



Fuel Management Systems

GIR W250 Software
Reference Manual

2026-04-02

Contents

1	Getting started	11
1.1	Connection	11
1.2	Home page	11
1.3	Main menu	11
1.4	Alert panel	12
1.5	Recording your first transaction	12
1.5.1	Set the identification mode	12
1.5.2	Create a vehicle	13
1.5.3	Pair a controller	14
1.5.4	Record a transaction	16
2	Common user interface elements	19
2.1	Collection view	19
2.2	Compact lists	21
2.3	Unit switching	22
3	Settings	23
3.1	QSC/RSC protection	23
3.2	General	23
3.2.1	Sites	23
3.2.2	Ident. modes	30
3.2.3	System	33
3.2.4	About	34
3.3	Users	34
3.3.1	User editor	34
3.3.2	Password recovery	37
3.4	Departments	38
3.4.1	Department editor	38
3.4.2	Levels	38
3.5	Fuel	39
3.5.1	Products	39
3.5.2	Models	39
3.5.3	Additional prompts	40
3.5.4	Suppliers	41
3.5.5	Strapping charts	41
3.5.6	Receipt	42
3.6	Advanced	43
3.6.1	Audit trail	43

3.6.2	Features	43
3.6.3	Impexp	43
4	Vehicles and drivers	45
4.1	Vehicles	45
4.1.1	Department reassignment	46
4.2	Drivers	46
4.2.1	Department reassignment	47
5	Fuel transactions	49
5.1	Transactions list	49
5.2	Manual transaction entry	52
5.3	Transaction modification	53
5.4	Transaction cancellation	53
5.5	Quick reports	53
5.6	Quick graphs	55
5.7	New meters	56
6	Supervision	57
6.1	Supervision	57
6.2	Fuel transac.	59
6.3	Access tr.	59
6.4	Manual Deliveries	60
6.5	Inventories	61
6.5.1	Auto. Deliveries	62
6.5.2	Auto. Deliveries unit price	63
6.5.3	Auto. Deliveries unit price – extra detail	65
6.5.4	Auto. Deliveries – false positives	66
6.5.5	Inventories data	66
6.5.6	Gauge readings	67
6.6	Events	68
6.7	Tanks	69
6.7.1	Past stocks	69
6.7.2	Future stocks	70
7	Controllers	73
7.1	Pairing	73
7.1.1	Start a pairing	73
7.1.2	Paired state	75
7.1.3	Automatically unpaired state	76
7.2	Communication	77
7.2.1	Ping	77
7.2.2	Synchronization	78
7.2.3	Diagnostic tools	78
7.2.4	Other actions	79
7.3	USB Communication	79
7.3.1	Pairing	79
7.3.2	Communication	80
7.3.3	Special cases	81
7.4	Controller usage	81

7.4.1	Pump selection	82
7.4.2	Identification	83
7.4.3	Meter entry	84
7.4.4	Activity code	84
7.4.5	NCE code entry	84
7.4.6	Pump timeouts	84
7.5	Operator menu	85
7.5.1	Pumps unblocking	85
7.5.2	Gauges	86
7.6	Operator identification	87
7.6.1	Vehicle code only	87
7.6.2	Vehicle badge only	87
7.6.3	Driver badge then vehicle code	87
7.6.4	Driver badge then vehicle badge	87
7.6.5	Driver code then vehicle badge	88
7.6.6	Driver code then vehicle code	88
7.6.7	Vehicle badge then driver code	88
7.6.8	Vehicle badge then driver badge	88
7.6.9	Vehicle code then driver badge	89
7.6.10	Vehicle code then driver code	89
8	Dashboard	91
8.1	Adding a sample report to the dashboard	91
8.2	Editing the dashboard	93
9	Specific features	95
9.1	KM+Miles odometers (imperial)	95
9.2	Third-party services	95
9.2.1	Geofencing	97
9.3	Generic badges	98
9.4	MFG badges	98
9.5	BS125 badges	98
9.6	RPK badges	98
9.7	ISO2 badges	98
9.8	iButton badges	99
9.9	Tacho badges	99
9.10	RS-232 / Wiegand badges	99
9.11	Legacy Mifare badges	100
9.12	PIN-protected badge or code	100
9.13	US units	100
9.14	Cons. km/L	100
9.15	AEAT exportación (AEAT export)	100
9.15.1	CAE (.csv)	101
9.15.2	CIM (.xml)	101
9.15.3	CIM (.csv)	102
9.15.4	ATC Canaria – CAE (.xml)	102
9.15.5	ATC Canaria – CIM (.xml)	103
9.16	Excises export	103
9.17	Name + First name	104
9.18	GIR mobile ID badges	104

9.19	Custom properties	104
9.20	File-based import	105
9.21	Transaction import	105
9.22	Editable prices	106
9.23	Reminders	107
9.24	Regions	108
9.25	Zones	108
9.25.1	Stay open schedule	109
9.26	Custom zones only	110
9.27	Advanced Vehicle Options	110
9.27.1	Default vehicle tank parameters	111
9.27.2	Consumption reporting	111
9.28	Group authorizations	111
9.29	Quotas	112
9.29.1	Departments quotas	113
9.30	Multiple Ident. Modes	114
9.31	Tankers	115
9.32	Expiry management	116
9.33	Multiple timezones & languages	116
9.34	Advanced MR-Pompe options	117
9.35	Prompt max volume	117
9.36	Electric vehicles	117
9.36.1	EV Chargepoints	117
9.36.2	Supervision	118
9.36.3	Ongoing charges	119
9.36.4	Electrical transactions	120
9.36.5	OCPP logs	120
9.37	Other features	121
9.37.1	Hide volume units	121
9.37.2	Show vehicles impexp IDs	121
9.37.3	Show drivers impexp IDs	121
9.37.4	Show settings impexp IDs	121
A	System prerequisites	123
A.1	Server machine	123
A.2	Client machine	123
A.3	Storage capacities	123
B	Fueling scenario	125
B.1	Vehicle only	127
B.2	Vehicle + driver	129
B.3	Driver + vehicle	131
C	Vehicles and drivers import and export	133
C.1	Vehicles	134
C.2	Drivers	136
C.3	Common mechanisms	138
C.3.1	Identifiers	138
C.3.2	References	139
C.4	Web services	140

C.4.1	Authentication	140
C.4.2	HTTP status codes	140
C.4.3	Requests/Reponses content	141
C.4.4	Idempotency	143
C.4.5	Examples	144
C.5	Bulk import	148
C.5.1	Basic usage	148
C.5.2	Bulk import fields	149
C.5.3	Bulk import preview	150
D	Transactions export	151
D.1	Export mechanisms	151
D.1.1	File export	151
D.1.2	Web service export	151
D.1.3	Recommendations	153
D.1.4	Fuel transactions ledger	153
D.2	HLF1 format	153
D.2.1	General	153
D.2.2	Export IDs (Identifiers)	153
D.3	C4 format	154
D.3.1	General	154
D.3.2	Export IDs (Identifiers)	154
D.3.3	Fuel transaction URL	154
D.3.4	Columns	154
D.4	Web service /api-impexp/transac_fuels	157
D.4.1	URL	157
D.4.2	Fields	157
E	Remote Control and Monitoring API	159
E.1	Web services	159
E.2	Sites	159
E.3	Inventories	161
E.4	Supervision actions	161
E.4.1	POST /api-impexp/rcm/block_pump	161
E.4.2	POST /api-impexp/rcm/unblock_pump	161
E.4.3	POST /api-impexp/rcm/start_pump	162
E.4.4	POST /api-impexp/rcm/force_refresh	162
E.5	Examples	162
E.5.1	Get the status of a site	162
E.5.2	Block a pump	164
E.5.3	Unblock a pump	165
E.5.4	Start a remote transaction	165
E.5.5	Trigger a new gauging	165
E.5.6	Fetch the first inventories	166
E.5.7	Get the next inventories (after c58709c9-d4f2-49e5-ac1e-da9d35652722)	167

Introduction

GIR W250 is a fuel distribution management software application.

It is used with fuel distribution controllers and offers features allowing a user to:

- Define a list of vehicles (and optionally: drivers) that are allowed to take fuel.
- Synchronize data with controllers using a network connection.
- Retrieve transactions from controllers.
- Monitor and generate reports on transaction history to track vehicle consumption.
- Export transactions for reprocessing in third-party applications.

GIR W250 is a web-based application. It is installed on a single computer (the server), and can be accessed from any other computer using a web browser.

Terminology

- **W250 server:** The computer where the application is installed.
- **Controller:** A device that performs vehicle/driver identification, controls the fuel distribution, and stores transactions. All controllers are connected to the W250 server.
- **Badge:** An item used to identify vehicles or drivers on the terminal. A number of vendor technologies are supported.
- **Transaction:** A collection of data relating to a single visit by a driver/vehicle to a refueling terminal (date, time, vehicle, volume, etc.).
- **User:** A person that connects to the application.
- **Driver:** A person that uses a terminal to take fuel.
- **SaaS:** Software as a Service. W250 installation option whereby the W250 server is installed and maintained from a remote server.
- **On-prem:** On-premises. W250 installation option whereby the W250 server is installed on a customer's computer.

Chapter 1

Getting started

1.1 Connection

When you access the GIR W250 application, you will be prompted to log in using an email address and password. These will have been provided to you. Both fields are required.







If you forgot your password, you can click on the “Forgot password?” link. It will then send an email to your address to reset your password.


1.2 Home page

Once you are logged in as a user, the home page will be displayed. It displays links to the main pages of the application.


1.3 Main menu

The main menu is displayed at the left of all pages. It contains the following items:

- **Home button:**  Returns to the home page
- **Vehicles:**  Shows the vehicles list
- **Drivers:**  Shows the drivers list
- **Fuel transactions:**  Shows a history of all fuel transactions
- **Supervision:**  Displays an overview to monitor and manage controllers, tanks and pumps
- **Alert icon:**  Indicates that there is one or more important notifications about the system state to communicate to the user (controller offline, pump blocked, etc.). Clicking this shows the Alert panel, which contains details about all system notices

- **Main dropdown menu:**  Displays the following items when clicked:
 - *Settings*: Shows the application settings (allows for configuration of system parameters, products, users, etc.)
 - *Events*: Shows the history of all events related to the fuel system
 - *My account*: Allows to change the current user password
 - *Logout*: Closes the session and returns to the login page

1.4 Alert panel

The Alert panel is a pop-over display that conveys important information about the health of the system to the user. When there is an important notification about the state of the system, the alert icon  is displayed in top-left corner of the screen. Clicking it reveals the Alert panel, with all notification messages. Clicking a message navigates to a page with further details on the nature of that particular notice. Notifications can include:

- *Controller link error*
- *Pump blocked or in manual mode*
- *Tank stock below the alert threshold*
- *Unauthorized refueling*
- *Auto. delivery without a price*
- *New auto. delivery*

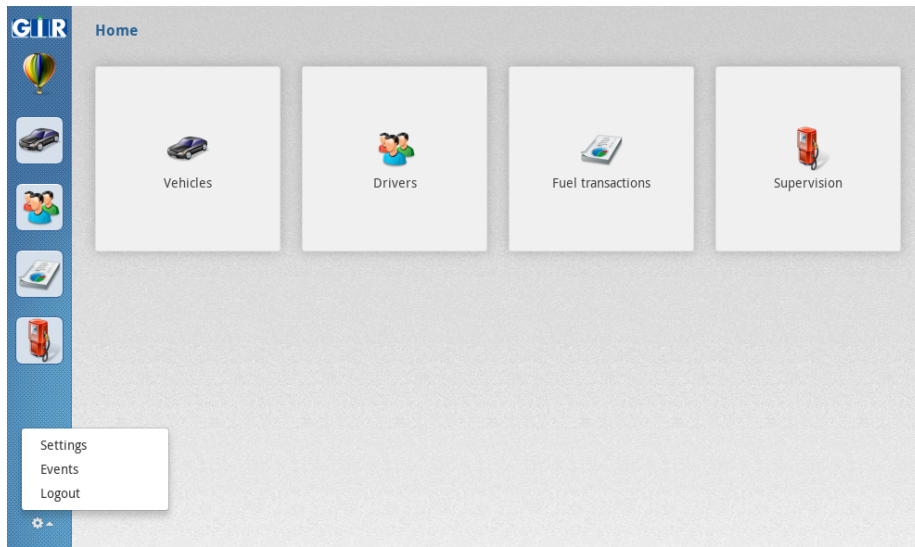
When there is nothing special to report and the Alert panel is empty, the alert icon is not shown.

1.5 Recording your first transaction

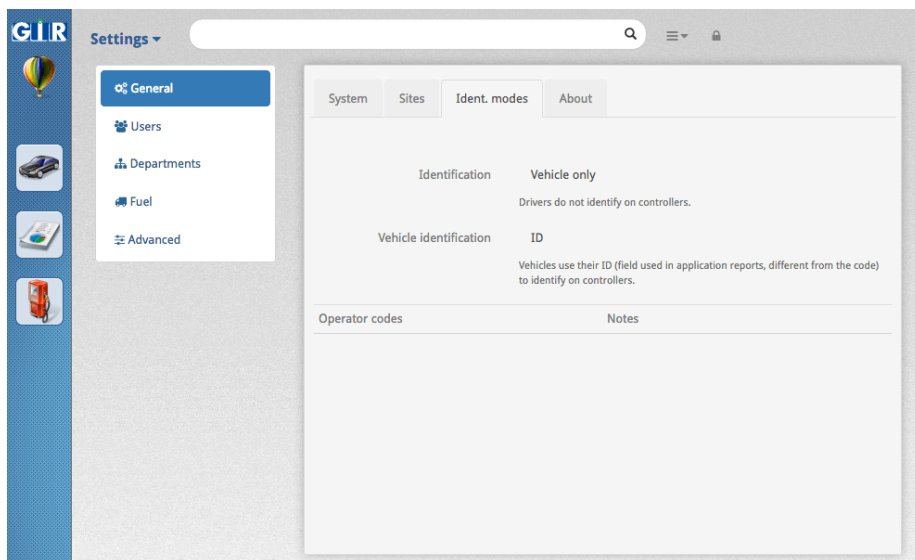
This section will explain how to configure the GIR W250 Server in order to make your first fuel transaction on the controller with a registered vehicle, starting from an empty database.

1.5.1 Set the identification mode

The first thing you will do is to configure how vehicles will identify themselves to the controllers. To do this, click on the main menu dropdown at the bottom left corner of the screen and click on *Settings*.



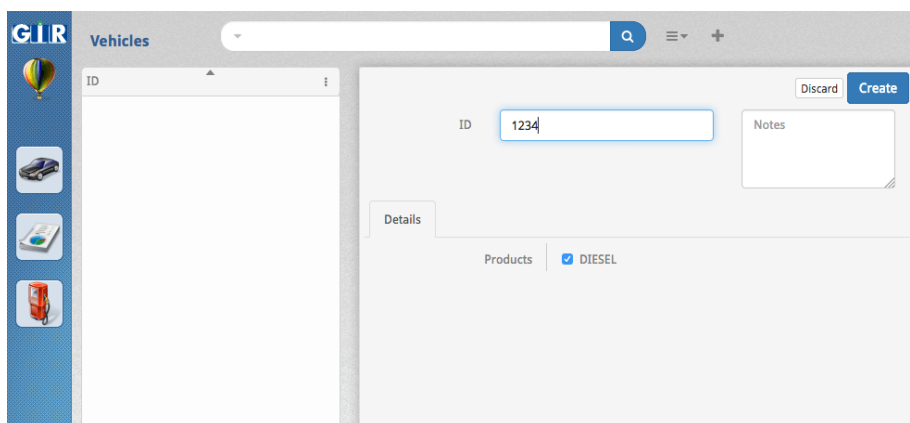
From this page, you will have access to all the W250 application settings (for more information on specific settings, refer to the Settings chapter of this document). Start by selecting the *General* section of the settings page and then select the *Ident. modes* tab within the general settings (see: Settings / Ident. modes for details). Verify that the *Identification mode* is set to *Vehicles only* and that the *Vehicle identification* is set to *ID*. In this configuration, only vehicle information (but no driver) is needed to identify at controllers, and their ID is used as the identification code.



1.5.2 Create a vehicle

You can now create a new vehicle. Click on the *Vehicles* page in the main menu. The list of vehicles in the system will be shown, which should be empty for now. Click on the **+** icon to create a new vehicle. The *ID* field is the only one

that is mandatory, and it will be used to identify this vehicle on the controller (per the GIR W250 application settings). Enter 1234 for the ID. The *DIESEL* product should be selected by default (note that this behaviour is controlled by the *Default product for new vehicles* option for the *DIESEL* product in the Settings). Click *Create* to add this vehicle.



1.5.3 Pair a controller

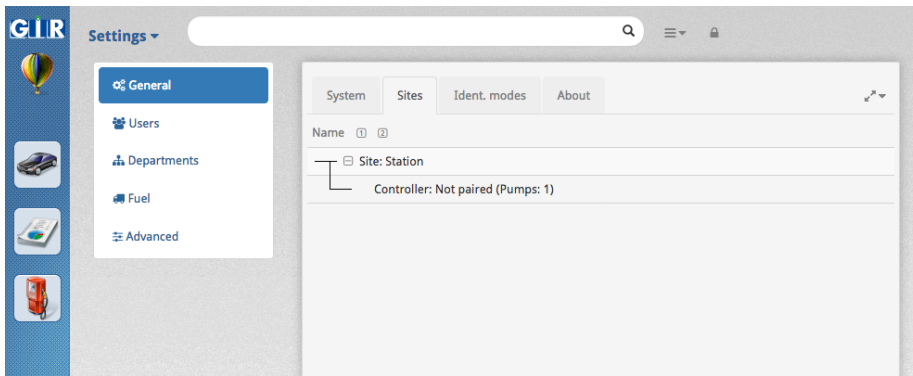
The next step is to pair a controller with your GIR W250 server.

