JSON API

Our GPS Fleet Software API allows third-party software to access data of our system and pass data as well to realize cooperative programs and apps. With our API we enable other developers to build a synergy between our GPS Fleet Software and their Server or App. For use cases in the following documentation I will use our Hosting, the API is also usable with an own server but the functionality then relies on the accessibility of the server.

How to access the GFS API

To access the API the same link is used as to log in to our software as a normal user. For each access a verification is necessary with can be done by username and password or an API key available inside the software. The simplest of uses is to check if a user is legitimate to the software. In this case the call would look like this:

http://gfs2.sw-management.at/GFS/api/std/checkuser?apikey=user----password

or

http://gfs2.sw-management.at/GFS/api/std/checkuser?apikey=apikey

The response then will be weather the user is accepted or not:

```
{"info":"ok"} or {"error":"100"}
```

Where to find the various calls

All the various possibilities available to access and pass data from/to our system can be found in a central xml file which holds all information such as necessary and optional parameters and there type. The xml with the full overview can be found at: http://gfs2.sw-management.at/GFS/api/application.wadl (or for a GFS running on a different server serverlink/api/application.wadl). For instance the checkuser call used above would look like this in our application.wadl:

```
<resource path="/std">
    <resource path="/checkuser">
      <method id="checkuser" name="GET">
        <request>
          <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="apikey" style="query" type="xs:string"/>
        </request>
        <response>
          <representation mediaType="application/json"/>
        </response>
      </method>
   </resource>
```

Basic calls

the most basic calls are found in the masterdata subgroup. Under masterdata there are further subgroups for different objects our server can provide. Such are for instance asset, geofence or driver. Some very basic calls may look like this:

http://gfs2.sw-management.at/GFS/api/masterdata/asset/getAssets?apikey=user----password,

http://gfs2.sw-management.at/GFS/api/masterdata/geofence/getGeofences?apikey=user----password or

http://gfs2.sw-management.at/GFS/api/masterdata/driver/getDrivers?apikey=user----password

New calls with 3.8.29

modify

- /logbook/set_operating_hours: calling this command with the parameters apikey, assetid and new_h_value will set a new value for the operating time of an asset.
- /insertpos: calling this command with the parameters apikey, assetid, lat, lng, d1, speed, ts_yyyyMMddhhmmss and optional opt_fuellevel will set a new position for the asset including ignition, speed and the time for this particular position and optionally the fuellevel.
- /insertpos_address: calling this command with the parameters apikey, assetid, address, d1, speed, ts_yyyyMMddhhmmss and optional opt_fuellevel will set a new position for the asset at the particular address.

time

- /start_time_record: calling this command with the parameters apikey, start_ts_yyyyMMddhhmmss, personid and optional opt_assetid, opt_custid_geofenceid, opt_activityid, opt_commentstr, opt_lat, opt_lng and opt_address will start a time record for the person starting with the given time and will optionally add the used asset, the geofence or the lng/lat or the address the person is working at, a comment and the activity the person is performing corresponding to the optional parameters given.
- /update_time_record: calling this command with the parameters apikey, end_ts_yyyyMMddhhmmss, personid and optional opt_assetid, opt_custid_geofenceid, opt_activityid, opt_commentstr, opt_lat, opt_lng and opt_address will end the last time record for the person with the given time and will optionally add the used asset, the geofence or the lng/lat or the address the person is working at, a comment and the activity the person is performing corresponding to the optional parameters given. In case of a vehicle the end address/position may vary from the start address/position.
- /insert_complete_time_record: calling this command with the parameters apikey, start_ts_yyyyMMddhhmmss, end_ts_yyyyMMddhhmmss, personid and optional opt_assetid, opt_custid_geofenceid, opt_activityid, opt_commentstr, opt_lat, opt_lng and opt_address will create a full time record from the given start time to the given end time and will optionally add the used asset, the geofence or the lng/lat or the address the person is working at, a comment and the activity the person is performing corresponding to the optional parameters given.

masterdata

- /data/getLastTimeRecord: calling this command with parameters apikey and personid will return the last time record of this person.
- /data/getActivityTypes: calling this command with the apikey returns all activity types of the account.
- /asset/getAssets_by_assetkey: calling this command with apikey and assetkey returns the asset for the given key. (The key must be set inside the GPS Fleet Software and can be used to reference to a third party system)

bin

- /upload: calling this command will allow to upload a file to the server (an image in most cases). As always the apikey is needed as parameter, optional opt_relationID, opt_pkid, opt_category, opt_name, opt_commentinfo, opt_binid can be set to specify the relation and the object the file relates to or to set the category, name or a comment, also the binid could be set to overwrite an existing file.

odr

- /getDynDataValues: calling this command with the apikey and optional opt_limit, opt_relationID or opt_pkid for filtering will return the dynamic fields which can be created and configured by the customer in the GPS Fleet Software allowing to set extra fields/flags for objects.
- /setDynDataValue_for_pkid_defid_relationID: calling this command with parameters apikey, pkid, relationID, defid and one parameter of v_int, v_str, v_bigint, v_float, v_date_yyyyMMddhhmmss, v_boolean will set the given field/flag of the given object to the given value. Important: the correct value parameter must be used to update the field/flag.