The Taxi Registry

Operator's Guide

Version	Description	Author	Date
1.0	Initial	David Beaudoin	16/08/2017
1.1	Révision	Stéphane Leblanc	07/09/2017
1.2	Révision	David Beaudoin	25/01/2018
1.3	Révision	Sébastien Blais	22/11/2018
1.4	Taxi position and status Mandatory column	Gaston-Andres Bellei	06/12/2019
1.5	Sections 2.4 and 3.3 updated	Gaston-Andres Bellei	15/06/2020
1.6	Sections 2.4 and 4.1 updated	Gaston-Andres Bellei	21/07/2020
1.7	Sections 2,1, 2.2, 2.3, 2.4 and 4.1 updated	Gaston-Andres Bellei	29/07/2020
1.8	Updated example column in section 4.1	Brian Di Croce	19/08/2020
1.9	Various text corrections	Gaston-Andres Bellei	10/11/2020
1.10	Bill 17 compliance added and general formatting	Gaston-Andres Bellei	20/11/2020
1.11	Change related to obsolete attributes in json	Daniel Brodeur	08/01/2021
1.12	Minor changes.	Stéphane Leblanc	02/02/2021
1.20	Change related to fake endpoints + miscellaneous	Matthieu Perrin	15/04/2021
1.21	Favor deep link over hailing	Matthieu Perrin	27/06/2022

Table of contents

1. Introduction

1.1 Overview

The Taxi Registry aim is to connect taxis and their customers. Customers can use taxi search engines to book taxis geolocated by taxi operators. The Taxi Registry mediates between search engines and operators.

All interactions with the Taxi Registry can be made from the central infrastructure of taxi operators, this also includes communication of the location and availability of taxis (using on-board equipment in the taxi).

1.2 Process of integration

To integrate with the Taxi Registry, the operator must send its contextual data (section 2) and the position and status of its taxis (section 3). Once the development is done in the acceptance environment, the operator must contact the Taxi Registry support team. When the Taxi Registry support team has verified that the operator is properly integrated, an API key will be sent to the operator for the production environment.

Sending the positions and status of the taxis is **the first milestone for the operator**. The law states that the taxi owners must send the position and the status of their taxis to the Taxi Registry via an authorized taxi operator. Operators will be given an API key for the production environment even if they do not offer a taxi booking solution (section 4).

Operators with a taxi booking solution (phone, website and/or applications) can be promoted to the general public by the Taxi Registry. Indeed, the Taxi Registry can promote the operator to the customers that use trip planning apps integrated with the Taxi Registry. Although an operator that offers only by phone booking can be promoted by the Taxi Registry, we strongly encourage the operators to support deep linking with their mobile applications to offer the best user experience. Section 4 describes how an operator can be promoted to the general public by the Taxi Registry.

The Taxi Registry support team can be contacted at: support.taxi.exchange.point@montreal.ca

Here are the links to communicate with the Taxi Registry services:
Acceptance : https://taximtl.accept.ville.montreal.qc.ca
Production : https://taximtl.ville.montreal.qc.ca

1.3 Mandatory HTTP Headers

The following HTTP Headers are mandatory for all requests to the Taxi Registry REST APIs:

Name	Value	Description
Accept	Application/json	Media types which are acceptable for the response
X-VERSION	2	Version of the API
X-API-KEY	token	API Key

1.4 Authentication

Authentication of your application is done for each query to the Taxi Registry by including a HTTP header X-API-KEY.

API keys are available for accredited developers and will be distributed by the BTM (Bureau Taxi Montréal) upon demand and validation.

1.5 APIs implementation

This documentation provides an overview of the Taxi Registry REST APIs. REST APIs provide access to resources (data entities) via URL paths. To use a REST API, your application will make an HTTPS request and parse the response. Your methods will be the standard HTTP methods like GET, PUT and POST. REST APIs operate over HTTPS making it easy to use with any programming language or framework. The input and output formats for the Taxi Registry REST APIs are JSON.

Note that data is isolated for each operator. No operator can see the other operator's data.

2. Contextual Data

2.1 Registering a Driver

The structure of the required driver object is described below. You should push this information on a daily basis to keep the data up to date.

Calls to this API are idempotent: you can update a driver simply by submitting the updated driver object with the same post method. If the department or professional license is different, a new driver will be created; if the department and professional license are unchanged, the driver will be updated.

Status on create	Status on update	Unique identifier(s)
201	200	departement and professional_licence

POST /api/drivers

Parameters

*Body (JSON) ** Send only one item at a time*

```
{
  "data": [
    {
      "birth_date": "1950-12-22",
      "departement": {
        "nom": "Québec",
        "numero": "1000"
      },
      "first_name": "Jon",
      "last_name": "Doe",
      "professional_licence": "L1531-171274-08"
    }
  ]
}
```

Response (JSON) status 200 / 201

```
{
  "data": [
    {
      "birth_date": "1950-12-22",
      "departement": {
        "nom": "Québec",
        "numero": "1000"
      },
      "first_name": "Jon",
      "last_name": "Doe",
      "professional_licence": "L1531-171274-08"
    }
  ]
}
```

Key	Value Type	Description
departement	department object	The departement object is constituted of the identifier numero and the name (nom) of the local authority. When a new driver is created by an Operator, an empty string or null can be passed instead of the name nom: only the identifier numero is used by the Taxi Registry. For Quebec, Since the adoption of Bill 17, the department should always be: departement.nom: "Québec" and departement.numero: 1000. When departement.numero is 1000 (Québec), the driver is identified by it's SAAQ driver's license number. Before Bill 17, drivers were part of the departement 660(Montreal) and were identified by their 'pocket number'.
professional_licence	string	Professional license number of the driver. It is often a string of digits but it might for some departments contain letters or other characters like dash or slashes. Warning: this identifier is not unique at the national level: two local authorities can each assign the same number to different drivers. Warning: the typo "licence" (French writing) instead of "license" (English writing) is still in the API (as of version 2). The couple of this professional license number (professional_licence) and the licensing local authority (departement) is used as the driver identifier when declaring a taxi as a vehicle/driver/license triplet. For Quebec, Since the adoption of Bill 17, the SAAQ driver's license number is used as the professional_licence. Before Bill 17, drivers were part of the departement 660(Montreal) and were identified by their 'pocket number'. For historical reasons, values of professional_licence are case sensitive. Even though, in reality, professional_licence should not have lower case letters.

last_name	string	Last name of the driver.
first_name	string	First name of the driver.
birth_date	string, RFC3339	Birth date of the driver in "YYYY-MM-DD" format. For Quebec, the birth date is ignored for privacy reasons.

2.2 Registering a Vehicle

The structure of the required vehicle object is described below. You should push this information on a daily basis to keep the data up to date.

Calls to this API are idempotent: you can update a vehicle simply by submitting the updated vehicle object with the same post method. If the license plate is different, a new vehicle will be created; if the license plate is unchanged, the vehicle will be updated.

Response on create	Response on update	Unique identifier(s)
201	200	licence_plate

POST /api/vehicles

Parameters

*Body (JSON) ** Send only one item at a time*

```
{
  "data": [
    {
      "licence_plate": "FAB1234",
      "vehicle_identification_number": "1FTFW1R6XBFD08251",
      "air_con": true,
      "horodateur": "aa",
      "color": "gris",
      "date_dernier_ct": "2016-12-22",
      "date_validite_ct": "2016-12-22",
      "credit_card_accepted": true,
      "electronic_toll": true,
      "fresh_drink": true,
      "pet_accepted": true,
      "tablet": true,
      "dvd_player": true,
      "taximetre": "aa",
      "every_destination": true,
      "nfc_cc_accepted": true,
      "baby_seat": true,
      "special_need_vehicle": true,
      "amex_accepted": true,
      "gps": true,
      "engine": "GO",
      "cpam_conventionne": true,
      "relais": true,
      "bank_check_accepted": true,
      "luxury": true,
      "horse_power": 2.0,
      "model_year": 1995,
      "wifi": true,
      "type_": "sedan",
      "nb_seats": 0,
      "constructor": "audi",
      "bike_accepted": true,
      "model": "a4"
    }
  ]
}
```

Response (JSON) status 200 / 201


```
{
  "data": [
    {
      "licence_plate": "FAB1234",
      "vehicle_identification_number": "1FTFW1R6XBFD08251",
      "air_con": true,
      "amex_accepted": true,
      "baby_seat": true,
      "bank_check_accepted": true,
      "bike_accepted": true,
      "color": "gris",
      "constructor": "audi",
      "cpam_conventionne": true,
      "credit_card_accepted": true,
      "date_dernier_ct": "2016-12-22",
      "date_validite_ct": "2016-12-22",
      "dvd_player": true,
      "electronic_toll": true,
      "engine": "GO",
      "every_destination": true,
      "fresh_drink": true,
      "gps": true,
      "horodateur": "aa",
      "horse_power": 2,
      "id": 36,
      "luxury": true,
      "model": "a4",
      "model_year": 1995,
      "nb_seats": 0,
      "nfc_cc_accepted": true,
      "pet_accepted": true,
      "private": false,
      "relais": true,
      "special_need_vehicle": true,
      "tablet": true,
      "taximetre": "aa",
      "type_": "sedan",
      "wifi": true
    }
  ]
}
```

