Subject: [railML3] How to deal with UUIDs
Posted by Larissa Zhuchyi on Thu, 22 Aug 2024 10:39:38 GMT
View Forum Message <> Reply to Message

Dear all

Following an issue [1] and a poll from the conference [2] in railML 3.3 railML.org is going to create a new attribute for specifying UUIDs and remove the union from the id type.

This is justified because of:
- validation issues;
- programming problems with type unions;
- understanding what user has when having a union in id attribute (is it a UUID or an XML id).

Please let us know if you have anything against adding the new UUID attribute and answer the following questions:

1) Should the references should also be split?
2) Should this be included in railML 3.3 or railML.org waits till railML 3.4?

Proposal: railML.org could only split the id and leave the ref as is.

For context see also [3, 4 5].

[1] https://development.railml.org/railml/version3/-/issues/560
[2] https://www.railml.org/en/event-reader/44th-railml-conferenc
e-rome.html?file=files/download/events/conferences/railml_44
th_Rome/2023-11-08_railML_Nygreen_CommonNewsAndQuestions.pdf &cid=11491
[3] https://www.railml.org/forum/index.php?t=msg&goto=3045
[4] https://www.railml.org/forum/index.php?t=msg&goto=2203
[5] https://www.railml.org/forum/index.php?t=msg&goto=1315

Sincerely,

Subject: Re: [railML3] How to deal with UUIDs
Posted by on Thu, 29 Aug 2024 10:26:11 GMT
View Forum Message <> Reply to Message

Hello everyone,

Firstly, I would like to say that I have not used UUIDs so far and therefore have not thought in detail about how I would integrate them within a railML data exchange. If we need references to data outside the railML file, we use the designator element or equivalent(train number, ...) However, from my experience as a developer of a railML import interface for timetable data, I would say that a distinction between UUID/Xml-Id on the reference side would be more important to me than with the Ids: When I parse a railML file and come to a reference, I need to know whether I have to search for the corresponding Id within the railML file or in some external

repository/database.
I also find it helpful that standard XML parsers can check whether there is an ID for each XML reference in the railML file. If the distinction between UUID/Xml-Id is only implemented in one place (id _or_ reference), this possibility would probably no longer exist.
For these reasons, I would be prefer to explicitly distinguish between UUID/Xml-Id at least at the reference location, or better in both places.

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] How to deal with UUIDs
Posted by David Lichti on Thu, 29 Aug 2024 13:27:59 GMT
View Forum Message <> Reply to Message

I agree with Christian on removing the tUUID type from the unions in tRef and tID. It removes ambiguity from the data.

But I would even question the general introduction of a replacement attribute for tUUID, as I don't see the generic utility of such an identifier.

For references within one export file, the tID type is very well suited.

For references between different exports, the dentifiers must not only be unique, but also stable. For tools that do not store their data in railML format (as is the case for our TPS), it would be necessary to maintain a mapping between the internal object identifiers, and the railML identifiers. In general, this may be an m-to-n relationship, since the two data schemas may have different structures. Maintaining such a mapping does not really seem practical.

We use specific business keys for objects that need common and stable identifiers for references to and from external systems. But these identifiers and their format are usually defined by external entities. Most often, they do not match the tUUID pattern. The designator type is much more suited for these identifiers.

In conclusion, I suggest

1. Removing the tUUID type from the unions in tID and tRef to make the schema less ambiguous.
2. Adding a generic entry UUID to the register code list as a replacement for tUUID.
3. Adding designator elements wherever there is a specific need for external references.

Best regards

David

---

Subject: Re: [railML3] How to deal with UUIDs
Posted by Thomas Nygreen on Wed, 04 Sep 2024 12:27:48 GMT
View Forum Message <> Reply to Message

Dear all,

One of the powerful ideas behind including UUIDs in the railML 3 syntax was the ability to reference objects not present in the railML file. When referencing heavily from one domain (such as TT) into another (such as IS), this simplifies the export and resulting file, by not having to include all the referenced elements every time you export. It also facilitates sending only the parts of a previous export that have been changed, without including unchanged objects that are referenced. Obviously, these references cannot be formally validated by any XML parser. Instead, it is implied that both the exporting and the importing software either know how to look up each UUID or can work with it as a black box.

To take advantage of the possibilities offered by UUIDs, the software systems involved must obviously either support UUIDs already or be modified to support them. Systems that do not support UUIDs can still read railML files containing UUIDs, but will not be able to preserve them, and may experience problems reading files where UUIDs are used to reference external entities. To my knowledge, the adoption of UUIDs in the railway sector has been quite limited. I appreciate all feedback from the community on your experiences concerning UUIDs.

It is worth noting that UUIDs were included before <designator>s were applied as widely as now. They both offer stable external references, but they have a key difference: An object has (or is supposed to have) one UUID across all systems that know the object. On the other hand it may have multiple other keys in different registers, each specified by a separate <designator> in railML. This means that a foreign key from a <designator> cannot be used in the same way as a UUID to reference something outside of the current file.

Splitting tID, so that internal XML IDs and UUIDs go into different attributes is an easy change, that will help with some of the practical problems railML users encounter with the current implementation.

Changing tRef is harder. We essentially have three alternatives:

1: The current implementation.

2: Change all (>100) attributes of type tRef to elements offering separate attributes for references of types tGenericRef and tUUID. This keeps the option to provide either a file-internal XML ID or a global UUID, but it explicitly specifies which option you have used.

3: Remove tUUID from tRef. This completely removes the ability to reference objects not present in the file, and one would instead have to include at least a minimal representation containing the XML ID and the external key, either in the form of a @uuid attribute or a <designator>. For systems that use <designator>s and cannot use UUIDs, this is already the only alternative.

I look forward to more input from the community.

Best regards,
Thomas

---