

TACHO ONLINE

Tacho Online API



VERSION 1.1.2

CONTENT

What's new?	3
Tacho activities	3
Welcome	3
More information	3
Terminology	3
TVS	3
Tacho Online	3
Tenant	3
Programming Guide	4
Introduction to The Tacho Online API	4
Preparing for Using the Tacho Online API	4
Making requests to Tacho Online API	5
Making HTTP requests	5
The base URL	5
Getting started with HTTP requests	6
Token request (OAuth)	6
Making an HTTP request (Example)	7
Request limits	8
Reference	9
Swagger	9
Tacho Online API - Single Tenant (For Customers)	10
Company	10
Drivers	14
Vehicle	21
TachographFiles	28
Tacho Online API - Multi tenants (For Partners)	31
Tenants	31
Company	32
Drivers	36
Vehicle	43
TachographFiles	51
Appendix A: Operation response codes	54
Example of Messages	54
Appendix B: Resources	55

Tacho Online API Resources	55
Other Resources	55
Appendix C: Solving known issues	55
Revision history.....	55
Information.....	56
Need help?.....	56
Important information.....	56

WHAT'S NEW?

Here you find an overview of what is new or has changed in Tacho Online API 1.0.0. Please find the [full revision history](#) at the end of this document.

TACHO ACTIVITIES

- Added new feature Tacho activities

WELCOME

Welcome to the Tacho Online API Reference Guide.

This document provides all the information you need to integrate Tacho Online into your applications using the Tacho Online API interface.

The documentation is divided into the following sections:

- [Terminology](#): In this chapter, you learn understanding terms used in the context of Tacho Online and Tacho Online API features.
- [Programming Guide](#): The programming guide contains a description of how to submit requests to the Tacho Online API and the data that is returned by the service, including an explanation of how to enable access to the service.
- [Reference](#): The Reference is a description of all available operations, including their parameters and the data returned by these operations.

MORE INFORMATION

Updated versions of this documentation can be acquired by contacting: support@tachoonline.dk.

TERMINOLOGY

Here is a description of the terms used in the context of Tacho Online and the Tacho Online API.

TVS

TungVognSpecialisten ApS (TVS) is the product owner

TACHO ONLINE

Tacho Online is the core component of the TVS solutions. Tacho Online is the webbased application for managing driving and rest times, files, drivers, vehicles and a set of advanced reporting and notifications tools.

TENANT

A tenant in Tacho Online is abbreviation for "The customer".

PROGRAMMING GUIDE

This programming guide is an introduction to using the Tacho Online API interface, how to access the service and how to interpret the output that is returned.

In order to access the Tacho Online service you need an API key and a Tacho Online account or a Partner account enabled. Otherwise, you will not be able to test the integration for your application.

NOTE: Please talk to your TVS contact or write to support@tachoonline.dk, if you do not have access to a Tacho Online or Partner account.

INTRODUCTION TO THE TACHO ONLINE API

The Tacho Online API allows you to access the Tacho Online service through a web-enabled application. These are the primary features accessible through the Tacho Online API:

- Driving & rest times files: file upload and download
- Tacho data: Tacho activities
- Drivers: Insert, update and delete drivers and retrieve driver information.
- Vehicles: Insert, update and delete vehicles and vehicle information.
- Control documents: Insert, update and delete documents and retrieve document information.

Tacho Online API is using access restrictions set up within Tacho Online. This affects all elements of the Tacho Online interface. For instance, you maybe be grant access rights to upload files, but not to download files. Please contact support@tachoonline.dk if you have questions about your access rights.

PREPARING FOR USING THE TACHO ONLINE API

The Tacho Online API can be made available to every customer or partner with a valid Tacho Online account.

Access the Tacho Online API with a valid token to enable access for your application, obtain an OAuth token (credentials) by doing the following:

- Partners: please contact your TVS contact, to receive your credentials (client_id, client_secret, username and password).
- Customers: please contact your sales contact or the support team: support@tachoonline.dk for accessing the API.

NOTE: After an access request has been made, the TVS team will authenticate the request. If the request is valid, the TVS Team will setup the appropriated access rights and you will received your access details and credentials.

MAKING REQUESTS TO TACHO ONLINE API

MAKING HTTP REQUESTS

This section explains how to use [HTTP](#) to issue requests to Tacho Online API.

IMPORTANT: only HTTPS requests are accepted. Requests using unencrypted HTTP are rejected!

The Tacho Online API generally uses HTTP:

- GET
- POST
- PATCH
- DELETE

Requests as the underlying transport mechanism for requests.

All requests are made using specific [URLs](#), passing parameter names and values as URL parameters. Responses are currently only available as [JSON](#).

You can experiment with Tacho Online API specific HTTPS requests by entering the request URL into the browser's address bar and submitting the request.

NOTE: If the format of the HTTP request is not valid you will get a corresponding error.

THE BASE URL

The base URL with the https scheme used is:

- <https://service.tachoonline.dk/>

Handling the response In case of an error, an error message is returned as plain text. The error message has the following layout:

- Errorcode: standard http errorcodes
- Message: custom message that provide a reason text (always in English)

400

Bad Request

Example Value | Model

```
{
  "version": "string",
  "statusCode": 0,
  "message": "string",
  "responseException": {
    "isError": true,
    "exceptionMessage": "string",
    "details": "string",
    "referenceErrorCode": "string",
    "referenceDocumentLink": "string",
    "validationErrors": [
      {
        "field": "string",
        "message": "string"
      }
    ]
  }
}
```

On success, responses will always return statusCode **200** "Success".

Code	Description
200	<i>Success</i>

GETTING STARTED WITH HTTP REQUESTS

For getting started with HTTP requests, we recommend using [POSTMAN](#).

Preconditions:

- POSTMAN (or any other tool where you can add headers)
- Valid client ID and Client secret
- Valid username and password

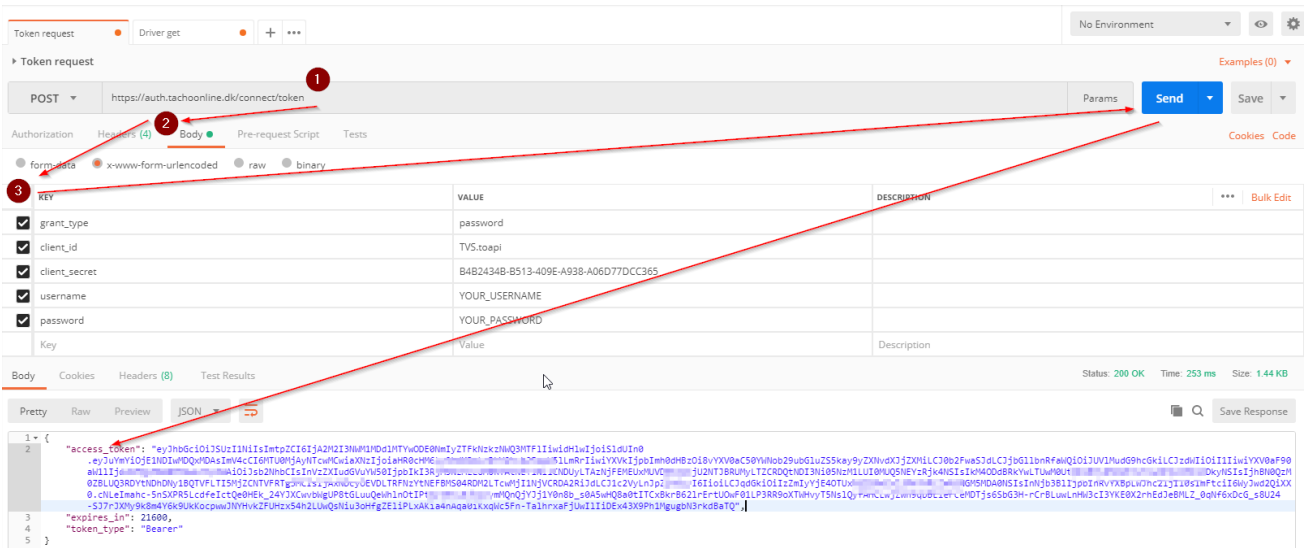
TOKEN REQUEST (OAUTH)

