Datatrack 247 - API

Version 5.5

May 18, 2022

Table of Contents

1. Setup

In order to access this service you will need to have an account with a Positioning Universal service (SVR, GCF or Positioning Universal). This API will give you access to all of the installed vehicles and AI cameras in the account.

All time values are Unix time based seconds since 1970. Click here for more details.
No dates are allowed to be specified more than 1 month in the past.

2. Servers

The API works through HTTPS GET requests being sent to the base url of:

Server	**Base URL**
Datatrack 247	https://fleet77.com/fleet4

The connection is protected by https (TLS) so it is safe from attack by third parties who attempt to intercept the communication.

3. Authentication

Each call to the API must be authenticated. You will need to set the Authorization header of each request with the API key we will provide you.

The API key will be in the form of **prefix.key**

When calling the API, you will send the following in the Authorization header of your request:

ApiKey prefix.key

For example:

ApiKey b8h88206CD.4f4f7007G704547d7f66 (this is a fake key for example purposes)

In this example, the prefix is b8a88206ce and the key is 4f4f7005d704547d7d66 (the two are separated by a period).

The prefix can be used to refer to the API key in the future. We never store the key in plain text.

You should store your API key in a safe place.

4. Methods

4.1 Locations

A GET call to this method will return up to 5000 locations for a specific vehicle. If you receive 5000 locations in response to a query you should expect that more locations are available than were returned and you need another query starting at the date of the last received location in order to get the missing locations.

The url has an endpoint of getLocations and is built using the following parameters:
- serial - the serial number of the vehicle to be queried
- start - UTC based unix time of date to start retrieving locations
- end - UTC based unix time of date to stop retrieving locations

Example url:

https://gpstrackservices.com/svr3/getLocations?serial=1234567890&start=1407945984&end=1407949568

The HTTP error codes for a request are listed in section: Error Codes

The request returns a list of locations encoded in JSON in the following format:

Name	**Example**	**Description**
date	1396038924	UTC based unix time in seconds
typeId	0	The reason for location report see table at Type Ids for description
lat	-23.6044613	The degrees North +, South - with 7 digits of precision
lng	-46.6925758	The degrees West -, East + with 7 digits of precision
speed	23	In kilometers per hour
status	3	The following values are added together GOOD_GPS = 1;

		GOOD_COMM = 2;
		OLD_FIX = 8;
		So a good gps, good comm location has value 3. A bad gps, good comm location has a value 2 etc.. A fix is considered old if it reported a time more than 1 hour older than server time when it is received. GOOD_COMM means an RSSI of over -90 and GOOD_GPS means more than 4 satellites and an HDOP less than 15.

voltage	13.355	Represents attached power supply voltage in Volts
heading	120	Direction of vehicle, if available
hdop	20	HDOP of GPS receiver when fix calculated, if available.
sats	6	# of satellites used in fix, if available

Example of returned JSON array (Array will be [] if no values are found):

```
{
    "status": 200,
    "data": [
        {
{
            "date": 1462309217,
            "typeId": 5,
            "speed": 91,
            "lat": 34.915946,
            "lng": -84.945603,
            "status": 0,
            "inputs": -2,
            "voltage": 13.99,
            "heading": 5,
            "hdop": 20,
            "sats": 6
        }...
    ]
}
```

4.2 Vehicle Statuses

4.2.1 All Vehicle Statuses

A GET call to this method will return the status of all vehicles in account.

The url has an endpoint of getStatuses.

Example url:

https://gpstrackservices.com/svr3/getStatuses

The request returns a list of statuses encoded in JSON in the following format:

Name	**Example**	**Description**
serial	1234567890	Serial number of device
date	1396038924	UTC based unix time in seconds
typeId	5	The reason for location report see table at Type Ids for description
speed	13	In kilometers per hour
disabled	0	UTC based unix time in seconds when vehicle had starter disabled. 0 is returned if starter is enabled
buzzer	0	UTC based unix time in seconds when a vehicle had its buzzer enabled. 0 if no buzzer is enabled.
lat	-23.6044613	The degrees North +, South - with 7 digits of precision
lng	-46.6925758	The degrees West -, East + with 7 digits of precision
distance	1000.123	Km the vehicle has travelled
voltage	13.355	Represents attached power supply voltage in Volts
name	Jen’s Car	Vehicle name

Example of returned JSON array (Array will be [] if no values are found):

```
{ status: 200,
  data: [
     {
        "serial": "A100004950047D",
        "date": 1467912839,
        "typeId": 3,
        "speed": 0,
        "disabled": 0,
        "lat": 35.030377,
        "lng": -85.154427,
        "buzzer": 0,
        "distance": 184702,
        "volts": 12820,
         "name": "ZBob V"
     }...

  ]
}
```

4.2.2 All Vehicle Statuses Min

A GET call to this method will return the status of all vehicles in account. However this call returns fewer fields than the All Vehicle Statuses call in 2.1 above.

