S02- Onboard Architecture specification Part10 - APC-II

Version History

Date	Version	Description	Author
early June 2021	0.1.0	Initial TWG draft	OAB
2021-06-24	0.2.0	Updates after first walk through	OAB
2021-07-06	0.2.1	Minor update after comments	OAB

Information Technology for Public Transport (ITxPT)
Rue Sainte-Marie 6, B-1080 Brussels | Belgium
Tel +32 (0)2 673 61 00 | Fax +32 (0)2 660 10 72
info@itxpt.org | www.itxpt.org

International non-profit making association (AISBL) incorporated by Royal Decree and the Belgian Ministry of Justice under Belgian law WL22/16.729 on 4th May 2016.

Important notice

The present document can be downloaded from https://wiki.itxpt.org/index.php?title=ITxPT_Technical_Specifications.

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ITxPT. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ITxPT deliverable is the one made publicly available in PDF format at https://wiki.itxpt.org.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ITxPT documents is available at https://wiki.itxpt.org/index.php?title=Releases.

If you find errors in the present document, please send your comment to the technical support https://itxpt.org/membership/contact-us.

Copyright Notification

All rights reserved. This document has been produced and approved by ITxPT and represents the views of those members who participated.

The content of the PDF version shall not be modified. The copyright and the foregoing restriction extend to reproduction in all media.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ITxPT documents is available at wiki.txpt.org

© ITxPT 2021
All rights reserved.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described below.

"**must**" and "**must not**" are **NOT** allowed in ITxPT deliverables except when used in direct citation.

In order to be able to claim compliance with an ITXPT deliverable, the user needs to be able to identify the requirements that are obligatory. The user also needs to be able to distinguish these requirements from other provisions where there is a certain freedom of choice.

This clause is clearly stating the verbal form that shall be used to express a particular kind of provision, i.e. a requirement, a recommendation or a permission.

In the first column of tables T1 to T4 the verbal form that shall be used to express each kind of provision is given. The equivalent expressions given in the second column may be used only in exceptional cases when the form given in the first column cannot be used for linguistic reasons

NOTE: Only singular forms are shown.

The verbal forms shown in Table T1 shall be used to indicate **requirements** strictly to be followed in order to conform to the standard and from which no deviation is permitted. For example, the requirements to be followed may relate to values, actions, features to be supported and/or used or presence/absence or optional elements.

Verbal form	Equivalent expressions used in exceptional cases[^1]
shall	is to is required to it is required that has to only ... is permitted it is necessary
shall not	is not allowed / permitted / acceptable /permissible is required to be not is required that ... be not is not to be

Table T1 - Requirement

The verbal forms shown in Table T2 shall be used to indicate that among several possibilities one is **recommended** as particularly suitable, without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required, or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited. For example, the recommendations may relate to values, actions, features to be supported and/or used or presence/absence or optional elements.

Verbal form	Equivalent expressions used in exceptional cases[1]
should	it is recommended that ought to
should not	should not it is not recommended that ought not to

Table T2 - Recommendation

The verbal forms shown in Table T3 shall be used to indicate what is **permitted** by the ITXPT deliverable, which can be values, actions, support and /or use of features or presence/absence of optional elements.

Verbal form	Equivalent expressions used in exceptional cases[1]
may	is permitted is allowed is permissible it is not required that

	it is not required that
may not	
	no ... is required

Table T3 - Permission

The verbal forms shown in Table T4 shall be used for statements of **possibility and capability**, whether material, physical or causal.

Verbal form	Equivalent expressions used in exceptional cases[1]
	be able to
can	there is a possibility of
	it is possible to
	be unable to
cannot	there is no possibility of
	it is not possible to

Table T4 - Possibility and capability

The verbal forms shown in Table T5 shall be used to indicate **behaviour** of equipment or sub-systems **outside** the scope of the ITXPT deliverable in which they appear. For example, in an ITXPT deliverable specifying the requirements of terminal equipment, these forms shall be used to describe the expected behaviour of the network to which the terminal is connected.

Verbal form	Equivalent expressions
will	-
will not	-

Table T5 - Inevitability

1. Concepts and Terminology

1.1. ITxPT Data Dictionary Concepts

Concepts from the ITxPT Data Dictionary is referred by UPPER CASE, when the concept definition is important.

The following Concepts from the ITxPT Data Dictionary are used.

COMPOUND TRAIN - A VEHICLE TYPE composed of a sequence of more than one vehicles of the type TRAIN.

ENTRANCE FOR PASSENGER SPACE - A PASSENGER ENTRANCE used as an entry- and/or exit- point to or from a certain PASSENGER SPACE along with information if the designated exit and entry directions of the PASSENGER ENTRANCE are aligned or reversed in relation to the PASSENGER SPACE.

PASSENGER ENTRANCE - A physical or virtual boundary point through which passengers can enter or exit, e.g. a vehicle door. A PASSENGER ENTRANCE has a designated enter-direction and a designated exit-direction.

PASSENGER ENTRANCE COUNT - Number of passengers and other objects that have entered and

exited through a specific PASSENGER ENTRANCE during a time span or since some implicit or explicit previous time/event. A possible implementation of LOGGABLE OBJECT.

PASSENGER SPACE - A passenger area within a VEHICLE. It may be limited to only a part of a VEHICLE such as a TRAIN ELEMENT, upper deck/lower deck, first class compartment or a bounded open space. A PASSENGER SPACE can be part of, overlap with, and be made up of other PASSENGER SPACEs.

PASSENGER SPACE CAPACITY - The number of passengers and other objects that are present in a PASSENGER SPACE when it is at 100% capacity.

PASSENGER SPACE ENTRANCE COUNT - Number of passengers and other objects that entered and exited a specific PASSENGER SPACE during a time span or since some implicit or explicit previous time/event. A possible implementation of LOGGABLE OBJECT. It is in essence an aggregation of relevant PASSENGER ENTRANCE COUNTs according to ENTRANCE FOR PASSENGER SPACE information.

