Subject: ocp's/stations and their properties
Posted by on Tue, 10 Apr 2012 17:48:01 GMT
View Forum Message <> Reply to Message

Dear Christian and all "along-readers",

I would like to draw your attention to one more "well-known" item I'd like to see solved in RailML 2.2.

Currently, it is not possible that one station has different properties (RailML: <ocp>.<propService>, <propOperational>, <propEquipment> a. s. o..) at different tracks or lines. This is because the properties attributes are only given at the <ocp> but not at the tracks.

From our experience, there are often stations where two or more lines meet and which have different properties depending on the line. First I thought these are seldom special cases but on the contrary - after a closer look you'll find plenty of them.

I think one can easily imagine a station where two lines meet and which has platforms at one line but has no platforms at the other line. So, the attribute "<propService>.passenger" should be 'true' at the tracks of the first line and 'false' at the tracks of the other line.

Also, you'll find often the arrangement where a junction for one line is a head of a station for the other line. (A line branches off at the head of a station but does not pass through the station.) A typical German example is Unterlemnitz (coords 50.469913° 11.624211°, http://www.openstreetmap.org/index.html?mlat=50.469913&m lon=11.624211&zoom=16) which is a junction for line #6683 (the line to the north-east) and a station for line #6709 (the line to the west). There are many of these examples, e. g. Bissenhofen, Niederhone/Eschwege West, Augsburg-Hochzoll, Dresden-Pieschen. Leipzig-Neuwiederitsch, Arnsdorf Nord/West, and many more.

We also will find them in other countries but it soon becomes clear that it depends on the definition of a 'station'. In countries which do not have the railway-operational term 'station' (like US or Canada) there are no such problems. (The term 'station' still refers to the traffic function - a place for accessing the railway - but not the operational function where there are just signal box areas or junctions and nothing more.)

So, for RailML we have two possibilities:
a) either simply to allow track-depending properties of a station because of the plenty examples where this is necessary,
b) or to avoid the term 'station' in an operational sense with all its properties.

a) I think for RailML 2.2 only the first is practicable. This can be done by repeating the properties of a station at the <track>.<trackTopology>.<crossSections> element below the "ocpRef" attribute. This would mean: If there are such attributes at <crossSections>, they overwrite the same attributes of the <ocp> element of "ocpRef". If there are no such properties at <crossSections> all stays as it is and the properties of "ocpRef" are valid.

b) We could possibly follow the (b) solution from RailML 3.0. This would mean to delete all the properties like <propService>, <propOperational>, <propEquipment> of an <ocp>. We should then introduce track elements (in <trackTopology> or <trackElements>) like "beginOcpArea" and "endOcpArea" both with an ocpRef attribute. All which lies in an ocp area belongs to the ocp. It is then up to a reading programme to decide whether it sees the ocp as a station or as a junction or as a halt or whatever. (For example, if there are at least two signals per direction between the station boundaries, it would be a station after the German definition of a station. If there is one signal per direction and at least one point, it is a junction, one signal and no point is a block post a. s. o.)

This would make it very difficult for a reading programme even to do such simple things as finding all passenger stations. Up to now (RailML 2.2) one can find the passenger stations simply by scanning all ocp's for <propService>.passenger=true. If we would delete these "station properties" from RailML 3.0 on, this would become to "scan all tracks for station areas which have at least one platform". So, I think despite all the arbitrariness which lies in the term 'station' and its properties, we should keep them even after RailML 3.0

Conclusion: I herewith make a request for the above described first solution (a) for RailML 2.2.

Best regards,
Dirk.

Subject: Re: ocp's/stations and their properties
Posted by Susanne Wunsch railML on Wed, 27 Jun 2012 20:34:23 GMT
View Forum Message <> Reply to Message

Hi Dirk and others interested,

Dirk Bräuer <dirk.braeuer@irfp.de> writes:

> Currently, it is not possible that one station has different
> properties (RailML: <ocp>.<propService>, <propOperational>,
> <propEquipment> a. s. o.) at different tracks or lines.
>

> From our experience, there are often stations where two or more lines
> meet and which have different properties depending on the line.
>
> I think one can easily imagine a station where two lines meet and
> which has platforms at one line but has no platforms at the other
> line. So, the attribute "<propService>.passenger" should be 'true' at
> the tracks of the first line and 'false' at the tracks of the other
> line.
>
> Also, you'll find often the arrangement where a junction for one line
> is a head of a station for the other line. (A line branches off at the
> head of a station but does not pass through the station.) A typical
> German example is Unterlemnitz (coords 50.469913° 11.624211°,
> http://www.openstreetmap.org/index.html?mlat=50.469913&m lon=11.624211&zoom=16)
> which is a junction for line #6683 (the line to the north-east) and a
> station for line #6709 (the line to the west).