The url has an endpoint of getStatusesMin.

Example url:

https://gpstrackservices.com/svr3/getStatusesMin

The request returns a list of statuses encoded in JSON in the following format:

Name	Example	Description
serial	1234567890	Serial number of device
date	1396038924	UTC based unix time in seconds
typeId	5	The reason for location report see table at Type Ids for description
distance	1000.123	Km the vehicle has travelled
volts	13.355	Represents attached power supply voltage in Volts
vin	JT8BF28G610337020	VIN of the vehicle device is installed in

name	Jen’s Car	Vehicle name

Example of returned JSON array (Array will be [] if no values are found):

```
{ status: 200,
  data: [
     {
        "serial": "A100004950047D",
        "date": 1467912839,
        "typeId": 3,
        "distance": 184702,
        "volts": 12820,
         "vin": "JT8BF28G610337020",
         "name": "ZBob V",
      }...

  ]
}
```

4.2.3 Single Vehicle Status

A GET call to this method will return the status of a single vehicle. It is similar in all respects to the method to get all statuses except it returns a single value. This method is provided so that a single device status can be polled after a starter disable request is sent to the device. It can take up to a minute for the vehicle to move into a disabled state due to latency in the cellular network.

The url has an endpoint of getStatus and is built using the following parameter:

- serial - the serial number of the vehicle to be mapped

Example url:

https://gpstrackservices.com/svr3/getStatus?serial=1234567890

Example of returned JSON array (status = 448 if no status found for vehicle):

```
{
    "status": 200,
    "data": {
        "serial": "1332058433",
        "date": 1437138676,
        "typeId": 3,
        "speed": 0,
        "disabled": 0,
        "lat": 33.4008179,
        "lng": -111.4851731,
        "buzzer": 0,
        "distance": 0,
        "volts": 0,
        "name": "Wolfgang's HVAC"
    }
}
```

4.3 Top Stops

A GET call to this method returns the top stops for a single vehicle. Top Stops represent the time the vehicle spent stopped the longest.

The url has an endpoint of getTopStops and is built using the following parameters:

- serial - the serial number of the vehicle to be mapped
- period - the time period to evaluate for top stops. 1 - 60 days is allowed range.
- number - number of top stops to compute. 1-10 stops is allowed range.

Example url:

https://gpstrackservices.com/svr3/getTopStops?serial=1234567890&period=30&number=5

The request returns a list of locations encoded in JSON in the following format (sorted by duration):

Name	**Example**	**Description**
duration	13960	seconds vehicle has spent at this location
total	12	number of actual stops at this location
address	4080 Jenkins Rd, Chattanooga, TN	stop address
lat	-23.6044613	The degrees North +, South - with 7 digits of precision

lng	-46.6925758	The degrees West -, East + with 7 digits of precision

Example of returned JSON (data will be [] if no values are found):

```
{  "status": 200,
   "data": [
        {
            "duration": 1791213,
            "total": 17,
            "address": "3295 Keith Rd, Ringgold, GA",
            "lat": 34.941821,
            "lng": -85.0316516
        },
        {
            "duration": 144399,
            "total": 3,
            "address": "10615-10699 Ned Ct, Charlotte, NC",
            "lat": 35.1178275,
            "lng": -80.9547222
        },
        {
            "duration": 60905,
            "total": 1,
            "address": "1001 Gould Dr, Bossier City, LA",
            "lat": 32.5189821,
            "lng": -93.7096275
        },
        {
            "duration": 58410,
            "total": 1,
            "address": "1611 Galleria Blvd, Brentwood, TN",
            "lat": 35.9698045,
            "lng": -86.8096769
        },
        {
            "duration": 48034,
            "total": 1,
            "address": "429 W Pearl St, Jackson, MS",
            "lat": 32.2995296,
            "lng": -90.1934981
        }
```

```
      ]
}
```

4.4 Starter Disable

The GET call to this method sends a message to a vehicle to either enable or disable the vehicle’s starter. This method simply sends the request, the vehicle status must be polled in order to ensure that the message was received and the vehicle has entered the desired state.

The url has an endpoint of setStarter and is built using the following parameters:

- lserial - the serial number of the vehicle to be affected
- disable - set to “true” to activate starter disable and “false” to deactivate starter disable

Example url:

https://gpstrackservices.com/svr3/setStarter?serial=1234567890&disable=true

Returns JSON `{ "status": 200}` if successful. Otherwise, it returns the appropriate error code in the status field.

4.5 Buzzer

The GET call to this method sends a message to a vehicle to activate the vehicle’s buzzer. The buzzer is activated on ignition on or on the first movement of the vehicle in a trip if no ignition is enabled. The buzzer in the device will sound 3 times on the first day. Then 2 times the second day and finally 1 time on the last day before turning off.

This method returns a status code of 200 if a message is sent to the vehicle.