First, ensure that your controller is configured to communicate with your GIR W250 server:

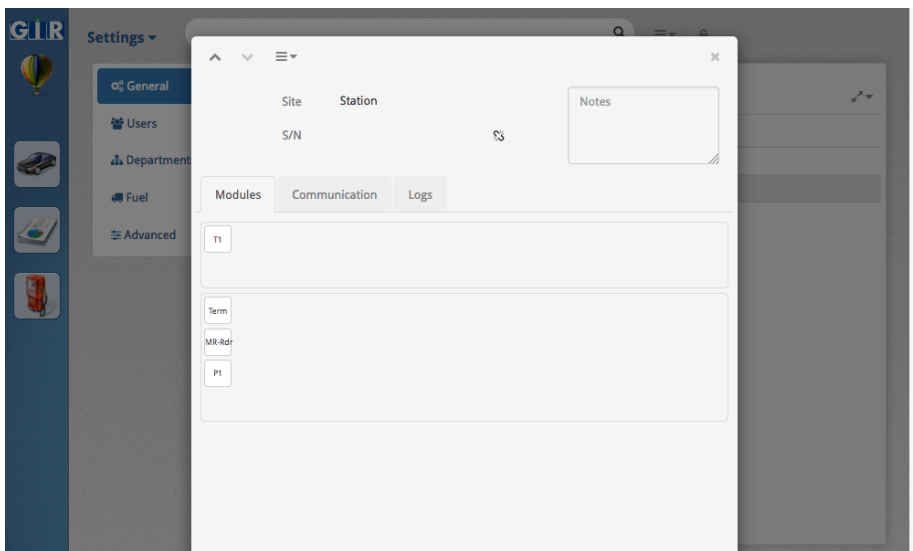
1. Enter the controller setup. On a controller in factory settings, which has no app, just power up the controller and press 9. On a controller with an app installed, set the terminal to address 15, reboot, and press 9 when prompted to do so.
2. Perform a factory reset if needed.
3. Set the communication settings:
 - (a) Go to the *1:LINK* menu.
 - (b) Set *LINK/MODE* to *ETH* OR *GPRS* (press 0 to edit the parameter).
 - (c) Set *LINK/URL/HOST* to the URL of your app (e.g. *my-app-gir.klervi.net*).
 - (d) For *ETH* communication, configure the IP and DNS settings (*LINK/ETH/DHCP*, *LINK/ETH/DNS*, ...).
 - (e) For *GPRS* communication, set the APN and the user/password if needed (*LINK/GPRS/APN*, *LINK/GPRS/USER* and *LINK/GPRS/PASSWD*).
4. Go back to the main menu, press 0 to save the configuration and reboot the controller. Don't forget to restore the terminal address if you changed it. For more information on the controller setup, see the *Configuring communication* section in the *TIP-Vatersay Controller – Reference Manual* document.


Once the controller restarts, it will try to connect to the server configured in *LINK/URL/HOST*. If the connection succeeds, *G+* will show up in the link indicator at the top left of the screen. If the connection fails, *G-* will show up instead: see the *Troubleshooting* section in the *TIP-Vatersay Controller – Reference Manual* document to diagnose and fix communication errors.

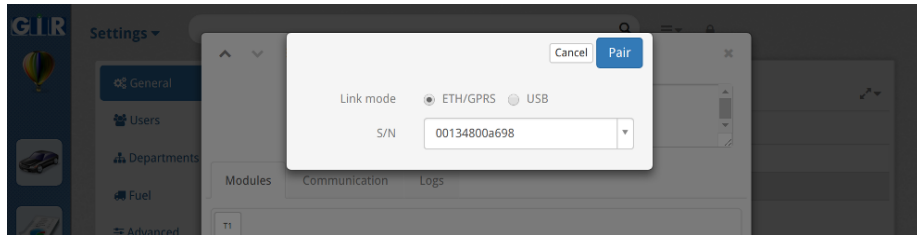
Once the link indicator on your controller shows *G+*, a connection is established between the controller and the server. No more configuration is needed on the controller: the final steps of the pairing process are performed on the server. Go the Settings page, then select the *Sites* tab in the *General* section. For new installations, there should be a default site and a default controller already entered in the GIR W250 server. The site name will appear as *Station* and its controller, labeled by (*Pumps: 1*), should appear below.





Click on the table row containing the controller (*Pumps: 1*) to display the controller editor window. The window contains various configuration settings for the controller with the GIR W250 server. From here the controller modules' configuration can be also updated (if the QSC/RSC code is entered). This window can also be used to pair your GIR W250 server with a controller by entering its serial number (S/N).

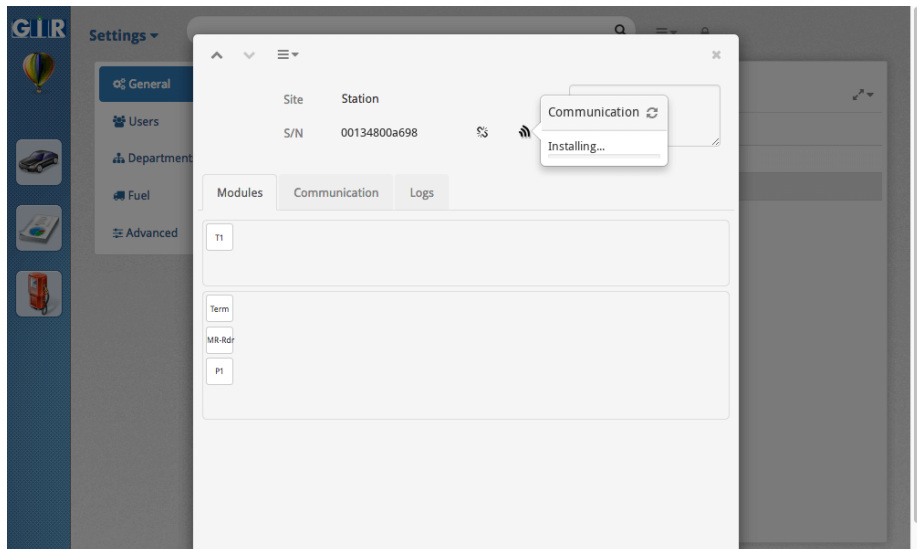


Click on the  icon in the *S/N* field. A new window will be displayed, allowing you to select a serial number from a list. If you don't see any serial numbers, it means that your controller is not correctly connected to the GIR W250 server. Click on *Pair* to begin the pairing process.

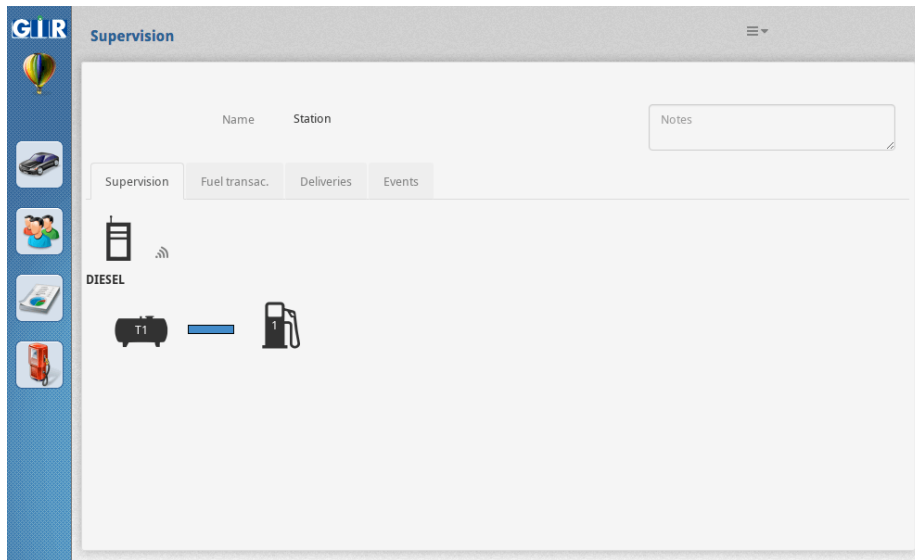


The pairing process will install a new app firmware on the controller if needed (e.g. if the controller was in factory settings and had no app), then synchronize server data with the controller. If you click on the  icon next to the *S/N* field, you'll be able to check the installation status of the controller. Once the  icon is displayed next to the *S/N* field with *A-OK*, your controller is paired and synchronized. This means that you can now start recording transactions.

1.5.4 Record a transaction



Close the controller window and go to the *Supervision* page. This page is automatically updated and can help you monitor in real-time controllers, tanks and pumps on your site.





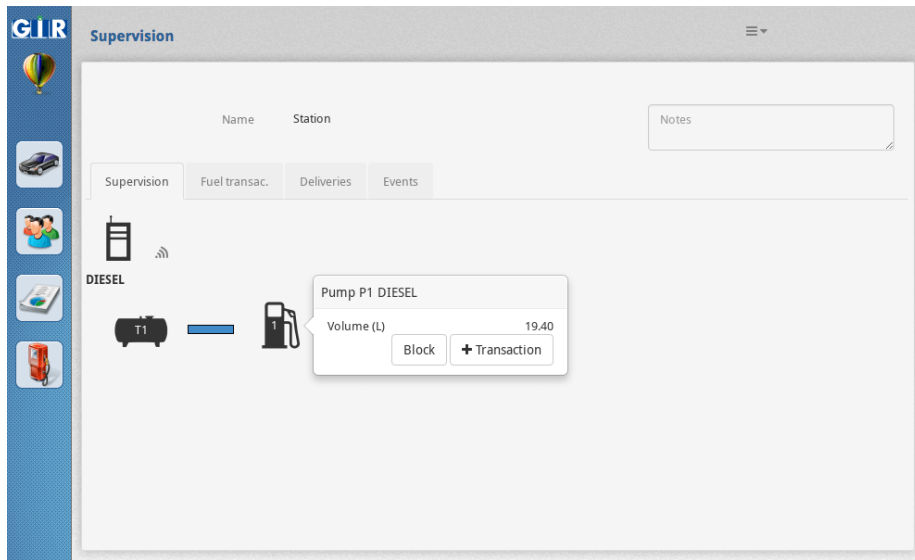
On your controller terminal screen, this is what you should see:

```
Tu 02/01/17 16:07 .+
VEHIC. CODE
```

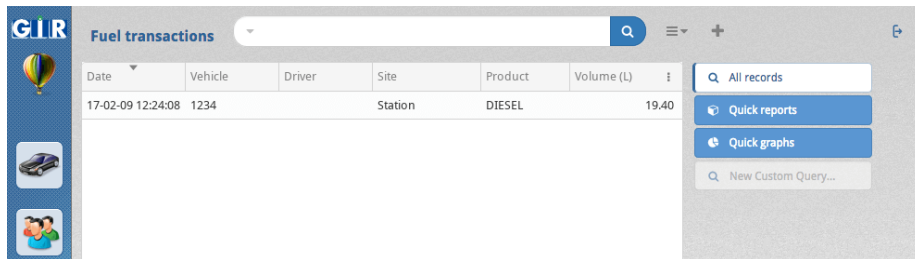
Enter your newly created vehicle code (1234) and validate. You should now be able to start refueling.

Meanwhile, on your GIR W250 Server, you should be able to observe the following from the Supervision page:

- While refueling, if you click on the pump icon  you should be able to see the current volume distributed for this transaction.
- Once the transaction is completed, you should see a new entry in the *Fuel transac.* tab. You can click on the transaction line to see more details.
- If you click on the tank icon , you should be able to observe the volume of fuel product remaining in the tank following subtraction from the recent transaction.



Finally, click on the *Fuel transactions* page in the main menu to see the transaction history. You should see an entry in the table for transaction that you just completed. By clicking on *Quick reports* you can generate fuel consumption reports by vehicle (see: Fuel transactions / Quick reports). By selecting *Quick graphs* you can produce similar reports in graph form (see: Fuel transactions / Quick graphs).

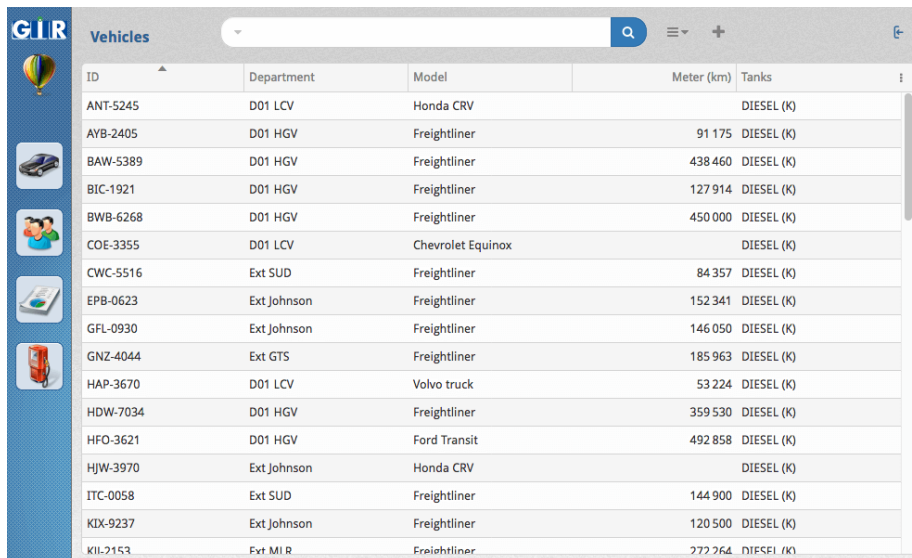


Chapter 2

Common user interface elements


2.1 Collection view

Most pages (such as the vehicles, drivers and fuel transactions pages) are organized around a collection of records and related elements – together referred to as the *Collection view*.


















ID	Department	Model	Meter (km)	Tanks
ANT-5245	D01 LCV	Honda CRV		DIESEL (K)
AYB-2405	D01 HGV	Freightliner	91 175	DIESEL (K)
BAW-5389	D01 HGV	Freightliner	438 460	DIESEL (K)
BIC-1921	D01 HGV	Freightliner	127 914	DIESEL (K)
BWB-6268	D01 HGV	Freightliner	450 000	DIESEL (K)
COE-3355	D01 LCV	Chevrolet Equinox		DIESEL (K)
CWC-5516	Ext SUD	Freightliner	84 357	DIESEL (K)
EPB-0623	Ext Johnson	Freightliner	152 341	DIESEL (K)
GFL-0930	Ext Johnson	Freightliner	146 050	DIESEL (K)
GNZ-4044	Ext GTS	Freightliner	185 963	DIESEL (K)
HAP-3670	D01 LCV	Volvo truck	53 224	DIESEL (K)
HDW-7034	D01 HGV	Freightliner	359 530	DIESEL (K)
HFO-3621	D01 HGV	Ford Transit	492 858	DIESEL (K)
HJW-3970	Ext Johnson	Honda CRV		DIESEL (K)
ITC-0058	Ext SUD	Freightliner	144 900	DIESEL (K)
KIX-9237	Ext Johnson	Freightliner	120 500	DIESEL (K)
KIL-2153	Ext MIR	Freightliner	272 264	DIESEL (K)

These pages generally contain the following elements:


- **Record list:** the scrollable table displaying all of the records of the collection without pagination. Clicking on an item displays its details in the Editor. Clicking on a column header sorts the collection by this column values. Clicking again on the same header will reverse the sorting order.
- **Column manager:** a pop-up window accessed by clicking  in the top-right corner of the Record list then selecting *Edit Columns* in the resulting

drop-down menu. This window allows for showing, hiding or reorganizing of table columns.

- **Editor**: area that appears next to the table that supports viewing and modification of the selected record.
- **Search bar**: input box at the top of the screen used to perform quick searches of the collection by entering keywords and then clicking  or pressing the ENTER key. The results of the query are shown in the Record list.
- **Advanced query builder** : interface for performing complex queries on one or more fields.
- **Record creation mode button** : enters Record creation mode. Displays a form for inputting a new record that is added to the collection once the form is submitted.
- **Action menu** : menu of actions specific to the information currently presented on the screen. Typically contains the following items:
 -  *Info*: displays information about the Record list or the selected record.
 -  *Print*: prints the Record list or the selected record.
 -  *Download CSV*: downloads the Record list in CSV format.
 -  *Download CSV (alt)*: downloads the Record list in an alternative CSV format.
 -  *Clone*: enters Record creation mode using the values from the selected record as a template.
 -  *Online help*: opens the online user manual at the relevant section for this page.
- **Delete button** : deletes the selected record.
- **Navigation controls**  : switches to the previous/next record when the Editor is opened.
- **Saved queries panel** : interface for restoring or creating a saved query. To create a new saved query:
 - filter the collection using the Search bar or the Advanced query builder
 - expand the Saved queries panel by clicking 
 - enter a label in the *New Custom Query...* input
 - click on *Add (+)*
 To restore a query, simply click the corresponding label in the Saved queries panel.

It is possible to share a query between all users by setting the *Visibility* field to *All users*.

The Editor panel typically contains the following elements:

- **Close button** : hides the editor panel and displays the Record list in full screen.



- **Save / Create button** (*Create* in Record creation mode, *Save* otherwise): Validates and saves the record data. This button is displayed once the user makes a change to a field.
- **Cancel / Discard button** (*Discard* in Record creation mode, *Cancel* otherwise): Cancels the changes to the record and closes the editor. Displayed alongside the Save / Create button.

