Subject: Re: Steckenunterbruch/line blocking
Posted by on Fri, 19 Jul 2013 17:10:34 GMT
View Forum Message <> Reply to Message

Dear Susanne,

> I just implemented the new element <state> with an attribute 'disabled'
> of type xs:boolean. It may be constrained with the attribute
> 'operatingPeriodRef' referring to an operatingPeriod/@id from the
> timetable subschema. Furthermore it may be constrained to a part of a
> <track> by relative positions.

Thank you, looks good.

> There are some assumptions, that should be documented in the wiki

This is very important, thank you very much.

>   * If no "from" is defined, it begins at the "trackBegin".
>   * If no "to" is defined, it begins at the "trackEnd".

+1

>   * If no "operatingPeriodRef" is defined, it is valid for all data of
>   the railML file.

+1

>   * For all other times (out of the referred operating period) the
>   defined track (section) is enabled/disabled depending on the
>   "disabled" value.
>   * If no "state" element is defined, the "track" is usable. That means
>   'disabled="false".

We both mean the same, but to avoid misunderstandings, I would prefer a
more clear wording:

If there is no <state> element saying anything else, a track can be
assumed to be enabled. Therefore, there is no need to create a <state>
element with disabled="false".

Now, either you could leave it as you have designed it, or you could
discard the attribute 'disabled' and name the element <state> to
<disabled> or such.

With the attribute disabled="false", there would be several possibilities
to express the same sequence of blockings:

```
    <state disabled="true" operatingPeriodRef="op_May">
    <state disabled="true" operatingPeriodRef="op_July">
or
    <state disabled="true" operatingPeriodRef="op_May-July">
    <state disabled="enabled" operatingPeriodRef="op_June">
```

may both be treated as the same. With the second solution, there may be again the problem of sequence: The different order of the two rows may result in a difference meaning. Therefore, we possibly would need a 'sequence' or 'priority' attribute or such again.

To avoid this redundancy, and to avoid the 'sequence' attribute, I would prefer to discard the attribute 'disabled' and to rename the element <state> into <disabled> or such.

> Currently the <from> and <to> elements may additionally refer to an
> 'ocp' via an 'ocpRef' attribute. Maybe that should be dropped because of
> redundancy reasons.
>
>   But otherwise that may be helpful if the exact blocking locations
>   (relative positions) are known but differ from the ocp locations. The
>   ocp references may be used as a hint, not overwriting the exact
>   locations.

I think this is not redundancy:

To refer to ocp's says: The writing software does not know the exact positions. Any movement between theses ocp's is not allowed; anyway where exactly the reason for that is situated.

To refer to exact positions says: The writing software knows the exact positions. The track between theses positions is not available.

The difference lies in the question "how much of the stations at the beginning/end of the blocking is usable": An <ocp> element normally is situated at the mileage position of the _middle_ of the corresponding station (middle of the platforms). Can you use the "half" of the station from the middle of the platforms into the direction of the blocking?

Let's assume ocp 'ABC' lies at relative position km 15.432.

```
    <state disabled="true">
      <from pos="15432"/>
      <to pos="23456"/>
    </state>
```

means: You cannot move even one meter behind km 15.432 (the middle of the station), so you cannot enter this station from below 15.432 at the normal kind. (Even half of the platform seams to be closed. There is a "Schutzhalttafel" exactly at the middle of the station.)

```
<state disabled="true">
  <from ocpRef='ABC'/>
  <to ocpRef='DEF'/>
</state>
```

means: The track section between ABC and DEF is blocked. There is here no information on a blocking inside the station of ABC. You can assume it is normally usable.

I would prefer to keep this difference in any kind: In some cases, we do need the possibility to name the exact positions of the "Schutzhalttafeln" either for reasons of the "building trade" (BETRA) or to allow shunting movements. In many other cases (e. g. timetabling), we do not know the exact positions but we have to describe that a section of track is closed.

Of course there would be other, more explicit solutions to express this difference. May be "more explicit" would be better. Anyway, I am satisfied if it is possible at least in any kind.

> The 'line blocking' has to be defined through the 'track blocking'.

+1

>> Such a reference would - as far as I know - the first time we would
>> create such a "forward-reference". Forward in the meaning of "from
>> infrastructure to timetable". We already have many references from
>> timetable to infrastructure which are "natural" since one _first_
>> needs infrastructure _before_ one can operate a train.
>
> We already put such a "forward-reference" into the infrastructure
> subschema with the implementation of speed profiles.
>
> The TSR (temporary speed restrictions, de:Langsamfahrstellen) also
> refer to an 'operatingPeriod' from the timetable subschema.

Ok, good to have this clarified as a general question: Forward references are not forbidden.

> For a future version we would like to move the operating periods to the
> common part of railML, in order to provide a better "straight-forward"
> structure. But this is a change for the next major release.

But why? Some sentences before, you did clarify: Forward references are not forbidden, sequential reading is not possible. So anybody must accept and implement this to come from now to the next major release. Why should we change it afterwards?

I would welcome this to avoid forward references just to make things
easier and to get a higher acceptance of RailML. But more importantly, I
would prefer consequence and not to switch between the philosophies. A
stable philosophy creates much more acceptance than one which is only
theoretically better.

So if you do not want (or cannot) avoid forward references now and in
general, you do not need to avoid them later.

Best regards,
Dirk.