# Datenformat Planungsauftrag (Solver Input) V1.0



Ein Planungsauftrag besteht aus mehreren Files. Minimal benötigt sind

- kundenfahrten.csv
- fahrzeuggruppen.csv
- relationen.csv

alle anderen sind optional.

## kundenfahrten.csv

Kundenfahrten, die durch Rollmaterial abzudecken sind.

| Attribut | Beschreibung | Datentyp |
|---|---|---|
| id | eindeutiger Identifikator (zBsp. Betriebstag + TrassenId) | string |
| zugnummer | Zugnummer für die fachliche Identifikation der Kundenfahrten in Darstellungen etc. | string |
| betriebstag | Betriebstag für die fachliche Identifikation der Kundenfahrten | date as string |
| bpAb | Betriebspunkt-Abkürzung Abfahrtsort | string |
| bpAn | Betriebspunkt-Abkürzung Zielort | string |
| zeitAb | Abfahrtszeit | datetime as string |
| zeitAn | Ankunftszeit | datetime as string |
| richtungscodeAb | Richtungscode Ausfahrt Abfahrtsbahnhof<br><br>Jeder Bahnhof hat eine Seite 0 und eine Seite 1. Definiert in den Topologiedaten von CERES. Der richtungscodeAb definiert, beim Abfahrtsort der Kundenfahrt, auf welche Seite ausgefahren wird. Dient der Unterscheidung von Weiterfahrt bei gleicher Fahrtrichtung (Bahnhofseite wechselt zwischen an und ab) oder Wende (Bahnhofseite für an und ab ist dieselbe). | integer (0 or 1) |
| richtungscodeAn | Richtungscode Ankunft Zielbahnhof | integer (0 or 1) |
| distanzInKm | Streckenlänge der Fahrt in km | float > 0.0 |
| bedarf | Anzahl benötigter Fahrzeuggruppen (Rollmaterial-Einheiten) für die Kundenfahrt | integer >= 1 |

Konsistenzregeln

- Valide Datentypen gemäss Tabelle oben. Keine leeren Attribute erlaubt
- Eindeutigkeit der Id über alle Kundenfahrten
- Zeitliche Konsistenz: zeitAb < zeitAn  (strikt ungleich, um Probleme mit uneindeutige Sortierungen von Leistungen zu vermeiden)

## wartungsfenster.csv

Wartungsfenster, die in der Teilneuplanung einzuplanen sind

| Attribut | Beschreibung | Datentyp |
|---|---|---|

| | | |
|---|---|---|
| id | eindeutiger Identifikator | string |
| bp | BP wo das Wartungsfenster verfügbar ist | string |
| startZeit | Zeit ab wann die Wartung stattfindet | datetime as string |
| endZeit | Zeit bis wann die Wartung dauert | datetime as string |

Konsistenzregeln

- Valide Datentypen gemäss Tabelle oben. Keine leeren Attribute erlaubt
- Eindeutigkeit der Id über alle Wartungsfenster
- Zeitliche Konsistenz: startZeit< endZeit

# fahrzeuggruppen.csv

Verfügbare Fahrzeuggruppen (verplanbare Rollmaterial-Einheiten)

| Attribut | Beschreibung | Datentyp |
|---|---|---|
| id | eindeutiger Identifikator | string |
| startZeit | Zeit ab wann Fahrzeuggruppe einsetzbar (optional) | datetime as string / leer |
| startBp | BP an welchem die Fahrzeuggruppe startet (optional) | string / leer |
| kmSeitWartung | Zurückgelegte km seit letzter Wartung | float >= 0.0 / leer (gilt als 0.0) |
| dauerSeitWartung | Dauer seit letzter Wartung | duration, non-negative / leer (gilt als 0) |

Konsistenzregeln

- Valide Datentypen gemäss Tabelle oben. Leere Attribute nur erlaubt wo erwähnt.
- Eindeutigkeit der Id über alle Fahrzeuggruppen
- Falls startZeit oder startBp gesetzt (nicht leer) ist, müssen beide gesetzt sein
- Falls dauerSeitWartung nicht leer ist, darf auch startZeit nicht leer sein
- kmSeitWartung <= config.ivog.distance
- dauerSeitWartung <= config.ivog.duration

# endpunkte.csv

Zur Abgrenzung des Planungsausschnitts vorgegebene Endpunkte, bei denen Leistungsketten enden und wieder an die bisherige Planung (Referenzplan) anknüpfen können.

| Attribut | Beschreibung | Datentyp |
|---|---|---|
| id | eingeutiger Identifikator | string |
| endZeit | Zeit bis wann Leistungskette endet | datetime as string |
| endBp | BP an welchem die Leistungskette endet | string |
| kmBisWartung | Zurückzulegende km bis zur nächsten Wartung | float >= 0.0 / leer (gilt als 0.0) |
| dauerBisWartung | Dauer bis zur nächsten Wartung | duration, non-negative / leer (gilt als 0) |

Konsistenzregeln

- Valide Datentypen gemäss Tabelle oben. Leere Attribute nur erlaubt wo erwähnt.
- Eindeutigkeit der Id über alle Endpunkte
- Es gibt maximal so viele Endpunkte wie Fahrzeuggruppen
- kmBisWartung <= config.ivog.distance
- dauerSeitWartung <= config.ivog.duration

# relationen.csv

Relationen für mögliche Betriebsfahrten, die der Solver in die Leistungsketten einplanen kann, um geographische Lücken zu schliessen.

