

Beschreibung der FBS-RailML®-Schnittstelle

Stand: August 2011
RailML-Version: 2.0
Schemeninstanz: 2.0.3
Kompatibilitätsnummer: 1

Inhalt

Allgemeines.....	2
Kopfinformationen (Versionsnummern, Dublin Core Metadata Element Set).....	3
Infrastruktur (<i>infrastructure</i>)	4
Strecken (<i>trackGroups</i>).....	4
Streckengleise (<i>tracks</i>)	4
Kilometrierungsbereiche (<i>mileageChanges</i>)	6
Querverweise auf Betriebsstellen (<i>trackBegin</i> , <i>trackEnd</i> und <i>crossSection</i>)	6
Betriebsstellen (<i>operationControlPoints</i>).....	7
Fahrzeug- und Zugbildungsdaten (<i>rollingstock</i>).....	9
Fahrzeugdaten (<i>vehicles</i>)	9
Zugbildungsdaten (<i>formations</i>)	10
Fahrplandaten (<i>timetable</i>)	10
Fahrplanperioden (<i>timetablePeriods</i>).....	10
Verkehrstagerregelungen / Saisonierungen (<i>operatingPeriods</i>)	11
Zuggattungen und Produktbezeichnungen (<i>categories</i>)	12
Züge und Zugteile (<i>trains</i> und <i>trainParts</i>).....	13
Zugteile (<i>trainParts</i>).....	13
Laufweg von Zugteilen (<i>ocpsTT</i>)	15
Betriebliche und verkehrliche Züge (<i>trains</i>).....	16
Betriebliche Züge (<i>operational trains</i>).....	17
Verkehrliche Züge (<i>commercial trains</i>)	18
Umlaufpläne (<i>rosterings</i>).....	19
Umlauf-Kopfangaben (<i>rostering</i>)	20
Fahrten und Dienste (<i>blockParts</i>).....	20
Fahrtengruppen (<i>blocks</i>)	21
Umläufe (<i>circulations</i>).....	22



© iRFP • Institut für Regional- und Fernverkehrsplanung
Niederlassung Leipzig
Fasanenweg 12
D-04420 Frankenheim bei Leipzig
Telefon: +49 341 9424508
Telefax: +49 341 9424507
www.irfp.de • leipzig@irfp.de

Niederlassung Dresden
Helmholtzstr. 1
D-01069 Dresden
Telefon: +49 351 4706819
Telefax: +49 351 4768190
dresden@irfp.de

Allgemeines

Die FBS-RailML-Schnittstelle erlaubt einen Export der meisten in FBS enthaltenen Anwendungsdaten (Infrastruktur und Fahrplan/Zugdaten einschl. Kalenderdaten usw.) in strukturierter, systematischer Form zur technischen Auswertung in anderen Programmen. Erzeugt werden 8-Bit-Textdateien mit XML-Struktur. Derzeit werden XML-Schemen auf Basis des RailML[®]-Standards 2.0 verwendet, die für FBS geringfügig angepasst wurden. Die jeweils aktuelle Version der FBS-RailML-Schemendateien finden Sie unter

www.irfp.de/deutsch/fbs/schnittstelle_railml.html

Die Abweichungen vom bzw. Ergänzungen zum RailML-Standard sind in der folgenden Dokumentation gekennzeichnet. Zusätzlich wird zu jedem Attribut angegeben, ob es in der konkreten FBS-RailML-Schnittstelle obligatorisch oder optional ist. Viele der in RailML als optional deklarierten Attribute sind in der FBS-RailML-Schnittstelle obligatorisch.

Die FBS-RailML-Dateien sind grundsätzlich UTF-8-codiert. Eine RailML-Datei enthält auch Beschreibungen (Textdarstellung z. B. von Verkehrstagen) in der Sprache, die beim Exportieren als FBS-Programmiersprache eingestellt war und ist daher nicht in jeder Hinsicht sprachneutral.

Auf Grund der Abhängigkeiten, die sich durch die im RailML-Standard festgelegten Referenzen (Id-Ref-Beziehungen) ergeben, ist der Exportumfang vom Anwender nur relativ eingeschränkt beeinflussbar. So ist es z. B. nicht möglich, Züge ohne Infrastruktur zu exportieren, da der RailML-Standard erfordert, dass die verwendeten Betriebsstellen auch in der RailML-Datei definiert sein müssen. Der Anwender kann beim Export u. a. wählen:

- Vollständigkeit der Infrastruktur (nur Mindestumfang - Betriebsstellen - oder auch Strecken, Gleise und ggf. Geschwindigkeiten, Höhen usw.),
- alle Züge oder nur einen eingeschränkten Teil der Züge (Filtermöglichkeit nach Zuggattung, Produkt oder Linienbezeichnung),
- Zugteile mit oder ohne Fahrzeiten, Entfernungen und Verweise auf Streckengleise der Infrastruktur (d. h. mit oder ohne *sectionTT*-Struktur),
- mit oder ohne *commercialTrains*-Struktur,
- Umfang der zu exportierenden Umlaufpläne.

Hinweise zur Bedienung der Schnittstelle finden Sie im Dokument *FBS-RailML2 - Hinweise zur Bedienung*, welches ebenfalls unter www.irfp.de/deutsch/fbs/schnittstelle_railml.html zur Verfügung steht.

Die Implementierung der Schnittstelle ist zu keinem Zeitpunkt als abgeschlossen zu betrachten, d. h. es können jederzeit Erweiterungen erfolgen. Entsprechend dem Grundprinzip von XML sollte ein lesendes Programm daher grundsätzlich unbekannte Strukturen und Attribute überspringen. Gleichfalls können Sie uns als FBS-Anwender oder Schnittstellenpartner jederzeit kontaktieren, sofern Sie in der Schnittstelle zusätzliche (noch nicht verfügbare) Informationen vermissen, die in FBS enthalten sind.

In der folgenden Beschreibung sind alle Strukturen und Attribute aufgeführt, die zum Zeitpunkt der Erstellung dieses Dokuments Bestandteil der FBS-RailML-Schnittstelle sind - mit Ausnahme des Attributs **id**. Das Attribut **id** ist in den meisten Strukturen obligatorisch, hat jedoch nur innerhalb der RailML-Datei eine Bedeutung als Primärschlüssel und Verweisziel. Die FBS-RailML-Schnittstelle sorgt dafür, dass die Id-Werte eindeutig und integer sind; der eigentliche Inhalt dieser Werte ist jedoch nicht von Bedeutung. Ein lesendes Programm darf in den Id-Attributen keinen speziellen (konkreten) Inhalt erwarten.

Kopfinformationen (Versionsnummern, Dublin Core Metadata Element Set)

Das Attribut **railml.version** ist derzeit festgelegt auf den Wert **2.0**.

Die implementierungsabhängigen Versionsnummern und andere Kopfinformationen sind entsprechend RailML-Standard im **Dublin Core Metadata Element Set** (Namensraum „dc“, Struktur railml.metadata) enthalten. Die Interpretation der einzelnen dc-Elemente ist derzeit nicht weiter standardisiert und wird daher für die FBS-RailML-Schnittstelle wie folgt festgelegt:

Das Attribut **metadata.format** enthält die interne Versionsnummer der Schemeninstanz. Diese Versionsnummer ändert sich dann, wenn sich die Interpretation oder die Vollständigkeit der Umsetzung des RailML-Schemas durch FBS ändert. Ein lesendes Programm sollte prüfen, dass diese Versionsnummer nicht niedriger ist als die Version, mit der das Programm frühestens getestet wurde. Insbesondere kann hiermit vom lesenden Programm einfach geprüft werden, ob bestimmte notwendige Daten vorhanden sein werden, die in RailML als optional gekennzeichnet sind und in die FBS-Implementation erst nachträglich (ab einer bestimmten Versionsnummer) ergänzt wurden.

Das Attribut **metadata.identifizier** enthält eine Kompatibilitätsnummer als einfachen Integer-Wert. Diese Nummer wird nur dann geändert, wenn ein bestehender Datenwert (Attribut oder Element) nachträglich uminterpretiert wird (z. B. in Folge einer Fehlerkorrektur). Ein lesendes Programm sollte dieses Attribut auf exakt den erwarteten Wert prüfen - sonst läuft es Gefahr, dass die auszuwertenden Datenfelder nicht mehr den erwarteten Inhalt enthalten. Beispielsweise könnte ein Geschwindigkeitsfeld bis zu einem bestimmten Zeitpunkt die Geschwindigkeit in km/h enthalten. Es sei angenommen, dass sich das nachträglich als nicht RailML-konform herausstellte. Um die RailML-Konformität wiederherzustellen, müsste das Geschwindigkeitsfeld nachträglich - ohne umbenannt zu werden - auf die Einheit m/s umgestellt werden. In diesem Falle würde metadata.identifizier um eins weitergezählt werden, um lesende, nicht aktualisierte Programme davon abzuhalten, den neuen Wert als km/h einzulesen.

Das Attribut metadata.identifizier wird nicht aktualisiert, wenn neue Daten hinzukommen, was es von metadata.format unterscheidet. Es wird erwartet, dass sich metadata.identifizier im Gegensatz zu metadata.format nur sehr selten ändert.

Das Attribut **metadata.language** enthält Nummer und Namen des Zeichensatzes, in dem die FBS-Daten im FBS vorliegen. Dieser Wert kann z. B. von Bedeutung sein, wenn die in der RailML-Datei enthaltenen UTF-8-Zeichenketten (Bahnhofsnamen usw.) vom lesenden Programm in eine Nicht-Unicode-Zeichenkette umgewandelt werden müssen.

Das Attribut **metadata.source** enthält eine Zeichenkette mit Versionsnummern der verwendeten FBS-Module (ausführbare Datei und Schnittstellen-Bibliothek). Das Attribut ist für lesende Programme wenig aussagekräftig, sondern dient eher der Fehleranalyse.

Das Attribut **metadata.date** enthält Datum und Uhrzeit des Exports (des Erzeugens der RailML-Datei) im xs:dateTime-Format.

