Allgemeines zu Datumsbezügen, Gültigkeitsperiode und Feiertagen

Im Falle des Datenaustauschs von konkreten Fahrplänen verfügen die in einer RailML-Datei enthaltenen Fahrplandaten i. d. R. auch über einen Datumsbezug. Der Datumsbezug wird im Element **timetablePeriod** und optional ergänzend in den Elementen **operatingPeriod** hergestellt.

Typischer Weise enthält ein Element timetablePeriod
- Beginn und Ende der Gültigkeitsperiode in den Attributen startDate und endDate,
- eine optionale Aufzählung von Feiertagen innerhalb der Gültigkeitsperiode.

```
<timetablePeriod id='ttp_2020_21' name='2020/21' startDate='2020-12-13' endDate='2021-12-11'>
  <holidays>
    <holiday holidayDate='2020-12-25'/>
    <holiday holidayDate='2020-12-26'/>
    <holiday holidayDate='2021-01-01'/>
    <holiday holidayDate='2021-04-02'/>
    <holiday holidayDate='2021-04-04'/>
    <holiday holidayDate='2021-04-05'/>
    <holiday holidayDate='2021-05-01'/>
    <holiday holidayDate='2021-05-13'/>
    <holiday holidayDate='2021-05-23'/>
    <holiday holidayDate='2021-05-24'/>
    <holiday holidayDate='2021-10-03'/>
    <holiday holidayDate='2021-10-31'/>
    <holiday holidayDate='2021-11-17'/>
  </holidays>
</timetablePeriod>
```

Im einfachsten Fall enthält nur das eine vorhandene Element timetablePeriod einen Datumsbezug über startDate+endDate und ggf. noch über holidays. Die Elemente operatingPeriod können ohne weitere Angabe eines konkreten Datums beliebige Wochentage aus dem Gültigkeitsbereich referenzieren (s. nachfolgende Beispiele).

Die Angabe der Feiertage ist die Bezugsbasis für die Unterelemente operatingDayDeviance der Elemente opertingPeriod. Die Elemente holidays und operatingDayDeviance sind daher zunächst nur gemeinsam sinnvoll anzuwenden. Wenn keine Elemente operatingDayDeviance angegeben sind, würden holidays nicht ausgewertet. Es ist aber dennoch denkbar, die Feiertage nur der Vollständigkeit halber anzugeben.

Eine RailML-Datei kann, muss aber nicht über eine Gültigkeitsperiode verfügen. Eine RailML-Datei kann auch mehrere Gültigkeitsperioden beinhalten, wobei sich jede operatingPeriod und damit auch jeder Zug und Zugteil immer nur auf eine Gültigkeitsperiode gleichzeitig beziehen kann.

RailML-Dateien ohne Gültigkeitsperiode haben entweder kein Element timetablePeriod oder nur Elemente timetablePeriod ohne startDate+endDate. Wenn keine Gültigkeitsperiode definiert ist, dürfen die Elemente operatingPeriod ebenfalls nicht über die Attribute startDate, endDate und bitMask und auch nicht über Unterelemente specialService verfügen.

StartDate und endDate dürfen nur gemeinsam angegeben werden, nicht nur eines der beiden, d. h. es darf keine „offenen“ Perioden geben. Die Differenz der Attribute startDate und endDate definiert die Länge der Bitmasken (Attribut bitMask) der Elemente operatingPeriod.

Abbildung spezieller Verkehrstageregelungen ohne Datumsbezug

Verkehrstageregelungen (Saisonierungen) werden im RailML-Element *timetable.operatingPeriods* codiert. In einer Verkehrstageregelung (*operatingPeriod*) sind die einzelnen Tage auf zwei redundante Arten codiert: Einmal im Attribut *bitMask* über die gesamte Periode sowie in den Unterelementen *operatingDay* und *specialService* aufgeschlüsselt nach konkreter Eingabe in Regel- und Sonderverkehrstage. Der Grund für diese Dopplung ist, dass sich die Bitmaske zwar technisch relativ gut weiterverarbeiten, jedoch nicht eindeutig in eine Nutzereingabe rückverwandeln lässt. (D. h. es gibt mehrere Eingaben, die zur gleichen Bitmaske führen.) Für einige Anwendungen ist es wichtig, die Angaben genau so zu übermitteln, wie der Anwender sie eingegeben hat. Im Übrigen kann es relativ aufwändig sein, aus einer Bitmaske eine Trennung in Regel- und Sonderverkehrstage vorzunehmen (es kann z. B. eine Korrelationsanalyse erfordern, die Eingabe „W[Sa], nicht 24., 31.12.; auch 17.11." aus einer Bitmaske wiederherzustellen).

Der Beschreibung mit *operatingDay*-Elementen kommt besondere Bedeutung zu, wenn z. B. bei „strategischen" (langfristigen) Fahrplänen keine Gültigkeitsperiode definiert ist (keine *timetablePeriod* bzw. ohne *startDate*+*endDate*; s. a. vorheriges Kapitel).

