Subject: Re: ocp's/stations and their properties
Posted by on Fri, 29 Jun 2012 15:27:36 GMT
View Forum Message <> Reply to Message

Hi Susanne and Christian,

thank you for your suggestion on <ocpGroups>.

> We (Christian and me) hope that the concept of <ocpGroups> help out for
> these cases.

For my understanding of the concept of <ocpGroups>, please check whether the following statements are right:

- An <ocpGroup> cannot be referred to - all other RailML elements refer to <ocps> but never to <ocpGroups>.

- An <ocp> can belong to more than one <ocpGroup> (which means: can be referred by more than one <ocpGroup>).

- An <ocpGroup> can have all the attributes of an <ocp>. There is no attribute of an <ocp> which wouldn’t also be available as an attribute of an <ocpGroup>.

- An attribute defined at an <ocp> can be overwritten by a corresponding <ocpGroup> (which means: by an <ocpGroup> which refers to that <ocp>).

- An attribute defined at an <ocpGroup> is valid for all its <ocps>.

Well, assuming these statements are right, I see one problem with this concept of <ocpGroups>: There is no simple (straight-forward) way to find out whether an <ocp> belongs to an <ocpGroup> and if so to which one.

A typical task for a software reading RailML timetable files is: “Import a train from RailML, referring to the stations by abbreviation or station number”. That means: “Find out the abbreviation or number of a station on the train’s route.”

So, the software takes the <ocpTT>.ocpRef and searches for the <ocp>. It has to find an <ocp> (otherwise it wouldn't be well-conformed RailML). If there is no such abbreviation or number (RailML 2.2: designator) at the <ocp>, it can hope there is an <ocpGroup> which refers to this <ocp> and has the appropriate attribute. So, the software searches through all the <ocpGroups>… And if the <ocp> would have had the desired attribute? Where should the software know that there isn't an <ocpGroup> overwriting that attribute? So, it has always to search through the <ocpGroups>, because there may be… And even if it finds an <ocpGroup> not overwriting the attribute, it has to search further on because there may be another one…

Surely possible, but not very straight-forward from my opinion. Do you think that anybody will implement this algorithm (without much discussion)? ;-)

I would prefer the other way ‘round: An <ocp> refers to it's <ocpGroup> (if there is one). If necessary, an <ocpGroup> can also name it's <ocps> but this would mean redundancy (crossing links). I see less need for the latter way so to avoid crossing links, I would choose the first one.

Another question is: With <ocps> and <ocpGroups> having the same attributes: Why don't we allow an <ocp> to refer to another <ocp>? An <ocp> can act as an ‘direct’ ocp or as an <ocpGroup> (or both).

This would allow more flexibility with less complex XSDs:

- A train could easily refer either to an <ocp> or an <ocpGroup> (because there is no difference between them - both are in the same list). A train referring to an <ocpGroup> reads like: “I want to arrive in Berlin Hbf at 12.00 o’clock. It is all the same to me whether it is “Bft Berlin Hbf oben” or “unten” or “S-Bahn” or whatever. Don't bother me with such railway stuff!”

- It would be possible to create tree structures of <ocps>: Home/starter signals (ESig F) belonging to station parts (Bft Dresden Hbf W9) belonging to stations (Bft Dresden Hbf) belonging to station groups (Bf Dresden). Sometimes we have tram or bus stations having the same station number (IBNR) as the adjacent railway station. With that principle, we could easily solve these ‘problems’.

- On the other hand, it would always be simple for a reading software to find a desired designator: An <ocpTT> refers to an <ocp>. If the <ocp> does not have the desired designator, follow it's parents until one has the desired designator.

So, my suggestion would be:
- to define an optional attribute ‘parentOcpRef’ (=tGenericRef) of an <ocp>.

I would also accept the solution of the two-level concept of <ocpGroups> but there should be a possibility for an <ocp> to refer to it’s parent <ocpGroup>.

Best regards,
Dirk.