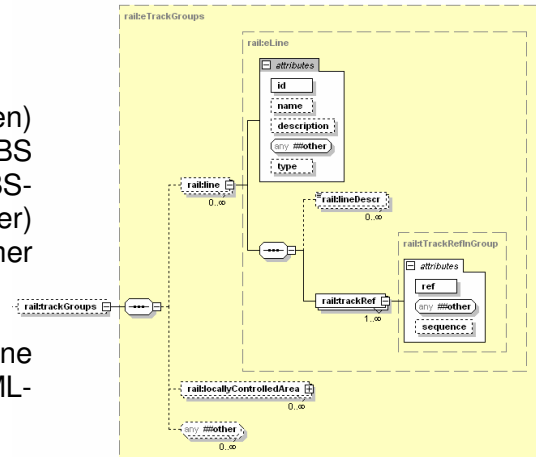
Das Attribut **metadata.creator** enthält den FBS-Lizenznamen der FBS-Lizenz, die beim Erzeugen der RailML-Datei verwendet wurde.

Infrastruktur (*infrastructure*)

Die Infrastruktur enthält immer mindestens die Struktur *operationControlPoints* mit den Betriebsstellen, die von den Zügen und Zugteilen aus *timetable* referenziert werden. Der Anwender kann wählen, ob zusätzlich die Strukturen *tracks* und *trackGroups* mit Strecken und Streckengleisen sowie innerhalb der Streckengleise die Strukturen für Geschwindigkeiten (*speedChanges*), Neigungen (*gradientChanges*), Bögen (*radiusChanges*) und Tunnel (*tunnels*) exportiert werden sollen.

Strecken (*trackGroups*)

Die Struktur **trackGroups** enthält (sofern vorhanden) für jede Strecke aus FBS einen Eintrag. Sofern in FBS Streckennummern definiert sind, wird eine FBS-Strecke in Bereiche zusammenhängender (gleicher) Streckennummer zerlegt, da die Streckennummer nach außen hin im Allgemeinen einen Primärschlüssel darstellt. Sofern in FBS keine Streckennummern angegeben sind oder eine FBS-Strecke eine durchgehende Streckennummer hat, sind die RailML-Strecken identisch mit den FBS-Strecken.



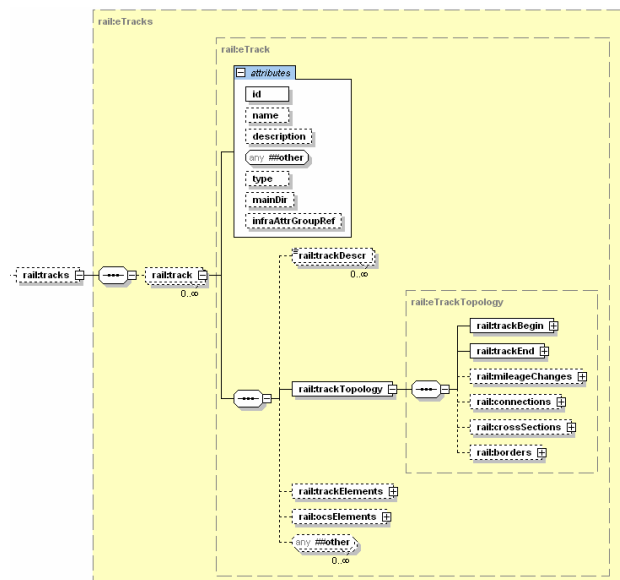
Attribut	Inhalt	optional	Bemerkungen
line.name	Abkürzung der ersten und letzten Betriebsstelle der Strecke		
line.uicNumber	sofern in FBS definiert: UIC-Nummer der Bahnverwaltung (EIU) der Strecke	x	ergänzend zu RailML
line.lineNumber	sofern in FBS definiert: Streckennummer der Strecke	x	ergänzend zu RailML
line.track.ref	Querverweis auf id eines Streckengleises in der <i>tracks</i> -Struktur		

Innerhalb der line-Struktur ist für jedes zur Strecke gehörende Streckengleis ein track.ref-Querverweis vorhanden. Jedes Streckengleis gehört zu genau einer Strecke. Jede Strecke enthält mindestens ein Streckengleis. Strecken, die abwechselnd ein- und zweigleisig sind, werden nach folgendem Prinzip in Streckengleise zerlegt:



Streckengleise (*tracks*)

Die Struktur **tracks** enthält (sofern vorhanden) Informationen zu jedem Streckengleis. Sofern **tracks** vorhanden ist, ist in jedem **track** auch je eine **trackTopology**-Struktur vorhanden, während **trackElements** optional ist und **ocsElements** derzeit nicht vorhanden ist.



Attribut	Inhalt	optional	Bemerkungen
track.name	Abkürzung der ersten und letzten Betriebsstelle des Streckengleises in Reihenfolge der Regelfahrrichtung		
track.type	derzeit immer <i>mainTrack</i> , nach Erweiterung auf nicht durchgehende Bahnhofshauptgleise zukünftig auch <i>stationTrack</i>		
trackTopology.trackBegin.pos	relative Kilometrierung der Betriebsstelle des Streckenanfangs in Metern		
trackTopology.trackBegin.absPos	absolute Kilometrierung der Betriebsstelle des Streckenanfangs in Metern		
trackTopology.trackBegin.macroscopicNode.ocpRef	Querverweis auf erste Betriebsstelle der Strecke in der Struktur <i>infrastructure.operationControlPoints</i>		
trackTopology.trackEnd.pos	relative Kilometrierung der Betriebsstelle des Streckenendes in Metern		
trackTopology.trackEnd.absPos	absolute Kilometrierung der Betriebsstelle des Streckenendes in Metern		
trackTopology.trackEnd.macroscopicNode.ocpRef	Querverweis auf letzte Betriebsstelle der Strecke in der Struktur <i>infrastructure.operationControlPoints</i>		
trackTopology.mileageChanges	s. u.	x	
trackTopology.crossSections	s. u.		
trackTopology.trackElements.speedChanges	s. u.	x	in Vorbereitung
trackTopology.trackElements.radiusChanges	s. u.	x	in Vorbereitung
trackTopology.trackElements.tunnels	s. u.	x	in Vorbereitung
trackTopology.trackElements.ownerChanges	s. u.	x	in Vorbereitung
trackTopology.trackElements.trainProtectionChanges	s. u.	x	in Vorbereitung
trackTopology.trackElements.electrificationChanges	s. u.	x	in Vorbereitung
trackTopology.trackElements.axleWeightChanges	s. u.	x	in Vorbereitung
trackTopology.trackElements.gaugeChanges	s. u.	x	in Vorbereitung

Die relative Kilometrierung eines Gleises (Attribute *pos*) ist immer ununterbrochen fortlaufend steigend, jedoch nicht unbedingt bei 0 beginnend. (Dann nicht, wenn innerhalb einer Strecke das Gleis gewechselt wird, d. h. wenn das Gleis nicht am Anfang der Strecke beginnt.) Die absolute Kilometrierung eines Gleises ist willkürlich (i. d. R. historisch bedingt) und kann be-

liebige Unstetigkeitsstellen aufweisen („springen“) sowie entgegen der relativen Kilometrierung verlaufen („fallen“). Der Zusammenhang zwischen relativer und absoluter Kilometrierung erschließt sich vollständig durch die *mileageChanges*-Struktur. Die Angabe der absoluten Kilometrierung an den Betriebsstellen (in *trackBegin*, *trackEnd* und *crossSection*) ist redundant.

Kilometrierungsbereiche (*mileageChanges*)

Die Struktur **mileageChanges** definiert den Zusammenhang zwischen relativer (fortlaufender) und absoluter (außen sichtbarer) Kilometrierung jedes Streckengleises und damit jeder Strecke. In FBS sind beide Gleise einer zweigleisigen Strecke grundsätzlich identisch kilometriert. In dem Ausnahmefall, bei dem die absolute Kilometrierung auf der gesamten Strecke identisch ist mit der relativen Kilometrierung, kann die gesamte *mileageChanges*-Struktur in der RailML-Datei fehlen. Es sind dann dennoch die absoluten Positionen der Betriebsstellen (in *trackBegin*, *trackEnd* und *crossSection*) angegeben und immer identisch mit der jeweiligen relativen Position.

Sofern die *mileageChanges*-Struktur vorhanden ist, ist immer mindestens ein *mileageChange*-Element am Anfang des Streckengleises (d. h. mit gleicher relativer *pos* wie *trackBegin*) vorhanden. Dieses Element definiert daher eigentlich keinen *Wechsel*, sondern die initiale Kilometrierung des Gleises vor dem ersten Wechsel.

Attribut	Inhalt	optional	Bemerkungen
absPosIn	absolute Kilometrierung des Kilometrierungswechsels in Richtung fallender relativer Kilometrierung	x	nicht beim initialen (ersten) Element (<i>abweichend von RailML</i>)
type	= <i>missing</i> , wenn absPosIn < absPos gilt = <i>overlapping</i> , wenn absPosIn > absPos gilt	x	
absPos	absolute Kilometrierung des Kilometrierungswechsels in Richtung steigender relativer Kilometrierung		
pos	relative Kilometrierung des Kilometrierungswechsels		
dir	= <i>up</i> , wenn absolute km-Werte in Richtung steigender relativer Kilometrierung steigen = <i>down</i> , wenn absolute km-Werte in Richtung steigender relativer Kilometrierung fallen		

Alle Positionen sind in Metern angegeben.