Key	Value Type	Description
licence_plate	String	Mandatory - License plate of the vehicle. Warning: the typo "licence" (French writing) instead of "license" (English writing) is still in the API (as of version 2). Before Bill 17, vehicles used to have a license plate starting with the letter T. With the recent changes the bill introduces, all vehicles will be issued a new plate either for commercial purpose (first letter F) or promenade. Plates starting with the letter T will be prohibited.The licence_plate is used as the vehicle identifier to declare a taxi as a vehicle/driver/license triplet. For historical reasons, values of licence_plate are **case sensitive**. Even though, in reality, license plates should not have lower case letters.
vehicle_identification_number	String	Optional - The licence_plate is the only mandatory identifier for vehicles in the Taxi Registry. Even though it is an optional attribute, the vehicle identification number must be transmitted when available.
constructor	String	Mandatory - Constructor of the vehicle.
model	String	Mandatory - Model of the vehicle.
color	String	Color of the vehicle.
type_	String	Type of the vehicle. The possible values are sedan, station_wagon, normal or mpv. Warning: the name of this key is type_ with the final underscore. If your type is not listed use "type_": null.
nb_seats	Integer	Number of seating positions available for passengers in the vehicle (not counting the seat of the driver). As per European Regulation EU/678/2011 the following requirements apply for the counting of the seating positions: (a) each individual seat shall be counted as one seating position; (b) in the case of a bench seat, any space having a width of at least 400 mm measured at the seat cushion level shall be counted as one seating position. (c) however, a space as referred to in point (b) shall not be counted as one seating position where: (i) the bench seat includes features that prevent the bottom of the manikin from sitting in a natural way - for example: the presence of a fixed console box, an unpadded area or an interior trim interrupting the nominal seating surface; (ii) the design of the floor pan located immediately in front of a presumed seating position (for example the presence of a tunnel) prevents the feet of the manikin from being positioned in a natural way. When available, the area intended for an occupied wheelchair shall be regarded as one seating position.

air_con	Boolean	This vehicle is equipped with air conditioning.
amex_accepted	Boolean	This vehicle accepts American Express card for any amount (no minimum).
baby_seat	Boolean	This vehicle is equipped with a baby seat.
bank_check_accepted	Boolean	This vehicle accepts national bank checks (foreign bank checks might still be refused).
bike_accepted	Boolean	This vehicle can transport a bicycle.
credit_card_accepted	Boolean	This vehicle accepts credit card payments for any amount (no minimum). This should be true for vehicles accepting at least Visa and MasterCard. There is a different Boolean amex_accepted for American Express.
dvd_player	Boolean	This vehicle has a DVD player at the disposal of customers during the ride.
electronic_toll	Boolean	This vehicle is equipped with an electronic device letting them use express toll booths on toll roads.
every_destination	Boolean	As per the French regulation, taxis can refuse service to customers whose destination is not within their zone. Some taxis do accept any destination outside of their zone. The every_destination boolean should be false by default, and true for taxis who renounce their right to refuse service to customers depending on their destination.
fresh_drink	Boolean	This taxi offers refreshments.
gps	Boolean	This vehicle is equipped with GPS navigation.
luxury	Boolean	This is a luxury vehicle.
nfc_cc_accepted	Boolean	This vehicle accepts NFC credit card payments.
pet_accepted	Boolean	This vehicle can accommodate pets (understood as cats or small dogs; other large or unusual pets might still be refused).
special_need_vehicle	Boolean	Wheelchair accessible vehicle as defined in "EU/678/2011" (which amends 2007/46/EC). Vehicles constructed or converted specifically so that they accommodate one or more persons seated in their wheelchairs when traveling on the road.
tablet	Boolean	This vehicle has a digital tablet at the disposal of the customers during the ride.
wifi	Boolean	This vehicle has complimentary Wi-Fi aboard.
cpam_conventionne	Boolean	This vehicle has a convention with social security to transport patients. This field is used for administrative purposes only. When a new vehicle is created by an Operator, this field can be omitted or passed with a null value.

date_dernier_ct	string, RFC3339	Date of the latest compulsory roadworthiness tests in "YYYY-MM-DD" format. This field is used for administrative purposes only. When a new vehicle is created by an Operator, this field can be omitted or passed with a null value.
date_validite_ct	String, RFC3339	Expiration date of the latest compulsory roadworthiness tests in "YYYY-MM-DD" format. This field is used for administrative purposes only. When a new vehicle is created by an Operator, this field can be omitted or passed with a null value.
engine	String	Engine type of the vehicle. This field is used for administrative purposes only. When a new vehicle is created by an Operator, this field can be omitted or passed with a null value.
horse_power	Integer	Fiscal power of the vehicle. This field is used for administrative purposes only. When a new vehicle is created by an Operator, this field can be omitted or passed with a null value.
model_year	Integer	Model year of the vehicle. This field is used for administrative purposes only. When a new vehicle is created by an Operator, this field can be omitted or passed with a null value.
relais	Boolean	True if this vehicle is a temporary replacement vehicle for a fully licensed one. This field is used for administrative purposes only. When a new vehicle is created by an Operator, this field can be omitted or passed with a null value.
taximetre	String	Brand and model of the taximeter. This field is used for administrative purposes only. When a new vehicle is created by an Operator, this field can be omitted or passed with a null value.
horodateur	String	Brand and model of the time clock. This field is used for administrative purposes only. When a new vehicle is created by an Operator, this field can be omitted or passed with a null value.
id	Integer	This field is used for administrative purposes only. When a new vehicle is created by an Operator, this field can be omitted or passed with a null value. There is no need for Operators or Search Engines to store the value returned by the Taxi Registry: the field used to uniquely identify vehicles in all transactions with the Taxi Registry is the licence_plate.
~~private~~	~~Boolean~~	**Obsolete**. See section “2.4 - Declaring a Taxi” instead.

2.3 Registering a Owner/license (ADS)

The structure of the required ads object is described below. You should push this information on a daily basis to keep the data up to date.

Calls to this API are idempotent: you can update an owner (ADS) simply by submitting the updated ads object with the post method. If the insee or numero is different, a new owner (ADS) will be created; if the insee and numero are unchanged, the owner (ADS) will be updated.

Owner (ADS) vs license (ADS)
Following adoption of Bill 17, the meaning of ADS has changed from license to owner.

The owner of a vehicle used as a taxi requires a license for each vehicule he owns. Before Bill 17, the Taxi Registry was keeping track of each individual license. Since the adoption of Bill 17, the Taxi Registry does not keep track of each individual license anymore. The taxi registration does now only keep track of the owner and the vehicles he owns. An owner may own many vehicules.

The ADS with the meaning of owner can be differentiated from the ADS with the meaning of license by the value of the taxi zone (insee). If the taxi zone is 1000 (Québec), it means owner, otherwise it means license.

Zone	ADS.insee	ADS.numero	Name	ADS.vdm_vignette
Before Bill 17				
Montréal Est	102005	4M000000011A	John Doe	8811
Montréal Est	102005	4M000000022B	John Doe	8822
After Bill 17				
Québec	1000	161555777	John Doe	Unused

Response on create	Response on update	Unique identifier(s)
201	200	insee and numero

POST /api/ads
Parameters *** Send only one item at a time*
Body (JSON)

```
{
  "data": [
    {
      "category": "",
      "insee": "1000",
      "numero": "161555777",
      "owner_name": "Co-op",
      "owner_type": "company",
      "doublage": false,
      "vdm_vignette": "string"
    }
  ]
}
```

Response (JSON) status 200 / 201


```
{
  "data": [
    {
      "category": "",
      "doublage": false,
      "insee": "1000",
      "numero": "161555777",
      "owner_name": "Co-op",
      "owner_type": "company",
      "vehicle_id": null,
      "vdm_vignette": "string"
    }
  ]
}
```


Key	Value Type	Description
insee	string	Bill 17 abolishes the existing taxi zones, Montreal West (A12), Montreal downtown (A11) and Montreal East (A5), with the exception of the YUL airport zone which is under federal jurisdiction. Since the adoption of Bill 17, all taxis, in Quebec province, belong to the 1000(Québec) taxi zone. Before the adoption of Bill 17, ADS were identified by their CTQ license number. ~~Three agglomerations exist for Montreal as follow:~~ ~~102005 : A5 - Eastern part of the island of Montreal~~ ~~102011 : A11 - Downtown/center Montreal~~ ~~102012 : A12- West part of the island of Montreal~~

numero	string	After Bill 17, numero represents: The SAAQ file number identifying a company or an individual that owns a vehicle. The SAAQ file number format varies depending on the owner being a company or an individual. Ex: company = 161902393 Ex: individual = L1531-171274-08 For individuals, the SAAQ file number is the same as the driver license number. See section *Owner (ADS) vs license (ADS)* above for more details. The couple ADS.insee and ADS.numero is used when declaring a taxi as a driver/vehicle/license triplet. Before the adoption of Bill 17, ADS were identified by their CTQ license number (12 alphanumeric characters). For historical reasons, values of numero are case sensitive. Even though, in reality, numero should not have lower case letters.
owner_name	string	Name of the holder of the license. Warning: It might be either an individual or a company.
owner_type	string	The two possible values are company or individual.
category	string	This field is used for administrative purposes. When a new license (aka ADS) is created by an Operator, an empty string has to be passed (not a null value).
doublage	boolean	Some regulations specific to the Paris area limit the working hours of the driver to 10 hours a day. Some licenses (ADS) can be used for 2 shifts a day (by two different drivers) and this field should then be set to true. Others can only be operated 10 hours a day and this field should be set to false. When a new license (aka ADS) is created by an Operator, this field should always be set to false if the local authority in the insee field is not 75056 (i.e. Paris).

vdm_vignette	string	This field represents the "Vignette" number given by the BTM (Bureau Taxi Montreal). Mandatory. This is ignored when *insee* (see above) is Québec-1000 zone.
~~vehicle_id~~	~~integer~~	**Obsolete**. The association between a vehicle and an owner (ADS) is done via the taxi. See section “2.4 - Declaring a Taxi”.

