Subject: the use of @dir in railML.
Posted by Torben Brand on Wed, 12 Sep 2018 12:57:50 GMT
View Forum Message <> Reply to Message

I would like to address the use of @dir in railML.
The direction of the objects on the track is always given in railML through the track direction (from the <trackBegin> to the <trackEnd> or seen in increasing relative position values)
@dir denotes the validity of the objects as seen from the direction of travel by the train. Or as it says similar in the gradientChange wiki: "dir: This defines the validity of gradientChange along the track."
Suggestion for a more clear term in railML3 that creates less confusion would be @validDir (or something similar).

Direction restrictions come in three pre-set levels in railML:
"lax" with the enumeration values "up","down","unknown","none","both"
"delimited" with the enumeration values "up","down","unknown"
"strict" with the enumeration values "up" or "down"

The values are defined to (taken from wiki for gradientChange):

Possible values are:
• none: gradientChange has no direction restriction.
• up: This denotes the direction from the <trackBegin> to the <trackEnd> (increasing relative position values).
• down: This goes opposite to up (decreasing relative position values).
• both: gradientChange is valid in both directions.
• unknown: gradientChange is restricted to a certain direction, but this direction is not known.

First having the value "none" and "both" make no sense. This as they both cover the same thing (glass is half full or half empty)

The wiki pages describing @dir for gradientChange, trackCircuitBorder and trainDetector (possible other pages that use @dir, I have not checked) are inconsistent.

The possible values listed are the 5 lax values.
Under constraint it is referenced to: "dir: xs:string, generic type for more constrained direction statements: enumeration up, down, unknown; derived from tLaxDirection; optional "
tLaxDirection are the 5 listed values, but the values listed in constraint are the tDelimitedDirection values.
In the XSD the tStrictDirection values are used for gradientChange and tDelimitedDirection values are used for trainDetector and trackCircuitBorder. So only the values "up" and "down" (and "unknown") are allowed under @dir.
As the XSD is the master I suggest to edit the wiki pages to the following:

Possible values are:
• up: This denotes the direction from the <trackBegin> to the <trackEnd> (increasing relative position values).

• down: This goes opposite to up (decreasing relative position values).
Setting no @dir attribute defines that the object has no direction restraint. This corresponds to the lax values "none" or "both". Modelling of "unknown" is currently not possible for @dir in railML2.

Constraint
dir: xs:string, generic type for more constrained direction statements: enumeration up, down; derived from tStrictDirection; optional

gradientChange: https://wiki.railml.org/index.php?title=IS:gradientChange
trackCircuitBorder: https://wiki.railml.org/index.php?title=IS:trackCircuitBorde r
trainDetector: https://wiki.railml.org/index.php?title=IS:trainDetector

---

Subject: directions in railML 3
Posted by christian.rahmig on Mon, 17 Sep 2018 10:01:42 GMT
View Forum Message <> Reply to Message

Dear Torben,

let me cut your posting in several small pieces to be answered...

Am 12.09.2018 um 14:57 schrieb Torben Brand:
> [...]
> @dir denotes the validity of the objects as seen from the
> direction of travel by the train. Or as it says similar in
> the gradientChange wiki: "dir: This defines the validity
> of gradientChange along the track."
> Suggestion for a more clear term in railML3 that creates
> less confusion would be @validDir (or something similar).

In railML 3, we follow the RailTopoModel that already defines the direction attribute @applicationDirection in classes LinearLocation and SpotLocation.

@all: Please let us know if you would prefer another name for this attribute.

Thank you very much and best regards
Christian Rahmig

--
Christian Rahmig - Infrastructure scheme coordinator
railML.org (Registry of Associations: VR 5750)
Phone Coordinator: +49 173 2714509; railML.org: +49 351 47582911
Altplauen 19h; 01187 Dresden; Germany www.railml.org

Subject: Values of attribute @dir
Posted by christian.rahmig on Mon, 17 Sep 2018 10:15:13 GMT
View Forum Message <> Reply to Message

Dear Torben,

here comes answer part 2...

