Subject: Re-factoring of <infraAttributes>
Posted by christian.rahmig on Mon, 18 Jun 2018 10:58:28 GMT
View Forum Message <> Reply to Message

Dear railML community,

railML 2.4 is approaching, but we still have many elements and attributes in our model that need some revision. For example: <infraAttributes> and their child elements...

<infraAttributes><owner>
The attribute @uic-no can be marked DEPRECATED, because the information is modelled as "companyCode" in codelist infrastructureManagers.xml.

<infraAttributes><operationMode>
Both parameters, @modeLegislative and @modeExecutive, are simple text strings and not documented in the wiki and the schema. My question to you: do you use these attributes? If yes, what do you model with these attributes?
The attribute @clearanceManaging lists the following values: "sight", "time", "blocking", "LZB-blocking" and "absBrakeDist". If needed at all, I suggest to mark at least "LZB-blocking" as DEPRECATED. Any comments?

<infraAttributes><trainProtection>
Currently, the train protection system is characterized by the attributes @monitoring (enumeration value list: "none", "intermittent", "continuous") and @medium (enumeration value list: "mechanical", "electric", "inductive", ...). How about adding an attribute @trainProtectionSystem (string) to reference the matching train protection system from codelist TrainProtectionSystems.xml. This is already possible for the single <trainProtectionElement>.

<infraAttributes><powerTransmission>
The attribute @type is used to distinguish between conventional railways and rack railways (de: Zahnradbahn) or others. The usage of the other attribute @style (string) is completely unclear. If nobody has any intelligent idea, I suggest to mark this attribute DEPRECATED. Btw: the <powerTransmissionChange> element has also an attribute @style, which is as myterious as the one in <infraAttributes><powerTransmission>.

<infraAttributes><speed>
What does the attribute @status stand for? Yes, this attribute appears also in <speedChange>, but remains mysterious there, too.

<infraAttributes><epsgCode>
The purpose of this element is to name the EPSG codes of the positioning systems that are used throughout the referencing infrastructure. In the <geoCoord> element we used attributes @epsgCode and @heightEpsgCode.

This differs from <infraAttributes><epsgCode> where the attributes are named @default and @extraHeight. I suggest to synchronize the naming using @epsgCode and @heightEpsgCode. Any objections?

<infraAttributes><generalInfraAttribute>
The documentation of this element in the wiki [1] is completely empty. Well, if there is no use case for the <generalInfraAttribute>, I suggest to remove it aka mark it DEPRECATED.

As usual, any kind of comment is highly appreciated...

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: Re-factoring of <infraAttributes>
Posted by christian.rahmig on Mon, 18 Jun 2018 12:17:44 GMT
View Forum Message <> Reply to Message

Dear all,

another remark about the <infraAttributes>:

Long time ago, we said that <infraAttributes> shall not be used, but that <*change> elements (e.g. <speedChange>) shall be used instead. However, <infraAttributes> have some advantages. In particular, <infraAttributes> can be used to model "global" parameters. This makes sense e.g. for specifying the track gauge of a whole railway network.

Therefore, we slightly adapted the documentation in the wiki [1] to conclude: please use <infraAttributes> for global parameters and use <*change> elements for all changes (and local definitions) of parameters.

Please let us know whether you have a different opinion on that issue or if you agree with that concept.

[1] https://wiki.railml.org/index.php?title=IS:infraAttributes

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: Re-factoring of <infraAttributes>
Posted by on Mon, 03 Sep 2018 15:49:37 GMT
View Forum Message <> Reply to Message

Dear Christian,

here are some short remarks on your enumeration from our side. I have no general objections nor problems. Items of your list which I did not mention here, I agree with your suggestion or we do not use and not need it.

> The attribute @clearanceManaging lists the following values: "sight", "time", "blocking", "LZB-blocking" and "absBrakeDist". If needed at all, I suggest to mark at least "LZB-blocking" as DEPRECATED. Any comments?

We have an attribute like this in our software but we do currently not use it in railML. We use that attribute to control the "spacing" of trains in our timetable construction: When can one train follow the previous one? Is the distance (in time or space) between two trains ok or is it a conflict? How much time buffer is in the timetable?

- For tramways, we use "spacing by sight": They can follow each other in any possible distance up to the (absolute) braking distance but they cannot overtake each other while driving at the same line.

- For streets/buses (yes, sometimes there are even graphic timetables for "street traffic", connected with replacement buses), we use "no spacing": They can follow each other in any possible distance and even overtake each other while driving at the same line.