The url has and endpoint of setStarter and is built using the following parameter:

- disable - set to “false” to activate buzzer and “true” to deactivate buzzer

Example url:

https://gpstrackservices.com/svr3/setBuzzer?serial=1234567890&disable=true

Returns JSON `{ "status": 200}` if successful. Otherwise, it returns the appropriate error code in the status field.

4.6 Alerts

A GET call to this method will return the alert history of a single vehicle or all vehicles in an account.

The url has an endpoint of getAlerts and is built using the following parameters:

- startSecs - UTC based unix time of date (in seconds) to start retrieving alerts, must be within one week of endSecs
- endSecs - UTC based unix time of date (in seconds) to stop retrieving alerts, must be within one week of startSecs
- serial - (optional) the serial number of the vehicle, if it is not provided alerts for the whole account will be returned
- alertTypeId - (optional) the value of the alert type you wish to retrieve, if not provided all alert types will be returned. Please see alert types at Alert Types.

Example url:

https://gpstrackservices.com/svr3/getAlerts?serial=1234567890&startSecs=1407945984&endSecs=1407949568&alertTypeId=2

The request returns a list of alerts encoded in JSON in the following format:

Name	Value	Description
accountId	123456789	The unique id associated with the account
vehicleId	1332058433	The unique id associated with the vehicle
driverId	0	The unique id associated with the driver
date	1622572135	UTC based unix time in seconds
alertId	-1996406265	The unique id associated with the alert
alertTypeId	2	Each alert type has an id, listed in Alert Types
accel	0	Acceleration in cm^2
distance	16471096	Distance in meters

engineSecs	0	Cumulative total seconds engine has been on
inputs	0	Current input status (use byte masks for individual inputs)
lat	368544730	The degrees North +, South - multiplied by 10000000
lng	-1197292560	The degrees West -, East + multiplied by 10000000
rssi	0	Relative Received Signal Strength for the modem in dbm
speed	0	Speed in kph
typeId	5	For acceleration alert: ● 0-all ● 1-acceleration ● 2-braking ● 3-hard left ● 4-hard right For maintenance alert: ● 0-distance ● 1-engine hours
voltage	0	Voltage * 1000
confirmationStatus	0	For the driver to confirm an incident alert 0-unknown, 1-yes, 2-no
confirmationDate	1622572135	UTC based unix time in seconds of when the driver confirmed the alert, if they did so
name	Idle Alert	The user-entered name of the alert
address	1375 N Willow Ave, Clovis, CA 93619	Address where the alert occured
cameraEventId	0	The unique id associated with each camera event, 0 if there is no associated camera event

Example of returned JSON array (status = 447 if startSecs to endSecs period is more than a week):

```
{
      "status":200,
      "data":[
            {"accountId":123456789,
            "vehicleId":1332058433,
            "driverId":0,
            "date":1622572135,
            "alertId":-1996406265,
            "alertTypeId":2,
            "accel":0,
            "distance":16471096,
            "engineSecs":0,
            "inputs":0,
            "lat":368544730,
            "lng":-1197292560,
            "rssi":0,
            "speed":0,
            "status":0,
            "typeId":5,
            "voltage":0,
            "confirmationStatus":0,
            "confirmationDate":0,
            "name":"Idle Alert",
            "address":"1375 N Willow Ave, Clovis, CA 93619",
            "cameraEventId":0},
            {...
            ]
      }
```

4.7 User Actions

A GET call to this method will return the user action history of a single user or an account.

The url has an endpoint of getUserActions and is built using the following parameters:

- startSecs - UTC based unix time of date to start retrieving user history, must be within one week of endSecs
- endSecs - UTC based unix time of date to stop retrieving user history, must be within one week of startSecs
- typeId - (optional) the value of the type of action you wish to retrieve, if not provided all

action types will be returned. Please see user action types at User Action Types
- userLogin - (optional) the login of the user whose history you wish to retrieve, if not provided all account users will be returned.

Example url:

https://gpstrackservices.com/svr3/getUserActions?startSecs=1407945984&endSecs=1407949568&typeId=109&userLogin=tester

The request returns a list of alerts encoded in JSON in the following format:

Name	**Value**	**Description**
accountId	123456789	The unique id associated with the account
date	1622572135	UTC based unix time in seconds
typeId	109	Each type has an id, listed in User Action Types
className	Vehicle	The class of the object on which the action has been done
newObjectName	XXXXX	The name of the object on which the action has been done (ex. Vehicle name or driver name)
userLogin	tester	The login associated with the user record
name	Tester User	Name of the user, entered in the UI
userTypeId	0	Id associated with the user type: ● 0-UI User ● 1-Admin User ● 2-UI Recipient ● 3-External User

Example of returned JSON array (status = 447 if startSecs to endSecs period is more than a week):

```
{
  "status":200,
  "data":[
    {
       "accountId":123456789,
       "date":1622754233,
       "typeId":109,
       "className":"Vehicle",
       "newObjectName":"XXXXX",
       "userLogin":"tester",
       "name":"Tester User",
       "userTypeId":0
     },
     {...
   ]
}
```

4.8 OBD Events

A GET call to this method will return the OBD Event history of a single device

The url has an endpoint of getObdEvents and is built using the following parameters:

- serial – The serial of the device you wish to receive events from
- startSecs – (optional) UTC based unix time of date to start retrieving OBD events. If excluded startSecs will be zero.
- endSecs – (optional) UTC based unix time of date to stop retrieving OBD events. If excluded endSecs will be the time of the call.
- limit – (optional) The number of records to retrieve. If excluded there will be no limit.
- direction – (optional) true or false. If false, the records will be sorted newest to oldest. If true, the records will be sorted oldest to newest. If excluded direction is false.

