Subject: Schema version 1.00 RC1 released
Posted by Ulrich Linder on Wed, 22 Sep 2004 09:07:21 GMT
View Forum Message <> Reply to Message

Hello,

the first release candidate of V1.00 is released:

http://www.railml.org/genesis/infrastructure

The handling of switches and crossings is improved. The orientation and the course of a connection is moved from the switch/crossing the the relevant "connection"-child. Look at the discussion thread of V0.95-02 for more informations about other (minor) changes.

With best regards

Ulrich Linder

Dr.-Ing. Ulrich Linder
Linder Rail Consult

D-14169 Berlin

Tel. +49.30.84 72 56 87
Fax. +49.30.84 47 11 56

Email mailto:Ulrich.Linder@linder-rail-consult.de
www http://www.Linder-rail-consult.de

Subject: V1.00 RC1: switchRef/crossingRef
Posted by Matthias Hengartner on Tue, 28 Sep 2004 13:01:08 GMT
View Forum Message <> Reply to Message

Hello,

I'm fine with the changes and answers in the discussion thread of V.095-02.

There are some little remarks and questions left about V1.0 RC1.

One of them is about switches/crossings on trackBegin/trackEnd (as mentioned in the discussion thread v.095.02).

In Berlin, we forgot to discuss about the implementation of switches/crossings which are placed on a <trackBegin>/<trackEnd>. In the last thread of this newsgroup, I suggested the following:

"I'd prefer to have only a reference to a switch/crossing which is located
in the <connections>-container."

Concretely, I have the following outlined example of a very simple possible
implementation of this idea:

```
-----------------------------
<track trackID="track1" ...>
    <trackTopology>
        <trackBegin>
            ...
        </trackBegin>
        <trackEnd>
            <switchRef elemIDRef="SW01"/>
        </trackEnd>
        <connections>
            <switch elemID="SW01" ...>
                <connection connectionID="connection1A"...(to track2,
connection2)/>
                <connection connectionID="connection1B"...(to track3,
connection3)/>
            </switch>
        </connections>
    <trackTopology>
</track>
<track trackID="track2" ...>
    <trackTopology>
        <trackBegin>
            <simpleConnection ...>
                <connection connectionID="connection2" ... (to track1,
connection1A)/>
            </simpleConnection>
        </trackBegin>
        <trackEnd>
            ...
        </trackEnd>
</track>
<track trackID="track3" ...>
    <trackTopology>
        <trackBegin>
            <simpleConnection ...>
                <connection connectionID="connection3" ... (to track1,
connection1B)/>
            </simpleConnection>
        </trackBegin>
        <trackEnd>
            ...
        </trackEnd>
```

</track>

Explanations:

- elemIDRef is required and must refer to a switch within the SAME <track>!!
For tracks which begin/end connected to a switch of another track, we have
the <simpleConnection>

- in the example above (and in fact in case in which we use this construct),
we have 3 tracks which are connected in one switch. The switch is defined in
exactly ONE of these tracks (in our example in track1), the other 2 tracks
are connected via simpleConnections.

```
                    / track3
                   /
                  /
track1 --------/--------------track2
```

- analogously, there would be a crossingRef as child of
<trackEnd>/<trackBegin>, which would refer to a <crossing>. analogous to the
previous explanation point, there are normally 4 tracks which are connected
in one crossing, the crossing is defined in exactly ONE of these tracks, and
the other 3 tracks are connected via simpleConnections.

This is a possible modelling. Please tell me your opinion about this.
Especially, I'm not quite sure if it's ok that we have only the attribute
"elemIDRef" in <switchRef>/<crossingRef>. Perhaps, we should include some
attributes like "pos" ore "elemID" there, too. What do you think?