To generate your token, please enter the URL:

- <https://auth.tachoonline.dk/connect/token>

Add the following parameters (key/value):

- Key: **grant_type** - Value: password
- Key: **client_id** - Value: YOUR.CLIENTID
- Key: **client_secret** - Value: YOUR.CLIENTSECRET
- Key: **username** - Value: YOUR.USERNAME
- Key: **password** - Value: YOUR.PASSWORD



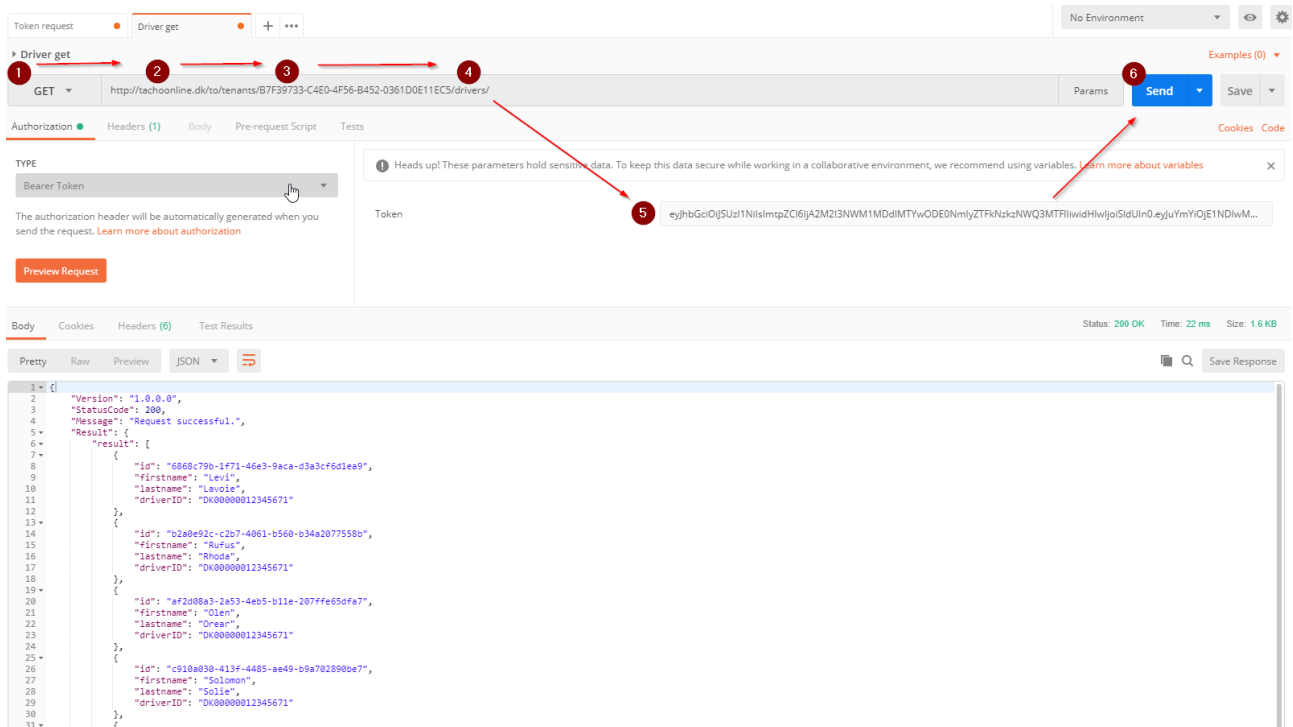
NOTE: you can decode the access token using [JSON Web Tokens](#) (or another service) to view the payload data.

MAKING AN HTTP REQUEST (EXAMPLE)

In this example, we will make a request for all [DRIVERS] on a specific [TENANT]:

Preconditions:

- POSTMAN (or any other tool where you can add headers)
 - Valid Access token
1. Select GET
 2. Add BaseURL: [https:// service.tachoonline.dk/](https://service.tachoonline.dk/)
 3. Specify the target tenant (add GUID): <https://service.tachoonline.dk/tenants/TENANTID>
 4. Specify what you want (in this case, we want a list off all the drivers for a tenant): <https://service.tachoonline.dk/tenants/TENANTID/drivers/>
 5. Add your Access token
 6. Click “Send”



REQUEST LIMITS

The number of requests that can be issued is limited. If the number of requests executed exceeds this limit, Tacho Online will return an error message and not process requests until there were no further requests within the limit-monitoring interval. Limits are defined by a maximum number of requests allowed in a certain time period.

- Request limit: **10** / second - **180** / minute

IMPORTANT: Should an application using Tacho Online API cause too much load on the system, the limit may be reduced at any time without prior notice and eventually access to Tacho Online API might be revoked completely if the problems are not fixed within a reasonable amount of time.

REFERENCE

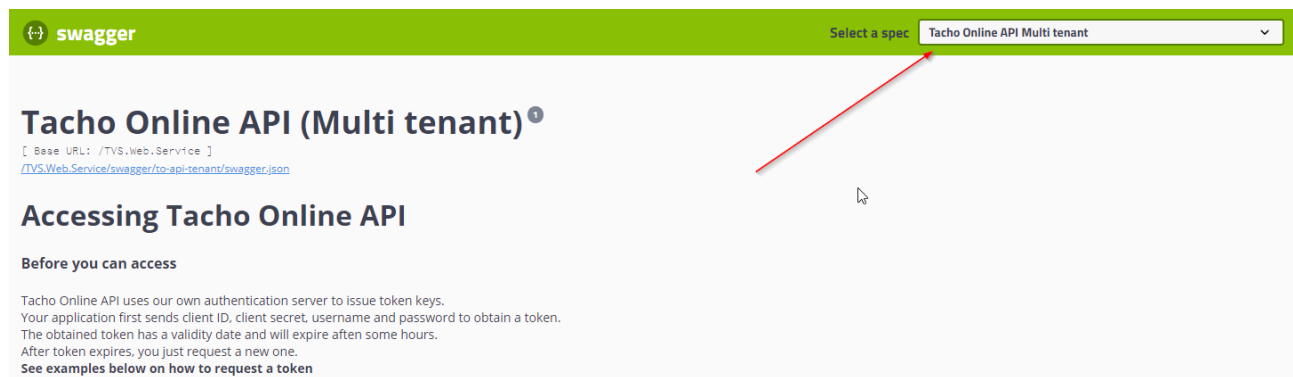
The Tacho Online API is divided into two areas:

- **Tacho Online API - Single tenant** (single usage): primarily used by our customers
- **Tacho Online API - Multi tenants** (multi usage): used by our partners

SWAGGER

Want to skip this reading and jump straight to the documentation provided using [Swagger](#), then [click here](#) or use the following link:

- <https://service.tachoonline.dk/doc/index.html>



NOTE: please select the appropriated “spec” before you start reading the swagger documentation.

