AutotelexPro DataAPI webservice

Interface description of the integrated DataAPI Webservice

Autotelex BV

Versie: 4.0

Niels Brouwers

Document History

Revisies

Version	Status	Date	Author	Changes
1.1	English	15-1-2016	Niels Brouwers	Initial english translation
1.2		17-3-2016	Niels Brouwers	Extensions for Peugeot Merged with AutotelexPRO documentation English descriptions
1.3		18-3-2016	Niels Brouwers	Added objects, fields, more detail
1.4		21-3-2016	Niels Brouwers	Added Biedingen description and relevant objects
1.5		14-4-2016	Niels Brouwers	Renamed GetvehicleData to GetVehicleDataPRO due to naming conflicts in code. Added Inruilwaarde in description. Added BiedingData object Minor changes.
1.6		5-4-2016	Niels Brouwers	Description for uploading images added.
1.7		30-5-2016	Niels Brouwers	Described status object in little more detail. Added BandenParameters, SchadeParameters, SchadeOmschrijving Object descriptions
1.8				Updated file uploading and manipulation section.
1.9				Added ExternalProcessing object to indicate what should be done with the vehicle for 3rd parties.
2.0				Added GetHandelaren() and updated GetBodCriteria() description
2.1				Added LicensePlateCheckResult object and description to the webservice. Updated GetHandelaren() filter.
2.2				Correction to Filter in GetHandelaren().
2.3		20-3-2017		Added new BandenParameters description. Segment, SegmentDescription added in BasisGegevens
2.4		31-3-2017		European Type Approval field in RDWGegevens added.
2.5		10-4-2017		Moved VehicleParameters to separate document.
2.6		11-07-2017	Brian Swart	Added TankInhoud, MotorCode, MaximumMassaAutonoomGeremd and MaximumMassaMiddenasGeremd.
2.7		23-8-2017	Niels Brouwers	Added NoInterest call.

2.8		28-9-2017	Niels Brouwers	Added VehicleIngekocht and VehicleNietIngekocht description
2.9	English	30-3-2018	Okan Kurt	NumberOfGears added in VehicleInfo
3.0		12-6-2018	Okan Kurt	Added InsertSignature Added DeleteSignature
3.1		06-09-2019	Okan Kurt	Nieuwprijs fields renamed
3.2		11-02-2020	Dennis Vreman	New fields added to VehicleBasisGegevens for electric vehicles.
3.3		7-4-2020	Niels Brouwers	Rewrote image upload method. .Net Core lacks support for MTOM, making that method obsolete for the future.
3.4		26-8-2020	Niels Brouwers	Added JSON documentation
3.5		4-11-2020	Niels Brouwers	Error in parameters fixed
3.6		10-02-2020	Nick Lindeloo de Jong	Added Options and Packets descriptions in VoertuigBasisGegevens.
3.6.1		26-04-2020	Niels Brouwers	TCO Result added
3.6.2		4-06-2021	Niels Brouwers	Added TCO input parameters and defaults
3.7		06-12-2021	Nick Lindeloo de Jong	Added EV tech specs and import alert values
3.71		20-12-2021	Niels Brouwers	Fixed errors
3.8		26-01-2022	Nick Lindeloo de Jong	Added charging times to EV techspecs
3.9		08-02-2022	Niels Brouwers	Additional explanations, typo’s fixed.
4.0		22-03-2022	Okan Kurt	Added Jato

Content

Introduction

This document describes the Autotelex data API webservice and the possibilities to integrate third party applications with the AutotelexPRO website using this webservice.

Two distinct modes are described, the basic mode provides stateless data delivery about a vehicle and the advanced mode describes vehicle file management including making full use of the features that AutotelexPRO offers to its users.

Usage by third party applications

This chapter will describe some characteristics of the webservice and some typical scenario's for using this webservice. But first, a couple of terms that will be used in this chapter.

Terms

Vendor

The vendor is the third party application that is using this webservice. It is responsible for supplying the information needed to successfully retrieve vehicle data and update and extend it with additional data. A vendor is a trusted party from Autotelex and has received the rights to use the webservice under certain conditions (contractual agreements apply but are not part of this document).

Third party application

The application that will access the Autotelex webservice to retrieve/use or manipulate the data supplied or stored.

Basic and Advanced mode

There are 2 modes of operation for this particular webservice, we will describe both of these in detail in the following sections.

Webservice

The webservice described in this document is a standardized XML SOAP webservice based on Microsoft WCF technology. By automatically generating the WSDL from the interface definition in code, correct typing is ensured. Due to this and the fact that the webservice is an ever expanding work in progress, this documentation may leap slightly behind the reality.

Whenever in doubt, the WSDL is leading and conforms to the actual implementation of the webservice.

Besides the XML endpoint, a specific JSON endpoint is maintained as well for this webservice, this will be briefly described in separate chapter.

Compatibility

It is our intention not to make breaking changes to this webservice, if this may happen we will inform you well ahead of time (at least 3 months) using the contact information you provided to us when signing the contracts.

Basic mode - data retrieval

The basic mode will suffice for most vendors, it enables a vendor to request data for a given license number and optionally choose a different type or adjust the vehicle parameters to influence the valuation of the vehicle.

Only 2 calls are valid when using this approach. The vendor is responsible for all costs of the data retrieval and can/should manage handling and distribution of these cost to possible third parties that use his service on his/her behalf.

Basic mode scenario

The steps to retrieve data are as follows:

1. Application calls GetVendorToken()
2. Upon successfully retrieving the vendor token, a call to GetVehicleDataPRO() can be done with at least a license number as parameter (and the vendor token).
3. *Optional*: a second call to GetVehicleDataPRO() specifying the license number AND the specific Autotelex type of the vehicle. In case the second call did not return the correct type, a second optional call can correct for that and the specific valuation for that type can be retrieved.

The third party application will receive data based on their contract, there are a couple of options which can be discussed with your Autotelex account manager.

Advanced mode – extended calls

Advanced mode was developed for vendors that want to make use of the same possibilities that the AutotelexPRO website offers to its users.

This includes:

- keeping and managing a file of earlier requested vehicles
- possibility for updating them as new data becomes available
- requesting bids
- retrieving notifications of vehicle changes (the vendor will be required to provide a webservice to receive these signals)
- advertise the vehicle
- vehicle lifecycle management (bought, not bought, received, advertised, sold, transported, etc.)

Obviously, this requires a lot more effort on the part of the vendor. In the following we will describe some scenario's and the additional requirements that must be met.

