CDM-Telematics Wiki Documentation

Contents

1 CDM-Telematics compiled wiki pages

Self-contained version with local images

1.1 Version

This document was generated on 2025-11-07 16:16:58 UTC

Operational entities

Entity details

Purpose

Operational entities

Operational entities may all share time-independent properties (attributes), which are described here.

Diagram

#
Operational entities

For the time being, a user-defined "name" is proposed. It is not unique (one entity may have several names; different entities can have the same name as the "name" is a ver-

---

Operational entities

Comments The universally unique identifier is expected to be the entity IRI. IRIs are the foundations of the Semantic Web.

#
Operational entities
Note: one reason for introducing the "name" property is that annotation properties (such as rdfs:label) that would play this role are left out of OWL to JSON transformers (CIM

#
Operational entities
Original page: 01a---Entity-details.md

2 Train run

2.1 Purpose

"Train run" (commonly "train", in an operational context) designated a train (characterized by a train number) running on a particular day. It is what a dispatcher has to manage, not to be confused with: * "train" as a railway timetable entry; * "train" as a set of railway vehicles, some of which are powered.

The expression "train run" was chosen because it best evokes the train as a process.

The equivalent class in the ERA TAF ontology is Train.

2.2 Diagram

2.2.1 Description

The GRAPHOL diagram is centered on the "Train run" class that has a set of static properties (on the right) and time-dependent states (on the left).

The static properties are directly derived from the Telematics TSIs and ontology, e.g. OTN (Operational Train Number).

The train states are further described in another diagram.

2.2.2 Comments

2.2.2.1 GRAPHOL representation of properties In GRAPHOL, object properties are represented by diamonds. Their name often starts with "has", but this is a convention, not a rule.

Object properties are displayed as class A <-[white square] - - [black square] -> class B, meaning P(A, B), i.e. individuals of A are related to individuals of B by a property named P. Example: some