2.2 Compact lists



Within the Editor view of a record you may encounter additional lists of items similar in appearance to the Record list in the Collection view. These are Compact lists. They are used to manage lists of data within records. The main difference compared to Record lists is that selecting an item in a Compact list will display an editor for that item in a pop-up window, rather than in an adjacent panel.





Details		Fuel transac.			
Date	Driver	Site	Volume (L)	Meter	+
17-02-08 00:14:...	William Wilson	San Francisco, ...	DIESEL 39.73	279 617	▲
17-02-04 00:02:...	William Wilson	New-York, NY	DIESEL 71.35	279 617	
17-02-02 04:05:...	William Wilson	New-York, NY	DIESEL 36.72	279 617	
17-01-28 00:05:...	William Wilson	San Francisco, ...	DIESEL 44.91	279 617	
17-01-26 00:44:...	William Wilson	Houston, TX	DIESEL 39.10	279 617 3.3	
17-01-23 23:57:...	William Wilson	New-York, NY	DIESEL 35.93	276 821 27.2	▼

The Compact list generally contains the following elements:

- **Item list**: a scrollable table. Clicking an item in the list displays an editor for that item in a new window.
- **Item creation mode button** : if enabled, enters item creation mode. Displays an editor in a new window for adding a new item to the list.
- **Expand list button** : if present, navigates to a full Collection view page for the given items.

The item editor window normally contains the following elements:

- **Close button** : closes the editor window.
- **Save / Create button**: Validates and saves the data.
- **Cancel / Discard button**: Cancels the changes to the item and closes the editor window.
- **Delete button** : deletes the current item and closes the editor window.

- **Action menu** : menu of actions specific to the current item. Usually contains the following items:
 -  *Info*: displays information about the current item.
- **Navigation controls**  : Switches to the previous/next item in the list.

2.3 Unit switching

When the *Km/Miles* feature of GIR W250 is enabled (see: Specific features), all odometer values can be dynamically displayed in either kilometers or miles. To switch between the two modes, simply click on the *Km* or *Mi* button in the bottom-left corner of the screen.

- In *Km* mode, fuel consumption is expressed in litres per 100 km (L/100)
- In *Mi* mode, fuel consumption is expressed in miles per gallon (Mpg)

Chapter 3


Settings

Application settings are accessible through the main dropdown menu in the bottom-left corner of the screen. The settings page is organized into 5 sections:

- *General*
- *Users*
- *Departments*
- *Fuel*
- *Advanced*

The search bar at the top of the page can be used to quickly find a specific setting by keyword.



3.1 QSC/RSC protection

Some sensitive settings (like *Sites*, *Ident. modes* and *Products*) are read-only until a valid QSC/RSC code is entered by clicking on the lock icon: . This is intended to restrict modification of these protected settings to authorized persons possessing the proper code.

3.2 General

3.2.1 Sites

Manages sites and controllers. Protected by a QSC/RSC code.

To create a new site, click the  button in the top-right corner of the table. A corresponding controller for the site will also be created automatically at the same time. To create another controller assigned to that site, click the  button at the right of the site name.

Site editor:

- **Name:** the site name.

- **GPS Coordinates:** latitude and longitude values separated by a comma (,). They are used to display the site in the map in the Supervision page (the map is only displayed when there is more than one site). In SaaS mode, clicking the gear icon shows a map preview, and can automatically get approximate coordinates from controller technical data, when available on Mobile or IP networks.

Controller editor:

- **S/N:** used to pair or unpair the controller to a device using its serial number. See the Controllers chapter.
- **Modules:** shows the modules (pumps, terminals, readers, etc.) used by the controller and allows for their customization.
- **Communication:** displays communication events for this controller. Also contains controls to perform manual actions on the controller such as upgrade, reboot, or diagnostics (see Controllers / Communication).
- **Logs:** displays technical logs for this controller.

When at least 2 controllers are defined on a same site, the pairing window for these controllers displayed when clicking on the **S/N** field contains a *Settings* tab with the following field:

- **Label:** controller optional custom label. When not empty, this value is displayed below the controller icon in the supervision page. It is also used as a suffix everywhere controller are displayed.

The controller modules editor allows to define multiple fuelsets by clicking on + then **+Fuelset** or multiple accesses by clicking on + then **+Access**.

The controller modules editor allows the configuration of the following items:

- Tanks.
- A terminal.
- Readers.
- Pumps.
- Pump readers (reader attached to a pump).
- A receipt printer. Available on kvgca 2.0.9 or higher versions. Adding a receipt printer will add the *Receipt* tab in the *Fuel* section of the settings (see below). It will also add a *Print receipt* action in the fuel transaction editor: this action displays a window containing the receipt content for this transaction, which allows to get the receipt if the print failed.
- Gauges.

Accesses also allow to configure the following modules:

- Readers

- A command relay

Tank editor:

- **Site:** site where the tank is located. As tanks are only referenced by a site, they are shared across all controllers in this site
- **Number:** number used to display this tank (ex: T1, T2...).
- **Product:** product delivered to this tank.
- **Capacity:** total volume the tank can hold.
- **Options:** tank options.
 - *Auto. deliveries:* displayed if the tank capacity is more than zero. This option is read-only: it shows whether automatic deliveries are enabled or not. Automatic deliveries are enabled when a tank has gauges, so this option can be “checked” by defining a gauge for the tank.
 - *Manual deliveries:* displayed if the tank capacity is more than zero. If enabled, it displays the *Theoretical volume* field, and allows to enter manual deliveries for this tank. Manual and automatic deliveries are fully separated: a tank can have either no deliveries, only manual deliveries, only automatic deliveries, or both. GIR recommends to keep manual deliveries disabled when a tank has gauges, as having both manual and automatic deliveries can cause user confusion.
 - *Unit price:* displayed if the tank capacity is more than zero, and if the *Auto. deliveries* or *Manual deliveries* option is set. If enabled, it displays the *Unit price* field. Tank price management is exclusive between automatic and manual deliveries: if manual deliveries are enabled, they manage tank price (see Supervision / Manual deliveries). Otherwise, tank prices are managed with automatic deliveries (see Supervision / Automatic deliveries unit price).
- **Theoretical volume:** current theoretical volume of the tank. This value is decreased by fuel transactions and increased by manual deliveries, but it can also be modified manually.
- **Gauge volume:** current volume of the tank as reported by gauges. This value is automatically updated when inventories are added.
- **Alert threshold:** if the tank current volume goes below this value, it displays an alert on the alert panel, and sends an email alert to users that have enabled the *Tank volume below alert threshold* notification. Available if either *Theoretical volume* or *Gauge volume* are displayed. If both fields are displayed, the alert threshold uses the theoretical volume.
- **Block threshold:** if the tank current volume goes below this value, it automatically blocks the tank pumps with the *Tank blocked* reason. Once pump are blocked, they can be unblocked with the existing pump unblocking methods. When a pump is unblocked in this situation, the pump won't be automatically blocked because of the tank stock until its current volume goes above the blocking threshold. Available only if *Gauge volume* is displayed, doesn't use *Theoretical volume*.

- **Unit price:** defines the current unit price for this tank. Usually updated automatically, but can be modified manually as well. If the *Auto. deliveries* or *Manual deliveries* options are enabled, this value is automatically updated each time a price is set on the newest delivery.

A list of all tanks can also be accessed through the “Tanks” link near the “Settings, Sites” tab, or from the Supervision page. In the list of tanks, extra fields are available:

- **Theoretical hullage:** amount of fuel that can be delivered into the tank, given the tank capacity and theoretical volume.
- **Gauge hullage:** amount of fuel that can be delivered into the tank, given the tank capacity when a gauge volume is available.
- **Gauge date:** date and time at which the gauge volume was last updated.

Terminal module:

- **Device:** available values: *TIP terminal*.
- **Address:** RS485 address.

Reader module:

- **Device:** available values: *EMG Terminal*, *TLG Terminal*, *MR-Access*, *TIP Terminal (keypad)*, *MR-Access (keypad)*, *Passive RS-232*, *ISO2*, *iButton*, *iButton (legacy M232-i)*, *Tacho*, *GemProx* (depends on the features enabled, see Specific features).
- **Address:** RS485 address (*EMG*, *TLG* or *GemProx*).
- **Bus:** RS232 address (*Passive RS-232*, *ISO2*, *iButton* or *Tacho*).
- **NCE code prompt:** Only displayed for *Passive RS-232*, *MR-Access* and *ISO2* readers if the NCE code prompt is enabled. If enabled, this reader will be interrogated when an NCE code is prompted on the terminal. If the reader reads a code, it will be used for the fuel transaction.
- **Advanced script:** Only displayed for *MR-Access (keypad)* readers. Allows to select an existing advanced keypad configuration, or to create a new configuration. This configuration is then used to decode the code read from the reader.

Pump module:

- **Device:** available values: *MR-Pompe*.
- **Number:** the pump number. The pump will be displays as “Px” where x is the pump number. Selecting a pump number to x will automatically set the RS485 addresss to x-1.
- **Product:** product distributed on this pump.

- **Tank:** tank associated to this pump. When transactions made on this pump are processed, the associated tank stock is decreased.
- **Timeout (before):** time after which a transaction is stopped if no pulse has been detected since the pump was commanded (See Controllers / Controller usage / Pump timeouts).
- **Timeout (after):** time after which a transaction is stopped if no pulse has been detected, after at least one pulse occurred (See Controllers / Controller usage / Pump timeouts).
- **Pulses/L:** number of pulses per litre for the counting device.
- **Options:**
 - *Block after 3 null transactions:* if set, automatically blocks the pump after 3 successive null transactions.
 - *Record unauthorized refuelings:* if set, when an unauthorized refueling is detected, a transaction is stored with its date, duration and volume.
- **Mode:** pump mode. Available values: *Simple (C1)*, *Double (C1=C2)* and *Addition (C1+C2)*.
- **Hangup:** pump hangup setting. It defines if a transaction ends when the RP input of a pump module changes. Available values:
 - *Auto:* the open/closed direction is automatically detected during the first transaction following an electrical restart.
 - *On opening:* transaction ends when the RP input changes to open (RP:0→1).
 - *On closing:* transaction ends when the RP input changes to closed (RP:1→0).
 - *None:* no hangup, transaction ends according to other criteria on volume and duration.
- **Address:** RS485 address.
- **Max duration:** Maximum duration of transactions for this pump, in minutes.

Pump reader module:

- **Device:** available values: *BS125* (if the *BS125 badges* feature is enabled, see Specific features), *RPK* (if the *RPK badges* feature is enabled, see Specific features).
- **Channel:** selects the device input (*RPK* devices only).
- **Bus:** RS232 bus.
- **Identification:** determines how the reader is used for identification. Possible values:

- *With terminal*: the reader requires terminal interactions to identify badges and start transactions
- *Without terminal*: the reader doesn't require terminal interaction to identify badges and start transactions
- *Both*: the terminal can be used with or without terminal interaction
- **Start (s)**: Time in seconds after which the transaction without terminal start from the moment where a same badge is first detected by the reader (only displayed if the *Identification* field is not set to *With terminal*). If set to empty, the transaction starts as soon as a badge is detected.
- **End (s)**: Time in seconds after which the transaction without terminal stops from the moment when the badge is no longer detected by the reader (only displayed if the *Identification* field is not set to *With terminal*). If set to empty or zero, the transaction don't stop if the badge is no longer detected by the reader, or if another badge is read.
- **Hangup control**: Nozzle hangup control option (only displayed if the *Identification* field is not set to *With terminal*). When this option is enabled, the transaction without terminal only starts after a successful identification if the pump RP input detects that the nozzle is released, which only works if the pump *Hangup* setting is not set to *Auto*.

Receipt printer module:

- **Device**: available values: *Hengstler, Epson, Generic*.
- **Bus**: RS232 bus.

Gauge module:

- **Device**: available values: *4-20 mA, 0-5 V, Veeder-Root, i201 height-only, Hectronic, Piusi Ocio, 4tech, Start Italiana, OLE, Horn/Tecalemit, Technoton, Weldann-console and Weldann-probe*.
- **Probe**: selects the gauge output (all devices except *Piusi Ocio* an *OLE*).
 - For *Start Italiana* gauges, the *Probe* field allows to select values from 0 to 99999. This value represents the device serial number.
 - For *4tech* gauges, the *Probe* field allows to select the “tank” channel on the console with address 0.
- **Modbus**: selects the gauge output using this Modbus address (*OLE* devices only).
- **Tank**: tank associated to this gauge.
- **Strapping chart**: strapping chart of the gauged tank, when applicable. Strapping charts are defined in Settings / Fuel / Strapping charts. They contain a name, and a list of height/volume couples (or percent/volume couples for *4-20 mA* and *0-5 V* gauges). Strapping charts are only available for gauges that don't directly return a volume value, i.e. gauges that return just a height or a ratio. For gauges that can return a volume

value, such as *Veeder-Root* or *OLE*, the height-volume translation is typically configured directly on the gauge device console. The *Technoton* and *Weldann-console* gauges can return both a volume or a height value, in that case the volume value is used only if the *Strapping chart* field is not defined.

- **Fuel offset:** offset for the gauge fuel height, only visible for height-based gauges. When defined, this offset is added to the original fuel height value reported by the gauge, before resolving the volume value using the strapping chart. The effective height value, displayed in the application supervision menu or the “gauges” menu on controller, is the result after applying the offset:

$$\text{Effective height} = \text{Original height} + \text{Offset}$$

The offset value can be negative to handle cases where the effective height is smaller than the original height.

- **Water offset:** offset for the gauge water height, only visible for height-based gauges which report a water height value. It is completely equivalent to the fuel offset, but applies to the water height instead of the fuel height.
- **Options – Ignore probe err.:** only available for the *Hectronic* and *Horn/Tecalemit* gauges. When enabled, this option silently ignores errors reported by the probe (ex: temperature errors).
- **Address:** RS485 address (*4-20 mA* and *0-5 V* devices only).
- **Bus:** RS232 bus (all devices except *4-20 mA* and *0-5 V*).

Command relay module:

- **Device:** available values: *MR-Lecteur*, *MR-Access*.
- **Relay duration (s):** duration during which the relay is activated after an identification
- **Address:** RS485 address

The controller modules editor also contain the following fuelset options:

- **T. Vol (s):** defines how long the transaction volume is displayed after the end of the transaction. If set to 0, the volume is not displayed during the transaction as well.
- **Prompt timeout (s):** defines how long the user can input something on the terminal before it timeouts. Possible values: 20 (default), 90 or 180 seconds. Requires at least kvgca 2.2.0.
- **Pump selection:** defines when the pump is selected. Allows to select two values:
 - *Before identification:* pump selection is done before identifying a vehicle and a driver (default value).
 - *After identification:* pump selection is done after identifying a vehicle and a driver. Auto-selects a pump if the vehicle only uses one product and if there is only one pump for that product in the fuelset.

- **Vehicle reuse (*min*):** displayed when the identification mode is *Driver + Vehicle*. Allows to quickly reselect the same vehicle as before when the same driver identifies several times on the same terminal. If meters (odometer or hour meter) were entered on the previous identification, they will be reused as well. On the fuel transaction editor, if a vehicle has been reused, a *Vehicle Auto. (Vehicle reuse)* tooltip is displayed at the right of the *Vehicle* field. Same thing for the *Odometer* and *Hour meter* fields. Possible values:
 - *None*: vehicle reuse is disabled (default value)
 - *3*: vehicle reuse is enabled for 3 minutes after the driver ident.
 - *10*: vehicle reuse is enabled for 10 minutes after the driver ident.
- **Disable gauges menu:** disables the access to the gauges menu which was added since kvgca 2.2.0.
- **Force pump selection:** displayed when there is only one pump and the *Pump selection* option is set to *Before identification*. Forces the selection of the pump on the terminal before displaying the identification screen.

The controller modules editor also contain the following access options:

- **Name:** customized access name
- **Mode:** access mode. Possible values:
 - *Punctual*: identifications trigger punctual commands, and store *Granted* access transactions (default value)
 - *Punctual with DS*: same as *Punctual*, but stores additional access transactions depending on the MR-Access DS state: *Forced opening*, *Door left open*, *Door closed* and *Door not opened*
 - *Stay open badges*: identifications start or stop *Stay open* mode for this access. Doesn't trigger any punctual commands
 - *ON/OFF according to DS*: works the same as *Punctual* (triggers punctual commands), but the *Granted* access transactions are replaced by *ON* or *OFF* depending on the MR-Access DS state. Also displays the current *ON* or *OFF* state in the supervision
- **DS:** only added when *Mode* is set to *Punctual with DS* or *ON/OFF according to DS*. Defines how the MR-Access DS state should be interpreted
 - When *Mode* is set to *Punctual with DS*, possible values are *NC* or *NO*
 - When *Mode* is set to *ON/OFF according to DS*, possible values are *ON:Closed* or *ON:Open*
- **Identification:** defines if the access identifies drivers or vehicles

3.2.2 Ident. modes

Configures how vehicles and drivers identify themselves to controllers. Protected by a QSC/RSC code.

- **Identification:** defines which entities must be identified to controllers. Possible values are:
 - *Vehicle only*: the controller only requests vehicle identification. Hides

the driver page and all references to drivers.

