ref="rt_sig04_sig06"/>
  </aspectRelation>
  <aspectRelation expectingSpeed="60.0" passingSpeed="130.0" endSectionTime="PT30S"
    id="sip03">
    <designator register="_SimpleRegister" entry="aspects 2-4 lower"/>
    <masterAspect>
      <refersToSignal ref="sig4"/>
      <showsAspect ref="sig_YL6"/>
    </masterAspect>
    <slaveAspect>
      <refersToSignal ref="sig2"/>
      <showsAspect ref="sig_GL"/>
    </slaveAspect>
    <appliesToRoute ref="rt_sig02_sig04"/>
  </aspectRelation>
  <aspectRelation expectingSpeed="0.0" passingSpeed="60.0" endSectionTime="PT30S"
    id="sip04">
    <designator register="_SimpleRegister" entry="aspects 4-6 lower"/>
    <masterAspect>
      <refersToSignal ref="sig6"/>
      <showsAspect ref="sig_Stop"/>
    </masterAspect>
    <slaveAspect>
      <refersToSignal ref="sig4"/>
      <showsAspect ref="sig_YL6"/>
    </slaveAspect>
    <appliesToRoute ref="rt_sig04_sig06"/>
  </aspectRelation>
</implementsSignalplan>

```

The following example shows the variations to reflecting the distant signals in the variants b)...c) from the above picture but using the aspects as defined in the simple example for variants b)...c). The differences between variant b) and c) would be in the location of the referred distant signals only.

Variant a)

```

<implementsSignalplan id="sipaIL_a">
  <designator register="_SimpleRegister" entry="signalplan illustration variant a)"/>
  <aspectRelation expectingSpeed="60.0" passingSpeed="130.0" endSectionTime="PT30S"
    id="sip03">
    <designator register="_SimpleRegister" entry="aspects 2-4 variant a)"/>
    <masterAspect>
      <refersToSignal ref="sig4"/>
      <showsAspect ref="sig_YL6"/>
    </masterAspect>
    <slaveAspect>
      <refersToSignal ref="sig2"/>
      <showsAspect ref="sig_GL"/>
    </slaveAspect>
  </aspectRelation>

```

```

    <appliesToRoute ref="rt_sig02_sig04"/>
  </aspectRelation>
  <aspectRelation expectingSpeed="0.0" passingSpeed="60.0" endSectionTime="PT30S"
    id="sip04">
    <designator register="_SimpleRegister" entry="aspects 4-6 variant a)"/>
    <masterAspect>
      <refersToSignal ref="sig6"/>
      <showsAspect ref="sig_Stop"/>
    </masterAspect>
    <slaveAspect>
      <refersToSignal ref="sig4"/>
      <showsAspect ref="sig_YL6"/>
    </slaveAspect>
    <appliesToRoute ref="rt_sig04_sig06"/>
  </aspectRelation>
</implementsSignalplan>

```

Variants b)...c) differ only by the position of the distant signals in the infrastructure.

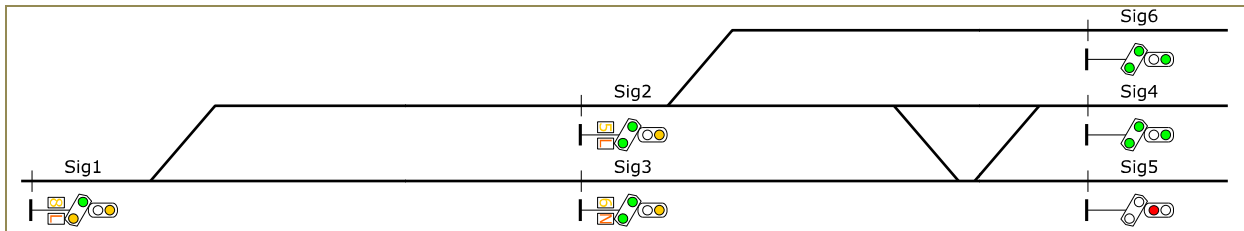
```

