Subject: How to model ETCS BL2 speed restrictions and gradients, in railML v2.x
Posted by Jörgen Strandberg on Thu, 29 Nov 2018 13:56:34 GMT
View Forum Message <> Reply to Message

In search of a best practice for, in railML v2.x, modeling ETCS BL2 speed restrictions and gradients (slope) for the complete track, I would like to propose these semantic restriction:

A gradientChange/speedChange shall be automatically ended at the trackBegin/trackEnd in @dir. Rationale: To avoid unnecessary processing of connected tracks when writing and interpreting data.

Additionally I would like to point to some parts of the railML documentation on the wiki that need clarification, such as valid values for @dir and @vMax special value 999.

See attached docx-file.

Regards,
Jörgen Strandberg

File Attachments

1) railml-forum-posting-etcs-profile.docx, downloaded 436 times

Subject: Re: How to model ETCS BL2 speed restrictions and gradients, in railML v2.x
Posted by Jörgen Strandberg on Fri, 07 Dec 2018 08:33:18 GMT
View Forum Message <> Reply to Message

For clarity, here is the content of the word document attached in the initial posting:

Posting on forum.railml.org about ETCS profile data
2018-11-29
In search of a best practice for, in railML, modeling ETCS BL2 speed restrictions and gradients (slope) for the complete track.

These elements and attributes can be used to model speed and gradient data of a track:
1. gradientChange
 a. @slope
2. infrastructure.infraAttrGroups.infraAttributes
 a. @id
 b. speeds.speed@vMax
 c. speeds.speed@etcsTrainCategory
 d. speeds.speed@profileRef - to define dependent on e.g. axle load
3. track.infraAttrGroupRefs
 a. infraAttrGroupRef@ref
4. speedChange

a. @pos
b. @dir
c. @vMax
d. @etcsTrainCategory
e. @profileRef

5. speedProfile

a. @maxAxleLoad

1. gradientChange
This seems to be the only way to enter slope of track.

Description of @dir value range on wiki.railml.org is incorrect. Only up and down are valid according to schema, and that is what the wiki should say too.

For data to be useful for ETCS the slope of each part of the track must be known.
I question whether both these approaches should be supported:
1. Deduce from connected tracks. The last gradientChanges in routes through the network ending up in the track in question would specify the slope. A gradientChange would then only be defined when contradicting values are deduced from the different routes.
2. Always define a gradientChange at pos=0 of a track, even if the value is unchanged compared to the connected tracks. Or in other words limit the effect of a gradientChange element to the track it is defined in.

Approach 1. (above) would require processing of both readers and writers of the railML data. I propose that approach 2. (above) is promoted to be the only valid approach.

@slope is given in relation to @pos and @dir so that: if standing at @pos, looking along the track in the direction of @dir, a positive value of @slope will be considered as seeing that the track continues uphill.

The slope in down direction should for ETCS purposes be allowed to have a different value than in up direction.
But if the slope value in down direction is omitted, then the inverse value of the slope in up direction is assumed (as slope value in the down direction).

2. infrastructure.infraAttrGroups.infraAttributes
Provides a way to define speeds (static, train category dependent and axle load dependent) that can be referenced for a whole track as default values. See speedChange below for suggestions regarding @vMax, @etcsTrainCategory, and @profileRef.

3. track.infraAttrGroupRefs
Provides a way to reference a number of predefined speed elements that make up the default values for a whole track. These defaults are then possible to override with speedChanges.
When an infraAttrGroupRef is defined all speedChanges stating that data can be removed. A speedChange that is not immediately followed by another speedChange needs to be terminated with an additional speedChange that sets @vMax=end (@vMax=999), and then the default values of the track shall be valid.

The type of element to reference with track.infraAttrGroupRefs is incorrectly documented in railML 2.2, however there is a Key/KeyRef definition that correctly describes the reference, which should make this a valid construction.

4. speedChange
Similar to gradientChange, whether or not to deduce values from speedChanges of connected tracks, is a relevant question.
And I propose to limit the effect of a speedChange element to the track it is defined in.
Additionally I propose that a track must be fully covered by speedChanges, unless an infraAttrGroupRef is defined.

Wiki should also describe how to represent the end of a speedChange: with @vMax=999 (for railML v2.2) or @vMax=end

Description of @dir value range on wiki.railml.org is incorrect. Only up and down are valid according to schema, and that is what the wiki should say too.

Static speed
Is defined with the @pos, @dir, and @vMax values.

Train category dependent speed
Is defined with the @pos, @dir, @vMax, and @etcsTrainCategory values. Setting @etcsTrainCategory defines that the speedChange in fact is a train category dependent speed.

Axle load dependent speed
Is defined with the @pos, @dir, @vMax, and @profileRef values.
To make a speedChange dependent on axle load, a speedProfile element (that specifies @maxAxleLoad) is referenced with the @profileRef value.

5. speedProfile
Provides a way to define certain criteria (such as @maxAxleLoad of train) that need to be met for a referencing speedChange or speed to be valid.

