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