API

Documentation

DRUST

Introduction

DRUST provides a RESTful API that includes a collection of endpoints as presented in this document. The available data contains:

- Detailed or aggregated information on trips
- Information data about vehicles
- Driving event data

In what follows, several guides present a set of common operation examples and a “reference” section exposes the complete set of data treated by the API and the list of allowed operations.

General description

This API is based on the HTTPS protocol. Consequently, it is compliant with the REST style and the JSON format.

After an authentication step (detailed hereunder), requests are executed depending on the selected operation. Each operation returns a collection of objects. Each collection is paginated and includes only a subset of data. Each page contains a pointer to the next page. The page size is configurable.

Authentication

The access to the resources served by the API is protected via the authorization protocol OAuth 2.0.

This protocol is standardized:

- https://tools.ietf.org/html/rfc6749
- https://tools.ietf.org/html/rfc6750

It provides an “access token” to the users (the “client”): a secured mean for them to access the API.

The OAuth 2.0 access type supported by the API is “Resource Owner Password Credentials”.

Here is the process to follow:

- First, the user needs to have the following information available:
    - A user account email.
    - The associated password.
    - A client identifier provided by DRUST.
    - A client secret, also provided by DRUST, that the user must keep secured.
- Initial access token request:
    - With the previous 4 pieces of information, the user can run the HTTP request.
    - The API returns, among other data, an access token and a refresh token. The access token allows the user to identify, for each API call, for a limited duration. The refresh token allows the user to get a new access token when the previous one is expired.
- API consumption:
    - API calls must include the access token in their HTTP header.
    - The API can return with an error due to an expired access token. In this case, the user needs to refresh it or go through the initial request again.
- Access token refresh process:
    - An HTTP request containing nothing but the refresh token (no identifier) can be executed.
    - The API will then return the new access and refresh tokens.

Guides

The following guides present some examples of common use cases of DRUST API. They are based on using the command line tool *curl* (https://curl.haxx.se/) but are easily adaptable to another API client tool.

Retrieve an access token

The initial token is retrieved via a request containing the password:

Request example

```
curl --request POST \
  --url "https://public.api.prd.drust.io/oauth/token" \
  --header "content-type: application/x-www-form-urlencoded" \
  --data "grant_type=password&scope=global\
    &username=myemail@example.com\
    &password=myPassw0rd\
    &client_id=myclientid\
    &client_secret=myclientsecret"
```

Response example

```
{
  "access_token": "ed65fca2-7f5c-41a5-9304-ca4b2e26e789",
  "refresh_token": "dc81f891-6757-405c-9dac-b7437b04d9d4",
  "token_type": "bearer",
  "expires in": 899,
  "scope": "global"
}
```

The string ed65fca2-7f5c-41a5-9304-ca4b2e26e789 represents the access token that can be used to authenticate other API calls.

Refresh an access token

Refreshing an access token is done by sending the refresh token received in the initial access token request.

Request example

```
curl --request POST \
      --url "https://public.api.prd.drust.io/oauth/token" \
      --header "content-type: application/x-www-form-urlencoded" \
      --data "grant type=refresh_token&scope=global\
       &refresh_token=dc81f891-6757-405c-9dac-b7437b04d9d4\
       &client_id=myclientid\
       &client_secret=myclientsecret"
```

Response example

```
{
  "access_token": "f67b4a05-0f5c-ffa5-4200-c0f6b2e26e456",
  "refresh_token": "57db5679-4ba9-42ee-9363-e8ea2f636516",
  "token_type": "bearer",
  "expires_in": 899,
  "scope": "global"
}
```

A new access token is then retrieved. For security reasons the password is not needed in this request and the token can easily be revoked in case of data corruption.

Listing raw data of a trip, for a specific telematic device

Hereunder is shown a scenario allowing to:

- list a limited number of trips, for a specific telematic device plugged in a vehicle
- retrieve the raw data, called timeseries, for the second trip of this list

Trip request example

```
curl --request GET \
  --url
"https://public.api.prd.drust.io/v2/devices/10659000-b6a7-aa7a-3611-1431
a2b99700/trips\
    ?limit=2\
    &start_date=1450297562000\
    &end_date=1461167669000" \
  --header "Authorization: \
Bearer ed65fca2-7f5c-41a5-9304-ca4b2e26e789" \
  --header "Accept: application/json"
```