Am 12.09.2018 um 14:57 schrieb Torben Brand:
> [...]
> @dir denotes the validity of the objects as seen from the
> direction of travel by the train. Or as it says similar in
> the gradientChange wiki: "dir: This defines the validity
> of gradientChange along the track."
> [...]
>
> Possible values are:
> • none: gradientChange has no direction restriction.
> • up: This denotes the direction from
> the <trackBegin> to the <trackEnd> (increasing
> relative position values).
> • down: This goes opposite to up (decreasing
> relative position values).
> • both: gradientChange is valid in both directions.
> • unknown: gradientChange is restricted to a certain
> direction, but this direction is not known.
>
> First having the value "none" and "both" make no sense. This
> as they both cover the same thing (glass is half full or
> half empty)

I think you are right. The value "none" does not make much sense since all possibilities of direction validity are covered by the other values:
* up (for elements being valid in up direction)
* down (for elements being valid in down direction)
* both (for elements being valid in both directions)

And if an information is unknown, you may leave this optional attribute empty.

So, we may put it on the agenda for a next railML version? By the way, in railML 3, the new attribute @applicationDir has been implemented exactly that way - with only three values. Instead of "up" and "down" the terms "normal" and "reverse" are being used, but the meaning is the same.

@all: Do you have any examples where you use direction enumeration values "none" or "unknown"?

Best regards
Christian

--
Christian Rahmig - Infrastructure scheme coordinator
railML.org (Registry of Associations: VR 5750)
Phone Coordinator: +49 173 2714509; railML.org: +49 351 47582911
Altplauen 19h; 01187 Dresden; Germany www.railml.org

Subject: Re: the use of @dir in railML.
Posted by Thomas Nygreen JBD on Mon, 08 Oct 2018 13:14:03 GMT
View Forum Message <> Reply to Message

I would like to look a bit broader at the use and interpretation of the @dir attribute. As Torben points out, the documentation is lacking, and in some cases incorrect. Currently this attribute is not documented in the wiki or XSD at all, apart from the quite uninformative "This defines the validity of <element> along the track" and the restrictions described in more technical terms. The description that Torben mentions says "derived from tLaxDirection" because it is a description of the (badly named) tDelimitedDirection, and this type is indeed derived from tLaxDirection.

I agree that "none" and "both" are two sides to the same coin, and one of them can be removed. Regarding "unknown", there is according to the documentation a subtle difference between @dir="unknown" and a missing @dir: with "unknown" the element "is restricted to a certain direction, but this direction is not known", while with a missing @dir we do not know if there is any restriction at all (or in most cases we assume that there is not). However, I have yet not seen a real use case, and unless anyone else has, it can maybe also be removed.

Suggested change:
* Remove "unknown" and "none" from enumeration

There are 26 elements in the railML 2.3/2.4 infrastructure schema that support the dir attribute. Additionally there are two elements where this attribute is deprecated (crossSection and mileageChange), which I will disregard below. The elements can be divided into three categories: elements without extent, elements with extent and elements that describe a change that occurs at a given point.

The first group, elements without extent (i.e. without a @length) include: balise, border, derailer, signal, stopPost, trackCircuitBorder, trainDetector and trainProtectionElement. The interpretation of @dir for most of the physical elements in this group is quite simple: they apply only to trains in that direction. For trackCircuitBorder, however, a rail joint cannot be isolated in only one direction. For the more abstract border element, I am not quite sure about the purpose of @dir. The only one I can think of is that there could be borders that are placed differently in the two directions.