Querverweise auf Betriebsstellen (*trackBegin*, *trackEnd* und *crossSection*)

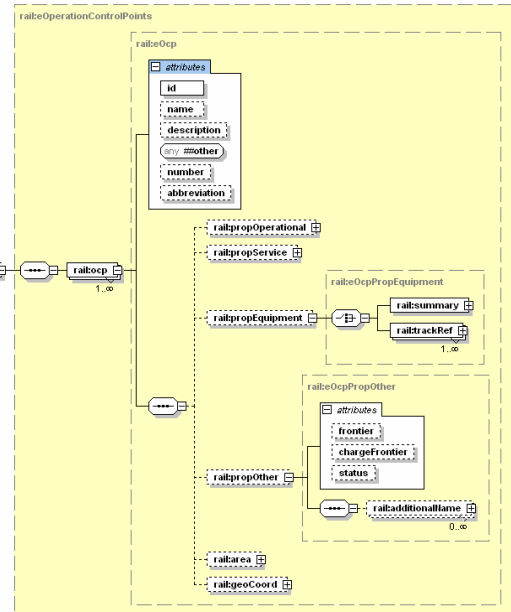
Attribut	Inhalt	optional	Bemerkungen
pos	relative Kilometrierung der Betriebsstelle in Metern		
absPos	absolute Kilometrierung der Betriebsstelle in Metern		
ocpRef	Querverweis auf die Betriebsstelle in der Struktur <i>infrastructure.operationControlPoints</i>		

Als „Betriebsstelle“ können hier im weitesten Sinne (je nach Konfiguration in FBS und Einstellung beim Export) auch Blocksignale, Blockkennzeichen oder reine Fahrzeitmesspunkte (ohne funktionalen Hintergrund) aufgezählt sein.

Betriebsstellen (*operationControlPoints*)

Die Struktur **operationControlPoints** enthält mindestens jede im Fahrplan (Struktur *timetable*) im Zuglauf oder in Umlaufplänen vorkommende Betriebsstelle in loser Folge. Dies sind mindestens (je nach Einstellung beim Export) die Betriebsstellen, an denen Züge beginnen, enden oder einen Verkehrshalt haben. Im Normalfall sind jedoch auch alle anderen betrieblich relevanten oder als Infrastruktur definierten Betriebsstellen bis hin z. B. zu Blocksignalen und Blockkennzeichen vorhanden.

Die Reihenfolge der Betriebsstellen entlang von Strecken wird (sofern vorhanden) durch die Strukturen *trackGroups* und *tracks* definiert.



Attribut	Inhalt	optional	Bemerkungen
name	(innerbetrieblicher) Name der Betriebsstelle	x	
abbreviation	Abkürzung der Betriebsstelle		Korrektur zu RailML
number	„interne Betriebsstellennummer“ aus FBS-Betriebsstellenverzeichnis (sofern dort definiert)	x	
propOperational.operationalType	nur vorhanden, wenn in FBS als <i>funktionale</i> Betriebsstelle definiert (<i>ensuresTrainSequence=true</i>): = <i>junction</i> bei Abzweigstelle = <i>crossover</i> bei Überleitstellen = <i>station</i> bei allen übrigen Zugmeldestellen = <i>blockSignal</i> bei Blocksignalen = <i>blockPost</i> bei allen übrigen Zugfolgestellen	x	
propOperational.orderChangeable	= <i>true</i> bei Zugmeldestellen, sonst = <i>false</i>	x	
propOperational.ensuresTrainSequence	= <i>true</i> bei Zugfolgestellen (einschl. Zugmeldestellen), sonst = <i>false</i>	x	
propOperational.uptime	sofern in FBS (zeitweise) ausgeschaltet oder (permanent) außer Betrieb, wird die Ausschaltzeitspanne mit <i>mode=off</i> angegeben.	x	*1
propService.passenger	= <i>true</i> , wenn in FBS als Zugangsstelle Reiseverkehr definiert	x	default=false
propService.service	= <i>true</i> , wenn in FBS mit Servicekategorie definiert	x	default=false
propService.ship	= <i>true</i> , wenn in FBS mit Übergang (Verknüpfungsstelle) zum Schiff definiert	x	default=false
propService.bus	= <i>true</i> , wenn in FBS mit Übergang (Verknüpfungsstelle) zum Bus definiert	x	default=false
propService.tariffpoint	= <i>true</i> , wenn in FBS als Tarifpunkt oder -grenze definiert	x	default=false
propService.goodsLoading	= <i>true</i> , wenn in FBS als Umschlagbahnhof oder Ladestelle definiert	x	default=false
propService.goodsSiding	= <i>true</i> , wenn in FBS als Anschlussstelle definiert	x	default=false
propService.goodsIntermodal	= <i>true</i> , wenn in FBS als Umschlagbahnhof definiert	x	default=false
propService.goodsMarshalling	= <i>true</i> , wenn in FBS als Rangierbahnhof definiert	x	default=false

propEquipment.summary.hasHomeSignals	= <i>true</i> , wenn in FBS als Bahnhof mit Einfahrsignalen definiert, sonst = <i>false</i>	x	
propEquipment.summary.hasStarterSignals	= <i>true</i> , wenn in FBS als Betriebsstelle mit Ausfahr- oder Blocksignalen definiert, sonst = <i>false</i>	x	
propEquipment.summary.hasSwitches	= <i>true</i> , wenn in FBS als Zugmeldestelle, jedoch nicht als Endpunkt definiert, sonst = <i>false</i>	x	
propOther.status	= <i>planned</i> , wenn in FBS als <i>in Planung</i> definiert = <i>closed</i> , wenn in FBS als <i>außer Betrieb</i> oder <i>stillgelegt</i> definiert	x	* ¹
propOther.additionalName	.name: alternativer Name der Betriebsstelle .type= <i>operationalName</i> bei abweichender Schreibweise des innerbetrieblichen Namens der Betriebsstelle im Bildfahrplan vom Betriebsstellenverzeichnis .type= <i>trafficName</i> bei abweichender Schreibweise des verkehrlichen Namens, alternativer Sprache (Görlitz - Zhorjelic) oder alternativem Zeichensatz (Thessaloniki - ΘΕΣΣΑΛΟΝΙΚΗ).	x	* ² <i>trafficName</i> kann mehrfach vorkommen (verschiedene Sprachen)
area.name	Name der Gemeinde der Betriebsstelle	x	* ²
area.number	Nummer der Gemeinde der Betriebsstelle	x	* ²
area.zip	Postleitzahl der Betriebsstelle	x	* ²
geoCoord.coord	Längen- und Breitengrad bzw. x- und y-Koordinate und Höhe der Betriebsstelle	x	* ²

*¹ Funktionale Betriebsstellen (Zugmelde- und Zugfolgestellen), die mit *propOperational.uptime.mode=off* oder *propOther.status=closed* definiert sind, wirken wie funktionslose Betriebsstellen (d. h. beeinflussen die Zugfolge nicht).

*² sofern im FBS-Betriebsstellenverzeichnis entsprechend definiert

Bei widersprüchlichen Angaben zwischen FBS-Bildfahrplan und FBS-Betriebsstellenverzeichnis wird die Angabe aus dem FBS-Bildfahrplan ausgegeben (betrifft insbesondere *propOperational* und *propService*). Betriebsstellen (nur Zugmeldestellen), die in FBS streckenabhängig unterschiedliche Eigenschaften haben (z. B. auf einer Strecke Bahnhof, auf einer anderen Strecke Abzweigstelle), werden mit der Vereinigungsmenge der Eigenschaften exportiert.

Es ist möglich, dass in RailML mehrere Betriebsstellen die gleiche Abkürzung (Attribut *abbreviation*) haben. Dies ist z. B. bei Infrastruktur der DB Netz AG bei Betriebsstellen an parallelen Strecken so (Beispiel: Haltepunkt Kürbitz, DB-Abkürzung DKUR; Templin Stadt, DB-Abkürzung WTV). Durch die gleiche Abkürzung wird indirekt codiert, dass hier eine verkehrliche Verbindung (Umsteigemöglichkeit) zwischen zwei betrieblich getrennten Stellen vorliegt.

Fahrzeug- und Zugbildungsdaten (rollingstock)

Die Struktur **rollingstock** enthält grundsätzlich jedes im Fahrplan (Struktur *timetable*) vorkommende Fahrzeug (in **vehicles**) sowie die im Fahrplan vorkommenden Zugbildungen (in **formations**) in loser Folge, jedoch nicht alle möglichen (jedoch im exportierten Fahrplan nicht vorkommenden) Fahrzeuge.

Fahrzeugdaten (vehicles)

Die in **vehicles** angegebenen Informationen stellen nur einen groben Auszug aus den in FBS vorgehaltenen Informationen über die Fahrzeuge dar, die im Wesentlichen zum Interpretieren der Fahrpläne (verkehrlichen Eigenschaften) dienen. Insbesondere sind technische Angaben zu Zugkraft und Bremsen hier nicht vorgesehen, da als Hauptzweck dieser Schnittstellenimplementierung das Austauschen von Fahrplaninformationen angesehen wird. Zum Austausch von Tfz.-Daten gibt es eine spezielle FBS-Schnittstellenimplementierung.

Attribut	Inhalt	optional	Bemerkungen
name	Tfz.-Baureihe oder Wagengattung, ggf. ergänzt um Abkürzung des einstellenden EVUs (z. B. DB.234)		
axleSequence	Achsfolge / Achsformel	x	* ¹
numberDrivenAxles	Anzahl angetriebener Achsen	x	* ¹
numberNonDrivenAxles	Anzahl nicht angetriebener Achsen	x	optional bei Tfz.
trackGauge	Spurweite in Metern (mit Nachkommastellen)	x	
length	Länge über alles in Metern		
speed	zulässige Höchstgeschwindigkeit in km/h		
bruttoWeight	Gesamtmasse in Tonnen		
nettoWeight	Leermasse in Tonnen	x	
nettoAdhesionWeight	Reibungsmasse in Tonnen	x	* ¹
classification. manufacturer. manufacturerName	Kurzbezeichnung des Herstellers	x	* ¹
classification. manufacturer. manufacturerType	Bezeichnung des Fahrzeugs beim Hersteller	x	* ¹
classification. operator. operatorName	Kurzbezeichnung des Betreibers (EVUs)	x	* ¹
classification. operator. operatorClass	Bezeichnung des Fahrzeugs beim EVU	x	* ¹
wagon. kinematicEnvelope	Kurzbezeichnung des Lichtraumprofils	x	
wagon. passenger.drivingCab	= <i>true</i> bei Triebwagen	x	* ¹
wagon. passenger.tilting	= <i>true</i> bei Neigetechnik-Fahrzeugen	x	* ¹
wagon. passenger.places	.count: Anzahl Plätze u. a. Kategorie .category: mögliche Werte: <i>class1, class2, other: foldingSeat, standing, wheelchair, bicycle, couchette, bed, other: WC</i>	x	
wagon. passenger.service	.name: mögliche Werte: <i>WR, Bistro</i> .count=1 wenn vorhanden	x	

*¹ nur bei Triebfahrzeugen

Zugbildungsdaten (*formations*)

Ein **formation**-Element enthält die Zugbildung eines Zugteils aus FBS. Dies kann eine Kombination aus Triebfahrzeug und Wagen sein, kann jedoch auch nur aus einem oder mehreren Wagen (z. B. einzelner Kurswagen oder Kurswagengruppe) oder nur aus Triebfahrzeugen bestehen (z. B. einzeln fahrende Lokomotive oder Triebwagen).

