Using Third-Party Devices with MyGeotab

Table of Contents

Overview

This document describes how Geotab provides support for third-party devices in MyGeotab. Third-party devices are defined as any telematics device that is not manufactured or sold by Geotab but is used in conjunction with Geotab's fleet management platform.

Integrating a third-party telematics device with MyGeotab requires the use of the MyAdmin SDK. Refer to the Third-Party Telematics Devices and MyGeotab (https://docs.google.com/document/d/1Mddnxc2qKBCNYvVu0-BXcyR-blPlHwa0Zun0mBzZt88/preview) document for a high-level overview of the registration process or contact your reseller for more information about obtaining a MyAdmin account or API privileges.

Device identification

A serial number will be provided for each device by Geotab. The serial number will be very similar to a standard Geotab GO device serial number. It consists of a 12 digit alphanumeric string and contains two pieces of important information:

1. What type of device this is
2. The unique identifier of the device

For example, a serial number for a custom device will look like this:

C1-800-000-0001

where C1 indicates the custom device type and 800-000-0001 is the unique identifier.

Device type allocation

The device type in the serial number (for example C1) is specific to each third-party device and will be allocated by Geotab for a particular vendor (i.e. the person or company interested in third-party integration).

MyGeotab User Interface

When entering a serial number, the device type will be used to show or hide features in the device editor. The standard set of fields for custom devices include:

- Device description
- Device identifier (not the serial number but third-party device identifier, as required)
- Groups

- Comments
- Odometer
- Engine hours
- Work Hours
- Time Zone
- Systems integration

Geotab provides various systems (MyAdmin, install.geotab.com, etc.) that tie in closely with its line of GO products. By using a standardized serial number scheme as described above, Geotab will be able to provide additional services in the future with regards to third-party devices. For example, recording a successful installation or summarizing to a MyAdmin user how many third-party devices are linked to their account.

API

The Third-Party Devices API is accessible through HTTPS and JSON.

The data that MyGeotab accepts from third-party devices is limited. Currently, Geotab supports the recording of the following data types from a third-party device:

- Latitude
- Longitude
- Speed
- Date and time
- Ignition status (on/off)
- Auxiliary relay status for up to 8 relays
- Engine/Diagnostic data
- Device data

Integrating third-party device data into MyGeotab requires only three methods: Authenticate, ProvisionDevice, and AddData. All three methods are part of the MyAdmin API and require special MyAdmin account privileges to run.

The following method calls are supported and should be called in the order they appear, as documented below, to have a successful integration:

Authenticate

```
Authenticate (string userName, string password)
```

This method authenticates the user with the MyAdmin API and returns an `ApiUser` object. It requires the username and password of a MyAdmin account with MyAdmin API privileges.

- string userName — The third-party email address to authenticate with.
- string password — The third-party password to authenticate with.

Note that sessions will expire after a period of time. Calling a method with an expired sessionId will throw a SessionExpiredException. If this occurs, call Authenticate to obtain a new sessionId.

Below is an example of the request made to the MyAdmin API to authenticate a user:

> **Note**
>
> When performing the Authenticate call, we expect you to use the POST HTTP method to pass sensitive username/password information to Geotab.

If authentication is successful, you will see a response similar to the following:

```
{
                                "result":{
                                "userId":"24k380h7-a472-947f-33ee-d4b1lo5e12b8",
                                "sessionId":"083c6dfb-3e3a-4fea-a3db-1fa546062ee4",
                                "lastLogonDate":"2017-01-25T15:21:03.723Z",
                                "accounts":[
        {
                                "accountId":"TEST01",
                                "currency":{
                                "code":"USD",
                                "name":"US Dollars"
           }
        }
     ],
                                "roles":[
        {
                                "comments":"Special user for MyAdminApi",
                                "name":"MyAdminApiUser"
        }
     ],
                                "name":"testuser@geotab.com"
   }
}
```


Note

The sessionID expires in approximately 20 minutes. Receiving a "SessionExpiredException" error as a result of a standard API call will require you to re-authenticate.

ProvisionDevice

```
ProvisionResult ProvisionDevice (Guid apiKey, Guid sessionId, int productId, string server, string promoCode, string subPlan)
```

This is a provisioning call that adds a device to MyAdmin and generates a Serial Number to be used for referencing or viewing the device in MyGeotab.

Note

This method only needs to be run once for each third-party device you wish to integrate into MyGeotab.