Suggested changes:
* DEPRECATE @dir for border and trackCircuitBorder
* Remove "both" from enumeration for derailer, signal and stopPost unless anyone has an

example

The second group, the elements with an extent (i.e. with @length) include: brigde, levelCrossing, platformEdge, serviceSection, trackCondition and tunnel. This group is less simple. Using the same logic and definition as above would imply that a tunnel, bridge etc. with @dir="up" is "not there" if you drive in the "down" direction, and can be ignored by those trains. If tunnels that are invisible and without air resistance in one direction actually exist, please inform me! The only element in this group that can be interpreted in this fashion is trackCondition. For the remaining elements the intention may have been to specify that it is only allowed to drive through/along the element in one direction, but this would mean that @dir must be interpreted very differently for these elements. Furthermore, such a restriction would not be a property of this element, but of the track it is placed on or the signalling and interlocking system. Consequently I suggest to deprecate @dir for all these elements, except trackCondition.

Suggested changes:
* DEPRECATE @dir for brigde, levelCrossing, platformEdge, serviceSection (?) and tunnel

The last group, elements that describe a change, include: axleWeightChange, clearanceGaugeChange, electrificationChange, gaugeChange, gradientChange, operationModeChange, ownerChange, powerTransmissionChange, radiusChange, speedChange, trainProtectionChange and trainRadioChange. In these cases the general interpretation of @dir is quite simple: the specified change only applies when traversing the track in the given direction. The best known example is probably speedChange, where it is common knowledge that speed profiles my differ in the two directions. The detailed interpretation can however be tricky, as I will soon describe. Also, for some of these characteristics, it does not make much sense allowing different values in different directions. I suggest to deprecate the @dir attribute for most of these elements, as I do not know of applications where their properties can differ by direction. I welcome all examples to the contrary. For speed, train protection and train radio, there can obviously be some differences between directions, at least in exactly where the change happens. For gradients, there are use cases where the exact gradient at every point is unknown but the average or characteristic gradient between consecutive signals is known, and since signal placement may vary between directions, so will these averaged gradients (even if the track itself must have the same gradient in both directions).

Suggested changes:
* DEPRECATE @dir for axleWeightChange, clearanceGaugeChange, electrificationChange, gaugeChange, operationModeChange, ownerChange, powerTransmissionChange and radiusChange unless anyone has an example
* Only allow up/down for trainRadioChange

The remaining question for all change elements when @dir is used, is what part of the track the given values apply to. Let's start with a speedChange example without @dir:

<speedChange pos="0" vMax="40"/>
<speedChange pos="100" vMax="80"/>

This is simple to interpret as the following. Generally, the new values apply to the segment of the track between this element and the next in the direction of the track.

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

Now, if we alter one direction, what should the @pos value of the element with @dir="down" be?

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

I cannot find any documentation answering that question. According to a very old thread on this forum, the intended use was like this:

```
<speedChange pos="0" dir="up" vMax="60"/>
<speedChange pos="0" dir="down" vMax="40"/>
<speedChange pos="100" vMax="80"/>
```

While, in my experience, and also mentioned in a later post in the same old forum thread, the common practice is to use @pos="100" on the @dir="down" element. I find the intended use to be more consistent: the given new values always apply to the segment of track between this change and the next change of the same type found in the direction of the track. The @dir attribute, if given, only affects which direction (of travel) the values apply to.

BUT, it is important what the common practice is. So unless I am mistaken about the current common practice, the definition would be: unless @dir="down" the given new values apply to the segment of track between this change and the next change of the same type found in the direction of the track. When @dir="down", the given new values apply to the segment of track between this change and the previous change of the same type.

Regardless of choice of definition, it should be the same for all change elements.

Suggested change:
* Document what part of the track the new values apply to when using the @dir attribute

---

Subject: Re: the use of @dir in railML.
Posted by on Fri, 19 Oct 2018 08:34:05 GMT
View Forum Message <> Reply to Message

Dear Thomas,

I found your composition on @dir very refreshing and fundamental and welcome the detailed work. Hopefully, it was not for nothing... ;-)

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".

To make it short: I don't have general objections against you suggestions. I also would welcome to make @dir more exact.

As often, there is unfortunately a "but":

> * 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.

Concerning <speedChange>s:

> BUT, it is important what the common practice is.

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.

> 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.

> 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.

With best regards,
@Dir_k.