TACHO ONLINE API - SINGLE TENANT (FOR CUSTOMERS)

COMPANY

[GET] /COMPANY

Get information about current company:

```
Response.CompanyDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       {
    <img alt="mouse cursor" data-bbox="243 308 258 323"/>
    <img alt="collapse icon" data-bbox="308 298 323 313"/> [
      uniqueItems: false
      CompanyDetails {
        id      string($uuid)
        name     string
        address1 string
        address2 string
        zipcode  string
        town     string
        country  string
        phone    string
        fax      string
        www      string
        email    string
        vatNumber string
        companyCardID string
      }
    ]
  }
}
```

[GET] /COMPANY/CONTROLDOCS

List companies control documents:

```
Response.ControlDoc {
  version      string
  statusCode   integer($int32)
  message      string
  result       {
    <img alt="collapse icon" data-bbox="323 708 338 723"/> [
      uniqueItems: false
      ControlDoc {
        id      string($uuid)
        name     string
        date     string($date-time)
        expiredDate string($date-time)
        daysBeforeWarning integer($int32)
        systemCreated boolean
      }
    ]
  }
}
```

[POST] /COMPANY/CONTROLDOCS

Create new company control document:

```
Response.ControlDocDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       ControlDocDetails {
    id          string($uuid)
               readOnly: true
    name        string
    date        string($date-time)
    expiredDate string($date-time)
    daysBeforeWarning integer($int32)
    internalNote string
    systemCreated boolean
               readOnly: true
  }
}
```

- **[REQUIRED] ControlDoc:** the control document object

[GET] /COMPANY/CONTROLDOCS/ {CONTROLDOCID}

Get control document details from the specific company:

```
Response.ControlDocDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       ControlDocDetails {
    id          string($uuid)
    name        string
    date        string($date-time)
    expiredDate string($date-time)
    daysBeforeWarning integer($int32)
    internalNote string
    systemCreated boolean
  }
}
```

- **[REQUIRED] ControlDocID:** the unique control document ID.

[DELETE] /COMPANY/CONTROLDOCS/ {CONTROLDOCID}

Delete existing company control document:

```
Response.ControlDocDetails ▾ {
  version      string
  statusCode   integer($int32)
  message      string
  result       ControlDocDetails ▾ {
    id          string($uuid)
               readOnly: true
    name        string
    date        string($date-time)
    expiredDate string($date-time)
    daysBeforeWarning integer($int32)
    internalNote string
    systemCreated boolean
               readOnly: true
  }
}
```

- **[REQUIRED] ControlDocID:** the unique control document ID.

[PATCH] /COMPANY/CONTROLDOCS/ {CONTROLDOCID}

Update: existing company document:

```
▾ [
  uniqueItems: false
  Operation ▾ {
    value      ▾ {
    }
    path       string
    op          string
    from        string
  }
}]
```

```

Response.ControlDocDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       ControlDocDetails {
    id          string($uuid)
               readOnly: true
    name        string
    date        string($date-time)
    expiredDate string($date-time)
    daysBeforeWarning integer($int32)
    internalNote string
    systemCreated boolean
               readOnly: true
  }
}

```

- **[REQUIRED] ControlDocID:** the unique control document ID.
- **[REQUIRED] ControlDoc:** the jsonPatchDocument object used to update the companys control document values

DRIVERS

[GET] /DRIVERS

Get list of existing drivers:

```
Response.Driver {
  version      string
  statusCode   integer($int32)
  message      string
  result       {
    <img alt="dropdown arrow" data-bbox="328 265 345 278"/> [
      uniqueItems: false
      Driver {
        id          string($uuid)
        firstname   string
        lastname    string
        driverID    string
      }
    ]
  }
}
```

[POST] /DRIVERS

Create new Driver:

```
Response.DriverDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       DriverDetails {
    id          string($uuid)
                 readOnly: true
    firstname   string
    lastname    string
    nickname    string
    driverID    string
    email       string
    mobile      string
    phone       string
    address1    string
    address2    string
    zipcode     string
    town        string
    country     string
    birthday    string($date-time)
    employedDate string($date-time)
  }
}
```

- **[REQUIRED] Driver:** driver object

[GET] /DRIVERS/{DRIVERID}

Get driver details:

```
Response.DriverDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       DriverDetails {
    id          string($uuid)
                readOnly: true
    firstname   string
    lastname    string
    nickname     string
    driverID    string
    email       string
    mobile      string
    phone       string
    address1    string
    address2    string
    zipcode     string
    town        string
    country     string
    birthday    string($date-time)
    employedDate string($date-time)
  }
}
```

- **[REQUIRED] DriverID:** the unique driver ID.

[DELETE] /DRIVERS/{DRIVERID}

Delete existing driver:

```
Response.DriverDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       DriverDetails {
    id          string($uuid)
                readOnly: true
    firstname   string
    lastname    string
    nickname     string
    driverID    string
    email       string
    mobile      string
    phone       string
    address1    string
    address2    string
    zipcode     string
    town        string
    country     string
    birthday    string($date-time)
    employedDate string($date-time)
  }
}
```

- **[REQUIRED] DriverID:** the unique driver ID.

[PATCH] /DRIVERS/{DRIVERID}

Update existing driver:

```

▼ [
  uniqueItems: false
  Operation ▼ {
    value
    path
    op
    from
  }
}]

```

```

Response.DriverDetails ▼ {
  version
  statusCode
  message
  result
  DriverDetails ▼ {
    id
    firstname
    lastname
    nickname
    driverID
    email
    mobile
    phone
    address1
    address2
    zipcode
    town
    country
    birthday
    employedDate
  }
}

```

- **[REQUIRED] DriverID:** the unique driver ID.
- **[REQUIRED] Driver:** the jsonPatchDocument object used to update the driver values

[GET] /DRIVERS/{DRIVERID}/CONTROLDOCS

List drivers control documents:

```
Response.ControlDoc {
  version      string
  statusCode   integer($int32)
  message      string
  result {
    uniqueItems: false
    ControlDoc {
      id          string($uuid)
      name        string
      date        string($date-time)
      expiredDate string($date-time)
      daysBeforeWarning integer($int32)
      systemCreated boolean
    }
  }
}
```

- **[REQUIRED] DriverID:** the unique driver ID.

[POST] /DRIVERS/{DRIVERID}/CONTROLDOCS

Create new driver control document:

```
ControlDocDetails {
  id          string($uuid)
              readOnly: true
  name        string
  date        string($date-time)
  expiredDate string($date-time)
  daysBeforeWarning integer($int32)
  internalNote string
  systemCreated boolean
              readOnly: true
}
```

```

Response.ControlDocDetails ▾ {
  version      string
  statusCode   integer($int32)
  message      string
  result       ControlDocDetails ▾ {
    id          string($uuid)
                readOnly: true
    name        string
    date        string($date-time)
    expiredDate string($date-time)
    daysBeforeWarning integer($int32)
    internalNote string
    systemCreated boolean
                readOnly: true
  }
}

```

- **[REQUIRED] DriverID:** the unique driver ID.
- **[REQUIRED] ControlDoc:** the control document object

[GET] /DRIVERS/{DRIVERID}/CONTROLDOCS/{CONTROLDOCID}

Get control document details from specific driver:

```

Response.ControlDocDetails ▾ {
  version      string
  statusCode   integer($int32)
  message      string
  result       ▾ [
    uniqueItems: false
    ControlDocDetails ▾ {
      id          string($uuid)
      name        string
      date        string($date-time)
      expiredDate string($date-time)
      daysBeforeWarning integer($int32)
      internalNote string
      systemCreated boolean
    }
  ]
}

```

- **[REQUIRED] DriverID:** the unique driver ID.
- **[REQUIRED] ControlDocID:** the unique control document ID.

[DELETE] /DRIVERS/{DRIVERID}/CONTROLDOCS/{CONTROLDOCID}

Delete existing driver control document:

```
Response.ControlDocDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       ControlDocDetails {
    id          string($uuid)
               readOnly: true
    name        string
    date        string($date-time)
    expiredDate string($date-time)
    daysBeforeWarning integer($int32)
    internalNote string
    systemCreated boolean
               readOnly: true
  }
}
```

- **[REQUIRED] DriverID:** the unique driver ID.
- **[REQUIRED] ControlDocID:** the unique control document ID.