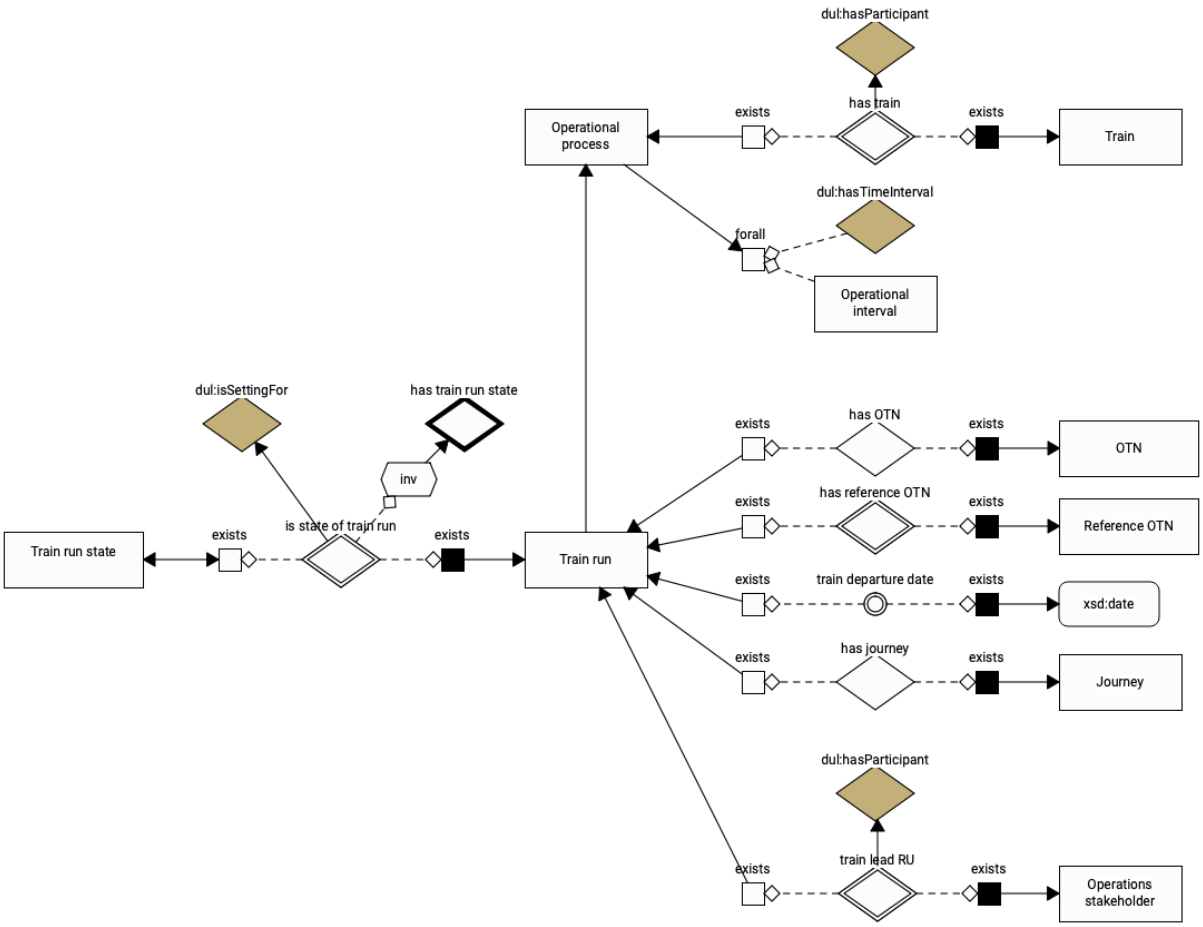

Figure 1: train run diagram

Train (subject) has a train operating RU (predicate) some Train operator (object).

Diamonds with a double-edged rim denote "functional properties". In OWL2, a functional property is such that any subject will be associated with at most one object. Inverse functional properties have a thick black rim, and properties that are both functional and inverse functional have a double-edged rim on one side and a thick black rim on the other side.

When the property objects are data types (such as strings, integers, timestamps…), the property is called a "data property" in OWL2. It is represented by a small circle, as is the case here with "train departure date". The double rim also means functional [data] property.

Note: in OWL2, object properties can be navigated in both directions, no matter if the inverse property is defined (as is the case here for "is state of train run") or not. Data properties are however one way (they have no inverses). Of course the end user can *query* OWL2 data to find, e.g., "all train runs occurring today", but the logics underpinning OWL2 ontologies (and executed by reasoners such as Pellet or HermiT) cannot achieve that.

2.2.2.2 OWL2 Subproperties In OWL2, properties are first-class objects; there are subproperties as there are subclasses. The semantics of "sub" is inclusion, and the symbol for inclusion is the simple arrow in both cases. Explanation:

- for classes: class A is a subclass of class B iff (= if and only if) any individual of A is also an individual of B.
- for properties: property P is a subproperty of property Q iff for any individuals X and Y satisfying P(X,Y), Q(X,Y) also holds.

The interest of defining subproperties is mainly a semantic one. Example: "empty mass" or "laden mass" are subproperties of "has mass", meaning that the expected value is a mass expressed in, say, kg, but their meaning and relevance is quite precise.

In the present case, the subproperty arrow links the general-purpose dul:isSettingFor property with its specialization, "is state of train run".

2.2.2.3 Train run states relate to exactly one train run This is partly expressed by property "is state of train run" being a functional property. "Functional" implies 0 or 1 object.

A train run *state* however is expected to describe exactly one train run, as common sense dictates, not "at most one".

Enforcing such cardinality "exactly one object" is possible and is done elsewhere, and in various ways (using OWL2 or SHACL constraints; using an explicit cardinality or an existential condition). Here, we chose to represent "at most one" by using a functional property, and "at least one" by stating an existential condition ("Train run state" is a subclass of all things X satisfying "X is the state of some train run"). The condition can be read from the diagram.

2.2.2.4 A train run involves one train … although its composition and loads may change between origin and destination. See the Train-related wiki page.

2.2.2.5 Where are the train origin and destination? The train destination may change in the course of the train run (think of a re-routed train). This is why the train journey (planned, foreseen, actual…) are time-dependent, and documented using

2.2.2.6 About OWL2 profiles Explicit cardinalities (min=1, max=1) would look nicer, but are avoided here for technical reasons. We try to stick to a subset of OWL2, namely the OWL2 RL profile, to secure the tractability of queries in polynomial time.

Original page: 02---Train-run.md

Train servicing

Operational Location

Purpose

Train servicing

Defining primary and subsidiary location codes. Locating fixed assets (tracks, facilities) at the places designated by these codes.