Subject: Re: the use of @dir in railML.
Posted by christian.rahmig on Fri, 19 Oct 2018 14:42:10 GMT
View Forum Message <> Reply to Message

Dear Thomas, dear @Dir_k ;-)

Am 19.10.2018 um 10:34 schrieb Dirk Bräuer:
> I found your composition on @dir very refreshing and fundamental and welcome the detailed work. Hopefully, it was not for nothing... ;-)

I agree with Dirk: Thomas, your notes on the @dir topic are very
detailed and useful for further railML (2 and 3) development. Thank you!

> To make it short: I don't have general objections against you suggestions. I also would welcome to make @dir more exact.

I take your advice very serious for railML 3.1 implementation. The
railML 2 attribute @dir is named @applicationDirection in RTM and railML 3.

>> * 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.

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?

> Concerning <speedChange>s:
>
>> BUT, it is important what the common practice is.
>
> 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.

In railML context, <speedChange> elements may have a direction while <speedProfile> don't have this attribute. However, Dirk is right when he says that <speedProfile> elements have a direction information, too. It is given implicitly by the <speedChange> elements referencing this <speedProfile>. So, if elements <speedChange>@dir=up and <speedChange>@dir=down reference the same <speedProfile>, this <speedProfile> is de-facto valid for both directions.

>> 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>.

>> 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?).

Best regards
Christian

--
Christian Rahmig - Infrastructure scheme coordinator
railML.org (Registry of Associations: VR 5750)
Phone Coordinator: +49 173 2714509; railML.org: +49 351 47582911
Altplauen 19h; 01187 Dresden; Germany www.railml.org

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

Subject: Re: the use of @dir in railML.
Posted by christian.rahmig on Mon, 06 Jan 2020 15:23:55 GMT
View Forum Message <> Reply to Message

Dear Thomas,
dear all,

Thomas Nygreen wrote on Wed, 24 October 2018 15:17[...]

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 ;)

In railML 3.1, the attribute @applicationDirection is documented like this: "direction in which the element is applied, related to the orientation of the <netElement>"

What do you think: Is that precise enough?

Best regards
Christian

Subject: Re: the use of @dir in railML.
Posted by christian.rahmig on Mon, 13 Jan 2020 14:09:32 GMT
View Forum Message <> Reply to Message

Dear all,

as Thomas correctly pointed out in his post [1], the discussion here has not yet been concluded. Therefore, I would like to summarize the proposal about the usage of @dir in railML 2.x:

Elements without extent (without @length attribute)
examples: balise, border, derailer, signal, stopPost...
usage: @dir describes the direction of travel, for which the element applies. Possible values are "up", "down" and "both". A missing @dir attribute means that the application direction of this element is unknown.
proposal: DEPRECATE @dir for trackCircuitBorder

Elements with extent (with @length attribute)
examples: bridge, levelCrossing, platformEdge, serviceSection...
usage: @dir describes the direction of travel, for which the element applies. Possible values are "up", "down" and "both". A missing @dir attribute means that the application direction of this element is unknown. By standard, the elements' orientation (not their application direction!) shall

be always in direction of track orientation (from trackBegin towards trackEnd).
proposal: DEPRECATE @dir for brigde, levelCrossing, platformEdge, serviceSection and tunnel

Elements that describe a change
examples: axleWeightChange, clearanceGaugeChange, electrificationChange, gaugeChange, speedChange...
usage: @dir describes the direction of travel, for which the change applies. Possible values are "up", "down" and "both". A missing @dir attribute means that the application direction of this change element is unknown. By standard, the change elements' orientation (not their application direction!) shall be always in direction of track orientation (from trackBegin towards trackEnd).
proposal: DEPRECATE @dir for elements where properties cannot differ by direction of travel, e.g. axleWeightChange, clearanceGaugeChange, electrificationChange, gaugeChange, ownerChange, powerTransmissionChange, radiusChange

Please let us know if you agree with these principles from your application point of view. Specifically: is that the same understanding how you use @dir currently and does it match with the concept of how you want to use @dir in future?

