Subject: Re: Explizite Kennzeichnung von gelöschten Zügen und Zugausfällen
Posted by christian.wermelinger on Mon, 16 Sep 2013 15:09:36 GMT
View Forum Message <> Reply to Message

Hallo zusammen

Erstmal ein DankeschÃ¶n an Andreas und Dirk fÃ¼r den wertvollen Input.

Andreas Tanner wrote:
> Ich denke, man sollte trennen, ob man
> a) Informationen aus der dispositiven Welt, oder

Andreas' Vorschlag, Informationen aus der dispositiven Welt und PlanÃ¤nderungen konsequent zu trennen, erachte ich als sinnvoll. GrundsÃ¤tzlich muss ich jedoch Dirk beipflichten.

> Bisher war die Philosophie in RailML so, dass eine RailML-Datei nur einen
> aktuellen Datenstand absolut und vollstÃ¤ndig abbildet: Sie bezieht sich
> nicht auf einen frÃ¼heren Datenstand.

Meines Erachtens sollte an dieser Philosophie festgehalten werden.

> Ich mÃ¶chte nur darauf hinweisen, dass wir damit ein kleines FÃ¤sschen
> aufmachen, bei dem sich vielleicht herausstellt, dass es so schnell keinen

werden
> wir sie vmtl. so schnell nicht wieder los.

kleines FÃ¤sschen, sondern ein ziemlich grosses Fass aufmachen wÃ¼rden! Dirk ist ja bereits auf mÃ¶gliche Konsequenzen eingegangen.

> einzufÃ¼hren.

Von einer halbfertigen, nicht zu Ende gedachten LÃ¶sung profitiert schlussendlich niemand. Andererseits fÃ¼hrt der Einsatz zeitabhÃ¤ngiger Attribute zu einer stark erhÃ¶hten KomplexitÃ¤t und blÃ¤ht die Daten weiter auf. Dies bringt insbesondere dann keinen Mehrwert, wenn dieser KomplexitÃ¤tsgrad gar nicht gefordert ist (z.B. weil das Zielsystem gar keine Historie bzw. zeitabhÃ¤ngigen Attribute kennt oder benÃ¶tigt).

> Letztendlich kann man es auch anders aufrollen: Bei DatenÃ¼bertragung in

> stattfinden. Dies kann beim Export _vor_ RailML stattfinden - RailML

frÃ¼heren
> Export - oder es kann beim Import _nach_ RailML stattfinden, d. h. beim

der
> RailML-Daten mit den aktuell im System vorhandenen Daten.
>
> Es scheint, dass beide Wege gleichwertig sind. Die bisherige

> Einlesen.

Wie bereits erwÃ¤hnt, erachte ich es als sinvoll weiterhin an der

Wir handeln dies nun auch so. Beim Import von FahrplÃ¤nen wird der bereits
im System vorhandene Datenbestand mit den zu importierenden RailML-Daten
abgeglichen. FÃ¼r den Datenabgleich bieten wir folgende Import-Optionen an:
1. neue ZÃ¼ge importieren: ja/nein
2. bestehende ZÃ¼ge aktualisieren: ja/nein
3. nicht mehr vorhanden ZÃ¼ge lÃ¶schen: ja/nein

Damit sind fÃ¼r uns zumindest die wichtigsten Use-Cases im Zusammenhang
mit PlanÃ¤nderungen abgedeckt.

Viele GrÃ¼sse,
Christian

Dirk BrÃ¤uer wrote:
>
> Hallo Christian, Andreas und alle Mitlesenden,
>

Allgemeinen ist
das
> eine schon mehrfach andiskutierte Sache in RailML.
>
> Bisher war die Philosophie in RailML so, dass eine RailML-Datei nur einen
> aktuellen Datenstand absolut und vollstÃ¤ndig abbildet: Sie bezieht sich
> nicht auf einen frÃ¼heren Datenstand. Wenn sich Daten geÃ¤ndert haben,
> musste man alles mit RailML neu Ã¼bertragen, auch das, was sich nicht
> geÃ¤ndert hat.
>
> Zum Ausfall gekennzeichnete ZÃ¼ge sind nun eine besondere Form von

formell
> auf einen frÃ¼heren Stand, nÃ¤mlich den, zu dem der Zug noch nicht ausfallen