Parameter description:

Property	Description
Guid apiKey	Based on the userID, the key used to authenticate the session. This is the result.userId property from the Authenticate response above.
Guid sessionId	Identification used with the apiKey to authenticate the session. This is also found in the result.sessionId property from the authenticate response above.
int productId	Integer code used to specify the type of the device to provision. This code will be provided by Geotab.
string server	String used for Server of the Store and Forward server. If null the default Store and Forward server is used. This code will be provided by Geotab.
string promoCode	String used to specify promo code for third-party device. This code will be provided by Geotab.
string subPlan	String used to provision OEM devices. This code will be provided by Geotab.

The following is an example of parameters that can be used to call ProvisionDevice within MyAdmin:

```
apiKey: “24k380h7-a472-947f-33ee-d4b1lo5e12b8”,
sessionId: “f684812e-377f-478f-a813-1eef7d25e1a2”,
productId: 10015,
server: "server",
promoCode: "PROMO",
subPlan: "subPlan"
```

If successful, the request will return a result similar to:

```
{
                                "result": {
                                "isSuccess": true,
                                "serialNo": "AB0C12DE3456"
  }
}
```

> **Note**
>
> You only need to call the Provision Device method once for each third-party device you want to provision.

The returned `serialNo` value can now be used to add a vehicle to your MyGeotab database using the steps below.

1. Select **Vehicles** from the main menu.
2. Click the **Add** button and select the **Add vehicle** button from the dropdown menu.
3. Fill in the **Serial number** field with the serial number you received at the end of the previous section.
4. Fill in the **Description** field with a value that will help you identify this device.
5. Click **OK** to finalize your input.

AddData

```
AddData(Guid apiKey, Guid sessionId, List<ThirdPartyDataRecord> recordsToAdd)
```

Once your device has been added to MyGeotab, this method is used to feed the device data to the MyGeotab database.

The ThirdPartyDataRecord parameter has three types: ThirdPartyGpsRecord, ThirdPartyStatusRecord, and ThirdPartyAccelerationRecord. All three types share the following common properties:

Common Properties

Property	**Description**
string serialNo	MyAdmin-assigned serial number of the device, which will be returned during device provisioning.
DateTime DateTime	DateTime of the record in UTC.

The remaining parameters depend on the type of ThirdPartyDataRecord you are using:

ThirdPartyGpsRecord

Property	**Description**
float Longitude	Longitude component of the GPS coordinates.
float Latitude	Latitude component of the GPS coordinates.
bool IsGpsValid	Are the GPS coordinates valid.
float Speed	Speed recorded by the device.
bool IsIgnitionOn	State of the ignition.
bool IsAuxiliary1On to IsAuxiliary8On	State of the auxiliary relays.

ThirdPartyStatusRecord

Property	**Description**
short Code	The diagnostic code (see table below).
int Value	The value of the status data, before applying any offset or conversion as configured in MyGeotab.

ThirdPartyAccelerationRecord

Property	**Description**

Property	Description
short X	The X-axis acceleration (forward or backward) in milli-g.
short Y	The Y-axis acceleration (right or left) in milli-g.
short Z	The Z-axis acceleration (up or down) in milli-g.

The AddData method will return an object of type AddDataResult that indicates if the data was added successfully:

AddDataResult

A successful AddData method call will return a list of AddDataResult objects. The number of AddDataResult items will match the number of items in the recordsToAdd parameter. Each AddDataResult item contains a boolean value of true or false for the IsSuccess property and a message in the Error property if any errors are encountered.

Property	Description
bool IsSuccess	What was the error during the sending of the data.
string Error	The Y-axis acceleration (right or left) in milli-g.

Referencing Geotab.Checkmate.ObjectModel.dll will give access to all namespaces and objects needed to make the calls to the Third-Party Devices API.

Overview of needed namespaces and objects:

1. Namespace Geotab.Checkmate.Web contains classes through which the remote calls can be made.
2. WebServerInvoker — The object can be used to make the remote JSON call MyAdminAPI server via an HTTP/HTTPS POST command. A demonstration of its usage is provided in the .NET examples (http://my3.geotab.com/sdk/dotnet.zip?geotabsdk%3D/dotnet.zip).
3. Namespace Geotab.Checkmate.ObjectModel contains objects used as parameters and return types for the remote calls mentioned above.

Below is an example of parameters that can be used with the AddData method to start sending data to the MyGeotab database:

```
apiKey: "24k380h7-a472-947f-33ee-d4b1lo5e12b8",
sessionId: "51fa8c5a-c065-4296-b21f-8cbb30158ed5",
recordsToAdd: [{"serialNo":"CG0B20DD7588","dateTime":"2016-11-23T20:28:40.648Z","isIgnitionOn":true,"isGpsValid":true,"latitude":43.
7550583,"longitude":-79.4769897,"speed":52.0,"type”: ThirdPartyGpsRecord}]
```

If successful, the resulting response will resemble the following:

```
{
                                "result": {
                                "isSuccess": true
  }
}
```

Note: You will need to specify a type for each third-party data record you send over. The API will respond with a JSON serialization exception if you don’t specify a record type.