Additionally @influence must be set to tell which speedChange+speedProfile to give precedence when several are valid for a specific train.

Wiki should describe precedence among multiple speedProfiles using @influence

---

Subject: Re: How to model ETCS BL2 speed restrictions and gradients, in railML v2.x
Posted by christian.rahmig on Tue, 08 Jan 2019 20:00:55 GMT

View Forum Message <> Reply to Message

Dear Jörgen,

happy new year 2019! Sorry for replying on your post that late.

Am 07.12.2018 um 09:33 schrieb Jörgen Strandberg:

> [...]
>
> 1. gradientChange
> This seems to be the only way to enter slope of track.
>
> Description of @dir value range on wiki.railml.org is
> incorrect. Only up and down are valid according to schema,
> and that is what the wiki should say too.

You are right, this is a mistake in the wiki. Since this wiki page uses many templates, changing it is quite complex, but I will trigger the changing process.

> For data to be useful for ETCS the slope of each part of the
> track must be known.
> I question whether both these approaches should be
> supported:
> 1. Deduce from connected tracks. The last gradientChanges in
> routes through the network ending up in the track in
> question would specify the slope. A gradientChange would
> then only be defined when contradicting values are deduced
> from the different routes.
> 2. Always define a gradientChange at pos=0 of a track, even
> if the value is unchanged compared to the connected tracks.
> Or in other words limit the effect of a gradientChange
> element to the track it is defined in.
>
> Approach 1. (above) would require processing of both readers
> and writers of the railML data. I propose that approach 2.
> (above) is promoted to be the only valid approach.

Both approaches have their advantages and disadvantages. However, it should be used consistently in import and export interfaces. So, we need a clarification based on "best practices" here. This, for sure, has substantial effects on railML modelling, because the current railML version leaves it up to the user how to use the <*Change> elements.

@all: Do you prefer approach 1 or approach 2?

> @slope is given in relation to @pos and @dir so that: if
> standing at @pos, looking along the track in the direction
> of @dir, a positive value of @slope will be considered as
> seeing that the track continues uphill.
> The slope in down direction should for ETCS purposes be
> allowed to have a different value than in up direction.
> But if the slope value in down direction is omitted, then
> the inverse value of the slope in up direction is assumed

> (as slope value in the down direction).

Using the @dir attribute allows to implement direction dependent gradient profiles. The slope value (positive or negative) shall be interpreted like you mentioned it: in the orientation defined by the @dir attribute.

> [...]
>
> 3. track.infraAttrGroupRefs
> Provides a way to reference a number of predefined speed
> elements that make up the default values for a whole track.
> These defaults are then possible to override with
> speedChanges.
> When an infraAttrGroupRef is defined all speedChanges
> stating that data can be removed. A speedChange that is not
> immediately followed by another speedChange needs to be
> terminated with an additional speedChange that sets
> @vMax=end (@vMax=999), and then the default values of the
> track shall be valid.

Sorry, but I don't understand your approach. Maybe an example can provide clarity?

> The type of element to reference with
> track.infraAttrGroupRefs is incorrectly documented in railML
> 2.2, however there is a Key/KeyRef definition that correctly
> describes the reference, which should make this a valid
> construction.

Yes, this issue has been solved with version 2.4 (see Trac ticket #233 [1]).

> 4. speedChange
> Similar to gradientChange, whether or not to deduce values
> from speedChanges of connected tracks, is a relevant
> question.
> And I propose to limit the effect of a speedChange element
> to the track it is defined in.
> Additionally I propose that a track must be fully covered by
> speedChanges, unless an infraAttrGroupRef is defined.

Like with the <gradientChange> issue above, we should come to a unique and consistent solution. Therefore, I would like to ask the community again: which approach do you prefer?
* option 1: track features (gradients, speeds, ...) are valid even beyond the end of track
* option 2: track features are only valid within the range of the track; every track needs to have appropriate <*Change> elements at the begin

and the end.

> Wiki should also describe how to represent the end of a
> speedChange: with @vMax=999 (for railML v2.2) or @vMax=end

Thank you for pointing on this missing documentation. I added a small remark on the wiki page [2].

> Description of @dir value range on wiki.railml.org is
> incorrect. Only up and down are valid according to schema,
> and that is what the wiki should say too.

See my answer for the @dir attribute of the <gradientChange> element above: this has to be corrected in the wiki.

> Static speed
> Is defined with the @pos, @dir, and @vMax values.
>
> Train category dependent speed
> Is defined with the @pos, @dir, @vMax, and
> @etcsTrainCategory values. Setting @etcsTrainCategory
> defines that the speedChange in fact is a train category
> dependent speed.
>
> Axle load dependent speed
> Is defined with the @pos, @dir, @vMax, and @profileRef
> values.
> To make a speedChange dependent on axle load, a speedProfile
> element (that specifies @maxAxleLoad) is referenced with the
> @profileRef value.