[1] https:// www.railml.org/forum/index.php?t=msg&th=665&start=0& amp;

Thank you very much and best regards
Christian

Subject: Re: the use of @dir in railML.
Posted by Thomas Nygreen on Mon, 27 Jan 2020 16:35:06 GMT
View Forum Message <> Reply to Message

Den 06.01.2020 16:23, skrev Christian Rahmig:
> In railML 3.1, the attribute @applicationDirection is
> documented like this: "direction in which the element is
> applied, related to the orientation of the <netElement>"
>
> What do you think: Is that precise enough?

I'm sorry, but no I do not think it is self-explanatory what "is applied" means. Not any more than the previous "is valid". If you put applicationDirection on a <track>, does it mean that the rails have been applied (as in laid down) in that direction? What about "direction *of traffic* for which the element applies/is applicable"?

Den 13.01.2020 15:09, skrev Christian Rahmig:
> Elements without extent (without @length attribute)
> examples: balise, border, derailer, signal, stopPost...
> usage: @dir describes the direction of travel, for which the
> element applies. Possible values are "up", "down" and
> "both". A missing @dir attribute means that the application

> direction of this element is unknown.
> proposal: DEPRECATE @dir for trackCircuitBorder

I agree

> Elements with extent (with @length attribute)
> examples: bridge, levelCrossing, platformEdge,
> serviceSection...
> usage: @dir describes the direction of travel, for which the
> element applies. Possible values are "up", "down" and
> "both". A missing @dir attribute means that the application
> direction of this element is unknown. By standard, the
> elements' orientation (not their application direction!)
> shall be always in direction of track orientation (from
> trackBegin towards trackEnd).
> proposal: DEPRECATE @dir for brigde, levelCrossing,
> platformEdge, serviceSection and tunnel

Rather than writing about orientation, I would just write that the extent of the element is from @pos to @pos + @length along the track.

> Elements that describe a change
> examples: axleWeightChange, clearanceGaugeChange,
> electrificationChange, gaugeChange, speedChange...
> usage: @dir describes the direction of travel, for which the
> change applies. Possible values are "up", "down" and "both".
> A missing @dir attribute means that the application
> direction of this change element is unknown. By standard,
> the change elements' orientation (not their application
> direction!) shall be always in direction of track
> orientation (from trackBegin towards trackEnd).
> proposal: DEPRECATE @dir for elements where properties
> cannot differ by direction of travel, e.g. axleWeightChange,
> clearanceGaugeChange, electrificationChange, gaugeChange,
> ownerChange, powerTransmissionChange, radiusChange

What is the orientation of a change element? Are you referring to the sign of the value for gradientChange and radiusChange? I think these are the only two where orientation matters. For gradients, positive values should always mean an uphill slope in the direction of the track. For radii, positive values should always mean a turn to the right in the direction of the track.

Best regards,
Thomas Nygreen - Common schema coordinator, railML.org

Subject: Re: the use of @dir in railML.
Posted by christian.rahmig on Wed, 26 Feb 2020 10:05:44 GMT
View Forum Message <> Reply to Message

Thomas Nygreen wrote on Mon, 27 January 2020 17:35Den 06.01.2020 16:23, skrev Christian Rahmig:
> In railML 3.1, the attribute @applicationDirection is
> documented like this: "direction in which the element is
> applied, related to the orientation of the <netElement>"
>
> What do you think: Is that precise enough?

I'm sorry, but no I do not think it is self-explanatory what "is
applied" means. Not any more than the previous "is valid". If you put
applicationDirection on a <track>, does it mean that the rails have been
applied (as in laid down) in that direction? What about "direction *of
traffic* for which the element applies/is applicable"?
I agree. The @applicationDirection shall be defined as the direction of traffic for which the element is applicable.