2.4 Declaring a Taxi

Status attribute is **OBSOLETE** and will be ignored.

The structure of the required taxi object is a minimalist version containing only the identifiers of the vehicle, driver and ads and the initial status of the taxi. The vehicle, driver and ads used to compose a taxi need to have been registered first through their respective API.

This request should be used on new taxi creation or when the private attribute changes.As per drivers, ads, vehicles etc, we recommend that the taxi is updated on a daily basis so the information is always up to date..

If successful, the API returns the complete taxi object as described including the characteristics of the vehicle and most importantly the unique identifier id of the taxi that will be used for subsequent communications.

Calls to this API are idempotent: if you resubmit the same triplet of vehicle, driver and ads, the taxi returned will have the same id.

Private parameters can be updated via this POST request.

Warning: Please make sure to save the returned Id, it will be required to update the taxi later on.

Response on create	Response on update	Unique identifier(s)
201	200	licence_plate (vehicle) and departement and professional_licence (driver) and insee and numero (ads)

POST /api/taxis

Parameters

*Body (JSON) ** Send only one item at a time*

```
{
  "data": [
    {
      "private": true,
      "vehicle": {
        "licence_plate": "FAB1234"
      },
      "driver": {
        "departement": "1000",
        "professional_licence": "L1531-171274-08"
      },
      "ads": {
        "insee": "1000",
        "numero": "161555777"
      }
    }
  ]
}
```

Response (JSON) status 200 / 201


```
{
  "data": [
    {
      "ads": {
        "insee": "1000",
        "numero": "161555777"
      },
      "crowfly_distance": 0.00145,
      "driver": {
        "departement": "1000",
        "professional_licence": "L1531-171274-08"
      },
      "id": "ueXs7TR",
      "last_update": null,
      "operator": null,
      "position": {
        "lat": null,
        "lon": null
      },
      "private": true,
      "rating": 4.5,
      "vehicle": {
        "licence_plate": "FAB1234",
        "characteristics": null,
        "color": null,
        "constructor": null,
        "model": "a4",
        "nb_seats": null,
        "type_": "sedan"
      }
    }
  ]
}
```

Key	Value Type	Description
ads	ADS	A partial ADS object with only the fields: insee, numero. When *ADS.insee* is Québec-1000 then: - *driver.departement* **must** be 1000 (Québec). - *vehicle.licence_plate* **cannot** be a 'T' license plate. For more detail, see section 5.3.4.
driver	driver	A partial driver object with only the fields: departement, professionnal_licence.
vehicle	vehicle	A partial vehicle object with only the fields: characteristics, color, constructor, licence_plate, model, nb_seats. Warning: some of those fields might not be returned (or be returned with a null value) if they were not provided by the taxi operator.
id	string	A long-lived 7 characters long identifier generated for this vehicle/ads/driver triplet by the Taxi Registry. This field should be omitted by operators when declaring a new taxi through a POST request; the newly generated id will be returned in the taxi object sent back as the response.
operator	string	Login of the certified operator.
private	boolean	As per VDM and BTM's requirements, as an option, you can set the taxi's private field to true or false. By default, the taxi's private field is set to false. A private taxi will never be promoted by the Taxi Registry even if it's the closest taxi from a customer. This field is typically used for independent taxi as they may deal with an operator to satisfy their geolocation obligation, but do not pay to receive ride requests. These taxis can therefore only be hailed on the street.

type_	String	Type of the vehicle. The possible values are sedan, station_wagon, normal or mpv. Warning: the name of this key is type_ with the final underscore. If your type is not listed use "type_": null.
rating	float	The mean of the ratings of last rides of the taxi. It is calculated by the Taxi Registry and falls between 0 and 5.
position	{lat, lon}	The latitude and longitude of the taxi. Warning: those values are only returned by the Taxi Registry in the response to a GET request on the /taxis/ API looking for taxis around a customer. They will be nulled when returned in the response to a GET request on the /taxis/{taxi_id}/ API looking for information on a specific taxi.
last_update	integer	Timestamp of the last geolocation update of the taxi. The format is the usual Unix time (IEEE P1003.1 POSIX) and as such is UTC (no timezone).
~~crowfly_distance~~	~~float~~	**Obsolete**. ~~The crow flies distance between the taxi and the customer. (km)~~
~~status~~	~~status~~	**Obsolete**. ~~Status of the taxi.~~ ~~The hailing feature is replaced by the deep link approach.~~

3. Taxi Positions and Status

3.1 Updating the location and status of a taxi

You should push this information in batches every 5 seconds to keep the data up to date.

The JSON payload should be as follows.

POST /api/taxi-position-snapshots
Parameters
Body (JSON) **items should contain all your taxis
{

```
   "items": [
     {
        "timestamp": "1430076493",
        "operator": "coop",
        "taxi": "tPc79rW",
        "lat": "45.38852053",
        "lon": "-73.84394873",
        "device": "phone",
        "status": "free",
        "version": "2",
        "speed": "50",
        "azimuth": "180"
     }
  ]
}
```

Key	Value Type	Description	Mandatory
timestamp	string	*Exact time at which the **location was determined by the taxi**, formatted as a Unix time (IEEE 1003.1-2008 POSIX).* ***Warning: as per the POSIX specification, this should be UTC time without any timezone information.*** ***Warning: Do not send locations in the future (or older than 1 minute) as they will return a http 400 error. Timestamp must be in second.***	Yes
operator	string	*Login of the certified operator.*	Yes
taxi	string	*The id of the taxi is the id that was sent back when the taxi was declared (see Declaring a taxi).*	Yes
lat	string	*Latitude of the taxi.* *Accepted range: -85.05112878 to 85.05112878.* *This should be in JavaScript double precision floating-point format, with decimal separator ".".* *You can truncate the values to 6 decimal places if you want to keep the payload as short as possible (6 decimal places is worth up to 10 cm).*	Yes

lon	string	*Longitude of the taxi.* *Accepted range: -180 to 180.* *This should be in JavaScript double precision floating-point format, with decimal separator ".".* *You can truncate the values to 6 decimal places if you want to keep the payload as short as possible (6 decimal places is worth up to 10 cm).*	Yes
device	string	*phone, tablet, taximeter or otherdevice.*	Yes
status	string	*Possible values: answering, free, occupied, off, oncoming or unavailable.* *Mandatory.* *For more details, see the table below.*	Yes
version	string	*"2" for now (geolocation version 2 of the API).*	Yes
speed	string	*The actual speed of the taxi (in km/h).*	Yes
azimuth	string	*The current orientation of the taxi (360°).* *Accepted range: 0 to 360.*	Yes

3.2 Taxis Status

Value	Description	Mandatory
answering	*The taxi is currently answering a ride request.*	*No, use unavailable if the information is not available.*
free	*The taxi is free to receive ride requests..*	*yes*
occupied	*The taxi has a customer on board.*	*yes*
off	*The taxi is not logged in or did not update its location recently enough.*	*yes*

oncoming	*The taxi is on its way to meet a customer.*	*No, use unavailable if the information is not available.*
unavailable	*The taxi is logged in, but cannot receive ride requests.*	*yes*

3.3 Updating the status of a taxi

This section is **OBSOLETE**. Please notice the following:

1. It is now recommended to use POST /api/taxis (section 2.4) to modify the private attribute.
2. Even if it is discouraged to use PUT /api/taxis/{taxi_id}, it is still a valid request.
3. Only status change submitted through POST /api/taxi-position-snapshots (section 3.1) will be considered. Sending status value through PUT /api/taxis/{taxi_id}, is still possible but status value will be ignored.

~~The status of the taxi should be sent to the Taxi Registry whenever there is a change of status from the operator. The possible status is free or occupied or off or answering or oncoming. This is done through a "HTTPS PUT request to the /taxis/{taxi_id}/ API".~~

~~You can only update the following attributes: status and private. For more details, see the attribute description in section 2.4 "Declaring a taxi".~~

~~PUT /api/taxis/{taxi_id}~~
~~**Parameters**~~
~~**Taxi_id (string)**~~
~~*Body (JSON) ** Send only one item at a time*~~

```
{
  "data": [
    {
      "status": "free",
      (mandatory)"private": "false"  (string or boolean)
    }
  ]
}
```

~~**Response**~~
~~*Return the taxi's details in JSON (see below* 3.4 Querying a taxi*)*~~

3.4 Querying a taxi

In order to check that the updating of the status or location of the taxi worked properly, you can use a “HTTPS GET request to the /taxis/{taxi_id}/ API”.

Warning: the GET /taxis/{taxi_id}/ API will return the status and the last_update (in UNIX TIME) but the “lat” and“lon” will be null (for privacy reasons).

Warning: in production, you should almost never need the GET /taxis/{taxi_id}/ API. The endpoint is provided only to improve the developer experience by allowing them to know the status, ads, driver and vehicle of a taxi.

