Montréal

Extraction

The Taxi Exchange Point Data Extraction Guide

Version	Description	Author	Date
1.0	Original Draft	David Beaudoin	28/08/2017

Montréal

Extraction

TABLE OF CONTENTS

Montréal

Extraction

1. INTRODUCTION

1.1 Data extraction

As an open platform the Taxi Exchange Point (TXP) expose some of it's data for selected organisms. The extraction is used to get all the data about the ADS("Authorisation De Stationnement" / "Permis"), Drivers, Vehicles, Taxis and Taxis Positions. The data is sent in JSON format.

To get help using the TXP platform please contact the administrators at this address: support.taxi.exchange.point@ville.montreal.qc.ca.

The service must be call with a header containing a api-key given by the BTM (Bureau Taxi Montréal).

Here are the links to communicate with the TXP services:
Acceptation : https://taximtl.accept.ville.montreal.qc.ca
Production : https://taximtl.ville.montreal.qc.ca

```
Header:

Accept: application/json

X-VERSION: 2

Content-Type: application/json

X-API-KEY: XXXXXXXXXXXXXXXXXXXXXXXXXXXX

etag: XXXXXXXXXXXXXXX (optional)
```

1.2 Etag

Most of the endpoint uses "etag" so that you can fetch the data if there's been changes. The first time you call the service you will receive a "etag" parameter in the response header. You must keep a copy of that "etag" and use it the next time you call the service by adding it to the header of your request. If you receive an HTTP 304-NotModified answer, you need to keep the current "etag" for the next call. If you receive an HTTP 200-OK answer, you need to keep the new "etag" received in the response header.

2. Services

2.1 ADS

```
GET /api/data-dumps/ads

Headers

Etag : Used to fetch only new element (received after the first call)

Response (JSON)

 [
    {
        "id": 1800,
        "numero": "523399",
        "doublage": false,
        "added_at": "2017-09-06T11:01:50Z",
        "added_by": 4,
        "added_via": "api",
        "last_update_at": null,
        "source": "added_by",
        "insee": "102011",
        "vehicle_id": 3241,
        "category": "",
        "owner_name": "test",
        "owner_type": "company",
        "zupc_id": 102011,
        "vdm_vignette": "2818",
        "nom_zupc": "A.11",
        "added_by_name": "coop"
    }
]
```

Key	Value Type	Description

Montréal

Extraction

insee	string	*Identifier of the local authority who attributed the license (ADS).* For *Quebec, similarly to the zupc_id, this field represents the six digits number assigned by the CTQ to the agglomeration that a taxi is allowed to operate.* ***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	*This is the taxi license number (ADS number).* For ***Quebec: number represents the taxi license number assigned by the CTQ (registration certificate). Proposed structure: (12 alphanumeric characters).***
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" and "individual".*
category	string	*This field is used for administrative purpose.* *When a new license (aka ADS) is created by an Operator, an empty string has to be passed (not a "null" value).*
doublage	boolean	*When a new license (aka ADS) is created by an Operator, this field should always be set to "false"*
vehicle_id	integer	*This field is used for administrative purpose.* *When a new license (aka ADS) is created by an Operator, a "0" or "null" has to be passed.*

vdm_vignette	string	***This field represent the "Vignette" number given by the BTM(Bureau Taxi Montreal). Mandatory.***
nom_zupc	string	*The name of the zone associated to the licence (insee).*
added_by_name	string	*Name of the person who added the licence.*

2.2 Vehicles

```
GET /api/data-dumps/vehicles

Headers

Etag : Used to fetch only new element (received after the first call)

Response (JSON)

[

{

        "licence_plate": "123",

        "added_at": "2017-09-06T11:01:50Z",

        "added_via": "api",

        "source": "added_by",

        "last_update_at": "2017-09-06T11:01:50Z",

        "id": 4029,

        "model_id": 20,

        "constructor_id": 4,

        "model_year": 2011,

        "engine": null,

        "horse_power": 0,

        "relais": false,

        "horodateur": null,

        "taximetre": null,

        "date_dernier_ct": null,

        "date_validite_ct": null,

        "special_need_vehicle": false,
```

```
        "type_": "mpv",
        "luxury": true,
        "credit_card_accepted": true,
        "nfc_cc_accepted": false,
        "amex_accepted": false,
        "bank_check_accepted": false,
        "fresh_drink": false,
        "dvd_player": false,
        "tablet": false,
        "wifi": false,
        "baby_seat": false,
        "bike_accepted": true,
        "pet_accepted": true,
        "air_con": false,
        "electronic_toll": false,
        "gps": true,
        "cpam_conventionne": false,
        "every_destination": true,
        "color": "GRISE",
        "vehicle_id": 4041,
        "added_by": 4,
        "status": "free",
        "nb_seats": 6,
        "private": false,
        "last_nonStatus_update_at": null,
        "modelname": "SIENNA",
        "constructorname": "TOYOTA",
        "added_by_name": "coop"
    }
]
```