Attribut	Inhalt	optional	Bemerkungen
name	zusammenfassender Text aus Tfz.-Baureihe (sofern vorhanden), Wagengattungen (sofern vorhanden) und jeweiligen Anzahlen, z. B.: DB.234+2xBocm242+WLAB176+2xDDm...		
length	Summe der Länge aller Fahrzeuge in Metern		
speed	Minimum der zulässigen Höchstgeschwindigkeit aller Fahrzeuge in km/h		
weight	Summe der Bruttomasse aller Fahrzeuge in t		
vehicleRef. orderNumber	laufende Nummer der Fahrzeuggruppe im Zug		Zählung ab 1
vehicleRef. vehicleRef	Querverweis auf id des Fahrzeugtyps der Fahrzeuggruppe in der Struktur <i>vehicles</i>		
vehicleRef. vehicleCount	Anzahl der Fahrzeuge des angegebenen Typs in der Gruppe		

Fahrplandaten (*timetable*)

Fahrplanperioden (*timetablePeriods*)

Die Struktur **timetablePeriods** ist nur vorhanden, wenn im exportierten FBS-Netz eine Fahrplanperiode definiert ist. Sie enthält dann immer nur einen **timetablePeriod**-Eintrag, da eine FBS-Netzdatei nicht mehr als eine Fahrplanperiode haben kann und derzeit nicht aus mehreren Netzdateien gleichzeitig exportiert werden kann.

Bei nicht definierter Fahrplanperiode (nicht vorhandener *timetablePeriods*-Struktur) können in den folgenden Fahrplandaten nur allgemeine Verkehrstagerregelungen der sieben Wochentage (Mo-So) und Wochenfeiertage, jedoch kein konkreter Datumsbezug vorkommen.

Attribut	Inhalt	optional	Bemerkungen
name	Kurzbezeichnung der Fahrplanperiode; je nach Länge z. B. „2010“ oder „2010/2011“ oder „06-09/2010“ oder „22.-28.08.2010“		
startDate	erster Gültigkeitstag der Fahrplanperiode		
endDate	letzter Gültigkeitstag der Fahrplanperiode		
holidays. holiday. holidayDate	Aufzählung der Feiertage innerhalb der Fahrplanperiode (sofern definiert)	x	

Verkehrstagerregelungen / Saisonierungen (*operatingPeriods*)

Die Struktur **operatingPeriods** enthält für jede in den Fahrplan- und Umlaufdaten vorkommende Verkehrstagerregelung je einen Eintrag. Diese Struktur ist immer dann vorhanden, wenn Züge oder Umläufe in der RailML-Datei enthalten sind - auch dann, wenn keine Fahrplanperiode angegeben ist.

Attribut	Inhalt	optional	Bemerkungen
name	kurze Textdarstellung der Verkehrstagerregelung, z. B.: „täglich; nicht 24.,31.12.; 9.,16.,23.7.“		sprachabhängig
description	lange Textdarstellung der Verkehrstagerregelung, z. B.: „verkehrt Montag-Freitag, auch an Feiertagen, nicht am 24., 31.12.“		sprachabhängig
timetablePeriodRef	sofern vorhanden, Querverweis auf id der (einzigen definierten) Fahrplanperiode	x	*1
bitMask	Binärzahl* ² , die so viele Ziffern enthält wie die Fahrplanperiode Tage (d. h. Anzahl Ziffern = endDate - startDate + 1); gibt für jeden Tag der Fahrplanperiode an, ob die Verkehrstagerregelung an diesem Tag gilt (verkehrt, =1) oder nicht (=0).	x	*1
operatingDay. operatingCode	Binärzahl* ² mit 7 Ziffern für die Wochentage Montag (erste Ziffer) bis Sonntag (letzte Ziffer); gibt an, ob die Verkehrstagerregelung am jeweiligen Wochentag gilt (verkehrt, =1) oder nicht (=0).		
operatingDay. onRequest	= <i>true</i> , wenn Züge mit dieser Verkehrstagerregelung an den angegebenen Tagen nur bei Bedarf verkehren (d. h. ohne Ankündigung ausfallen können)	x	default= <i>false</i>
operatingDay. startDate	erster Gültigkeitstag der <i>operatingDay</i> -Angabe (liegt immer innerhalb der Fahrplanperiode)	x	*1
operatingDay. endDate	letzter Gültigkeitstag der <i>operatingDay</i> -Angabe (liegt immer innerhalb der Fahrplanperiode)	x	*1
operatingDay. operatingDayDeviance.operatingCode	Binärzahl* ² mit 7 Ziffern für die Wochentage Montag (erste Ziffer) bis Sonntag (letzte Ziffer); gibt an, dass die Verkehrstagerregelung abweichend von <i>operatingCode</i> verkehrt (=1) oder nicht verkehrt (=0), wenn der Wochentag in der unter <i>holidayOffset</i> angegebenen Relation zu einem Feiertag steht	x	
operatingDay. operatingDayDeviance.holidayOffset	Versatz in Anzahl Tagen des <i>operatingDayDeviance</i> -Elements zu einem Feiertag; z. B.: <i>holidayOffset</i> =0 bedeutet „an einem Feiertag“ <i>holidayOffset</i> =1 „einen Tag nach einem Feiertag“ <i>holidayOffset</i> =-1 „einen Tag vor einem Feiertag“	x	
operatingDay. operatingDayDeviance.ranking	definiert die Priorität bei Vorhandensein mehrerer gleichzeitig zutreffender <i>operatingDayDeviance</i> -Elemente (z. B. 25.12. ist gleichzeitig Feiertag und ein Tag vor einem Feiertag): <i>operatingDayDeviance</i> -Elemente mit niedrigerem <i>ranking</i> -Wert überschreiben solche mit höherem	x	
specialService	Aufzählung von Ausnahmeverkehrstagen, an denen von den durch die <i>operatingDay</i> -Elemente definierten Bildungsregeln abgewichen wird .type= <i>include</i> für Zusatzverkehrstage .type= <i>exclude</i> für Ausfalltage .startDate: erster Gültigkeitstag der Ausnahme .endDate: letzter Gültigkeitstag der Ausnahme .singleDate: verwendet im Fall startDate=endDate	x	*1

*1 nur bei definierter Fahrplanperiode

*2 Zeichenkette, bestehend nur aus den Zeichen 0 und 1

Jedes *operatingPeriod*-Element hat mindestens ein *operatingDay*-Element. Falls mehrere *operatingDay*-Elemente in einem *operatingPeriod*-Element vorkommen, sind diese durch ihre *startDate-endDate*-Bereiche oder ihre *operatingCode*-Bitmaske immer disjunkt.

Beispiele zu Verkehrstagerregelungen finden Sie in der iRFP-Beispielbeschreibung zu RailML.

Zuggattungen und Produktbezeichnungen (*categories*)

Die Struktur **categories** enthält einen **category**-Eintrag für jede im Fahrplan vorkommende Zuggattung und Produktbezeichnung.

Attribut	Inhalt	optional	Bemerkungen
abbreviation	Kurzbezeichnung / Abkürzung der Zuggattung (z. B. „ICE“)		
name	Langbezeichnung der Zuggattung (z. B. „InterCityExpress“)	x	
description	Beschreibung der Verwendung der Zuggattung, z. B. „internationale Ganzgüterzüge“	x	
trainUsage	= <i>goods</i> bei Gattungen für Güterzüge = <i>passenger</i> bei Gattungen für Reisezüge = <i>mixed</i> bei Gattungen für beide Zugarten	x	nicht vorhanden bei Gattungen für Tfz.-Leerfahrten oder Dienstzüge
deadrun	= <i>true</i> , wenn die Zuggattung für einen Leerzug steht (Leerreise- oder Leergüterzug oder Tfz.-Leerfahrt)	x	default= <i>false</i>

Die optionalen Attribute sind nur dann vorhanden, wenn die entsprechende Zuggattung bzw. das Produkt im FBS-Gattungsverzeichnis definiert wurden.

Züge und Zugteile (*trains* und *trainParts*)

Die Grundphilosophie der RailML-*timetable*-Version 2.0 ist, den vielfältigen Praxisanforderungen des Eisenbahnbetriebs wie z. B. Flügeln von Zügen, Verstärken, Wagenzugdurchlauf, Einsatz von Kurswagen usw. durch „Zerlegen“ von Zügen in kleinste, unteilbare Einheiten gerecht zu werden. Diese kleinsten unteilbaren Zugeinheiten werden **Zugteile** genannt. Die eigentlichen Zuginformationen wie Zeiten, Fahrzeuge usw. sind an den Zugteilen zu finden. Die **Zug**-Struktur (***trains***) in RailML hingegen fasst lediglich Zugteile zu Zügen zusammen, enthält aber darüber hinausgehend i. d. R. keine weiteren Informationen.

Weitere Informationen und Beispiele zu Zügen und Zugteilen finden Sie in der iRFP-Beispielbeschreibung zu RailML.