GET /api/taxis/{taxi_id}

Parameters

Path

taxi_id (string)(required)

Response (JSON) status 200

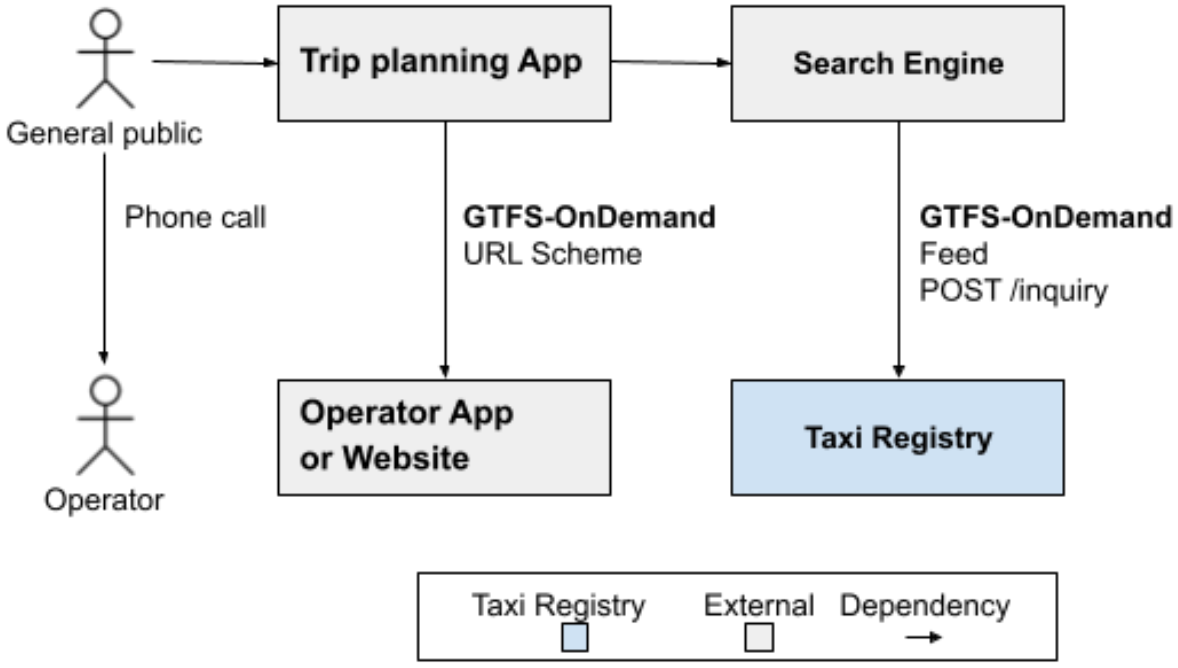

```
{
  "data": [
    {
      "ads": {
        "insee": "1000",
        "numero": "161555777"
      },
      "crowfly_distance": 0.00145,
      "driver": {
        "departement": "1000",
        "professional_licence": "L1531-171274-08"
      },
      "id": "VsLwptA",
      "last_update": 1502819736,
      "operator": "coop",
      "position": {
        "lat": null,
        "lon": null
      },
      "private": false,
      "rating": 4.42332039594968,
      "status": "answering",
      "vehicle": {
        "licence_plate": "FAB1234",
        "characteristics": [
          "every_destination",
          "gps",
          "pet_accepted",
          "bike_accepted",
          "credit_card_accepted",
          "luxury"
        ],
        "color": "GRISE",
        "constructor": "TOYOTA",
        "model": "SIENNA",
        "nb_seats": 6,
        "type_": "sedan",
      }
    }
  ]
}
```


See section 2.4 Declaring a Taxi for details about the different fields.

4. GTFS-OnDemand

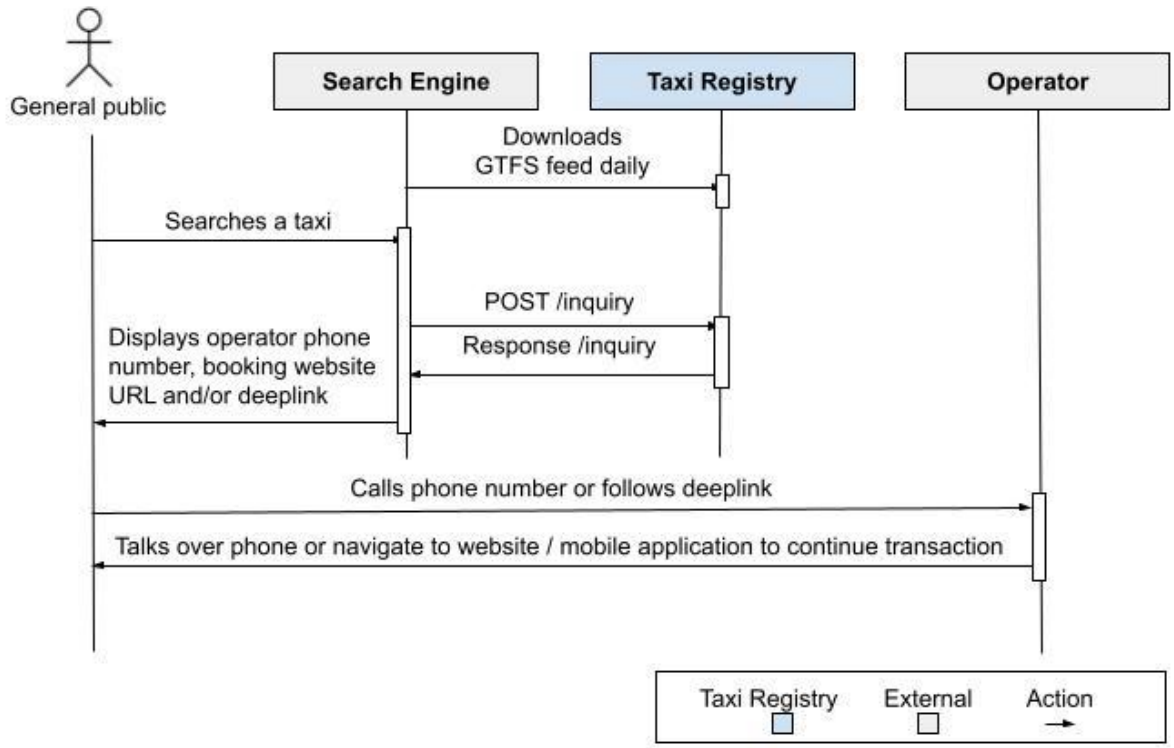
GTFS-OnDemand is an extension to the GTFS specification. The General Transit Feed Specification (GTFS) is a data specification that allows public transit agencies to publish their transit data in a format that can be consumed by a wide variety of software applications.

The GTFS-OnDemand extension aims at facilitating the discoverability and booking of OnDemand transportation services, such as a taxicab service. The specification describes a standard way of communicating between the different actors.



On the server-side, search engines interact with the Taxi Registry by downloading a standard GTFS-OnDemand feed that describes the services offered by the operators and how to book them. Search engines also send requests to the Taxi Registry via POST /inquiry (an endpoint standardized by the GTFS-OnDemand specification) to receive an estimate for a trip with the operator that has the closest taxi to the customer. The Taxi Registry ensures the equity between operators by always promoting the operator with the closest taxi to the customer.

On the customer-side, the customer can book the operator's promoted by the Taxi Registry directly with the operator's app or website. In this case, the trip planning apps interact with the operator app or website by following URLs that respect the GTFS-OnDemand URL Scheme. After following the URL, the customer continues directly on the operator app or website to book the trip. The customer can also book the operator promoted by the Taxi Registry by phone.



4.1 GTFS-OnDemand Deep Link Compliance

The Taxi Registry offers several ways to book a taxi:

- By phone
- Through a website,
- Through an Android app,
- Through an iOS app,

In order to be promoted by the Taxi Registry and receive ride requests, the operator needs to offer booking at least with a phone number, a website address, or both Android and iOS mobile apps. The following subsection focuses on the URL Scheme, and does not apply to booking by phone.

Although an operator that offers only booking by phone can be promoted by the Taxi Registry, we strongly encourage the operators to support deep linking with your mobile applications to offer the best user experience.

4.1.1 URL Scheme

To support booking with an application through the Taxi Registry, a mobile application or a website must comply with the GTFS-OnDemand URL scheme. In order to offer services on mobile devices, an operator must either support both Android and iOS platforms or offer a separate way of booking (phone number or web for instance) and just use one platform (Android only for instance).

To facilitate the integration process, the Taxi Registry **does not support** standard deep links. Deep linking for mobile applications **must use** Android App Links or iOS Universal Links which are more recent and easier to use (see this link to differentiate between deep links formats).

To conform to the GTFS-OnDemand URL scheme, a mobile application or a website must support HTTPS GET requests with the following parameters:

Field Name	Details
`service_type`	The type of service the rider intends to book (standard, minivan or special need taxi). If an app supports multiple service types, the app must accept a Taxi Registry route id as a valid service_type. The expected service_type format will be: ${operator.public_id}-standard-route ${operator.public_id}-minivan-route ${operator.public_id}-special-need-route Note: your public_id will be communicated by the Taxi Registry once the acceptance process has started and will remain the same in the acceptance and the production environnement.
`pickup_latitude`	The latitude for the pickup location. This field should have a precision of 6 decimal places (0.000001). If this field is not populated, the rider's current latitude is taken as the pickup latitude.
`pickup_longitude`	The longitude for the pickup location. This field should have a precision of 6 decimal places (0.000001). If this field is not populated, the rider's current longitude is taken as the pickup longitude.
`dropoff_latitude`	The latitude for the dropoff location. This field should have a precision of 6 decimal places (0.000001). Trip planning applications may leave this field blank and allow riders to input their drop off details later in the booking application, or to coordinate the drop off with the driver.
`dropoff_longitude`	The longitude for the dropoff location. This field should have a precision of 6 decimal places (0.000001). Trip planning applications may leave this field blank and allow riders to input their drop off details later in the booking application, or to coordinate the drop off with the driver.