Das Element *operatingDay* enthält ein Unterelement **operatingDayDeviance**, dessen Bedeutung an den folgenden Beispielen erläutert werden soll:

Im Allgemeinen gibt das Element *operatingDayDeviance* an, das die Verkehrstageregelung abweichend von *operatingCode* verkehrt oder nicht verkehrt, wenn ein Wochentag in der unter *holidayOffset* angegebenen Relation zu einem Feiertag steht. Das Attribut *ranking* definiert, welches *operatingDayDeviance*-Element gilt, falls es mehrere gleichzeitig zutreffende gibt. Beispielsweise könnte ein Wochentag gleichzeitig *Feiertag* und *ein Tag vor einem Feiertag* sein (in Deutschland z. B. der 25.12.).

Beispiel 1

Die Verkehrstageregelung W[Sa] = Mo-Fr[S] = „Montag bis Freitag, jedoch nicht an Feiertagen" wird wie folgt abgebildet:

```
<operatingPeriod id='…' name='W[Sa]' …>
  <operatingDay operatingCode='1111100'>
    <operatingDayDeviance operatingCode='0000000' holidayOffset='0'/>
  </operatingDay>
</operatingPeriod>
```

Die Angabe `operatingCode='1111100'` bedeutet, dass die Regelung an den ersten fünf Wochentagen (=Montag bis Freitag) verkehrt. Die Angabe `operatingCode='0000000' holidayOffset='0'` bedeutet, dass die Regelung abweichend davon *nicht* verkehrt (=0000000), wenn der jeweilige Tag auf einen Feiertag fällt (holidayOffset=0 - „null Tage Versatz zu einem Feiertag" hat).

Beispiel 2

Die Verkehrstageregelung S = „Sonntag und alle Wochenfeiertage" wird wie folgt abgebildet:

```
<operatingPeriod id='…' name='S' …>
  <operatingDay operatingCode='0000001'>
    <operatingDayDeviance operatingCode='1111111' holidayOffset='0'/>
  </operatingDay>
</operatingPeriod>
```

Die Angabe `operatingCode='0000001'` bedeutet, dass die Regelung nur an Sonntagen verkehrt. Die Angabe `operatingCode='1111111' holidayOffset='0'` bedeutet, dass die Regelung abweichend davon an allen Tagen verkehrt, die auf einen Feiertag fallen.

Beispiel 3
Die Verkehrstageregelung vS = „Werktage vor Sonn- und Feiertagen“ wird wie folgt abgebildet:

```
<operatingPeriod id='…' name='vS' …>
  <operatingDay operatingCode='0000010'>
    <operatingDayDeviance operatingCode='1111110' holidayOffset='-1' ranking='2'/>
    <operatingDayDeviance operatingCode='0000000' holidayOffset='0' ranking='1'/>
  </operatingDay>
</operatingPeriod>
```

Die Angabe `operatingCode='0000010'` bedeutet, dass die Regelung nur an Sonnabenden verkehrt (=vor Sonntagen). Die Angabe `operatingCode='1111110' holidayOffset='-1'` bedeutet, dass die Regelung auch an Montagen bis Sonnabenden verkehrt, sofern sie einen Tag vor einem Feiertagen liegen (holidayOffset=-1). Die Angabe `operatingCode='0000000' holidayOffset='0'` bedeutet, dass die Regelung nicht an Feiertagen verkehrt. Falls ein Tag gleichzeitig Feiertag und ein Tag vor einem Feiertag ist (in Deutschland z. B. 25.12.), verkehrt die Regelung nicht, denn es gilt dann die holidayOffset=0-Angabe. Entsprechend RailML-Dokumentation (http://wiki.railml.org/index.php?title=TT:operatingDayDeviance) gilt im Zweifelsfall die Angabe mit niedrigerem *ranking*-Wert.

Insgesamt sind die drei Zeilen (von oben nach unten) in etwa wie folgt zu interpretieren:

Zeile 1:	„verkehrt an Samstagen,...
Zeile 2:	... sowie an Montagen bis Samstagen wenn vor einem Feiertag,...
Zeile 3:	...jedoch nicht an Feiertagen“

Beispiel 4
Ein Zug beginnt seine Fahrt an Sa+S = „Samstagen sowie Sonn- und Feiertagen“ und verkehrt im Laufe der Fahrt über Mitternacht. Die Rückfahrt nach Mitternacht muss zwangsläufig an „Folgetagen von Sa+S“ stattfinden. In RailML würde Sa+S wie folgt abgebildet:

```
<operatingPeriod id='…' name='Sa+S' …>
  <operatingDay operatingCode='0000011'>
    <operatingDayDeviance operatingCode='1111111' holidayOffset='0'/>
  </operatingDay>
</operatingPeriod>
```

Die Folgetage von Sa+S sind demzufolge (alle Bitmasken um ein Bit nach rechts rotiert):

```
<operatingPeriod id='…' name='Folgetage von Sa+S' …>
  <operatingDay operatingCode='1000001'>
    <operatingDayDeviance operatingCode='1111111' holidayOffset='+1'/>
  </operatingDay>
</operatingPeriod>
```