Zugteile (*trainParts*)

Die Struktur *trainParts* enthält je einen *trainPart*-Eintrag für jede zusammenhängende Teilmenge eines Zuges mit homogenen Eigenschaften. Während sich innerhalb eines Zuglaufes z. B. die Verkehrstage oder die Anzahl Fahrzeuge ändern können, sind innerhalb eines *trainParts* immer alle Eigenschaften konstant.

Attribut	Inhalt	optional	Bemerkungen																						
name	Zugteilnummer (in FBS vom Anwender eingegeben; kann alphanumerisch sein)																								
line	Linienbezeichnung des Zugteils aus FBS	x																							
trainNumber	Zugnummer des (betrieblichen) Trägerzuges aus FBS (s. a. <i>timetable.trains.trainNumber</i>)		kann alphanumerisch sein																						
processStatus	<table border="0"> <tr> <td><u>Status in FBS</u></td> <td><u>Ausgabe in RailML</u></td> </tr> <tr> <td>angelegt/konstruiert</td> <td>'planned'</td> </tr> <tr> <td>zu bestellen</td> <td>'toBeOrdered'</td> </tr> <tr> <td>bestellt (EVU→EIU)</td> <td>'ordered'</td> </tr> <tr> <td>Bestellung bearbeitet (EIU)</td> <td>'edited'</td> </tr> <tr> <td>angeboten (EIU→EVU)</td> <td>'offered'</td> </tr> <tr> <td>bestätigt (EVU→EIU)</td> <td>'confirmed'</td> </tr> <tr> <td>veröffentlicht</td> <td>'published'</td> </tr> <tr> <td>zurückgestellt</td> <td>'placedBack'</td> </tr> <tr> <td>gefahren</td> <td>'driven'</td> </tr> <tr> <td>abgerechnet</td> <td>'settled'</td> </tr> </table>	<u>Status in FBS</u>	<u>Ausgabe in RailML</u>	angelegt/konstruiert	'planned'	zu bestellen	'toBeOrdered'	bestellt (EVU→EIU)	'ordered'	Bestellung bearbeitet (EIU)	'edited'	angeboten (EIU→EVU)	'offered'	bestätigt (EVU→EIU)	'confirmed'	veröffentlicht	'published'	zurückgestellt	'placedBack'	gefahren	'driven'	abgerechnet	'settled'		sofern ergänzend zu RailML mit 'other.'+...
<u>Status in FBS</u>	<u>Ausgabe in RailML</u>																								
angelegt/konstruiert	'planned'																								
zu bestellen	'toBeOrdered'																								
bestellt (EVU→EIU)	'ordered'																								
Bestellung bearbeitet (EIU)	'edited'																								
angeboten (EIU→EVU)	'offered'																								
bestätigt (EVU→EIU)	'confirmed'																								
veröffentlicht	'published'																								
zurückgestellt	'placedBack'																								
gefahren	'driven'																								
abgerechnet	'settled'																								
remarks	Bemerkungen zum Zugteil aus FBS	x	erst ab V2.0.3																						
timetablePeriodRef	sofern das FBS-Netz eine definierte Fahrplanperiode hat, enthält dieses Feld die id der Fahrplanperiode aus der Struktur <i>timetable.timetablePeriods</i> der RailML-Datei.	x																							
categoryRef	Querverweis auf id des Eintrags der Produktbezeichnung / Gattung des Zugteils in der Struktur <i>timetable.categories</i> der RailML-Datei	x																							
formationTT.formationRef	Querverweis auf id des Eintrags der Zugbildung des Zugteils in der Struktur <i>rollingstock.formation</i> der RailML-Datei																								
formationTT.weight	Gesamtmasse des Zugteils in t		*1																						
formationTT.load	Last des Zugteils in t (Masse ohne Tfz.)		*1																						
formationTT.length	Länge des Zugteils in m		*2																						
formationTT.speed	Höchstgeschwindigkeit in km/h		*1																						
formationTT.equipmentUsage.equipment	Zugbeeinflussung des Trägerzuges des Zugteils	x																							

formationTT. passengerUsage. places	.count: Anzahl Plätze u. a. Kategorie .category: mögliche Werte: <i>class1, class2, other:foldingSeat, standing, wheelchair, bicycle, couchette, bed, other:WC</i>	x	*1 *3
formationTT. passengerUsage. service	.name: mögliche Werte: <i>WR, Bistro</i> .count=1 wenn vorhanden	x	*1 *3
formationTT. reservationInfo. booking	enthält Auflistung der Wagennummern für die Platzreservierung (Reservierungs-/Buchungsnummern) zu jedem Fahrzeug in der Formation .bookingNumber: Reservierungsnummer .posInFormation: Index des Wagens innerhalb der Formation (Zählung ab 1) ^{*4} .vehicleRef: Querverweis auf id des Fahrzeugs innerhalb der Struktur <i>rollingstock.vehicles</i>	x	
operatingPeriodRef. ref	Querverweis auf Eintrag der Verkehrstageregelung des Zugteils in der Struktur <i>timetable.operatingPeriods</i> der RailML-Datei <i>Die übrigen Felder von operatingPeriodRef werden nicht benutzt, da die sich dann ergebenden Verkehrstagerregelungen keine Bitmaske hätten.</i>		

*1 kann vom entsprechenden Wert der Formation abweichen

*2 ist identisch mit dem entsprechenden Wert der Formation

*3 Wenn die Fahrzeug-Formation eines Zugteils Plätze besitzt (Sitzplätze, Bettplätze usw.) und der Zugteil nicht öffentlich ist (in FBS: Gattung oder Produkt nicht als Reisezug definiert oder Zugteil explizit auf „nicht veröffentlichen“ gesetzt), werden die Sitzplätze der Formation unter *trainPart.formationTT.passegerUsage.places* explizit mit *count=0* angegeben (überschrieben). Wenn der Zugteil öffentlich ist (in FBS: Gattung oder Produkt als Reisezug definiert und Zugteil nicht explizit auf „nicht veröffentlichen“ gesetzt), sind die Sitzplätze des Zugteils identisch mit denen der Fahrzeug-Formation. Wenn Gattung und Produkt des Zugteils nicht angegeben oder undefiniert sind, fehlt auch die Struktur *trainPart.formationTT.passegerUsage*.

*4 Wenn Fahrzeuge ohne Buchungsnummer in der Formation sind, werden diese von *posInFormation* übersprungen; *posInFormation* ist dann nicht fortlaufend. *posInFormation* beginnt ab 1 für das erste Fahrzeug zu zählen; wenn das erste Fahrzeug ein Triebfahrzeug ist und keine Buchungsnummer hat, ist der erste *posInFormation*-Wert demzufolge 2.

Im Allgemeinen gibt es zu jedem Zug einen Status der Bearbeitung. Aus diesem geht hervor, wo der Zug in der Planung bzw. „Verhandlung zwischen EVU und EIU“ steht. Alle letztendlich zu veröffentlichenden Züge sollten beim Infrastrukturunternehmen einen abgeschlossenen Trassenvertrag haben, d. h. mindestens den Status *confirmed*. Der Rang der Werte des Attributs *processStatus* ist steigend von oben nach unten. Es sind (aus FBS heraus) folgende Zustände möglich:

<i>planned</i>	Zug in FBS vorhanden, aber nicht zum Bestellen freigegeben
<i>other:toBeOrdered</i>	Zug soll beim EIU bestellt werden, ist aber noch nicht bestellt
<i>other:ordered</i>	Zug wurde beim EIU bestellt
<i>other:edited</i>	Zug wurde vom EIU bearbeitet
<i>other:offered</i>	Zug wurde vom EIU an EVU angeboten
<i>other:confirmed</i>	EVU hat Trasse beim EIU bestätigt = Vertrag ist gültig
<i>other:published</i>	EVU hat Trasse veröffentlicht bzw. zum Veröffentlichen freigegeben
<i>other:placedBack</i>	Zug wurde zurückgestellt - vmtl. Detailverhandlungen zur Trasse
<i>other:driven</i>	Zug ist bereits gefahren
<i>other:settled</i>	Zug ist gefahren und abgerechnet

Laufweg von Zugteilen (ocpsTT)

Die Struktur **ocpsTT** jedes *trainPart*-Elements enthält einen **ocpTT**-Eintrag für jede Betriebsstelle, die der Zugteil auf seinem Laufweg passiert. Es sind immer mindestens zwei Betriebsstellen vorhanden; je nach Export-Einstellungen können jedoch bestimmte Zwischenbetriebsstellen „übersprungen“ werden (z. B. reine Blocksignale/Zugfolgestellen oder auch Bahnhöfe, an denen kein Verkehrshalt stattfindet).