More parameters are available in the GTFS-OnDemand specification, but are not recommended by the Taxi Registry. The acceptance test will only cover the above parameters.

4.1.2 Services available

The Taxi Registry currently offers 3 types of services that a customer can request:

- **Standard taxi booking**
  for regular users, mostly carried by sedans

- **Minivan booking**
  for group of users or when extra cargo space is required

- **Special need taxi booking**
  for riders with capability issues

The `service_type` parameter described in the section “4.1.1 URL Scheme” is used to differentiate the type of service requested by the user. The service_type parameter must be taken into account if the application allows booking for different types of services.

Some operators may use a distinct app for special need taxi booking. The Taxi Registry offers two options to accommodate existing applications:

1. Use a distinct app for special need taxi booking

2. Use the same app for standard taxi booking, minivan booking and special need taxi booking and use the `service_type` parameter to distinguish between the different service types

4.2 GTFS-OnDemand Acceptance Process

As mentioned previously in the section “4. GTFS-OnDemand”, an acceptance process will be performed between the operators and the Taxi Registry to validate the GTFS-OnDemand URL scheme compliance.

4.2.1 Prerequisites

Before starting the GTFS-OnDemand acceptance process, the operator must be integrated with the Taxi Registry as described in sections “2. Contextual Data” and “3. Taxi positions and status” and the operator must be sending the positions and status of its taxi fleet to the Taxi Registry every 5 seconds.

4.2.2 Communicate your information to the Taxi Registry support team

First, you need to communicate the phone number and endpoints you will be using to receive ride requests. You can do so by sending an email with detailed information to support.taxi.exchange.point@montreal.ca

In this email, you must clearly identify yourself using the same email you are already using when communicating with the Taxi Registry. Here is the extra information you need to communicate to the Taxi Registry (if available):

For standard taxis:

- Phone number
- Website booking URL
- Android booking URL
- Android store URL
- iOS booking URL
- iOS store URL

For minivans:

- Whether a minivan can be booked from the standard taxis website booking URL
- Whether a minivan can be booked from the standard taxis android booking URL
- Whether a minivan can be booked from standard taxis iOS booking URL

For special need taxis (if different than regular taxi endpoint):

- Phone number
- Website booking URL
- Android booking URL
- Android store URL
- iOS booking URL
- iOS store URL

4.2.3 Make the applications and/or website conform to GTFS-OnDemand

Once the information described in the previous section is submitted, you can make the change to your applications and/or website and verify that they conform to the GTFS-OnDemand URL Scheme with the acceptance test. The acceptance test is generated based on the specific information sent by the operator (see previous section) and can be downloaded with the following route:

GET /api/current-user/gtfs-url-scheme-acceptance-test

Response (HTML) status 200

Here is an excerpt of an acceptance test:

GTFS-OnDemand URL scheme Acceptance Test

Example operator

Public ID: 1234

Important: To pass a test, the user should not have to re-enter the information provided through the GTFS-OnDemand URL scheme.

Phone

Standard booking

Can book a standard cab by calling +15145555555.

Minivan booking

Can book a minivan by calling +15145555555.

Special need booking

Can book a cab adapted to riders with capability issues by calling +15145555556.

Web

Standard booking

Can book a standard cab from a Montreal address (80 Queen) to another Montreal address (City Hall) with the website:
https://www.example-operator.com?service_type=1234-standard-route&pickup_latitude=45.497271007&pickup_longitude=-73.554539698&dropoff_latitude=45.50891801&dropoff_longitude=-73.554333425
Can book a standard cab from a Montreal address (80 Queen) to the airport with the website:
https://www.example-operator.com?service_type=1234-standard-route&pickup_latitude=45.497271007&pickup_longitude=-73.554539698&dropoff_latitude=45.465683693&dropoff_longitude=-73.74548144
Can book a standard cab from a Montreal address (80 Queen) to an address in the ARTM zone (Old Longueuil) with the website:
https://www.example-operator.com?service_type=1234-standard-route&pickup_latitude=45.497271007&pickup_longitude=-73.554539698&dropoff_latitude=45.538120632&dropoff_longitude=-73.51005992
Can book a standard cab from a Montreal address (80 Queen) to a Montreal location without an addressv (middle of Angrignon Parc) with the website:
https://www.example-operator.com?service_type=1234-standard-route&pickup_latitude=45.497271007&pickup_longitude=-73.554539698&dropoff_latitude=45.441481488&dropoff_longitude=-73.603012772
Can book a standard cab from a Montreal location without address (middle of Angrignon Parc) to a Montreal address (80 Queen) with the website:
https://www.example-operator.com?service_type=1234-standard-route&pickup_latitude=45.441481488&pickup_longitude=-73.603012772&dropoff_latitude=45.497271007&dropoff_longitude=-73.554539698

Your application must handle all the parameters correctly, meaning the customer should not have to reenter information that was already provided to the trip planning app of the search engine. This information is provided to the applications and/or website of the operator through the GTFS-OnDemand URL Scheme.

Here is an example of a URL based on the GTFS-OnDemand URL Scheme:
https://www.example-operator.com?service_type=1234-special-need-route&pickup_latitude=45.49742&pickup_longitude=-73.55457&dropoff_latitude=45.50868&dropoff_longitude=-73.55374

Using the example above, your application must open directly where you can book a special need taxi from 80 queen street to the city hall. Not handling the parameters properly will result in a failure of the acceptance process.

Moreover, the acceptance test contains the `public_id` of the operator. This information is required if your applications and/or website offer many types of services (e.g. Standard taxi booking and Minivan booking). For more details about the `public_id`, see the `service_type` parameter in section “4.1.1 Deep Link Compliance”.

4.2.4 Final verification

The Taxi Registry support team will verify that your applications and/or website conform to the GTFS-OnDemand URL Scheme using the exact same acceptance test described in the previous section. Thus, the operators **must rigorously verify the compliance to GTFS-OnDemand by themselves** before contacting the Taxi Registry support team for the final verification. Moreover, the operators need to make sure their applications or website are accessibles externally so the Taxi Registry support team can perform the acceptance test on their end.

To initiate the process, you can contact the Taxi Registry support team at support.taxi.exchange.point@montreal.ca.

Upon approval, the Taxi Registry will promote the operator publicly within 24 hours.

5. Tests

This section illustrates tests that operator’s IT staff can perform to verify proper Taxi Registry integration and to ensure that changes, required by Bill 17, are properly implemented.

Steps grouping

Tests steps are grouped, when appropriate, in two sections:

- Initial State.
    - Indicates the situation before tests start. This situation may already exist or can be created in the acceptance environnement by following proposed steps.
    - On test step’s table this section starts with the title: **Initial State**
    - On pictograms this section is delimited by dashed lines.
- Test.
    - Indicates steps to follow in order to conform to Bill 17.
    - On test step’s table this section starts with the title: Test

Pictograms

Pictograms are used to facilitate comprehension by clearly differentiating entities. (Different vehicles, drivers, taxis etc.) Also, at the end of test procedures, they help illustrate, not only the entities created by the test procedure, but also the link between different entities.

- A particular color or an uppercase letter/number combination identifies a single entity.
- Different entities of the same color are not necessarily linked, though they generally are.
- Arrows formally bind different entities.

Icons contain a textual reference, consisting of an uppercase letter: *D* for driver, *V* for vehicle, *P* for permit and *T* for taxi followed by and an index number: Uniquely identifying the entity.

Icon	Entity (generic)	Identification
	Driver	Drive number 1
	Vehicle	Vehicle number 1
	Permit (ADS)	Permit number 1
	Taxi	Taxi number 1

5.1 Contextual Data Tests

5.1.1 The license plate of a vehicle changes

Unlike the driver's license number and the SAAQ file number which are immutable, the license plate can change over time. An owner can change, at will, his vehicle’s license plate.

Step	Action	Request	Response	Icon
Initial State				

1	Create a driver.	POST /api/drivers
		{
		"data": [
		{
		"professional_licence": "L1006-221166-01",
		"departement": {
		"nom": "Québec",
		"numero": "1000"
		},
		...
		}
		]
		}
			HTTP 201 Created	

2	Create a vehicle.	POST /api/vehicles
		{
		"data": [
		{
		"licence_plate": "FAA0011",
		"type_": "sedan",
		"constructor": "audi",
		"model": "a4"
		...
		}
		]
		}
			HTTP 201 Created	

3	Create a permit/ADS.	POST /api/ads
		{
		"data": [
		{
		"insee": "1000",
		"numero": "161000011",
		...
		}
		]
		}
			HTTP 200 OK	

