Subject: Change from xml:id to UUID in future?
Posted by on Mon, 08 Dec 2014 09:40:53 GMT
View Forum Message <> Reply to Message

(English below)

Hallo allerseits.

Beim <timetable>-Treffen am 03.12.2014 wurde die Frage aufgeworfen, ob in Zukunft (etwa ab 3.0) anstatt der <xml:id>s ausschließlich “UUID”s zu verwenden wären.

Meiner Meinung nach ist das insbesondere im <timetable>-Subschema nicht sinnvoll möglich. Wir müssen im Allgemeinen davon ausgehen, dass viele Elemente in <timetable> nicht persistent existieren, sondern nur für eine RailML-Datei quasi „dynamisch“ erzeugt werden. Etwa können sich <trainPart>s allein dadurch definieren, dass innerhalb eines Zuglaufs bestimmte Eigenschaften wechseln. Würde ein <trainPart> eine UUID haben, müsste ein schreibendes Programm feststellen können, der Eigenschaftswechsel persistent ist (bereits schon einmal mit einer UUID versehen wurde) oder nicht. Wenn <trainPart> als Objekt im Sinne der RailML-Struktur in der Quell- oder Ziel-Software in dieser Form nicht vorliegt, ist das von den beteiligten Programmen möglicherweise etwas zuviel verlangt.

Das mag bei <infrastructure> etwas anders aussehen, da die meisten <infrastructure>-Elemente wohl eher dauerhaften (quasi: ortsfesten) Charakter haben und daher leichter an dauerhafte UUIDs zu binden sind. Bei <timetable> haben wir es aber zumindest im Detail mit häufig wechselnden, rein semantischen Elementen zu tun, die nicht dauerhaften Charakters sind. Dennoch kann ich mir auch bei <infrastructure> gut vorstellen, dass gewisse Elemente nur für die RailML-Ein- oder -Ausgabe existieren, nicht jedoch im internen Datenmodell – etwa <line>s, die aus <track>s abgeleitet werden oder umgekehrt.

Fazit:
- UUIDs sollten ermöglicht, aber nicht erzwungen werden.
- UUIDs sollten verwendet werden, wenn eine Software ausdrücken möchte, dass sie das Element auch über die Gültigkeit der RailML-Datei hinaus identifizieren kann.
- Nur UUIDs zuzulassen würde provozieren, dass nur lokal und temporär gültige UUIDs erzeugt werden. Das täuscht de facto nicht vorhandene Persistenz vor.

Ich plädiere daher für folgende Regel:
- Eine UUID soll verwendet werden, wenn das (schreibende) Programm diese persistent wiederverwenden kann.
- In allen anderen Fällen darf nur eine xml:id verwendet werden.

In diesem Zusammenhang möchte ich nochmal anregen, dass id’s generell möglichst nicht erzwungen werden (Thema http://www.railml.org/forum/ro/?group=4&offset=0&thr ead=86&id=161).

Dear all,

At the <timetable> meeting at 03.12.2014, the question has been discussed whether in future (may be from 3.0) “UUID”s have to be used exceptionally instead of <xml:id>s.

From my opinion, this is not practicable especially in the <timetable> sub-schema. In general, we have to take into account that many elements in <timetable> do not exist persistently but are created for the railML file only. For example, <trainPart>s may be defined only by changes of several properties in a train’s run. If a <trainPart> would have an UUID then a writing programme would need to be able to detect whether this property change exists persistently (whether an UUID already has been assigned to it). This may be a little bit too much demanded if <trainPart> as an object in the sense of the railML element does not exist in the source or target software.

This may look a little bit different at <infrastructure> since most of the <infrastructure> elements have a more “permanent” character. As such, they are more easily linkable to perma-nent UUIDs. At <timetable> we have by tendency more semantical elements which are of no permanent character. However, I can even imagine that there are some elements at <infrastructure> which are only created for import or export of railML files - but do not exist in the data models of the software involved. For instance, <line>s which are created from <track>s or vice versa.

Conclusion:
- UUIDs shall be allowed but not enforced.
- UUIDs shall be used if a software wants to express that it can identify an element beyond the validity of a railML file.
- To allow only UUIDs provokes that UUIS would have to be created which are valid only locally and temporarily. This ‘fakes’ persistence where there is none.

I plead for the following rule:
- An UUID shall be used if the writing software can handle it persistently and re-use it.
- In all other case an xml:id is to be used only.

In this context I would again like to draw the attention to make id’s optional in general (theme http://www.railml.org/forum/ro/?group=4&offset=0&thr ead=86&id=161).

Best regards,
Dirk.

Subject: Re: Change from xml:id to UUID in future?
Posted by nicolas.gatez on Thu, 05 Mar 2015 07:26:36 GMT
View Forum Message <> Reply to Message

Dear all,

I'm Nicolas Gatez, from the belgian IM Infrabel, and member of the
<infrastructure> <topology>
modelling group.

My opinion is also that the UUID, while strongly recommendable, is not
always the best solution,
especially when dealing with legacy source systems where UUIDs are not
supported or
implemented.

It would lead to run-time/export-time generated UUID, meaning that the
same object would have a
different UUID each time a new extract is created.

so, on top of the "Fake persistence" problem, it would also add a problem
of "fake variability",
where you will never know if you see a different object or the same
object with a changed ID.

However, when designing a new system it is always best to ensure that the
used primary keys
are "as unique as possible", for which UUID seems a sound solution.

So, I reach the same conclusions, UUIDs should be allowed, and even
recommended when
possible, and xml:id (which i see as the legacy system id) could be used
when UUIDs are
impractical.

However, ids still have to be unique within the scope of one file/extract.

Best regards,
Nicolas.

--