PASSENGER SPACE OCCUPANCY COUNT - Number of passengers and other objects that are in a PASSENGER SPACE at a given time. A possible implementation of LOGGABLE OBJECT.

TRAIN - A VEHICLE TYPE composed of TRAIN ELEMENTs in a certain order, i.e. of wagons assembled together and propelled by a locomotive or one of the wagons.

TRAIN ELEMENT - An elementary component of a TRAIN (e.g. wagon, locomotive).

VEHICLE - A public transport vehicle used for carrying passengers.

1.2. Other Terminology

In addition the following terminology is used in the specification.

APC: Automatic Passenger Counting

APC Aggregator: A component/module/process/etc that generates APC data based on inputs.

2. APC-II Solution description

2.1. Relation to ITxPT S02P07 APC Specification

The specification is broadly similar to S02P07, and in some aspects *very* similar. However it also changes a number of things: - Modified data structures to support Heavy Rail - Changed data format to JSON - Changed Transport to MQTT

As such the old specification have been kept as a Legacy specification to support existing equipment and consumers, and this specification has been introduced as a new specification, APC-II

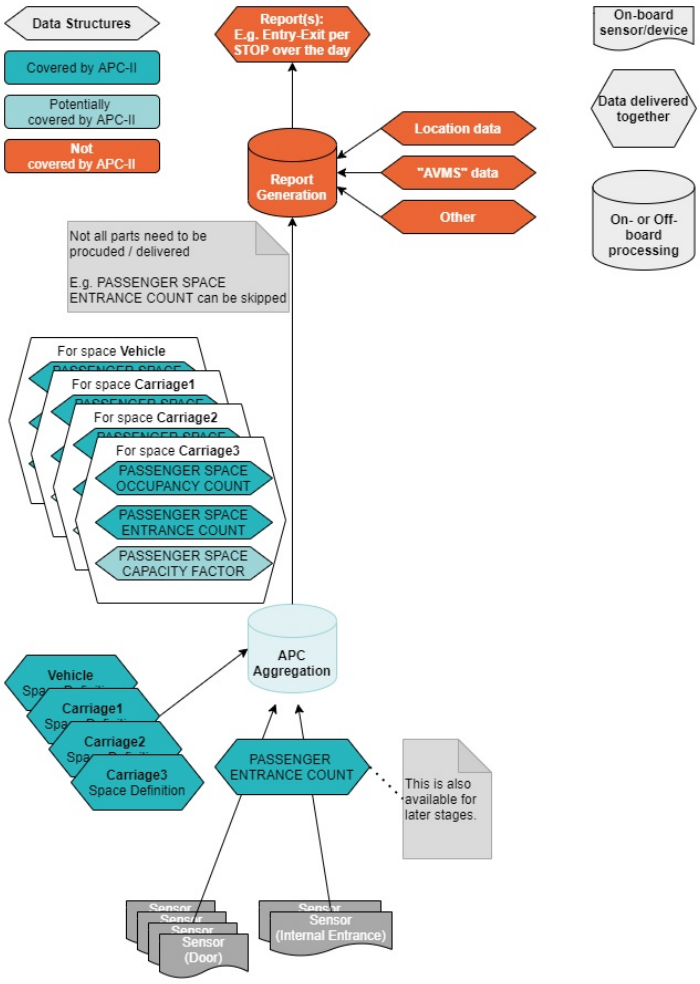
2.2. Purpose of APC

Put some purpose here

2.3. High Level solution

The APC specification specifies a few different things: - The information from Entrance (Door) Sensors - The information aggregated from Sensors (of any type) for PASSENGER SPACES - How to express how Entrances relates to Spaces, and Spaces to other Spaces.

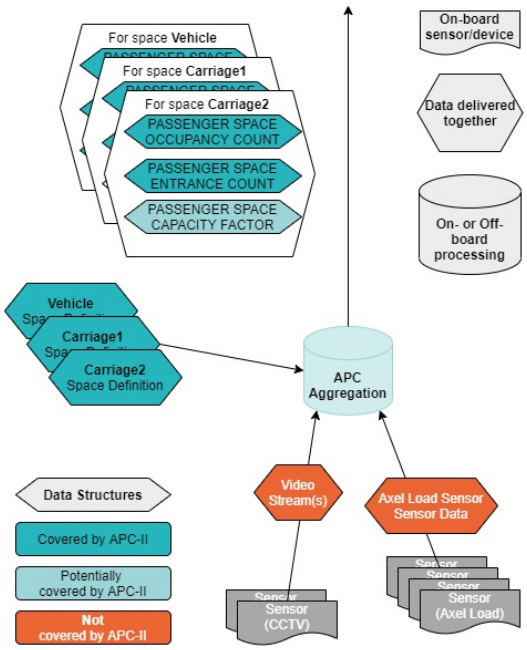
The specification specifies the Data being exchanged, not where or how the Data is being processed or generated. In particular it does not contain any requirements on what is done onboard and what is done in the back office.



Overview door based

Above is a “traditional” setup using door sensors on the vehicle doors, and similar sensors between carriages. For the defined PASSENGER SPACES, in this case the entire Vehicle and the three Carriages, the Occupancy and Entries/Exits are calculated.

This specification does not constrain what sources/data are used for calculating the Data. A solution that uses Entrance Sensors for the Vehicle Doors, and Video analysis for tracking internal movement between Spaces is one possibility. Another would be Video analysis supported by Axle load sensors, as shown below:



Overview axel/video based

Many other configurations are possible.

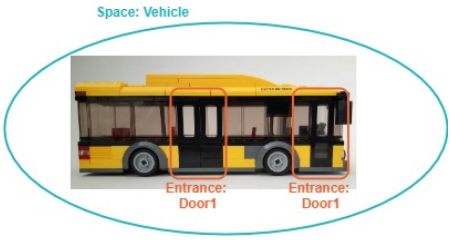
2.4. What is not standardized

Currently the specification only standardizes the data from Entrance (Door) sensors, not e.g. Axel Load sensors. And it does not standardize any reports, e.g. Entry/exit per STOP PLACE.