[PATCH] /DRIVERS/{DRIVERID}/CONTROLDOCS/{CONTROLDOCID}

Update existing driver control document:

```
{
  uniqueItems: false
  Operation {
    value {
      path string
      op   string
      from string
    }
  }
}
```

```

Response.ControlDocDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       ControlDocDetails {
    id          string($uuid)
                readOnly: true
    name        string
    date        string($date-time)
    expiredDate string($date-time)
    daysBeforeWarning integer($int32)
    internalNote string
    systemCreated boolean
                readOnly: true
  }
}

```

- **[REQUIRED] DriverID:** the unique driver ID.
- **[REQUIRED] ControlDocID:** the unique control document ID.
- **[REQUIRED] ControlDoc:** the jsonPatchDocument object used to update the drivers control document values

[GET] /DRIVERS/{DRIVERID}/TACHOACTIVITY /{ACTIVITYMIN }/{ACTIVITYMAX }

Get Tacho activity details from a specific driver:

```

Response.DriverActivity {
  version      string
  statusCode   integer($int32)
  message      string
  result       {
    uniqueItems: false
    DriverActivity {
      description: DTO used for tachographfile activities
      time        string($date-time)
                  example: 2018-01-01T00:00:00
      inserted    integer($int32)
                  example: 1
                  Not inserted = 0,
                  Inserted = 1
      slot        integer($int32)
                  example: 0
                  No slot = null,
                  Slot 1 = 0,
                  Slot 2 = 1
      activity    integer($int32)
                  example: 0
                  No activity = null
                  Rest = 0,
                  Available = 1,
                  Work = 2,
                  Drive = 3
      regNo       string
                  example: AB12345
    }
  }
}

```

- **[REQUIRED] DriverID:** the unique driver ID.
- **[REQUIRED] ActivityMin:** the minimum activity date
- **[REQUIRED] ActivityMax:** the maximum activity date

VEHICLE

[GET] /VEHICLES

Get list of existing vehicles:

```
Response.Vehicle {
  version
  statusCode
  message
  result
    {
      uniqueItems: false
      Vehicle {
        id
        name
        vin
      }
    }
}
```

[POST] /VEHICLES

Create new vehicle:

```
Response.VehicleDetails {
  version
  statusCode
  message
  result
    VehicleDetails {
      id
      name
      licensePlate
      vin
      odometer
    }
}
```

- **[REQUIRED]** Vehicle: Vehicle object

[GET] /VEHICLES/{VEHICLEID}

Get vehicle details:

```
Response.VehicleDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       VehicleDetails {
    id          string($uuid)
               readOnly: true
    name        string
    licensePlate string
    vin         string
    odometer    number($double)
  }
}
```

- **[REQUIRED]** VehicleID: the unique vehicle ID.

[DELETE] /VEHICLES/{VEHICLEID}

Delete existing vehicle:

```
Response.VehicleDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       VehicleDetails {
    id          string($uuid)
               readOnly: true
    name        string
    licensePlate string
    vin         string
    odometer    number($double)
  }
}
```

- **[REQUIRED]** VehicleID: the unique vehicle ID.

[PATCH] /VEHICLES/{VEHICLEID}

Update existing vehicle:

```

{
  uniqueItems: false
  Operation {
    value {
      path
      op
      from
    }
  }
}

```

```

Response.VehicleDetails {
  version
  statusCode
  message
  result
  VehicleDetails {
    id
    name
    licensePlate
    vin
    odometer
  }
}

```

- **[REQUIRED] VehicleID:** the unique vehicle ID.
- **[REQUIRED] Vehicle:** the jsonPatchDocument object used to update the vehicle values.

[GET] /VEHICLES/{REGNO}

Get vehicle details from vehicle registration number:

```

Response.VehicleDetails {
  version
  statusCode
  message
  result
  VehicleDetails {
    id
    name
    regNo
    vin
    odometer
  }
}

```

- **[REQUIRED] RegNo:** the vehicles registration number

[GET] /VEHICLES/{VIN}

Get vehicle details from vehicle identification number:

```
Response.VehicleDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       VehicleDetails {
    id          string($uuid)
               readOnly: true
    name        string
    regNo       string
    vin         string
    odometer    number($double)
  }
}
```

- **[REQUIRED] VIN:** the vehicles identification number

[GET] /VEHICLES/{VEHICLEID}/CONTROLDOCS

List vehicles control documents:

```
Response.ControlDoc {
  version      string
  statusCode   integer($int32)
  message      string
  result       [
    uniqueItems: false
    ControlDoc {
      id          string($uuid)
      name        string
      date        string($date-time)
      expiredDate string($date-time)
      daysBeforeWarning integer($int32)
      systemCreated boolean
    }
  ]
}
```

- **[REQUIRED] VehicleID:** the unique vehicle ID.

[POST] /VEHICLES/{VEHICLEID}/CONTROLDOCS

Create new vehicle control document:

```
ControlDocDetails {
  id          string($uuid)
               readOnly: true
  name        string
  date        string($date-time)
  expiredDate string($date-time)
  daysBeforeWarning integer($int32)
  internalNote string
  systemCreated boolean
               readOnly: true
}
```

```
Response.ControlDocDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       ControlDocDetails {
    id          string($uuid)
               readOnly: true
    name        string
    date        string($date-time)
    expiredDate string($date-time)
    daysBeforeWarning integer($int32)
    internalNote string
    systemCreated boolean
               readOnly: true
  }
}
```

- **[REQUIRED] VehicleID:** the unique vehicle ID.
- **[REQUIRED] ControlDoc:** the control document object.

[GET] /VEHICLES/{VEHICLEID}/CONTROLDOCS/{CONTROLDOCID}

Get control document details from specific vehicle:

```
Response.ControlDocDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       {
    uniqueItems: false
    ControlDocDetails {
      id          string($uuid)
      name        string
      date        string($date-time)
      expiredDate string($date-time)
      daysBeforeWarning integer($int32)
      internalNote string
      systemCreated boolean
    }
  }
}
```

- **[REQUIRED] VehicleID:** the unique vehicle ID.
- **[REQUIRED] ControlDocID:** the unique control document ID.

[DELETE] /VEHICLES/{VEHICLEID}/CONTROLDOCS/{CONTROLDOCID}

Delete existing vehicle control document:

```

Response.ControlDocDetails ▾ {
  version          string
  statusCode        integer($int32)
  message           string
  result            ControlDocDetails ▾ {
    id              string($uuid)
                   readOnly: true
    name            string
    date            string($date-time)
    expiredDate     string($date-time)
    daysBeforeWarning integer($int32)
    internalNote    string
    systemCreated   boolean
                   readOnly: true
  }
}

```

- **[REQUIRED] VehicleID**: the unique vehicle ID.
- **[REQUIRED] ControlDocID**: the unique control document ID.