Trip response example

```
{
  "limit": 2,
  "offset": "0",
  "items": [
    {
      "id": "893b25ec-a706-13fb-30d2-e9ddf04b74ee",
      "car": {
        "id": "3918c95b-c4d6-b1e5-d1af-ed109c8cc2f4"
      },
      "efficiency": 0.7505115,
      "device": {
        "id": "10659000-b6a7-aa7a-3611-1431a2b99700"
      },
      "distance": 85698.06,
      "duration": 50647000,
      "start": {
        "date": 1455088585000
      },
      "stop": {
        "date": 1455139232000
      },
      "trip_event_counts": {
        "hard_turns": 3,
        "hard_accelerations": 0,
        "hard_brakes": 9
      }
    },
    {
      "id": "ba023697-b0da-7b57-04d7-2def1427b614",
      "car": {
        "id": "3918c95b-c4d6-b1e5-d1af-ed109c8cc2f4"
      },
      "efficiency": 0.6505123,
      "device": {
        "id": "10659000-b6a7-aa7a-3611-1431a2b99700"
      },
      "distance": 1250.1,
      "duration": 4437000,
      "start": {
        "date": 1460163232000
      },
      "stop": {
        "date": 1460167669000
      },
      "trip_event_counts": {
        "hard_turns": 0,
        "hard_accelerations": 0,
        "hard_brakes": 1
      }
    }
  ],
  "next": {
    "offset": "a1b2"
  }
}
```

Timeseries request example

In order to get the raw data of the second trip, the ID of this specific trip is needed:

```
curl --request GET \
  --url
"https://public.api.prd.drust.io/v2/trips/ba023697-b0da-7b57-04d7-2def14
27b614/timeseries" \
  --header "Authorization: \
Bearer ed65fca2-7f5c-41a5-9304-ca4b2e26e789" \
  --header "Accept: application/json"
```

Timeseries response example

```
{
  "limit": 1000,
  "offset": "0",
  "items": [
    {
      "device": {
        "id": "10659000-b6a7-aa7a-3611-1431a2b99700"
      },
      "date": 1460163232000,
      "speed": 5.15,
      "engine_rpm": 450,
      "engine_coolant_temperature": 20.47,
      "location": {
        "lon": 1.3869272,
        "lat": 43.5656068,
        "hdop": 1.44363,
        "altitude": 144,
        "satellites_used": 2,
        "satellites_visibles": 5,
        "cog": 136
      }
    },
    {
      "device": {
        "id": "10659000-b6a7-aa7a-3611-1431a2b99700"
      },
      "date": 1460163232250,
      "acceleration": {
        "x": 5.48,
        "y": 0,
        "z": -0.1
      }
    },
    ...
    // Data removed for brevity
  ],
  "next": {
    "offset": "c3d4"
  }
}
```

Reference

Data Types

Collections

Each object of type "collection" identifies a set of items, which have themselves a certain subtype. For instance, a collection of trips or a collection of events can be retrieved.
Collections are listed by pages. Each page gathers a set of data, a maximum size, an identifier pointing the current page and an identifier pointing to the next page (if applicable).

Name	Type	Description
`limit`	`Integer`	The maximum number of items listed in each page
`offset`	`String`	The pointer to the current page
`next`	`Object`	An object containing an "offset" field, pointing to the next page
`items`	`Array <{Sub-type}>`	The set of items of the current page

Trips

The type “Trip” describes the data summarizing a trip

Name	Type	Description
`id`	`String`	Unique identifier of a trip
`car`	`Car`	The vehicle used for this trip
`device`	`Device`	The telematic device used for this trip
`distance`	`Float`	Distance travelled (in meters) during this trip
`duration`	`Long`	Duration of this trip (in milliseconds)
`efficiency`	`Float`	Driving efficiency scoring, in percentage
`start_date`	`Long`	Start date of this trip (UNIX timestamp, in milliseconds)
`stop_date`	`Long`	End date of this trip (UNIX timestamp, in milliseconds)
`event_count`	`TripEventCount`	Number of driving events

Cars

The type “Car” describes vehicles and their characteristics.

Name	Type	Description
`id`	`String`	Unique identifier of the vehicle
`brand`	`String`	OEM brand
`engine_type`	`String`	Engine type: ● GASOLINE (essence) ● DIESEL (diesel)
`gearbox_type`	`String`	Gearbox type: ● MANUAL ● AUTOMATIC ● AUTOMATED_MANUAL ● VARIABLE_TRANSMISSION ● UNKNOWN
`model`	`String`	Specific vehicle model
`year`	`Integer`	Year of registration for this vehicle

Timeseries

The “Timeseries” type gathers the raw information contained in a trip. A trip can be considered as a big set of samples measured by a telematic device. Each measured value is stored in a unique time serie, and a consistent set of time series defines a trip.