2.5. PASSENGER SPACES

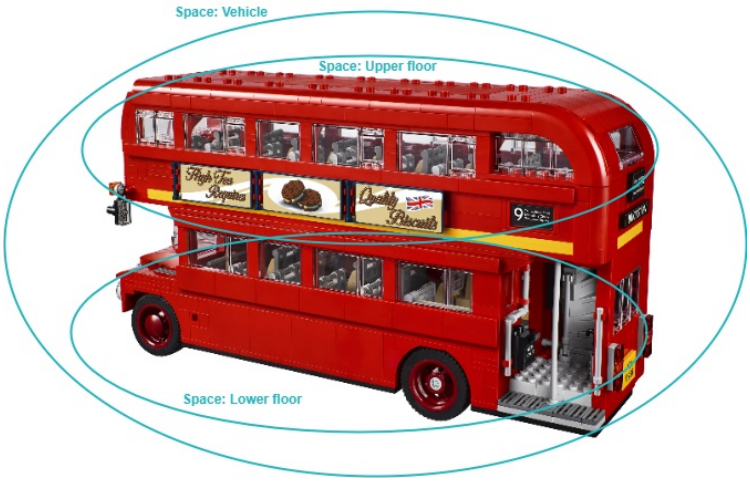
PASSENGER SPACES will be described more formally below. But as it the major addition compared to the S02P07 APC Specification, and the key concept to support Heavy Rail Requirements, a bit of less formal explanation is added here to aid the readability of the specification.

The simplest configuration is a Vehicle with one Space for the entire vehicle, and one or more Entrances:



Spaces - Simple Vehicle

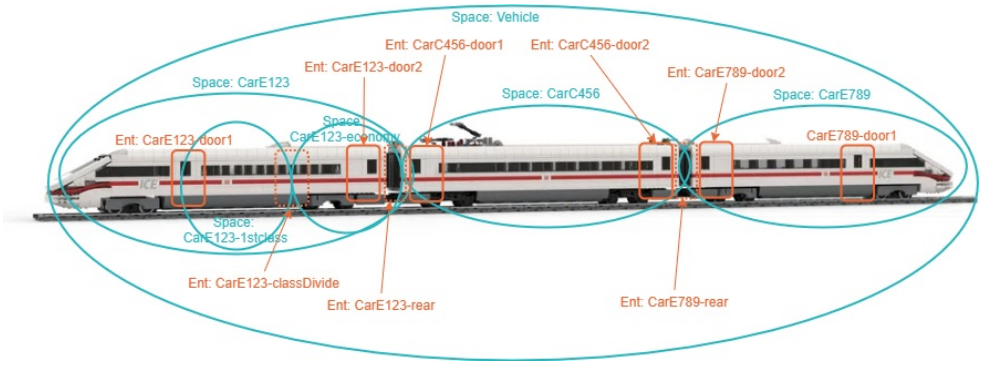
For a more complicated configuration there is still the Space for the entire Vehicle, but there are also Spaces for parts of the Vehicle:



Spaces - Bilevel Vehicle

In this Double Deck Bus, APC data can be reported for the entire Vehicle (Space: Vehicle), but also for the upper floor (Space: Upper floor) and the lower floor (Space: Lower floor).

There is no limit in the specification how Spaces are nested, or how many nested spaces there is.

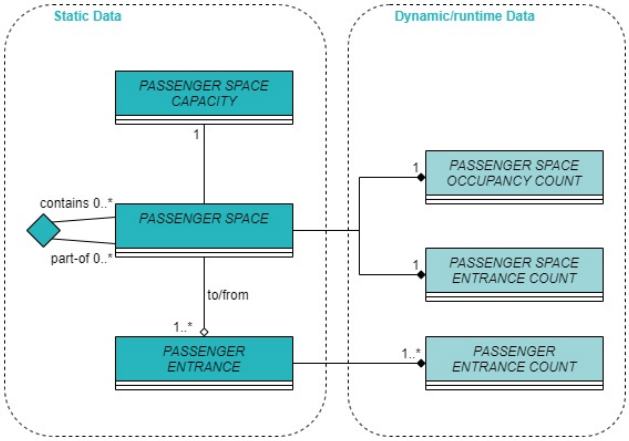


Spaces - Complex Vehicle

In this three carriage EMU there is a Space for the entire Vehicle, one Space for each of the three Carriages, and then one of the Carriages is further divided in a 1st-Class Space and an Economy-Class Space. Here are also the Entrances marked, both the external entrances (orange solid, into/out of the Vehicle) and the internal entrances (orange dotted, between Spaces).

And it does not stop there. The way this specification supports seat occupancy reporting is by defining a *Space for each seat*, and reporting the Occupancy on the “seat Space”.

