Subject: usage of @branchingSpeed and @joiningSpeed
Posted by Torben Brand on Wed, 20 Nov 2024 13:52:32 GMT
View Forum Message <> Reply to Message

It seems we need to more clearly define the usage of @branchingSpeed and @joiningSpeed in element <leftBranch<,<rightBranch>,<straightBranch> and <turningBranch>
The definition of the attributes @branchingSpeed and @joiningSpeed are the same for all four elements:

@branchingSpeed: speed limit for a train diverging relative to the direction of a switch (from blades to a frog)
@joiningSpeed: speed limit for a train converging relative to the direction of a switch (from blades to a frog

Based on best Practice / Examples in the wiki I have seen three different implementations:

A. define speed over the branch. Independent if the <*branch> is defined as branching course or continue course. As in the current wiki example.
B. define the branching speed independent if the <*branch> is defined as branching course or continue course.
C. define the branching speed only on the <*branch> which is defined as branching course

Note the correct wording of the term "branch" as the switch "legs" vs. "branching" as "deflecting/joining", having a speed restriction in comparison to line speed (on continueCourse).

If we choose:
- A the definitions for @branchingSpeed and @joiningSpeed are wrong
- B the example is wrong (se changed example B bellow)
- C the example is wrong (se changed example C bellow; removed rightBranch@branchingSpeed and @joiningSpeed)

I would suggest choosing option C.
Please give feedback here in forum and we will discuss in a upcoming SCTP workgroup meeting.

If we keep the definitions for @branchingSpeed and @joiningSpeed (option C) we should add semantic constraint:
- if trackcontinueCourse=left then do NOT use leftBranch@branchingSpeed and @joiningSpeed
- if trackcontinueCourse=right then do NOT use rightBranch@branchingSpeed and @joiningSpeed

Ps. We should update the example anyway as spotLocation@pos has been deprecated and the attribute @defaultCourse does not exist, its: switchIL@preferredPosition.

Example A:
```
<functionalInfrastructure>
    ...
    <switchesIS>
      <switchIS id="swi01" continueCourse="right" branchCourse="left" type="ordinarySwitch">
```

```
          <name name="68W02" language="en"/>
          <spotLocation id="swi01_sloc01" netElementRef="ne_a03" applicationDirection="reverse"
intrinsicCoord="0.0">
            <linearCoordinate positioningSystemRef="lps01" measure="500.0"/>
          </spotLocation>
          <leftBranch netRelationRef="nr_a02a03" branchingSpeed="60" joiningSpeed="60"
radius="-500"/>
          <rightBranch netRelationRef="nr_a01a03" branchingSpeed="80" joiningSpeed="80"
radius="0"/>
        </switchIS>
```

Example B:

```
<functionalInfrastructure>
      ...
      <switchesIS>
        <switchIS id="swi01" continueCourse="right" branchCourse="left" type="ordinarySwitch">
          <name name="68W02" language="en"/>
          <spotLocation id="swi01_sloc01" netElementRef="ne_a03" applicationDirection="reverse"
intrinsicCoord="0.0">
            <linearCoordinate positioningSystemRef="lps01" measure="500.0"/>
          </spotLocation>
          <leftBranch netRelationRef="nr_a02a03" branchingSpeed="60" joiningSpeed="60"
radius="-500"/>
          <rightBranch netRelationRef="nr_a01a03" branchingSpeed="60" joiningSpeed="60"
radius="0"/>
        </switchIS>
```

Example C:

```
<functionalInfrastructure>
      ...
      <switchesIS>
        <switchIS id="swi01" continueCourse="right" branchCourse="left" type="ordinarySwitch">
          <name name="68W02" language="en"/>
          <spotLocation id="swi01_sloc01" netElementRef="ne_a03" applicationDirection="reverse"
intrinsicCoord="0.0">
            <linearCoordinate positioningSystemRef="lps01" measure="500.0"/>
          </spotLocation>
          <leftBranch netRelationRef="nr_a02a03" branchingSpeed="60" joiningSpeed="60"
radius="-500"/>
          <rightBranch netRelationRef="nr_a01a03" [removed  branchingSpeed="XX"
joiningSpeed="xx"] radius="0"/>
        </switchIS>
```

[1] https://wiki3.railml.org/wiki/IS:switchIS

Subject: Re: usage of @branchingSpeed and @joiningSpeed
Posted by Dominik Looser on Thu, 21 Nov 2024 08:39:14 GMT
View Forum Message <> Reply to Message

Dear all,

I agree that suggestion C makes the most sense.
The same decision should also be made when it comes to double and single slip switches.
The example in the wiki[1] should also be changed for the speed-values, but otherwise makes sense:

- The <switchIS> of type "doubleSwitchCrossing" has no course-attributes.
- The <switchIS> of type "doubleSwitchCrossing" has speeds-attributes only on the two <turningBranch>-subelements
- The <switchIS>es of type "switchCrossingPart" have course-attributes.
- The <switchIS>es of type "switchCrossingPart" should have speeds-attributes only on the deflecting branch-subelements (same rule as for ordinary switches, suggestion C)

Best regards,
Dominik

[1] https://wiki3.railml.org/wiki/Dev:Double_and_single_switch_c rossing

Subject: Re: usage of @branchingSpeed and @joiningSpeed
Posted by christian.rahmig on Mon, 16 Dec 2024 11:31:05 GMT
View Forum Message <> Reply to Message

Dear Torben, dear Dominik,
dear all,

as discussed in the SCTP developer group meeting on Decembre 6. 2024, I propose a better documentation of the attributes @branchingSpeed and @joiningSpeed used in the elements <*branch> of <switchIS> and <crossing>.

branchingSpeed ... describes the maximum allowed speed in km/h for a railway vehicle passing the switch in the direction from switch begin to switch end.

joiningSpeed ... describes the maximum allowed speed in km/h for a railway vehicle passing the switch in the direction from switch end to switch begin.