- For normal railway operation, we use "spacing by room/block": The blocking sections of a line decide about the spacing of trains.

- We do not support "spacing by time" since, afaik, there is no such railway line any-more.

"Spacing by sight" is functionally the same as "spacing in absolute braking distance". The difference is only who calculates the braking distance (the driver in his mind or any technology). We do not support "spacing in relative braking distance" despite this happens sometimes at our streets... ;-)

Concerning "spacing by room/block", we do not need to distinguish the kind of block (Main signals, "LZB-blocking", ETCS or such) here because this depends on the actual infrastructure alongside the certain tracks.

Conclusion: For railML, I would still see "spacing by sight" and "spacing by room/block" at least

because there are such railways. It would be nice to have a representation for street traffic (buses) in case someone needs to encode this for timetables of replacement buses. Or may be railML wants to allow even regular buses?

> <infraAttributes><trainProtection>
> Currently, the train protection system is characterized by the attributes @monitoring (enumeration value list: "none", "intermittent", "continuous") and @medium (enumeration value list: "mechanical", "electric", "inductive", ...). How about adding an attribute @trainProtectionSystem (string) to reference the matching train protection system from codelist TrainProtectionSystems.xml. This is already possible for the single <trainProtectionElement>.

This would be redundant since any actual train protection system (model) fits into one of monitoring and medium types. I guess there is no need to describe the generalised kind of train protection. So since we have a list of certain models, to reduce redundancy and effort, I would recommend to declare @monitoring and @medium as deprecated and/or skip them in railML 3.x.

> <infraAttributes><powerTransmission>
> The attribute @type is used to distinguish between conventional railways and rack railways (de: Zahnradbahn) or others. The usage of the other attribute @style (string) is completely unclear.

I guess the type shall describe the rack type in case it is a rack railway (like 'Strub', 'Riggenbach', 'Abt 3 Lamellen', Abt 2 Lamellen' etc.). So the question is: Attributes like gauge, electrification and rack type help to allow only fitting vehicles to fitting tracks. If somebody sends me a <railML>.<infrastructure> file with a railway line of 1000 mm gauge, I want to prevent myself from creating timetables with 1435 mm vehicles on it. Do I want to prevent myself from calculating an 'Abt' rack engine at a 'Riggenbach' rack railway? I am not sure but I tend to answer: Yes, for the sake of consequence.
-> Can we offer an enumeration list of rack types here instead of the string attribute? (There is only a small number of!)
-> By the way: Do we have the same types of racks at <rollingStock>.<vehicles>?

Best regards,
Dirk.

Subject: Re: Re-factoring of <infraAttributes>
Posted by on Mon, 03 Sep 2018 16:15:27 GMT
View Forum Message <> Reply to Message

Dear Christian,

in general, I agree with your statement.

But concerning

> please use <infraAttributes> for global parameters and use <*change> elements for all changes

(and local definitions) of parameters

I want to write: Careful please! This may create much additional effort for importing software.

Please always be aware that an importing software has to "search" and "check" all places where an attribute may be given: There may be implicit default values, global values (global for one railML file) and <*change> elements.

- An importing software may be assuming that a track is straight and level as long as there is no <gradientChange> or <radiusChange> given.

- But, in case there are global values for gradients and curves, does that mean that the global value applies at each <track> until the first <*change> applies or does it mean that the global value only applies if the <track> has no individual <*change> elements?

- Is a <track> which has no <electrificationChange> a non-electrified track or a track which is electrified with the global value? I guess the latter. So, for a network which has all-electrified tracks except one line, is it still possible to use the global value (of 'electrified') and overwrite it at the one <track> with 'non-electrified'?

I agree with you that it is "appetising" to make the gauge global for most of the tracks of a network.
But please, when allowing this

- Do not allow it in existing railML versions (like 2.0, 2.2, 2.3 etc.). Do only allow it in future versions. We should avoid that the same railML file suddenly means something different.

- Always define (in Wiki or XSD) weather and where a global value applies (before the first <*change> or in case there are no <*change>s at all).

- Always ask yourself weather it is possible to overwrite the global value with all other possible values (not to "hide" an implicit default value).

- Do not allow global values for all <*change> elements across-the-board. A global value for gauge may be nice to have but in my opinion, we do not need global values for radius or gradient (to overwrite 'level' and 'straight').

With best regards,
Dirk.