Diagram

Comments

Train servicing

Property "at operational location"

#	Train servicing
	This property only applies to (= has range) spatially fixed assets such as station tracks, facilities (yards, depots, stations, terminals). There is no time information attached to it.

Train servicing

Please note that a yard *has* a location; a yard *is not* a location. This confusion often occurs because a yard happens to be a fixed thing. The confusion never occurs with

---

Train servicing

So where are moving things located at time t ?

Train servicing

The answer lies in the dul:Situation class and its subclasses (generally called "... status" in the present ontology). dul:Situation is the class where time-dependent information is asserted.

Train servicing
Original page: 03---Operational-Location.md

3 Train

3.1 Purpose

The “Train” class designates the physical object (here: dul:DesignedArtefact) that is moved in the process of running a train, the “Train run”. That includes not only the rolling stock (considered in running order, with all supplies included), but also the loaded freight and (with a minus sign) the spent fuel or sand.

3.1.1 Diagram

The composition of the Train varies between Journey Segments (where wagons can be added or removed, locomotives can be changed, etc.). We choose to consider that for each particular train run, we have a single train (i.e. a single individual of class Train) with variable composition and loads. This choice best coincides with the way people speak.

Figure 2: Train

Original page: 04---Train.md

Wagon

Intermodal Transport Unit

Purpose

Diagram

Comments

Original page: 06---Intermodal-Transport-Unit.md

---

4 Cargo

4.1 Purpose

4.2 Diagram

4.3 Comments

Original page: 06a---Cargo.md

Track

Facility

Track

Purpose

Diagram

Comments

Original page: 08---Facility.md

---

5 Traction role

5.1 Purpose

5.2 Diagram

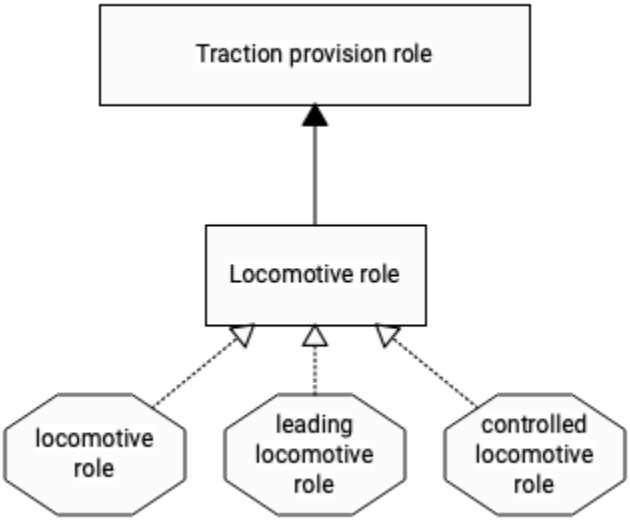

Figure 3: Traction role

5.3 Comments

Original page: 09---Traction-role.md

Load Role

Operational roles

Purpose

#	Load Role
	Companies may play different operational roles at different times. Examples of operational roles are: Train operator, Cargo carrier, Lead RU, Lead carrier (the latter is encountered in the

---

Load Role

Diagram

Operational roles are currently listed as members of class Operational-Role. In the future, these may be replaced by a SKOS concept scheme provided by ERA.

Load Role

The answer to "who plays the role" is outside this ontology. Such companies are by definition subclasses of dul:Organization and regorg:RegisteredOrganization.

#	Load	Role
dul:Organization	tells	that its members are able to play a role.
regorg:Organization	provides	lots of useful attributes (legal name, jurisdiction, registration authority, etc.). The W3C RegOrg ontology

---

Load Role

Comments

No dedicated class for organizations - just a blank node

#	Load	Role
We simply use a “blank node” (a class without a name) that is the intersection (“and”) of dul:Organization and rerorg:RegisteredOrganization.		

Load Role

What the diagram expresses (and what OWL2 says) is that anything that plays an operational role in the context of telematics is, by definition, both a dul:Organization (hence not a person) and a rerorg:Organization

---

Load Role

Original page: 11---Operational-roles.md

---

6 Versioned description

6.1 Purpose

6.2 Diagram

6.3 Comments

Original page: 12---Versioned-description.md

Journey

Journey Schedule

Purpose

Journey

Represent the train journey (planned or executed) and possibly the train path in one uniform way.