[PATCH] /VEHICLES/{VEHICLEID}/CONTROLDOCS/{CONTROLDOCID}

Update existing vehicle control document:

```

▾ [
  uniqueItems: false
  Operation ▾ {
    value          ▾ {
                  }
    path           string
    op             string
    from           string
  }
}]

```

```

Response.ControlDocDetails ▾ {
  version      string
  statusCode   integer($int32)
  message      string
  result       ControlDocDetails ▾ {
    id          string($uuid)
                readOnly: true
    name        string
    date         string($date-time)
    expiredDate  string($date-time)
    daysBeforeWarning integer($int32)
    internalNote string
    systemCreated boolean
                readOnly: true
  }
}

```

- **[REQUIRED] VehicleID:** the unique vehicle ID.
- **[REQUIRED] ControlDocID:** the unique control document ID.
- **[REQUIRED] ControlDoc:** the jsonPatchDocument object used to update the vehicles control document values

TACHOGRAPHFILES

[GET] /TACHOGRAPHFILES

Gets a list of existing tachograph files:

```
Response.TachographFile[] {
  version      string
  statusCode   integer($int32)
  message      string
  result       {
    uniqueItems: false
    TachographFile {
      id          string($uuid)
      filename    string
    }
  }
}
```

[POST] /TACHOGRAPHFILES

Upload a tachograph file to Tacho Online:

```
TachographFileData {
  description:
    Upload DTO for upload of tachograph files

    filename must also contain the file extension
    blob is the files content in base64

    {
      filename: "M_20180913_1100_AB12345_ZFF77XJTB0204054.ddd",
      blob: "YmFzZTY0...ZXhhbXBsZQ=="
    }

  filename    string
  example:    M_20180913_1100_AB12345_ZFF77XJT3F0204054.ddd
  blob        string($byte)
  example:    YmFzZTY0...ZXhhbXBsZQ==
}
```

[GET] /TACHOGRAPHFILES/{TACHOGRAPHFILEID}

Read information about existing tachograph files:

```
Response.TachographFile {
  version      string
  statusCode   integer($int32)
  message      string
  result       TachographFile {
    id          string($uuid)
    filename    string
  }
}
```

[GET] /TACHOGRAPHFILES/{TACHOGRAPHFILEID}/DOWNLOAD

Download tachograph file as base64:

```
Response.TachographFileData {
  version      string
  statusCode   integer($int32)
  message      string
  result       TachographFileData {
    description:

    filename    string
    blob        string($byte)
  }
}
```

Upload DTO for upload of tachograph files

*filename must also contain the file extension
blob is the files content in base64*

```
{
  filename: "M_20180913_1100_AB12345_ZFF77XJT3F0204054.ddd",
  blob: "YmFzZTY0...ZXhhbXBsZQ=="
}
```

example: `M_20180913_1100_AB12345_ZFF77XJT3F0204054.ddd`
 example: `YmFzZTY0...ZXhhbXBsZQ==`

[GET] /TACHOGRAPHFILES/{TACHOGRAPHFILEID}/DOWNLOAD/FILE

Download tachograph file as octet-stream:

```

Response.Error {
  version          string
  statusCode       integer($int32)
  message          string
  responseException
  ApiError {
    isError          boolean
    exceptionMessage string
    details          string
    referenceErrorCode string
    referenceDocumentLink string
    validationErrors
      [
        uniqueItems: false
        ValidationErrors {
          field          string
          message        string
          readOnly: true
          readOnly: true
        }
      ]
  }
}

```

TACHO ONLINE API - MULTI TENANTS (FOR PARTNERS)

TENANTS

[GET] /TENANTS

Get list of tenants that is accessible:

```
Response.TenantDto {
  version      string
  statusCode   integer($int32)
  message      string
  result       [
    uniqueItems: false
    TenantDto {
      description:

      id           string($uuid)
                  example: 695add9d-cc3c-415a-8e96-830a90554ae9
      name         string
                  example: Tacho Online
      vatNumber    string
                  example: DK12345678
    }
  ]
}
```

[GET] /TENANTS/{ID}

Get tenant details:

```
Response.TenantDto {
  version      string
  statusCode   integer($int32)
  message      string
  result       TenantDto {
    description:

    id           string($uuid)
                  example: 695add9d-cc3c-415a-8e96-830a90554ae9
    name         string
                  example: Tacho Online
    vatNumber    string
                  example: DK12345678
  }
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customer in Tacho Online

COMPANY

[GET] /TENANTS/{TENANTID}/COMPANY

Get information about company:

```
Response.CompanyDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       {
    uniqueItems: false
    CompanyDetails {
      id          string($uuid)
      name        string
      address1    string
      address2    string
      zipcode     string
      town        string
      country     string
      phone       string
      fax         string
      www         string
      email       string
      vatNumber   string
      companyCardID string
    }
  }
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online

[GET] /TENANTS/{TENANTID}/COMPANY/CONTROLDOCS

List companies control documents:

```
Response.ControlDoc {
  version      string
  statusCode   integer($int32)
  message      string
  result       {
    uniqueItems: false
    ControlDoc {
      id          string($uuid)
      name        string
      date        string($date-time)
      expiredDate string($date-time)
      daysBeforeWarning integer($int32)
      systemCreated boolean
    }
  }
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online

[POST] /TENANTS/{TENANTID}/COMPANY/CONTROLDOCS

Create new company document:

```
ControlDocDetails {
  id          string($uuid)
             readOnly: true
  name        string
  date        string($date-time)
  expiredDate string($date-time)
  daysBeforeWarning integer($int32)
  internalNote string
  systemCreated boolean
             readOnly: true
}
```

```
Response.ControlDocDetails {
  version      string
  statusCode    integer($int32)
  message      string
  result       ControlDocDetails {
    id          string($uuid)
               readOnly: true
    name        string
    date        string($date-time)
    expiredDate string($date-time)
    daysBeforeWarning integer($int32)
    internalNote string
    systemCreated boolean
               readOnly: true
  }
}
```

- [REQUIRED] **TenantID**: is the unique id that is used to identify customers in Tacho Online
- [REQUIRED] **ControlDoc**: the control document object

[GET] /TENANTS/{TENANTID}/COMPANY/CONTROLDOCS/ {CONTROLDOCID}

Get control document details from the specific company:

```
Response.ControlDocDetails {
  version      string
  statusCode    integer($int32)
  message      string
  result       ControlDocDetails {
    id          string($uuid)
    name        string
    date        string($date-time)
    expiredDate string($date-time)
    daysBeforeWarning integer($int32)
    internalNote string
    systemCreated boolean
  }
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] ControlDocID:** the unique control document ID

[DELETE] /TENANTS/{TENANTID}/COMPANY/CONTROLDOCS/ {CONTROLDOCID}

Delete existing company control document:

```
Response.ControlDocDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       ControlDocDetails {
    id          string($uuid)
               readOnly: true
    name        string
    date        string($date-time)
    expiredDate string($date-time)
    daysBeforeWarning integer($int32)
    internalNote string
    systemCreated boolean
               readOnly: true
  }
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] ControlDocID:** the unique control document ID

[DELETE] /TENANTS/{TENANTID}/COMPANY/CONTROLDOCS/ {CONTROLDOCID}

Update existing company control document:

```
{
  uniqueItems: false
  Operation {
    value {
      path
      op
      from
    }
  }
}
```

```

Response.ControlDocDetails ▾ {
  version          string
  statusCode        integer($int32)
  message          string
  result           ControlDocDetails ▾ {
    id              string($uuid)
                   readOnly: true
    name            string
    date            string($date-time)
    expiredDate     string($date-time)
    daysBeforeWarning integer($int32)
    internalNote    string
    systemCreated   boolean
                   readOnly: true
  }
}

```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] ControlDocID:** the unique control document ID
- **[REQUIRED] ControlDoc:** the jsonPatchDocument object used to update the companys control document values

DRIVERS

[GET] /TENANTS/{TENANTID}/DRIVERS

Get list of existing drivers:

```
Response.Driver {
  version      string
  statusCode   integer($int32)
  message      string
  result       {
    uniqueItems: false
    Driver {
      id          string($uuid)
      firstname   string
      lastname    string
      driverID    string
    }
  }
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online

[POST] /TENANTS/{TENANTID}/DRIVERS

Create new driver:

```
Response.DriverDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       {
    DriverDetails {
      id          string($uuid)
      firstname   string
      lastname    string
      nickname    string
      driverID    string
      email       string
      mobile      string
      phone       string
      address1    string
      address2    string
      zipcode     string
      town        string
      country     string
      birthday    string($date-time)
      employedDate string($date-time)
    }
  }
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] Driver:** the driver object

[GET] /TENANTS/{TENANTID}/DRIVERS/{DRIVERID}

Get driver details:

```
Response.DriverDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       DriverDetails {
    id          string($uuid)
    firstname    string
    lastname     string
    nickname     string
    driverID     string
    email        string
    mobile       string
    phone        string
    address1     string
    address2     string
    zipcode      string
    town         string
    country      string
    birthday     string($date-time)
    employedDate string($date-time)
  }
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] DriverID:** the unique driver ID

[DELETE] /TENANTS/{TENANTID}/DRIVERS/{DRIVERID}

Delete existing driver:

```
Response.DriverDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       DriverDetails {
    id          string($uuid)
    firstname    string
    lastname     string
    nickname     string
    driverID     string
    email        string
    mobile       string
    phone        string
    address1     string
    address2     string
    zipcode      string
    town         string
    country      string
    birthday     string($date-time)
    employedDate string($date-time)
  }
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] DriverID:** the unique driver ID

[PATCH] /TENANTS/{TENANTID}/DRIVERS/{DRIVERID}

Update existing driver:

```

▼ [
  uniqueItems: false
  Operation ▼ {
    value
    path
    op
    from
  }
}]

```

```

Response.DriverDetails ▼ {
  version
  statusCode
  message
  result
  DriverDetails ▼ {
    id
    firstname
    lastname
    nickname
    driverID
    email
    mobile
    phone
    address1
    address2
    zipcode
    town
    country
    birthday
    employedDate
  }
}

```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] DriverID:** the unique driver ID
- **[REQUIRED] Driver:** the jsonPatchDocument object used to update the driver values

[GET] /TENANTS/{TENANTID}/DRIVERS/{DRIVERID}/CONTROLDOCS

List drivers control documents:

```
Response.ControlDoc {
  version      string
  statusCode    integer($int32)
  message      string
  result       {
    uniqueItems: false
    ControlDoc {
      id          string($uuid)
      name        string
      date        string($date-time)
      expiredDate string($date-time)
      daysBeforeWarning integer($int32)
      systemCreated boolean
    }
  }
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] DriverID:** the unique driver ID

[POST] /TENANTS/{TENANTID}/DRIVERS/{DRIVERID}/CONTROLDOCS

Create new driver control document:

```
ControlDocDetails {
  id          string($uuid)
              readOnly: true
  name        string
  date        string($date-time)
  expiredDate string($date-time)
  daysBeforeWarning integer($int32)
  internalNote string
  systemCreated boolean
              readOnly: true
}
```



```

Response.ControlDocDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       ControlDocDetails {
    id          string($uuid)
               readOnly: true
    name        string
    date        string($date-time)
    expiredDate string($date-time)
    daysBeforeWarning integer($int32)
    internalNote string
    systemCreated boolean
               readOnly: true
  }
}

```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] DriverID:** the unique driver ID
- **[REQUIRED] ControlDoc:** the control document object.

[GET] /TENANTS/{TENANTID}/DRIVERS/{DRIVERID}/CONTROLDOCS/{CONTROLDOCID}

Get control document details from specific driver:

```

Response.ControlDocDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       [
    ControlDocDetails {
      id          string($uuid)
      name        string
      date        string($date-time)
      expiredDate string($date-time)
      daysBeforeWarning integer($int32)
      internalNote string
      systemCreated boolean
    }
  ]
  uniqueItems: false
}

```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] DriverID:** the unique driver ID
- **[REQUIRED] ControlDocID:** the unique control document ID

[DELETE] /TENANTS/{TENANTID}/DRIVERS/{DRIVERID}/CONTROLDOCS/{CONTROLDOCID}

Delete existing driver control document:

```
Response.ControlDocDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       ControlDocDetails {
    id          string($uuid)
               readOnly: true
    name        string
    date        string($date-time)
    expiredDate string($date-time)
    daysBeforeWarning integer($int32)
    internalNote string
    systemCreated boolean
               readOnly: true
  }
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] DriverID:** the unique driver ID
- **[REQUIRED] ControlDocID:** the unique control document ID

[PATCH] /TENANTS/{TENANTID}/DRIVERS/{DRIVERID}/CONTROLDOCS/{CONTROLDOCID}

Update existing driver control document:

```
{
  uniqueItems: false
  Operation {
    value {
      path string
      op   string
      from string
    }
  }
}
```

```
Response.ControlDocDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       ControlDocDetails {
    id          string($uuid)
               readOnly: true
    name        string
    date        string($date-time)
    expiredDate string($date-time)
    daysBeforeWarning integer($int32)
    internalNote string
    systemCreated boolean
               readOnly: true
  }
}
```

- **[REQUIRED] TenantID**: is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] DriverID**: the unique driver ID
- **[REQUIRED] ControlDocID**: the unique control document ID
- **[REQUIRED] ControlDoc**: the jsonPatchDocument object used to update the drivers control document values

[GET] /TENANTS/{TENANTID}/DRIVERS/{DRIVERID}/TACHOACTIVITY /{ACTIVITYMIN}/{ACTIVITYMAX}

Get tacho activity details from a specific driver:

```

Response.DriverActivity ▾ {
  version          string
  statusCode       integer($int32)
  message          string
  result           ▾ [
                    uniqueItems: false
                    DriverActivity ▾ {
                      description:    DTO used for tachographfile activities
                      time            string($date-time)
                                      example: 2018-01-01T00:00:00
                      inserted       integer($int32)
                                      example: 1
                                      Not inserted = 0,
                                      Inserted = 1
                      slot           integer($int32)
                                      example: 0
                                      No slot = null,
                                      Slot 1 = 0,
                                      Slot 2 = 1
                      activity       integer($int32)
                                      example: 0
                                      No activity = null
                                      Rest = 0,
                                      Available = 1,
                                      Work = 2,
                                      Drive = 3
                      regNo          string
                                      example: AB12345
                    }
  ]
}

```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] DriverID:** the unique driver ID
- **[REQUIRED] ActivityMin:** the minimum activity date (yyyy-MM-dd)
- **[REQUIRED] ActivityMax:** the maximum activity date (yyyy-MM-dd)

VEHICLE

[GET] /TENANTS/{TENANTID}/VEHICLES

Get list of existing vehicles:

```
Response.Vehicle {
  version      string
  statusCode   integer($int32)
  message      string
  result       {
    <img alt="mouse cursor" data-bbox="148 278 165 295" style="vertical-align: middle;"/>
    <img alt="collapse icon" data-bbox="325 265 342 282" style="vertical-align: middle;"/> [
      uniqueItems: false
      Vehicle {
        id      string($uuid)
        name    string
        vin     string
      }
    ]
  }
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online

[POST] /TENANTS/{TENANTID}/VEHICLES

Create new Vehicle:

```
VehicleDetails {
  <img alt="collapse icon" data-bbox="325 525 342 542" style="vertical-align: middle;"/> {
    id      string($uuid)
    name    string
    licensePlate string
    vin     string
    odometer number($double)
  }
  <img alt="mouse cursor" data-bbox="435 525 452 542" style="vertical-align: middle;"/>
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] Vehicle:** vehicle object

[GET] /TENANTS/{TENANTID}/VEHICLES/{VEHICLEID}

Get vehicles details:

```

Response.VehicleDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       VehicleDetails {
    id          string($uuid)
    name        string
    licensePlate string
    vin         string
    odometer    number($double)
  }
}

```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] VehicleID:** the unique vehicle ID

[DELETE] /TENANTS/{TENANTID}/VEHICLES/{VEHICLEID}

Delete existing vehicle:

```

Response.VehicleDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       VehicleDetails {
    id          string($uuid)
    name        string
    licensePlate string
    vin         string
    odometer    number($double)
  }
}

```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] VehicleID:** the unique vehicle ID

[PATCH] /TENANTS/{TENANTID}/VEHICLES/{VEHICLEID}

Update existing vehicle:

```

  ✓ [
    uniqueItems: false
    Operation ✓ {
      value
      path
      op
      from
    }
  ]

```

```

Response.VehicleDetails ✓ {
  version
  statusCode
  message
  result
  VehicleDetails ✓ {
    id
    name
    licensePlate
    vin
    odometer
  }
}

```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] VehicleID:** the unique vehicle ID
- **[REQUIRED] Vehicle:** the jsonPatchDocument object used to update the vehicle values

[GET] /TENANTS/{TENANTID}/VEHICLES/{REGNO}

Get vehicle details from vehicle registration number:

```

Response.VehicleDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       VehicleDetails {
    id          string($uuid)
                readOnly: true
    name        string
    regNo       string
    vin         string
    odometer    number($double)
  }
}

```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] RegNo:** the vehicles registration number

[GET] /TENANTS/{TENANTID}/VEHICLES/{VIN }

Get vehicle details from vehicle identification number:

```

Response.VehicleDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       VehicleDetails {
    id          string($uuid)
                readOnly: true
    name        string
    regNo       string
    vin         string
    odometer    number($double)
  }
}

```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] VIN:** the vehicles identification number

[GET] /TENANTS/{TENANTID}/VEHICLES/{VEHICLEID}/CONTROLDOCS

List vehicle control documents:

```
Response.ControlDoc {
  version          string
  statusCode       integer($int32)
  message          string
  result           {
    <img alt="mouse cursor" data-bbox="278 232 295 250"/>
    <img alt="dropdown arrow" data-bbox="388 250 405 265"/> [
      uniqueItems: false
      ControlDoc {
        id          string($uuid)
        name         string
        date         string($date-time)
        expiredDate  string($date-time)
        daysBeforeWarning integer($int32)
        systemCreated boolean
      }
    ]
  }
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] VehicleID:** the unique vehicle ID

[POST] /TENANTS/{TENANTID}/VEHICLES/{VEHICLEID}/CONTROLDOCS

Create new vehicle control documents:

```
ControlDocDetails {
  id          string($uuid)
              readOnly: true
  name         string
  date         string($date-time)
  expiredDate  string($date-time)
  daysBeforeWarning integer($int32)
  internalNote string
  systemCreated boolean
              readOnly: true
}
```



```
Response.ControlDocDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       ControlDocDetails {
    id          string($uuid)
                readOnly: true
    name        string
    date        string($date-time)
    expiredDate string($date-time)
    daysBeforeWarning integer($int32)
    internalNote string
    systemCreated boolean
                readOnly: true
  }
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] VehicleID:** the unique vehicle ID
- **[REQUIRED] ControlDoc:** the control document object

[GET] /TENANTS/{TENANTID}/VEHICLES/{VEHICLEID}/CONTROLDOCS/{CONTROLDOCID}

Get control document details from specific vehicle:

```
Response.ControlDocDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       [
    uniqueItems: false
    ControlDocDetails {
      id          string($uuid)
      name        string
      date        string($date-time)
      expiredDate string($date-time)
      daysBeforeWarning integer($int32)
      internalNote string
      systemCreated boolean
    }
  ]
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] VehicleID:** the unique vehicle ID
- **[REQUIRED] ControlDocID:** the unique control document ID

[DELETE] /TENANTS/{TENANTID}/VEHICLES/{VEHICLEID}/CONTROLDOCS/{CONTROLDOCID}

Get control document details from specific vehicle:

```
Response.ControlDocDetails {
  version      string
  statusCode   integer($int32)
  message      string
  result       ControlDocDetails {
    id          string($uuid)
               readOnly: true
    name        string
    date        string($date-time)
    expiredDate string($date-time)
    daysBeforeWarning integer($int32)
    internalNote string
    systemCreated boolean
               readOnly: true
  }
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] VehicleID:** the unique vehicle ID
- **[REQUIRED] ControlDocID:** the unique control document ID

[PATCH] /TENANTS/{TENANTID}/VEHICLES/{VEHICLEID}/CONTROLDOCS/{CONTROLDOCID}

Get control document details from specific vehicle:

```
{
  uniqueItems: false
  Operation {
    value {
      path string
      op   string
      from string
    }
  }
}
```

```

Response.ControlDocDetails ▾ {
  version          string
  statusCode        integer($int32)
  message          string
  result           ControlDocDetails ▾ {
    id              string($uuid)
                   readOnly: true
    name            string
    date            string($date-time)
    expiredDate     string($date-time)
    daysBeforeWarning integer($int32)
    internalNote    string
    systemCreated   boolean
                   readOnly: true
  }
}

```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] VehicleID:** the unique vehicle ID
- **[REQUIRED] ControlDocID:** the unique control document ID
- **[REQUIRED] ControlDoc:** the jsonPatchDocument object used to update the vehicles control document values

TACHOGRAPHFILES

[GET] /TENANTS/{TENANTID}/TACHOGRAPHFILES

Get list of existing tachograph files:

```
Response.TachographFile[] {
  version      string
  statusCode   integer($int32)
  message      string
  result       {
    uniqueItems: false
    TachographFile {
      id          string($uuid)
      filename    string
    }
  }
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online

[POST] /TENANTS/{TENANTID}/TACHOGRAPHFILES

Upload a tachograph file to Tacho Online:

```
TachographFileData {
  description:

    Upload DTO for upload of tachograph files

    filename must also contain the file extension
    blob is the files content in base64

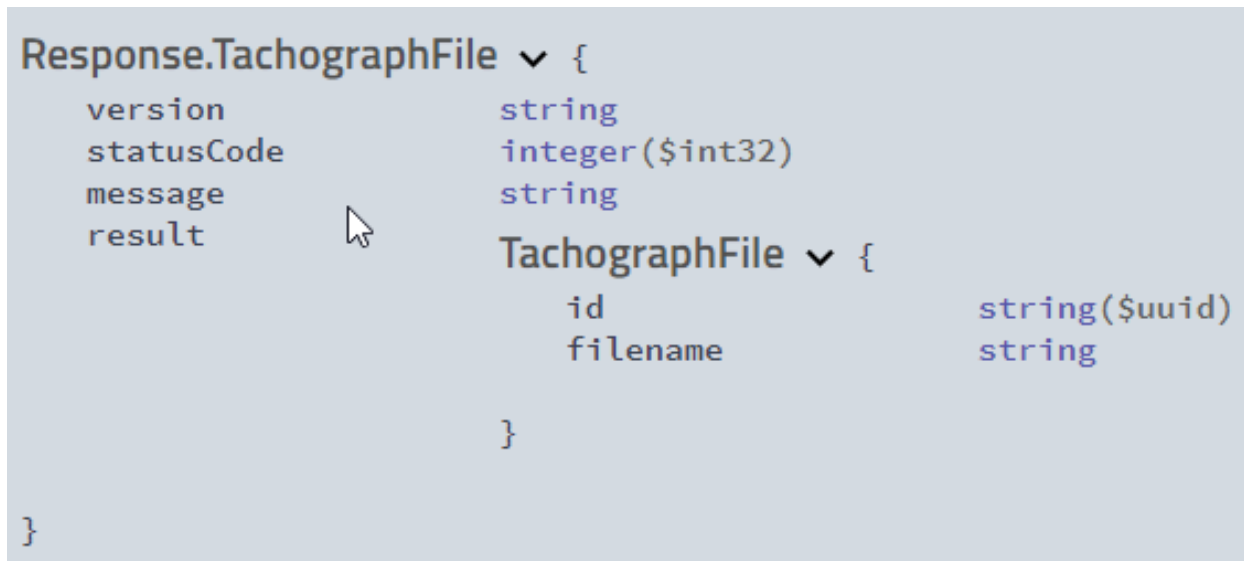
    {
      filename: "M_20180913_1100_AB12345_ZFF77XJT3F0204054.ddd",
      blob: "YmFzZTY0...ZXhhbXBsZQ=="
    }

  filename      string
  example:      M_20180913_1100_AB12345_ZFF77XJT3F0204054.ddd
  blob          string($byte)
  example:      YmFzZTY0...ZXhhbXBsZQ==
}
```