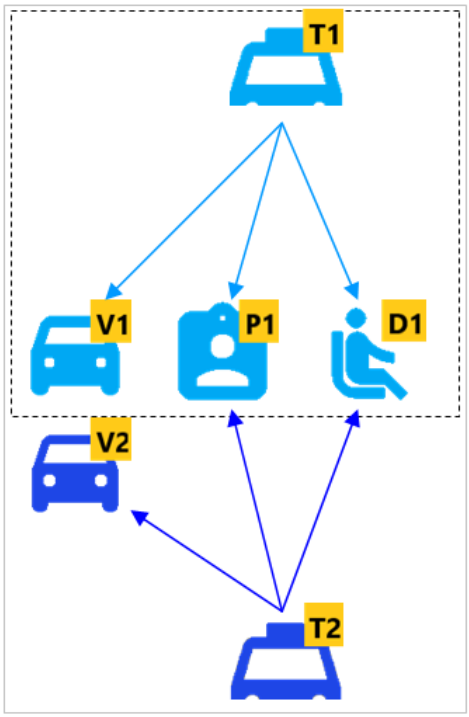
4	Create a taxi using the driver D1, vehicle V1 and permit/ADS P1.	POST /api/taxis
		{
		"data": [
		{
		"ads": {
		"insee": "1000",
		"numero": "161000011"
		},
		"vehicle": {
		"licence_plate": "FAA0011"
		},
		"driver": {
		"departement": "1000",
		"professional_licence": "L1006-221166-01"
		},
		…
		}
		]
		}
			HTTP 201 Created
			{
			"data": [
			{
			"id": “$T1”,
			...
			}
			]
			}
				

Test				

5	Create a vehicle identical to V1 but with a different licence plate.	POST /api/vehicles
		{
		"data": [
		{
		"licence_plate": "FBB0022",
		"type_": "sedan",
		"constructor": "audi",
		"model": "a4"
		...
		}
		]
		}
			HTTP 201 Created	

6	Create a taxi for vehicle with the new license plate	POST /api/taxis
		{
		"data": [
		{
		"ads": {
		"insee": "1000",
		"numero": "161000011"
		},
		"vehicle": {
		"licence_plate": "FBB0022"
		},
		"driver": {
		"departement": "1000",
		"professional_licence": "L1006-221166-01"
		},
		…
		}
		]
		}
			HTTP 201 Created
			{
			"data": [
			{
			"id": “$T2”,
			…
			}
			]
			}
				

7	Start sending taxi positions and status with the taxi created (T2).	POST /api/taxi-position-snapshots
		{
		"items": [
		{
		"taxi": "$T2",
		…
		}
		]
		}
			HTTP 200 OK	



Entities present in the system after the license plate renewal

5.2 Bill 17 Tests

This section is intended for the operator's IT staff responsible for implementing the changes, required by Bill 17, in the operator's IT system. This section presents all the information and the links required to implement these changes.

To understand how these changes impact the operator at the business level, see the document Guide d'accompagnement loi 17.

For information on how to read the tests presented in this section, see *Steps grouping* and *Pictograms* in section 5.

The operator's IT staff is responsible for verifying that the changes required by Bill 17 are correctly implemented in the operator's IT system.

The operator's IT staff is responsible for deciding when to deploy these changes to the production environment.To ensure proper migration, tests can be performed, at will, in the acceptance environment at: https://taximtl.accept.ville.montreal.qc.ca.

BTM's staff remains available for answering any related questions at: support.taxi.exchange.point@montreal.ca.

If needed, the operator's IT staff can contact the BTM to check that the Bill 17 changes are properly implemented by verifying the test results in the acceptance environment.

When available, vehicle_identification_number must be transmitted

See section 2.2 for more information.

Obligation to continue transmitting vehicle's positions

In accordance with the Act regulating the remunerated transport of persons by automobile, vehicles must remain connected to the Taxi register of the Bureau du taxi de Montréal (BTM). During the transition period (October 10, 2020 to March 31, 2021), positions of non-migrated vehicles **must continue** to be transmitted as long as the position's transmission of migrated vehicles is not active.

5.2.1 Migrate a driver when the vehicles he drives have not been migrated yet

This migration can be performed as soon as possible. There are no prerequisites.

Step	Action	Request	Response	Icon
Initial State				

1	Create a driver with Montréal-600 department.	POST /api/drivers
		{
		"data": [
		{
		"first_name": "John",
		"last_name": "Doe",
		"professional_licence": "00011",
		"departement": {
		"nom": "Montréal",
		"numero": "660"
		}
		}
		]
		}
			HTTP 201 Created	

2	Create a vehicle with a ‘T’ licence plate.	POST /api/vehicles
		{
		"data": [
		{
		"licence_plate": "T00011A",
		...
		}
		]
		}
			HTTP 201 Created	

3	Create a permit (ADS)	POST /api/ads
		{
		"data": [
		{
		"insee": "102005",
		"numero": "4M000000011A",
		"vdm_vignette": "5511",
		...
		}
		]
		}
			HTTP 201 Created	

4	Create the first taxi driven by D1 and link it to V1 and P1.	POST /api/taxis
		{
		"data": [
		{
		"ads": {
		"insee": "102005",
		"numero": "4M000000011A"
		},
		"vehicle": {
		"licence_plate": "T00011A"
		},
		"driver": {
		"departement": "660",
		"professional_licence": "00011"
		},
		…
		}
		]
		}
			HTTP 201 Created	

5	Create a second vehicle with ‘T’ licence plate.	POST /api/vehicles
		{
		"data": [
		{
		"licence_plate": "T00012B",
		...
		}
		]
		}
			HTTP 201 Created	

6	Create a second permit (ADS)	POST /api/ads
		{
		"data": [
		{
		"insee": "102005",
		"numero": "4M000000012B",
		"vdm_vignette": "5512",
		...
		}
		]
		}
			HTTP 200 OK	

7	Create the second taxi driven by D1 and link it to V2 and P2.	POST /api/taxis
		{
		"data": [
		{
		"ads": {
		"insee": "102005",
		"numero": "4M000000012B"
		},
		"vehicle": {
		"licence_plate": "T00012B"
		},
		"driver": {
		"departement": "660",
		"professional_licence": "00011"
		},
		…
		}
		]
		}
			HTTP 201 Created	

Test				

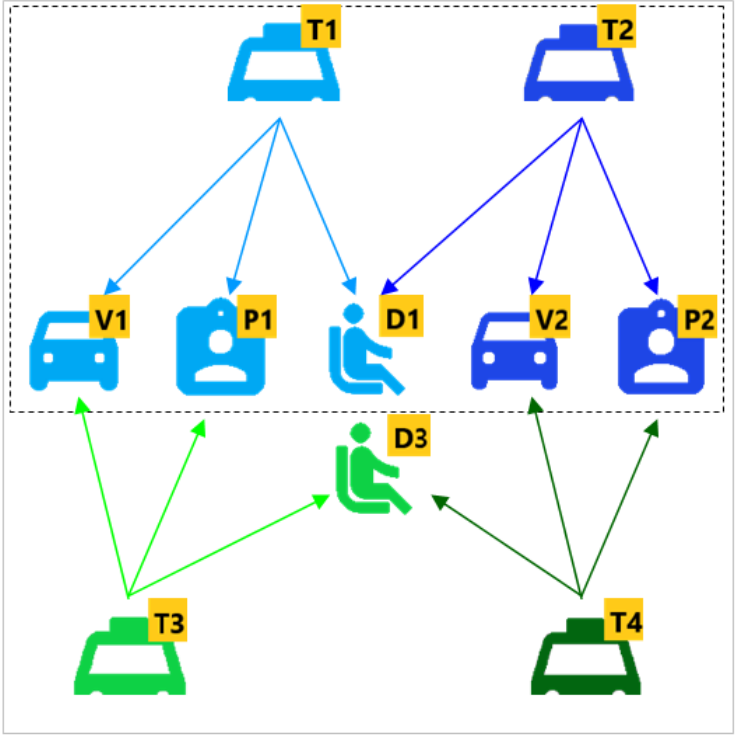
8	In accordance with Bill 17, create a new driver using his driving license number and Québec-1000 department.
	
	For more details on how to submit a driver in accordance with Bill 17, see the description of department and professional licence in section 2.1.
		POST /api/drivers
		{
		"data": [
		{
		"departement": {
		"nom": "Québec",
		"numero": "1000"
		},
		"first_name": "John",
		"last_name": "Doe",
		"professional_licence": "L0006-221166-01"
		}
		]
		}
			HTTP 201 Created	

9	Create a taxi using the driver in accordance with Bill 17 (D3) and the first vehicle that has not been migrated yet (V2 with P2).	POST /api/taxis
		{
		"data": [
		{
		"ads": {
		"insee": "102005",
		"numero": "4M000000011A"
		},
		"vehicle": {
		"licence_plate": "T00011A"
		},
		"driver": {
		"departement": "1000",
		"professional_licence": "L0006-221166-01"
		},
		…
		}
		]
		}
			HTTP 201 Created
			{
			"data": [
			{
			"id": “$T3”,
			…
			}
			]
			}
				

10	Start sending taxi positions and status with the taxi created (T3).	POST /api/taxi-position-snapshots
		{
		"items": [
		{
		"taxi": "$T3",
		…
		}
		]
		}
			HTTP 200 OK	

11	**A driver may drive many vehicles. Make sure to create a new taxi for each vehicle driven by the driver.**
	
	Create a second taxi using the driver in accordance with Bill 17 (D3) and the second vehicle that have not been migrated yet (V2 with P2)
		POST /api/taxis
		{
		"data": [
		{
		"ads": {
		"insee": "102005",
		"numero": "4M000000012B"
		},
		"vehicle": {
		"licence_plate": "T00012B"
		},
		"driver": {
		"departement": "1000",
		"professional_licence": "L0006-221166-01"
		},
		…
		}
		]
		}
			HTTP 201 Created
			{
			"data": [
			{
			"id": “$T4”,
			…
			}
			]
			}
				

12	Start sending taxi positions and status with the taxi created (T4).	POST /api/taxi-position-snapshots
		{
		"items": [
		{
		"taxi": "$T4",
		…
		}
		]
		}
			HTTP 200 OK	



Entities present in the system after the migration

5.2.2 Migrate a vehicle when the drivers who drive it have been migrated

This migration can be performed when an owner transmits the new license plate in conformity with Bill 17 (No *T* license plate).