Attribut	Inhalt	optional	Bemerkungen
ocpRef	Querverweis auf die id der Betriebsstelle in der Struktur <i>infrastructure.operatingControlPoints</i>		
ocpType	= <i>pass</i> bei Durchfahrt * ¹ = <i>begin</i> bei beginnendem Gesamtzuglauf = <i>end</i> bei endendem Gesamtzuglauf = <i>stop</i> bei Betriebs- oder Verkehrshalt		
trackInfo	Gleisbezeichnung des Bahnhofsgleises, welches der Zug(teil) innerhalb der Betriebsstelle benutzt; keine Angabe steht hier i. A. für das durchgehende Hauptgleis der jeweiligen Fahrtrichtung	x	
remarks	Bemerkungen des Anwenders zum Halt oder zur Durchfahrt an dieser Betriebsstelle	x	
trainReverse	= <i>true</i> , wenn der Zug(teil) an dieser Betriebsstelle die Fahrtrichtung wechselt („kopfmacht“)	x	default= <i>false</i> - in Vorbereitung -
times.scope	= <i>scheduled</i>		
times.arrival	Ankunftszeit des Zuglaufs an dieser Betriebsstelle	x	nicht vorhanden bei Durchfahrten und an der ersten Betriebsstelle des Gesamtlaufs
times.arrivalDay	Anzahl der Mitternachtsübergänge (Folgetage) zwischen Abfahrt des ersten Zugteils am ersten Bahnhof und Abfahrt an dieser Betriebsstelle * ²	x	
times.departure	Abfahrts- oder Durchfahrtszeit an dieser Betriebsstelle	x	nicht vorhanden an der ersten Betriebsstelle des Gesamtlaufs
times.departureDay	Anzahl der Mitternachtsübergänge (Folgetage) zwischen Abfahrt des ersten Zugteils am ersten Bahnhof und Abfahrt an dieser Betriebsstelle * ²	x	
sectionTT	enthält Informationen zum Abschnitt zwischen der aktuellen und der nächsten Betriebsstelle im Zuglauf		enthält <i>keine Attribute</i> an der letzten Betriebsstelle
sectionTT.section	Abkürzung der aktuellen und der nächsten Betriebsstelle im Zuglauf, durch - getrennt	x	
sectionTT.lineRef	Querverweis auf id der Strecke aus der Struktur <i>infrastructure.trackGroups</i> , welche vom Zug von hier bis zur nächsten Betriebsstelle benutzt wird	x	
sectionTT.trackRef	Querverweis auf id des Streckengleises aus der Struktur <i>infrastructure.tracks</i> , welches vom Zug von hier bis zur nächsten Betriebsstelle benutzt wird * ³	x	abweichend von RailML
sectionTT.trackInfo	=1 für Regelgleis =2 für Gegengleis	x	nur auf zweigleisigen Abschnitten * ³
sectionTT.percentageSupplement	linearer Fahrzeitzuschlag zwischen der aktuellen und der nächsten Betriebsstelle in Prozent	x	
sectionTT.distance	Entfernung zwischen der aktuellen und der nächsten Betriebsstelle in km (mit drei Nachkommastellen)	x	
sectionTT.runTimes.minimalTime	kürzeste (berechnete) Fahrzeit zwischen der aktuellen und der nächsten Betriebsstelle ohne linearen Fahrzeitzuschlag	x	

sectionTT. runTimes. operationalReserve	Differenz aus kürzester und planmäßiger Fahrzeit der aktuellen und der nächsten Betriebsstelle	x	
sectionTT. runTimes. additionalReserve	nichtlinearer Fahrzeitzuschlag (sofern vorhanden), d. h. planmäßige Fahrzeit - kürzeste Fahrzeit - linearer Fahrzeitzuschlag = operationalReserve - linearer Fahrzeitzuschlag	x	
stopDescription...	Enthält Informationen zum Halt an der aktuellen Betriebsstelle	x	nicht vorhanden bei Durchfahrten
stopDescription. commercial	= <i>true</i> bei Verkehrshalt = <i>false</i> bei Betriebshalt	x	
stopDescription. stopOnRequest	= <i>true</i> bei Bedarfshalt (bedingtem Verkehrshalt) = <i>false</i> bei unbedingtem Verkehrshalt	x	
stopDescription. onOff	= <i>on</i> für „hält nur zum Einsteigen“ = <i>off</i> für „hält nur zum Aussteigen“	x	
stopDescription. stopTimes minimalTime	Regelaufenthaltszeit des Zuges an dieser Betriebsstelle (kann von der planmäßigen Aufenthaltszeit abweichen)	x	

*¹ *pass* kann auch an der ersten oder letzten Betriebsstelle vorkommen bei durchfahrend ineinander übergebenden Zugteilen (Laufwegabschnitten)

*² Die Tageszählung von *arrivalDay* und *departureDay* beginnt mit 0 an der Abfahrt des ersten Zugteils einer Reihe sequentiell zusammenhängender Zugteile. Daher kann der Wert >0 sein an der ersten Betriebsstelle eines Zugteils.

*³ Die Angaben *sectionTT.trackRef* und *sectionTT.trackInfo* sind redundant: Wenn der Zug(teil) auf dem Gegengleis fährt (*sectionTT.trackInfo=2*), verweist auch *sectionTT.trackRef* auf die **id** des jeweiligen (abhängig von der Fahrtrichtung) Gegengleises.

Die RailML-Datei enthält die Zeiten sekundengenau, wie es für XML definiert ist. In der Eisenbahnwelt ist es jedoch üblich, Zeiten nur 0,1-min-genau zu planen. Dem Anwender von FBS werden die Zeiten i. d. R. auch nur auf 0,1 min angezeigt. Dies sollte beim Auswerten vor allem der Ankunfts- und Abfahrtszeiten beachtet werden. Es wird empfohlen, diese Werte auf 6 Sekunden / 0,1 min zu runden.

Bei in das FBS-Netz „einbrechend“ beginnendem oder „ausbrechend“ endendem Zuglauf werden die Übergabezeiten in das / aus dem Netz nicht angegeben, da dies Widersprüche zu *ocpType* (Halteart: *begin* oder *stop*?) und problematische Mitternachtsübergänge (*arrivalDay=-1* bei Mitternachtsübergang im Abfahrtsbahnhof) ergeben würde. Bei durchfahrend ein- oder ausbrechenden Zügen wird jedoch zwangsläufig auch die Übergabezeit angegeben.

Betriebliche und verkehrliche Züge (trains)

Das Zusammensetzen der Zugteile zu Zügen kann aus zwei verschiedenen Sichtweisen erfolgen: aus betrieblicher und aus verkehrlicher Sicht.

Das Zusammensetzen aus betrieblicher Sicht ergibt Züge, wie sie der Eisenbahner und das Eisenbahninfrastrukturunternehmen „sehen“, jedoch nicht unbedingt wie sie der Reisende sieht. Die Grundeigenschaft betrieblicher Züge ist, dass zu einem Zeitpunkt auf einem Streckengleis nur ein Zug fahren kann. Dieser Zug muss klar durch eine betriebliche Zugnummer definiert sein. Dieser Sichtweise kommt teilweise ein Sicherheitsaspekt zu (z. B. Verständigung zwischen den Betriebsstellen); sie wird daher als grundlegend empfunden.

Das Zusammensetzen aus verkehrlicher Sicht ergibt „Züge“, wie der Reisende sie i. d. R. sieht und wie sie z. B. in Aushang- oder Tabellenfahrplänen (Kursbuch) oder elektronischen Fahrplanmedien angegeben werden. Hierbei können zu einem Zeitpunkt auf (quasi) einem

Streckengleis mehrere (verkehrliche) Züge gleichzeitig „fahren“: In einem Tabellenfahrplan werden gekuppelte Züge durch zwei benachbarte Spalten mit gleichen Zeiten dargestellt. In einem Aushangfahrplan können zwei Zeilen mit gleicher Abfahrtszeit und gleichem Gleis vorhanden sein, wobei beide Zeilen unterschiedliche Ziele angeben, die Zugteile aber erst im späteren Zuglauf getrennt werden. Bedingt durch die Sichtweise trifft der Begriff „Zug“ hier nur im weitesten Sinne zu: So müssen „verkehrliche Züge“ nicht unbedingt ein Triebfahrzeug haben (z. B. Kurswagen).

Jeder Zugteil wird durch genau je einen betrieblichen und einen verkehrlichen Zug erfasst.

Jeder Zug (**train**-Eintrag) führt alle Zugteile, aus denen er besteht, in seiner Struktur **trainPartRef** mit *ref* und *position* auf. Ein Zug kann dabei entweder gleichzeitig (an einer Stelle seines Zuglaufs) oder nacheinander (in aufeinanderfolgenden Abschnitten seines Zuglaufs) aus verschiedenen Teilen bestehen. Der Wert *trainPartRef.position* zählt die Stellung im Zug an einer Stelle im Zuglauf. Es kann daher mehrere *trainPartRef*-Einträge mit gleicher *position* geben, wenn die entsprechenden Zugteile an verschiedenen Abschnitten im Zuglauf vorkommen.

Mehrere Zugteile an einer Stelle eines Zuglaufs kommen z. B. beim (abschnittsweisen) Verstärken oder Flügeln von Zügen vor. Aufeinanderfolgende Abschnitte eines Zuglaufs kommen z. B. vor, wenn unterwegs Verkehrstage wechseln.

Zu beachten ist, dass *trainPartRef.position* nicht unbedingt die tatsächliche Stellung im Zug angibt - durch unterschiedliche Verkehrstage der Zugteile können Teile mit niedrigerem *position*-Wert an bestimmten Tagen im Zugverband „fehlen“.

Betriebliche Züge (*operational trains*)

Betriebliche Züge sind in der RailML-Datei immer vorhanden, wenn Zugteile vorhanden sind. Sie sind in der Struktur **trains** enthalten und werden dort durch das Attribut **type=operational** von verkehrlichen Zügen unterschieden.

Attribut	Inhalt	optional	Bemerkungen
type	= <i>operational</i>		
trainNumber	innerbetriebliche Zugnummer; dient der eindeutigen Bezeichnung des Zuges (innerhalb eines Verkehrstags) beim Infrastrukturbetreiber		kann alphanumerisch sein
scope	dient der eindeutigen Bezeichnung des Zuges über alle Verkehrstage (Primärschlüsselbildung); mögliche Werte: <i>primary</i> , <i>secondary</i> , <i>secondaryStart</i> , <i>secondaryEnd</i> , <i>secondaryInner</i>		
scopeIndex	dient der eindeutigen Bezeichnung des Zuges über alle Verkehrstage (Primärschlüsselbildung); laufende Nummer innerhalb gleicher <i>trainNumber</i> und <i>scope</i> -Werte	x	abweichend von RailML; in Vorbereitung
trainPartRef.ref	Querverweis auf id eines Zugteils innerhalb der Struktur <i>timetable.trainParts</i>		
trainPartRef.position	Stellung des Zugteils im Zugverband, sofern ein Zug gleichzeitig aus mehreren Teilen besteht		Zählung ab 1
equipmentUsage.equipment	Zugbeeinflussung des Zuges	x	
brakeUsage...	Bremsangabe des Gesamtzuges (d. h. über alle eventuell gekuppelt verkehrenden Zugteile)		
brakeUsage.brakeType	= <i>compressedAir</i> bei Druckluftbremse = <i>vacuum</i> bei Saugluftbremse = <i>handBrake</i> bei handgebremstem Zug = <i>cableBrake</i> bei Seilzugbremse		

brakeUsage. airBrake- ApplicationPosition	nur bei Druck- oder Saugluftbremse: Grundbremsstellung =G, P oder R	x	
brakeUsage. meanDeceleration	nur wenn in FBS keine Brems Hundertstel definiert wurden: mittlere Bremsverzögerung in m/s ²	x	
brakeUsage. regularBrake- Percentage	im regulären (planmäßigen) Betrieb anrechenbare Brems Hundertstel (d. h. meist ohne Magnetschienenbremse)	x	
brakeUsage. emergencyBrake- Percentage	im Gefahrenfall (bei Not-/Schnellbremsungen) anrechenbare Brems Hundertstel (d. h. mit Magnetschienenbremse, jedoch ggf. ohne elektrische oder hydraulische Bremse)	x	
brakeUsage. brakePercentage	nur wenn in FBS identische Schnell- und Betriebsbrems Hundertstel definiert wurden	x	
brakeUsage. auxiliaryBrakes	Aufzählung der planmäßig vorhandenen Zusatzbremsen: Mg, Wb, E, H, ep	x	