Thanks for your answers and best regards

Matthias Hengartner

Matthias Hengartner

++ 41 1 633 31 09
hengartner@ivt.baug.ethz.ch

"Ulrich Linder" <ULinder@Railways.TU-Berlin.de> wrote in message
news:cirfid$b0k$1@sifa.ivi.fhg.de...
> Hello,
>
> the first release candidate of V1.00 is released:
>
> http://www.railml.org/genesis/infrastructure
>
> The handling of switches and crossings is improved. The orientation and
the
> course of a connection is moved from the switch/crossing the the relevant
> "connection"-child. Look at the discussion thread of V0.95-02 for more
> informations about other (minor) changes.
>
> With best regards
>
> Ulrich Linder
>
> ------------------------------
> Dr.-Ing. Ulrich Linder
> Linder Rail Consult

> D-14169 Berlin
>
> Tel. +49.30.84 72 56 87
> Fax. +49.30.84 47 11 56
>
> Email   mailto:Ulrich.Linder@linder-rail-consult.de
> www   http://www.Linder-rail-consult.de
>
>
>
>

Subject: Re: Schema version 1.00 RC1 released
Posted by Matthias Hengartner on Tue, 28 Sep 2004 14:02:04 GMT
View Forum Message <> Reply to Message

Hello again,

here the rest of my remarks about V1.00 RC1 for the present:

- In the <connection>-element, we have the attribute "branchTopologyIDRef".
We should rename it in "branchInfrastructureIDRef", since we refer to
another <infrastructure> with an "infrastructureID". (It think this

inconsistency bases on a former proposal of mine).

(Creating Instructions for railML partial
schemes) from Nils Poldrack
 (www.railml.org/documents/general/creatinginstructions_ger_R C1.pdf), section

is(among other) the following rule:
* Abbreviations or contractions may not be used as part of designator names
---> this rule collides with different designators in the infrastructure
schema (e.g. infraAttrGroup, ocp***, dir, posOnNet, absPos, ...)
It's not my opinion that we have to rename these designators, but perhaps we
should change the wording of this rule ("should not" instead of "may not").

- And finally, there are some missprints in the demofile (for those who
already use it):
1. branchTrackIDRef "B-SW13-SW14" should be "C-SW13-SW14"
2. branchTrackIDRef "B-SW17-SW18" should be "C-SW17-SW18"
3. crossection of track "A-4" at km 50.313 should have ocpIDRef "A" haben
(not "D")
4. branchIDRef "SW26L" (in track "A-4") should be "SW29L"
5. and EndTrackD-4 is an "empty" <simpleConnection>. By intention?

(Ulli, I'll send you the demofile with points 1. to 4. corrected, so you
don't have to do the work that I've already done.)

Have a nice evening

Matthias Hengartner

Matthias Hengartner

++ 41 1 633 31 09
hengartner@ivt.baug.ethz.ch

Subject: Re: V1.00 RC1: switchRef/crossingRef
Posted by Volker Knollmann on Wed, 29 Sep 2004 09:54:52 GMT
View Forum Message <> Reply to Message

On 28.09.2004 15:01, Matthias Hengartner wrote:

> In Berlin, we forgot to discuss about the implementation of
> switches/crossings which are placed on a <trackBegin>/<trackEnd>. In the
> last thread of this newsgroup, I suggested the following:
> "I'd prefer to have only a reference to a switch/crossing which is located
> in the <connections>-container."

Jepp, that's the way I would prefer it, too. But as usual, things are
not as easy as in your example, although it was perfect to understand
your intention. So I will do my very best to make things complicated ;-)

If we start or end a track with a switch, we can distinguish between 2
cases:

(1) the switch element belongs to the straight track

```
               / first connected track
              /
             o

           o
          /
         /
-------o---------o   o------ second connected track
```

(2) the switch element belongs to the branch track

```
                    /
                   /
                  o
                 /
                /
first -----o   o---------o   o------ second connected track
```

The crucial thing is the required "orientation"-attribute in the
<connection>-element of a switch. "orientation" can be either
"incoming", "outgoing", "right angled" (???) or "unknown". Which value
is to be chosen for the second track in case (1) and for both tracks in
case (2)?