User

A user is a user of the third party application, our webservice needs a way to uniquely identify a user so it can manage access to the user's data (only the user + vendor are be able to access it).

With AutotelexPRO subscription

A user can be an AutotelexPRO website user, and if so, his data can also be accessed through that site. More importantly, the financial part of administering his actions and the accompanied costs that go with it can also be handled through the webservice. This eases the burden on the third party, they do not need to keep a record of the payable actions anymore as was the case in the basic mode.

Without AutotelexPRO subscription

If the user does not have a subscription to the AutotelexPRO Website, but the third party application still wants to be able to access the advanced features then a user must be created or uniquely identified (through his email-address) so the data security can be guaranteed at all times.

Scenario's

Scenario 1 – creating a vehicle file

An example scenario for the advanced mode looks like this:

1. Application calls GetVendorToken()
2. Application calls GetCustomerToken(), the user is required to enter his/her AutotelexPRO website credentials through the third party application
3. Upon successfully retrieving the vendor token and customer token, a call to GetVehicleDataPRO() can be done with at least a license number as parameter (and the tokens). Vehicle data will be returned with a unique identifier for the vehicle's file that is created.

4. A call to UpdateVehicleData() is done to add some selected options and change the vehicle type to the correct type, using the unique identifier to identify the vehicle file.
5. A GetVehicleDataPRO() can be performed anytime to retrieve the latest state of the vehicle file.

Scenario 2 – adding/updating data

Once a file is created, the third party application must at least store the user and the unique file identifier locally to be able to access the file in the future.

Let's say, the car was damaged during transport and some damages need to be added (damages affect the valuation of a car):

1. A call to UpdateVehicleData() is done to add the damages to the vehicle using the unique identifier, the customertoken and the vendortoken.
2. Upon success of the previous call, a call to GetVehicleDataPRO() is done. It could be that besides the damages, some other data is updated as well (e.g. vehicle reported as stolen, images added through website, vehicle arrived at company etc.), it is up to the third party application to handle this data.

An UpdateVehicle() overwrites the entire vehicle file, meaning that properties that are not filled out in the VehicleParameters() but were retrieved during the GetVehicleDataPRO(), will be deleted in the vehicle file and can not be retrieved again.

Scenario 3 – requesting bids

The scenario for requesting bids is as follows:

1. Assuming a vehicle is created and configured, the third party application must first call GetBodCriteria() to check if all the required fields are filled out. Failure to comply with these criteria may result in no bid, a bid of 1 euro or a plain reject of the request – depending on the specific bid partner.
2. One or more bids are requested for the vehicle using the GetBiedingen() call.
3. When the bid is received a signal is called to the third party application's webservice.
4. The signal indicates that a call to GetVehicleDataPRO() should be done with the unique identifier of the file so the third party application can update its data accordingly (in this case, presumably only the bid was added – but in the meantime, more bids could have been added as well so a complete re-synchronization is advised).

Signal webservice

As mentioned earlier, in the advanced mode a means of communication must be realized between the Autotelex webservice and the third party application. We have implemented an easy and flexible way to deal with this.

Third party applications are subscribed to particular events that can occur in the Autotelex webservice. Bidding partners can receive a signal when a request for a bid on a vehicle is executed, when the vehicle is allocated or not, when the bid needs to be renewed etc. Third party applications can receive a signal when a vehicle is updated (e.g. a bid was received or damages were added).

In general the working is always the same. Autotelex will call a very simple webservice supplied by the partner that needs to receive an event-type and the ID of the vehicle to which the event applies. The partner then usually does a GetVehicleDataPRO() using the ID and updates its internal data accordingly.

Signal vehicle update available

Upon request of our major partners, the signal is implemented by issuing a simple HTTP POST request with a couple of parameters in the body.

Request:

```
POST http://signaal.3rdparty.nl/signalReceiver HTTP/1.1
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Host: signaal.3rdparty.nl
Content-Length: 57
Expect: 100-continue

token=fdfd2fdd3c75056adcf4c5e6845346856b0616ba&id=2247282&operation=update&reason=newbid
```

(expected) Response:

```
HTTP/1.1 200 OK
Date: Mon, 12 Jan 2015 09:19:17 GMT
Server: Apache/2.2.15
X-Powered-By: PHP/5.3.29
Connection: close
Transfer-Encoding: chunked
Content-Type: application/json; charset=UTF-8

{"success":true,"code":0,"response":""}
```

In case the response is not successful (error during transmission or success = false), the error will be logged and the event marked as not received. A backup method will be available to retrieve all the missed signals since the last call, it is advised to call this backup method daily.

Authorisation

Authorisation through dynamic or static token

Oftentimes, partners will want to know who sent the request or enforce some form of authorization or additional constraints on the caller.

We provide for that by calling a login and logout before and after the signal call. In the most common scenario, a static token can be supplied that does not change and may be a dummy (unused). If the third party chooses to use this, fine.

Whenever the third party wants to realize a dynamic token, there is a standardized procedure for that as well. The entire call for the signal will then be:

1. Call to login URL from third party with credentials, a token / unique identifier should be returned.
2. Call to signal webservice of third party using the token.
3. Logout with the token

Login

Login can be realized using one 'autotelex' token or using credentials per user. Using credentials per user may impact the lead-time of a project as this involves minor configurations that need to be planned.

Request

```
POST http://signaal.3rdparty.nl/ login HTTP/1.1
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Host: signaal.3rdparty.nl
Content-Length: 47
Expect: 100-continue

user=autotelexpro@autotelex.nl&password=1234567890
```

(expected) Response

```
HTTP/1.1 200 OK
Date: Mon, 12 Jan 2015 09:19:17 GMT
Server: Apache/2.2.15
X-Powered-By: PHP/5.3.29
Connection: close
Content-Type: application/json; charset=UTF-8
Content-Length: 99

{"success":true,"code":0,"response":"","data":{"token":"fdfd2fdd3c75056adcf4c5e684534685
6b0616ba"}}
```

Logout

Request

```
POST http://signaal.3rdparty.nl/logout HTTP/1.1
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Host: signaal.3rdparty.nl
Content-Length: 46
Expect: 100-continue

token=fdfd2fdd3c75056adcf4c5e6845346856b0616ba
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 12 Jan 2015 09:19:18 GMT
Server: Apache/2.2.15
X-Powered-By: PHP/5.3.29
Connection: close
Transfer-Encoding: chunked
Content-Type: application/json; charset=UTF-8

{"success":true,"code":0,"response":"","data":true}
```

Webservice interfaces

This chapter describes the interfaces that can be called and the parameters and objects that are involved when calling these interfaces.