Journey

The Journey may be composed of a sequence of journey sections, each having an origin and a destination (operational locations). Obviously, the destination of section N is ex-

---

Journey

Diagram

Comments

Nested Lists

Journey

Since any Journey section is a Journey schedule, it can be in turn broken down. This is convenient, as it allows to insert pass-through locations at a later stage when deemed convenient.

Journey

Not a speed profile

Journey

Locations documented by Journey Schedule are, for the time being, only Operational Locations. The Journey Schedule is not suitable for representing a speed profile with temporary speed

---

Journey

Data consistency

Journey

It is a user responsibility to check the order and consistency of journey sections. The ontology will preserve the sequence (using a List ontology, because OWL2 has no prim-

---

Journey

List ontology

Journey

The List ontology used here is different from the one used in IfcOwl (the ontology version of Industry Foundation Classes). Later on, one of the two ontologies may be elimi-

#
Jour-
ney

Original
page:
12b-
--
Journey-
Schedule.md

7 Journey Schedule properties

7.1 Purpose

Properties are made available by the List ontology that allow to "chain" and navigate the Journey Sections. Some are illustrated here.

7.2 Diagram

Journey sections are chained with no branches and no loops. This is expressed by having properties (next section, previous section) that are both functional and inverse functional: the diamonds have both a double rim (functional, multiplicity 0..1) and a thick black rim (inverse functional, multiplicity 0..1).

On the left, you see the GRAPHOL representation of the OWL2 assertion: "in journey schedule" is the exact equivalent (double arrow!) of the inverse of "has journey section".

Note: since "in journey schedule" is the inverse of an inverse functional property, it consequently is a functional property. The diagram does not declare it (simple rim instead of double rim) but OWL2 reasoners will infer it.

7.3 Comments

(about the identification of the first and last item)