I suggest an additional value "straight" (which perfectly coincides with
the possible values for "trackContinueCourse") and the __convention__ to
let the <switch>-element be part of track at the switch's tip. Thus,
role of every track is unambiguous.

The second point I want to make about switches is the
"length"-attribute. If a switch is connected at the start or end of a
track and the switch's "length"-attribute is set, where does the track
effictively end or start? At the connection point of the switch or at
the connection point + "length"?

And which length does "length" describe? I would interprete it as follows:

```
                /
     _          o
     |         o
     c         /
     |        /
     -       /
-------o----------------o   o-------------

      |- a --|--- b --|
```

(a) is the "length"-attribute of <switch>

(b) is the "branchDist"-attribute of the <connection>-element to the
straight track

(c) dito for the branching track

But this still doesn't solve the problem where the <track> ends. From my
point of view this is important to generate proper information for the
vacancy detection of elements.

And by the way: "branchDist" is to be defined in [m], not in [km].
Should this be changed to [km] to harmonize all distances and positions
in the file?

So much for the moment... maybe I'll terrorize you with more posts as I
work my way through 1.0RC1 and try to adapt our internal file format
specs...

Thanks in advance for your comments!

Best regards from Braunschweig,
Volker Knollmann

Subject: Re: Schema version 1.00 RC1 released
Posted by Volker Knollmann on Wed, 29 Sep 2004 16:10:46 GMT

View Forum Message <> Reply to Message

Hi again,

I have a few remarks regarding the concept of the "generalElements".
First of all: I really appreciate the idea of that element.

As I explained in Berlin, we have to store various kinds of data and the
scope of that data is (currently) beyond the scope of railML. It is a
good concept to have a kind of "allround-element" / "dustbin" /
"whatever-you-like-to-call-it" to store non-railML data while still
being conformitive with the standard.

I would suggest to extend the concept of "generalElements" to other
branches of <track>. I can imagine a <generalOCSElements>-container
(child of <ocsElements>) and a <generalTrackData>-container (child of
<track>).

I give you an example:
I need to store the information, from which station / interlocking
(ocpID) a switch is controlled. One method would be to introduce an
additional attribute for <switch> (e. g. ocpRefID). Doing so, my
xml-file could never again be validated against the official scheme.

The second method would be to have an item called <generalTrackData>,
which stores arbitrary data. I would introduce a child
<switchOcpMappings> and store my information in some childs like
<switchOcpMapping switchRefID="42" ocpID="88"/>. No application except
our laboratory would care about that, the file remains valid,
everybody's happy.

Of course I see the danger that everybody stores information in the
general container instead of the "real" railML-tree. But we should try;
and maybe we can move some data structures which proove to be well
designed and accepted to the official tree in a later release.

We talked about that topic shortly in Berlin and there seemed to be some acceptance. Since my suggestion is only a minor change to the schema (but with major improvements), we can get an agreement on that topic before the final freeze of V1.0.

Looking forward to your comments,
Volker Knollmann

Subject: Train Detection and Train Protection
Posted by Gregor.Theeg on Fri, 01 Oct 2004 10:16:24 GMT
View Forum Message <> Reply to Message

Dear railML partners,

the basic difference between detectors and train protection elements is the following:
Detection elements detect a wheel/axle, vehicle or obstacle and give information from the train to the infrastructure (signal box). The signal box uses this information for proving a track free (axle counters, track circuits) or for switching purposes (e.g. switching a level crossing on/of, release or approach locking of routes etc.
Train protection elements (such as balises, trackside magnets or mechanical barriers (which are still in use at Berlin S-Bahn) are actors; they give information from the infrastructure to the train, e.g. order to brake immediately.
Based on this difference, we should divide the "small things along the track" into 2 containers: "detectionElements" and "trainProtectionElements".

"detectionElements" can contain 0...many elements "detector" and "trackCircuitChange" and maybe more others in later versions.