3. Conceptual Data Model



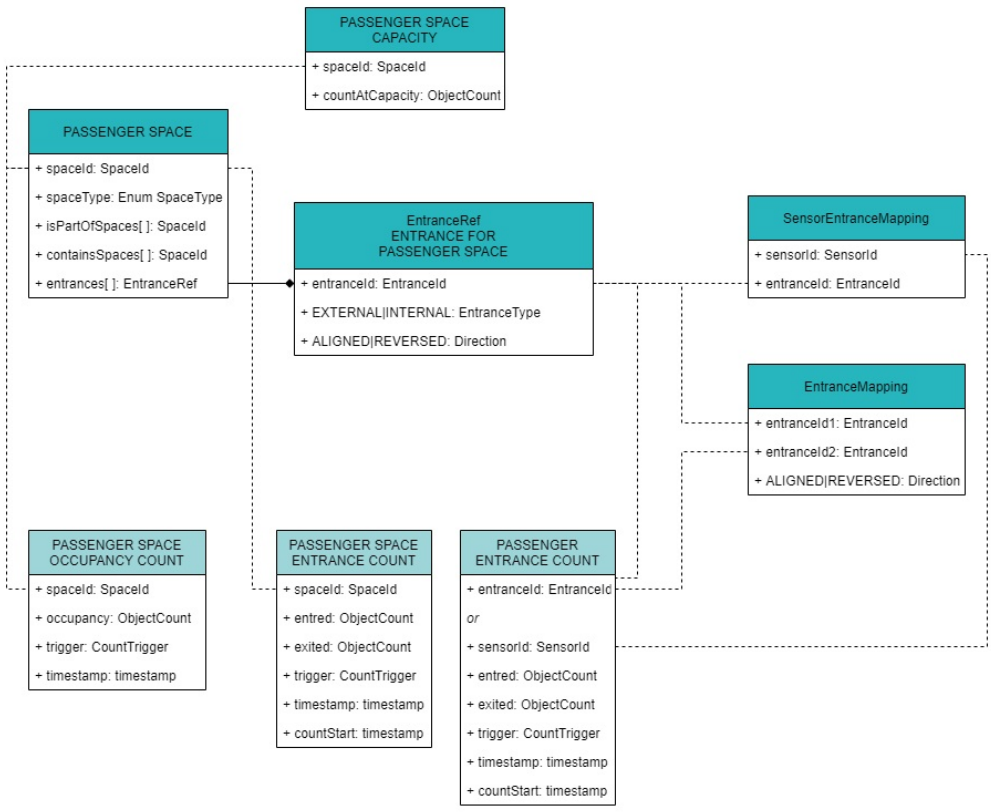
Conceptual Data Model

PASSENGER SPACE, PASSENGER SPACE CAPACITY and PASSENGER ENTRANCE are all statically configured (they will not change during operation) and are shown in darker colour. PASSENGER ENTRANCE COUNT, PASSENGER SPACE ENTRANCE COUNT, and PASSENGER SPACE OCCUPANCY COUNT all contain passenger count data and will change regularly during operation and are shown in lighter colour.

4. Detailed Data Model

The Conceptual Data Model above shows how the defined Concepts relates to each other. But it does not provide enough details to be the basis of a standard format.

To do that additional details are provided.



Detailed Data Model

The darker colour is used for static data, while the lighter colour is used for Counting data.

A number of things have changed. Relationships between different objects are now set by spaceId and entranceId, as *indicated* by the dotted lines.

PASSENGER ENTRANCE is not present as a separate class, as it did not carry any information that needed stand-alone representation.

4.1. Uniqueness of IDs

There is no namespaces for Space, Entrance and Sensor IDs. All IDs must be unique within the Vehicle it is used.

It is *strongly* recommended that all sensor IDs be globally unique, so that one sensor can never be mistaken for another.

The uniqueness of entrance and sensor IDs is not a problem for Vehicles that are fixed, e.g. a bus. In these Vehicles entrance IDs can be be “door1” and “door2”, space IDs “forward”, “middle”, and “rear” or something else simple.

But in a Vehicle made up of potentially interchangeable elements, typically a TRAIN made up of TRAIN ELEMENTS, it needs to be ensured that entrance and space IDs are unique regardless of which TRAIN ELEMENTS make up the the TRAIN.

It must either be ensured that the IDs are unique for all the possible combinations of TRAIN ELEMENTS, *or* each TRAIN must set (new) unique entrance and space IDs each time the makeup of the TRAIN changes.

4.2. Entrances

One Entrance may belong to multiple Spaces. E.g. an external entrance, Door7, may belong to the Vehicle Space, to the Carriage1 Space, and to the Carriage1-SilentCompartment Space. When one passenger enters through Door7 this is only one new passenger in the Vehicle, one new passenger in Carriage1 and one new passenger in Carriage1-SilentCompartment.

4.2.1. Sensor Entrance Mapping

The SensorEntranceMapping entity allows the sensors to not know their entranceId, as long as another actor supplies that. It also make it possible to have two, or more, sensors for one entrance, e.g. a left and right sensor for a wide entrance.

4.2.2. External entrances

An entrance of type EXTERNAL by definition leads into/out-of the Vehicle. Because of this the Vehicle Space does not *need to* contain any external entrances. Instead these can be found by looking that the Spaces that the Vehicle Space contains. E.g. for a three car train, the Vehicle Space may have an empty entrances list, while the external doors found in the car1, car2 and car3 Spaces’ entrance lists.

4.3. Relations of Spaces

Each Space that is not a Vehicle Space will be next to one or more other spaces. Currently Spaces have no direct relations to each other. Which spaces connects to which other spaces must be determined by looking at the entrances each space have. Two spaces that contain the same entrance connect to each other.

If the entrances are ALIGNED, then on entry a PASSENGER will be present in both Spaces. If the entrances

are REVERSED, then a PASSENGER will enter one Space while exiting the others.

> ***SPEC QUESTION:*** While not rocket science, the potential structure of Spaces could grow fairly complex. In ITxPT Labelling should we have different levels of supported complexity?

4.3.1. Entrance-Entrance Mappings

In cases where two spaces are next to each other, but *don't* have the same entrance ID on what is the passage between them, this is resolved by and Entrance-Entrance Mapping. This will then be used to determine that e.g. the entrance "car123-rear" for space "car123" is now connected to the entrance "car345-front" for space "car345". Now a car123-rear exit will be a car345-front entry, and vice versa.

A mapped entrance could have one Entrance Sensor, so that a PASSENGER ENTRANCE COUNT will be generated with one of the entrance IDs, but not the other. Or there could be a sensor for each of them, so that for each PASSENGER entry/exit there will be a PASSENGER ENTRANCE COUNT generated with each of the entrance IDs. The APC Aggregator needs to account for this, and handle both possibilities.