<implementsSignalplan id="sipaIL_b-c">
  <designator register="_SimpleRegister" entry="signalplan illustration variant b)"/>
  <aspectRelation expectingSpeed="60.0" passingSpeed="130.0" endSectionTime="PT30S"
    id="sip03">
    <designator register="_SimpleRegister" entry="aspects 2-4 variant b)"/>
    <masterAspect>
      <refersToSignal ref="sig4"/>
      <showsAspect ref="sig_reducproceed_21"/>
    </masterAspect>
    <slaveAspect>
      <refersToSignal ref="sig2"/>
      <showsAspect ref="sig_fullproceed_22"/>
    </slaveAspect>
    <distantAspect>
      <refersToSignal ref="dsig4"/>
      <showsAspect ref="sig_warning_25"/>
    </distantAspect>
    <appliesToRoute ref="rt_sig02_sig04"/>
  </aspectRelation>
  <aspectRelation expectingSpeed="0.0" passingSpeed="60.0" endSectionTime="PT30S"
    id="sip04">
    <designator register="_SimpleRegister" entry="aspects 4-6 variant a)"/>
    <masterAspect>
      <refersToSignal ref="sig6"/>
      <showsAspect ref="sig_closed_20"/>
    </masterAspect>
    <slaveAspect>
      <refersToSignal ref="sig4"/>
      <showsAspect ref="sig_reducproceed_21"/>
    </slaveAspect>
    <distantAspect>
      <refersToSignal ref="dsig6"/>
      <showsAspect ref="sig_caution_23"/>
    </distantAspect>
    <appliesToRoute ref="rt_sig04_sig06"/>
  </aspectRelation>
</implementsSignalplan>

```

The following example shows the combination of signal aspects with additional speed and direction indicators. There are two relations shown in the picture below:

- Sig1 (reduced, "8", "L") → Sig2 (reduced, "5", "L") → Sig6 (proceed)
- Sig3 (reduced, "6", "N") → Sig4 (proceed)



```

<implementsSignalplan id="sipaILc">
  <designator register="_SimpleRegister" entry="signalplan sample combination"/>
  <aspectRelation expectingSpeed="50.0" passingSpeed="80.0" endSectionTime="PT30S"
    id="sip12">
    <designator register="_SimpleRegister" entry="aspects 1-2"/>
    <masterAspect>
      <refersToSignal ref="sig2"/>
      <showsAspect ref="sig_reducproceed_21"/>
      <showsAspect ref="isp50"/>
      <showsAspect ref="idirL"/>
    </masterAspect>
    <slaveAspect>
      <refersToSignal ref="sig1"/>
      <showsAspect ref="sig_reducproceed_21"/>
      <showsAspect ref="isp80"/>
      <showsAspect ref="idirL"/>
    </slaveAspect>
    <signalsSpeedProfile ref="sps01"/>
    <appliesToRoute ref="rt_sig1_sig2"/>
  </aspectRelation>
  <aspectRelation expectingSpeed="120.0" passingSpeed="50.0" endSectionTime="PT30S"
    id="sip26">
    <designator register="_SimpleRegister" entry="aspects 2-6"/>
    <masterAspect>
      <refersToSignal ref="sig6"/>
      <showsAspect ref="sig_fullproceed_22"/>
    </masterAspect>
    <slaveAspect>
      <refersToSignal ref="sig2"/>
      <showsAspect ref="sig_reducproceed_21"/>
      <showsAspect ref="isp50"/>
      <showsAspect ref="idirL"/>
    </slaveAspect>
    <signalsSpeedProfile ref="sps02"/>
    <appliesToRoute ref="rt_sig2_sig6"/>
  </aspectRelation>
  <aspectRelation expectingSpeed="160.0" passingSpeed="60.0" endSectionTime="PT30S"
    id="sip34">
    <designator register="_SimpleRegister" entry="aspects 3-4"/>
    <masterAspect>
      <refersToSignal ref="sig4"/>
      <showsAspect ref="sig_fullproceed_22"/>
    </masterAspect>
    <slaveAspect>
      <refersToSignal ref="sig3"/>
      <showsAspect ref="sig_reducproceed_21"/>
      <showsAspect ref="isp60"/>
      <showsAspect ref="idirN"/>
    </slaveAspect>
    <signalsSpeedProfile ref="sps03"/>
    <appliesToRoute ref="rt_sig3_sig4"/>
  </aspectRelation>
</implementsSignalplan>