Althought in most (not in all) cases there are different forms of detectors used for axle counting and for activating a level crossing and they must meet different requests, principally they are all the same thing, which means that in line schema they shall be the same type of element with different attributes. Only the signal box makes the one an axle counter and the other a switching-device, sometimes the information from one detector is used for several purposes, so the detector is both.
I suggest the following (mostly not-required) attributed:
- The ID and positioning attributes we already have.
- The kind of detected object, e.g. wheel, vehicle (=railway vehicle), obstacle (including road vehicles), end of train. Axle counters must detect wheels, for switching on a level crossing sometimes induction loops (which are vehicle detectors) are used.
-The technical principle, e.g. mechanical, hydraulic, pneumatic, magnetic, inductive and optical. Most modern forms are inductive, but optical forms

are increasingly used, especially for obstacle detection, and at Innotrans 2004 I even saw a modern mechanical detector!
- Position: Wheel detectors can be at right or left rail, vehicle and obstacle detectors between or ...meters right or left from the rails.
- Detection of direction ("true"/"false"): Today detectors used for axle counting always have to distinguish directions.
- Type (Bauform; according to the producer)
- Information, if this detector is used for axle counting ("true"/"false"). Sometimes in layout maps we want to draw all "axle counters".

Track circuits are detectors that are no point, but have a length. In our definition we should clearly distinguish between the track circuit and the limit of a track circuits (e.g. insulated rail joint). The attributes "length" and "frequency" refer to the track circuit, not to its limit.
Because of the functional analogy between an axle counting point and an insulated rail joint (limiting track sections to be proven free) I would put the trackCircuitChange into the container "detectionElements", although they are no detector. Because of the same analogy between track circuits (a physical element which actually belongs into line schema) and axle counting circuits (a logical thing which actually belongs into interlocking schema) I like to have both of them at the same position in railML DTD. Thus, we should leave the track circuit (not the trackCircuitChange!) out for the moment (which means in version 1.0) and think about this problem later during the development of interlocking schema.
trackCircuitChanges I would like to give additional attributes:
- If the right, left, both or no rail is insulated. No rail for insulated-rail-joint-free track circuits.
- If at upper, lower or both sides there is a track circuit.

"trainProtectionElements" in most cases are a "tracksideMagnet" or a "balise". A trackside magnet can also be a combination of 2 magnets, e.g. Signum. The basic difference is that a trackside magnet submits only its condition (1 bit), e.g. "I'm under alternate current 1000 Hz" or "I'm under direct current with polarity ...", whereas a balise submits a data telegram. Additional attributes should be the system, e.g. "PZB90", "Signum", "ZUB123", "ETCS Level1", "Crocodile",...; and the type of the element (Bauform).

Subject: signals etc.
Posted by Gregor.Theeg on Fri, 01 Oct 2004 15:13:04 GMT
View Forum Message <> Reply to Message

Dear Ulrich Linder,

Referring to signals, I suggest the following changes:

They should get following additional attributes:
- "trackDist", which means the distance from the track they belong to in [m]. + is right, - left from the track.
- "height", which means height of the signal (= its red light) above top of rail

The attribute "maskable" we should divide into 2 different attributes: maskableRoute and maskableATC.
Background: There are 3 different types of "black" signals in German terminology:
- Extinct (erloschen) is a signal which shows no aspect due to damage. A driver who sees this signal has to assume the aspect that requires the highest caution, that means for main signals to stop immediately. This should not be of our special interest.
- Switched off (abgeschalten) is a signal which is not needed for the route set at the moment, but stands at this route. Usually we have this case at intermediate signals in stations. To distinguish them from the first mentioned (which would force the driver to stop), in Germany they are marked with a small white light (Kennlicht).
- Switched dark (dunkel geschalten): When a line has automatic train control with cab signalling (e.g. LZB or ETCS level2), the cab signal has higher priority than the signals on the line. Driver has to obey only the cab signals, whichever colour the line signals show. Because we don't want to confuse the driver, these signals can be switched dark.