> ***SPEC QUESTION:*** This concept seemed great on paper. But thinking about the implementation it may be surprisingly complex to handle that there could be one or two sensors without knowing which is the case. Should we add to EntranceRef if the Entrance have a Entrance Sensor or not?

4.3.2. Multiple sensors for one Entrance

In some cases there may be multiple independent sensors for one PASSENGER ENTRANCE. E.g. there may be both a door sensor *and* a video based monitoring system, both reporting entry/exit data on the same physical entrance. In these cases the systems needs to report entrance data using *different* entrance Ids, and then an Entrance-Entrance Mapping is used to define that these sensors are covering the same PASSENGER ENTRANCE.

4.4. No Spaces defined

If there are NO Spaces defined for a Vehicle, it shall be assumed that it is a single Space Vehicle. All PASSENGER ENTRANCE COUNT data items should be assumed to be EXTERNAL and PASSENGER SPACE OCCUPANCY COUNT, PASSENGER SPACE ENTRANCE COUNT, etc generated accordingly.

Since a single Space is assumed, both sensor ID and entrance ID shall be accepted as identification of a PASSENGER ENTRANCE COUNT data item, without a need for a SensorEntranceMapping.

> ***SPEC QUESTION:*** This is added to simplify the single-space vehicle/bus case, which does not gain anything by having a Space defined(?) OTOH defining a Space is not hard, and removes this special case. Lets discuss.

5. Data Format JSON

The Data Format for APC-II is JSON. As this is a Data Centric specification more formats could be added in future, but no such needs have been identified for now.

The Data Format JSON section is organized in two sections. First is the static data that provides the information about how the Vehicle is "organized", and then is the data that contains the Passenger Counts themselves.

For brevity the JSON schemas are not included in this document, but can be found on the ITxPT github. For APC-II the schemas are regarded as part of the specification with no distinction in priority. If a conflict or contradiction is found between the schemas and this document an issue should be raised with ITxPT, which will resolve the problem with a minimum impact.

5.1. API version

The property `apiVersion` is used to track the current version of the data. This is *not* the ITxPT specification version, which may change without the data being affected.

> ***NOTE OAB:*** `apiVersion` a good candidate for more general standardization, perhaps as a Data Dictionary standard type.

5.2. APC Static Data

Static Data describes how the Vehicle is configured. It should change only when the vehicle is reconfigured, which should be never for many Vehicles and seldom for the others. In particular the static data will not change during operation. The exception to this could be COMPOUND TRAINs, but even there each TRAIN would stay fixed.

5.2.1. SensorId to entranceId mapping

For Entrance Sensors that identify with the sensorId, a mapping of the sensorId to entranceId must be provided.

```
{
    "$schema": "https://github.com/ITxPT/<something>/sensorId-entranceId-mapping.json",
    "apiVersion": 1.0,
    "sensorId": "AcmeCo-23456789",
    "entranceId": "car2397-door7"
}
```

This is used by the APC Aggregator to know that a PASSENGER ENTRANCE COUNT data item with a sensorId(s) belong to a specific entranceId. While this could be posted by anyone, the use case is that sensors don't (need to) know where they are in the Vehicle, but this is known to some other onboard or back office system.

5.2.2. PASSENGER SPACE CAPACITY

This is the number of Objects that a Space (Vehicle) can accommodate at 100% utilization.

```
{
    "$schema": "https://github.com/ITxPT/<something>/PASSENGER-SPACE-CAPACITY.json",
    "apiVersion": 1.0,
    "spaceId": "Vehicle",
    "capacity": "<see objectCount def>"
}
```

This structure contains the maximum number of *each* objects the Space can accommodate at at 100% utilization. E.g. a Space that can fit 90 adults, 120 children, 2 wheelchairs, and 4 prams, cannot fit these *at the same time* but would have a 100% occupancy at 60 adults, 25 children, 1 wheelchair and 2 prams.

> ***SPEC QUESTION:*** **This is now resolved in text above?** *Previous question:* Is this always in number of Adults, and then any Capacity calculation will need to account for the fact that one wheel chair or pram takes up more space than one Adult? If it always in Adults, perhaps having an `adults` property would be better than having a full objectCount structure. Or is it always the max number of things? If so a priority needs to be established, because while a Space could fit 3 wheelchairs or 4 prams, it cannot fit 3 wheelchairs *and* 4 prams. Perhaps wheelchairs are a bit of a special case, so perhaps this should be "number of wheelchair spaces and number of adults when wheelchair spaces are occupied"?

> ***SPEC QUESTION:*** Should there be separate “seated” and “standing” capacity structures? This is always (?) known, so should be no extra effort to have that, and could potentially benefit systems that want to make judgments about current occupancy levels without knowing anything about the specific space/vehicle? E.g. instead of / in addition to “capacity” also have “seatedCapacity” and “standingCapacity”?

5.2.3. PASSENGER SPACE

The space structures is what APC count structures can be reported on.

```
{
    "$schema": "https://github.com/ITxPT/<something>/PASSENGER-SPACE.json",
    "apiVersion": 1.0,
    // spaceId must be unique in the Vehicle
    "spaceId": "VIN1234567HG",
    // spaceType not a strict enum as custom values are allowed
    "spaceType": "VEHICLE",
    "entrances": [
        {"entranceId": "door1", "entranceType": "EXTERNAL", "direction": "ALIGNED"},
        {"entranceId": "door2", "entranceType": "EXTERNAL", "direction": "ALIGNED"}
    ],
    // Usually one of isPartOfSpaces and containsSpaces are used, not both
    "isPartOfSpaces": [],
    "containsSpaces": ["forward", "mid-door-area", "rear"]
}
```