Name	Type	Description
`device`	`Device`	The telematic device used for data acquisition
`date`	`Long`	UNIX timestamp (in milliseconds) when the measure was done
`speed`	`Float`	Instant speed (in km/h)
`gear`	`Integer`	Engaged gear
`location`	`Location`	Geolocation
`acceleration`	`Acceleration`	Instant acceleration
`engine_coolant_temperature`	`Float`	Engine coolant temperature (in °C)
`engine_rpm`	`Float`	Engine speed in RPM

Events

The “Event” type gives information about driving events, such as hard breaks and turns. Each event is associated to a specific trip and vehicle.

Name	Type	Description
`id`	`String`	Identifier of the event
`trip`	`Trip`	Trip while the event occured
`date`	`Long`	UNIX timestamp (in ms) when the acquisition of the event was done
`type`	`String`	Event type: ● HARD_BRAKE ● HARD_ACCELERATION ● HARD_TURN ● GEAR_CHANGE ● PARKED ● STOPPED ● MOVING
`payload`	`<Depending on the type of event>`	Additional data describing the event, depending on the the event type

Device

The type “Device” describes a telematic device associated to a car, enabling its data acquisition.

Name	Type	Description
`id`	`String`	Unique identifier of the telematic device

Trip Event Counts

The type “TripEventCount” is for counting the driving events in a trip.

Name	Type	Description
`hard_turns`	`Integer`	Number of hard turns
`hard_accelerations`	`Integer`	Number of hard accelerations
`hard_brakes`	`Integer`	Number of hard brakes

Locations

The “Location” type is for describing a geolocation.

Name	Type	Description
`lat`	`Double`	Latitude in degrees
`lon`	`Double`	Longitude in degrees
`hdop`	`Double`	The horizontal dilution of GPS precision
`altitude`	`Integer`	The Altitude in meters
`satellites_used`	`Integer`	Number of satellites used for this acquisition
`satellites_visibles`	`Integer`	Number of visible satellites when this acquisition was done
`cog`	`Integer`	“Course Over Ground” (NMEA VTG frame), in degrees

Accelerations

The type “Acceleration” describes a measure of the instant acceleration

Name	Type	Description
`x`	`Float`	Instant acceleration on the longitudinal axis of the vehicle (in g)
`y`	`Float`	Instant acceleration on the lateral axis of the vehicle (in g)
`z`	`Float`	Instant acceleration on the vertical axis of the vehicle (in g)

API Errors

Describes an error occuring when performing an API request (other than the authentication request)

Name	Type	Description
`status`	`String`	Error code
`description`	`String`	Error description

Authentication Successes

Describes a successful authentication.

Name	Type	Description
`access_token`	`String`	The token used to authenticate the calls to the API
`refresh_token`	`String`	The token used to request a new access token
`token_type`	`String`	Always “bearer”
`expires_in`	`Integer`	The number of seconds before the access token expiration
`scope`	`String`	This authorized scope for this token. Always “global”

Authentication Errors

Describes an authentication error when performing a API call.

Name	Type	Description
`error`	`String`	Error code
`error_description`	`String`	Error description

Operations

Listing of the available operations.

The standard base URL is `https://public.api.prd.drust.io`

Method	URL	Operation description
`POST`	`/oauth/token`	Request a new session
`GET`	`/v2/devices`	Retrieve the list of the accessible devices for the user
`GET`	`/v2/cars`	List the vehicles
`GET`	`/v2/trips`	List the trips
`GET`	`/v2/devices/{id}/trips`	Retrieve trips for a specific device
`GET`	`/v2/trips/{id}/events`	Retrieve the list of driving events for a specific trip
`GET`	`/v2/trips/{id}/timeseries`	Retrieve raw data for a specific trip
`GET`	`/v2/devices/{id}/timeseries`	Retrieve the raw data coming from specific device

Retrieve an authentication token

Description:

Method	`POST`
URL	`/oauth/token`
Header Content-Type	`application/x-www-form-urlencoded`

Body parameters:

Name	Type	Mandatory?	Description
`grant_type`	`String`	Mandatory	The token request type. Accepted values: ● *password* ● *refresh_token*
`scope`	`String`	Mandatory	The token authorization scope. Always *global*.
`username`	`String`	If grant_type *password*	The DRUST account username.
`password`	`String`	If grant_type *password*	The DRUST account password.
`refresh_token`	`String`	If grant_type *refresh_token*	The token needed to get a new access token.
`client_id`	`String`	Mandatory	The client identifier.
`client_secret`	`String`	Mandatory	The client personal secret.

