Subject: [railML3] Handling changes between minor versions
Posted by Thomas Nygreen on Tue, 16 Jun 2020 08:20:08 GMT
View Forum Message <> Reply to Message

Dear community!

As announced in this month's news article on railML.org the coordinators have been discussing compatibility between minor versions of railML 3.x. On the one hand, full compatibility results in very little flexibility to correct mistakes or improve concepts that have already been modelled in railML. On the other hand, full flexibility to make changes means we cannot guarantee compatibility between minor versions. In railML 2.x the policy has been and will continue to be that we allow new elements and attributes, but keep downward compatibility by deprecating unwanted elements and attributes instead of removing them.

railML 3.x is even more comprehensive than, and has been completely remodeled from, railML 2.x. We can expect the increased practical application of railML 3.x to result in proposals for changes that will further improve the standard. That leaves us with the question of how we should balance flexibility to improve versus the need for stability and compatibility. The coordinators are proposing the following three alternatives:

Follow the path of railML 2.x in railML 3.x: downward compatibility will be guaranteed. Changes may lead to elements or attributes being deprecated, but not removed.
Changes may lead to elements or attributes being deprecated. Elements and attributes that are deprecated in one minor version will be removed in the following minor version. Compatibility is only guaranteed between one minor version and the next.
Changes may lead to elements or attributes being removed in a new minor version, without first being deprecated. No compatibility guaranteed.

Please post your feedback here before August 31st to allow us to include it in the development of railML 3.2, scheduled for beta release around the end of 2020.

Subject: Re: [railML3] Handling changes between minor versions
Posted by on Thu, 13 Aug 2020 13:41:59 GMT
View Forum Message <> Reply to Message

Hello,

> The coordinators are proposing the > following three alternatives:> > Follow the path of railML 2.x in
railML 3.x: downward> compatibility will be guaranteed. Changes may lead
to> elements or attributes being deprecated, but not removed.> Changes
may lead to elements or attributes being> deprecated. Elements and
attributes that are deprecated in> one minor version will be removed in
the following minor> version. Compatibility is only guaranteed between
one minor> version and the next.> Changes may lead to elements or
attributes being removed in> a new minor version, without first being
deprecated. No> compatibility guaranteed.

We would prefer option 3 (give up compatibility between minor versions).
The current compatibility rule often stands in the way of further
development of the schema and, in our view, has little advantage when
implementing the interface software.
Furthermore, backward compatibility does not only apply to the removal
of attributes / elements: Adding a new attribute can also change the
semantics of other existing attributes, for example, by semantically
overwriting or negating the contents of other attributes. For this
reason, it is not always predictable anyway whether a change will break
downward compatibility or not.
However, different revisions of an (already released) version should
remain backwards compatible.

Best regards
Christian Rößiger
--
iRFP e. K. · Institut für Regional- und Fernverkehrsplanung
Hochschulstr. 45, 01069 Dresden
Tel. +49 351 4706819 · Fax. +49 351 4768190 · www.irfp.de
Registergericht: Amtsgericht Dresden, HRA 9347

Subject: Re: [railML3] Handling changes between minor versions
Posted by Thomas Nygreen on Wed, 19 Aug 2020 14:17:05 GMT
View Forum Message <> Reply to Message

Hello and thank you for the feedback!

>> The coordinators are proposing the following three alternatives:
>> 1: Follow the path of railML 2.x in railML 3.x: downward compatibility
>> will be guaranteed. Changes may lead to elements or attributes being
>> deprecated, but not removed.
>> 2: Changes may lead to elements or attributes being deprecated.
>> Elements and attributes that are deprecated in one minor version will
>> be removed in the following minor version. Compatibility is only
>> guaranteed between one minor version and the next.
>> 3: Changes may lead to elements or attributes being removed in a new
>> minor version, without first being deprecated. No> compatibility
>> guaranteed.

> We would prefer option 3 (give up compatibility between minor versions).
> The current compatibility rule often stands in the way of further
> development of the schema and, in our view, has little advantage when
> implementing the interface software.

This is interesting feedback, and it confirms my own suspicions and
experience.

> Furthermore, backward compatibility does not only apply to the removal
> of attributes / elements: Adding a new attribute can also change the
> semantics of other existing attributes, for example, by semantically
> overwriting or negating the contents of other attributes. For this
> reason, it is not always predictable anyway whether a change will break
> downward compatibility or not.

It is possible to achieve backward compatibility in these cases, but only if you keep using the old attributes in exactly the same way as before. So even if a new, better modelling is introduced, you will have to populate all the old elements and attributes as before. It will be compatible in the sense that when you remove the new attributes and elements from the file, it will look like it did with a previos version. But the result of importing that stripped file will most likely generate a result that does not look like the full model. In many cases there will be business rules to fill inn the missing information, and these will probably not match the actual data 100 %.

> However, different revisions of an (already released) version should
> remain backwards compatible.

So far, we have not had updates to already released versions. What kind of revisions are you thinking about?

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

Subject: Re: [railML3] Handling changes between minor versions
Posted by on Wed, 26 Aug 2020 10:30:30 GMT
View Forum Message <> Reply to Message

> So far, we have not had updates to already released versions. What kind of revisions are you thinking about?

Thomas, Christian refers to "Corrigendum" [1], Unofficial & Developers Versions [2] and "unstable" Schemas [3].

Dirk.

[1] https://wiki2.railml.org/wiki/Dev:versions#Corrigendum
[2] https://wiki2.railml.org/wiki/Dev:versions#Unofficial.2FDeve lopers_Version
[3] https://wiki2.railml.org/wiki/Dev:versioning#Exchange_a_rail ML_file_based_on_unstable_schemas

---