Original page: 12c---Journey-Schedule-properties.md

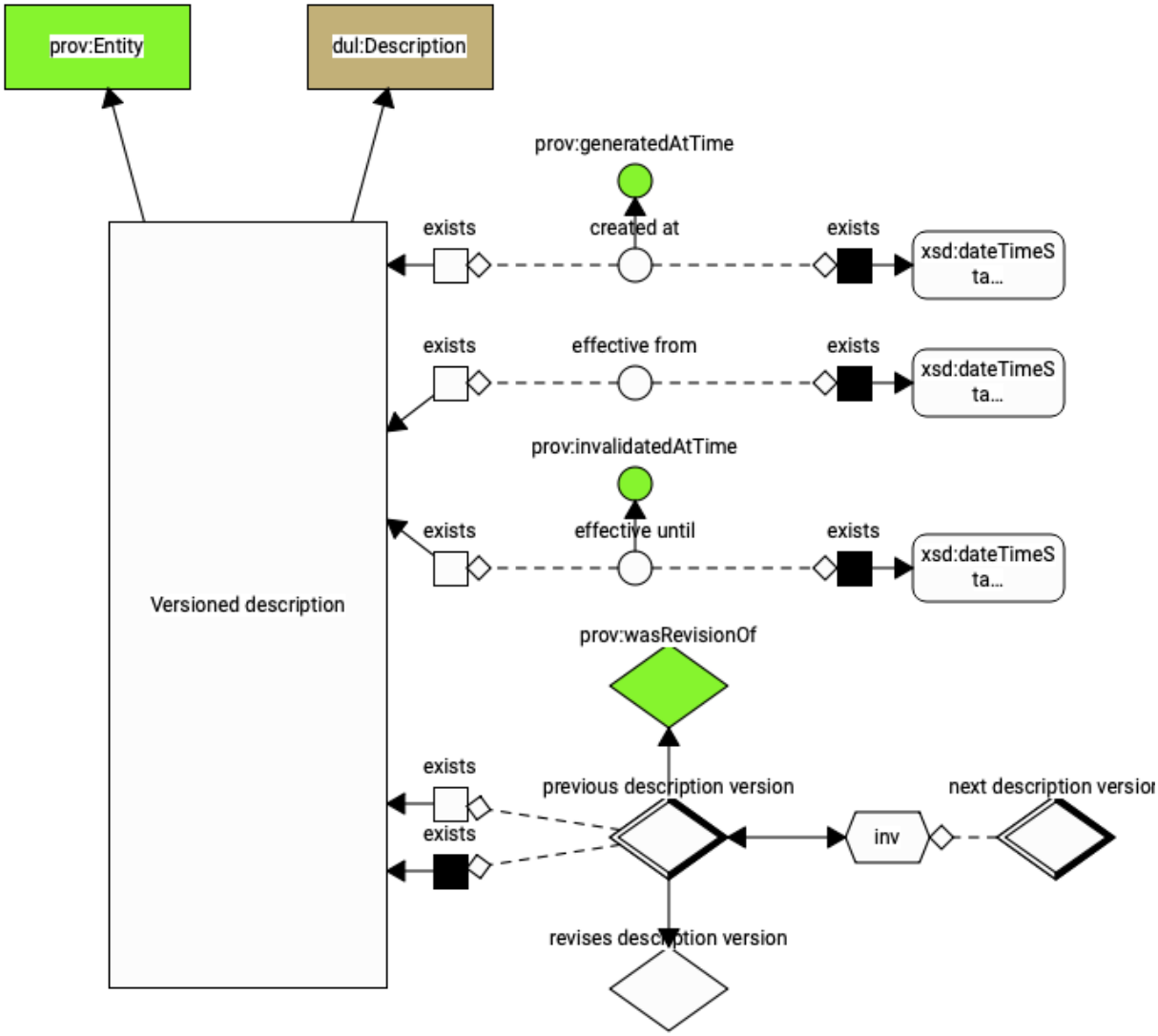

Figure 4: Versioned description

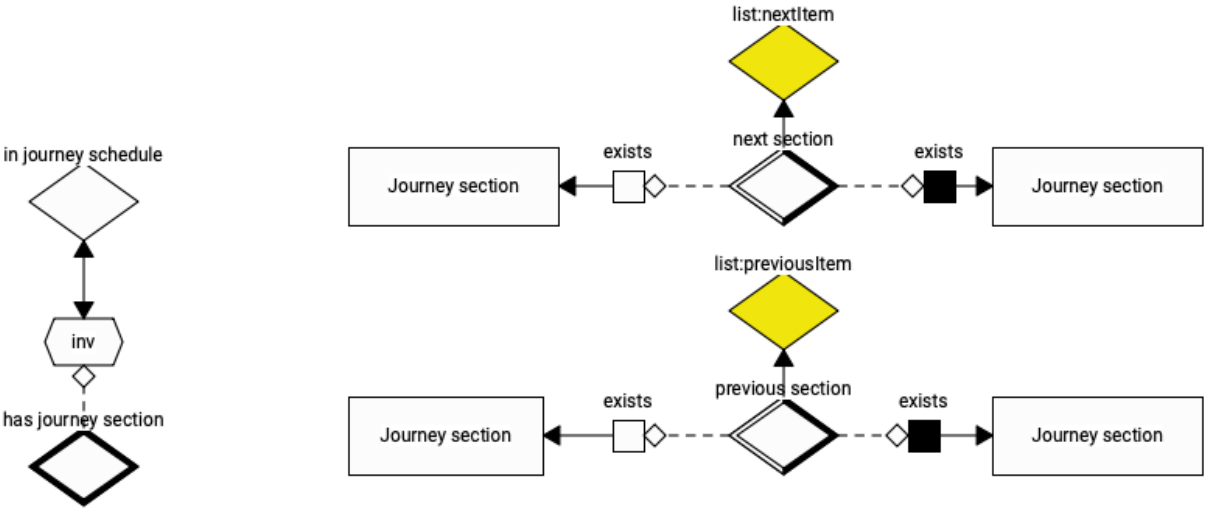

Figure 5: Journey schedule properties

Operational State

Train run state

Purpose

Diagram

Comments

Operational State

Original page: 13a---Train-run-state.md

8 Train state

8.1 Purpose

8.2 Diagram

8.3 Comments

Original page: 13b---Train-state.md

Load State

Message

Purpose

Diagram

Comments

#
Load
State

Original
page:
14---
Message.md

9 Image

9.1 Purpose

Provide a reference to images, and access image contents (e.g. a PNG file).

9.2 Diagram

The image reference (URI) is separated from the contents (Content location, identified by its own URI). It is likely that they will not share the same workspace. Persistence and backup mechanisms are likely to be very different: e.g. images will require frequent archiving, owing to their "weight".