*****
  <hasAspect id="sig_reducproceed_21" genericAspect="LimitedProceed" >

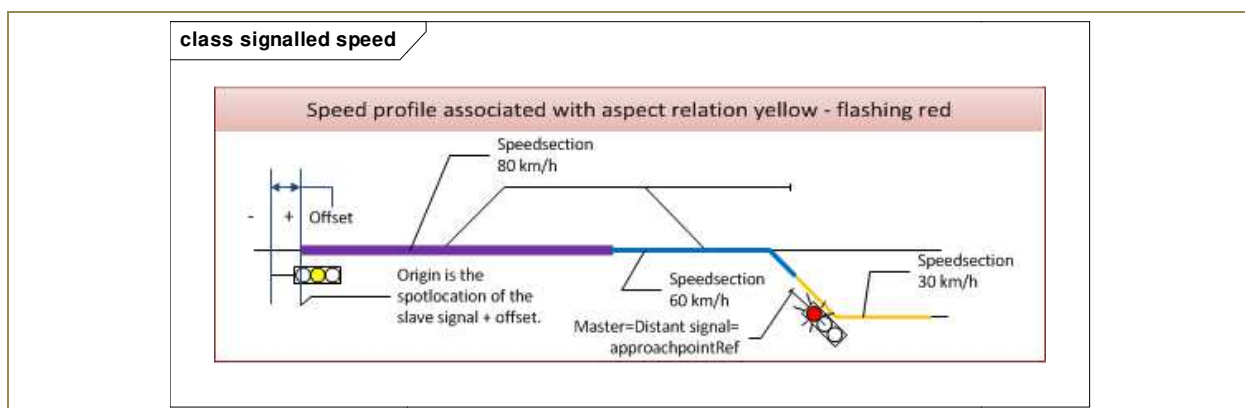
```

```

<designator register="_SimpleRegister"
  entry="Signal 21 «Kjør med redusert hastighet»"/>
</hasAspect>
<hasAspect id="isp80" genericAspect="Informative" >
  <designator register="_SimpleRegister" entry="speed80"/>
</hasAspect>
<hasAspect id="isp60" genericAspect="Informative" >
  <designator register="_SimpleRegister" entry="speed60"/>
</hasAspect>
<hasAspect id="isp50" genericAspect="Informative" >
  <designator register="_SimpleRegister" entry="speed50"/>
</hasAspect>
<hasAspect id="idirL" genericAspect="Informative" >
  <designator register="_SimpleRegister" entry="directionL"/>
</hasAspect>
<hasAspect id="idirN" genericAspect="Informative" >
  <designator register="_SimpleRegister" entry="directionN"/>
</hasAspect>

```

There are cases when the signalled speed is not applicable for the entire length of the route. Typical situation is that there are diverging switches on one part of the way like in exit routes. Depending on the signalling system this additional information can be communicated to the train and its driver. Then the interlocking shall be aware of the situation in order to trigger the information transfer.



The extract shows the aspect relations for both exit routes from station Arnau (ARN) on to the open line towards main signal 69A. According the speed restrictions on switch 68W02 the signalled speed for the diverging route is 60 km/h. For both start signals the speed to expect is set to 0 km/h that is related to the slave aspect (expect to stop) of distant signal 69Va. These values are included here merely for illustration. From the configuration in the simple example the destination signal 69A has a stand-alone distant signal 69Va. Thus no relation to master aspect is really shown at either start signal 68N1 and 68N2. The use of `<masterAspect>` and `@expectingSpeed` is limited to cases of real aspect relations between destination and start signal.