We (Christian and me) hope that the concept of <ocpGroups> help out for
these cases. I try to figure out this as an example code snippet:

```
<tracks>
  <track id="t_1" .../>
  <track id="t_2" .../>
</tracks>
<trackGroups>
  <line id="l_1" code="6709">
    <trackRef ref="t_1"/>
  </line>
  <line id="l_2" code="6683">
    <trackRef ref="t_2"/>
  </line>
</trackGroups>
<operationControlPoints>
  <ocp id="o_1" code="UUTL" name="Unterlemnitz">
    <propService passenger="true"/>
    <propEquipment>
      <trackRef ref="t_1"/>
    </propEquipment>
  </ocp>
  <ocp id="o_2" code="UUTL" name="Unterlemnitz">
    <propService passenger="false"/>
    <propEquipment>
      <trackRef ref="t_2"/>
    </propEquipment>
  </ocp>
</operationControlPoints>
<ocpGroups>
  <ocpGroup id="og_1">
```

```
    <ocpRef ref="o_1"/>
    <ocpRef ref="o_2"/>
  </ocpGroup>
</ocpGroups>
```

The ocpGroup could be defined in several ways. One is the above minimum variant.

The other is a more comprehensive one. It would define some general properties of an ocp (composed of "ocp parts"). The ocps itself override these general properties as shown "o_2".

```
  <ocpGroup id="og_1" code="UUTL" name="Unterlemnitz">
    <propService passenger="true"/>
    <ocpRef ref="o_1"/>
    <ocpRef ref="o_2"/>
  </ocpGroup>
```

Different 'codes' for a <line> are not taken into consideration here, they should be discussed in another thread. [1]

Introducing <ocpGroups> may become a really powerful concept.

Any comments and questions are highly appreciated.

Kind regards...
Susanne

[1] https://trac.assembla.com/railML/ticket/152

--
Susanne Wunsch
Schema Coordinator: railML.common

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.

Subject: Re: ocp's/stations and their properties
Posted by Christian Rahmig on Mon, 02 Jul 2012 05:05:25 GMT
View Forum Message <> Reply to Message

Hello Dirk and anyone interested,

> 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>.

Well, for my understanding it should be possible to refer to an <ocpGroup> as well, because it has the same attributes like an <ocp> element.

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

This is basically possible, but is that a realistic case?

> - 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>.

Yes.

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

I think the concept of overloading attribute values needs to be turned upside-down: The attributes defined in the <ocpGroup> should be overwritten by corresponding attributes from a referenced <ocp>.

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

Considering the above concept, this will be only correct if there is not a corresponding attribute in the <ocp> element itself.

> [...]
> 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).

That is an interesting idea. If this bottom-up-referencing is better applicable for the <ocp> modeling, we should adapt our first idea.

> 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).

The idea of <ocpGroup> was supposed to follow the pattern of <track> and <trackGroup>. However, your question brings it to the point: Do we need an explicit <ocpGroup> or may we reference from one <ocp> to the next/parent <ocp>? What do others think about this question regarding their usage of <ocp>?

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

Thank you, Dirk, for the approach.
Best regards

Christian Rahmig
railML.infrastructure coordinator

Subject: Re: ocp's/stations and their properties
Posted by on Thu, 05 Jul 2012 14:28:14 GMT
View Forum Message <> Reply to Message

Dear Christian,

thank you for your answer.

Your answers to my understanding statements tell me that I have misunderstood the principle of <ocpGroups>. The differences are:

- Trains can refer either to <ocps> or <ocpGroups>.
- Properties of <ocps> override the same properties of an associated <ocpGroup> but not the other way around.

> Well, for my understanding it should be possible to refer to an
> <ocpGroup> as well, because it has the same attributes like an <ocp>
> element.

I have assumed a strongly hierarchical structure, that's why I would not have dared to expect that trains being allowed to refer to <ocps> or <ocpGroups>. As far as I can see, it would be the first time that the target of a 'ref' attribute in RailML cannot be read from the attribute's name. (For example, a 'trainRef' always refers to a 'train', an 'operatingPeriodRef' always refers to an 'operatingPeriod'. But now, an 'ocpRef' could refer to an 'ocp' or an 'ocpGroup'? It then should have the name 'ocpOrOcpGroupRef'? ;-)

Anyway, it is all the same to me if only it works. But I would prefer a simple, clear and straight-forward solution.

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

> However, your question brings it to the point: Do we need an explicit
> <ocpGroup> or may we reference from one <ocp> to the next/parent <ocp>?
> What do others think about this question regarding their usage of <ocp>?

It seams to me that an optional attribute ‘parentOcpRef’ (=tGenericRef) of an <ocp> is the simplest way to allow the desired function: No additional structures, no 'ref's to more than one target list, direct (cascaded) cross-references from a train to it's ocp's, more flexibility.

