Subject: [railML3] missing @ruleCode?
Posted by Torben Brand on Tue, 25 Feb 2020 10:37:26 GMT
View Forum Message <> Reply to Message

It seems that the very practical attribute @ruleCode (in railML2) describing the type of signal in a national specific unique way is missing in railML3?

The type can be defined in railML2 (by @type and @function) and in a more detailed way in railML3 (by the used sub elements) in a railML generic way. But the national specific way needs also to be modelled. Both from an asset perspective (a home signal looks different in Germany than in Norway) and in an interlocking perspective. As the rules for a home signal can be different.

Subject: Re: [railML3] missing @ruleCode?
Posted by Jörg von Lingen on Fri, 28 Feb 2020 05:33:28 GMT
View Forum Message <> Reply to Message

Hi,

from the brief description in wiki I am not sure about the use of @ruleCode. As it seems to be more functional related I
would see it more in SignalIL than SignalIS. In railML2 the attribute is just a plain string. However, the question is,
whether each signal will have its individual code or there is a limited list for an IM to choose from?

Can you explain your using of the attribute a little bit more?

Regards,
Jörg von Lingen - Interlocking Coordinator
Torben Brand wrote on 25.02.2020 11:37:
> It seems that the very practical attribute @ruleCode (in
> railML2) describing the type of signal in a national
> specific unique way is missing in railML3?
>
> The type can be defined in railML2 (by @type and @function)
> and in a more detailed way in railML3 (by the used sub
> elements) in a railML generic way. But the national specific
> way needs also to be modelled. Both from an asset
> perspective (a home signal looks different in Germany than
> in Norway) and in an interlocking perspective. As the rules
> for a home signal can be different.
>

Subject: Re: [railML3] missing @ruleCode?
Posted by christian.rahmig on Fri, 28 Feb 2020 15:12:32 GMT

View Forum Message <> Reply to Message

Dear Torben,

I agree with Jörg: the content of railML 2 attribute @ruleCode seems to be better located in the interlocking domain of the signal since it refers to aspects the signal can show.

From my understanding, the information should be splitted in two aspects: the name/reference of the rule book (e.g. "Ril301" for DB's "Signalbuch") and the rule code of the signal in the referenced rule book (e.g. "SH2"). So, the structure of this information is very similar to the <designator> element, with the difference that a rule book is being referenced instead of an register and the entry is named rule code.

Best regards
Christian

---

Subject: Re: [railML3] missing @ruleCode?
Posted by Jörg von Lingen on Sat, 29 Feb 2020 04:10:56 GMT
View Forum Message <> Reply to Message

Just a remark.
If you speaking about the "coded name" of a signal aspect, the it shall be placed in the definition of the <hasAspect>
for the <specificIM>.

Regards,
Jörg von Lingen - Interlocking Coordinator
Christian Rahmig wrote on 28.02.2020 16:12:
> I agree with Jörg: the content of railML 2 attribute
> @ruleCode seems to be better located in the interlocking
> domain of the signal since it refers to aspects the signal
> can show.

---

Subject: Re: [railML3] missing @ruleCode?
Posted by Thomas Nygreen on Wed, 04 Mar 2020 18:35:45 GMT
View Forum Message <> Reply to Message

Hi,

@ruleCode originatedas a way to include the national code of a stop post (and then more generally any signal). The given examples seem to me to be more about identifying the exact type of signal (e.g. to be able to look up its physical appearance), rather than its aspects. For boards there is little difference, as they have only one aspect. To provide an example, there are multiple light signal types (in any country) capable of showing the aspects "closed", "proceed" and "limitedProceed". In railML3 we can separate between some of these using <signalConstruction>@type and <signalIL>@function, but there may still be more than one type

matching the @type and @function. How do you specify which specific type of signal it is? Another example is boards, where IL does not provide much, but where signalbooks usually have specific codes for each kind of board.

Best regards,
Thomas

---

Subject: Re: [railML3] missing @ruleCode?
Posted by Dominik Looser on Wed, 25 Mar 2020 10:01:53 GMT
View Forum Message <> Reply to Message

Dear all,

As this is my first forum post, it seems to be common here to introduce myself first. I work for trafit solutions gmbh in Switzerland and together with Jernbanedirektoratet we are working on railOscope/NRV with a railML-interface.

I also propose to include the @rulecode attribute in railML3 as the information we store there in railML2.4 cannot be placed in any other attribute.
The rulecode does not refer to a particular aspect of a signal, but only implies which aspects that signal has. Therefore, putting the rulecode into the aspects would only make sense for boards, where each board can only show one aspect.

I could also imagine Christian's idea to work: using a "designator-like" format for the rulecode. Since rulecodes already have a format like "NOR:TJN:§9-28", this could easily be split into a "register"(or rulebook) and an "entry". Designators now are used to store unique object-specific identifiers, while for rulecodes we need something not unique.

I see three possibilities now:
- A new @rulecode attribute is introduced in a future railML3 version.
- A new "designator-like" element is introduced with attributes @rulebook and @entry. This could be called <typeDesignator>.
- We use the existing <designator> element for "type designators" as well. Using several designators is already supported.

Best regards,
Dominik

---

Subject: Re: [railML3] missing @ruleCode?
Posted by christian.rahmig on Sat, 19 Feb 2022 22:48:28 GMT
View Forum Message <> Reply to Message

Dear railML community,

in order to close this issue with upcoming railML 3.2, I want to ask you:

Which of the three options raised by Dominik do you prefer?

Dominik Looser wrote on Wed, 25 March 2020 11:01Dear all,
I see three possibilities now:
- A new @rulecode attribute is introduced in a future railML3 version.
- A new "designator-like" element is introduced with attributes @rulebook and @entry. This could be called <typeDesignator>.
- We use the existing <designator> element for "type designators" as well. Using several designators is already supported.

Best regards
Christian

---

Subject: Re: [railML3] missing @ruleCode?
Posted by Torben Brand on Fri, 18 Mar 2022 11:46:36 GMT
View Forum Message <> Reply to Message

Dear railML community,

Jernbanedirektoratet recommends the use of <typeDesignator> for the described UC.
The reason being that an attribute @ruleCode is obvious not powerful enough to describe both the register and the entry in a unique manner without merging the two values. The use of <designator> would water out the principle that <designator> is used to designate individual items. Thus, the use of <typeDesignator> would be best suited to univocally define the type of item. (here signal or board) We also thought about adding attributes like an enumeration list @kind or a boolean attribute @isType in <designator>, but concluded that this might also muddle <designator> breaking the principle we stated earlier.

---

Subject: Re: [railML3] missing @ruleCode?
Posted by christian.rahmig on Fri, 18 Mar 2022 12:26:01 GMT
View Forum Message <> Reply to Message

Dear Torben,

thank you for your feedback. I adapted the Git issue #443 [1] accordingly.
Just one last question: we will implement the <typeDesignator> for all functional infrastructure elements. Or would you like to limit it to some functional infrastructure elements only (which ones?)?

[1] https://development.railml.org/railml/version3/-/issues/443

Best regards
Christian

---

Subject: Re: [railML3] missing @ruleCode?
Posted by Torben Brand on Fri, 18 Mar 2022 13:07:03 GMT
View Forum Message <> Reply to Message

Yes, we at Jernbanedirektoratet agree that it makes sense to implement <typeDesignator> for all functional infrastructure elements and let the user decide where it would be relevant to use them.