GetVendorToken

Description

The vendortoken identifies the calling application. Autotelex will grant the vendor one or more rights which enable him to call this webservice and retrieve / manipulate data. A vendor is considered a trusted partner. The idea is that for each new application a new vendortoken will be requested (so we can distinguish which application called which interface).

Parameters

The following parameters are required for a succesful call:

- username
  Vendorname / username to identify the vendor. These credentials are managed by Autotelex.
- password
  The password for the vendor.

Result

An object that will contain the status and a GUID (token). If the status indicates success, the token can be used in subsequent calls.

In general, tokens do not expire, unless Autotelex blocks them. It is safe to store a token on a device, if need be, Autotelex can refresh the token and all calls from devices using the older token will fail.

GetVendorToken Objects

Status

Description

In general, for each methods a status will be returned so the caller can quickly see if the call succeeded or not.

Attributes

- GenericCode
  Generic code indicating the status of the call. See table 1 for possible values.
- Code - Int
  Deprecated, use GenericCode.
- Message - String
  Textual field that explains what went wrong or a comprehensive error message (test environment).

The status field is a generic object that is used in all calls to return the status of the entire call or the status of specific objects in the result. The GenericCode field is leading but can be accompanied by different messages to accommodate for the different situations.

The messages listed here are incomplete, because this webservice relies on other systems to gather it's data and if there is a specific error in one of those systems it will be propagated through the message field (usually with GenericCode, GeneralError).

GenericCode	Description	Possible Messages
0	OK	- OK
1	OK, with addition info	- Kies (optioneel) een andere uitvoering.
2	General Error	- Ongeldig of onbekend Kenteken. - Er is geen kenteken en datum eerste toelating bekend, er kan daarom geen taxatiewaarde worden afgegeven. - Taxatie kan niet worden uitgevoerd. - U heeft geen rechten voor deze data of de gegevens zijn niet bekend. - Geen data. - VehicleStateID niet geldig.
3	Customertoken invalid	- Ongeldig customertoken. -Gebruiker onbekend.
4	Vendortoken invalid	- Vendortoken ongeldig.

Table 1 possible values to be found in the status object.

The same Status object is used in all of the calls and contains the same GenericCode values for all objects (hence the name).

ClientID

Description

The ClientID object is the result object of interest in a call to GetVendorToken(). It contains the token if the call succeeded.

Attributes

- Status – Status object
  Object indicating the status of the call.
- Token - GUID
  GUID that belongs to the vendor credentials supplied. The format is 4 bytes, 3 times 2 bytes, and finally 6 bytes, so: 00000000-0000-0000-0000-000000000 (however, if you get all 0's then something definitely went wrong, check the status for OK first).

GetVehicleDataPRO

Description

The GetVehiclaData call returns the vehicle's data and exposes different behavior depending on the mode that the caller wants to use it. Choosing the mode of operation is controlled by the vehicle parameters.

Parameters
The following parameters are expected:

- Token - String
  The vendor token of the caller.
- Vehicle - VehicleParameters
  This parameter contains all the variables that can be adjusted on a vehicle to influence the valuation or update the vehicle file that is stored.

Result

The resulting DataAPI_Vehicle object is an object containing a status and the entire vehicle description.

VehicleData Objects

This chapter describes the objects and fields involved in the GetVehicleDataPRO call.

DataAPI_Vehicle

Description

This object is returned by the GetVehicleDataPRO() and UpdateVehicle() calls. Depending on the configuration at the server side and the input parameters this object will be filled out with vehicle data.

Attributes

- Status – Status object
  indicates if the call succeeded or not.
- AlternatieveUitvoeringen – VehicleType object
  A list with alternative types. Default the beste mathcing vehicle type is returned, the caller can however override this by selecting another type from this list and calling the GetVehicleDataPRO()/UpdateVehicleData() again with the type explicitly specified in the vehicle parameters.
  BasisGegevens – VehicleBasisGegevens object
  Basic information about the vehicle.
- EigenaarHistorieRDWGegevens – RDWGegevensEigenaar object
  Chronologic list of events and owners pertaining to the vehicle based on RDW data.

- UitgebreideGegevens – RDWGegevens object
  Data directly taken from the official RDW register.
- WaardeGegevens – VehicletaxData object
  Valuation of the vehicle.
- VehicleParams – VehicleParameters
  Vehicle specific information, this is typically data recorded when a valuation of the car must be done.
- LicensePlateCheckData – LicensePlateCheckResult object
  An object indicating results on some checks on the license plate formatting or link with the VIN number (meldcode).

VehicleBasisGegevens

Description

Dit object bevat de basisgegevens die kunnen worden opgevraagd van een voertuig.

Attributes

- Status – Status object
  Geeft aan of de call gelukt is of niet.
- MMT – VehicleType object
- VoertuigSoort – string
- Aandrijving – string
- Wielbasis – int
- Kenteken – string
- Koetswerk – string
- Transmissie – string
- NumberOfGears – int?
- Brandstof – string
- Cilinders – int
- Cilinderinhoud – int
- Vermogen – int
- Koppel – int
- Topsnelheid – int
- Acceleratie – decimal
- CO2 – int
- Energielabel – string
- Euroklasse – string
- Verbruik – decimal
- VerbruikBinnen – decimal
- VerbruikBuiten – decimal
- GVW – int

- AutotelexKMStand – int
- NieuwPrijs – decimal
- Wegenbelasting – string
- StandaardOpties – Options object
- Opties – Options object
- Pakketten – Packets object
- AantalDeuren – int
- AantalZitplaatsen – int
- Turbo – bool
- Roetfilter – bool
- Fijnstof – int
- UitvoeringID – int
- GeleverdVan – DateTime?
- GeleverdTot – DateTime?
- ModelVariantGeleverdVan – DateTime?
- ModelVariantGeleverdTot – DateTime?
- SegmentId – int?
- Segment - string
- SegmentDescription – string
- TankInhoud – int
- VerbruikKwh – decimal?
- Actieradius – int?
- Accucapaciteit – int?
- ElectricEngine1Power – int?
- ElectricEngine2Power – int?
- SystemOutput – int?
- MaxChargingSpeedAC – decimal?
- MaxChargingSpeedDC – decimal?
- MinChargingTimeHoursAC – int?
- MinChargingTimeMinutesAC – int?
- MinChargingTimeFromAC – int?
- MinChargingTimeToAC – int?
- MinChargingTimeHoursDC – int?
- MinChargingTimeMinutesDC – int?
- MinChargingTimeFromDC – int?
- MinChargingTimeToDC – int?
- ImportDate – DateTime?