Because signals already have a lot of attributes and will have much more later, we should sort them into 4 container elements:
element "identification" with attributes elemID, name, absPos, absPosOffset, switchable, virtual, signalBoxID, stationID
element "position" with attributes pos, dir, trackDist and the sub-element geoCoord
element "physical" with attributes sight, height and many more attributes in later versions
elements "signalAspects" with attributes type, function, sigSystem, maskableRoute, maskableATC and later the definition of aspects

If signals are automatically controlled (e.g. automatic block), this is a problem of interlocking (or better: opertion control), thus we should leave it out here and spare it for interlocking schema. The same refers to the danger point where, depending on route and overlap, can be several danger points behind the same signal.

The signal aspects are a very complex topic if we want to make a schema which covers all European systems. Although the idea "vDirect, vDistant" covers Central and Eastern Europe to a large extent, for the rest of the continent it is absolutely unsufficient. Some countries have junction or direction signalling instead of our speed signalling. Often 3 (not only 2) block sections are signalled at the same signal and some countries even give information on the whole way through a station already at the entrance distant signal (such as Spain or Belgium), others (like Russia and China) have aspects like "free until next station" even if there are several block signals between. In Sweden a speed restriction 40 km/h implies stop at the next signal. More information you can find in my presentation from 6th railML meeting. I think we should leave it out for the moment, it will be some of the hardest work when thinking about interlocking. Or does somebody urgently need it?

At speed changes, we shall leave out the attribute "restricted speed". This is a problem of diverging routes in interlocking and we should define it there. At the same place, there can be very different diverging speeds, depending on the route.

The blocks also belong into the interlocking schema.

In interlocking a switch crossing (Kreuzungsweiche) is handled as 2 single switches. To refer to them, each of these 2 parts needs an own ID. I suggest to add 2 additional attributes besides the ID of the switch: ID1 and ID2, where ID1 is the ID of the lower and ID2 the ID of the higher part in direction of internal positioning of the track where the switch is defined. When element ID is 3, for example, ID1 would be 3a und ID2 = 3b.

Best regards,
Gregor Theeg

Subject: Re: V1.00 RC1: switchRef/crossingRef
Posted by Matthias Hengartner on Mon, 04 Oct 2004 12:57:30 GMT
View Forum Message <> Reply to Message

> Jepp, that's the way I would prefer it, too. But as usual, things are
> not as easy as in your example, although it was perfect to understand
> your intention. So I will do my very best to make things complicated ;-)

Yes, you're right, my example was quite a "model example".

>
> If we start or end a track with a switch, we can distinguish between 2
> cases:

BTW: These 2 cases can have 2 sub-cases: The switch can be placed on trackEnd (a) or on trackBegin (b), and the "orientation"-attribute refers to the direction of the track (which is defined by trackBegin and trackEnd). See below in the ASCII-drawing.

```
>
>
>  (1) the switch element belongs to the straight track
>
>
>                  / first connected track
>                 /
>                o
>
>              o
>             /
>            /
>  -------o---------o   o------ second connected track
(1a) ---> (trackEnd)
(1b) <--- (trackBegin)

>
>
>
>  (2) the switch element belongs to the branch track
>
>
>                       /
>                      /    /             ^
>                     o    /             /
>                    /    V  (2a)       /  (2b)
>                   /  (trackEnd) (trackBegin)
>  first -----o   o---------o   o------ second connected track
>
```

> The crucial thing is the required "orientation"-attribute in the
> <connection>-element of a switch. "orientation" can be either
> "incoming", "outgoing", "right angled" (???) or "unknown". Which value
> is to be chosen for the second track in case (1) and for both tracks in
> case (2)?
>
> I suggest an additional value "straight" (which perfectly coincides with
> the possible values for "trackContinueCourse")

In case (1a), I'd take "outgoing" for _both_ connected tracks (1b: "incoming"). The attribute "course" would have the value "straight" for the second connected track (and "left"/"right" for the first [1a/1b]).
So in my opinion, we _could_ introduce the value "straight" in the "orientation"-attribute, but there's no need for it.