Durch das Attribut **scope** werden mehrere Züge mit gleicher Zugnummer (die disjunkte Verkehrstage haben müssen) unterschieden. Man kann diese auch als alternative zeitliche Lagen oder „Varianten“ ein und desselben Zuges auffassen. Das Attribut **scopeIndex** dient der Unterscheidung von Einträgen mit gleicher Zugnummer und gleichem Scope-Attribut. In Anwendungsfällen, in denen die Zugnummer selbst immer eindeutig ist, haben *scope* und *scopeIndex* keine Bedeutung; hier ist immer *scope=primary*.

Den terminologischen Zusammenhang zwischen FBS-Bezeichnungen, RailML-Bezeichnungen und Bezeichnungen der DB Netz AG gibt folgende Tabelle wieder:

in RailML	in FBS	bei DB Netz
train.scope	Mehrfachzuglauf	„Ergänzungsfahrplan“
train.scope=primary	Hauptlauf (mit Index=1)	„Stammfahrplan“ oder „Stammzug“
	Hauptlauf (mit Index>1)	- nicht möglich -
train.scope=secondaryStart	Vornebenlauf	„Startflügel“
train.scope=secondaryEnd	Nachnebenlauf	„Zielflügel“
train.scope=secondaryInner	Zwischennebenlauf	„Doppelfahrplan“

In RailML fehlt der Index zu *train.scope* derzeit noch.

Verkehrliche Züge (*commercial trains*)

Verkehrliche Züge sind in der RailML-Datei optional vorhanden, d. h. können beim Export explizit ein- oder ausgeschlossen werden. Sie sind in der Struktur **trains** enthalten und werden dort durch das Attribut **type=commercial** von betrieblichen Zügen unterschieden.

Attribut	Inhalt	optional	Bemerkungen
type	= <i>commercial</i>		
name	Nummer zur Identifikation des verkehrlichen Zuges (z. B. Spaltentitel im Kursbuch - nicht unbedingt mit der betrieblichen Zugnummer identisch!)		kann alphanumerisch sein
trainPartRef.ref	Querverweis auf id eines Zugteils innerhalb der Struktur <i>timetable.trainParts</i>		
trainPartRef.position	Stellung des Zugteils im Zugverband, sofern ein Zug gleichzeitig aus mehreren Teilen besteht		Zählung ab 1

Der Wert *trainPartRef.position* gibt bei verkehrlichen Zügen die Stellung des Zugteils im eigentlichen Zugverband (des betrieblichen Zuges!) an. Es können in der Zählung daher

scheinbar Indizes fehlen, weil die entsprechenden Zugteile zu anderen verkehrlichen Zügen gehören.

Zugbeeinflussung, Bremsen usw. sind bei verkehrlichen Zügen nicht angegeben.

Umlaufpläne (rosterings)

Beim Export kann der Anwender wählen, ob und in welchem Umfang die Umlaufpläne aus FBS exportiert werden sollen. Jeder Umlaufplan von FBS führt zu einem **rostering**-Element.

Ein Umlaufplan gilt i. d. R. nur für *eine* Fahrzeugbaureihe und ggf. nur für einen begrenzten (Teil-)Zeitraum der Fahrplanperiode. Es kann daher mehrere Umlaufpläne für die gleiche Fahrzeugbaureihe geben. Die Teilzeiträume müssen nicht disjunkt sein, sondern können sich überlappen. I. d. R. ist ein **Grund- oder Regelumlauf** für das Fahrplanjahr vorhanden, welcher ununterbrochen ohne konkreten Datumsbezug gilt. Dieser kann für bestimmte Zeiträume (Weihnachten, Ostern, Ferien, Bauarbeiten) durch **Sonderumlaufpläne** überschrieben werden. Das Überschreiben von Umlaufplänen ist in der Praxis notwendig, da zu Beginn eines Fahrplanjahres i. d. R. noch nicht in jedem Falle feststeht, wann der Grundumlaufplan nicht gefahren werden kann (z. B. wann Bauarbeiten sein werden).

In RailML sind **offene** und **geschlossene Umläufe** möglich. Geschlossene Umläufe können (theoretisch beliebig oft) wiederholt werden, da auf die letzte Fahrt im Umlauf wieder die erste Fahrt folgt. Offene Umläufe hingegen müssen nicht wiederholt werden können - bei ihnen folgt auf die letzte Fahrt entweder ein anderer Umlaufplan oder das Ende der Fahrplanperiode. Typischer Weise sind geschlossene Umläufe solche in einem Grund- oder Regelumlaufplan und offene Umläufe solche in einem Sonderumlaufplan. Im Allgemeinen kann man davon ausgehen, dass geschlossene Umläufe über einen längeren Zeitraum gelten (daher die Wiederholung, um nicht jeden Tag des längeren Zeitraums individuell abzubilden), während offene Umläufe über einen kürzeren Zeitraum gelten, in dem keine Wiederholungen möglich oder notwendig sind. Eine typische Zeitgrenze ist eine Woche, denn spätestens nach einer Woche sind Umlaufwiederholungen üblich. Offene Umläufe sind daher meist (fast immer) kürzer als eine Woche.

Wichtig für RailML ist, dass auf Grund der Möglichkeit offener Umläufe zu bestimmten Fahrten keine nachfolgende Fahrten angegeben sind. Zur begrifflichen Eindeutigkeit werden im Folgenden als *offene* Umlaufpläne solche bezeichnet, bei denen nicht an allen Fahrten eine Folgefahrt angegeben ist. *Geschlossene* Umläufe sind demzufolge solche, bei denen an allen Fahrten eine Folgefahrt angegeben ist - ungeachtet ihrer Gültigkeitsdauer und der Frage, ob sie tatsächlich wiederholt werden sollen.

Ein Umlaufplan kann mehrere **Umlaufgruppen** enthalten. Umlaufgruppen sind eine Eigenart insbesondere von Eisenbahn-Umlaufplänen: Jede Umlaufgruppe stellt für sich einen eigentlichen Umlauf dar; daher kann es sein, dass in anderen terminologischen Zusammenhängen das als *Umlauf* bezeichnet wird, was in der (deutschsprachigen) Eisenbahn-Fachwelt nur eine *Umlaufgruppe* ist. Jede Umlaufgruppe könnte theoretisch in einem eigenen Umlaufplan abgebildet werden. Dies wird in der Praxis aber oft so nicht vorgenommen, weil man u. U. bemüht ist, *alle* Fahrzeuge einer Baureihe in *einem* Umlaufplan darzustellen.

Wagenzugumläufe können für einzelne Wagen oder Wagengruppen vorhanden sein. Triebfahrzeugumläufe gelten für Einzelfahrzeuge (wobei in diesem Sinne jedoch *betrieblich* nicht trennbare Einheiten als Einzelfahrzeug gelten, z. B. ein Triebzug der BR 403 oder eine Kombination aus BR 628+928).

Umlauf-Kopfangaben (*rostering*)

Jeder Umlaufplan wird in einem **rostering**-Element abgebildet. Die Attribute des *rostering*-Elements enthalten allgemeine Angaben zum Umlaufplan:

Attribut	Inhalt	optional	Bemerkungen
name	Bezeichnung des Umlaufplans durch den Anwender in FBS (enthält meist Fahrzeugbezeichnung)		
description	Überschrift des Umlaufplans durch den Anwender in FBS sowie Fahrzeugbezeichnung	x	
vehicleRef	Querverweis auf id des Fahrzeugs des Umlaufplans im der Struktur <i>rollingstock.vehicles</i> (nur bei Umläufen von Einzelfahrzeugen vorhanden)	x	
formationRef	Querverweis auf id des Wagenzugs des Umlaufplans im der Struktur <i>rollingstock.formations</i> (nur bei Wagenzugumläufen vorhanden)	x	
defaultPreProcessingTime	definiert die (Standard-)Vorbereitungszeit, die gilt, wenn keine abweichende Angabe bei <i>blockPartSequence</i> erfolgt		
defaultPostProcessingTime	definiert die (Standard-)Nachbereitungszeit, die gilt, wenn keine abweichende Angabe bei <i>blockPartSequence</i> erfolgt		

Fahrten und Dienste (*blockParts*)

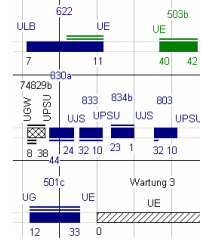
Die Struktur **blockParts** enthält je einen **blockPart**-Eintrag für jeden im Umlaufplan benötigten Zugteil oder Dienst. Ein Dienst (hier im Gegensatz zu Zugteil) kann z. B. Tanken, Wartung, Reinigung, Rangierdienst usw. sein und unterscheidet sich i. d. R. dadurch von den Zugteilen, dass die Betriebsstelle nicht gewechselt wird.