Before performing this migration, the migration 5.3.1 must have been performed for all the drivers who drive the vehicle.

Note that an owner can change the license plate for reasons unrelated to Bill 17. See section 5.1.1 for more information.

Step	Action	Request	Response	Icon
Initial State				

1	To emulate the results of the migration described section 5.3.1, create a new driver using his driving license number and Québec-1000 department.
	
	For more details on how to submit a driver in accordance with Bill 17, see the description of department and professional licence in section 2.1.
		POST /api/drivers
		{
		"data": [
		{
		"departement": {
		"nom": "Québec",
		"numero": "1000"
		},
		"first_name": "John",
		"last_name": "Doe",
		"professional_licence": "L1006-221166-11"
		}
		]
		}
			HTTP 201 Created	

2	Create a vehicle with a ‘T’ licence plate.	POST /api/vehicles
		{
		"data": [
		{
		"licence_plate": "T00011A",
		"type_": "sedan",
		"constructor": "audi",
		"model": "a4"
		...
		}
		]
		}
			HTTP 201 Created	

3	Create a permit (ADS)	POST /api/ads
		{
		"data": [
		{
		"insee": "102005",
		"numero": "4M000000011A",
		"vdm_vignette": "5511",
		"owner_name": "Taxi-Pro",
		...
		}
		]
		}
			HTTP 201 Created	

4	Create the first taxi driven by D1 and link it to V1 and P1.	POST /api/taxis
		{
		"data": [
		{
		"ads": {
		"insee": "102005",
		"numero": "4M000000011A"
		},
		"vehicle": {
		"licence_plate": "T00011A"
		},
		"driver": {
		"departement": "1000",
		"professional_licence": "L1006-221166-11"
		},
		…
		}
		]
		}
			HTTP 201 Created	

5	To emulate the results of the migration described section 5.3.1, create a second driver using his driving license number and Québec-1000 department.
	
	For more details on how to submit a driver in accordance with Bill 17, see the description of department and professional licence in section 2.1
		POST /api/drivers
		{
		"data": [
		{
		"departement": {
		"nom": "Québec",
		"numero": "1000"
		},
		"first_name": "Jane",
		"last_name": "Din",
		"professional_licence": "L2006-221166-22"
		}
		]
		}
			HTTP 201 Created	

6	Create the second taxi driven by D2 and link it to V1 and P1.	POST /api/taxis
		{
		"data": [
		{
		"ads": {
		"insee": "102005",
		"numero": "4M000000011A"
		},
		"vehicle": {
		"licence_plate": "T00011A"
		},
		"driver": {
		"departement": "1000",
		"professional_licence": "L2006-221166-22"
		},
		…
		}
		]
		}
			HTTP 201 Created	

Test				

7	In accordance with Bill 17, create a vehicle identical to V1 but with the new licence plate.
	
	For more details on how to submit a vehicle in accordance with Bill 17, see the description of licence_plate in section 2.2
		POST /api/vehicles
		{
		"data": [
		{
		"licence_plate": "FAA0012",
		"type_": "sedan",
		"constructor": "audi",
		"model": "a4"
		...
		}
		]
		}
			HTTP 201 Created	

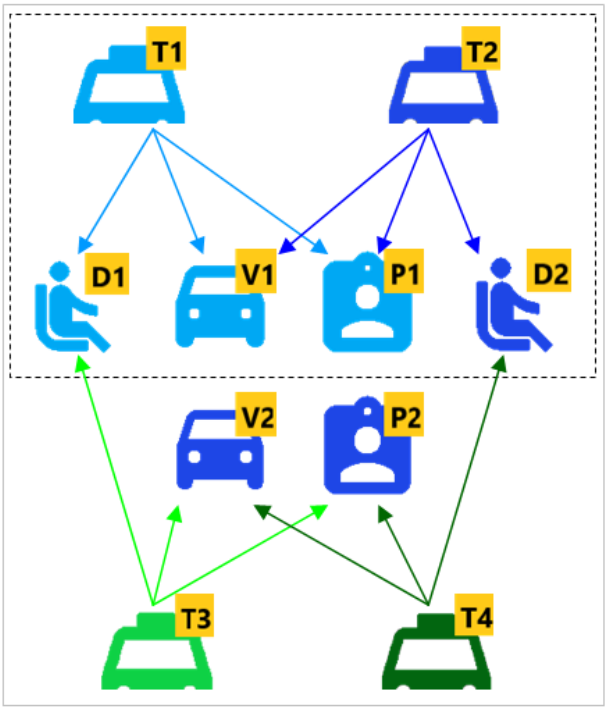
8	In accordance with Bill 17, create a permit/ADS identical to P1 but with zone Québec-1000 and SAAQ file number.
	
	For more details on how to submit a permit/ADS in accordance with Bill 17, see the description of insee and numero in section 2.3
		POST /api/ads
		{
		"data": [
		{
		"insee": "1000",
		"numero": "161000012",
		"owner_name": "Taxi-Pro",
		...
		}
		]
		}
			HTTP 200 OK	

9	Create a taxi using driver D1, vehicle V2 and permit/ADS P2.	POST /api/taxis
		{
		"data": [
		{
		"ads": {
		"insee": "1000",
		"numero": "161000012"
		},
		"vehicle": {
		"licence_plate": "FAA0012"
		},
		"driver": {
		"departement": "1000",
		"professional_licence": "L1006-221166-11"
		},
		…
		}
		]
		}
			HTTP 201 Created
			{
			"data": [
			{
			"id": “$T3”,
			…
			}
			]
			}
				

10	Start sending taxi positions and status with the taxi created (T3).	POST /api/taxi-position-snapshots
		{
		"items": [
		{
		"taxi": "$T3",
		…
		}
		]
		}
			HTTP 200 OK	

11	Create a taxi using driver D2, vehicle V2 and permit/ADS P2.	POST /api/taxis
		{
		"data": [
		{
		"ads": {
		"insee": "1000",
		"numero": "161000012"
		},
		"vehicle": {
		"licence_plate": "FAA0012"
		},
		"driver": {
		"departement": "1000",
		"professional_licence": "L2006-221166-22"
		},
		…
		}
		]
		}
			HTTP 201 Created
			{
			"data": [
			{
			"id": “$T4”,
			…
			}
			]
			}
				

12	Start sending taxi positions and status with the taxi created (T4).	POST /api/taxi-position-snapshots
		{
		"items": [
		{
		"taxi": "$T4",
		…
		}
		]
		}
			HTTP 200 OK	



Entities present in the system after the migration

5.2.3 Migrate a vehicle and the drivers who drive it at the same time

This migration can be performed as soon as an owner transmits the new license plate in conformity with Bill 17 (No *T* license plate).

This scenario is an alternative to scenarios described in sections 5.3.1 and 5.3.2. If this scenario does not simplify the changes required by Bill 17 in the operator's IT system, then just ignore it and use the two-step migration as described in sections 5.3.1 and 5.3.2 instead.

This scenario is more complex, because during the transition period, a driver can drive the migrated vehicle and the non-migrated vehicle. For this scenario to succeed, the operator's IT system must be able to continue to identify the driver by the pocket number when he is driving the non-migrated vehicle and identify the same driver by his driving license number when he drives the migrated vehicle.

Note that in order to keep this scenario simple, it presents the case where the migrated vehicle is driven by a single driver. **However, the operator's IT system must also support the scenario where the migrated vehicle is driven by multiple drivers.**

Step	Action	Request	Response	Icon
Initial State				

1	Create a driver with Montréal-600 department.	POST /api/drivers
		{
		"data": [
		{
		"first_name": "John",
		"last_name": "Doe",
		"professional_licence": "00011",
		"departement": {
		"nom": "Montréal",
		"numero": "660"
		}
		}
		]
		}
			HTTP 201 Created	

2	Create a vehicle with a ‘T’ licence plate.	POST /api/vehicles
		{
		"data": [
		{
		"licence_plate": "T00011A",
		"type_": "sedan",
		"constructor": "audi",
		"model": "a4"
		...
		}
		]
		}
			HTTP 201 Created	

3	Create a permit (ADS)	POST /api/ads
		{
		"data": [
		{
		"insee": "102005",
		"numero": "4M000000011A",
		"vdm_vignette": "5511",
		...
		}
		]
		}
			HTTP 201 Created	

4	Create the first taxi driven by D1 and link it to V1 and P1.
	
	This taxi will remain unmigrated at the end of this test.
		POST /api/taxis
		{
		"data": [
		{
		"ads": {
		"insee": "102005",
		"numero": "4M000000011A"
		},
		"vehicle": {
		"licence_plate": "T00011A"
		},
		"driver": {
		"departement": "660",
		"professional_licence": "00011"
		},
		…
		}
		]
		}
			HTTP 201 Created
			{
			"data": [
			{
			"id": “$T1”,
			...
			}
			]
			}
				

5	Start sending taxi positions and status with the taxi that will not be migrated. (T1)	POST /api/taxi-position-snapshots
		{
		"items": [
		{
		"taxi": "$T1",
		...
		}
		]
		}
			HTTP 200 OK	

6	Create a second vehicle with a 'T' licence plate.	POST /api/vehicles
		{
		"data": [
		{
		"licence_plate": "T00022B",
		"type_": "mpv",
		"constructor": "toyota",
		"model": "camry"
		...
		}
		]
		}
			HTTP 201 Created	

7	Create a second permit (ADS)	POST /api/ads
		{
		"data": [
		{
		"insee": "102005",
		"numero": "4M000000022B",
		"vdm_vignette": "5522",
		...
		}
		]
		}
			HTTP 201 Created	

8	Create a second taxi driven by driver D1 and link it to V2 and P2.
	
	This taxi will be migrated in the Test section.
		POST /api/taxis
		{
		"data": [
		{
		"ads": {
		"insee": "102005",
		"numero": "4M000000022B"
		},
		"vehicle": {
		"licence_plate": "T00022B"
		},
		"driver": {
		"departement": "660",
		"professional_licence": "00011"
		},
		…
		}
		]
		}
			HTTP 201 Created
			{
			"data": [
			{
			"id": "$T2",
			...
			}
			]
			}
				

Test				

9	In accordance with Bill 17, create a driver, identical to D1 but using his driving license number and Québec-1000 department.
	