Example url:

https://gpstrackservices.com/svr3/getObdEvents?serial=1234567890&startSecs=1594591955&endSecs=1626987072&limit=1&direction=true

The request returns a list of obd events encoded in JSON in the following format:

Name	**Value**	**Description**
date	1622572135	UTC based unix time in seconds
serial	1234567890	The serial number of the device

celStatus	0	The status of the check engine light • 0 - Unknown • 1 - On • 2 - Off
dtcList	[…]	A list of diagnostic trouble codes
dtcList-date	1622572135	UTC based unix time in seconds
dtcList-status	1	The status of the diagnostic trouble code • 0 - Unknown • 1 – On • 2 - Off
dtcList-code	P0200	The DTC, a standardized set of codes used to diagnose vehicles and heavy machinery. Also called an OBD-II code. The codes can be looked up here: https://repairpal.com/obd-ii-code-chart
dtcList-description	Injector Circuit Malfunction	An English description of the DTC

Example of returned JSON array (status = 445 if no device can be found matching the serial provided)

```
{
  "status":200,
  "data":[
    {
        "date":1599780925,
        "serial":"1234567890",
        "celStatus":0,
        "dtcList":[
          {
            "date":1599780925,
            "status":1,
            "code":"P0100",
            "description":"Mass or Volume Air flow Circuit Malfunction"
          },
          {
            "date":1599780925,
            "status":1,
            "code":"P0200",
            "description":"Injector Circuit Malfunction"
           }
         ]
       },
       {...
    ]
}
```

4.9 Vehicles

4.9.1 Single Vehicle

A GET call to this method will return one Vehicle's data

The url has an endpoint of getVehicle and is built using the following parameter:

- serial – The serial of the device installed in the vehicle you wish to receive data about

Example url:

https://gpstrackservices.com/svr3/getVehicle?serial=1234567890

The request returns vehicle data encoded in JSON in the following format:

Name	**Value**	**Description**
vin	1NXBB02E8VZ590293	The VIN of the vehicle, set by the user.
plate	SVJ506	The license plate of the vehicle, set by the user
make	Toyota	The make of the vehicle, set by the user.
model	Corolla	The model of the vehicle, set by the user.
odometerOffset	-431	A value that represents the difference between the vehicle's odometer reading and how many kilometers the device has reported. The value can be negative if a device is transferred to a new vehicle. Units are in kilometers.

notes	Bob’s second work car	Notes about the vehicle, set by the user.
name	Bob’s Toyota Corolla	Name of the vehicle, set by the user.
year	1997	The year of the vehicle, set by the user.
vehicleColor	0	A value between 0 and 10 representing the color of the vehicle, set by the user. • 0 – Unknown • 1 – Red • 2 – Black • 3 – White • 4 – Blue • 5 – Gray • 6 – Orange • 7 – Yellow • 8 – Green • 9 – Silver • 10 – Other
alternateName	Red Car	This field is called alias in the UI, an alternate name specified by the user.
serial	1234567890	The serial number of the device installed in the vehicle.

Example of returned JSON array (status = 445 if no device can be found matching the serial provided)

```
{
  "status":200,
  "data":
      {
        "vin":"",
        "plate":"",
        "make":"",
        "model":"",
        "odometerOffset":-431,
        "notes":"",
        "name":"1234567890",
        "year":0,
        "vehicleColor":0,
        "alternateName":"",
        "serial":"1234567890"
      }
}
```

*To determine the vehicle's actual mileage (in km), the following formula can be applied:

mileage = odometerOffset + (totalDistance / 1000)

To convert to miles, divide the mileage by 1.609.