Options

Description

This object contains information about a certain option on the vehicle.

Attributes

- Selected – boolean
  Indicates whether this option is present on the vehicle. Always true for standard equipment.
- Name – String
  The name of the option.
- ID – string
  ID of the option.
- Price – int
  Price of the option.
- ValueAddingOption – boolean
  Indicates if the option increases the value (restwaarde) of the vehicle.
- ManufacturerOptionCodes – List of ManufacturerOption objects
  List of possible option codes and names from the manufacturer that belong with this option. Can be empty since not all options have this data.

ManufacturerOption

Description

Object containing manufacturer information about an option. An option can have multiple possible manufacturer codes and names. In that case, the Selected attribute will indicate whether this manufacturer's option code is selected by the user in PRO.

Attributes

- Code – String
  Manufacturer's code for this option
- Name – String
  Manufacturer's name/description for this option.
- Selected – boolean
  Indicates if this option was manually selected by the user in PRO in case multiple manufacturer codes exist for this option. Only available when vehicle is retrieved using a customertoken.

Packets

Description

Object containing option packet information.

Attributes

- Selected – Boolean
  Indicates if the packet is present on this vehicle.
- Name – String
  Name of the packet.
- ID – String
  ID of the packet.
- Price – int
  Price of the packet.
- ValueAddingOption – Boolean
  Indicates if this packet increases the value (restwaarde) of the vehicle.
- Opties – List of Options object

List of Options that are in this packet.
- ManufacturerCode – String
  Manufacturer code of the packet.
- ManufacturerName – String
  Manufacturer name of the packet.

EigenaarHistorieRDWGegevens

Description

This object contains an array with owner data in chronological order.

Attributes

- Status – Status object
- RDWVoertuigData – VoertuigData object

VoertuigData

Description

This data is identical to the data shown in AutotelexPRO:

RDW-gegevens (raadplegen)

Voertuigdata

Datum	Extra info	Eigenaar	Dagen
26-07-2006	Eerste toelating Datum deel IA / I	Fleetowner	1829
26-07-2011	APK vervaldatum		-
29-07-2011		(erkend) bedrijf	19
17-08-2011		(erkend) bedrijf	12
29-08-2011	Laatste tenaamstelling	(erkend) bedrijf	12

Figuur 1 RDW data in AutotelexPRO

Attributes

- Datum - DateTime
  Date when the described event happened.
- TypeEigenaar - string
  Type of owner.
- ExtraInfo – string
  Additional information about the event (e.g. WOK status update).
- Aantaldagen - string
  The number of days since the previous event.

RDWGegevens

Description

This object contains RDW data.

Attributes

- Status – Status object
- Nieuwprijs - int
  Fiscale waarde RDW (previously "RDW nieuwprijs").
- Berekendeoptiebedrag * – int
  Optiebedrag berekent uit de RDW BPM.
- BPM – int
  RDW BPM (de BPM zoals die bij het RDW bekend is, staat ook op kenteken).
- Afschrijvingspercentage* – decimal
  Afschrijving in een percentage.
- RestBPM* – string
  Rest BPM bij export.
- KentekenStatus* – string
  Status van het kenteken, een van de volgende waarden: "Geldig", "Ongeldig", "Wacht op keuren", "Export", "Sloopvoertuig"
- Kleur* – string
  RDW kleur, een van de RDW kleuren (actueel lijstje met waarden via RDW website te verkrijgen).
- GemiddeldeStatijd* – string
  Deze string beschrijft de gemiddelde statijd. Er komt een zin in de vorm: "93 dagen, op basis van 5 voertuigen"
- RDWZitplaatsen* – int
  Aantal zitplaatsen.
- RDWStaanplaatsen* – int
  Aantal staanplaatsen.
- RDWMaximaleConstructieSnelheid* – string
  Constructiesnelheid zoals bekend bij het RDW in Km/Uur
- RDWEmissiecode* – string
  Emissiecode
- RDWG3Installatie* – bool
  Ja indien het voertuig een gasinstallatie heeft, nee in andere gevallen.
- RDWKleur* – string
  Kleur van het voertuig.
- RDWKleur2* – string
  Secundaire kleur van het voertuig.
- RDWMaximumMassa* – int
  Maximum massa in Kg.

- RDWParallelImport* – bool
  Geeft aan of het een import voertuig betreft of niet.
- RDWInrichtingscode* – string
  Inrichtingscode.
- RDWCo2emissie* – string
- RDWEuroklasse* – string
- RDWEnergielabel* – string
- RDWBrandstofVerbruikStad* – string
- RDWBrandstofVerbruikBuitenweg* – string
- RDWBrandstofVerbruikGecombineerd* – string
- RDWDatumVervalAPK* – DateTime?
- RDWDatumEersteToelatingInternationaal* – DateTime?
- RDWDatumEersteToelatingNationaal* – DateTime?
- RDWDatumAansprakelijkheid* – DateTime?
- RDWMassaLedigVoertuig* – decimal
- RDWLaadvermogen* – decimal
- RDWMassaRijklaar* – decimal
- RDWMassaOngeremd* – decimal
- RDWMassaGeremd* – decimal
- RDWVoertuigClassificatie* – string
- RDWVoertuigClassificatieCode* – int
- RDWLengteVoertuigMax* – int
- RDWBreedteVoertuigMax* – int
- RDWHoogteVoertuigMax *– int
- ApprovalNumber –string
  European Type Approval number
- ApprovalDate – DateTime?
  This value is usually not filled out but can be enabled against additional cost.
- VariantCode – string
  European Type Approval variant code
- TypeCode – string
  European Type Approval type code
- MotorCode* – string
  Motorcode of the vehicle.
- MaximumMassaAutonoomGeremd* – int
  Maximum mass of an autonomous trailer that this vehicle may pull.
- MaximumMassaMiddenasGeremd* – int
  Maximum mass of an middle axle trailer that this vehicle may pull.

- ImportPurchaseValueLicensePlate* – int
  Import value according to license plate data
- ImportPurchaseValueTaxRecord* – int
  Import value according to official tax records table

- ImportPurchaseValueDeviation* – decimal
  Deviation between the above two values
- ImportPurchaseValueDeviationAlert* – boolean
  True or false whether the deviation exceeds a certain amount and should be brought to attention.

* Indicates that the field may or may not be filled out, whenever the data is available the field will have the value but if it is unknown the non-value (Null or empty) it should be handled without problems by the calling application.