Attribut	Inhalt	optional	Bemerkungen
begin	Anfangszeit des Dienstes bzw. Abfahrtszeit des Zugteils an dessen ersten Bahnhof		
end	Endzeit des Dienstes bzw. Ankunftszeit des Zugteils an dessen letzten Bahnhof		
endDay	Anzahl der Mitternachtsübergänge (Folgetage) zwischen <i>begin</i> und <i>end</i>	x	default=0
startOcpRef	Querverweis auf id der Betriebsstelle des Anfangs des Dienstes oder Zugteils in der Struktur <i>infrastructure.operatingControlPoints</i>		
endOcpRef	Querverweis auf id der Betriebsstelle des Endes des Dienstes oder Zugteils in der Struktur <i>infrastructure.operatingControlPoints</i>		
trainPartRef	Querverweis auf id des Zugteils in der Struktur <i>timetable.trainParts</i> (sofern dem <i>blockPart</i> ein Zugteil zugrunde liegt)	x	
mission	= <i>timetable</i> bei Verweisen auf Zugteile = <i>fullRun</i> oder <i>emptyRun</i> bei Fahrten, die vom Anwender manuell in den Umlaufplan eingegeben wurden und daher nicht auf Zugteile verweisen = <i>shunting</i> bei Rangierdiensten = <i>maintenance</i> bei Wartungsdiensten = <i>standBy</i> bei Bereitschafts- und Reservediensten = <i>preheating</i> bei Vorheizdiensten = <i>refuel</i> bei Auffüllen von Vorräten (Tanken usw.)		

operatingPeriodRef	Querverweis auf id der Verkehrstagerregelung des Dienstes in der Struktur <i>timetable.operatingPeriods</i> der RailML-Datei; beschreibt alle Tage, an denen der Dienst im Umlauf vorkommt	x	nicht bei Zugteilen (d. h. nicht wenn <i>trainPartRef</i> angegeben ist)
runLength	Entfernung, die während des Dienstes angerechnet oder während der Fahrt zurückgelegt wird in km (mit drei Nachkommastellen)		

Fahrtengruppen (blocks)

Die Struktur **blocks** enthält für zusammenhängende Gruppen oder Abfolgen aus *blockParts* einen **block**-Eintrag. Jeder *blockPart* kommt in mindestens einem *block* vor. Im einfachsten (und recht häufig vorkommenden) Fall enthält ein *block* genau einen *blockPart*. Die Gruppierung von *blockParts* zu *blocks* ist in RailML im Allgemeinen willkürlich (nicht weiter spezifiziert); bei aus FBS exportierten Umläufen sind jedoch insbesondere die unmittelbar zu einem (betrieblich durchgehenden) Zug verketteten *blockParts* zu einem *block* zusammengefasst. Damit entsprechen die *blocks* im Wesentlichen den in grafischen Umlaufplänen typischer Weise dargestellten Blöcken (Rechtecken).



Attribut	Inhalt	optional	Bemerkungen
name	Titel des Blocks im FBS-Umlaufplan; i. d. R. bestehend aus Zuggattung/Produkt + Zugteilnummer + Verkehrstagekürzeln		
blockPartSequence.sequence	laufende Nummer der <i>blockPartSequence</i> -Elemente, gezählt ab 1		
blockPartSequence.blockPartRef.ref	Querverweis auf id eines Elements der Struktur <i>blockParts</i> des Umlaufplans		genau 1x je <i>blockPartSequence</i> vorhanden *1
blockPartSequence..preProcessingTime	Vorbereitungszeit vor Beginn des Block sofern abweichend von <i>rostering.defaultPreProcessingTime</i>	x	*2
blockPartSequence..postProcessingTime	Nachbereitungszeit nach Ende des Block sofern abweichend von <i>rostering.defaultPostProcessingTime</i>	x	*2

*1 Bei aus FBS exportierten RailML-Dateien kann ein *block* nicht gleichzeitig aus mehreren *blockParts* bestehen (d. h. durch ein Fahrzeug können nicht gleichzeitig mehrere Dienste oder Züge gefahren werden). Daher enthält jedes *blockPartSequence*-Element immer nur genau ein *blockPartRef*-Element.

*2 Die in Umläufen üblicherweise für Fahrten oder Dienste reservierte / in Anspruch genommene Zeit ist um die Vor- und Nachbereitungszeiten größer als die durch *begin* und *end* des jeweils ersten bzw. letzten *blockParts* definierte Zeitspanne.

blocks dienen auch dazu, die Mitternachtsgrenze im Sinne eines Betriebstages zu überwinden („Fahrten nach Mitternacht zählen zum Vortag“), indem Fahrten nach Mitternacht mit denen vor Mitternacht zu einem Block zusammengefasst werden. Da sich *operatingPeriodRef* eines Blocks bzw. eines *circulation*-Elements nur auf den Anfangstag bezieht, kann der Block in jeden beliebigen Folgetag hineinreichen - unabhängig davon, um welche Tagessart es sich handelt.

Umläufe (*circulations*)

Die eigentliche Verkettung (Reihenfolge) der im Umlauf vorkommenden Fahrten und Dienste (im Folgenden allgemein als „Blöcke“ bezeichnet) ist in der **circulations**-Struktur enthalten. Es ist je ein **circulation**-Element für jeden Block an jedem Umlaufstag enthalten. Durch die Attribute **nextBlockRef** und **nextOperatingPeriodRef** jedes *circulation*-Elements ergibt sich die Verkettung der Blöcke und damit die eigentliche Umlaufabfolge. Das Attribut **operating-PeriodRef** jedes *circulation*-Elements gibt (indirekt) an, ob und wie oft ein Umlaufstag wiederholt werden muss, um die gesamte Gültigkeit des Umlaufplans abzudecken.

Bei aus FBS exportierten Umlaufplänen verweist jede *circulation.operatingPeriodRef* immer nur auf gleiche Wochentage (z. B. nur auf Montage oder nur auf Sonntage) - in FBS sind Umläufe immer wochentagsgenau. (Dies muss jedoch bei RailML-Dateien im Allgemeinen nicht immer so sein.) Wenn ein *circulation.operatingPeriodRef* z. B. auf drei aufeinanderfolgende Montage verweist, dann soll der Umlauf offensichtlich diese drei Wochen lang wiederholt werden - es ist dann immer auch ein geschlossener Umlauf. Im Grund- oder Regelumlauf einer Fahrzeugbaureihe sind zunächst i. d. R. Verweise für alle Wochentage der Fahrplanperiode enthalten. Dies ist die theoretische Ausgangssituation jeder Fahrplanperiode - ein ununterbrochen geltender Umlauf. Da einige „unterbrechend wirkende“ Ereignisse - z. B. Weihnachten, Ostern oder Ferien - jedoch bereits von vornherein feststehen, kann es ebenso sein, dass der Regelumlauf von Anfang an Unterbrechungen enthält oder von anderen Umläufen überschrieben wird.

Attribut	Inhalt	optional	Bemerkungen
blockRef	aktueller Block: Querverweis auf id eines Elements der Struktur <i>blocks</i> des Umlaufplans		
operatingPeriodRef	aktueller Umlaufstag: Querverweis auf id einer Verkehrstagerregelung der Struktur <i>timetable.operatingPeriods</i>		
vehicleIdx	laufende Nummer des Fahrzeugs im Umlauf		Zählung ab 1
groupIdx	laufende Nummer der Umlaufgruppe	x	Zählung ab 1
nextBlockRef	nachfolgender Block: Querverweis auf id eines Elements der Struktur <i>blocks</i> des Umlaufplans	x	nicht bei letzten Fahrten des jeweiligen Fahrzeugs in offenen Umläufen
nextOperatingPeriodRef	nachfolgender Umlaufstag: Querverweis auf id einer Verkehrstagerregelung der Struktur <i>timetable.operatingPeriods</i>	x	

Die Attribute *vehicleIdx* und *groupIdx* sind redundant, da sie genau genommen aus *nextBlockRef* und *nextOperatingPeriodRef* rekonstruiert werden könnten. Sie erleichtern das Auswerten jedoch deutlich, da der Zusammenhang zwischen *nextBlockRef*, *nextOperatingPeriodRef*, *vehicleIdx* und *groupIdx* u. U. nicht trivial ist:

- Bei einem offenen Umlauf ist die Anzahl der Fahrzeuge gleich der Anzahl der *circulation*-Elemente ohne *nextBlockRef* - wenn ein Fahrzeug durch einen offenen Übergang den Umlauf „verlässt“, muss für die folgende Fahrt ein anderes Fahrzeug verwendet werden.
- Bei einem geschlossenen Umlauf ist die Anzahl der Fahrzeuge gleich der Anzahl der *circulation*-Elemente, bei denen *nextOperatingPeriodRef* in der Zeit „zurückspringt“: Da ein Fahrzeug nicht gleichzeitig verschiedene Fahrten durchführen kann, enthält der zeitliche Rücksprung implizit das nächste Fahrzeug.
- Ein Umlauf ist geschlossen, wenn jedes *circulation*-Element die Attribute *nextBlockRef* und *nextOperatingPeriodRef* hat.
- Da es bei einem offenen Umlauf keine Wiederholungen gibt, gibt es auch nie zeitliche Rücksprünge von *nextOperatingPeriodRef*.
- Wenn ein *circulation*-Element durch seine Attribute *nextBlockRef* und *nextOperatingPeriodRef* auf einen Block verweist, der bereits durch ein früheres *circulation*-Element erfasst

wurde, ist die Umlaufgruppe beendet (geschlossen). Wenn an dieser Stelle noch weitere Blöcke (d. h. unerfasste) im Umlauf sind, wird eine weitere Umlaufgruppe begonnen. Das nächste *circulation*-Element wird einen um eins erhöhten Gruppenindex haben.

- Die Gruppenindizes sind nur bei geschlossenen Umläufen angegeben und nur dann, wenn mehr als eine Gruppe im Umlauf vorhanden ist. Bei offenen Umläufen ist der Gruppenindex irrelevant (da ohnehin keine Wiederholungen stattfinden, ist es eine rein philosophische Ansicht, ob alle Fahrzeuge eine Gruppe bilden oder jedes Fahrzeug eine eigene Gruppe).