Relationen sind aggregierte Start-Ziel Verbindungen, ohne Betrachtung von Teilabschnitten. Siehe auch Topologie Granularität.

| Attribut | Beschreibung | Datentyp |
|---|---|---|
| bpAb | Betriebspunkt-Abkürzung Abfahrtsort | string |
| bpAn | Betriebspunkt-Abkürzung Ankunftsort | string |
| distanzInKm | Streckenlänge der Fahrt in km | float > 0.0 |
| fahrtdauer | Dauer der Fahrt | duration as string > 0 |
| richtungscodeAb | Richtungscode Ausfahrt Abfahrtsbahnhof | integer (0 or 1) |
| richtungscodeAn | Richtungscode Ankunft Zielbahnhof | integer (0 or 1) |

Konsistenzregeln

- Valide Datentypen gemäss Tabelle oben. Keine leeren Attribute erlaubt
- Eindeutigkeit der bpAb, bpAn Kombination über alle Relationen
- bpAn nicht gleich bpAb

# sperren.csv

Temporär gesperrte Relationen für Betriebsfahrten.

Bei einer Sperre einer Teilstrecke müssen alle betroffenen Relationen gesperrt werden. Siehe auch Topologie Granularität.

| Attribut | Beschreibung | Datentyp |
|---|---|---|
| bpAb | Betriebspunkt-Abkürzung Abfahrtsort | string |
| bpAn | Betriebspunkt-Abkürzung Ankunftsort | string |
| zeitVon | Start der Sperre | datetime as string |
| zeitBis | Ende der Sperre | datetime as string |

Konsistenzregeln

- Valide Datentypen gemäss Tabelle oben. Keine leeren Attribute erlaubt
- Referentielle Integrität: Kombination aus bpAb+bpAn ist auflösbar in relationen.csv
- Zeitliche Konsistenz: zeitVon< zeitBis

# leistungsverknuepfungen.csv

Abhängigkeiten zwischen Leistungen als zusätzliche Randbedingungen für die Solver.

Use Cases:

- Im Rahmen der Interaktiven Planung kann der User eine bestehende Lösung anpassen, indem er ausgewählte Paare von Leistungen 'aneinander bindet' oder 'auseinander bricht'. Solche Randbedingungen werden im hier beschriebenen File aufgelistet als Bestandteil des Planungsauftrags.
- Die Referenzlösung wird mittels Leistungsverknüpfungen an die Solver gegeben, um Abweichungen in der Zielfunktion bestrafen zu können.

Später braucht es dann zudem Möglichkeiten für Präzisierungen, welche Leistung gemeint ist, zB. wenn mehrere pro Kundenfahrt.

| Attribut | Beschreibung | Datentyp |
|---|---|---|
| leistungTyp1 | KUNDENFAHRT, WARTUNG, STARTPUNKT | string |
| leistungId1 | Id von Kundenfahrt, Wartungsfenster (WARTUNG), Fahrzeuggruppe (STARTPUNKT) | string |
| kettenLabel1 | Kettenlabel für verknüpfte Leistung 1<br>gesetzt bei spezifischem Verknüpfungstyp (REFERENCE, JOINT, SEPARATE)<br>und wenn leistungsTyp1 == KUNDENFAHRT | string / leer |
| leistungTyp2 | KUNDENFAHRT, WARTUNG, ENDPUNKT | string |

| | | |
|---|---|---|
| leistungId2 | Id von Kundenfahrt, Wartungsfenster (WARTUNG), Endpunkt (ENDPUNKT) | string |
| kettenLabel2 | Kettenlabel für verknüpfte Leistung 2<br>gesetzt bei spezifischem Verknüpfungstyp (REFERENCE, JOINT, SEPARATE)<br>und wenn leistungsTyp2 == KUNDENFAHRT | string / leer |
| type | REFERENCE, JOINT, SEPARATE<br><br>Für die Bedeutung der einzelnen Typen, siehe Geschäftsregel 'Spezifische Leistungsveknüfpung' und Zielfunktionskomponente 'Abweichung Referenzlösung'.<br><br>⚠️ Zu beachten: JOINT und REFERENCE Leistungsverknüpungen übersteuern zudem die Prozesszeiten, die für Leistungsübergänge benötigt sind. Siehe Geschäftsregeln zu den Prozesszeiten. | Enum as string |