```

<implementsSignalplan id="sipaAC">
  <designator register="_SimpleRegister" entry="signalplan ARN-CST"/>
  <aspectRelation expectingSpeed="0.0" passingSpeed="60.0" endSectionTime="PT30S"
    id="sip01">
    <designator register="_SimpleRegister" entry="aspects 68N1-69A"/>
    <masterAspect>
      <refersToSignal ref="ls_sig04"/>
    </masterAspect>
  </aspectRelation>
</implementsSignalplan>

```

```

        <showsAspect ref="sig_caution_23"/>
    </masterAspect>
    <slaveAspect>
        <refersToSignal ref="mb_sig02"/>
        <showsAspect ref="sig_reducproceed_21"/>
    </slaveAspect>
    <signalsSpeedProfile ref="sps01"/>
    <appliesToRoute ref="rt_sig02_sig04"/>
</aspectRelation>
<aspectRelation expectingSpeed="0.0" passingSpeed="80.0" endSectionTime="PT30S"
    id="sip02">
    <designator register="_SimpleRegister" entry="aspects 68N2-69A"/>
    <masterAspect>
        <refersToSignal ref="ls_sig04"/>
        <showsAspect ref="sig_caution_23"/>
    </masterAspect>
    <slaveAspect>
        <refersToSignal ref="mb_sig01"/>
        <showsAspect ref="sig_fullproceed_22"/>
    </slaveAspect>
    <signalsSpeedProfile ref="sps01"/>
    <appliesToRoute ref="rt_sig01_sig04"/>
</aspectRelation>
</implementsSignalplan>

```

### 5.3 Configuration

The properties of an interlocking itself can be described by some general attributes of its configuration. This information is used for specifying particular characteristics of the interlocking system.

- **@model** – This contains the vendor-specific model or make of the interlocking system, e.g. Simis-C, VPI, NSI-63, Thales-L90, Smartlock, Westrace MkII, Westlock, SSI.
- **@technologyType** – It gives the basic type of technology the interlocking is using. It can be one of the following:
  - **"manual"** – The operation of field elements is done by signalman directly outside near the element itself. There may be some lock dependencies for basic safety principles.
  - **"mechanical"** – The operation of field elements is done by mechanical transmission of power from a centralised signalbox to the outside locations. Mechanical locking beds ensure the safety principles to avoid conflicting operation.
  - **"electromechanical"** – The operation of field elements is done by a combination of electric and mechanical systems to transmit the power from a centralised signalbox to the outside locations. It is just a mechanical interlock where the transmission to the outside is done by electrical cables to power individual engines at the elements. The interlocking logic is mainly mechanically.
  - **"relay"** – The operation of field elements is done by electric power. The interlocking logic is realised by relay circuits.
  - **"electronic"** – The operation of field elements is done by electric power. The interlocking logic is realised by electronic devices like programmable logic circuits (PLC) or computers.

- “digital” – This is the next evolution step from electronic interlocking. The interlocking is communicating to the field elements by digital connection. The power supply may be independent of this.
- “other:...” – The technology is of another type. This is the optional extension of the list. Each entry needs to start with the string “other:” and shall have at least two letters in addition.
- @SWversion – This contains the installed software version of an electronic interlocking for information. In most cases this will be a vendor specific string.
- @signalSystem – This contains the name of the signal system the interlocking is controlling for information.

The configuration example below just shows the use of the attributes without using realistic values especially for @model and @SWversion.