So, if there comes no objection during the next days or weeks, I would ask you to provide a ticket therefore for 2.2 and add it into the schemes. I would like to try it before final release of RailML 2.2 (as agreed), which we assume to be at InnoTrans, so I do need the XSDs latest in August. Thank you!

With best regards,
Dirk.

Subject: Re: ocp's/stations and their properties
Posted by Christian Rahmig on Sat, 01 Sep 2012 10:12:32 GMT
View Forum Message <> Reply to Message

Hello Dirk & everyone interested,

>>> So, my suggestion would be:
>>> - to define an optional attribute ‘parentOcpRef’ (=tGenericRef) of an
>>> <ocp>.
>
>> However, your question brings it to the point: Do we need an explicit
>> <ocpGroup> or may we reference from one <ocp> to the next/parent
>> <ocp>? What do others think about this question regarding their usage
>> of <ocp>?
>
> It seams to me that an optional attribute ‘parentOcpRef’ (=tGenericRef)
> of an <ocp> is the simplest way to allow the desired function: No
> additional structures, no 'ref's to more than one target list, direct
> (cascaded) cross-references from a train to it's ocp's, more flexibility.

And that is the way, how we will do it with railML 2.2. An <ocp> may refer to its one and only parent <ocp> using the new optional attribute 'parentOcpRef' of type tGenericRef.

This requires the following changes within infrastructureTypes.xsd (c.f. trac ticket [1]):

```
  <xs:complexType name="tOperationControlPoint">
    <xs:complexContent>
      <xs:extension base="rail:tElementWithIDAndName">
        ...
        <xs:attribute name="parentOcpRef" type="rail:tGenericRef">
          <xs:annotation>
```

```
            <xs:documentation>references the one and only parent ocp of
this ocp</xs:documentation>
          </xs:annotation>
        </xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

However, I still think that concerning the overloading of attributes, we should follow the concept that the attributes of a referenced "parentOcp" should be overwritten by corresponding attributes from the referencing <ocp>.

Regards

[1] https://trac.assembla.com/railML/ticket/153

--
Christian Rahmig
railML.infrastructure coordinator

Subject: Re: ocp's/stations and their properties
Posted by on Tue, 02 Oct 2012 16:11:42 GMT
View Forum Message <> Reply to Message

Dear Christian,

> And that is the way, how we will do it with railML 2.2.

Very good, thank you.

> However, I still think that concerning the overloading of attributes, we
> should follow the concept that the attributes of a referenced
> "parentOcp" should be overwritten by corresponding attributes from the
> referencing <ocp>.

I agree. The more 'down' it goes in the hierarchy, the more detailed it is. A parent OCP may have the attribute 'passenger traffic', but that does not necessarily mean that all children have 'passenger traffic'.

(Example: The parent OCP 'Dresden' has passenger traffic. The child OCP 'Bft. Dresden-Altstadt' does not have. So the children overwrite the parents.*)

We should make a clear documentation on the priority of overwritten attributes (parent overwrites child or child overwrites parent). Do you write it or shall I?

Best regards,
Dirk.

* Many children would wish to have this principle in real life...

Subject: Re: ocp's/stations and their properties
Posted by Christian Rahmig on Wed, 03 Oct 2012 07:25:01 GMT
View Forum Message <> Reply to Message

Hello Dirk,

>> However, I still think that concerning the overloading of attributes,
>> we should follow the concept that the attributes of a referenced
>> "parentOcp" should be overwritten by corresponding attributes from the
>> referencing <ocp>.
>
> I agree. The more 'down' it goes in the hierarchy, the more detailed it
> is. A parent OCP may have the attribute 'passenger traffic', but that
> does not necessarily mean that all children have 'passenger traffic'.
>
> (Example: The parent OCP 'Dresden' has passenger traffic. The child OCP
> 'Bft. Dresden-Altstadt' does not have. So the children overwrite the
> parents.*)

thank you for your example, which I am going to include in the wiki notes. For the sake of completeness I want to add, that the principle of overwriting requires a corresponding attribute in the child element. In particular, if the child ocp does not specify the 'passenger traffic' attribute, the value from the parent ocp is taken.

Your example in source code:

```
  <rail:operationControlPoints>
    <rail:ocp id="ocp01" name="Dresden" ...>
      <rail:propService passenger="true" />
    </rail:ocp>
    <rail:ocp id="ocp02" name="Bft. Dresden-Altstadt" ...
parentOcpRef="ocp01">
      <rail:propService passenger="false" />
    </rail:ocp>
  </rail:operationControlPoints>
```

Regards

--
Christian Rahmig
railML.infrastructure coordinator
