Subject: Re: the use of @dir in railML.
Posted by Thomas Nygreen JBD on Wed, 24 Oct 2018 13:17:37 GMT
View Forum Message <> Reply to Message

Dear Dirk, dear Christian,

Thank you for your kind words!

Quote:In many aspects I see a kind of rather philosophical background in it which we also often had and have in <timetable> discussions: Whether to make railML rather "flexible" or rather "strong, exact".

My intention is to ensure that we know how to interpret a given railML file. I welcome flexibility in the use and purposes of railML, but not in interpretation. When I suggest to deprecate @dir for a lot of elements it is simply because I cannot find a good interpretation for it.

Quote:The railML 2 attribute @dir is named @applicationDirection in RTM and railML 3.

Much better! However, this attribute will probably never be self-explanatory unless you use a stupidly long and complex name, so documentation will still be important ;)

Quote:> * DEPRECATE @dir for border and trackCircuitBorder

I can imagine borders which need a direction. As I mentioned earlier in a post with Torben about station limits, in Germany we have direction-dependant station limits. For instance, from the view of interlocking, for a train entering a station, the limit is the home signal (Einfahrsignal). For a train of the opposite direction, that leaves the station, the limit is the shunting-limit marker (Rangierhalttafel Ra10).

So, if a <border> can for instance encode the interlocking or operational station limits, we may need a @dir attribute.

As station is a valid @type for <border>, and you describe a use case for @dir for that @type, we should keep @dir for <border>.

Quote:And what about track circuit borders, where there is a track circuit
only on one side (like we have it in the Simple Example at signal 69A)?
The attribute @applicationDirection has been used there to define on
which side(s) of the track circuit border track circuits can be found.
Do you consider this being logical?

Short answer: no.

Longer answer:
* As a general rule, please do not use the same attribute name to mean very different things for different objects. Contrary to e.g. a signal, a track circuit border applies to trains running in both directions, even if there is a track circuit only on one side. For trains in one direction it marks where they are entering the track circuit section, and in the other direction it marks where they are

leaving it.
* The need for information about the track circuits themselves should be solved by adding a trackCircuit element, not by abusing a vague attribute of <trackCircuitBorder>. If you really need to use an attribute, create a new one.
* In the railML2.3 version of the Simple Example, @dir is not used for the <trackCircuitBorder>s. (Instead it looks like you are using @insulatedRail="none", which to me makes no sense, but that is probably another discussion.)
* In case anyone argues that track circuits belong to interlocking: I disagree. They are physical objects (wired circuits), that are used by the interlocking to determine the current state, just like the interlocking also uses positions of switches and derailers.

Quote:For many years, we use <speedChange> in the way you describe it, with @dir=up or =down dependant to the validity direction of speedProfiles:
- We always encode speedProfiles in the direction of (relative) mileage, means in order of raising @pos attributes, independent of the validity direction (of travel) of the speedChanges.
- In cases of a speedProfile is valid for the =down direction, our <speedChange>@speed value is valid from that @pos until the previous (!)(not next!) <speedChange>.
- The <speedChange>@dir never encodes the validity direction of the <speedProfile>. It only says in which direction (next or previous element) the @speed is valid. The <speedProfile> has its own direction an can be valid for both directions.
- This is documented in our own documentation of FBS-railML-interface, which is a kind of extension and concretion of the general Wiki.

From your description, it is not fully clear to me if you are referring to the way I described as the intended usage or the assumed common usage. It is probably my fault, as I did not write the railML code for the last one. So, just to be as clear as possible, using the same example:

```
Pos     0               100               200
Track   |-------------------------------------------->
vMax -> |       60        |        80
vMax <- |       40        |        80
```

The two alternatives for the first section (0100 m) in railML2.x are (differences in bold):

Alternative A (intended use, according to Susanne Wunsch in 2012):
<speedChange pos="0" dir="up" vMax="60"/>
<speedChange pos="0" dir="down" vMax="40"/>

Alternative B (suspected common use):
<speedChange pos="0" dir="up" vMax="60"/>
<speedChange pos="100" dir="down" vMax="40"/>

There is one aspect I did not raise in my previous post, which is the handling of sections that are equal in the two directions. Here there are three alternatives:

Alternative 1 (no @dir):
<speedChange pos="100" vMax="80"/>

Alternative 2 (with @dir as Alternative A):
<speedChange pos="100" dir="up" vMax="80"/>
<speedChange pos="100" dir="down" vMax="80"/>

Alternative 3 (with @dir as Alternative B):
<speedChange pos="100" dir="up" vMax="80"/>
<speedChange pos="2xx" dir="down" vMax="80"/>

Alternatives 1 and 2 are equally compatible with Alternative A, while I assume that a generator using Alternative B would always use Alternative 3. I will check which alternatives are used by the different railML certified software available to me, and I welcome you to do the same. As I have already stated, I find A+1 (or A+2) to be the preferred way, but it is more important not to break interfaces already in use. So if there is a consistent common way, we should keep it and document it.

Quote:>> Suggested change:
>> * Document what part of the track the new values apply to
>> when using the @dir attribute
>
> I can do it if Christian Rahmig agrees and/or if there will be no objections here.

Absolutely no objections from my side! I am thankful for any
contribution. I suggest to do it on the discussion page of element
<speedChange> in the railML wiki
(https://wiki.railml.org/index.php?title=IS:speedChange). Once, we have
an agreement on the final solution, we may integrate it as "Best
Practice" on the main wiki page of <speedChange>.

I would be happy to contribute on the wiki. However, my request for access long ago was never answered.

Quote:>> Suggested changes:
>> * DEPRECATE @dir for brigde, levelCrossing, platformEdge,
>> serviceSection (?) and tunnel
>
> I think is is a misunderstanding:
> The intention behind @dir was never to encode that the tunnel or bridge is not visible for the other direction.
> The intention may have been:
>   <tunnel @pos=1234 @length=200 @dir=up> = a tunnel from km 1,234 to 1,434
>   <tunnel @pos=1234 @length=200 @dir=down> = a tunnel from km 1,034 to 1,234
> This would fit in a certain way to the way <speedChanges> are to interpret from @dir.
>
> However, I have no objections against killing this redundancy and deprecate @dir from brigdes,

tunnels etc.

For final answering this question I would like to receive more feedback from the community. Do we want to introduce a rule that says: "infrastructure elements with length have to be defined always in direction of the track orientation (increasing @pos)"? If there is a majority supporting this approach, we may think about an implementation (in railML 2.5?).

I never meant to suggest that we should interpret a <brigde dir="up"> as solid ground in the other direction. I just meant that using @dir at all was illogical for these elements. I think Dirk's suggestion is the most probable for what the intended purpose may have been. Still, until confirmed we do not know if this was really the intended use, and how logical it is depends on how you interpret @dir for <speedChange>. As long as there is no documented way to interpret @dir for bridges and tunnels, we need to do something. We cannot have different software interpreting the location of a tunnel differently. From the existing documentation of @pos (distance from <trackBegin>) I would assume that Christians proposed rule was already established, making @dir redundant. (If you really, really need tunnels going backwards, @length allows negative values...)

Best regards,
Thomas