**totalDistance can be retrieved using the getStatus call.

4.9.1 All Vehicles

A GET call to this method will return all the vehicles in the account hierarchy of your account or all the vehicles of a specific account in your hierarchy

The url has an endpoint of getVehicles and is built using the following parameters:

- accountId – (optional) returns the vehicles in the account, if the account is a distributor account it returns the vehicles in the account's hierarchy
- startSecs – (optional) only returns vehicles that have been modified after startSecs

Example url:

https://gpstrackservices.com/svr3/getVehicles?accountId=123456789&startSecs=1650665924

The request returns vehicle data encoded in JSON in the following format:

Name	**Value**	**Description**
accountId	123456789	The accountId of the account the vehicle is in
vin	1NXBB02E8VZ590293	The VIN of the vehicle, set by the user.
plate	SVJ506	The license plate of the vehicle, set by the user
make	Toyota	The make of the vehicle, set by the user.
model	Corolla	The model of the vehicle, set by the user.
odometerOffset	-431	A value that represents the difference between the vehicle's odometer reading

		and how many kilometers the device has reported. The value can be negative if a device is transferred to a new vehicle. Units are in kilometers.
notes	Bob’s second work car	Notes about the vehicle, set by the user.
name	Bob’s Toyota Corolla	Name of the vehicle, set by the user.
year	1997	The year of the vehicle, set by the user.
vehicleColor	0	A value between 0 and 10 representing the color of the vehicle, set by the user. • 0 – Unknown • 1 – Red • 2 – Black • 3 – White • 4 – Blue • 5 – Gray • 6 – Orange • 7 – Yellow • 8 – Green • 9 – Silver • 10 – Other
alternateName	Red Car	This field is called alias in the UI, an alternate name specified by the user.
serial	1234567890	The serial number of the device installed in the vehicle.

Example of returned JSON array (status = 445 if no device can be found matching the serial provided)

```
{
  "status":200,
  "data":
      [{
        "accountId":"123456789",
        "vin":"",
        "plate":"",
        "make":"",
        "model":"",
        "odometerOffset":-431,
        "notes":"",
        "name":"1234567890",
        "year":0,
        "vehicleColor":0,
        "alternateName":"",
        "serial":"1234567890"
      },
        {"accountId":"123456789",…
   }]
}
```
*To determine the vehicle's actual mileage (in km), the following formula can be applied:

mileage = odometerOffset + (totalDistance / 1000)

To convert to miles, divide the mileage by 1.609.

**totalDistance can be retrieved using the getStatus call.

5. AI Camera Methods

5.1 Get Media List

A GET call to this method returns the list of camera media (events and on demand) for a given camera serial and for a given timeframe (maximum 1 week). The on demand media being images and video that you have already requested to be uploaded to the cloud previously.

Note - this method is not intended to be used as your main way of discovering events. Accounts not using our backend tracking should subscribe to our push service to be made aware of events as they happen.

The url has an endpoint of getMediaList and is built using the following parameters:
- cameraSerial - the serial number for the camera (not the tracking device)
- startSecs - UTC based unix time of date to start retrieving camera events, must be within one week of endSecs
- endSecs - UTC based unix time of date to stop retrieving camera events, must be within one week of startSecs

Example url:

https://gpstrackservices.com/svr3/getMediaList?cameraSerial=ABC123&startSecs=1407945984&endSecs=1407949568

The request returns a list of camera events encoded in JSON in the following format:

Name	Value	Description
serial	ABC123	The serial number for the camera
eventId	1234556	The id for the event
eventTypeId	111	Each event type has an id, listed in AI Camera Event Types
date	1637544707	The time the camera event occurred. UTC based unix time in seconds.
lat	-23.6044613	The degrees North +, South - with 7 digits of precision
lng	-46.6925758	The degrees West -, East + with 7 digits of precision

Example of returned JSON array (array will be [] if no media found):

```
{
  "status":200,
  "data":[
    {
       "serial":"ABC123",
       "eventId":123556,
       "eventTypeId":"111",
       "date":1622754233,
       "lat":-23.6044613,
       "lng":-46.6925758
     },
     {...
   ]
}
```

5.2 Get Media Data

A GET call to this method returns the secure display links for a given camera event (event or on demand image or video). The on demand media being images and video that you have already requested to be uploaded to the cloud previously.

The url has an endpoint of getMediaData and is built using the following parameter:

- eventId - The id for the camera event

Example url:

https://gpstrackservices.com/svr3/getMediaData?eventId=123456

The request returns the information for the requested event encoded in JSON in the following format:

Name	**Value**	**Description**
serial	ABC123	The serial number for the camera
eventId	123456	The eventId for an event we pushed to you (webhook) or via the above GetMediaList call.
eventTypeId	111	Each event type has an id, listed in [AI Camera Event Types]
date	1637544707	The time the camera event occurred. UTC based unix time in seconds.
mediaList	List of image or video files	List of image or video files depending on eventTypeId. If there are two cameras in the vehicle, there will be two links - one for each camera's video/image. Note the the media files will have plkconnected-uswest.s3.us-west-1.amazonaws.com as their address (AWS) followed by a long string of AWS signatures for security (see below example.

Example of returned JSON object:

```
{
  "status":200,
  "data":{
        "serial":"ABC123",
        "eventId":123556,
        "eventTypeId":"111",
        "date":1622754233,
        "mediaList": "https://plkconnected-uswest.s3.us-west-
```

```
      1.amazonaws.com/fmsvideo/ .../ABC123/….mp4?X-Amz-
      Algorithm=...,https://plkconnected-uswest.s3.us-west-
      1.amazonaws.com/fmsvideo/../ABC123/...mp4?X-Amz-Algorithm=..."
   }
}
```

5.3 Get On Demand List

A GET call to this method returns the list of camera videos and images that are available to download for a given camera serial and for a given search timeframe (maximum 4 hour search).