Above is a Vehicle Space with three sub-spaces with two of these sub-spaces below

```
{
    "$schema": "https://github.com/ITxPT/<something>/PASSENGER-SPACE.json",
    "apiVersion": 1.0,
    // spaceId must be unique in the Vehicle
    "spaceId": "mid-door-area",
    // spaceType not a strict enum as custom values are allowed
    "spaceType": "wheelchair-standing-area",
    "entrances": [
        {"entranceId": "door2",
            "entranceType": "EXTERNAL", "direction": "ALIGNED"},
        {"entranceId": "endForwardSeating",
            "entranceType": "INTERNAL", "direction": "REVERSED"},
        {"entranceId": "startRearSeating",
            "entranceType": "INTERNAL", "direction": "REVERSED"}
    ],
    "isPartOfSpaces": [],
    "containsSpaces": []
}

{
    "$schema": "https://github.com/ITxPT/<something>/PASSENGER-SPACE.json",
    "apiVersion": 1.0,
    // spaceId must be unique in the Vehicle
    "spaceId": "rear-area",
    // spaceType not a strict enum as custom values are allowed
    "spaceType": "seating-area",
    "entrances": [
```

```
        {"entranceId": "startRearSeating",
            "entranceType": "INTERNAL", "direction": "ALIGED"}
    ],
    "isPartOfSpaces": [],
    "containsSpaces": []
}
```

The two spaces are linked by both having the “startRearSeating” entrance.

5.2.3.1. PASSENGER SPACE - spaceType

> ***SPEC QUESTION:*** It is possible standard spaceTypes should be in the Detailed Data Model rather than the JSON section. This comments equally applies to other enums and some other spec-text found in Data Format JSON section.

The intention with spaceType is to make data more readable, but also to allow processes that process output from the APC Aggregator to recognize a few common Space types, and generate reports (or other output) on those without understanding the specific naming of Spaces in all Vehicles.

ÏTxPT standardized names are UPPER CASE, while other names should be lower case, avoiding conflict between non-standard names and future additional standard additions.

Standardized spaceTypes

`COMPOUND_TRAIN` - A space made up of several `TRAINS/VEHICLES`

`VEHICLE` - A Space capable of independent movement. Expectation is that passengers can move between any Spaces that make up the `VEHICLE`.

`TRAIN_ELEMENT` - A car/carriage that makes up part of `VEHICLE`

`UPPER_LEVEL` - Upper level of a bi-level `VEHICLE/TRAIN_ELEMENT`

`LOWER_LEVEL` - Lower level of a bi-level `VEHICLE/TRAIN_ELEMENT`

`STAIRCASE` - A staircase between different levels.

`WHEELCHAIR` - An area with one or more wheelchair spaces.

`SEAT` - A seat for a single `PASSENGER`

`BENCH_SEAT` - A longer seat that may fit two or more PASSENGERs.

> ***SPEC QUESTION:*** More of these we should have? The standard don’t require any action so relatively harmless to add more types I would think.

5.2.4. ENTRANCE FOR PASSENGER SPACE

The entrance list of a Space consists of ENTRANCE FOR PASSENGER SPACE objects. Each object is made up of

```
{
    "$schema": "https://github.com/ITxPT/<something>/ENTRANCE-FOR-PASSENGER-SPACE.json",
    // entranceId must be unique in the Vehicle
    "apiVersion": 1.0,
    "entranceId": "door2",
    "entranceType": "EXTERNAL",
    "direction": "ALIGNED"
}
```

- entranceId must be unique in the Vehicle
- entranceType must be either “EXTERNAL” or “INTERNAL”
- direction must be either “ALIGNED” or “REVERSED”

“ALIGNED” is used when any associated entrance sensor has entry/exit events of the same direction. “REVERSED” is used when entry/exit events are opposite, and an entry event for the entrance (sensor) is a exit event for the Space. If there is no associated sensor the value shall be null.

5.2.5. Entrance Mapping

Entrance Mapping is used when two entrances are next to each other or overlapping, and this makes two entrances work as one entrance. A typical use case would be two TRAIN ELEMENTS pared up so the entrance on the rear of the first carriage is the entrance at the front of the second.

```
{
    "$schema": "https://github.com/ITxPT/<something>/entranceId-mapping.json",
    "apiVersion": 1.0,
    "entranceId1": "car1287-rearEntrance",
    "entranceId2": "car2397-frontEntrance"
}
```

`ALIGNED` is used when an entry/exit is the same event for both entrances. `REVERSED` is used when an entry event for one is an exit event for the other.

5.3. APC Counting Data

Counting data is the data that are actually wanted! These data items contains the information about entry/exit and occupancy of vehicles/spaces.

5.3.1. Time synchronization

APC Counting Data contains timestamps, and data analysis relies on the timestamps of different Providers to be in sync. Producers of APC Counting Data shall ensure that the timestamp is based on a synchronized UTC time. Onboard this means using the onboard SNTP service according to ITxPT S02P02 specification.

> ***SPEC QUESTION:*** Perhaps not the best fit for the “Data format JSON” section. Perhaps have an “Other requirements” section between “Detailed Data Model” and “Data format JSON”? Or perhaps it is a good fit, as other formats may use other mechanisms to connect data?

5.3.2. Quality Factor

The counting data all all have a quality factor, `qf`, which indicates the quality of the *data*. It applies to the changes since the last transmission. It has the values - `HIGH` - No known problem with data. - `MODERATE` - Data *may* have some problems, but usable. - `LOW` - Data *may* have major differences with the actual data. - `ERROR` - The data is not reliable, and should not trusted.

`qf` is the quality of the *data* and not the state of the sensor/detection. E.g. a video based system may work fine, but have low confidence in reported data because someone carried an Billy bookshelf onboard and is obscuring most of the camera field of view.

> ***NOTE OAB:*** `qf` could be a good candidate for more general standardization, perhaps as a Data Dictionary standard type.