Thomas Nygreen wrote on Mon, 27 January 2020 17:35
Den 13.01.2020 15:09, skrev Christian Rahmig:
> Elements without extent (without @length attribute)
> examples: balise, border, derailer, signal, stopPost...
> usage: @dir describes the direction of travel, for which the
> element applies. Possible values are "up", "down" and
> "both". A missing @dir attribute means that the application
> direction of this element is unknown.
> proposal: DEPRECATE @dir for trackCircuitBorder

I agree
Great! Any other opinions?

Thomas Nygreen wrote on Mon, 27 January 2020 17:35
> Elements with extent (with @length attribute)
> examples: bridge, levelCrossing, platformEdge,
> serviceSection...
> usage: @dir describes the direction of travel, for which the
> element applies. Possible values are "up", "down" and
> "both". A missing @dir attribute means that the application
> direction of this element is unknown. By standard, the
> elements' orientation (not their application direction!:wink:
> shall be always in direction of track orientation (from
> trackBegin towards trackEnd).
> proposal: DEPRECATE @dir for brigde, levelCrossing,
> platformEdge, serviceSection and tunnel

Rather than writing about orientation, I would just write that the
extent of the element is from @pos to @pos + @length along the track.

Yes, this formulation is short and precise. Let's use it.

Thomas Nygreen wrote on Mon, 27 January 2020 17:35
> Elements that describe a change
> examples: axleWeightChange, clearanceGaugeChange,
> electrificationChange, gaugeChange, speedChange...
> usage: @dir describes the direction of travel, for which the
> change applies. Possible values are "up", "down" and "both".
> A missing @dir attribute means that the application
> direction of this change element is unknown. By standard,
> the change elements' orientation (not their application
> direction!:wink: shall be always in direction of track
> orientation (from trackBegin towards trackEnd).
> proposal: DEPRECATE @dir for elements where properties
> cannot differ by direction of travel, e.g. axleWeightChange,
> clearanceGaugeChange, electrificationChange, gaugeChange,
> ownerChange, powerTransmissionChange, radiusChange

What is the orientation of a change element? Are you referring to the sign of the value for gradientChange and radiusChange? I think these are the only two where orientation matters. For gradients, positive values should always mean an uphill slope in the direction of the track. For radii, positive values should always mean a turn to the right in the direction of the track.
A change element always contains a change of a feature from an old value to a new value. E.g. referring to an <electrificationChange> there is a state of electrification before the change as well as after the change. By setting the "change element orientation" I want to specify that the old value applies for the track until <*Change>@pos and the new value (as provided by the change element) applies from <*Change>@pos ongoing.

Any further comments are highly appreciated...

Best regards
Christian

Subject: Re: the use of @dir in railML.
Posted by Thomas Nygreen on Thu, 27 Feb 2020 18:08:27 GMT
View Forum Message <> Reply to Message

Dear Christian,
Dear all,

It looks like we are finally near to closing this issue. I just have a few final questions/comments:

<lock> was introduced after I did my review of all elements with @dir. I suggest that we deprecate @dir also for <lock>.

We agree to deprecate @dir on <trackCircuitBorder>. But what about <trainDetector>? Are there axle counters that only detect axles going in one direction? I have a feeling that we should treat <trackCircuitBorder> and <trainDetector> the same way.

As I understand Christian's proposals, there will no longer be three different sets of values for @dir. All occurences will allow "up", "down" and "both". I think that is a good solution, as it also allows equal interpretation across elements of a missing @dir as unknown. I assume we will have to keep three different types in the XSD, where tLaxDirection will have two deprecated values ("unknown" and "none"), [the badly named] tDelimitedDirection will have one new value ("both") and one deprecated ("unknown") and tStrictDirection will have one new value ("both").

And then, finally, there are the <*Change> elements.

Den 13.01.2020 15:09, skrev Christian Rahmig:
>>> By standard, the change elements' orientation (not their
>>> application direction!) shall be always in direction of track
>>> orientation (from trackBegin towards trackEnd).