- *Vehicle + Driver*: the controller requests vehicle identification, then driver identification.
 - *Driver + Vehicle*: the controller requests driver identification, then vehicle identification.
 - *Vehicle + Optional driver*: the controller requests vehicle identification, then driver identification only if the *Driver identification* option is enabled in the vehicle editor.
 - *Account only*: same thing as *Vehicle only*, but replaces the *Vehicle* label as *Account* in the application interface. Changes the controller identification prompt label to "BADGE", "CODE" or "ID."
- **Vehicle identification**: defines the identification method for vehicles. Possible values are:
 - *ID*: Vehicles use their ID (the field used in application reports – different from the code) to identify to controllers.
 - *Code (hidden)*: vehicles use their code (different from the ID used in application reports) to identify to controllers. The code won't be displayed on the terminal.
 - *Code (visible)*: vehicles use their code (different from the ID used in application reports) to identify to controllers. The code will be displayed on the terminal.
 - *EMG badge*: vehicles use EMG badges to identify to controllers.
 - **Driver identification**: same thing as vehicles except that the *ID* method is replaced by *Name* in which drivers use their name to identify to controllers.

Operator badges / Operator codes: at the bottom of the Ident. modes tab is a list of special codes or badges. They can be used to replace a normal identification. When an operator badge or code is used on the terminal, it prompts for a vehicle code. The transaction will then be assigned to the selected vehicle (see: Controllers / Operator identification).

- **Badge**: operator badge (when vehicles or drivers are identified with a key).
- **Code**: operator code (only when both vehicles and drivers are identified with a code).

Additional ident. modes parameters can be accessed by clicking on the gear icon at the right of the *Identification* field. This opens a window containing the following tabs:

- **Custom labels**: allows to define custom labels for vehicles and drivers in the user interface.
- **Custom prompts**: allows to customize the prompts messages displayed on the controller terminals.
- **Advanced**: advanced ident. modes parameters.

Custom labels

This tab makes it possible to define custom labels for vehicles and drivers in the user interface. Allows to choose an ordered selection of the following fields:

- **ID** (Vehicle-only)
- **Name** (Vehicle-only)
- **First name** (Driver-only, if the *Name + First name* feature is enabled)
- **Badge**
- **Code**
- Custom properties with the *Text* type if there are any for this entity (if the *Custom properties* feature is enabled)
- **Department**: available if at least one department is defined in the settings. Adds the department label to the vehicle or driver custom label. If this value is ordered in the first position, the *Department* column in the vehicles or drivers page is moved a the first position, and the default collection sort is ordered by departments first.

Custom prompts

This tab allows to customizes the prompts messages displayed on the controller terminals. The following messages can be customized:

- "ID VEHICLE": displayed when a vehicle needs to be identified.
- "ID DRIVER": displayed when a driver needs to be identified.
- "PIN CODE": displayed when a vehicle/driver PIN code needs to be validated.
- "PUMP": displayed when a pump need to be selected (only displayed when fuelset "Pump selection" parameter is set to "After identification").
- "READY TO PUMP": displayed when a fuel transaction starts.

Advanced

It is possible to enable multiple identification modes and then select one individually for each driver and vehicle by using the *Multiple Ident. Modes* feature.

If at least two identifications modes are defined for an entity (vehicle or driver), this adds the following options to the *Advanced* tab of the additional ident. modes parameter window:

- **2 identifications per vehicle**
- **2 identifications per driver**

Refer to the section Specific features / Multiple Ident. Modes for details.

If at least two identifications modes are defined for an entity (vehicle or driver), the following options will also be available in in the *Advanced* tab:

- **Custom ident.modes-readers association:** This option allows to restrict identification modes to some readers. It adds:
 - A *Type* field in the reader settings in the controller module editor
 - A *Readers* field in the advanced badge settings window, which allows to select the readers targeted by this identification mode.

If the *Identification* field is set to *Driver + Vehicle*, the following option will also be available in the *Advanced* tab:

- **Skip vehicle id when unambiguous:** When enabled, allows to automatically skip the vehicle identification on the controller if the driver is only authorized to use one vehicle (requires at least kvgca 2.2.0).

If the default ident. mode for vehicles displays a *Code* field:

- **New vehicles: Random code:** when not empty, creation form for vehicles initializes the *Code* field to a random and unique value. Allows to select the number of digits of the generated value.

If the default ident. mode for vehicles displays a *PIN Code* field:

- **New vehicles: Random PIN code:** when not empty, creation form for vehicles initializes the *PIN code* field to a random value. Allows to select the number of digits of the generated value.

If the default ident. mode for drivers displays a *Code* field:

- **New drivers: Random code:** when not empty, creation form for drivers initializes the *Code* field to a random and unique value. Allows to select the number of digits of the generated value.

If the default ident. mode for vehicles displays a *PIN Code* field:

- **New drivers: Random PIN code:** when not empty, creation form for drivers initializes the *PIN code* field to a random value. Allows to select the number of digits of the generated value.

3.2.3 System

Defines the language, localization and email settings.

- **SMTP:** SMTP server through which all emails will be sent. When set, a “Test” button at the right of the field allows to test sending an email with this SMTP server, with a detailed error message
- **Email From:** sending address of all the emails sent by the application. Ex: noreply@company.com.
- **Backup directory:** when defined, all the backups generated in `data\backup` will be copied to that directory as well. When it does, a *Backup* audit trail entry is added. It contains *OK* if the copy is successful or *Error:* followed by the error message (available for On-Prem apps only).

- **Public URL:** URL of the application accessible outside of the local network.
- **Language:** default language of all users.
- **Currency:** currency used in the application.
- **Date format:** date format used in the application.
- **Number format:** number format used in the application.
- **Weeks mode:** defines which day is the first of the week.

3.2.4 About

Displays technical information about the application, the server, the maximum number of paired controllers and the current user's web browser.

In the *Support* field there is a *Send data...* button that will send a support file to the support team for further analysis when something is wrong on the application. A support file contains the application database and technical logs. Two options are available: either send the full database (which could take some time depending on the database size), or send a partial database (which only contains the most recent transactions and technical logs).

Once a support file is generated, you can track the sending progression and optionally download the support file to send it by another way. From there, you can also cancel the sending or send a new support file.


3.3 Users

3.3.1 User editor

- **Email:** email address used by the user to log in to GIR W250 and by the system to contact the user.
- **Authentication:** available for SaaS applications only: allows to select the identification method for this user. If *Password* is selected, the user has to connect using the email and password credentials. If a Single Sign On (SSO) service is selected, then the user doesn't use a password, but his account on this service. After the user is created, an icon next to this field allows to send an email to this user containing the application URL and the user credentials. Possible values:
 - *Password:* the user uses password to login (default value).
 - *Microsoft:* the user uses his Microsoft account to login. Displays a *Login with Microsoft* link on the login page, redirecting the user to the Microsoft login page.
 - *Google:* the user uses his Google account to login. Displays a *Login with Google* link on the login page, redirecting the user to the Google login page.
- **Password:** password the user uses to log in. Displayed when *Authentication* is set to *Password*.

- **Name:** family name of the user.
- **First name:** first name of the user.

- **Level:** defines the authorization level.
 - *Admin:* the user can modify all data, users and system settings.
 - *Manager:* the user can modify all data and other settings.
 - *Limited manager:* the user can modify all data but no settings.
 - *Consult:* the user has read-only access to all data.
 - *Custom:* the user has a customized set of permissions. Note that if no permissions are set for this user that they will have access to no vehicles or transactions. See **Permissions tab** below.

Notifications tab: defines notification emails for users. Click on  to add a new rule. Note that in order to send emails, the *SMTP* setting must be defined in General / System, so this tab is not visible if this setting is empty.

- **All sites:** the selected notifications will be sent for all sites.
- **Select a site:** the selected notifications will be sent only when they concern a specific site.
- **Controller link:** notify when controllers go online or offline.
- **Pump block:** notify when pumps are blocked or unblocked.
- **Manual pump mode:** notify when pumps switch to manual or automatic mode.
- **Tanks stock:** notify when a tank current volume goes below the alert threshold.
- **Unauthorized refueling transactions:** notify each time that an unauthorized refueling is detected.

Permissions tab: is only visible if the *Level* parameter in the *Details* tab is set to *Custom*. Defines the access permissions of the user for a particular site, department or generally. Note that these permissions are cumulative in that a user having general read and write permission and a second read-only permission entry to a specific department will still be able to write to that department.

Permissions are associated with one of following scopes:

- **General:** The permission is granted application-wide (all sites and departments)
- **Site:** The permission is granted for a specific site. If the *Regions* feature is enabled and if the selected site is a region, then the permission will apply to the region and all of the sites of this region
- **Department:** The permission is granted for a specific department. If hierarchical departments are enabled, then the permission will apply to the department and any of its descendants

- **Zone:** The permission is granted for a specific zone (only if the *Zones* feature is enabled)

The *General* scope contains the following permissions:

- **View+Edit Users:** allows to edit users in the settings.
- **Edit General / System:** allows to edit the General / System tab in the settings
- **Edit Advanced / Impexp:** allows to edit the Advanced / Impexp tab in the settings
- **Edit other settings:** allows to edit all the other settings
- **View Vehicles:** allows to view all vehicles
- **Edit Vehicles:** allows to edit all vehicles
- **View Drivers:** allows to view all drivers
- **Edit Drivers:** allows to edit all drivers
- **View supervision:** allows to view all supervision pages
- **Perform superv. actions:** allows to execute actions in all supervision pages
- **Edit Deliveries:** allows to edit all deliveries and update all tanks *Volume* and *Unit. price* fields
- **Edit Fuel transactions:** allows to edit all fuel transactions
- **View zones:** allows to filter all zones in the *Vehicles* and *Drivers* pages (only if the *Zones* feature is enabled)
- **Assignment Zone “Everywhere”:** allows to assign all zones to vehicles and drivers, and allows to select *Everywhere* in the *Authorization* field (only if the *Zones* feature is enabled)
- **Default Zone “Everywhere”:** the *Everywhere* value is selected by default in the *Authorization* field when creating a new vehicle or driver (only if the *Zones* feature is enabled)

The *Site* scope contains the following permissions:

- **View supervision:** allows to view the supervision page of the selected site
- **Perform superv. actions:** allows to execute actions in the supervision page of the selected site
- **Edit Deliveries:** allows to edit the deliveries and update the *Volume* and *Unit. price* fields of the selected site’s tanks
- **Edit Fuel transactions:** allows to edit the fuel transactions of the selected site

The *Department* scope contains the following permissions:

- **View Vehicles:** allows to view all vehicles derived from this department
- **Edit Vehicles:** allows to edit all vehicles derived from this department
- **View Drivers:** allows to view all drivers derived from this department
- **Edit Drivers:** allows to edit all drivers derived from this department
- **Edit Fuel transactions:** allows to edit the fuel transactions derived from this department

The *Zone* scope contains the following permissions: (only if the *Zones* feature is enabled)

- **View:** allows to filter the zones in the *Vehicles* and *Drivers* pages
- **Assignment:** allows to assign the zones to vehicles and drivers
- **Default:** the zone is assigned by default for new vehicles/drivers

Last activity tab: contains two sections described below. An icon at the top right corner of the tab allows to access to the dedicated pages for each section:

- **Audit trail:** history of modifications done by this user
- **Email logs:** last emails sent to this user

3.3.2 Password recovery

When a user loses his password, there are several ways to recover an access:

- Click the “Forgot password ?” link on the login page, and enter the user email address. This is only possible when the login is an email address, and when the application has the ability to send emails (“SMTP” and “Email from” configured).
- Log in with an admin account, and go to “Settings, Users” to generate a new password for the user.

If none of the methods above are applicable, and you are using a SaaS app, you can contact the tech support, who will restore your access.

If you are using an OnPrem app, a last resort method exists. It requires an access to the server machine:

- If you have desktop access to the server machine, there should be a GIR W250 shortcut on the Windows Desktop. Click it to log in automatically.
- If not, make sure you have access to the application directory on the server hard disk (typically `c:\girw250\data\`).
- Open a web browser on `<app_url>/adminlogin`.
- If you’re on the local machine (localhost), you’ll be logged in automatically.
- Otherwise, you’ll be prompted to enter an “Admin ID” code, which can be obtained by looking for a file named `adminid-*.txt` in the `data\tmp` directory. Filling in the code will log you in with an admin account, which can be used to reset passwords on other user accounts.

3.4 Departments

3.4.1 Department editor

Each vehicle and driver can be assigned to a department. Departments are used for grouping transactions for reporting purposes (see: Fuel transactions / Quick reports).

- **Name:** the department's name.
- **Parent:** the department's parent. This field is only available when the app is configured for more than one level of departments.

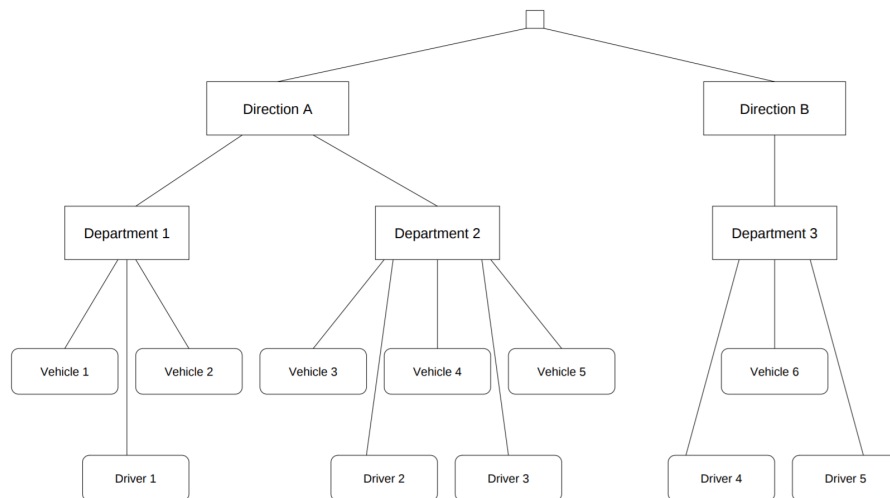
3.4.2 Levels

Up to two additional levels of hierarchy can be used for organizing departments for reporting purposes (for a total of up to three levels of organization). The **Levels** input defines the total number of organizational levels required (1–3). The labels defining each level of this hierarchy can also be customised here – by default they are *Division*, *Direction* and *Department*.

Vehicles and drivers can only be assigned to departments at the bottom of the hierarchy (ie. to a “*Department*” in the default three-level scenario). However user permissions can be scoped to any level.

Note that when increasing the number of organizational levels, all existing departments are automatically nested below a new top-level entry (named “-”). Equivalently, reducing the number of organizational levels removes the top-level entry and is only supported when there is a single top-level entry is present.

The following chart shows an example of an organizational hierarchy with 2 levels:



3.5 Fuel

3.5.1 Products

Manages all the fuel products used in the application.

You can manually order products by clicking on the *up* or *down* icons to the left of the product name. This will define the default order each time products are displayed.

Product editor:

- **Name:** the product name.
- **Type:** the product type. Possible values are:
 - *Fuel main engine* (e.g. *Diesel, Unleaded, Super, etc.*)
 - *Fuel secondary engine* (e.g. *Red Diesel*)
 - *Additive* (e.g. *DEF*)
 - *Other*
- **Default for new vehicles:** if *Authorize by default* is checked, this product will be selected by default each time a new vehicle is created.

3.5.2 Models

Manages vehicle models. Models assist in grouping vehicles with similar characteristics. Note that if the *Advanced vehicle options* is enabled under *Settings / Advanced / Features* then all of the options and product configurations below must be defined for each individual vehicle rather than by model. They will be hidden in the Model editor. Refer to the section *Specific features / Advanced vehicle options* for details.

Model editor:

- **Name:** the model name.
- **Options:**
 - *Activity code:* if set, vehicles of this model will have to select an activity before each transaction.
 - *NCE (Non Checked Entry) code:* if set, vehicles of this model will have to enter a code before each transaction.

A separate tab is shown for each of the product types:

- **Products:** shows the products belonging to the product type (Fuel main engine, etc.).
- **Capacity:** defines the vehicle tank maximum capacity for this product
- **Vol. max every (h):** if **Capacity** is defined, limits how often a vehicle can take this amount of fuel.
 - *Unlimited:* A vehicle can refuel indefinitely
 - *1h - 24h:* A vehicle can take at most **Capacity** during the specified period

- **Meter(s)**: configures the metering policy for vehicles of this model that use the specified products:
 - *None*: no meter is required.
 - *km/mi*: odometer entry (km/mi) is required before starting a transaction.
 - *h*: hour meter entry (hours) is required before starting a transaction.
 - *km/mi + h*: both odometer (km/mi) and hour meter (hours) entries are required.
- **Odometer** (km/mi): optional restriction on odometer values (this setting is displayed only when km/mi or km/mi + hours meter is required).
 - *without control*: all values are accepted.
 - *range*: limits accepted odometer values to within *range* above the previously-recorded value. A value outside of this range can be forced if entered twice.
 - *range (strict)*: limits accepted odometer values to within *range* above the previously-recorded value. A value outside of this range can not be forced.
- **Meter** (h): optional restriction on hour meter values.

3.5.3 Additional prompts

This tab configures which additional prompts are enabled for drivers or vehicles. It is protected by QSC/RSC code.

Available additional prompts:

- **NCE code**: Non-checked entry code
- **Activity**: Allows to select an activity code in a list

If *NCE code* is enabled, it displays the “*NCE code*” text field which allows to customize the “NCE code” label.

If *Activity* is enabled, it displays the “*Activity*” text field which allows to customize the “Activity” label. It also displays a list at the bottom of the tab which configures the activities that can be selected when an activity code is required on the controller.