> and the __convention__ to
> let the <switch>-element be part of track at the switch's tip. Thus,
> role of every track is unambiguous.

I agree fully with you!
If I _had to_ realize case (2) in railML, I probably would say that the
first connected track is "outgoing" (2a) / "incoming" (2b), and the second
connected track is "incoming" (2a) / "outgoing" (2b).
But this is of course a very "dirty" implementation. And I don't think that
the possibility to implement case (2) is really needed (It can easily be
avoided).

*****
Another possibility... We could abandon the special treatment of
switches/crossings which are placed on trackBegin/trackEnd (I feel a little
uncomfortable about saying this, because this idea is penned by me...).
However, then we'd have a <simpleConnection> and a <switch> which have the
same position (on trackBegin/trackEnd). So, in case (1) we'd have a
<simpleConnection> to one of the connected track and a <switch>/<connection>
to the other. In case (2), we'd have the <simpleConnection> to the first and
a <switch>/<connection> to the second connected track. It would be clear,
which "orientation" a track has, as a simpleConnection is always "straight".
Disadvantages of this solutions are:
- we have data redundancy (but not very much)
- we have to compare the position of the
switches/crossings/simpleConnections to get the information, that a
switch/crossing is placed on a trackEnd/trackBegin

Or, final idea (for the moment ;-)): We could combine these 2 approaches:
We could have a <simpleConnection> with a reference to a
<switch>/<crossing>.

What do you think?

Best regards from sunny Zurich
Matthias Hengartner

Matthias Hengartner

++ 41 1 633 31 09

hengartner@ivt.baug.ethz.ch

Subject: "right angled"
Posted by Matthias Hengartner on Mon, 04 Oct 2004 13:01:29 GMT
View Forum Message <> Reply to Message

> The crucial thing is the required "orientation"-attribute in the
> <connection>-element of a switch. "orientation" can be either
> "incoming", "outgoing", "right angled" (???) or "unknown". Which value
> is to be chosen for the second track in case (1) and for both tracks in
> case (2)?

BTW: A "right angled" connection is used only for a (right-angled)
<crossing>, not for a <switch> (Ulrich Lindner, please correct me if I'm
wrong).

mh

Subject: Re: V1.00 RC1: switchRef/crossingRef
Posted by Volker Knollmann on Tue, 05 Oct 2004 06:53:05 GMT
View Forum Message <> Reply to Message

On 04.10.2004 14:57, Matthias Hengartner wrote:
>> I suggest an additional value "straight" (which perfectly coincides with
>> the possible values for "trackContinueCourse")
>
>
> In case (1a), I'd take "outgoing" for _both_ connected tracks (1b:
> "incoming"). The attribute "course" would have the value "straight" for the
> second connected track (and "left"/"right" for the first [1a/1b]).
> So in my opinion, we _could_ introduce the value "straight" in the
> "orientation"-attribute, but there's no need for it.

Waaaaaaaaaaaaaaaah

Shame on me! I didn't see the attribute "course" in the
<connection>-element. With the proper setting of "course", no further
attributes or values are required, you are right. Thanks for pointing me
on that...

>> and the __convention__ to
>> let the <switch>-element be part of track at the switch's tip. Thus,

>> role of every track is unambiguous.
>
>
> I agree fully with you!
> [...]
> But this is of course a very "dirty" implementation. And I don't think that
> the possibility to implement case (2) is really needed (It can easily be
> avoided).