```
<hasConfiguration technologyType="electronic" model="Westrace 0815" signalSystem="ETCS L2"
  SWversion="WR31.0.12" />
```

## 6 Interfaces

### 6.1 Generic I/O-Interface

An interlocking can have several interfaces of different type with field elements and neighbouring interlockings. If the connecting components are from the same supplier, their interfaces to the interlocking do not need to be defined in a special way. Third party components usually have interfaces with rather individual features. Thus it needs to be defined which information is exchanged and in which way to allow a clear description of the interface. The information may be transferred by particular telegrams, bit masks and/or relays. Independent of the physical implementation it ends up with a list of commands/messages send in either direction as well as the related initial status assumed on start-up.

Per definition “commands” is named anything send from the interlocking to the connecting component via the interface, i.e. orders to the field. On the other side “messages” is named anything received at the interface for the interlocking, i.e. notifications from the field.

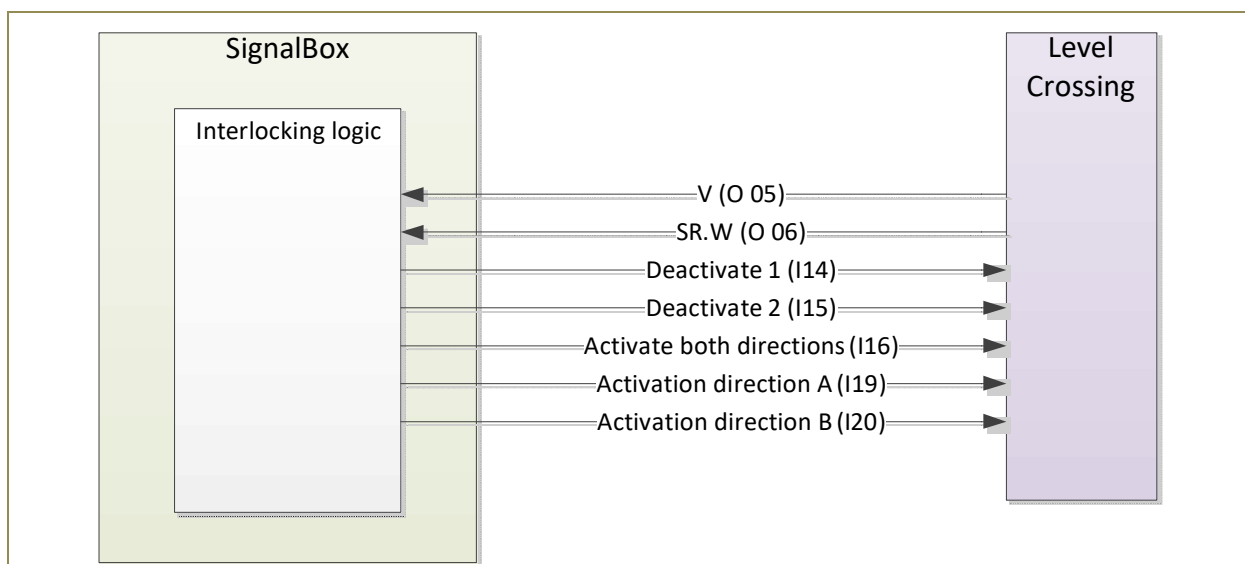
The interface is described by the following attributes and elements

- @invalidTolerationTime – This is the time an invalid status of commands or messages is tolerated before a failure action is performed.
- @switchoverTolerationTime – This is the time it will take for a command or message to switch over before a stable status is expected.
- <command> – This contains the description of a command send from the interlocking to the field element.
  - @bitNr – This is the order number of the command in the list. In case of transmission as complete bitmask it defines the bit position in the byte.
  - @description – This is just the description/meaning of the command send.

- **@normalState** – This gives the normal state of the command when not energised. The values are derived from relay contacts. Thus “open” means open contact, inactive command state or false depending on the particular technology. In contrast “closed” stands for a closed contact, an active command state or true.