The url has an endpoint of getOnDemandList and is built using the following parameters:

- cameraSerial - the serial number for the camera (not the tracking device)
- startSecs - UTC based unix time of date to start searching, must be within 4 hours of endSecs
- endSecs - UTC based unix time of date to stop searching, must be within 4 hours of startSecs

Example url:

https://gpstrackservices.com/svr3/getOnDemandList?cameraSerial=ABC123&startSecs=1407945984&endSecs=1407949568

The request returns a list of camera media encoded in JSON in the following format:

For videos:

Name	**Value**	**Description**
serial	ABC123	The serial number for the camera
type	video	This record is for an available video
key	20211122_013147	The name of the video file
date	1637544707	The time the video occurred. UTC based unix time in seconds.
lat	-23.6044613	The degrees North +, South - with 7 digits of precision
lng	-46.6925758	The degrees West -, East + with 7 digits of precision

For images:

Name	Value	Description
serial	ABC123	The serial number for the camera
type	image	This record is for an available image
key	2021112201	The first identifier for the image
suffix	3147	The second identifier for the image
date	1637544707	The time the video occurred. UTC based unix time in seconds.
lat	-23.6044613	The degrees North +, South - with 7 digits of precision
lng	-46.6925758	The degrees West -, East + with 7 digits of precision

Example of returned JSON array (array will be [] if no media found):

```
{
  "status":200,
  "data":[
      {
        "serial":"ABC123",
        "type": "video",
        "key":"20211122_013147",
        "date":1622754233,
        "lat":-23.6044613,
        "lng":-46.6925758
      },
      {
        "serial":"ABC123",
        "type":"image",
        "key":"2021112201",
        "suffix": "3147",
        "date":1622754233,
        "lat":-23.6044613,
        "lng":-46.6925758
      },
        {...
    ]
}
```

5.4 Get On Demand Item

A GET call to this method is used to upload a video or image file from the camera to the cloud to be processed. The results of 5.3 above can be used to determine available media for a given timeframe.

Note that it can take up to 15 minutes after this request for the video to be available (AI processing time). Also, the camera must be awake for the transfer to occur.

Upon receiving this request, we add this item to your list of camera events and on demand items. The item will then be returned in the getMediaList and getMediaData calls above. If the media is not yet ready on the cloud, the mediaList returned from getMediaData will be blank.

The url has an endpoint of getOnDemandItem and is built using the following parameters:

- cameraSerial - the serial number for the camera (not the tracking device)
- type - video or image
- key - the file key sent in 5.3 above
- suffix - the file suffix (required for images only) sent in 5.3 above
- lat - the lat sent in 5.3 above
- lng - the lng sent in 5.3 above
- date - the date sent in 5.3 above

Example url:

https://gpstrackservices.com/svr3/getOnDemandItem?cameraSerial=ABC123&type=video&key=20211122_013147&lat=-23.6044613&lng=-46.6925758&date=1622754233

The request returns the event info for this request encoded in JSON in the following format:

Name	Value	Description
serial	ABC1234	The serial number for the camera
eventId	100012312	The identifier for the event
eventType	109	Each event has a type. See AI Camera Event Types

date	1637544707	The time the video occurred. UTC based unix time in seconds.
lat	-23.6044613	The degrees North +, South - with 7 digits of precision
lng	-46.6925758	The degrees West -, East + with 7 digits of precision

Example of returned JSON object:

```
{
  "status":200,
  "data": {
      “serial”   :”ABC1234”,
      “eventId”:100012312,
      “eventType”:109,
      “date”:1637544707,
      “lat”:-23.6044613,
      “lng”:-46.6925758
}
```

6. Rate Limiting

Each account is allowed a maximum of 3600 requests per hour.

7. Testing

Going forward, all new clients of the public API should be using the ApiKey authentication. To test the calls, you can use cURL from your command line.

For example:

curl --header "ApiKey: b8h88206CD.4f4f7007G704547d7f66" "https://gpstrackservices.com/svr3/getStatuses"

Our existing test site that uses login and password authentication will be phased out (https://cdn.gps256.com/api/getTestView5.html).

8. AI Camera Webhooks

For the AI Camera solution, we also offer webhooks for video events and camera statuses.

Webhooks are the opposite of the above API in that we push the data to you as it happens (as opposed to you requesting data from us).

8.1 Setup

In order to use our webhooks you will need to have a web service set up that can accept POST calls from us. The endpoint needs to allow secure communications via TLS.

As camera events and statuses occur, we will POST data to this url. For example, https://api.customer.com/aicamera

8.2 Security

You will need to provide us with a secret key. We recommend creating a random string (with high entropy) of at least 20 characters/numbers.