NCE code validation and extraction

If *NCE code* is enabled, a gear icon is displayed at the right of the “*NCE code*” text field. Clicking on this icon opens a window containing an editor for a lua script. If this script is defined, it can extract and validate the NCE code inputs in the following contexts:

- Manual fuel transactions
- Controller terminals
- Controller readers (with the *NCE code prompt* option enabled)

This script editor also contains:

- A *Test* button to check if a text input is a valid NCE code or not, and what is the extracted value

- A *Logs* tab containing *Test* and *Production* logs

If this script is defined, it enable the following changes:

- When creating or updating a manual fuel transaction, if the entered *NCE code* value is not valid according to the script, this displays an *Invalid NCE code* error message
- When entering a NCE code from a controller terminal or reader, if the value is not valid according to the script, this stores an *Invalid NCE code* event for this controller

Activities

Activity editor:

- **Name:** the activity name displayed in reports.
- **Code:** the activity code used in controllers.

3.5.4 Suppliers

This tab is displayed when at least one tank has manual or auto deliveries enabled. Allows to define suppliers for manual and auto deliveries. When at least one supplier is defined, this adds a *Supplier* field in the manual and auto deliveries editors.

Supplier editor:

- **Name:** the supplier name.
- **Notes:** additional text field for comments.

3.5.5 Strapping charts

This tab is only displayed if a controller has defined a gauge module that needs a strapping chart. It is possible to connect digital readout gauges to tanks in order to monitor in real-time the amount of fuel that is left in a tank in the Supervision page. Some gauges output units other than volume, such as the height of the fuel level. For such cases, a strapping chart is used: a table of values to convert gauge values (mm/inches) to volume (litres/gallons). These lookup tables are managed here.

The strapping chart editor (used to create or edit a strapping chart) has the following elements:

- **Name:** strapping chart label.
- **Type:** type of units output by the gauge — *mm* (*Piusi OCIO, Hectronic, ...*) or *%* (*4-20mA, 0-5V*).
- **Table:** spreadsheet interface for editing the strapping chart data.
- **Graph:** graphic representation of the strapping chart data. This is useful to spot incorrect values.
- **CSV Data:** text input box for direct editing of strapping chart data in CSV format.

3.5.6 Receipt

This tab manages the receipt settings for this application. Receipts can be delivered using two methods:

- Using receipt printers defined in the controllers modules configurations.
- Or using email sending by defining a script in the *Email receipt* field.

When receipt printers are defined in the controller configuration, those settings are split in two sections: *Printer* and *Email*.

Printer

This section is only displayed if a receipt printer is defined in at least one controller modules configuration. It contains receipt printing settings that apply on all controllers.

Available printing settings:

- ***Number of copies***: defines how many receipts will be printed each time a print is required.
- ***Print mode***: configures the printing strategy. Either receipts are printed on each transactions for all vehicles, or there is a prompt that asks drivers if they want to print the receipt or not.
- ***Format***: selects the content of the receipt printed. If *Custom* is selected, it allows to customize the receipt content with a script in lua. In the script form, a *Test* button allows to test the lua script against the latest transaction received by previewing the receipt content for this transaction.

Email

This section contains a single *Email receipt* field. If left to *Disabled*, no email receipts will be sent. When set to *Custom*, a gear icon gets visible at the right of the field. When clicked, this icon opens a new window for editing the email receipt script in lua.

The email ticket script can generate an email every time a fuel transaction is created or modified. The script can be used to define:

- If an email should be sent or not for all transaction creation or modification.
- The email recipients.
- The email contents, which can be in plain-text or in HTML.

This script editor also contains:

- A *Test* button to test the lua script against the latest transaction in the database, and preview the email that would be sent.
- A *Logs* tab containing *Test* and *Production* logs.

When the email ticket script is defined, all new transactions creations or updates may automatically send emails based on the script configuration. The emails that were sent by the application can be found in the *Email logs* link in the action menu of the *Receipt* settings tab.

Defining a script also adds an *Email receipt* action in the fuel transaction editor. When selected, this action displays an email preview with the latest transaction data. The email recipients can be edited, and a new email can be sent using the *Send* button.

3.6 Advanced

3.6.1 Audit trail

Lists the activity history of all the application users. The audit trail can be used to track down changes to a specific record or by a particular user.

3.6.2 Features

This tab manages specific features detailed in the chapter Specific features. It is protected by QSC/RSC code and is hidden until successful code entry.

3.6.3 Impexp

The options in this tab configure how fuel transactions are exported to other third-party software. For more information on this subject, please refer to the Export appendix.

- **Impexp API key:** API key allowing to use the impexp API web services. A gear icon at the right of the field opens an advanced window containing the following parameters:
 - **Impexp API key:** allows to generate a new API key or reset a previous value
 - **Authorization:** defines the permissions of the web services using this API key. Possible value:
 - *Read/Write:* the impexp API web services can view and edit records (GET, PUT, POST and DELETE methods)
 - *Read-only:* the impexp API web services can only view records, but edition is forbidden (only GET method)
 - *Disabled:* the impexp API web services are all disabled
- **Format:** defines the file export format of fuel transactions. Possible values:
 - *GIR C4:* described in the Export appendix of this document.
 - *GIR HLF1:* described in the *HLF formats* appendix of the GIR W100 user manual.
 - *Custom:* defines a custom format, by setting a lua script accessible from an icon at the right of the *Format* field. The script allows to define a header, a footer, and to customize each line output along with the filename. It also allows to define an optional output directly where the file is copied.

A *Test* button in this window allows to test the custom script against the 10 latest fuel transaction in the database. An error message displayed on the top of the window if something went wrong during the script execution. The *Console logs* tab displays the logs registered with the lua `print()` function.

- **Type:** defines the method of export.
 - *Scheduled:* transactions are automatically exported into a file in the server data directory. When selected, the *Export...* action in the Fuel transactions page allows to browse the last exported files. This action also shows a *On demand export* tab which allows to perform a new export manually.
 - *On demand (creation date):* transactions are exported from the Fuel transactions page by selecting a date range.
 - *On demand (visible transactions):* transactions are exported from the Fuel transactions page. All the transactions displayed in the collection are exported. If some controllers impacted by the current collection filter are not synchronized for the selected period, this will display a warning in the export window.
- **Transac. change:** defines the export behavior with respect to modified transactions.
 - *None:* do not export the changes made to transactions.
 - *Update:* add one new line to the export for each change made to transactions.
 - *Diff:* add two lines (one with a negative volume, the second with a positive volume) for each change.
- **Scheduled export:** when *Scheduled* is selected, defines when the export is done.
 - *Every n minutes:* sets an export frequency in minutes. Setting the frequency to 1 minute performs a real-time export, with one file per transaction.
 - *Once a day:* sets the time of day when the export is performed.
- **Common options:**
 - *Don't export external transactions:* if checked, export only transactions from stations managed by the application (see: Fuel transactions / Manual transaction entry).
- **GIR C4 format options:**
 - *Don't add header to exports:* if checked, omit the first line with column headers to the CSV file.
- **GIR HLF1 format options:**
 - *Delimiter:* selects which character (semicolon ; or comma ,) separates each field in the CSV file.
 - *Extension:* selects file extension (txt, csv or dat).
- **Last export:** when *Scheduled* is selected, this read-only field indicates the date of the last automated export of fuel transactions.

Chapter 4

Vehicles and drivers

Vehicle and driver lists are maintained in GIR W250 to authorize and to track fuel transactions. They are identified to controllers either by presenting a badge or entering a code. This information is captured during each transaction and stored along with other relevant details (date, volume, etc.) in the transaction record.



Driver identification is optional and depends on the settings of your GIR W250 server (see: Settings / General / Identification modes).


4.1 Vehicles

Vehicle editor fields:

- **ID**: text label for the vehicle to be used throughout GIR W250 . Used in all lists and reports of the application.
- **Department**: associated department for the vehicle. Can be used as a criterion in reports and related histories.
- **Badge/Code**: depending on the vehicle identification mode setting, defines either the badge number or the code used for identifying the vehicle to controllers.
- **Odometer/Meter**: current meter value. This field is automatically updated with the meter data entered during fuel transactions. Generally, a vehicle meter should not be modified manually. The only case where this might be necessary is to correct an invalid meter entered during a transaction that causes subsequent meter values entered at terminals to be rejected.
- **Model**: associated model for the vehicle. It can be used as a criterion in reports and related histories, and also defines several settings that will apply to the vehicle.
- **Products**: List of all products authorized for the vehicle.
- **Options**: only displayed if at least one option is available
 - *No gas mileage computation*: displayed if the vehicle as meter prompt

enabled. When enabled, this option prevents from computing the *Covered* and *Cons.* fields in fuel transactions for this vehicle, which can be useful when a vehicle record is shared by many physical resources.

Fuel transac.: Compact list of all fuel transactions associated with the selected vehicle, starting with the most recent. From here it is possible to expand this view and browse these records from the *Fuel transactions* page  or to manually create a transaction record for this vehicle .

Access tr.: Compact list of all access transactions associated with the selected vehicle. This tab is displayed only if at least one access is defined in controllers, and if they identify vehicles. From here it is possible to expand this view and browse these records from the *Access transactions* page .

Note that if the *Advanced vehicle options* feature is enabled in the application settings, all of the options defined in Models will also be available to define on a per-vehicle basis in the *Options* field (see: Settings / Fuel / Model).

4.1.1 Department reassignment

When a fuel transaction is processed, the department of the vehicle is stored in the transaction record. This way, when a vehicle department changes, existing transactions remain assigned to the previous department while future transactions will be assigned to the new department.

In practice, the vehicle department is not always changed at the exact time of its administrative transfer. Therefore, when a vehicle department is changed, the application will prompt to change the department in previous fuel transactions for that vehicle.

Possible actions:

- Re-assign all of the vehicle fuel transactions to the new department.
- Re-assign some of the vehicle fuel transactions to the new department given some time range.
- Don't change existing transactions.

4.2 Drivers


Driver editor fields:

- **Name, First name:** fields used to label to the driver throughout GIR W250 . Used in all lists and reports of the application.
- **Department:** associated department. It can be used as a criterion in reports and related histories.
- **Badge/Code:** depending on the driver identification mode setting, defines either the badge number or the code used for identifying the driver to controllers.

- **Options**

- *Activity code*: if checked, the driver will have to enter an activity code before each transaction.
- *NCE code*: if checked, the driver will have to enter a NCE code before each transaction.

Fuel transac.: Compact list of all fuel transactions associated with the selected driver, starting with the most recent.

Access tr.: Compact list of all access transactions associated with the selected driver. This tab is displayed only if at least one access is defined in controllers, and if they identify drivers. From here it is possible to expand this view and browse these records from the *Access transactions* page .

4.2.1 Department reassignment

When a fuel transaction is processed, the department of the driver is stored in the transaction record if the vehicle has no department. This way, when a driver department changes, existing transactions remain assigned to the previous department while future transactions will be assigned to the new department.

In practice, the driver department is not always changed at the exact time of its administrative transfer. Therefore, when a driver department is changed, the application will prompt to change the department in previous fuel transactions for that driver.

Possible actions:

- Re-assign all of the driver fuel transactions to the new department.
- Re-assign some of the driver fuel transactions to the new department given some time range.
- Don't change existing transactions.

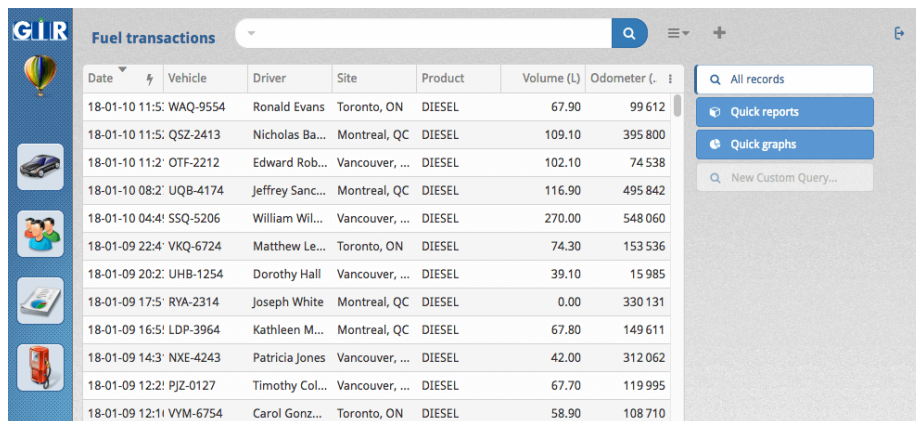
Chapter 5

Fuel transactions

Fuel transactions consist of data related to refueling visits at terminals. Fuel transactions can either be retrieved from controllers or entered manually into the GIR W250 web application.

5.1 Transactions list

The Fuel transactions page contains a list of all recorded fuel transactions, starting with the most recent. This list is automatically updated when the ⚡ icon is displayed in the *Date* column header.



Date ⚡	Vehicle	Driver	Site	Product	Volume (L)	Odometer (km)
18-01-10 11:5	WAQ-9554	Ronald Evans	Toronto, ON	DIESEL	67.90	99 612
18-01-10 11:5	QSZ-2413	Nicholas Ba...	Montreal, QC	DIESEL	109.10	395 800
18-01-10 11:2	OTF-2212	Edward Rob...	Vancouver, ...	DIESEL	102.10	74 538
18-01-10 08:2	UQB-4174	Jeffrey Sanc...	Montreal, QC	DIESEL	116.90	495 842
18-01-10 04:4	SSQ-5206	William Wil...	Vancouver, ...	DIESEL	270.00	548 060
18-01-09 22:4	VKQ-6724	Matthew Le...	Toronto, ON	DIESEL	74.30	153 536
18-01-09 20:2	UHB-1254	Dorothy Hall	Vancouver, ...	DIESEL	39.10	15 985
18-01-09 17:5	RVA-2314	Joseph White	Montreal, QC	DIESEL	0.00	330 131
18-01-09 16:5	LDP-3964	Kathleen M...	Montreal, QC	DIESEL	67.80	149 611
18-01-09 14:3	NXE-4243	Patricia Jones	Vancouver, ...	DIESEL	42.00	312 062
18-01-09 12:2	PJZ-0127	Timothy Col...	Vancouver, ...	DIESEL	67.70	119 995
18-01-09 12:1	VYM-6754	Carol Gonz...	Toronto, ON	DIESEL	58.90	108 710

Transactions list columns that are displayed by default:

- **Date**: date and time of the transaction from the instant the fuel pump is engaged. Recorded with one-second precision.
- **Vehicle**: identified vehicle.
- **Driver**: identified driver.
- **Site**: site where the transaction took place.
- **Product**: product distributed.

- **Volume:** volume distributed.
- **Odometer:** meter entered (may display hour meter instead).

Additional columns available:

- **Pump:** pump number used on the controller to distribute the fuel product.
- **Tank:** tank from which the fuel is pumped.
- **Duration:** total time elapsed to complete the transaction.
- **Department:** the vehicle department (at the time of the transaction), or the driver department if the vehicle has no department.
- **Model:** the current model of the vehicle.
- **Unit price:** unit price of the product (at the time of the transaction).
- **Activity:** activity which code has been entered on the terminal.
- **NCE code:** non-checked entry code entered on the terminal.
- **Operator:** displays the operator badge or code used for identification. Only available when at least one operator is defined
- **Odometer only:** meter entered.
- **Meter only (h):** meter entered.
- **Covered:** distance covered since the previous transaction (and/or time in the case of hour metering).
- **Cons.:** fuel consumption rate since the previous transaction in L/100km or mpg (and/or L/h or gal/h in the case of hour metering).
- **Notes:** additional notes.
- **Totalizer:** total volume counted by the pump.
- **Ident. mode (Driver):** identification mode used for driver identification.
- **Ident. mode (Vehicle):** identification mode used for vehicle identification.
- **Unauthorized refueling:** the transaction was not authorized.
- **Remote transaction:** the transaction was triggered from the supervision page.
- **Modified:** the transaction was modified (*[*]* indicator).
- **Created manually:** the transaction was created manually (*[M]* indicator).
- **Type:** internal or external (*[E]* indicator).

- **Meter was forced:** the meter entry was forced (*[F]* indicator).
- **Max capacity reached:** the transaction stopped after reaching the maximum volume (*[Max]* indicator).
- **End reason:** the cause of the transaction stop. Only set for pump modules starting from version 4.0.9. Possible values:
 - *Timeout:* the *timeout (before)* or *timeout (after)* pump parameter was reached
 - *Nozzle switch:* manual pump hang-up
 - *Max capacity reached:* the transaction reached its maximum volume authorized
 - *Max. duration:* the transaction reached its maximum duration
 - *Reboot:* the pump module power was lost
 - *C1/C2 error:* pump module counting error
 - *Triggered stop:* the transaction was stopped remotely
 - *Internal error:* pump module error

Transaction records may contain a special indicator next to one or more fields. These indicators are generally shown between brackets.

Possible indicators include:

- *[*]* in the *Product* field: the transaction was modified.
- *[M]* in the *Product* field: the transaction was created manually.
- *[E]* in the *Product* field: the transaction was made on an external station.
- *®* in the *Product* field: the transaction was started remotely.
- *[T]* in the *Product* field: the transaction was delivered by a tanker vehicle (only if the *Tankers W250* feature is enabled).
- *[F]* in the *Odometer* field: the meter entry was forced.
- *[O]* in the *Odometer* field (*Offline: no meter control*): the meter control was disabled because the controller was offline at the start of the transaction (only enabled for large applications).
- *[Max]* in the *Volume* field: the transaction stopped after reaching the maximum volume.
- *[Op. 1234]* in the *Vehicle* or *Driver* field: the 1234 operator badge or code was used to identify this vehicle or driver.
- *(1234)* in the *Vehicle* column: NCE code 1234 was entered. Note: only displayed in the *Vehicle* column if the *NCE code* column isn't visible.
- *(Activity)* in the *Vehicle* column: The code for this activity was entered. Note: only displayed in the *Vehicle* column if the *Activity* column isn't visible.