  - **@pulseDuration** – This is the time for the related command being in the reverse state if no permanent switchover is done.
- **<message>** – This contains the description of a message received by the interlocking from the field element. The attributes are the same as for a **<command>**.
- **<initStatus>** – This is the description of the interface status in command and message direction which is assumed in start-up cases, i.e. when both sides of the system are just powered up.
  - **@comString** – The states of the single commands are represented within one bitmask string. Their order is related to the **@bitNr** defined in the command description. The allowed values within the string are “0” for inactive command, “1” for active command and “x” for command does not matter.
  - **@messString** – This is the representation of the single messages status. The use is the same as for the **@comString**.

## 6.2 Sample Interface for NOR Level Crossing

A level crossing is a typical field element with a special interface to the interlocking. Here is the example of a particular level crossing type as used in Norway. The first picture show the signal flow in an overview.



The signal flow is amended by the detailed list of inputs and outputs whereas input is seen as command to the level crossing side. This is because the interface lists are typically provided by the supplier of the level crossing.

Input	Function	Normal state	Name (Command)
-------	----------	--------------	----------------

Input	Function	Normal state	Name (Command)
I 14	Remote deactivation, ch.1	NOT active, open circuit	Deactivate 1 (C1)
I 15	Remote deactivation, ch.2	NOT active, open circuit	Deactivate 2 (C1 / C2 )
I 16	Activation, both directions	NOT active, closed circuit	Activate (C3)
I 19	Activation direction A	NOT active, closed circuit	Enable A (C4)
I 20	Activation direction B	NOT active, closed circuit	Enable B (C5)

Output	Function	Normal state	Name (Message)
O 05 (V)	LC not activated, both directions	active, closed circuit	Opened (M1)
O 06 (SR.W)	Train signals cleared	NOT active, open circuit	Protected (M2)

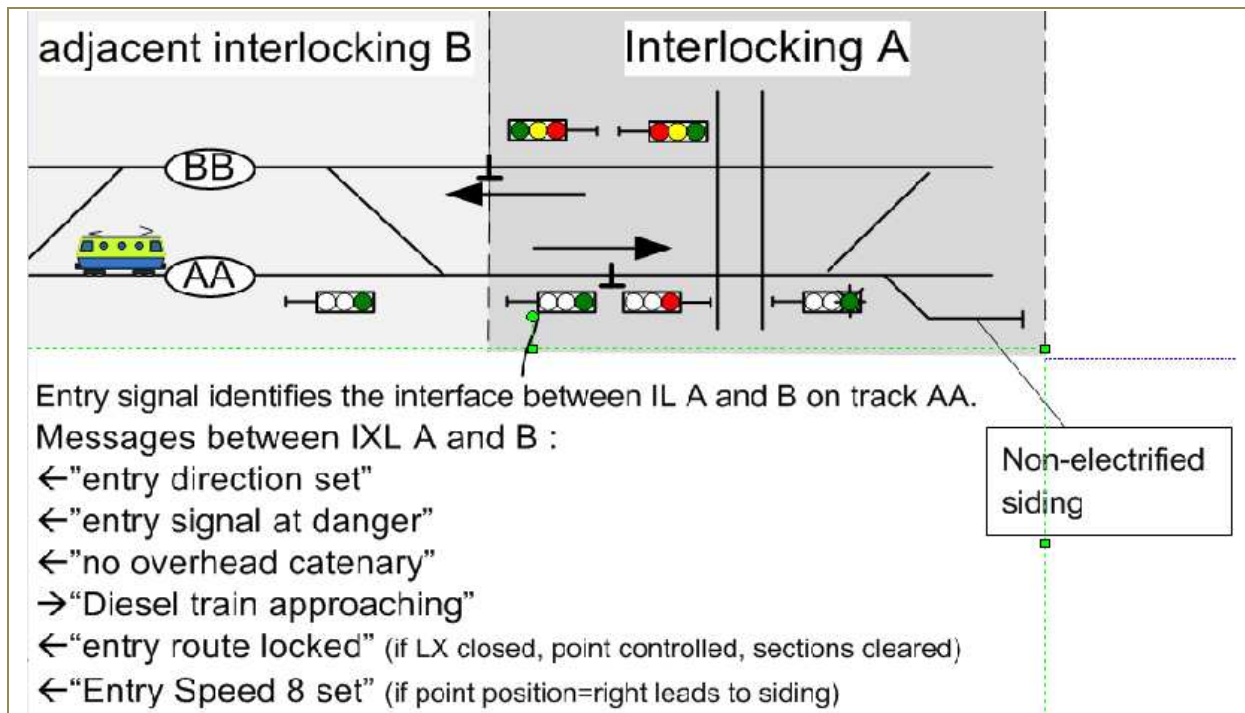
Finally the detailed list of inputs and outputs are transformed into the equivalent in railML as shown below.