In the Authorization header of the request, we will SHA256 hash the secret key with the data payload and send that as the signature:

Authorization: Signature-256 SHA256(secret key, payload)

The payload being the plain text data we are sending you in the body of the request. You can verify the POST by doing the same on your side with the POST plain text data and the secret key. If you get the same result as the string we send in the authorization header, then you know the data is valid.

Here is an example:

Secret Key
```
42f08d83e5c1706f868e2121c7e2f8
```

Payload

Event object (see 6.3.1 below) JSON:

```
{"serial":"ABC1234","eventId":100012312,"eventType":109,"date":1637544
707,"lat":-23.6044613,"lng":-46.6925758}
```

Get string representation of the payload JSON object (for example in Java, payload.toString())

Putting it together:

```
SHA256(42f08d83e5c1706f868e2121c7e2f8,payload.toString())
=
b9cf1b209aca49e09f2f6a2a3192824b667d0f635b247534b4b8bd438bdb63f0
```

Header sent:

```
Authorization: Signature-256
b9cf1b209aca49e09f2f6a2a3192824b667d0f635b247534b4b8bd438bdb63f0
```

8.3 Methods

8.3.1 Events

When a camera event occurs, we will POST the event data to you. The POST will be sent to the url you provided us in 6.1 with /event added to the url. For example:

https://api.customer.com/aicamera/event

The data will be JSON in the following format:

Name	Value	Description
serial	ABC1234	The serial number for the camera
eventId	100012312	The identifier for the event
eventType	109	Each event has a type. See AI Camera Event Types
date	1637544707	The time the video occurred. UTC based unix time in seconds.
lat	-23.6044613	The degrees North +, South - with 7 digits of precision
lng	-46.6925758	The degrees West -, East + with 7 digits of precision

Data JSON object:

```
{
        “serial” :”ABC1234”,
        “eventId”:100012312,
        “eventType”:109,
        “date”:1637544707,
        “lat”:-23.6044613,
        “lng”:-46.6925758
}
```

Example POST using cURL:

*Need to stringify the JSON data before sending (after creating authorization signature)

```
curl --header "Signature-256
b9cf1b209aca49e09f2f6a2a3192824b667d0f635b247534b4b8bd438bdb63f0" -d
"{\"serial\":\"ABC1234\",\"eventId\":100012312,\"eventType\":109,\"dat
e\":1637544707,\"lat\":-23.6044613,\"lng\":-46.6925758}"
https://api.customer.com/aicamera/event
```

8.3.2 Camera Status

Camera statuses are reported every 5 minutes per camera. When a camera status is reported, we will POST the status data to you. The POST will be sent to the url you provided us in 6.1 with /cameraStatus added to the url. For example:

https://api.customer.com/aicamera/cameraStatus

The data will be JSON in the following format:

Name	**Value**	**Description**
serial	ABC1234	The serial number for the camera
date	1637544707	The time the video occurred. UTC based unix time in seconds.
connected	true	True if the camera is online (awake), false if not.

errorCodes	[2,26]	One or more error codes. See AI Camera Status Error Codes
memoryStatus	4	Memory status code. See AI Camera Status Memory Status Codes
subCameraStatus	7	Sub camera (secondary camera) status code. See AI Camera Sub Camera Status Codes

Data JSON object:

```
{
        “serial” :”ABC1234”,
        “date”:1637544707,
        “connected”:true
        “errorCodes”:”[2,26]”,
        “memoryStatus”:4,
        “subCameraStatus”:7
}
```

Example POST using cURL:

*Need to stringify the JSON data before sending (after creating authorization signature)

```
curl --header "Signature-256
f9906f869626ab7f542053eb16440ea869ac8e80d2322ae35075dd1db235059b
" -d
"{\"connected\":true,\"date\":1637544707,\"subCameraStatus\":7,\"seria
l\":\"ABC1234\",\"memoryStats\":4,\"errorCodes\":\"[2,26]\"}"
https://api.customer.com/aicamera/cameraStatus
```

9. Definitions

9.1 Location Types

Name	Value	Description
MOVING_HEARTBEAT	5	Sent every 2 minute vehicle is in motion

STOPPED_HEARTBEAT	3	Sent every 4 hour when vehicle is stopped
IGNITION_OFF	2	If ignition sense is configured and the vehicle is started.
IGNITION_ON	4	If ignition sense is configured and the vehicle is turned off.
STARTER_DISABLED	24	A command to disable the vehicle's starter motor has been received by the vehicle
STARTER_ENABLED	25	A command to enable the vehicle's starter motor has been received by the vehicle
INPUT_1_HIGH	6	Represents a change in state of vehicle input 1
INPUT_1_LOW	7	Represents a change in state of vehicle input 1
INPUT_2_HIGH	8	Represents a change in state of vehicle input 2
INPUT_2_LOW	9	Represents a change in state of vehicle input 2
POWER_CONNECTED	22	For devices with internal battery power, this represents when a device is connected to external power
POWER_DISCONNECTED	23	For devices with internal battery power, this represents when a device is removed from external power
STOP	26	Vehicle has stopped for more than 5 minutes
GPS_ACQUIRED	30	Device has acquired a workable connection with GPS satellites
IMPACT ALERT	32	Device detected an impact based on ACCEL Settings
HARSH ACCEL	33	Device detected excessive G force during acceleration
HARSH BRAKE	34	Device detected excessive G force during deceleration
SWERVE LEFT	35	Device detected excessive G force during a left turn
SWERVE RIGHT	36	Device detected excessive G force during a right turn
COLD BOOT	39	Device application started after power was applied
USER LOCATE	40	User requested location request