- *[ABCD]* in the *Site* column: the transaction was imported with the format “ABCD” (only if the *Transactions import* feature is enabled).
- *(Vehicle)* in the *Site* column: the transaction was delivered by the “Vehicle” tanker vehicle (only if the *Tankers W250* feature is enabled).

5.2 Manual transaction entry

Transactions can be entered manually. This generally occurs in one of two cases:

- Refuelings that took place outside of the stations managed by the application.
- Refuelings that were performed manually — for instance during a temporary terminal malfunction.

Click on the **+** button in the Fuel transactions page to create a new manual transaction record.

You will be prompted for the following fields:

- **Type**: transaction type.
 - *External*: refueling that was performed in a station that is not managed by the application.
 - *Internal*: manual refueling on a pump defined in the application that was not properly recorded.
- **Date**: date and time of the refueling.
- **Vehicle**: vehicle that made the refueling.
- **Driver**: driver that made the refueling (when driver identification is enabled).
- **Volume**: quantity of fuel distributed.
- **Meter/NCE code**: additional information required depending of the vehicle configuration.

If you selected the *External* type, you will also be prompted for:

- **Product**: the distributed product.
- **Unit price**: the unit price of the product.

If you selected the *Internal* type, you will also be prompted for:

- **Pump**: pump used for the transaction.

Once you’ve entered all the necessary information, the *Create* button allows you to store the record in the transaction history. The *Discard* button cancels the manual transaction creation and closes the editor.


5.3 Transaction modification

It is possible to modify a transaction after it has been recorded. Modifying transactions is useful in several cases:

- To correct an error in a manually-entered transaction.
- To correct a bad value entered into the controller (meter, NCE code, etc.).
- To reassign the transaction to another vehicle or driver, e.g. if a vehicle was mistakenly identified using the key of another vehicle.
- To correct the distributed volume, in case of pump malfunction.

Click on a transaction in the list to open the fuel transaction editor. A form with all editable transaction fields will be displayed. Note that not all fields always editable:

- For transactions retrieved from a controller, the date, site, product, pump and duration fields can not be modified.
- For transactions entered manually, any visible field can be modified.

The history of changes performed on a transaction (the audit trail) can be viewed by clicking the Info  button in the action menu. Each line in this list corresponds to a modification by a user on this transaction.

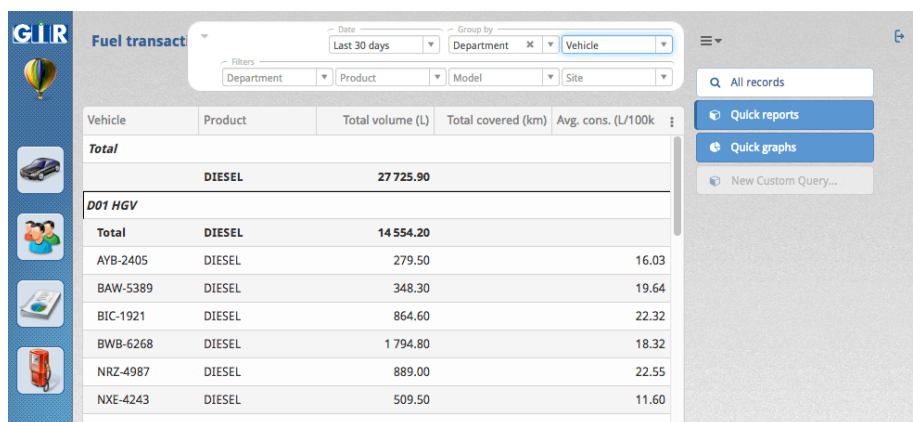
5.4 Transaction cancellation

Transaction cancellation is useful to remove manual entries created by mistake. Only manual transaction can be cancelled — transactions retrieved from the controller can not be cancelled.

Click on the delete  button to cancel a transaction.

5.5 Quick reports

The easiest way to generate fuel transaction reports is to select the *Quick reports* view. By default, the sum of all fuel consumed for all vehicles/products for the last 30 days is shown.



The screenshot shows the GIR Fuel transactions interface. At the top, there are filters for Date (Last 30 days), Department, and Vehicle. Below the filters is a table with columns: Vehicle, Product, Total volume (L), Total covered (km), and Avg. cons. (L/100k). The table is grouped by Vehicle, showing a total for DIESEL (27 725.90 L) and a sub-total for D01 HGV (14 554.20 L). Below the sub-total, individual transactions are listed with their vehicle IDs, products, volumes, and average consumption.

Vehicle	Product	Total volume (L)	Total covered (km)	Avg. cons. (L/100k)
Total		27 725.90		
D01 HGV		14 554.20		
AYB-2405	DIESEL	279.50		16.03
BAW-5389	DIESEL	348.30		19.64
BIC-1921	DIESEL	864.60		22.32
BWB-6268	DIESEL	1 794.80		18.32
NRZ-4987	DIESEL	889.00		22.55
NXE-4243	DIESEL	509.50		11.60
OTE-2212	DIESEL	2 250.70		24.51

On the right side of the interface, there is a sidebar with navigation options: All records, Quick reports, Quick graphs, and New Custom Query...


The report content is defined by a small form at the top of the screen. The first row contains the following fields:

- **Date:** the period during which transactions should be summed.
- **Group by:** defines the categories over which transactions are totalled and sub-totalled. For instance, selecting *Department* and *Vehicle* as criteria will show the total fuel consumption of each vehicle during the selected period, grouped by department.


The second row contains various filters that can be used to restrict the report to a particular subset of transactions (such as a specific Department, Site, Product, etc.).


The result of the report is presented in a table with the following columns:

- **Product:** fuel product concerned.
- **Total volume:** sum of transaction volumes.
- **Total covered (km or mi):** sum of transaction distances covered.
- **Avg. cons. (L/100k or mpg):** average consumption, computed from the Total volume and Total covered. Note that the volume of New meter transactions (refer to the New meters section below) is ignored from this calculation. For this reason, the value shown may not correspond exactly with the values for the Total volume and Total covered in the same row.

If the hour meter is enabled there will be two additional columns visible: *Total covered (h)* and *Avg. cons. (L/h or L/mi)*. These and other columns can be hidden by accessing the columns menu  in the top-right corner of the list. This menu can also be used to enable the display of two additional columns: *Count* and *Price*.

Report totals are computed for each Group by category as well as a global total for all transactions matching the requested filter and Date period. Selecting a row in the reports table will display a button to the left of the reports form. Clicking this button will display a detailed list of the fuel transaction records used to calculate the totals in the selected reports table row.

The dropdown button  to the left of the reports form switches to an *Advanced* view of the current report. This view allows for detailed customization of query parameters.

The action menu () allows to perform the following operations:

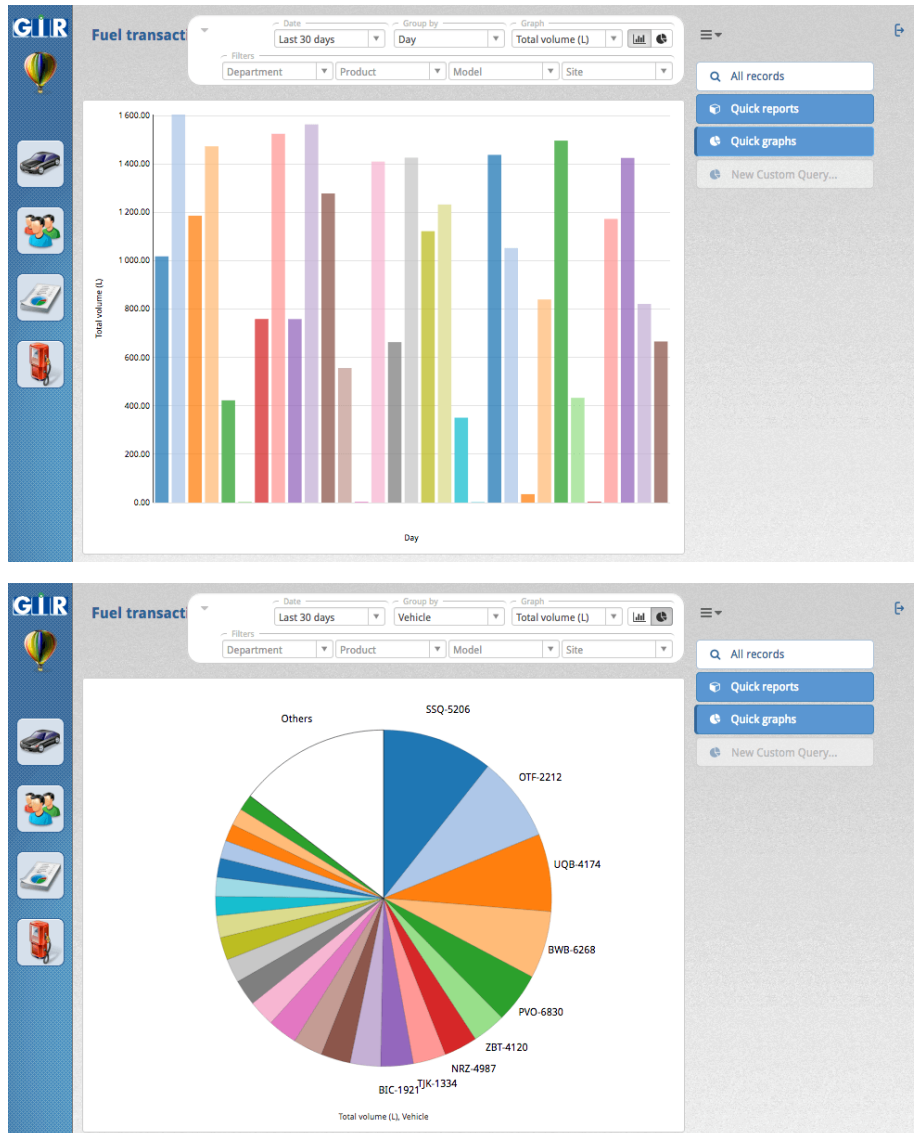
- **Print:** prints the report
- **Download CSV:** exports the report in CSV and downloads it
- **Download CSV (alt):** exports the report in CSV in an alternative format and downloads it

The report can also be stored by entering a name in the *New custom query...* field and clicking *Add (+)* in the Stored query panel on the right of the page.

5.6 Quick graphs

There are two types of graphs that can be produced in GIR W250 : histograms and pie charts.

The Quick graphs button is just below the Quick reports button. By default this will show a histogram of fuel consumption (total volume) by day for the last 30 days.



At the top of the screen is a form for changing the content of the graph. The first row contains the fields:

- **Date**: the period during which transactions should be summed.
- **Group by**: the categories over which transactions are totalled. For histograms, the possible values are units of time (per day, per week, per

month or per year). For pie charts, possible values are transaction fields (Vehicle, Driver, Department, etc.).

- **Graph:** the computed data that is shown on the graph (Total volume, Avg. price, etc.) as well as the graph type (histogram or pie chart).

The second row contains various filters that can be used to restrict the graph to a particular subset of transactions (such as a specific Department, Site, Product, etc.).

As with Quick reports, the bars in the histogram and slices of the pie charts can be selected to get a detailed view of all fuel transaction records contained therein. It is also possible to switch to an *Advanced* view, store the current graph or print it using the same controls as before.

5.7 New meters

Computing an accurate average consumption can be tricky. For instance, consider a new vehicle that was purchased with an initial odometer reading of 1234 km. It makes 3 transactions during its first month of service. Its transaction history would look like this:

Date	Volume (L)	Meter (km)	Covered (km)
29/06/07 08:15	109.00	1924	324
25/06/07 07:32	115.00	1600	366
22/06/07 14:20	150.00	1234	0

If these transactions were summed, a total volume of 374 L would be obtained along with a covered distance of 690 km. The average consumption would then be calculated as 54.2 L/100km. This results because the volume of the first transaction is not associated with any covered distance. In reality, the vehicle has only consumed 224 L in travelling 690 km. The real-world average consumption is 32.5 L/100km.

To address this situation, GIR W250 supports a *New meter* option for transactions. When a transaction record has this option enabled, its volume is ignored in average consumption calculations. The New meter option is automatically set for the first transaction of a vehicle.



Chapter 6

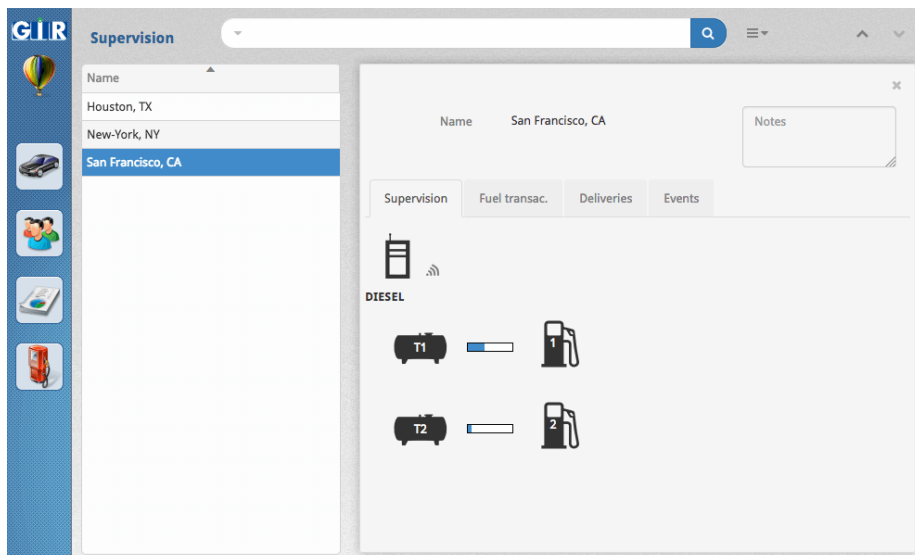
Supervision

The Supervision page provides a monitoring overview of controllers, tanks and pumps at each of the sites managed by the application. When there is only one site, the supervision window is shown full-screen. Otherwise, a list of sites is initially shown (on the left side of the page) along with a map (on the right side of the page) of site locations (using the GPS coordinates defined in the Settings).


All of the data presented on this page is automatically updated in real-time.

6.1 Supervision

The top of this tab lists all of the controllers defined for this site. Clicking the controller icon  shows the controller editor (see: Settings). Next to the controller icon is the controller communication status  (see: Controllers).



Below the list of controllers are the pumps assigned to this site, organized by tank.

The current volume of each tank is represented by a small graph with a blue bar representing the quantity of fuel remaining. If a gauge is defined for this tank, a second graph below the first one is added representing the real quantity a fuel measured by the gauge. Additional tank information is found by clicking the tank icon :


- **Theoretical volume:** current volume / total volume (in liters)
- **Unit price:** current unit price of the fueling product in this tank
- **Gauge volume:** if a gauge is defined, then it's the value measured by the gauge. Clicking on this value opens a window with the details of the gauging data.

Clicking a pump icon  exposes two actions:


- **Block / Unblock:** blocks or unblocks the pump. A blocked pump cannot be used by the controller. A pump may be automatically blocked by a controller to prevent inconsistencies in the event of a technical malfunction.
- **+ transaction:** remotely initiates a transaction on the controller for this pump. Opens a new window where the vehicle, driver and meters can be specified. Once this form is submitted, a new transaction immediately begins on the controller and the pump can be used.

If something is wrong with a pump, its icon is colored in red. Clicking a pump icon will provide a message indicating the cause. Possible reasons:

- **Link error:** communication error between the controller and the pump module
- **Pump blocked:** the pump has been blocked either automatically by the system or manually through the supervision page
- **Manual:** the pump is in manual mode



If there is another problem, a warning icon is displayed next to the controllers . Clicking the warning icons icon will provide a message indicating the cause. Possible reasons:

- **Invalid date:** the controller date is incorrect
- **Memory error:** the controller cannot store any transaction
- **Memory full:** the controller memory is full
- **Terminal error:** the terminal is not responding
- **Reader error:** a reader is not responding
- **Printer error:** a printer is not responding


If an access is defined on this site, it will be represented by an access icon . Clicking on that icon exposes one action:

- **+ transaction:** remotely initiates a transaction on the controller for this access. Opens a new window where the vehicle and driver can be specified. Once this form is submitted, a new transaction immediately begins on the controller which activates the command relay.

6.2 Fuel transac.

Compact list of the most recent fuel transactions associated with this site, starting with the most recent. From here it is possible to expand this view  and browse these records from the *Fuel transactions* page or to manually create a transaction record for this site .

6.3 Access tr.

This tab is only visible if an access is configured on at least one controller. It contains a compact list of the most recent access transactions associated with this site, starting with the most recent. From here it is possible to expand this view  and browse these records from the *Access transactions* page.

