Subject: Re: [railML3] Semantics of the attributes arrivalDay and departureDay
Posted by Roman Naumann on Tue, 09 Mar 2021 16:47:05 GMT
View Forum Message <> Reply to Message

David Lichti wrote on Mon, 18 January 2021 07:52But appart from identification, there is also the problem that many train line operations do not align with the 00:00-24:00 cycle. Actual operating times may rather resemble 05:00-01:00 or 04:00-02:00, where the latest trains of a given operating day actually run on the next calendar day. Rather than defining custom day cycles (which might differ between different lines in the same file, anyways), such trains could be modeled by having their first departure shifted by one day.

I fully agree with David's remark. In addition, I believe that being less restrictive with the departureDay/arrivalDay offsets does not make the status quo notably more complex, so I do not see anything speaking against the proposed change.

For example, a part of a split train as descripted in Figure 1 in Red here [1] currently (RailML 2) has two valid representations: a shifted operating period with a "negative jump" of departureDay between train parts or, alternatively, a positive departureDay for the beginning of the second train. Likewise, the first example ("route of the train is not completely mapped in the railML file") also explicitly allows for positive departureDays at the beginning of RailML 2 trains.

These corner cases in RailML 2, where we currently allow for positive departureDays at the beginning of trains, show, that any import implementation already has to implement mapping logic from shifted operating periods to calendaric validities with RailML 2.

Last, I would like to add that in example 2, if we consider named operating periods ("Mo-Fr") for timetable exchange, disallowing a positive departureDay at the beginning results in oddities or inconsistencies: We would have to export trains that are planned in the same operating period in the source system with two operating periods ("Mo-Fr" and "Tue-Sat") and shifted bitmasks - except if all trains shortly after midnight are part of a split train where the other part starts before midnight. The forced "splitting" of operating periods seems odd and the exception to it inconsistent (why should the operating period's naming of one train be affected by another, after all?).

This point was even made in the RailML 2 documentation, despite the restriction in place [2]: "Why not use shifted/rotated <operatingPeriod>s" -> "For several <operatingPeriod>s which are often used for passenger information it may be very difficult to find (understandable) text expressions if they would be shifted by one or more days."

So in summary, the "less restrictive" version described by Christian would in my opinion not introduce new complexity, but reduce inconsistencies and improve expressiveness.

[1] https://wiki2.railml.org/wiki/Dev:Examples_for_a_non-zero_operating_day_offset_at_the_first_ocpTT_of_a_train_run
[2] https://wiki2.railml.org/wiki/Dev:Midnight_overrun