It is important to understand, that this speed restriction is caused by constructional features and it can be overwritten by more restrictive operational speed constraints e.g. shown by signals ahead of the switch.

Does this clarify the usage of the attributes?

Best regards

Christian

Subject: Re: usage of @branchingSpeed and @joiningSpeed
Posted by Torben Brand on Wed, 15 Jan 2025 12:24:46 GMT
View Forum Message <> Reply to Message

Dear Christian,

You have chosen option A from my options listed in initial post. You are thus changing branchING and joinING speed (deflecting speed) into branch speed with normal and reverse application direction. In my understanding (of the English terms) this is quite a radical change. But we can accept this as long as the definition is unambiguous in its definition. For this requirement, I suggest making it quite clear that the attribute is only valid for the element/branch it is placed on. This might seem redundant, but (I think) important to point out.

The reason I can accept the change in definition (as I see it) is as the attribute is optional, so we can continue to only define the @branchingSpeed for the branches that are branchING (deflecting speed). This would then be the same <XBranch> element as branchCourse="X" where X is either "left" or "right".

We could consider a term cleanup in railML3.4? But it's a minor thing so we could keep the terms as is, as long as the definition is precise.
Old (3.3): --> New suggestion (3.4):
@branchingSpeed --> @branchSpeed
@joinningSpeed --> @branchSpeedReverse
@branchCourse --> @branchingCourse

Further, concerning your definition of "caused by constructional features" I have the following questions:
- Is the value @branchingSpeed & @joiningSpeed "that is caused by constructional features" (always) signalised/communicated to the driver?
- On which UC is this based? Would this also be relevant for bridges or other functionalElements that have a constructive speed limitation? Only the signalised/communicated speeds as the BranchING speed is relevant for RTCI UC.

Subject: Re: usage of @branchingSpeed and @joiningSpeed
Posted by christian.rahmig on Mon, 03 Mar 2025 13:40:47 GMT
View Forum Message <> Reply to Message

Dear Torben, dear all,

let me summarize the latest state of discussion following the SCTP use case working group meeting on February 28, 2025:

Option A: clarify documentation
Existing attributes shall be documented more precisely.
branchingSpeed ... describes the maximum allowed speed in km/h for a railway vehicle passing the switch in the direction from switch begin to switch end.
joiningSpeed ... describes the maximum allowed speed in km/h for a railway vehicle passing the switch in the direction from switch end to switch begin.

Option B: rename attributes
Only applicable for future railML versions 3.4ff
In <switchIS> rename @branchCourse into @branchingCourse (values: left, right).
In <leftBranch> and <rightBranch> rename @branchingSpeed into @branchSpeedFromTip.
In <leftBranch> and <rightBranch> rename @joiningSpeed into @branchSpeedToTip.

Option C: get rid of constructional speed constraints in <switchIS>
Only applicable for future railML versions 3.4ff
The discussion was about whether a speed limitation resulting from constructional parameters of the switch is needed at all. What are the problems: switch branch specific speed constraints cannot be linked from a (speed) signal, because <signalIS/isSpeedSignal> can only refer to <speedSection> elements and not to speeds at switches. Further, it is not possible to link from a switch constructional speed value to a speed profile.
Considering these drawbacks, should we get rid of the speed constraint in the <switchIS> element and only allow for speeds to be modeled with <speedSection> elements?

Dear community, what is your opinion? As usual, any kind of feedback is highly appreciated.

Best regards
Christian

Subject: Re: usage of @branchingSpeed and @joiningSpeed
Posted by Thomas Nygreen on Wed, 19 Mar 2025 14:00:49 GMT
View Forum Message <> Reply to Message

Dear Christian,

I do not have any strong opinion about which option is best, but I would support clarifying the documentation (option A) also if you choose to rename the attributes from 3.4 onwards (option B). So these two are not mutually exclusive. Furthermore, do the attributes describe the allowed speed for "passing the switch" (Torben's example B) or should the wording be "passing the branch" (Torben's examples A and C)?

Best regards
Thomas

Subject: Re: usage of @branchingSpeed and @joiningSpeed
Posted by Stéphane Kaloustian on Tue, 22 Apr 2025 10:22:33 GMT

Dear Christian

On our signalling plans, all speeds are signalled speeds. There may be plans earlier in the process, where construction-related speeds are indicated, but we have no use case where we need to export those construction-related speeds to third-parties. We need to exchange signalling speeds between us (IM) and our signalling suppliers. The use case for exchanging construction-related speeds would be a data exchange between track specialist, vehicle dynamics specialist and (as input) signalling department, but it is not the use case we are pursuing.

So we can't say railML should get rid of constructional speeds; we can only say we don't need them in our use cases at the moment.

In addition, for lineside signalling, we think speedsections will be sufficient and we won't need branchSpeed. In optical signalling, we do use a speed with the meaning of "maximal signalled speed", often depending on switch position, on track portions where no speed profile is defined. It may be convenient to relate them to a branch but one can achieve the same goal by using speedsections.

Kind regards
Stéphane