Für diese Regelung gibt es im deutschsprachigen Raum keine standardisierte Kurzbezeichnung. Sie ist nicht identisch mit So+nS, denn letztere würde abgebildet durch:

```
<operatingPeriod id='…' name='So+nS' …>
  <operatingDay operatingCode='1000001'>
    <operatingDayDeviance operatingCode='1111110' holidayOffset='+1' ranking='2'/>
    <operatingDayDeviance operatingCode='0000000' holidayOffset='0' ranking='1'/>
  </operatingDay>
</operatingPeriod>
```

Mit RailML lassen sich auch Folge- und Vortage „sauber“ abbilden, was allein mit deutschen Kurzbezeichnungen so nicht möglich ist. Solange eine Fahrplanperiode definiert ist, ließen sich solche Folge- und Vortagesregelungen durch Verwendung von Sonderverkehrstagen (*specialService* in RailML) umschreiben. Wenn jedoch keine Fahrplanperiode zur Verfügung steht, ist die Lösung mit *operatingDayDeviance* die einzig mögliche. Dies betrifft insbesondere die bei Wettbewerbsverfahren (Ausschreibungsfahrplänen ohne konkrete Fahrplanperiode) im Vor- und Nachlauf um Wochenenden recht häufig vorkommenden Regelungen „Arbeitstage vor Ⓐv© (Arbeitstage vor arbeitsfreien Tagen), Ⓐn© (Arbeitstage nach arbeitsfreien Tagen), ©vⒶ (arbeitsfreie Tage vor Arbeitstagen) und ©nⒶ (arbeitsfreie Tage nach Arbeitstagen).

Abbildung spezieller Verkehrstageregelungen mit Datumsbezug

Hierbei geht es um solche Fälle wie „verkehrt nicht am 24. und 31.12." oder „nur vom 01.07.-31.08.". Voraussetzung hierfür ist, dass auch eine zugehörige Gültigkeitsperiode (*timetablePeriod* mit angegebenem *startDate+endDate*) definiert ist.

Für diese Fälle können folgende zusätzlichen Elemente und Attribute verwendet werden:
- Attribute *startDate+endDate* des Elements *operatingDay,*
- Unterelemente *specialService* des Elements operatingPeriod.

Die Angabe der Attribute **startDate** und **endDate** in *operatingDay* beschränkt die Anwendung der Bildungsregel des operatingDay-Elements auf diesen Datumsbereich. Beide sind nur gemeinsam zu benutzen und dürfen nur Tage innerhalb der Gültigkeitsperiode der *timetablePeriod* referenzieren.

Bei **specialService** handelt es sich um eine Aufzählung von **Ausnahmeverkehrstagen**, an denen von den durch die *operatingDay*-Elemente definierten Bildungsregeln abgewichen wird. Dabei steht das Attribut type mit Ausprägung
=*include* für Zusatzverkehrstage,
=*exclude* für Ausfalltage.
Die Attribute *startDate* und *endDate* definieren den ersten und letzten Gültigkeitstag der Ausnahme, sind nur gemeinsam zu benutzen und dürfen nur Tage innerhalb der Gültigkeitsperiode der *timetablePeriod* referenzieren. Das Attribut *singleDate* ist im Fall *startDate=endDate* anstatt dieser beiden zu verwenden.

Beispiel 1: verkehrt nur vom 14.12. bis 20.12.

```
<operatingPeriod id='…' name='nur 14.12.-28.12.' timetablePeriodRef='ttp_2020_21'
        bitMask='01111111111111100000…000'>
    <operatingDay operatingCode='1111111' startDate='2020-12-14' endDate='2020-12-28'/>
</operatingPeriod>
```

Beispiel 2: verkehrt nicht am 25.12. und nicht am 01.01.

```
<operatingPeriods>
  <operatingPeriod id='…' name='täglich; nicht 25.12.; 1.1.'
       timetablePeriodRef='ttp_2020_21'
       bitMask='11111111111101111110111…111'>
    <operatingDay operatingCode='1111111' startDate='2020-12-13' endDate='2021-12-11'/>
    <specialService type='exclude' singleDate='2020-12-25'/>
    <specialService type='exclude' singleDate='2021-01-01'/>
  </operatingPeriod>
</operatingPeriods>
```

Beispiel 3: verkehrt vom 13.12. bis 31.01. nur Sa und vom 01.07.-31.08. täglich; auch am 25.12. und 01.01, nicht am 15.08."

(Verzicht auf Darstellung von name, bitMask usw.)

```
<operatingPeriod id='…' timetablePeriodRef='ttp_2020_21'>
  <operatingDay operatingCode='0000010' startDate='2020-12-13' endDate='2021-01-31'/>
  <operatingDay operatingCode='1111111' startDate='2020-07-01' endDate='2021-08-31'/>
  <specialService type='include' singleDate='2020-12-25'/>
  <specialService type='include' singleDate='2021-01-01'/>
  <specialService type='exclude' singleDate='2021-08-15'/>
</operatingPeriod>
```

Falls mehrere *operatingDay*-Elemente in einem *operatingPeriod*-Element vorkommen, müssen diese durch ihre *startDate-endDate*-Bereiche oder ihre *operatingCode*-Bitmaske immer disjunkt sein.