5.3.3. APC objectCount structure

The objectCount structure is used in several structures to count objects that enters, exits or are present in vehicles.

```
{
    "$schema": "https://github.com/ITxPT/<something>/apc-object-count.json",
    "apiVersion": 1.0,
    "adults": {"count": 21},
    "children": {"count": 4},
    "wheelchairs": {"count": 0},
    "prams": {"count": 1},
    "bikes": {"count": 0},
    "luggage": {"count": 1,
        "composition": {
            "xs": {"count": 0},
            "small": {"count": 1},
            "medium": {"count": 0},
            "large": {"count": 1},
            "xl": {"count": 0}
        }
    },
    "others": {"count": 3,
        "ext_custom": {"flatpack_furniture": {"count": 1}}
    }
}
```

In example above `"ext_customer": {"flatpack_furniture": {"count": 1}}` is a custom extension.

The object count is reported on several defined object types.

> ***SPEC QUESTION:*** Object types here only a proposal. Also we need to think about sensors / detection methods that support a subset of defined types. Are all of these mandatory to be ITxPT compliant? Are any?
>
> ***SPEC QUESTION:*** There are multiple object types, which cannot cleanly be translated between each other. 1 adult != 1 child; 1 wheelchair != 1 adult; etc. Should there be some toplevel "passenger = float-nbr" or "adultEquiv = float-nbr" that is a estimate how many adult-equivalents the total sum makes up? Perhaps also as part of each type?

Each object type has a structure containing a "count" property with the number of items counted. This can be further broken down into standardized subtypes via the "composition" property, with subtypes having a "count" property of their own. Custom

The defined types are:

- `adults` - A passenger that is not a child
- `children` - Usually delimited from adults by a height threshold
- `wheelchairs` - TBD
- `prams` - TBD
- `bikes` - A bike. This includes all types of bikes, including motorbikes (when applicable). Defined subtypes:
  - `small` - e.g. kick bikes and children's bikes. May not be included in "count".
  - `standard` - a standard bike of some sort
  - `wide` - normal in length, but wider. E.g. three wheel bikes, scooter.
  - `long` - normal in width, but longer. E.g. a tandem cycle.
  - `wideAndLong` - wider and longer than `standard`. E.g. a cargo bike.
- `luggage` - Any carried luggage, including suite cases, backpacks, bags, shopping bags, etc. Only luggages `medium` or larger should be counted towards the "count". Defined subtypes with *examples*:
  - `xs` - A handbag, shopping bag, small backpack. Less than 15-20 litres.
  - `small` - A small suitcase, large shopping bag, normal gym bag 25 - 50 litres.

    - `medium` - A “normal” sized suitcase or similar. 50 - 75 litres.
    - `large` - A larger suite case, backpack, etc. 75 - 100 litres.
    - `xl` - Above 100 litres.
- `animals` - only animals `medium` or larger should be counted towards the “count”.
    - `small` - E.g. cats and lapdogs. Does not use capacity, but may not be permitted in some Spaces.
    - `medium` - A “normal sized” dog or similar
    - `large` - A large dog
- `others` - Anything that does not fit into the categories, or is defeats categorization.

  NOTE OAB: The reason I added `xs` and `small` for luggage was that while these are not useful for vehicle level occupancy, they could be used in *seat level* occupancy where one would like to know how many seats are taken by people putting a bag or similar on them.

  SPEC QUESTION: Should we have sized-based subtypes for `others`? Like `small` - half adult sized, `medium` - adult sized, `large` - double adult sized, `xl` - multiple adults, `xxl` - more than one square meters.

  SPEC QUESTION: Should Space have some sort of ‘restriction’ property where it can be be specified that bikes/animals/etc are not allowed?

5.3.4. PASSENGER ENTRANCE COUNT

Number of entries/exits through a specific Entrance, for a specified *period* in time. Note that counts are cumulative since tsCountStart; to get the entries/exits for a smaller period, e.g. during a stop, the ‘before-value’ must be subtracted from the ‘after-value’. The use of cumulative counts means that the correct entries/exits of an entrance can be calculated even if some messages are lost.

```
{
    "$schema": "https://github.com/ITxPT/<something>/PASSENGER-ENTRANCE-COUNT.json",
    "apiVersion": 1.0,
    "entranceId": "door2",
    "qf": "HIGH",
    "entered": "<objectCount>",
    "exited": "<objectCount>",
    "trigger": "DOORCLOSE",
    "timestamp": "2021-06-09T07:33:04",
    "tsCountStart": "2021-06-09T05:42:38"
    // Do we also want a msgSeqNbr ?
}
```

entered and exited are cumulative counts. Once the Vehicle/Space goes empty -> occupied -> empty, the entered and exited count should - with perfect detection - be the same.

5.3.5. Passenger Count Triggers

An updated Passenger Count (any type) is produced for a reason. As part of the updated count, the reason for produced the update (at that moment) is provided, as a hint to analysis. While a trigger value is required, other values than those defined are allowed; trigger values that are not understood should be skipped by the analysis function, while still processing the count data.

By convention standardized triggers are UPPER CASE. Non-standard values should be lower case to not interfere with future standardization.

`DOORS_CLOSED` - produced directly after a door close event.

`DOORS_CLOSING` - produced while the doors are closing.

`THRESHOLD` - produced after reaching some configured threshold, e.g. 10 entries+exits.

`PERIODIC` - produced because a time period has elapsed since last produced.

`COUNT_ADJUST` - the count has been adjusted, by a non-apc event. E.g. the driver has indicated the Vehicle is empty.

`PROVIDER_RESET` - The producer has reset (intentionally or unintentionally) and start from “zero”.

> ***SPEC QUESTION:*** Should there be `DOORS_OPEN` trigger? While not useful for the counting as such, it could be useful for analysis of how long doors were open without pulling in further data.
>
> ***SPEC QUESTION:*** Should there be `ALL_DOORS_CLOSED` trigger? Could be more descriptive for the SPACE COUNTS than `DOORS_CLOSED` that does not say which if doors close at different times.
>
> ***SPEC QUESTION:*** Also discussed on meeting if `VEHICLE_MOVING` / `LEFT_STOP` would be useful. Don’t think it would be useful by current scope of APC spec that don’t know anything about movement, but it could be useful for higher level reports that does know about STOP PLACES and VEHICLE movement.
>
> ***SPEC QUESTION:*** Are any of these TRIGGERS mandatory for ITxPT label? If yes, which?