The content URI may be a URL (dereferenceable) or another type of URI if only a "unique key" in a specific database is needed.

9.3 Comments

9.3.1 Metadata

Image metadata persistence (provenance…) is of course crucial and shall be discussed in due time.

Original page: 15---Image.md

#
RID
codes

#
Time
##
Pur-
pose

#	RID codes
	Propose "time entities" to be consumed by time-dependent entities (such as "train speed" or "custom clearance status").

RID codes
The time entities are derived from the W3C time ontology and aligned with the DUL TimeInterval concept.
Diagram

RID codes

An operational time is either an instant or an interval: these are disjoint classes, as the black flattened hexagon indicates in the diagram. The distinction between instant and interval rests

#	RID codes
The GRAPHOL diagram expresses that each interval is expected to have exactly one beginning and one end, by means of OWL universal restrictions (“forAll”), and that the beginning or	

#	RID codes
Note: if the data provide two beginnings for an interval, the logical consequence is that the beginnings B1 and B2 are the same individual (:B1 owl:sameAs :B2) and any OWL reasoner will	

#	RID codes	Operational times (instants or intervals) may have a "date and time of issue". This non-mandatory information is of interest in the case of repeated forecasting or revision exercises. The bulk of exchanged times,

RID codes


#	RID codes
	The bottom part (Temporal role class and individuals) add meaning to time-dependent situations, i.e. whether they relate to a planned, revised, forecast, actual, etc. situation (the DUL term) or state (our

#
RID
codes

##
Com-
ments

###
Real
Time
appli-
ca-
tions

#	RID codes
The time representation is conceptually compatible with the UML MARTE profile, so it can serve as a basis for a demanding real-time application. The main missing concept	

#	RID codes
We do not distinguish intervals from “proper intervals” (length > 0) in the sense of the W3C Time ontology: as no assumptions are made regarding the timestamps, if may well happen	

# RID codes	### Against open-ended inter-vals

RID codes

Leaving out the beginning or end of an interval *may* be interpreted as the interval being "open-ended". We would like to sternly warn against such convention, since it breaks the monotonicity of time

#
RID codes

In the context of railway applications, nothing is actually open-ended in time, and most data may be made available with some delay, so the "open ended" interpretation of any inter-

---

#
RID
codes

Original
page:
90---
Time.md

---

10 Varia

10.1 Purpose

10.2 Diagram

10.3 Comments

Original page: 95---Varia.md

#
De-
pen-
den-
cies
##
Gen-
eral
and
up-
per
on-
tolo-
gies
###
W3C
Time
on-
tol-
ogy
###
SOSA/SSN

###
QUDT
###
DOLCE+DnS
Ul-
tra-
lite
###
RE-
GORG,
ORG
##
Se-
man-
tic
RSM
on-
tolo-
gies
###
Rolling
stock
on-
tolo-
gies
####
Con-
sist
####
Ty-
pol-
ogy
##
ERA
Con-
cept
Schemes

Original page: 97---Dependencies.md

11 References

...