Indeed, modelling speed changes that depend on certain (train) criteria is not done consistent here. A possible conclusion from this situation may be to move the attribute @etcsTrainCategory from the <speedChange> element to the <speedProfile> element. Upcoming railML 3.x will take care of this issue.

> 5. speedProfile
> Provides a way to define certain criteria (such as
> @maxAxleLoad of train) that need to be met for a referencing
> speedChange or speed to be valid.
>
> Additionally @influence must be set to tell which
> speedChange+speedProfile to give precedence when several are
> valid for a specific train.
> Wiki should describe precedence among multiple speedProfiles
> using @influence

The railML wiki page about <speedProfile> could be more exhaustive in terms of best practices. Here, I hope for some input from the community: If you have a nice example with overlaying speed profiles (increasing and decreasing), let's bring it into the wiki together with some graphical representation of the resulting speed profiles along the track.

[1] https://trac.railml.org/ticket/233
[2] https://wiki.railml.org/index.php?title=IS:speedChange

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: How to model ETCS BL2 speed restrictions and gradients, in railML v2.x

Posted by Thomas Nygreen JBD on Tue, 08 Jan 2019 21:07:50 GMT

View Forum Message <> Reply to Message

Quote:
> For data to be useful for ETCS the slope of each part of the
> track must be known.
> I question whether both these approaches should be
> supported:
> 1.   Deduce from connected tracks. The last gradientChanges in
> routes through the network ending up in the track in
> question would specify the slope. A gradientChange would
> then only be defined when contradicting values are deduced
> from the different routes.
> 2.   Always define a gradientChange at pos=0 of a track, even
> if the value is unchanged compared to the connected tracks.
> Or in other words limit the effect of a gradientChange
> element to the track it is defined in.
>
> Approach 1. (above) would require processing of both readers
> and writers of the railML data. I propose that approach 2.
> (above) is promoted to be the only valid approach.

Both approaches have their advantages and disadvantages. However, it should be used consistently in import and export interfaces. So, we need a clarification based on "best practices" here. This, for sure, has substantial effects on railML modelling, because the current railML version leaves it up to the user how to use the <*Change> elements.

@all: Do you prefer approach 1 or approach 2?

Approach 1 was chosen for railML2.4nor, so it is the preference of the Norwegian sector.

Quote:
> @slope is given in relation to @pos and @dir so that: if
> standing at @pos, looking along the track in the direction
> of @dir, a positive value of @slope will be considered as
> seeing that the track continues uphill.
> The slope in down direction should for ETCS purposes be
> allowed to have a different value than in up direction.
> But if the slope value in down direction is omitted, then
> the inverse value of the slope in up direction is assumed
> (as slope value in the down direction).

Using the @dir attribute allows to implement direction dependent
gradient profiles. The slope value (positive or negative) shall be
interpreted like you mentioned it: in the orientation defined by the
@dir attribute.

This would contradict the current definitions, which is that the value is in track direction. This is not documented on the wiki page for gradientChange, but it is clearly stated for radiusChange: "the parameter radius describes the new value of the radius that is valid from there into the direction of track orientation".

If you do not have different gradient profiles for the two directions, you should not use @dir. The way that the documentation defines @dir, having only gradientChanges with @dir="up" would mean that you do not know anything about the gradients in the "down" direction. There is a separate thread on what we should do with @dir.

Quote:
> 4. speedChange
> Similar to gradientChange, whether or not to deduce values
> from speedChanges of connected tracks, is a relevant
> question.
> And I propose to limit the effect of a speedChange element
> to the track it is defined in.
> Additionally I propose that a track must be fully covered by
> speedChanges, unless an infraAttrGroupRef is defined.

Like with the <gradientChange> issue above, we should come to a unique
and consistent solution. Therefore, I would like to ask the community
again: which approach do you prefer?
* option 1: track features (gradients, speeds, ...) are valid even
beyond the end of track
* option 2: track features are only valid within the range of the track;
every track needs to have appropriate <*Change> elements at the begin

and the end.

Again, approach 1 was chosen for railML2.4nor, so it is the preference of the Norwegian sector.

Subject: Re: How to model ETCS BL2 speed restrictions and gradients, in railML v2.x
Posted by christian.rahmig on Fri, 11 Jan 2019 10:22:04 GMT
View Forum Message <> Reply to Message

Dear all,

Am 08.01.2019 um 21:00 schrieb Christian Rahmig:
> Am 07.12.2018 um 09:33 schrieb Jörgen Strandberg:
>> [...]
>>
>> 1. gradientChange
>> This seems to be the only way to enter slope of track.
>>
>> Description of @dir value range on wiki.railml.org is
>> incorrect. Only up and down are valid according to schema,
>> and that is what the wiki should say too.
>
> You are right, this is a mistake in the wiki. Since this wiki page uses
> many templates, changing it is quite complex, but I will trigger the
> changing process.

I filed a Trac ticket for this issue (see [1]). In this ticket, I also
listed all the other elements, where this problem of wrong @dir
enumeration values occurs.

[1] https://trac.railml.org/ticket/352

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