VehicleTaxData

Description

This objects contains the valuation data from the Autotelex valuation algorithm. A couple of parameters from the VehicleParameters object will influence the vehicle's valuation. Specifically the following properties:

- vehicle.kenteken
  The license plate will be used to find the vehicle type and derive the consumer price from this specific vehicle
- vehicle.kilometerstand
  This is the mileage of the vehicle used in the valuation.
- vehicle.DatumVerwacht
  This is the date on which the vehicle will be valuated.
- vehicle.isBTWVoertuig
  Indicates if BPM was already paid (consumer) or not (business), this affects the valuation as BPM is a big chunk of the consumer price.
- vehicle.AutotelexUitvoeringID
  If the caller forces the vehicle type to be different from the Autotelex derived type (based on the license plate link from Autotelex), this will affect the consumerprice and subsequently all calculations and extrapolations based off of that.

Attributes

- Status – Status object
- Jato – Jato object
- Restwaarden– Restwaarden object
- DatumTaxatie – DateTime
- Version – int

Jato

Description

This object contains Jato data.

Attributes

- Jatocode – string
  Concatenation of Jato persistent vehicle number and price list date.
- PersistentId – string
  Jato persistent vehicle number.
- SchemaId20602 – string
  Jato data_value from schema 20602: transmission type.
- SchemaId20608 – string
  Jato data_value from schema 20608: automatic clutch (manual).
- SchemaId20622 – string
  Jato data_value from schema 20622: manual sequential.
- SchemaId20624 – string
  Jato data_value from schema 20624: transmission description.
- SchemaId6502 – string
  Jato data_value from schema 6502: drive type.
- SchemaId41601 – string
  Jato data_value from schema 41601: particulate filter system.
- SchemaId605 – string
  Jato data_value from schema 605: local number of doors.
- SchemaId702 – string
  Jato data_value from schema 702: no. seats.
- SchemaId603 – string
  Jato data_value from schema 603: body type.
- SchemaId24102 – string
  Jato data_value from schema 24102: max. permissible total weight (gvw).
- SchemaId6106 – string
  Jato data_value from schema 6106: cargo space height.
- SchemaId5806 – string
  Jato data_value from schema 5806: wheelbase.
- SchemaId604 – string
  Jato data_value from schema 604: cabinetype.

RestWaarden

Description

This object describes a valuation.

Attributes

- Naam - string
  The name indicates what value is contained in this object. “Handelswaarde”, “Inruilwaarde” and “Verkoopwaarde” indicate Autotelex valuations. Other possible values are:
  - “Internetwaarde”
  - “Prijslijstprijs zonder opties RDW” (previously “Consumentenprijs RDW”)
  - “Prijslijstprijs zonder opties Autotelex” (previously “Consumentenprijs Autotelex”)
  - “Prijslijstprijs met opties” (previously “Consumentenprijs incl. opties”)
- Waarde – int
  The value.
- Percentage – double
  Percentage related to the catalog price.
- WaardeExclusief – int
  Valuation excluding BTW/BPM
- BPM – int
  BPM
- BTW – int
  BTW

VehicleType

Description

This object is part of the vehicle description and describes the possible alternative types for a given license type.

Attributes

- Type - String
  Textual representation of the vehicle type.
- Brand - String
  Brand of the vehicle.
- Model - String
  Model of the vehicle.
- ID - Int
  Autotelex ID of the type of the vehicle..
- FotoURLFront - String
  URL to picture of the front of the car.
- FotoURLRear - String
  URLto picture of the rear of the car.

- FotoURLInterior - String
  URL to picture of the interior of the car.

VehicleParameters

The VehicleParameters object has been moved to a separate document ('AutotelexPRO Vehicle Inspection Parameters'). It will be included with this document (either as an appendix or add-on) when you receive it.

ExternalProcessing

Description

An object describing variables intended to indicate what the 3rd party should do with this vehicle.

Attributes

- ActionId - string
  String representing the action that should be performed on this vehicle.
- URL - string
  Reference to an Autotelex supplied webpage that can be used to view the vehicle data.
- VehicleLocation - CustomerData
  Customerdata object describing the current location of the vehicle.
- VehicleDestination – CustomerData
  (currently unused) Object describing the destination for the vehicle in case a transport was requested.

CustomerData

Description

An object describing address information.

Attributes

- Name – string
  Name of the contact person
- Title – string
  Function of the contact person
- StreetNameAndNumber – string
- PostalCode – string
- City – string
- Telephone – string
- Mobile – string

- Fax – string
- Email – string
- Remark – string
- ID – int
  EstablishmentId of the address (vestigingsnummer)
- CompanyName – string
- CompanyStreetNameAndNumber – string
- CompanyPostalCode – string
- CompanyCity – string
- CompanyTelephone – string
- CompanyMobile – string
- CompanyFax – string
- CompanyEmail – string

In general, the address (street, postalcode) for the person and the company will be identical. The company contact information can differ from the person's data.

LicenseplateCheckResult

Description

This object contains validations for several possible license plate checks that can be done.

Attributes

- Status – Status object
  Indicates OK if the data was successfully filled out.
- LicensePlateCheckCodeMatch – bool?
  Indicates if the license plate and check-code are a match. The check-code are the last four digits of the VIN number. This value is unknown if either the 'Kenteken' or 'MeldCode' field were missing in the input.
- LicensePlateValid – bool?
  Indicates if the given string for 'Kenteken' is a valid license plate based on the string formatting only .

UpdateVehicleData

The UpdateVehicleData method uses the exact same signature as the GetVehicleDataPRO method. The difference is, that the user can modify a stored vehicle file using this function. The result will always be the current state of the vehicle file (after the updates).

GetCustomerToken

Description

The customertoken identifies a user within the AutotelexPRO website.

Parameters

The following parameters are required for a successful call:

- companyId
  Company number of the user
- username
  Username to identify the user.
- password
  The password.

Result

An object that will contain the status and a GUID (token). If the status indicates success, the token can be used in subsequent calls.

In general, tokens do not expire, unless Autotelex blocks them. It is safe to store a token on a device, if need be, Autotelex can refresh the token and all calls from devices using the older token will fail.

GetVendorToken Objects

Status

Description

In general, for each methods a status will be returned so the caller can quickly see if the call succeeded or not.

Attributes

- GenericCode
  Generic code indicating the status of the call. See table 1 for possible values.
- Code - Int
  Deprecated, use GenericCode.

- Message - String
  Textual field that explains what went wrong or a comprehensive error message (test environment).