	For more details on how to submit a driver in accordance with Bill 17, see the description of department and professional licence in section 2.1
		POST /api/drivers
		{
		"data": [
		{
		"departement": {
		"nom": "Québec",
		"numero": "1000"
		},
		"first_name": "John",
		"last_name": "Doe",
		"professional_licence": "L3006-221166-33"
		}
		]
		}
			HTTP 201 Created	

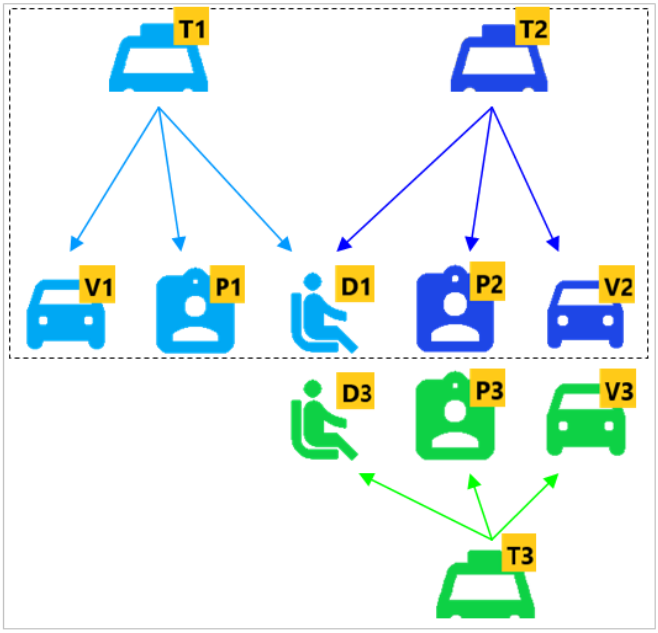
10	In accordance with Bill 17, create a vehicle, identical to V2 but using a license plate without T prefix.	POST /api/vehicles
		{
		"data": [
		{
		"licence_plate": "FCC0013",
		"type_": "mpv",
		"constructor": "toyota",
		"model": "camry"
		...
		}
		]
		}
			HTTP 201 Created	

11	In accordance with Bill 17, create a permit/ADS identical to P2 but with zone Québec-1000 and SAAQ file number.
	
	For more details on how to submit a permit/ADS in accordance with Bill 17, see the description of insee and numero in section 2.3
		POST /api/ads
		{
		"data": [
		{
		"insee": "1000",
		"numero": "163000013",
		"owner_name": "Taxi-Pro",
		...
		}
		]
		}
			HTTP 200 OK	

12	Create a taxi driven by driver D3 and link it to V3 and P3.	POST /api/taxis
		{
		"data": [
		{
		"ads": {
		"insee": "1000",
		"numero": "163000013"
		},
		"vehicle": {
		"licence_plate": "T00013C"
		},
		"driver": {
		"departement": "1000",
		"professional_licence": "L3006-221166-33"
		},
		…
		}
		]
		}
			HTTP 201 Created
			{
			"data": [
			{
			"id": “$T3”,
			...
			}
			]
			}
				

13	Start sending taxi positions and status with the taxi created (T3).	POST /api/taxi-position-snapshots
		{
		"items": [
		{
		"taxi": "$T3",
		...
		}
		]
		}
			HTTP 200 OK	

14	Taxi T1 continues to send positions and status. T1 remains unmigrated.
	
	At this point John Doe drives a migrated vehicle (V3) as driver D3 and an unmigrated vehicle (V1) as driver D1.
		POST /api/taxi-position-snapshots
		{
		"items": [
		{
		"taxi": "$T1",
		...
		}
		]
		}
			HTTP 200 OK	



Entities present in the system after the migration

5.2.4 Unallowed migration paths

Owners will not all regularize their situation with the SAAQ at the same time. During the transition period, some vehicles will be migrated and others will not. However, when a taxi is linked to an owner (ADS) in the Québec-1000 zone, that taxi must be fully migrated. (Driver, vehicle and owner/license/ADS)

1. It is not possible to migrate an owner (ADS) without migrating the drivers that drive the vehicle belonging to that owner.

   As soon as possible, drivers must send their driver's license number to the operator.

If the owner (ADS) is in the Québec-1000 zone, then a linked driver must be in the Québec-1000 department; otherwise a http 400 error will occur.

2. It is not possible to migrate the owner without migrating the vehicle.

   Following the license plate change, the owner must communicate his SAAQ file number and his new license plate number to the operator.

   If the owner is in the Québec-1000 zone, then the license plate must not start with a T; otherwise a http 400 error will occur.

5.2.5 Many vehicles can have the same owner

As described in section 2.3, following adoption of Bill 17, the meaning of ADS has changed from permit to owner's license. This example illustrates this change. Please make sure this change is well supported by the operator's IT system.

To simplify, all vehicles will be driven by the same driver.

Test				

1	Create a driver in the Québec-1000 department.	POST /api/drivers
		{
		"data": [
		{
		"departement": {
		"nom": "Québec",
		"numero": "1000"
		},
		"first_name": "John",
		"last_name": "Doe",
		"professional_licence": "L1006-221166-11"
		}
		]
		}
			HTTP 201 Created	

2	Create a vehicle.	POST /api/vehicles
		{
		"data": [
		{
		"licence_plate": "FAA0011",
		...
		}
		]
		}
			HTTP 201 Created	

3	Create a second vehicle.	POST /api/vehicles
		{
		"data": [
		{
		"licence_plate": "FBB0022",
		...
		}
		]
		}
			HTTP 201 Created	

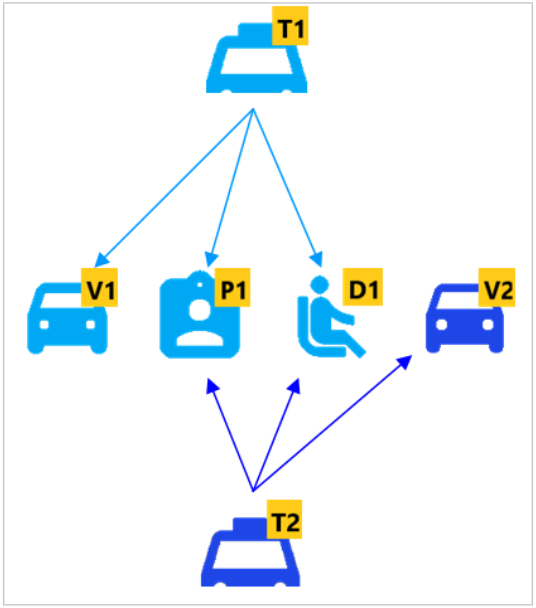
4	Create a permit/ADS in the zone Québec-1000 and SAAQ file number.
	
	For more details on how to submit an ADS in accordance with Bill 17, see the description of insee and numero in section 2.3
		POST /api/ads
		{
		"data": [
		{
		"insee": "1000",
		"numero": "161000011",
		"owner_name": "Taxi-Pro",
		...
		}
		]
		}
			HTTP 200 OK	

5	Create a taxi driven by driver D1 and link it to V1 and P1.	POST /api/taxis
		{
		"data": [
		{
		"ads": {
		"insee": "1000",
		"numero": "161000011"
		},
		"vehicle": {
		"licence_plate": "FAA0011"
		},
		"driver": {
		"departement": "1000",
		"professional_licence": "L1006-221166-11"
		},
		…
		}
		]
		}
			HTTP 201 Created
			{
			"data": [
			{
			"id": “$T1”,
			...
			}
			]
			}
				

6	Create a second taxi driven by driver D1 and link it to V2 and P1.	POST /api/taxis
		{
		"data": [
		{
		"ads": {
		"insee": "1000",
		"numero": "161000011"
		},
		"vehicle": {
		"licence_plate": "FBB0022"
		},
		"driver": {
		"departement": "1000",
		"professional_licence": "L1006-221166-11"
		},
		…
		}
		]
		}
			HTTP 201 Created
			{
			"data": [
			{
			"id": “$T2”,
			...
			}
			]
			}
				

7	Start sending taxi positions and status with the taxi T1.	POST /api/taxi-position-snapshots
		{
		"items": [
		{
		"taxi": "$T1",
		...
		}
		]
		}
			HTTP 200 OK	

8	Start sending taxi positions and status with the taxi T2.	POST /api/taxi-position-snapshots
		{
		"items": [
		{
		"taxi": "$T2",
		...
		}
		]
		}
			HTTP 200 OK	



Entities present in the system after the test