Access transaction fields:

- **Date:** date and time of the access transaction.
- **Access:** access name.
- **Vehicle:** vehicle identified.
- **Driver:** driver identified.
- **Status:** Possible values:
 - *Granted:* the relay was commanded after a successful identification
 - *Denied:* the identified vehicle or driver was not authorized for this access point
 - *Button:* the relay was commanded because the push button was pressed (only for *MR-Access* modules)
 - *A/M – Manual:* the relay switch was set to manual mode (only for *MR-Access* modules)
 - *A/M – Auto:* the relay switch went back to auto mode (only for *MR-Access* modules)
 - *Stay open – Start:* start of *Stay open* mode from supervision or badge identification when access mode is *Stay open badges*, or from supervision only when the *Zones* feature is enabled and the access has stay open time ranges
 - *Stay open – End:* end of *Stay open* mode from supervision or badge identification when access mode is *Stay open badges*, or from supervision only when the *Zones* feature is enabled
 - *Stay closed – Start:* start of *Stay closed* mode from supervision (only when the *Zones* feature is enabled and the access has stay open time ranges)


- *Stay closed – End*: end of *Stay closed* mode from supervision (only when the *Zones* feature is enabled and the access has stay open time ranges)
- *Forced opening*: the door was open without commanding the relay (only when the access mode is *Punctual with DS*)
- *Door left open*: the door was not closed after being opened normally (only when the access mode is *Punctual with DS*)
- *Door closed*: the door was finally closed after a *Forced opening* or *Door left open* transaction (only when the access mode is *Punctual with DS*)
- *Door not opened*: the door was not opened after the relay was commanded (only when the access mode is *Punctual with DS*)
- *ON*: alternative label for *Granted* status based on MR-Access DS state (only when the access mode is *ON/OFF according to DS*)
- *OFF*: alternative label for *Granted* status based on MR-Access DS state (only when the access mode is *ON/OFF according to DS*)

6.4 Manual Deliveries

A manual delivery is a record of a fuel product being resupplied to a tank. This tab is only visible if the *Manual Deliveries* option has been enabled in a tank. It shows the fuel manual deliveries for this site, starting with the most recent. This list of fuel manual deliveries is filtered by tank, so by default only the manual deliveries for the first tank (*T1*) are displayed. To visualize the manual deliveries for one tank, click on its name (*T1*, *T2* or *T3...*) to select the corresponding tank.

Manual Delivery fields:

- **Date**: date and time of the manual delivery.
- **Tank**: tank that was resupplied.
- **Product**: delivered product.
- **Volume**: volume delivered.
- **Unit price**: unit price of the delivered product (only available if the *Unit price* option has been enabled for this tank).

New fuel manual delivery records can be created by clicking the  button and then completing the corresponding form. Once submitted, the tank current volume and unit price will be updated accordingly.

The tank volume and unit price before and after the manual delivery can be found in the “Before delivery” and “After delivery” tabs. Those values can be invalidated if the current or a previous manual delivery is modified or deleted.

When creating a manual delivery, the unit price of the tank product will be automatically updated based on the unit price of the remaining product (before delivery) and that of the product delivered, weighted by volume. The exact formula is as follows:

$$\text{new price} = (P_s * V_s + P_t * V_t) / (V_s + V_t)$$

with :

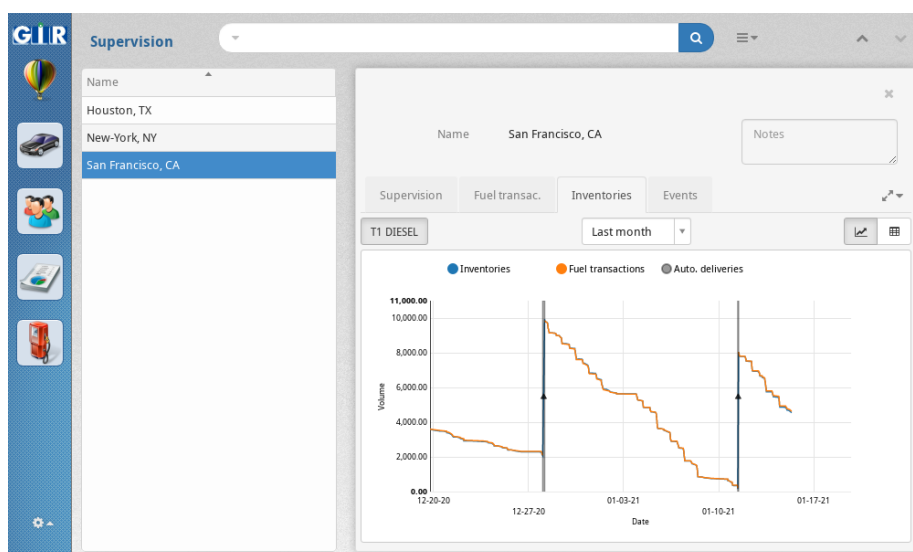
- P_t : remaining tank stock unit price
- V_t : remaining tank stock volume before the manual delivery
- P_s : manual delivery unit price
- V_s : delivered volume

Prices can also be set to automatic deliveries, generated from gauges (see below).

6.5 Inventories

This tab is only visible if a gauge is configured for at least one tank. It shows a chart with the history of a tank stock on the selected time range. This chart shows three different items:

- In blue, the history of inventories, which shows the evolution of the tank fuel stock, as measured by the gauge.
- In orange, the evolution of the tank fuel stock computed from fuel transactions volumes, as measured by pumps. This graph is only visible after the first automatic delivery.
- In gray, automatic deliveries, which are significant variations of the volume measured by the gauge, after taking into account fuel transactions. A positive variation generally means that a delivery was made. A negative variation generally means that fuel was taken from the tank without having been stored in fuel transactions. See below for more detail.





6.5.1 Auto. Deliveries

When a tank has both pumps and gauges, there are two independent channels for counting fuel:

- on one hand, pumps measure the volume that goes out of the tank, which is stored in fuel transactions.
- on the other hand, gauges measure the volume in the tank, which is stored in inventories.

By comparing those two measures, GIR W250 can detect fuel stock variations that are not caused by transactions, and show them as “Auto. Deliveries” (automatic deliveries).

Detected automatic deliveries are visible:

- on the chart, as gray areas with a ▲ triangle icon at their center. Details for an automatic delivery can be displayed by clicking on the triangle icon.
- in a scrollable list, toggled from the chart view with the  icon at the right top of the chart.
- in a dedicated *Auto. Deliveries* menu within the  dropdown menu at the right of the *Inventories* tab.

An automatic delivery has the following properties:

- **Start, End** : Time range when the volume variation was detected. Depending on inventories frequency, this range can vary between a few minutes and a whole day.
- **Tank**: Tank on which the automatic delivery was detected.
- **Product**: Tank product.
- **Notes**: Text field for additional comments.
- **Volume** : Delivery volume: the amount of fuel delivered into (or removed from) the tank. It can be formally expressed as “Gauge_at_end - Gauge_at_start + Concurrent_transactions”, with:
 - *Gauge_at_end* : volume measured by the gauge at the end of the delivery.
 - *Gauge_at_start* : volume measured by the gauge at the start of the delivery.
 - *Concurrent_transactions* : total volume of fuel transactions made during the delivery.

A positive volume is an increase of the tank stock, typically indicating a delivery. A negative volume is a decrease of the tank stock, which was not stored in fuel transactions.

- **Drift** : Starting from the second automatic delivery, a *Drift* value is computed. It is the difference between volumes measured by the gauge since the previous delivery, and the total volume of fuel transactions made during the same time range. The drift indicates the relative error between fuel volumes measured by pumps and fuel volumes measured by gauges. In other words, the relative error between blue and orange curves. The drift depends on the precision of the hardware used for counting, and can be useful to verify that pumps and gauges are correctly configured. In parenthesis, a percentage value named *Drift ratio* compares the *Drift* value with the total transactions volumes since the previous automatic delivery.

The system also shows computed tank stock volumes before and after the delivery. From a stock accounting perspective, we must always have:

$$\text{Volume (After)} = \text{Volume (Current)} + \text{Volume (Before)}$$

with:

- **Volume (After)**: Tank stock after the delivery. This is the volume as reported by the gauge at the end of the delivery.
- **Volume (Current)**: Delivery volume, see **Volume** just above.
- **Volume (Before)**: Tank stock “before” the delivery. When no transaction occurred during the delivery, this is the volume as reported by the gauge at the start of the delivery. When transactions did occur during the delivery, to preserve the stock accounting equality above, the system considers all these transactions to have occurred, by convention, “just before” the delivery. Therefore, *Volume (Before)* is computed as “Gauge_at_start – Concurrent_transactions”.

The good detection of automatic deliveries depends on how reliable the measures are, in both gauges and pumps. Especially, when a vehicle refueling occurs at the same time an inventory is measured, the fluctuations caused by the pump can cause a loss of precision in the gauge measures. Even when the hardware works as expected and is correctly configured, some inventories can be approximate. When the system detects that an inventory has been made at the same time as a fuel transaction, the symbol \approx is shown before the automatic delivery volume, to indicate it may be unreliable.

Automatic deliveries are detected automatically by the GIR W250 server when enough data is received, and when all controllers related to a tank (through pumps or gauges) are synchronized. When new deliveries are detected, a alert is shown in the alert panel, and a email is sent to users that have enabled the *Tank stock* notification.

6.5.2 Auto. Deliveries unit price

As a full replacement for manual deliveries, automatic deliveries can be assigned prices, which are then used to compute unit prices for tank stocks and fuel transactions.

Prices for automatic deliveries are enabled by the *Unit price* option in the tank settings. They require *Manual deliveries* on this tank to be disabled: if

both manual deliveries and prices are enabled on a tank, prices are then managed in manual deliveries, not automatic ones.

Prices can only be defined for automatic deliveries with a positive volume. Negative deliveries are supposed to leave the tank stock price unchanged, and as such can't be assigned prices. In all this section about prices, "automatic delivery" is to be understood as "priceable automatic delivery", i.e. a positive delivery.


When prices are enabled, additional fields are available in the automatic delivery editor:


- **Unit. price (Before):** the tank stock unit price before the delivery. It is always read-only, but can be recomputed when needed.
- **Unit. price (Current):** the delivery unit price.
- **Unit. price (After):** the tank stock unit price after the delivery. It is automatically computed from the other values, and can also be specified manually.

On the inventory graph, the automatic delivery  triangle icon is red for deliveries with no price defined.

When the most recent automatic deliveries for a given tank have no price, an alert is also shown in the alert panel: "Auto. delivery without a price".


Overall, the system is designed to:

- Allow prices to be automatically computed when they are entered chronologically. This is the simplest and recommended use: just start from the first delivery for which you know the price, set it, and then just enter *Unit. price (Current)* chronologically as new deliveries arrive, and let the system compute *Unit. price (After)*.
- Allow the user to override the default computation and define a custom *Unit. price (After)* at each step.
- Automatically propagate prices to fuel transactions, but never further than the "next segment". In other words, when an automatic delivery gets a *Unit. price (After)* value, this price is set to all fuel transactions between the current delivery and the next delivery. When the current delivery is the last one, the unit price is also set to the tank stock itself, which is then used by new fuel transactions.
- For each delivery, show a  custom price icon when *Unit. price (After)* has been customized by the user. Clicking the icon recomputes the after price based on before and current prices and volumes.
- Allow prices to remain optional when unknown: if no price is set on a given delivery, the tank stock just keeps its existing price, and a price can still be set on the next one.

- Allow past prices to be modified, for instance to fix input errors. Whenever this is done, the previous rule regarding the “next segment” of fuel transactions still applies:
 - prices are then changed in all fuel transactions between the current delivery and the next one.
 - a  warning icon is shown near *Unit. price (Before)* in the next delivery, indicating that past prices have been changed. Clicking it shows a *Recompute* button which allows to update the current delivery prices, and propagate the price change further. In terms of tanks stocks prices and fuel transactions pricing, changing prices in past deliveries and manually propagating them gives the exact same values as if these prices had been initially entered.

6.5.3 Auto. Deliveries unit price – extra detail

In detail, the capabilities described in the previous section are achieved through the following rules and UI elements on the automatic deliveries price fields:


- *Unit. price (Before)* can be:
 - *Unknown* in gray if this is the first priceable automatic delivery of the tank, and no stock price has been defined in the tank yet.
 - *Unknown* in red if there are deliveries with prices before the current one, but the previous delivery doesn't have a price. Clicking on *Unknown* selects the previous delivery, allowing to enter prices chronologically. If some prices are not known, or if one doesn't wish to go back too far in the past, it is also possible to leave *Unit. price (Before)* as *Unknown* and to just manually enter *Unit. price (After)*.
 - A numeric value with no icon: *Unit. price (Before)* in the current delivery is then the same value as *Unit. price (After)* in the previous delivery.
 - A numeric value with a  warning icon: This indicates that prices were retroactively changed in the previous delivery, so *Unit. price (Before)* in the current delivery is no longer the same as *Unit. price (After)* in the previous delivery. *Unit. price (Before)* can then be updated with the *Recompute* button to propagate the price change.
- *Unit. price (Current)* can be:
 - Empty when no price has been defined yet
 - A numeric value when a price has been defined
- *Unit. price (After)* can be:
 - Empty when no price has been defined yet
 - A numeric value with no icon when the after price was automatically computed by the system from the before and current price and volume values.

This computation is just an average of the existing tank stock price and the current delivery price, weighted by volume. The exact formula is as follows:

$$\text{Unit. price (After)} = (P_s * V_s + P_t * V_t) / (V_s + V_t)$$

with :


- * P_t : *Unit. price (Before)*
- * V_t : *Volume (Before)*
- * P_s : *Unit. price (Current)*
- * V_s : *Volume (Current)*

- A numeric value with a  custom price icon when the after price was set manually by the user. When the before price is not unknown, clicking this icon recomputes the after price using the default formula.

Whenever *Unit. price (After)* changes, it is assigned:


- to all fuel transactions until the next delivery. A transaction is considered as being between two deliveries D1 and D2 when the transaction start date is between D1 end date and D2 end date. Any price change performed after the first assignment shows a confirmation prompt indicating how many transactions will be impacted by the operation.
- to the tank stock if this was the last delivery for this tank

6.5.4 Auto. Deliveries – false positives

When the gauged volume values is imprecise or sometimes false, it is possible that auto. deliveries get incorrectly detected by the system. In this case, it is possible to mark the auto. delivery as a false positive by using the  deletion button. Marking an auto. delivery as false positive ignores this volume variation, and updates the *Drift* value of the following delivery if it exists. When deliveries unit. price is enabled, this action is only possible when no price has been assigned on the delivery yet.


False positive deliveries are not displayed in the *Inventory* graph, but can be found in the *Auto. deliveries* page. Those records are masked by default, but can be displayed when filtering the collection with the *False positive* field. Those deliveries have an additional *False positive* checkbox which is the only field that can be edited. This field allows to cancel the false positive marking for this delivery.

6.5.5 Inventories data


The  dropdown menu at the right of the *Inventories* tab allows to access the list of stored gauge inventories.

Each inventory contains the following fields:

- **Date** : Date and time when the inventory was measured.
- **Tank** : Tank associated to this inventory.
- **Volume**: Volume of fuel measured.

- **Temperature:** Temperature measured.
- **Type:** What caused the inventory:
 - *Scheduled:* an inventory is scheduled daily at midnight for all gauges. It is only stored if the controller is powered on at that time.
 - *Transaction:* an inventory is stored after each fuel transaction.
 - *Supervision:* stored when the supervision page is accessed, and the volume measured changed significantly.
 - *Auto:* stored automatically by the controller when gauged volume changed significantly.
- **Gauge readings:** Technical detail of gauge readings for each gauge of this tank. A click on a line shows a pop-up with the reading details. The  button allows to access the gauge readings list. See *Gauge readings* below.

6.5.6 Gauge readings

The gauge readings list is available with the  button in the details of an inventory. It contains technical information returned by gauges, mostly useful for troubleshooting purpose:

- **Date :** Date and time when the gauge reading was made.
- **Tank :** Tank associated to this reading.
- **Result:** Gauge reading result. Possible values:
 - *OK:* Successful reading.
 - *Link error:* The gauge module was not detected.
 - *Error (Timeout):* The gauge device didn't respond.
 - *Error (Invalid response):* The gauge device response couldn't be interpreted.
 - *Probe error:* The gauge device returned an error in response (see *Vendor error code*).
- **Vendor error code:** Error code returned by the gauge device.
- **Volume :** Volume of fuel measured by the gauge.
- **Height :** Height of fuel measured by the gauge.
- **Water volume :** Volume of water measured by the gauge
- **Water height :** Height of water measured by the gauge.
- **Temperature :** Temperature measured.
- **Duration :** Duration of the measure.
- **Probe :** Name of the gauge.
- **Inventory :** Link to the inventory for this reading.
- **Version, Offset :** additional technical data returned by the gauge (Hec-tronic gauges only)

6.6 Events

The event history is a general-purpose list of various status changes and notifications generated by either a specific controller or the GIR W250 server. The event history is accessible from the home sub-menu (see Getting Started: Main Menu). It can also be accessed on the Supervision page under the details for a particular site, in which case only events for that site are displayed and the list is updated automatically.

Event fields:

- **Date:** date and time of the event.
- **Site:** the site where the event occurred.
- **Category:** event category.
- **Type:** event type.
- **Pump:** the pump concerned by the event (if applicable).
- **Content:** the event details. May vary depending on the event type.
- **Vehicle:** the vehicle concerned by the event (if applicable).
- **Driver:** the driver concerned by the event (if applicable).
- **Badge:** the badge stored in the event (if applicable).
- **Code:** the code stored in the event (if applicable).
- **Ident. mode:** the identification mode stored in the event (if applicable).

Event categories as well as the specific event types are organized as follows:

- **Communication:** controller communication related events.
 - *Pairing...*: a pairing was started on a controller.
 - *Pairing succeeded*: a pairing was completed.
 - *Pairing failed*: a pairing failed after some time.
 - *Pairing cancelled*: a pairing was cancelled by a user.
 - *Unpairing*: a controller was unpaired manually by a user.
 - *Unpairing (error)*: a controller was unpaired automatically (after a synchronization error, for example).
 - *Reboot*: a controller was rebooted.
 - *Offline*: a controller went offline.
 - *Online*: a controller went back online.
 - *Upgrade...*: a controller upgrade request was sent.
 - *Upgrade succeeded*: a controller upgrade was completed.
 - *Upgrade failed*: a controller upgrade failed after some time.
 - *Restart*: a controller was restarted.
- **Identification:** identification on controllers related events.
 - *Unknown identification*: unknown code or badge.
 - *Denied identification*: known badge but used in an invalid context.