Konsistenzregeln

- Valide Datentypen gemäss Tabelle oben. Keine leeren Attribute erlaubt.  Leere Attribute nur bei Kettenlabels, bei Referenzen auf Wartung, Start oder -Endpunkt (nicht bei Referenzen auf Kundenfahrten)
- Referentielle Integrität: (leistungTyp1. leistungId1) und (leistungTyp2. leistungId2) sind je nach Typ auflösbar in kundenfahrten. csv, wartungsfenster.csv, fahrzeuggruppen.csv, endpunkte.csv
- Keine Selbstverknüpfung: leistungId1 unterscheidet sich von leistungId2
- Zeitliche Konsistenz: kundenfahrt1.zeitAn / wartungsfenster1.endZeit / fahrzeuggruppe1.startZeit <= kundenfahrt2.zeitAb / wartungsfenster2. startZeit / endpunkt2.endZeit
- Flusserhaltungskonsistenz JOINT: Pro leistungsTyp+leistungsId+kettenLabel gibt es max. eine ausgehende und max. eine eingehende Leistungsverknüpfung vom Typ JOINT
- Flusserhaltungskonsistenz REFERENCE: Pro leistungsTyp+leistungsId+kettenLabel gibt es max. eine ausgehende und max. eine eingehende Leistungsverknüpfung vom Typ REFERENCE
- Uniqueness über den ganzen Datensatz (inkl. Kettenlabels): Mehrere identische Einträge sind nicht zulässig
- Widerspruchsfreiheit pro Leistungspaar: Pro leistungId1+leistungId2 (jeweils inkl. Kettenlabels) kann es nicht sowohl JOINT und auch noch SEPARATE Verbindungen haben.

# config.yaml

Dieses File gibt es einmal im Roma Helper hinterlegt für die Defaultwerte und einmal (optional) im Planungsauftrag. Letzteres beinhaltet übersteuernde Werte mit Vorrang.

| parameter name | description | data type | default value |
|---|---|---|---|
| **required process durations between Leistungen** | | | |
| duration_between_leistungen.minimal | required process durations between Leistungen | duration as string | "PT19S" |
| duration_between_leistungen.wende | required process duration between Leistungen in turnround case | duration as string | "PT2M" |
| duration_between_leistungen.betriebsfahrt | required process duration before and after Betriebsfahrt | duration as string | "PT2M" |
| duration_between_leistungen.kuppeln | required process duration between Leistungen when bedarf changes | duration as string | "PT4M" |
| duration_between_leistungen.event | minimal time between ordering-relevant events (arrival at Stärkung with same RC, departure at Schwächung with same RC) choosable by SMT solver | duration as string | "PT1M" |
| **penalty weights in objective function (1 unit = 1 Betriebsfahrtkilometer)** | | | |
| objective.cost_per_fahrzeuggruppe_planned | cost per Fahrzeuggruppe performing Leistungen (at least one) | integer, non-negative | 100 |
| objective.<br>cost_per_violated_reference_leistungsverknuepfung | cost per violated Leistungsverknüpfung of type Reference | integer, non-negative | 20 |
| objective.continuous_idle_time.minimum | bonus to favor long continuous idle times to prevent a tattered timetable<br><br>cost_factor * max(fuer.ab - von.an - minimum, 0)^(exponent) | iso duration, non-negative | PT1H |
| objective.continuous_idle_time.exponent | | float, >= 1.0 | 1.1 |
| objective.continuous_idle_time.cost_factor | | float, <= 0.0 | -5.0 |
| objective.bathtub.marginal_cost_per_deceeded_km | cost of performing maintenance on a Rollmaterialeinheit whose km counter is below lower bound bathtub.distance.lb (accounted for each km). | float, non-negative | 0.05 |
| objective.bathtub.marginal_cost_per_exceeded_km | cost of performing maintenance on a Rollmaterialeinheit whose km counter is above upper bound bathtub.distance.ub (accounted for each km). | float, non-negative | 0.05 |

| | | | |
|---|---|---|---|
| objective.bathtub.marginal_cost_per_deceeded_second | cost of performing maintenance on a Rollmaterialeinheit whose time counter is below lower bound bathtub.duration.lb (accounted for each second). | float, non-negative | 0.0002 |
| objective.bathtub.marginal_cost_per_exceeded_second | cost of performing maintenance on a Rollmaterialeinheit whose time counter is above upper bound bathtub.duration.ub (accounted for each second). | float, non-negative | 0.0002 |
| **ivog for Wartungsintervalle (**will be moved to fahrzeugtypen, as soon as we can model different types) | | | |
| ivog.duration | maximum duration between consecutive "Wartungen" | duration, positive | "P30D" |
| ivog.distance | maximum number of kilometers between two "Wartungen" | float, positive | 12500.0 |
| ivog.bathtub.distance.lb | lower bound of optimal km counter range for performing maintenance | float, positive | 0.0 |
| ivog.bathtub.distance.ub | upper bound of optimal km counter range for performing maintenance | float, positive | 12500.0 |
| ivog.bathtub.duration.lb | lower bound of optimal time counter range for performing maintenance | float, positive | "P0D" |
| ivog.bathtub.duration.ub | upper bound of optimal time counter range for performing maintenance | float, positive | "P30D" |
| **postprocessing Teilleistungsketten-Graph (TLK)** | | | |
| postprocessing.cut_gap | Minimum length of gap between two Leistungen, such that a cut is inserted between them in the TLK. | duration, positive | "PT4H" |