Error Status:

Status	Description
`{"error": "invalid client"}`	The *client_id* or *client_secret* are invalid or not correct.
`{"error": "unauthorized"}`	The *client_id* or *client_secret* are missing.
`{"error": "invalid grant"}`	The username and/or password are not correct.
`{"error": "server error"}`	The username and/or password are missing, or any other error.
`{"error": "invalid scope"}`	The specified scope is missing or invalid, should be *global*.
`{"error": "unsupported grant type"}`	The grant type is not correct, should be *password* or *refresh_token*.
`{"error": "invalid request"}`	The authorization request is not correct.

List devices

Description:

Method	`GET`
URL	`/v2/devices`
Response type	`Collection<Devices>`

Request parameters:

Name	Type	Mandatory?	Description
`limit`	`Integer`	Optional	Page size (between 1 and 10000, by default: 1000)
`offset`	`String`	Optional	Page identifier

List cars

Description:

Method	`GET`
URL	`/v2/cars`
Response type	`Collection<Car>`

Request parameters:

Name	**Type**	**Mandatory?**	**Description**
`limit`	`Integer`	Optional	Page size (between 1 and 10,000, by default: 1,000)
`offset`	`String`	Optional	Page identifier

List trips

Description:

Method	`GET`
URL	`/v2/trips`
Response type	`Collection<Trips>`

Request parameters:

Name	Type	Mandatory?	Description
`start_date`	`Long`	Mandatory	UNIX “epoch” date (in ms)
`end_date`	`Long`	Mandatory	UNIX “epoch” date (in ms)
`limit`	`Integer`	Optional	Page size (between 1 and 10,000, by default: 1,000)
`offset`	`String`	Optional	Page identifier

Error status:

Status	Description
`TRIPS_NOT_CONSOLIDATED`	Non consolidated trips have been detected in the time interval. The set received is then empty.

List events

Description:

Method	`GET`
URL	`/v2/events`
Response type	`Collection<Events>`

Request parameters:

Name	Type	Mandatory?	Description
`start_date`	`Long`	Mandatory	UNIX “epoch” date (in ms)
`end_date`	`Long`	Mandatory	UNIX “epoch” date (in ms)
`limit`	`Integer`	Optional	Page size (between 1 and 10,000, by default: 1,000)
`offset`	`String`	Optional	Page identifier

Error status:

Status	Description
`TRIPS_NOT_CONSOLIDATED`	Non consolidated trips have been detected in the time interval. The set received is then empty.

List timeseries

Description:

Method	`GET`
URL	`/v2/timeseries`
Response type	`Collection<Timeseries>`

Request parameters:

Name	Type	Mandatory?	Description
`start_date`	`Long`	Mandatory	UNIX “epoch” date (in ms)
`end_date`	`Long`	Mandatory	UNIX “epoch” date (in ms)
`limit`	`Integer`	Optional	Page size (between 1 and 10,000, by default: 1,000)
`offset`	`String`	Optional	Page identifier

Error status:

Status	Description
`TRIPS_NOT_CONSOLIDATED`	Non consolidated trips have been detected in the time interval. The set received is then empty.

List trips for for one specific device

Description:

Method	`GET`
URL	`/v2/devices/{id}/trips`
Response type	`Collection<Trips>`

Request parameters:

Name	**Type**	**Mandatory?**	**Description**
`limit`	`Integer`	Optional	Page size (between 1 and 10,000, by default: 1,000)
`offset`	`String`	Optional	Page identifier
`start_date`	`Long`	Mandatory	UNIX “epoch” date (in ms)
`end_date`	`Long`	Mandatory	UNIX “epoch” date (in ms)

List driving events for one specific trip

Description:

Method	`GET`
URL	`/v2/trips/{id}/events`
Response type	`Collection<Events>`

Request parameters:

Name	Type	Mandatory?	Description
`limit`	`Integer`	Optional	Page size (between 1 and 10,000, by default: 1,000)
`offset`	`String`	Optional	Page identifier

List timeseries for one specific trip

Description:

Method	`GET`
URL	`/v2/trips/{id}/timeseries`
Response type	`Collection<Timeseries>`

Request parameters:

Name	Type	Mandatory?	Description
`limit`	`Integer`	Optional	Page size (between 1 and 10,000, by default: 1,000)
`offset`	`String`	Optional	Page identifier

List timeseries for one specific device

Description:

Method	`GET`
URL	`/v2/devices/{id}/timeseries`
Response type	`Collection<Timeseries>`

Request parameters:

Name	**Type**	**Mandatory?**	**Description**
`limit`	`Integer`	Optional	Page size (between 1 and 10,000, by default: 1,000)
`offset`	`String`	Optional	Page identifier
`start_date`	`Long`	Mandatory	UNIX “epoch” date (in ms)
`end_date`	`Long`	Mandatory	UNIX “epoch” date (in ms)