```
<interface id="int01" invalidTolerationTime="PT0.5S" switchoverTolerationTime="PT0.8S">
  <designator register="_SimpleRegister" entry="Level crossing Norge"/>
  <command bitNr="1" description="Remote deactivation, ch.1" id="I14" normalState="open">
    <designator register="_SimpleRegister" entry="deactivate1"/>
  </command>
  <command bitNr="2" description="Remote deactivation, ch.2" id="I15" normalState="open">
    <designator register="_SimpleRegister" entry="deactivate2"/>
  </command>
  <command bitNr="3" description="Activation, both directions" id="I16"
    normalState="closed">
    <designator register="_SimpleRegister" entry="activate"/>
  </command>
  <command bitNr="4" description="Activation direction A" id="I19" normalState="closed">
    <designator register="_SimpleRegister" entry="enableA"/>
  </command>
  <command bitNr="5" description="Activation direction B" id="I20" normalState="closed">
    <designator register="_SimpleRegister" entry="enableB"/>
  </command>
  <message bitNr="1" description="LC not activated, both directions" id="O05"
    normalState="closed">
    <designator register="_SimpleRegister" entry="Opened"/>
  </message>
  <message bitNr="2" description="Train signals cleared" id="O06" normalState="open">
    <designator register="_SimpleRegister" entry="Protected"/>
  </message>
  <initStatus comString="00111" messString="10"/>
</interface>
```

## 6.3 Interlocking Interface

At the border of one interlocking system's authority there are interfaces to other interlocking systems for exchange of train control information. For each interlocking it is necessary to define the location of the interface and what particular information is transmitted in which direction.

Thus it is important to know from which of the both sides the interface viewed by the interlocking. Otherwise the interfaces would need to be defined twice. The elements and attributes of the physical interface shall be considered according to the view of the interlocking onto the interface, i.e. the value of `@isOnCommandSide` within the `<SignalBox>` definition.



The above figure illustrates the situation of an interlocking interface with some information transmitted between both interlocking. The elements and attributes used for description of the interface are:

- `@interfaceLocation` – This is the specification of the topological location of the interface. It determines the base type of interface instantiation for the interlocking.
  - `"inStation"` – The interface between two interlockings is located in station area and no block routes are related to this interface, i.e. two stations bordering directly to each other without open line in-between.
  - `"atStationBorder"` – The interface between two interlockings is located at the border of one station with open line on the other side.
  - `"onOpenLine"` – The interface between two interlockings is located on the open line with only block routes related to the interface.
- `@isOnCommandSide` – This is flag gives the information how to read the physical interface information.
  - `"true"` – If the flag is true then any `<command>` in the physical interface is a real command for this interlocking.
  - `"false"` – If the flag is false then any `<message>` in the physical interface is a command for this interlocking, i.e. the interface is viewed from the opposite side.
- `<lastOwnTvdSection>` – This is the reference to the last TVD section in front of the interface that the interlocking has full control of.

- `<firstRemoteTvdSection>` – This is the reference to the first TVD section beyond the interface the interfacing interlocking has full control of but may be notify its status to the interlocking.
- `<incomingRoute>` – This is the reference to all routes the interlocking knows that start at the interface location or in rear of it in direction towards the interlocking.
- `<outgoingRoute>` – This is the reference to all routes the interlocking knows that end at the interface location or in advance of it in direction towards the other interlocking.
- `<hasInterface>` – This is the reference to the description of the physical interface with commands and messages transmitted.

## 7 SystemAssets

The railway signalling has elements in addition to the signalbox and the controller, which are not directly related to a particular track. These more common elements are called “system asset”. As this can be a wide range of objects like RBC, Command and Control System, communication system, power supply, passenger information system suitable containers are to be defined in `<assetForIL>`. These classes shall at least provide `@id` and `<designator>` as for all IL elements.

Currently there is only the container `<powerSuppliesIL>` in `<assetForIL>`.

### 7.1 PowerSupplyIL

The power supply provides electrical power to the control and the field elements. It might be located at the signalbox or close to the particular field elements like signals or switches it is connected to. The detailed features like overall power or voltage levels are very specific for the manufacturer and the location it is installed. They are not included in railML.

The information of interest for the interlocking are these attributes:

- `@numberOfSimultaneousSwitchingActuators` – This is the maximum number of switch actuators that can be activated simultaneously with this power supply. In case of more switch actuators connected to this power supply are commanded for switching the interlocking will stagger their operation in order to avoid overload of the power supply.
- `@signalVoltageMode` – This gives the mode of switching signal voltage for day and night voltage. The values can be:
  - `“automatic”` – The switching of signal voltage is done automatically according input from daylight detector.
  - `“manual”` – The switching of signal voltage is done by manual operator command.
  - `“nightOnly”` – The signal voltage is always kept on night voltage level as necessary for tunnel signals.
  - `“none”` – The system does not know about any daylight dependent voltage adjustments.

The example below shows the instance of a power supply capable to activate four switch actuators at once and automatically changing the signal voltage between day and night value.

