Subject: Issue with unique tID in commons3
Posted by mark.lorenz@thalesgroup.c on Wed, 18 Jan 2023 14:27:22 GMT
View Forum Message <> Reply to Message

Hi,

For all of you not knowing me: I am Mark from Thales. Usually, I try to participate at the TT workgroup.

Recently we faced bug here regarding unique IDs:

commons3.xsd defines tID as follows:

```
<xs:simpleType name="tID">
  ...
  <xs:union memberTypes="rail3:tGenericID rail3:tUUID"/>
</xs:simpleType>
```

Means a tID can be either of type tGenericID or tUUID. Internally, tGenericID is correctly classified as "unique" and is fine. tUUID, on the other hand, is not "unique". Thus, depending on the interpretation, tID can be sometimes unique and sometimes not. Depending on which type the value is assigned to.

In our system the JAXB parser seems to work as follows:

- If an ID starts with a letter, it becomes a tGenericID (which is funny, because "a8986caf-b0d1-32ca-b1a9-09f86213da2f" is pretty sure an UUID) and is therefore unique (because upper element xs:ID is unique by definition).
- If an ID starts with a digit, it becomes a tUUID ("68986caf-b0d1-32ca-b1a9-09f86213da2f") and therefore doesn't have to be unique anymore.

Why JAXB works in such a funny way is up to the user. But there is clearly a lack of consistency for the type tID. Ideally, tID should be unique, because otherwise you can't really talk about IDs, can you?

We implemented a quick-and-dirty workaround to satisfy JAXB, but maybe you can look over it to find a proper solution?

Greetings Mark

Subject: Re: Issue with unique tID in commons3
Posted by Thomas Nygreen on Tue, 28 Mar 2023 10:38:21 GMT

Hi all,

I'll try to sum up the email discussion I had with Mark.

It is correct that there is no uniqueness constraint on UUIDs in the railML 3 XSDs. Similarly, the large number of type matching constraints on references that you find in railML 2 are gone in railML 3. Partly, there is a simple practical reason for this: Sparx Enterprise Architect, the modelling tool used for railML 3, simply does not support them. If we decided to include such constraints, we would have to add them to the XSDs after exporting them, or find another (set of) tool(s).

JAXB seems to be mapping the provided IDs to the first pattern that matches, which is not so strange. So, since rail3:tGenericID is the first type to match against, UUIDs that are valid XML IDs will be interpreted as that. The order in the type definition should therefore preferably be
<xs:union memberTypes="rail3:tUUID rail3:tGenericID"/>
This would mean all UUIDs were recognised as such. Unfortunately, I have not found a way to implement this in Sparx Enterprise Architect, as it sorts the types alphabetically.

Finally, even if we had a uniqueness constraint on UUIDs in the XSDs, this constraint would only ensure that UUIDs were not reused within one single file. As an important part of the purpose of UUIDs is to reference elements that are not present in the same file, the writing and reading systems will still have to take care of the proper use and handling of UUIDs across files.

Best regards,
Thomas