GetBodCriteria

Description

This method returns a list of possible bid partners (coupled to a vendor / user, configured by Autotelex) together with a list of parameters that are required to be filled out before the vehicle can be accepted in the bid process.
Based on a server side setting (vendor setting), this method can return one single list of bid-criteria that is the accumulation of all required criteria for all possible bid partners for the user.

Parameters

The following parameters are required for a successful call:

- vendorToken
  Vendor token
- customerToken
  Customer token.

Result

A BiedingCriteriaResult that contains a status and a list of bidding partners including their requirements for accepting a car into their system.

BiedingCriteriaResult

Description

Describes the list of bidding partners including their requirements for accepting a car into their system.

Attributes

- Status - Status
  Status object indicating the success or failure of the call.
- BiedingCriteriaLijst - List<BiedingCriteriaLijst>
  For each possible bidding partner (configured per customer by Autotelex), this list contains an entry and the corresponding requirements for that bidding partner.

BiedingCriteriaLijst

Description

Describes a bidding partner with the criteria that it uses at the moment (beware: they can change but not very often, requesting these once every time your application starts or when the user logs in should be sufficient).

Attributes

- BiedingSoortID - int
  Unique number to identify the bidding partner.
- BiedingSoort – string
  Descriptive name of the bidding partner.
- Hertaxatie – bool
  Deprecated, do not use.
- KenmerkenLijst – List<BiedingCriterium>
  List of requirements that need to be filled out before the bidding partner will place a serious bid on the car.

BiedingCriterium

Description

Describes a single requirement that needs to be fulfilled before a bidding partner will accept the car into their system.

Attributes

- Omschrijving - string
  Descriptive name.
- KenmerkID – string
  ID of the requirement.
- VoertuigKenmerkID – int
  Vehicle ID of the requirement.
- Verplicht – bool
  Indicates if the requirement must be filled out or not.
- Zichtbaar – bool
  Indicates if the requirement should be shown in the App or not (usually coupled with the Verplicht attribute).
- Percentage – bool
  Indicates if the field value is a percentage or not.
- Relevant – bool
  Deprecated.
- Aantal – int
  Indicates the amount of items expected. E.g. if the Omschrijving is ‘Fotos’, and Aantal is 5 then at least 5 photos are required.

For a list of possible requirements, their names and IDs, see the “Standaard waarden.xlsx” (tab Bod Criteria).

GetHandelaren

Description

This method returns the list of dealers for which the user is allowed to request bids.

Parameters

The following parameters are required for a successful call:

- vendorToken - string
  Vendor token identifying the vendor.
- customerToken - string
  Customer token identifying the user.
- Filter – int
  0 returns all traders, 1 returns the trade partners , 2 returns the local dealers (trade relations) and.

Result

A handelarenResult that contains a status and a list of bidding partners / dealers.

HandelarenResult

Description

Describes the list of bidding partners.

Attributes

- Status - Status
  Status object indicating the success or failure of the call.
- HandelarenLijst - List<Handelaar>
  List of partners/dealers.

Handelaar

Description

Describes a bidding partner.

Attributes

- ID - int
  Unique number to identify the bidding partner.
- Naam – string
  Descriptive name of the bidding partner.
- Bedrijfsnaam – string
  Companyname.
- Telefoonnummer - string
  Telephone number of the bidding partner.
- Mobiel – string
  Mobile number of the bidding partner.

- Email – string
  Email address of the bidding partner
- Opmerking - string
  Room for internal remark that the user could have added to the bidding partner (e.g. ‘Universal dealer’, ‘Only interested in vans’)

AanvragenBod

Description
AanvragenBod will request one or more bids for a certain vehicle.

Parameters
The following parameters are required for a successful call:

- vendorToken
  Vendor token
- customerToken
  Customer token.
- Ibp – InsertBodParameters
  Parameters indicating which bid(s) are to be requested
- version – int
  Version number of the interface. Use 1 as a value for this field.

Result
A BiedingDataResult that contains a status and a list of bidding partners including their requirements for accepting a car into their system.

InsertBodParameters

Description

Describes the bid parameters object that is used in UpdateBod, AanvragenBod, Biedingen etc. Due to its multipurpose use not all fields apply for all calls.

Attributes

- BiedingId – int?
  ID of the bid
- SoortBod – int?
  Type of bid (this ID matches to the BiedingCriteriaLijst.BiedingSoortID)
- Bod – int?
  Bid
- isBTWVoertuig – bool?
  Deprecated.

- URL – string
  Optional URL to bidding partner's website where the user can view the vehicle and see how it arrived in the system.
- GeldigTot – DateTime
  Indicates how long the bid is valid.
- Naam – string
  Name of the bidding partner.
- ExternalID – int?
  Unique identifier of the vehicle file.
- DatumVerwacht – DateTime?
  Date when the vehicle is expected to arrive. Not valid outside of U-Name it system.
- Status – int?
  Status of the bid.
- Opmerking – string
  Remark that was added to the bid.
- HandelarenLijst – List<int>
  List of dealers.
- BestemmingSoortId – int?
  Type of destination of the vehicle (keep intern and intend to sell to end-customer, sell to dealers, sell to auction).
- BestemmingId – int?
  Identifier of the destination
- BestemmingOmschrijving – string
  Description of the destination
- InclExclBTW – bool?
  Indicates if the bid was Including BTW (taxes) or excluding.
- GeenMailVersturen – bool?
  Indicates not to send an email. Not valid outside of Autotelex website/app.
- HoogsteBodIsGeldig – bool?
  Indicates that the highest bid is valid. Not valid outside of Autotelex website/app.

BiedingDataResult

Description

Describes the result of the AanvragenBod call.

Attributes

- Status - Status
  Status of the call.
- BiedingId – string
  ID of the bid that was just requested (in case the status indicates OK).
- vehicles - List<VehicleBiedingData>
  Not used in AanvragenBod call. When calling GetBiedingen this list if filled out – only available/authorized for the Autotelex App.

VehicleBiedingData

Description

Contains the status of a (request for) bid and other related data.

Attributes

- Biedingen – List<BiedingData>
  List of bids
- ExternalID – int
  ID of the car that the bids are valid for.

BiedingData

Description

Contains the status of a (request for) bid and other related data.