```
<powerSupplyIL id="ups01" numberOfSimultaneousSwitchingActuators="2"
  signalVoltageMode="automatic">
  <designator register="_SimpleRegister" entry="UPS-ARN01"/>
</powerSupplyIL>
```

## 8 Controllers

A **<Controller>** is the element in a signalling control system that is used for operation of the railway system. It can be an individual terminal, commonly a workstation, which can control the interlocking. It can be also an automatic dispatch system commanding the interlocking for automatic train routing. The **<Controller>** is normally situated in a control centre. railML provides a logical link between an interlocking and the individual controller. The user can attach useful data to this link, such as addresses that may be granted control over this interlocking.

railML will not define the nature of the addresses, i.e IP-addresses or hexadecimal address of terminals that communicate with the interlocking via some serial bus. The protocol (IP, UDP, serial, parallel) is irrelevant to railML. Note that a Control Centre (DE: Leitstelle, FR: Poste de controle, NL: VL-post) is likely to control multiple interlockings and vice versa, one interlocking can be controlled from multiple control centres, an n:m relation.

This implies that a control centre can have multiple controllers, defined as a terminal from which a signalman controls an interlocking. The interlocking is unaware of the control centre but aware of the controller.

The associations for a controller are defined with these elements:

- **<controlledAssets>** – This is the container with the lists of all assets, which can be controlled from this **<Controller>**. There are the two possible types of child element:
  - **<controlledInterlocking>** – This is the reference to any **<SignalBox>** (interlocking), which can be controlled from this **<Controller>**. Additionally the extent of control according chapter 5.1 is to be given.
  - **<controlledSystemAsset>** – This is the reference to any **<SystemAsset>**, which can be controlled from this **<Controller>**. This will especially cover any system assets, which are only controlled from here but not from any interlocking. Additionally the extent of control according chapter 5.1 is to be given.
- **<itineraries>** – This is the container for all route combinations as **<intinerary>** this **<Controller>** has available to dispatch any train on its way from starting OCP to the destination OCP. Per route combination the start and destination point is referred and each single route forming the continuous path are referred. The principle is similar to combined routes as described in chapter 4.11. Within such **<intinerary>** there are no alternative sections defined.

## 9 References

- [1] railML.org Wiki: *Use case Schematic Track Plan*. In:  
[https://wiki.railml.org/index.php?title=UC:IS:Schematic\\_Track\\_Plan](https://wiki.railml.org/index.php?title=UC:IS:Schematic_Track_Plan); last access:  
10.11.2018
- [2] railML.org Wiki: *Use case Interlocking Module Engineering Data*. In:  
<https://wiki.railml.org/index.php?title=UC:IL:InterlockingEngineering>; last access:  
10.11.2018
- [3] railML.org Wiki: *Use case Routes for timetable simulation*. In:  
<https://wiki.railml.org/index.php?title=UC:IL:Simulation>; last access: 10.11.2018
- [4] Norwegian Operational Railway Rules (*Forskrift om togframføring på det nasjonale jernbanenettet*). In: <https://lovdata.no/dokument/SF/forskrift/2008-02-29-240>; last  
access 10.11.2018
- [5] ERA: European Rail Traffic Management System (ERTMS). In:  
[https://www.era.europa.eu/activities/european-rail-traffic-management-system-ertms\\_en](https://www.era.europa.eu/activities/european-rail-traffic-management-system-ertms_en); last access 10.11.2018
- [6] IEC 60050-821: *International Electrotechnical Vocabulary Part 821: Signalling and security apparatus for railways*. April 1998
- [7] DCC Wiki: *Crossing*. In: <https://dccwiki.com/Crossing>; last access 25.11.2018
- [8] EULYNX Initiative: *EULYNX Domain Knowledge*; In: Eu.Doc.10, EULYNX Baseline Set: 2;  
<https://www.eulynx.eu/>