5.3.6. PASSENGER SPACE OCCUPANCY COUNT

```
{
    "$schema": "https://github.com/ITxPT/<something>/PASSENGER-SPACE-OCCUPANCY-
        COUNT.json",
    "apiVersion": 1.0,
    "spaceId": "VIN1234566HJ",
    "qf": "HIGH",
    "occupancy": "<objectCount>",
    "trigger": "PERIODIC",
    "timestamp": "2021-06-09T07:33:48"
    // Do we also want a msgSeqNbr ?
}
```

Property “trigger” according to Passenger Count Triggers above.

“timestamp” is the time the Count was produced, or if production is not real-time the timestamp of the newest input data to the production.

> ***SPEC QUESTION:*** For some methods like weight-based sensors it may be difficult to produce a valid count?
>
> ***SPEC QUESTION:*** If PSOC is based on entrance sensors, then if they are registering more people in one direction than the other, then over a full day, this could mean the occupancy has little connection to reality. This standard should probably recommend that OCCUPANCY COUNT is reset when the Vehicle/Space is (presumed) empty, but should the mechanism for signaling such a reset be part of the spec?

5.3.7. PASSENGER SPACE ENTRANCE COUNT

Number of entries/exits through all entrances of a space. Has a spaceId instead of a sensorId/entranceId. Apart from that identical to PASSENGER SPACE ENTRANCE COUNT

```
{
    "$schema": "https://github.com/ITxPT/<something>/PASSENGER-ENTRANCE-COUNT.json",
    "apiVersion": 1.0,
    "spaceId": "car3",
```

```
    "qf": "HIGH",
    "entered": "<objectCount>",
    "exited": "<objectCount>",
    "trigger": "DOORCLOSE",
    "timestamp": "2021-06-09T22:33:30",
    "tsCountStart": "2021-06-09T05:42:38"
    // Do we also want a msgSeqNbr ?
}
```

6. Transport MQTT using JSON format

The APC-II specification uses MQTT to transport messages with Format JSON. As this is a Data Centric specification more combinations of formats and transport could be added in the future, but no such needs have been identified for now.

6.1. General Requirements

All MQTT Providers specified in this specification and Consumers using this data shall follow the requirements for MQTT Clients in S02P00 (Info: this only exists as internal draft)

6.2. Security Considerations

The potential security impact of APC are low. Disrupted or incorrect APC data are unlikely to lead to any safety concerns or serious operational problems.

6.3. Performance Considerations

For a simple vehicle there is unlikely to be more than half a dozen entrance sensors, sending a few messages per minute. Including an onboard APC aggregator, there should still be less than one message per second. A more complex vehicle like a bi-level EMU *may* have multiple messages per second in total, but this should have a negligible impact on performance.

Performance impact of APC is low.

6.4. Functional Group and Provider Names

APC has its own functional Group apc. (To Be Confirmed!)

APC has four types of providers:

`apc_static` - data about the vehicle. This is not tied to a `apc_static` provider, and *may* be posted by/with any Provider (ID).

`entrance_counts` - information about entry/exit for an entrance

`space_entrance_counts` - information about entry/exit for a space.

`occupancy_count` - information about occupancy of a space.

6.5. MQTT APC Topic tree

topic	payload	retain	msg exp
apc			
apc_static			
[Provider ID]			
spaces			
[spaceId]	PASSENGER SPACE	True	not set (inf)
sensor_entrance_mappings			
[sensorId-entranceId]	sensorId-entranceId mapping	True	not set (inf)
entrance_entrance_mappings			
[entranceId1-enctranceId2]	entranceId1-entranceId2 mapping	True	not set (inf)
entrance_counts			
[Provider ID]			
[sensorId] OR [entranceId]	PASSENGER ENTRANCE COUNT	False?	?
space_entrance_counts			
[Provider ID]			
[spaceId]	PASSENGER SPACE ENTRANCE COUNT	False?	?
occupancy_counts			
[spaceId]	PASSENGER SPACE OCCUPANCY COUNT	False?	?

MQTT Topic Tree

6.6. Note on Provider IDs

If an entity publishes on multiple of the `*_counts` Providers it *may* use the Provider name `apc_counts` for all of them, rather than a separate Provider name/ID for each.

6.7. Static Data

Static data is published under `apc/apc_static`. There are no rules on which entities/providers publish the static data; as long as the needed static data is available it is within spec.

The `[spaceId]`, `[sensorId-entranceId]` and `[entranceId1-enctranceId2]` part of the topics exists to separate data items of the same type and shall not be used by the clients. The Client shall use the Ids found in the the payload. Subscribes shall be to `..apc/vehicledefine/*/spacedefines/*/*`, `..apc/vehicledefine/*/sensorentrancemapping/*/*` and `..apc/vehicledefine/*/entranceentrancemapping/*/*`.

6.8. `entrance_counts` Providers

The `entrance_counts` Provider(s) are *not* required to be a physical sensors at each entrance. E.g. in a video based system the “video analysis engine” could provide `PASSENGER ENTRANCE COUNT`.

6.8.1. Inventory

In addition to what is required by Inventory MQTT (which is TBD) the provider shall **TBD**. (But probably something about how it is configured at least.)

6.9. `space_entrance_counts` Providers

6.9.1. Inventory

In addition to what is required by Inventory MQTT (which is TBD) the provider shall **TBD**. (But probably something about how it is configured at least.)

6.10. `space_occupancy_counts` Providers

6.10.1. Inventory

In addition to what is required by Inventory MQTT (which is TBD) the provider shall **TBD**. (But probably something about how it is configured at least.)