Attributes

- BiedingId - int
  ID of the bid
- BedrijfsNaam - string
  Company name of the bidding partner.
- Datum - DateTime
  Date of the bid.
- Waarde - int
  Value.
- Soort - int
  Type of bid
- SoortNaam - string
  Name of type of bid
- Opmerking - string
  Remark made by the bidder.
- Status - int
  Status of the bid (Onbekend, Open, Lopend, Bestemd, Afgehandeld)
- StatusNaam - string
  Name of the status.
- Hertaxatie – bool
  Indicates wether or not a re-valuation is requested. Currently not supported through the webservice.
- InclExlBTW – bool?
  Including or excluding taxes.
- HBH_ID - int
  ID of the car-dealer if the Soort is of type Handelsbod

- GeldigTot - DateTime
  Date and time until the bid is valid.
- TMStatusHistorieLijst
  Deprecated

VehicleIngekocht

Description
Sets the status of the vehicle to 'bought'.

Parameters
The following parameters are required for a successful call:

- vendorToken - string
  Vendor token
- vp - VehicleParameters
  This should have the vp.ExternalId value filled out with the specific vehicle Id that should be set to bought.
- customerToken – string
  This must be the token of the AutotelexPRO user that sets the vehicle to bought.

Result
A GenericResult indicating success or failure.
A failure reason may be because the vehicle Id is invalid.

VehicleNietIngekocht

Description
Sets the status of the vehicle to 'finished' (no further action required as the car is not bought).

Parameters
The following parameters are required for a successful call:

- vendorToken - string
  Vendor token
- vp - VehicleParameters
  This should have the vp.ExternalId value filled out with the specific vehicle Id that should be set to bought.
- customerToken – string
  This must be the token of the AutotelexPRO user that sets the vehicle to bought.

Result
A GenericResult indicating success or failure.
A failure reason may be because the vehicle Id is invalid.

NoInterest

Description

NoInterest indicates that the vendor has no interest in bidding on a particular vehicle. The bid request will be manipulated to the values indicating this.

Parameters

The following parameters are required for a successful call:

- vendorToken
  Vendor token
- vehicleId
  Id of the vehicle.

Result

A GenericResult indicating success or failure.
A failure reason may be because the vehicle Id is invalid or if there was no request issued to the vendor for this particular vehicle.

Uploading and manipulating images

This section describes how to upload an image using a HTTP POST of a binary file to a specific URL supplied by the AutotelexPRO webservice.

As .NET CORE does not support MTOM webservices anymore, this is an alternative to adding images to a vehicle file. The MTOM webservice mechanism described in earlier versions of this document will remain available until further notice.

How to upload an image

Uploading an image is a two step process. First, the caller retrieves a URL from the API Webservice, and if that is received successfully the caller can post an image in its binary for to the URL.

The URL needs to be retrieved for each image that the caller wants to upload, it can not be re-used.

UploadFile

Description

This method will generate and return the URL where an image can be posted to. The UploadFile() needs a vendor and customertoken to identify the caller and the person on behalf the vendor is calling. To describe the image, an object called up is used to indicate what type of image is being sent and to which vehicle file it should be linked.

Parameters

The following parameters are required for a successful call:

- vendorToken – string
  Token to identify 3rd party.
- customerToken – string
  Token to identify the user
- up – UploadFileParameters
  Object describing the file and the requested changes
- version – int
  Version of the call, unused.

Manipulating uploaded images

Once an image is uploaded, it will be marked as a ‘vehicle’ image by default. The other option, is that the image is a ‘damage’ image, indicating that there is damage visible on the image.

The method to change the type of picture is called UpdateFile().

UpdateFile

Description

The Upload() accepts an object that includes a stream, the result of this method is a string.

Parameters

The following parameters are required for a successful call:

- vendorToken – string
  Token to identify 3rd party.
- customerToken – string
  Token to identify the user
- up – UploadFileParameters
  Object describing the file and the requested changes
- version – int
  Version of the call, unused.

Result
An UploadFileResult object indicating success or failure of the operation.

UploadFileParameters

Description

Describes the parameters for an uploaded file. Most important (required) is the vehicleID, this will be used to link the image to an existing vehicle file.

Attributes

- Aanzicht – int
  Side of the vehicle that is in the picture.
- AanzichtID – string
  Id of side of the vehicle that is in the picture
- **VehicleID** - string
  This is the unique AutotelexPRO id of the vehicle file that you received when creating the vehicle (or retrieved from the list of vehicles).
- AdvertentieID – string
  Deprecated
- Kenteken – string
  Licenseplate of the vehicle.
- URL – string
  URL to the image (if known, as supplied by Autotelex), this is used as identifier in calls to update the image.
- FotoTag – int?
  Nullable integer to indicate what type of image this is. Valid values are 5 to indicate a vehicle image, 2 to indicate a damage image.
- FileName – string
  Filename of the image.
- Version – int
  Unused.

UploadFileResult

Description

Object returned from UploadFile() call to indicate success or failure and the URL where the caller can post the image to.

Attributes

- URL – string
  URL to post the image to.
- Status – Status object
  Status object, if the status is OK, the URL will contain a valid endpoint to send the image to

Connecting damage images to specific damage items

Once an image has been uploaded, it can be set to be a damage image (general damages, unconnected to a specific damage item). But the image can also be connected to a specific damage item after it has been set to the damage type.

This can be done by supplying the specific damage object (SchadeOmschrijving) in the UpdateVehicle with a list of the URLs to the images in the field SchadefotoURLs. Currently a maximum of 2 images per damage is supported. Make sure the upload succeeded and the image has been set to type Damage using the UpdateFile() call before updating.

Raw request example for uploading an image

Once the URL has been retrieved, a binary image is expected to be sent to the URL with a HTTP POST call. Here we present an example of such a call:

```
POST https://test.autotelexpro.nl/PostFileHandler.ashx?prm=TcjLFgi/KH+k+8dZz40 HTTP/1.1
Content-Type: multipart/form-data; boundary=----------8d3809e393c5442
Host: test.autotelexpro.nl
Content-Length: 75282
Expect: 100-continue

----------8d3809e393c5442
Content-Disposition: form-data; name="file"; filename="HG075R_1.jpg"
Content-Type: application/octet-stream
ÿØÿà JFIF      ÿþ ;CREATOR: gd-jpeg v1.0 (using IJG JPEG v80), quality = 90
.
```

.
.
----------8d3809e915ad401--

The result should be a HTTP 200, indicating success and the response will give some information about the uploaded image in the following format (in this case, the upload went OK but the image was identical to another one uploaded for that vehicle file so it was not added, instead the URL of the existing image was returned):

HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/8.5
Set-Cookie: AutotelexPro=2uct21txcmoznlvug012nnlj; path=/; HttpOnly
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: POST
Access-Control-Allow-Headers: X-Requested-With,authorization,cache-control
Date: Fri, 20 May 2016 09:04:51 GMT
Content-Length: 297

File 1 uploaded and successfully received.
Filename: HG075R_1.jpg
Length: 75086
Content-type: multipart/form-data; boundary=----------8d3809e915ad401
URL: https://autotelexpro.nl/Data/Productie/1/images/image_fd4JFm7aJkZtph3w9S0ig.jpg?timestamp=1463742279
Error: duplicate image.