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online

[GET] /TENANTS/{TENANTID}/TACHOGRAPHFILES/{TACHOGRAPHFILEID}

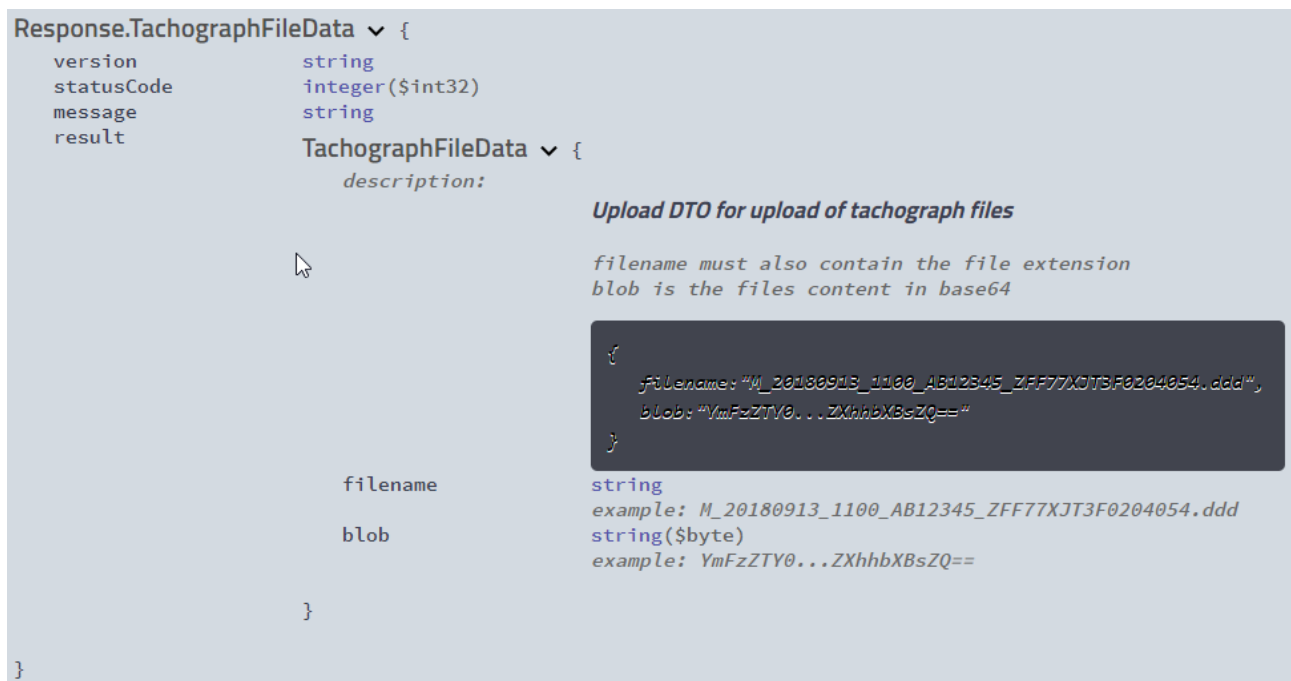
Read information about an existing tachograph file:



- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] TachographFileID:** the unique tachograph file ID (UUID)

[GET] /TENANTS/{TENANTID}/TACHOGRAPHFILES/{TACHOGRAPHFILEID}/DOWNLOAD

Download tachograph file as base64:



- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] TachographFileID:** the unique tachograph file ID (UUID)

[GET] /TENANTS/{TENANTID}/TACHOGRAPHFILES/{TACHOGRAPHFILEID}/DOWNLOAD/FILE

Download tachograph file as octet-stream:

- **[REQUIRED] TenantID:** is the unique id that is used to identify customers in Tacho Online
- **[REQUIRED] TachographFileID:** the unique tachograph file ID (UUID)

APPENDIX A: OPERATION RESPONSE CODES

Handling the response In case of an error, an error message is returned as plain text. The error message has the following layout:

- Errorcode: standard http error codes
- Message: custom message that provide a reason text (always in English)

400

Bad Request

Example Value | Model

```
{
  "version": "string",
  "statusCode": 0,
  "message": "string",
  "responseException": {
    "isError": true,
    "exceptionMessage": "string",
    "details": "string",
    "referenceErrorCode": "string",
    "referenceDocumentLink": "string",
    "validationErrors": [
      {
        "field": "string",
        "message": "string"
      }
    ]
  }
}
```

EXAMPLE OF MESSAGES

In this example, we look at the error codes from a bad request from: Uploading of a tachograph file to Tacho Online ([POST] /Tenants/{TenantID}/TachographFiles):

Possible 400 responses

<i>statusCode</i>	<i>message</i>
400	<i>FileUploadErrorNo file found in POST body</i>
400	<i>FileUploadErrorNo filename provided</i>
400	<i>FileUploadErrorFilename not valid</i>
400	<i>FileUploadErrorFilename contains invalid chars</i>
400	<i>FileUploadErrorFilename not valid, missing extension</i>
400	<i>FileUploadErrorFilesize is to small</i>
400	<i>FileUploadErrorUnable to read timestamp from filename</i>
400	<i>FileUploadErrorFile is more than 12 months old</i>
400	<i>FileUploadErrorInvalid file content, unable to parse file structure</i>
400	<i>FileUploadErrorFiletype unsupported, Only driver card or vehicle unit file are supported</i>
400	<i>FileUploadErrorUnable to read driver card number from file, please check for invalid data structure</i>
400	<i>FileUploadErrorUnable to read vehicle identification number(VIN) from file, please check for invalid data structure</i>

APPENDIX B: RESOURCES

TACHO ONLINE API RESOURCES

- Get the latest documentation and examples from: support@tachoonline.dk (online archive “Developer” will be added in the future)
- Tacho Online API key request. Please contact: support@tachoonline.dk (online form will be added in the future)
- Technical support: contact our customer support team: support@tachoonline.dk

OTHER RESOURCES

- [Wikipedia article about REST API](#)
- [OAuth](#)
- [POSTMAN](#)
- [JSON Web Tokens](#)
- [HTTP Error Codes](#)
- [JSON](#)
- [StackOverflow](#)

APPENDIX C: SOLVING KNOWN ISSUES

No, known issues. Please contact support@tachoonline.dk if you encounter any problems.

REVISION HISTORY

Below you see the full revision history for the Tacho Online API

REVISION	DATE	DESCRIPTION	AUTHOR
1.0.0	2018-11-09	Document created	SPE
1.1.0	2018-11-19	Added support for handling Tacho activities and general editorial changes.	SPE
1.1.1	2018-12-14	Documentation corrections to the base URL	SPE
1.1.2	2019-12-16	Updated general layout + removed “draft” water mark	SPE

INFORMATION

NEED HELP?

If you have any questions, feel free to contact us on: support@tachoonline.dk.

IMPORTANT INFORMATION

This guide is made by **TungVognsSpecialisten ApS** (32090990) and may be shared and used by third parties as long as reference is owned by TVS, as well as a link to our websites: www.tungvogn.dk and www.tachoonline.dk.

If you wish to use or change the material in any other way, please contact TVS at tvsv@tungvogn.dk.