Subject: Production interfaces with further limitations
Posted by Stefan de Konink on Wed, 12 Feb 2020 20:00:17 GMT
View Forum Message <> Reply to Message

As a producer of RailML we would like to go for certification of our interface as-is. If we comply with the RailML standard, we expect that a certified consumer would be able to use our data. It seems that this is not true, which surprises us greatly.

For example something trivial like the code attribute on Block. One of the consumers of a production interface stating to support RailML mandates this attribute is filled with a decimal value.

The wiki https://wiki2.railml.org/index.php?title=TT:block seems to be clear on it.

code (introduced with version 2.1): Machine-interpretable string (e.g. an abbreviation) used for identification of the object across exchange partners, usecase specific uniqueness constraints may apply.
Maschineninterpretierbare Zeichenkette (z.B. Abkürzung), die zur Identifizierung des Objekts auch bei Austauschpartnern verwendet wird, wobei spezifische Eindeutigkeitsbeschränkungen gelten können.
This is the short string which is known for identifying the block across the involved staff.

code: xs:string, optional

Could someone elaborate how this consumer could have a certified RailML interface, but not supporting the types defined in the standard?

Subject: Re: Production interfaces with further limitations
Posted by Milan Wölke on Wed, 04 Mar 2020 12:58:25 GMT
View Forum Message <> Reply to Message

Hi Stefan,
sorry for the delayed reply. In general I would agree with your interpretation of the attribute code. It needs to be some kind of string that
describes the block across systems. Since the standard does
not restrict it to decimal, neither should a consumer.

Best regards, Milan

Subject: Re: Production interfaces with further limitations
Posted by on Mon, 09 Mar 2020 09:51:23 GMT
View Forum Message <> Reply to Message

Dear Stefan, Milan and all,

first: I do not want to disagree with Milan's statement.

But I think we must distinguish between railML in general and certain concrete, practical interfaces (instances of railML). railML gives a base standard which can be substantiated by practical interfaces. (From railML 3 at least, they are called "use cases".) Such a substantiation or use case can define more restrictions than railML in general.

For instance, in some cases a train number (value of attribute trainNumber) can be alphanumeric, so can have letters. "2S05" may be a typical British train "number", 32768a may be a German train number of a commercial train in some use cases. But many experts and many use cases linked with passenger information would probably say that a train number always must be pure numerical. So it may of course be possible that for instance, a valid railML file containing a non-numerical train number may be refused for a certain use case.

Since the attribute code is something similar to train number (an external identifier), I can at least imagine the same concerning code. Of course you can always expect the importing software to replace the incompatible identifier by one which respects the own restrictions, but exchanging an identifier would at the same time mean incompatibility. So, such a case is not easy, the discussion on external identifiers in railML is a very long one and life is full of compromises...

Best regards,
Dirk.