Full ack. But the thing is to take EVERY case into account that CAN be
realized in railML and therefore must be handled by the interpreting
software.
Of course there's no need to implement switches in such a queer way like
case (2). But sooner or later, someone will fail to withstand the
temptations of the dark side of the force^W^W^W eeeeer railML... ;-)

> Another possibility... We could abandon the special treatment of
> switches/crossings which are placed on trackBegin/trackEnd (I feel a little
> uncomfortable about saying this, because this idea is penned by me...).
> However, then we'd have a <simpleConnection> and a <switch> which have the
> same position (on trackBegin/trackEnd).

We are brothers in mind.
That's exactly the way I defined the implementation of switches for data
exchange in our lab. Given a switch like this:

```
           o
          /
         /
        /
o-----------------o
```

I transformed this into two <track>-elements in railML:

```
     o
    /
   /
  /
o
o------------------o
```

with a switch of zero size at pos="0.000" of the lower track. This
implementation has another advantage: you can easily set up different
speed restrictions for the branching and the straight part, because you
have two <track>-elements (I know that there is a "vMax"-attribute for
the <connection>-element, but it was introduced too late for me).

> Or, final idea (for the moment ;-)): We could combine these 2 approaches:
> We could have a <simpleConnection> with a reference to a
> <switch>/<crossing>.

Hmm? That would be the above mentioned way: define a switch at
pos="0.000" or pos="[length]" and refer to it from the branching
element. The straight element is implicitly given by the parent of
<switch>. This avoids some extra attributes and the handling of special
cases....

Best regards from Braunschweig,
Volker Knollmann

Subject: Re: "right angled"
Posted by Volker Knollmann on Tue, 05 Oct 2004 06:55:48 GMT
View Forum Message <> Reply to Message

On 04.10.2004 15:01, Matthias Hengartner wrote:

> BTW: A "right angled" connection is used only for a (right-angled)
> <crossing>, not for a <switch>

Ah, I see.

Thanks for the explanation,
Volker

Subject: Re: V1.00 RC1: switchRef/crossingRef
Posted by Matthias Hengartner on Thu, 14 Oct 2004 16:00:25 GMT
View Forum Message <> Reply to Message

Hello...

>> But this is of course a very "dirty" implementation. And I don't think
that
>> the possibility to implement case (2) is really needed (It can easily be
>> avoided).
>
> Full ack. But the thing is to take EVERY case into account that CAN be
> realized in railML and therefore must be handled by the interpreting
> software.

> Of course there's no need to implement switches in such a queer way like
> case (2). But sooner or later, someone will fail to withstand the
> temptations of the dark side of the force^W^W^W eeeeer railML... ;-)

hmm. Since the railML schema does only define the syntax of a railML file (and not the semantics/rail logics/consistency/...), there have always lots of restrictions and conventions to be made in the documentation/specification. So we could also "prohibit" such "queer ways" of implenting switches.

>> Or, final idea (for the moment ;-)): We could combine these 2 approaches:
>> We could have a <simpleConnection> with a reference to a
>> <switch>/<crossing>.
>
> Hmm? That would be the above mentioned way: define a switch at
> pos="0.000" or pos="[length]" and refer to it from the branching
> element. The straight element is implicitly given by the parent of
> <switch>. This avoids some extra attributes and the handling of special
> cases....

No, this was not my idea (I have to admit that the description of my idea was not very comprehensible...):

Given a switch on the beginning or end of a track (track1)

```
    o track2
   /
  /
 /
o
o------------------o track1
```

then we have a switch at pos="0.000" / pos="[length]" with a connection to the <trackBegin>/<trackEnd> - <simpleConnection> - <connection> of track2.
So far nothing new.
Of course, we also have <trackBegin>/<trackEnd> - <simpleConnection> (***) on track1, with a connection to the previous/next <track> (which would be on the left in our ASCII-drawing).
Still nothing new.

The only additional thing would be a new attribute of the <simpleConnection>(***), something like "switchIDRef", which would refer to the switch (which is on the same track).

Is this a bit more comprehensible?

Best regards from Zurich in dark clouds
Matthias