Original page: 99---References.md

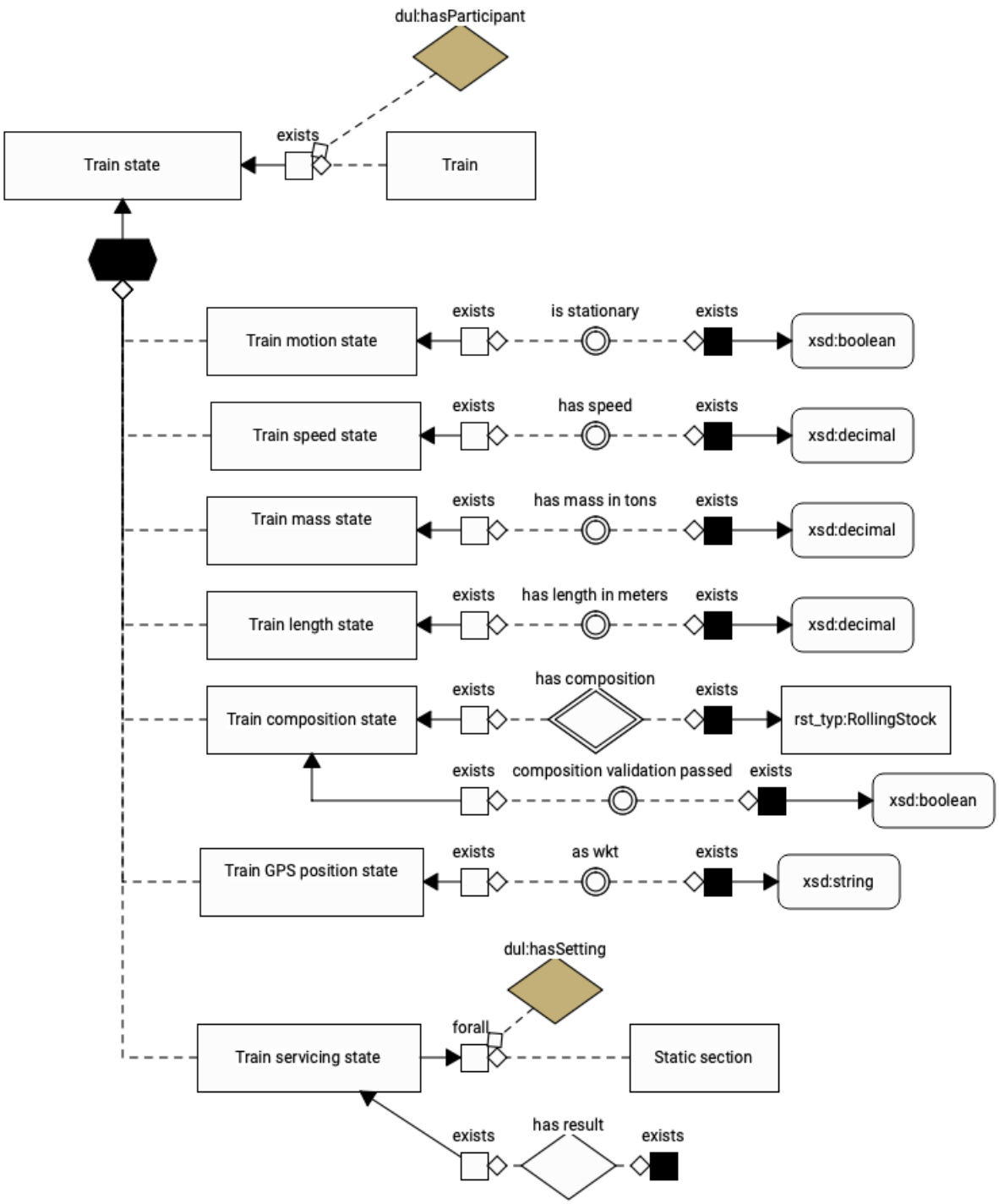

Figure 6: Train state

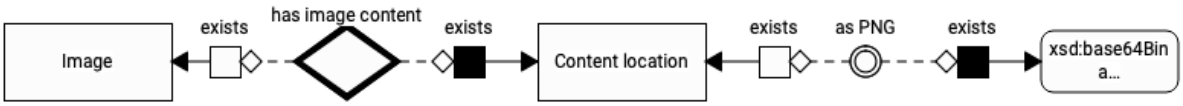

Figure 7: image class

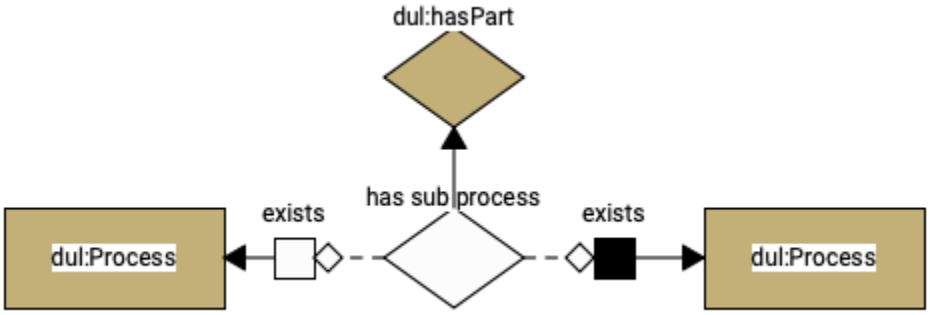

Figure 8: Varia