> sollte. Das ganze wird deutlich, wenn man bedenkt, dass ein Zug mehrfach
> ein- und ausgelegt werden kÃ¶nnte, d. h. erst soll er fahren, dann wieder
> nicht, dann doch usw. Und das in unterschiedlicher Abfolge fÃ¼r jeden
> mÃ¶glichen Verkehrstag des Zuges.
>
> Es ist mir klar, dass man konkret den Ausfall eines Zuges nicht so

> operatingPeriodRef bzw. Verkehrstage-Bitmaske zu ergÃ¤nzen, die aussagt,
> wann der Zug abweichend von der eigentlichen (bisher einzigen)
> operatingPeriodRef _nicht_ verkehrt - also an denen er irgendwann in der
> Vergangenheit mal verkehren sollte und nach aktuellem Erkenntnisstand mal
> jemand entschieden hat, dass er nicht verkehren soll.
>
> Ich mÃ¶chte nur darauf hinweisen, dass wir damit ein kleines FÃ¤sschen
> aufmachen, bei dem sich vielleicht herausstellt, dass es so schnell keinen

werden
> wir sie vmtl. so schnell nicht wieder los. Der Anwender kennt ja den
> Werdegang von RailML nicht und akzeptiert auch nicht unbedingt einen

Element

> jemand mal so geplant hatte.
>

Boden
des
> Fasses, wenn man bedenkt, dass im Falle von ZÃ¼gen eventuell noch wichtig
> sein kÃ¶nnte, auf welcher Planungsebene er ein- und ausgelegt wurde. War er
> schon beim Infrastrukturbetreiber bestellt und muss daher abbestellt

> offiziellen Status erlangte? Und das nach Verkehrstagen unterschieden: Zug
> ist im Juli bis September bestellt worden, im Juli soll er immer noch
> fahren, im August ist er bereits abbestellt, im September ist er noch
> abzubestellen...
>
> - - -
> Letztendlich kann man es auch anders aufrollen: Bei DatenÃ¼bertragung in

> stattfinden. Dies kann beim Export _vor_ RailML stattfinden - RailML

frÃ¼heren
> Export - oder es kann beim Import _nach_ RailML stattfinden, d. h. beim

der
> RailML-Daten mit den aktuell im System vorhandenen Daten.
>
> Es scheint, dass beide Wege gleichwertig sind. Die bisherige

> Einlesen. Das Einlesen ist ohnehin ein komplexerer Prozess als das
> Rausschreiben, wegen der zusÃ¤tzlich notwendigen DatenintegritÃ¤tsprÃ¼fungen.
>
> Zu bedenken ist immer auch, dass u. U. verschiedene Datenquellen in

> hingegen wÃ¼rde sich immer auf die zuvor aus dem eigenen System
> exportierten Daten beziehen, d. h. hier ist im Gesamtprozess kein
> Informationsgewinn mÃ¶glich.
>
> Wir (iRFP) haben uns dieser Lesart angepasst, d. h. wir kÃ¶nnen beim

anbieten.
> Wir wÃ¼rden aus AufwandsgrÃ¼nden in absehbarer Zeit nicht beides anbieten,
> zumal wie gesagt derzeit kein Mehrwert erkennbar ist. Insofern wÃ¼rde ich
> das von unserer Seite erst einmal zurÃ¼ckhaltend betrachten wollen.
>
> Ich verstehe, dass die Situation vielleicht anders aussieht, wenn zwei
> nicht gleichwertige Systeme Daten austauschen - d. h. wenn beim Einlesen
> aus irgendwelchen GrÃ¼nden weniger ProzesskapazitÃ¤t vorliegt als beim
> Rausschreiben. Das wÃ¤re dann aber vielleicht ein Fall, der nicht mehr im
> allgemeingÃ¼ltigen RailML auftauchen muss - eher ein individueller Aufsatz
> oder eine bilaterale LÃ¶sung.
>

> Dirk.
>
>

--
----== posted via PHP Headliner ==----