InsertSignature

Description

Inserts signature for specific context.

Parameters

- vendorToken – String
  Token to identify 3rd party.
- customerToken – String
  Token to identify the user.
- sp – SignatureParameters
  Object describing the signature and it's context.

Result

The result is a GenericResult object, containing the status of this call.

InsertSignatue Objects

This chapter describes the objects and fields involved in the InsertSignature call.

GenericResult Object

Description

Generic result object containing result information.

Attributes

- Status – Status object
  Indicates the status of the call.

Status Object

Description

Status object indicating success or failure.

Attributes

- GenericCode – int
  Generic code indicating the status of the call. See table 1 for possible values.
- Code – int
  Deprecated, use GenericCode.
- Message – string
  Textual field that explains what went wrong or a comprehensive error message (test environment).

SignatureParameters Object

Description

Describes the parameters for a signature to insert.

Attributes

- SignatureOwnerType – SignatureOwnerType
  Describes the ownertype of the signature. Customer = 1, Appraiser = 2.
- SignatureEvent – SignatureEvent
  Describes the event of the signature. VehicleInspection = 1, VehicleIntake = 2.
- VehicleStateId – int
  The ID of the vehicle.
- Signature – string
  The signature itself in base64 format.

DeleteSignature

Description

Deletes signature for specific context.

Parameters

- vendorToken – String
  Token to identify 3rd party.
- customerToken – String
  Token to identify the user.
- sp – SignatureParameters
  Object describing the signature and it's context.

Result

The result is a GenericResult object, containing the status of this call.

InsertSignatue Objects

This chapter describes the objects and fields involved in the InsertSignature call.

GenericResult Object

Description

Generic result object containing result information.

Attributes

- Status – Status object
  Indicates the status of the call.

Status Object

Description

Status object indicating success or failure.

Attributes

- GenericCode – int
  Generic code indicating the status of the call. See table 1 for possible values.
- Code – int
  Deprecated, use GenericCode.
- Message – string
  Textual field that explains what went wrong or a comprehensive error message (test environment).

SignatureParameters Object

Description

Describes the parameters a signature to delete.

Attributes

- SignatureOwnerType – SignatureOwnerType
  Describes the ownertype of the signature. Customer = 1, Appraiser = 2.
- SignatureEvent – SignatureEvent
  Describes the event of the signature. VehicleInspection = 1, VehicleIntake = 2.
- VehicleStateId – int
  The ID of the vehicle.
- Signature – string
  The signature itself in base64 format.

TCOResult Object

Description

Describes the TCO result if it is available for the account. The cost is calculated for the requested period or for the default period determined by Autodisk if no period was indicated.

Attributes

- DepreciationAndInterest
  The amount of money that the vehicle was depreciated and the interest that was paid for.
- RepairAndMaintenance – float
  Total cost of repairs and maintenance
- Tyres – float
  Total cost of tyres
- MRB – float
  Total MRB cost
- TCOTotalExcludingFuel – float
  Total TCO excluding cost for fuel
- Fuel – float
  Total cost of fuel
- TCOTotalIncludingFuel – float
  Total cost including fuel
- Insurance
  Total cost of insurance
- InputParameters – TCOParameters object
  Input fields for TCO result, see Inspection Parameters for full description of that object.

Environments

This chapter contains the information to connect to the webservice.

Test environment

The test environment can be approached through the following link:

Webservice location:

https://apitest.autotelexpro.nl/autotelexproapi.svc

Note, depending on the used protocol it could be necessary to add the following to the URL:

https://apitest.autotelexpro.nl/autotelexproapi.svc/xml – for XML based applications.

Note: the direct link to the WSDL is:

https://apitest.autotelexpro.nl/autotelexproapi.svc?wsdl

For testing, a test account is necessary, the account credentials do not have to match those for the production environment. Test accounts are automatically invalidated after two months, a typical implementation will take much less time than two months to complete (2-3 weeks on average).

Production environment

The production environment can be accessed through the following URLs:

Webservice location:

https://api.autotelexpro.nl/autotelexproapi.svc

Note, depending on the used protocol it could be necessary to add the following to the URL:

https://api.autotelexpro.nl/autotelexproapi.svc/xml - for XML based applications.

Note: the direct link to the WSDL is:

https://api.autotelexpro.nl/autotelexproapi.svc?wsdl

Changes

Autotelex can change the content of a request- and/or response message. Should this have any consequences for the usage of the webservice, we shall notify you, the user, about this change if it might concern breaking changes.

Constraints / batch processes

The calls to the webservice are actively monitored. When inappropriate use is detected or suspected, Autotelex can shut down the webservice for a specific user.

If you require to call this webservice with a high frequency, please contact your account manager due to restrictions on the maximum number of calls per user in a specific interval.

Calling the webservice using JSON instead of SOAP XML

The webservice can be called with JSON as input/output, this requires connecting to a different endpoint and of course the formatting of the parameters should be in JSON instead of XML.

The JSON messages are identical to the XML messages in terms of content, the WSDL can be used to determine how messages should look like (unfortunately there is currently no OpenAPI / Swagger documentation available). This makes using the JSON endpoint a little cumbersome and together with JSON's other disadvantages will probably take some more development effort to get it working.

For the JSON variant, the /XML in the endpoint can be replaced with /JSON/[**methodname**]
The global endpoint URL should thus be:

https://api.autotelexpro.nl/autotelexproapijson.svc/json

An example of the URL for the call to GetVehicleDataPRO looks like this:

https://api.autotelexpro.nl/autotelexproapijson.svc/json/GetVehicleDataPRO

All the JSON calls should be POST requests with the parameters in the body of the request. An example of the call to GetVehicleDataPRO would look like this:

```
POST http://api.autotelexpro.nl/autotelexproapijson.svc/json/GetVehicleDataPRO HTTP/1.1
Content-Type: application/json; charset=utf-8
Host: api.autotelexpro.nl
Connection: close
User-Agent: Paw/3.1.7 (Macintosh; OS X/10.13.6) GCDHTTPRequest
Content-Length: 80

{"token":"your-own-vendor-token","vehicle":{"kenteken":"10xnk1"}}
```