STATUS CHECK	41	Device diagnostics request
WARM BOOT	42	Device application restarted after software reboot request.

9.2 Alert Types

Name	**Value**
Fuel Alert	1
Idle Alert	2
Low Battery Alert	4
Maintenance Alert Distance	5
After Hours Alert	6
Speed Alert	8
Stop Alert	9
Tow Alert	10
Lost Comm Alert	12
Lost GPS Alert	13
Maintenance Alert Engine	14
Input High Alert	21
Input Low Alert	22
Zone Enter Alert	23
Zone Exit Alert	24
Power On Alert	25
Power Off Alert	26
Installation Alert	27
Impound Alert	28
Country Enter Alert	29

Country Exit Alert	30
Theft Alert	37
SOS Alert	38
Inactivity Alert	40
Fuel Stop Alert	41
Incident Alert	42
Confirmed Incident Alert	43
Maintenance Alert Calendar	44
Check Engine Alert	45
Expiring Devices Alert	46
Crash Alert Camera	100
Harsh Acceleration Alert Camera	101
Harsh Braking Alert Camera	102
Harsh Left Alert Camera	103
Harsh Right Alert Camera	104
Harsh Turn Alert Camera	105
Driver Initiated Alert Camera	106
Roll Over Alert Camera	107
Tailgating Alert Camera	108
Collision Risk Alert Camera	109
Reckless Driving Alert Camera	110
Stop Sign Violation Alert Camera	111
Red Light Violation Alert Camera	112
Camera Errors Alert	113

9.3 User Action Types

Name	**Value**
View User History	105
Enable Starter	107
Disable Starter	108
Modify	109
Create	110
Delete	111
Forgot Password	112
Change Password	113
Login	114
Logout	115
Buzzer	117
View DMS Map	118
View Status	119
Door Unlocked	120
Locate Now	121
Status Check	122
Reboot	123

9.4 AI Camera Event Types

Name	**Value**
Crash	100
Harsh Acceleration	101

Harsh Braking	102
Harsh Turn	105
Driver Initiated	106
Tailgating	108
Collision Risk	109
Reckless Driving	110
Stop Sign Violation	111
Red Light Violation	112
On Demand Image	1000
On Demand Video	1001

9.5 AI Camera Status Error Codes

Name	**Value**
Real time clock issue	4
Camera H/W connection or memory issue	7
GPS connection status error	10
Serial communication fail	21
Low voltage (<10v)	23
Angle changed compared to initial install	26

9.6 AI Camera Status Memory Status Codes

Name	**Value**
No SD card	1

SD card error - disk bad sector (format or replace)	2
Abnormal end routine (SD card)	4
No USB memory stick (UMS)	16
UMS Error (Disk bad sector - format or replace)	32
Abnormal end routine (UMS)	64
Abnormal end routine (SD Card & UMS)	68

9.7 AI Camera Status Sub Camera Error Codes

Name	Value
Sub camera connection error	1
Re-install sub camera	2
Sub camera software error	3
Re-install sub camera 2	6

9.8 Error Codes

If a request is successful it will return an HTTP code of 200, otherwise it will return one of the following error codes:

Name	Value	Description
MISSING_PARAMETERS	441	One of the required parameters is missing
UNAUTHORIZED	443	The password is incorrect for the server.
UNKNOWN_USER	444	The reported user (looked up from login) is not known by the server.
UNKNOWN_VEHICLE	445	The requested serial is not known by the server.

THROTTLE_REQUEST	446	The client has exceeded the max request rate of the API. The max request rate is around 10 requests per second but can be increased if needed.
BAD_REQUEST	447	The request is not understood for any other reason.
MISSING_STATUS	448	Vehicle doesn't have a status yet. This is because the GPS tracking device has not yet reported to the server. (Usually means the device is not yet installed)
DATE_IN_PAST	449	A date has been specified more than 1 month in the past.
DEVICE_SUSPENDED	471	The plan for the device has been expired for more than 30 days and the device is now suspended. Devices can be renewed via the customer UI or the distributor site (applies to SVR and PUI only)
DEVICE_INACTIVE	472	The device is inactive. This is usually because the plan has been expired for > 1 year (SVR/PUI) or the plan was cancelled (fleet).