> **Note**
>
> Each third-party data record needs to have a unique UTC date/time value.

Multiple records can be batched together in a single call of the AddData method. A sample of such can be seen below:

```
apiKey: "24k380h7-a472-947f-33ee-d4b1lo5e12b8",
sessionId: "f54f6ef5-e6de-43bc-99f2-3ce6f83e2171",
recordsToAdd: [
{"serialNo":"CG0B20DD7588","isIgnitionOn":true,"isGpsValid":true,"dateTime":"2016-11-23T20:30:10.648Z","latitude":43.4347,"longitud
e":-79.7710571,"speed":57.0,"type”: ThirdPartyGpsRecord},
{"serialNo":"CG0B20DD7588","isIgnitionOn":true,"isGpsValid":true,"dateTime":"2016-11-23T20:30:11.648Z","latitude":43.4353676,"longit
ude":-79.7701874,"speed":41.0,"type”: ThirdPartyGpsRecord},
{"serialNo":"CG0B20DD7588","isIgnitionOn":true,"isGpsValid":true,"dateTime":"2016-11-23T20:30:12.648Z","latitude":43.4352112,"longit
ude":-79.7703934,"speed":52.0,"type”: ThirdPartyGpsRecord},
{"serialNo":"CG0B20DD7588","isIgnitionOn":true,"isGpsValid":true,"dateTime":"2016-11-23T20:30:13.648Z","latitude":43.4353676,"longit
ude":-79.7701874,"speed":41.0,"type”: ThirdPartyGpsRecord},
{"serialNo":"CG0B20DD7588","isIgnitionOn":true,"isGpsValid":true,"dateTime":"2016-11-23T20:30:14.648Z","latitude":43.4355202,"longit
ude":-79.7700119,"speed":15.0,"type”: ThirdPartyGpsRecord},
{"serialNo":"CG0B20DD7588","isIgnitionOn":true,"isGpsValid":true,"dateTime":"2016-11-23T20:30:15.648Z","latitude":43.4355469,"longit
ude":-79.7699814,"speed":11.0,"type”: ThirdPartyGpsRecord}
]
```

If successful, the resulting response will resemble the following:


```
{
                                "result": [
    {
                                "isSuccess": true
    },
    {
                                "isSuccess": true
    },
    {
                                "isSuccess": true
    },
    {
                                "isSuccess": true
    },
    {
                                "isSuccess": true
    },
    {
                                "isSuccess": true
    }
  ]
}
```


Third-Party Diagnostic Codes

Third-party status data diagnostics have their own source filter in MyGeotab called Third-Party.

Before a third-party diagnostic can be used, the customer must provide the desired details of the diagnostic to Geotab. Once this information has been received, Geotab will add it to the next available release of MyGeotab.

The following diagnostics are currently available. If not otherwise noted, the value is an integer without a unit of measurement.

1. Low battery
2. Tow detection
3. Starter tamper
4. Power tamper
5. Battery voltage (unit: V, conversion: 0.001, example: 12.001 V must be inserted as the integer value 12001)
6. Temperature (unit: C, conversion: 0.01, example: 1.01 C must be inserted as the integer value 101)
7. GSM signal strength
8. Loaded voltage (unit: V, conversion: 0.001, example: 12.001 V must be inserted as the integer value 12001)
9. Battery level (unit: %)
10. Successful uploads
11. Failed uploads
12. GPS fix attempts
13. GPS on time (unit: s)
14. Trip count

AddGps

> **Deprecation Notice**
>
> This method has been deprecated as of January 2017. See AddData and AddDataResult for the current implementation.

```
List<AddGpsResult> AddGps(Guid apiKey, Guid sessionId, List<ThirdPartyLogRecord> recordsToAdd)
```

This call is used to feed the device data to MyGeotab. Parameter description:

1. Guid apiKey — Based on the userID, the key used to authenticate the session.
2. Guid sessionId — Identification used with the apiKey to authenticate the session.
3. List of ThirdPartyLogRecord containing the data.

Referencing Geotab.Checkmate.ObjectModel.dll will give access to all namespaces and objects needed to make the calls to the Third-Party Devices API.

Overview of needed namespaces and objects:

1. Namespace Geotab.Checkmate.Web contains classes through which the remote calls can be made.
2. WebServerInvoker — The object can be used to make the remote JSON call MyAdminAPI server via an HTTP/HTTPS POST command. A demonstration of its usage is provided in the .NET examples.
3. Namespace Geotab.Checkmate.ObjectModel contains objects used as parameters and return types for the remote calls. The list of the objects is:

ProvisionResult — Return type for provisioning call

- string SerialNo — Device serial number generated by MyAdmin
- bool IsSuccess — Is provisioning call was successful
- string Error — Reason for provisioning call failure

ThirdPartyLogRecord — Used to describe the device data.