Den 26.02.2020 11:05, skrev Christian Rahmig:
> A change element always contains a change of a feature from
> an old value to a new value. E.g. referring to an
> <electrificationChange> there is a state of electrification
> before the change as well as after the change. By setting
> the "change element orientation" I want to specify that the
> old value applies for the track until <*Change>@pos and the
> new value (as provided by the change element) applies from
> <*Change>@pos ongoing.

Just to make sure we are thinking alike here:

For the <*Change> elements where we deprecate @dir (axleWeightChange, clearanceGaugeChange, electrificationChange, gaugeChange, ownerChange, powerTransmissionChange and radiusChange) this means that the value is simply a property of the infrastructure, regardless of the direction of traffic. The new value given in the <*Change> element describes the infrastructure from the point it is placed (@pos) towards the track end. This is as documented in the wiki for <radiusChange>.

For the <*Change> elements where we keep @dir (gradientChange, operationModeChange, speedChange, trainProtectionChange and trainRadioChange), @dir only describes which direction of traffic the new value applies to. The value is applied to the infrastructure in the same way as for the other <*Change> elements. To reiterate a previous example, this means that

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

would be exported as:

<speedChange pos="0" dir="up" vMax="60"/>
<speedChange pos="0" dir="down" vMax="40"/>
<speedChange pos="100" dir="both" vMax="80"/>

Best regards,
Thomas

Subject: Re: the use of @dir in railML.
Posted by on Mon, 09 Mar 2020 10:39:06 GMT
View Forum Message <> Reply to Message

Dear Thomas, Christian and all,

sorry, Thomas, for not answering your question from 24.10.2018 so far. Somehow, I lost connection...

Concerning <speedChange>s, I want to record as a summary from our side:

- I understand Thomas' suggestion to interpret "simply a property of the infrastructure, regardless of the direction of traffic". I have full sympathy for this kind of usage and would normally not plead against it.

- However, unfortunately the usage was different in the last decade or more: Our (FBS) railML export interface clearly produces speedChanges in the following way:

> Die Geschwindigkeitswechsel sind entsprechend der Gültigkeitsrichtung ihres Geschwindigkeitsprofils angegeben. Im Falle eines Geschwindigkeitsprofils, das für beide Fahrtrichtungen gültig ist, sind die Informationen redundant doppelt vorhanden – je einmal pro Fahrtrichtung.

(The <speedChange>s are exported depending on the validity direction of the <speedProfile>. In case of a <speedProfile> valid for both directions, the <speedChange>s are exported redundantly doubled once per driving direction.)

> Je nach Gültigkeitsrichtung des Geschwindigkeitsprofils ist an der Position des Streckenanfangs (<trackBegin>.pos) bzw. Streckenendes (<trackEnd>.pos) immer ein Geschwindigkeitswechsel enthalten.

(Depending on the validity direction of the <speedProfile>, there is always a <speedChange> at the position of <trackBegin>.pos or <trackEnd>.pos.)

- The reason for this (possibly unexpected) behaviour is, as far as I remember, an objection of Susanne Wunsch in the time of publishing and certifying our FBS-RailML2-Export (Susanne was common coordinator in that time). You can still read Susannes opinion in [1].

I want to repeat that I have full sympathy for Thomas' kind of interpretation, which is different than

our current usage. But a specification by railML now may not lead to existing valid railML files becoming invalid at once. So, unfortunately, the only solution I see is either to clarify the "new" interpretation as a future recommendation only, clearly allowing the "old" interpretation for backward compatibility in existing use cases, or to limit this discussion on railML 3.x.

However, the "advantage" of the old interpretation is that it keeps to the "rule": railML infrastructure orientates on what you can touch in the outside world. There is a speed change sign (post) at the beginning of each speed change depending on the driving direction, not depending on the definition direction. There are two speed change signs in case of tracks which are used in both directions. Therefore, the old interpretation may be more complicated and redundant, but at least it is closer to the outside world and therefore may be easier to understand by non-informatics, beginners etc.

Best regards,
Dirk.

[1] https://www.railml.org/forum/index.php?t=msg&th=120& goto=276&#msg_276