Key	Value Type	Description

`licence_plate`	string	*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).* *The `licence_plate` is used as the vehicle identifier to declare a taxi as a vehicle/driver/license triplet.* *For **Quebec, the licence plate is an alphanumeric combination of 6 characters.***
`constructor`	string	*Constructor of the vehicle.*
`model`	string	*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	
`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 vehicle 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 clients 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 clients 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 clients 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).* *Vehicle constructed or converted specifically so that they accommodate one or more persons seated in their wheelchairs when travelling on the road.*

`tablet`	boolean	*This vehicle has a digital tablet at the disposal of the clients during the ride.*
`wifi`	boolean	*This vehicle has complimentary WiFi aboard.*
`cpam_conventionne`	boolean	*This vehicle has a convention with social security to transport patients.* *This field is used for administrative purpose 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 purpose 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 purpose 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 purpose 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 purpose 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 purpose 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 purpose 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 purpose 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 purpose 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 purpose 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 engine to store the value returned by the TXP: the field used to uniquely identify vehicles in all transactions with the TXP is the licence_plate.*
added_by_name	string	*Name of the person who added the licence.*

2.3 Drivers

```
GET /api/data-dumps/drivers

Headers

Etag : Used to fetch only new element (received after the first call)

Response (JSON)

[
   {
        "added_at": "2017-09-06T11:01:50Z",
        "added_via": "api",
        "source": "added_by",
        "last_update_at": null,
        "id": 1246,
        "departement_id": 660,
        "added_by": 4,
        "birth_date": null,
        "first_name": "David",
        "last_name": "Gratton",
        "professional_licence": "679638",
        "added_by_name": "coop"
    }
]
```

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 TXP.* ***For Quebec, departement should always be set to 660-Montréal as shown in the above example.***
`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, the professional_licence is the 'pocket number'. Proposed structure: XXXXX (A five digit code number).***
`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.
`added_by_name`	string	*Name of the person who added the licence.*

2.4 Taxis

```
GET /api/data-dumps/taxis

Headers

Etag : Used to fetch only new element (received after the first call)

Response (JSON)

[
     {
         "added_at": "2017-09-06T11:01:50Z",
         "added_via": "api",
         "source": "added_by",
         "last_update_at": "2017-09-06T11:01:50Z",
         "id": "22XDUw9",
         "vehicle_id": 6846,
         "ads_id": 5342,
         "added_by": 4,
         "driver_id": 5962,
         "rating": 4.5,
         "current_hail_id": null,
         "ban_begin": null,
         "ban_end": null,
         "added_by_name": "coop"
    }
]
```

Key	Value Type	Description
`vehicle_id`	integer	*The id representing the vehicle.*
`ads_id`	integer	*The id representing the ADS (licence).*
`driver_id`	integer	*The id representing the driver.*
`id`	string	*A long-lived identifier generated for this `vehicle`/`ads`/`driver` triplet by the TXP.* *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.*
`current_hail_id`	integer	*The id of the hail currently associated with this taxi. ID is null if no current hail.*
`ban_begin`	date	*The date at which the ban begin.*
`ban_end`	date	*The date at which the ban end.*
`rating`	float	*The mean of the ratings of last rides of the taxi.* *It is calculated by the TXP and falls between 0 and 5.*

2.5 Taxis Positions

For the positions, it is not permitted to obtain a complete list of all the positions since the data would be too big. The date parameter is mandatory and is only working in slice of 10 minutes, if the date's minutes aren't a multiple of ten it will return a HTTP 404 error. The service will return all the data with a creation/modification date greater or equal to the given date parameter.

```
GET /api/data-dumps/taxi-positions/:date

Parameters

date (format: YYYY-MM-DDThh:mm:ss.nnnZ)

Response (JSON)

{

     "id": "2017-07-20T20:00:00Z",
```

Montréal

Extraction

```
"items":
   [
      {
       "id": "348593485934085",
       "items": [
       {
         "taxi": "AXbNHEp",
         "lat": "45.44066",
         "lon": "-73.73685",
         "version": "2",
         "device": "phone",
         "operator": "coop",
         "timestamp": "1500580203",
         "status": "unavailable",
         "speed": "50",
         "azimuth": "234",
         "timestampUTC": "2017-07-20T19:50:03.000Z"
       },
     …
     "receivedAt" : ISODate("2017-07-17T13:29:35.053Z")
    }
   ]
}
```

Key	Value Type	Description
`taxi`	string	*This is the taxi id.*
`lat`	string	*Represent the latitude for the taxi at this moment.*
`lon`	string	*Represent the longitude for the taxi at this moment.*
`version`	string	*Version of the TXP api.*
`device`	string	*Device from which the position is taken.*
`operator`	string	*The name of the operator of this taxi.*
`timestamp`	string	*The timestamp when the position was sent.*
`status`	string	*Status of the taxi at this moment.*
`speed`	string	*Current speed of the taxi(km/h)*
`azimuth`	string	*Current orientation of the taxi.*
`timestampUTC`	string	*The timestamp in UTC format.*