- string SerialNo — MyAdmin assigned serial number of the device, which will be returned during device provisioning
- DateTime DateTime — DateTime of the log
- float Longitude — Longitude component of the GPS coordinates
- float Latitude — Latitude component of the GPS coordinates
- bool IsGpsValid — Are the coordinates valid
- float Speed — Device speed
- bool IsIgnitionOn — State of the Ignition
- bool IsAuxiliary1On to IsAuxiliary8On — State of the auxiliary relays

AddGPSResult

> **Deprecation Notice**
>
> This method has been deprecated as of January 2017. See AddData and AddDataResult for the current implementation.

A successful AddGps method call will return a list of AddGpsResult objects. The number of AddGpsResult items will match the number of items in the recordsToAdd parameter.

bool IsSuccess — Is the sending of the results successful

string Error — What was the error during the sending of the data

Please refer to the provided example for the usage of the objects in the SDK .NET examples.

Integration Verification

Once your API calls are returning successfully, you can provide the end user with the serial number (the number that was returned by the ProvisionDevice method) for their third-party device.

The end user or reseller can verify that your integration is operational by logging in to their MyGeotab database and doing the following:

1 Adding a third-party device to a database:

Navigate to Vehicles, select the **Add** dropdown, then select the **Add vehicle** option. On the page that follows, input the device serial number and click **OK**.

2 Verifying the trips history:

Trip logs can be verified by navigating to the **Map** and clicking the **Trips history** button. Select a vehicle from the list or use the search box to filter for the relevant vehicle. The integration is successful if the user can see trip information for the vehicle with the third-party device installed.

3 Verifying Status Data:

If you are sending Status Data from your third-party device to MyGeotab, you can view this data by navigating to **Activity** > **View Accidents & Log Data** and using the filters to find your third-party device records.

Representing your custom device on the Live Map and rules-engine processing

Most integrations will require the third-party device to indicate its current state to Geotab users (via MyGeotab Live Map) or to the Geotab platform rules engine (for exceptions processing).

This is accomplished by the third-party device manipulating the `isIgnitionOn` and `isGpsValid` properties in tandem.

```
{
                            "serialNo": "CG0B20DD7588",
                            "isIgnitionOn": true,
                            "isGpsValid": true,
                            "dateTime": "2016-11-23T20:30:10.648Z",
                            "latitude": 43.4347,
                            "longitude": -79.7710571,
                            "speed": 57.0,
                            "type”: ThirdPartyGpsRecord
}
```

To indicate that a third-party device has stopped moving (either at the end of a trip or the leg of a journey) it needs to send records in the following sequence, with its properties set as follows:

- A record with isIgnitionOn=false and isGpsValid=true
- Followed by a record with (a) identical GPS coordinates as previous with (b) a timestamp difference >200 seconds

To be certain, a follow up record with isIgnitionOn=false and isGpsValid=false to force the Live Map to interpret a stop

The third-party device will indicate that it is moving by sending a record with isIgnitionOn=true and isGpsValid=true.

> **Note**
>
> When `isGpsValid` is set to `false`, the Rules Engine will consider the GPS coordinates for this record inaccurate and will ignore them. To deem a coordinate accurate there must be a GPS latch with at least 5 satellites, at which point `isGpsValid` is set to `true`. Less than that, by default on a GO device, the coordinates are still sent but with isGpsValid=false and possibly ignored by rules engine processing.

On the Geotab Live Map, the only Iconography for vehicles states are as follows:

- Square - Vehicle is stopped and outside a zone
- Star - Vehicle is stopped and inside a zone
- Triangle - Vehicle is moving

Third-Party Device Integration Using .NET

All Third-Party Devices API calls are executed through the MyAdmin API (https://myadmin.geotab.com/sdk/api/apiReference.html).

The .NET DLLs for MyAdmin can be downloaded here (https://myadmin.geotab.com/sdk/downloads/MyAdminApiLib.zip). You can find a demo of third-party telematics device provisioning for .NET here (https://my.geotab.com/sdk/#/dotNetExamples).

To call the Third-Party Devices APIs using the MyAdmin .NET DLL, use the WebServerInvoker object to specify your intended MyAdmin method along with its dependent parameters.

> **Note**
>
> The .NET DLL has no classes relating to Third-Party Telematics Device MyAdmin API calls.

You can find examples of using MyAdmin methods with the MyAdmin .NET DLLs in the .NET examples (https://myadmin.geotab.com/sdk#/dotnet-examples).