- **Pumps:** pumps related events.
 - *Blocked:* a pump was blocked either by the controller or from the supervision page.
 - *Unblocked:* a pump was unblocked either by the controller or from the supervision page.
 - *Manual:* : a pump was switched to manual mode.
 - *Auto:* a pump was switched back to automatic mode.

6.7 Tanks

The *Tanks* page allows to display and update all the tanks in the application. It is possible to access to this page from the *Supervision* page, by clicking on the page title at the top left corner, then by selecting the *Tanks* element in the dropdown menu.

Columns displayed by default:

- **Site:** Site name.
- **Number:** Tank number.
- **Product:** Tank product.
- **Unit price:** Current tank unit price.
- **Capacity:** Tank capacity.
- **Theoretical volume:** Current tank theoretical volume.
- **Gauge volume:** Last gauged volume.
- **Gauge date:** Last gauged volume date

6.7.1 Past stocks

This report can be accessed from the *Tanks* page, by selecting the *Past stocks* element in the saved queries panel.

This report allows to compute an estimation of the past stock value of tanks, at a selected date. This estimation is fixed in time, as long as the data related to deliveries and transactions on the period between the selected date and now are unchanged.

The content of the report is determined by the form at the top of the screen, containing the following fields:

- **Date:** date used for the past stock estimation. The stock will be computed for the selected date at midnight (00:00).
- **Site:** allows to filter by site or region.
- **Product:** allows to filter by product.
- **Type:** allows to filter by stock management type (automatic or manual).

The report result is presented in the form of a table with the following columns:


- **Tank:** Tank name. If the *[M]* tag is present, this line is about manual stock management. Otherwise, it is about auto stock management. If a tank has both auto and manual stock management, then two lines will be added to the report, one for each stock management type.
- **Past stock:** Tank stock computed at the selected date at midnight (00:00).
- **Fuel transactions:** Sum of fuel transactions volumes from the selected date up to now.
- **Deliveries:** Sum of deliveries volumes from the selected date up to now. For auto stock management, then it's auto. deliveries. For manual stock management, it's manual deliveries.
- **Drift:** Cumulative error in reconciliations between gauges and fuel transactions between the selected date up to now. Only present for automatic stock management.
- **Current stock:** Current theoretical or gauged stock.

The calculation of the past stock is different depending the stock management type. For manual stock management, the past stock is computed from current stock, fuel transactions, and deliveries. For any past date, the system guarantees that:

$$\text{Past stock} - \text{Fuel transactions} + \text{Deliveries} = \text{Current stock}$$

For automatic stock management, a past stock is obtained from the nearest tank inventory after the selected date. The stock at exactly midnight is then computed from fuel transactions and auto. deliveries. For any past date, the system guarantees that:

$$\text{Past stock} - \text{Fuel transactions} + \text{Deliveries} + \text{Drift} = \text{Current stock}$$

The action menu () allows to perform the following operations:

- **Print:** prints the report
- **Download CSV:** exports the report in CSV format and downloads it
- **Download CSV (alt):** exports the report in CSV in an alternative format and downloads it

6.7.2 Future stocks

This report can be accessed from the *Tanks* page, by selecting the *Future stocks* element in the saved queries panel.

This report allows to compute an approximation of the future stocks evolution for each tanks. The calculation of this approximation is done by computing average fuel consumption over a given interval.

The content of the report is determined by the form at the top of the screen, containing the following fields:


- **Sample:** sampling period used to calculate consumption averages.
- **Site:** allows to filter by site or region.
- **Product:** allows to filter by product.
- **Type:** allows to filter by stock management type (automatic or manual).


It is important to select a right sampling period to get reliable estimates. Selecting a too short sampling period or for period of abnormal activity may bias the results.

The report result is presented in the form of a table with the following columns:

- **Tank:** Tank name. If the *[M]* tag is present, this line is about manual stock management. Otherwise, it is about auto stock management. If a tank has both auto and manual stock management, then two lines will be added to the report, one for each stock management type.
- **Capacity:** Tank capacity.
- **Current stock:** Current theoretical or gauged stock.
- **Alert threshold:** Tank alert threshold.
- **Days before alert:** estimation of the number of days before the tank volume drops below the alert threshold.
- **Block threshold:** Tank block threshold.
- **Days before blocking:** estimation of the number of days before the tank volume drops below the block threshold.
- **Days before zero:** estimation of the number of days before the tank is completely empty.

By clicking on one of the report lines, it displays a graph simulating the future stock evolution of the selected tank.

The  menu at the table top right corner can be used to display or hide some columns.

The action menu () allows to perform the following operations:

- **Print:** prints the report
- **Download CSV:** exports the report in CSV format and downloads it
- **Download CSV (alt):** exports the report in CSV in an alternative format and downloads it

Chapter 7

Controllers




Controllers are autonomous: they can operate without a permanent link to the GIR W250 server. To this effect, each controller stores its own local copy of the database, with the list of authorized vehicles and drivers. When a transaction is made, it is stored in the controller memory, waiting to be retrieved by the GIR W250 server.

Connecting a new controller to the GIR W250 server is called a pairing. Each controller has its own unique serial number — the pairing associates the GIR W250 server with the controller using the unique serial number. Once controllers are paired, they are able to communicate with the server.

The sending of drivers and vehicles and the retrieval of fuel transactions is called a “synchronization” in GIR W250 and is part of the regular communication process between the server and the controllers. The synchronization is done automatically once a controller is paired and online.

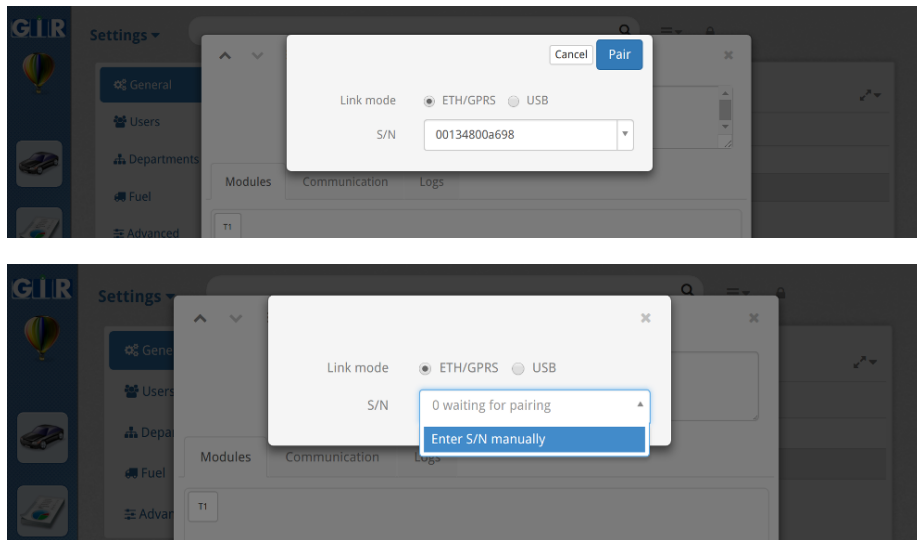
7.1 Pairing

The pairing functionality is accessible from the controller editor through the *S/N* field (the controller editor itself is accessed through the Settings page — see: Settings / Sites). When a controller is paired, its serial number is displayed in this field. The icon next to this field indicates the controller pairing status:

- : either not paired (empty S/N) or pairing in progress (non-empty S/N).
- : paired.
- : unpaired automatically (icon appears in red).

7.1.1 Start a pairing

To pair a controller, first create a new controller under a Site in the Settings / General / Sites tab. Next, click the  icon. A new window is displayed:



This window first invites you to select a *link mode*. This parameter defines how the controller communicates with the GIR W250 server, and two options are available:

- **ETH/GPRS**: online communication through Ethernet or GPRS
- **USB**: offline communication through files exchanged on USB devices

This window also supports input of a serial number. If the controller is already configured to communicate in *ETH/GPRS* with the GIR W250 server, the S/N selection box will display *1 waiting for pairing* and the controller serial number it will already be available. In that case, simply select it and click *Pair* to begin the pairing process.

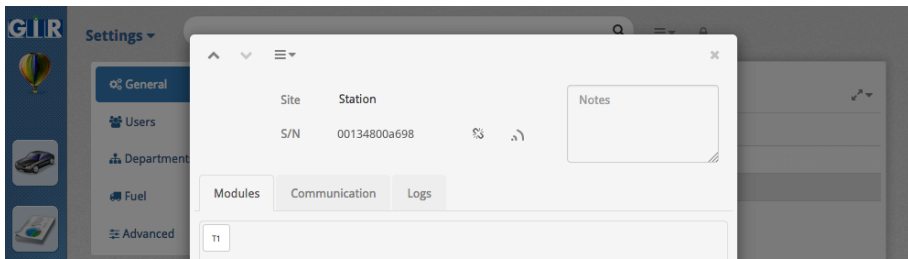
Otherwise, if the controller is not already configured to communicate with the server, select the option *Enter S/N manually* from the S/N selection box. This will change the selection box into a free-form text field. Manually enter the controller serial number and click *Pair*. The pairing process will then wait for the controller to connect before starting the installation.

If the *USB* link mode is set, you also have to enter the controller serial number manually. For more information, please refer to the USB communication section below.

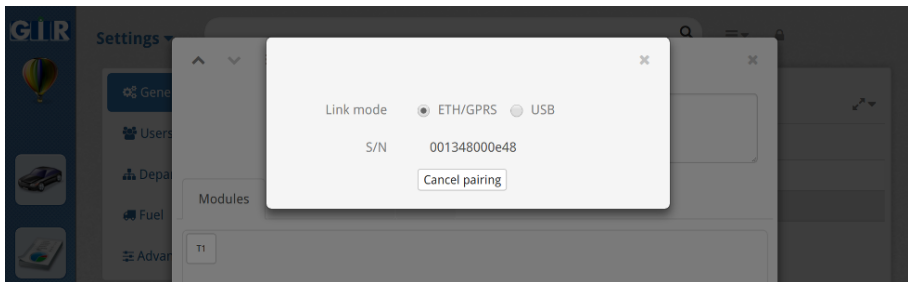
Pairing process in *ETH/GPRS* link mode:


1. Wait for the controller to connect
2. Install new firmware if needed
3. Restart the controller
4. Configure the application
5. Send the drivers & vehicles list

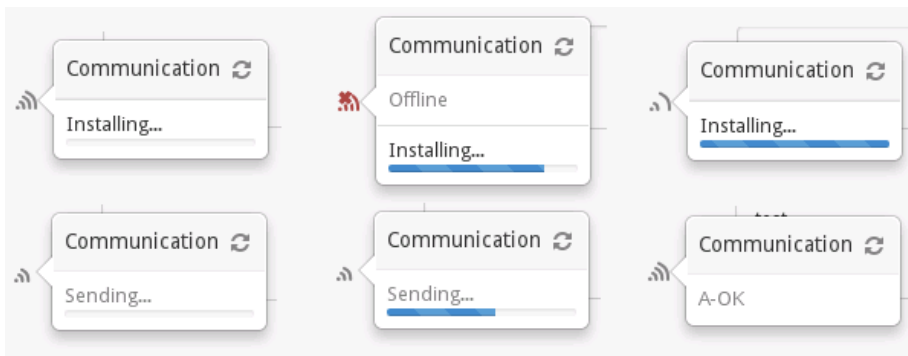
Once pairing has started in *ETH/GPRS* link mode, the controller editor will display the following:



Pairing can be cancelled by clicking .

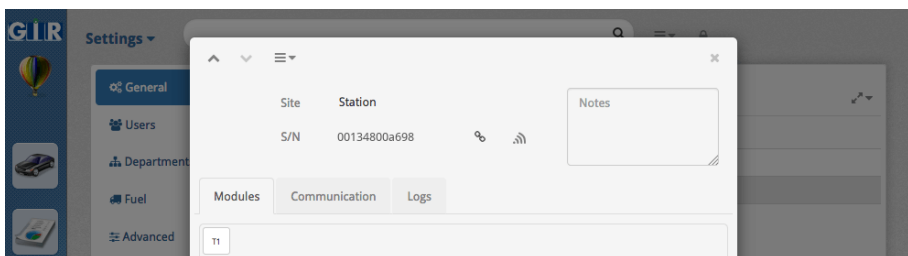



In *ETH/GPRS* link mode, the progression of the pairing process can be seen by clicking  (refer to the *Communication* section below).



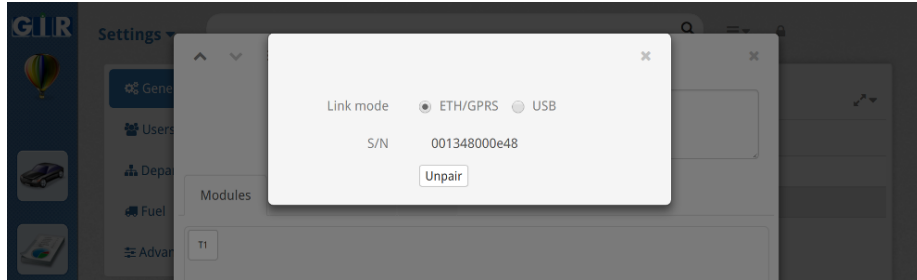
7.1.2 Paired state

After pairing has completed, the controller editor will display:



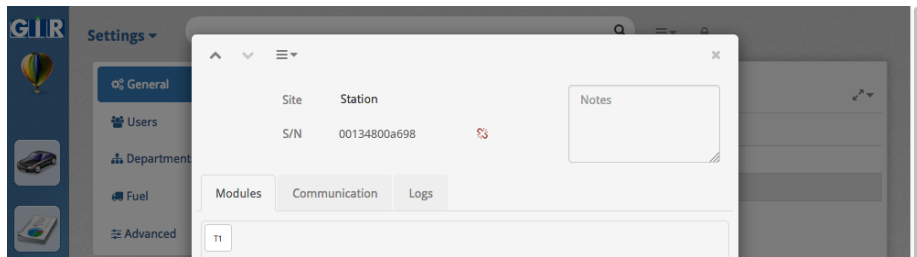
A controller can be unpaired by clicking .

Unpairing a controller is a dangerous operation that can result in data loss. It should only be done when you are certain that all transactions have been synchronized to the GIR W250 server. If you unpair a controller that has not completely synchronized, all transactions not yet retrieved can be lost.

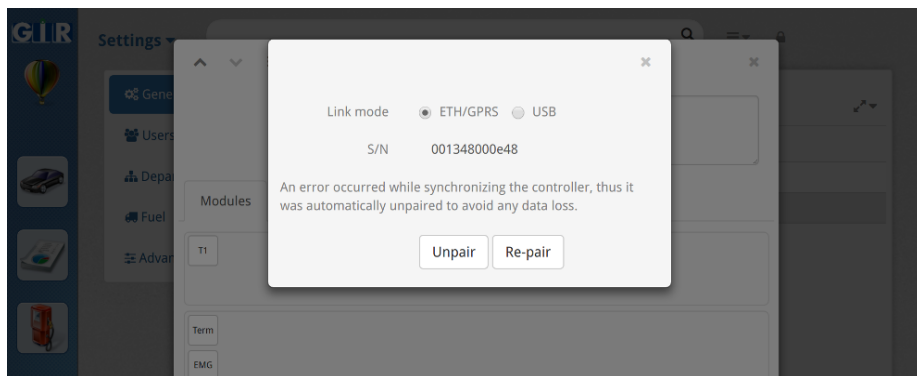


7.1.3 Automatically unpaired state

In the event of synchronization error, it is possible that a controller is unpaired automatically by the GIR W250 server in order to prevent inconsistencies or data loss. In this case, the following will be visible in the controller editor:



You'll find more details and possible actions if you click on the  icon.










Actions:

- **Unpair**: completely unpairs the controller with this serial number. The controller will then be in an uninitialized state and it will be possible to pair it with another serial number.
- **Re-pair**: initiates a new pairing with the serial number. In that case, it will reset all the controller data and send again all the vehicles and drivers.

7.2 Communication




The communication state indicator is an icon that provides a quick overview of the status of the communication between a controller and the GIR W250 server. It is found in the controller editor to the right of the *S/N* field. Clicking this icon displays the communication window pop-up with details of the current state.


Communication state:

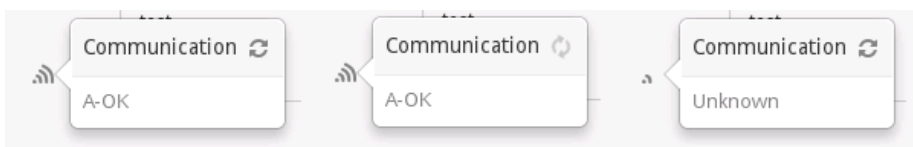
- (no icon): the controller is not yet paired.
- : *Offline* – the controller is offline.
- : *Unknown* – the controller may be offline.
- : *Loading* – the controller is currently pairing or upgrading its firmware (the icon is animated).
- : *Not synchronized* – the controller is synchronizing some data (the icon is animated).
- : *A-OK* – synchronized and operating normally. Nothing to report.
- : *Invalid pairing* – the controller is connected but cannot communicate with the server due to a pairing problem.
- : *USB communication* – the controller communicates with files using USB devices. See the USB communication section below.

7.2.1 Ping



The synchronization process between controllers and the GIR W250 server is optimized such that messages are not continuously exchanged when there is no data needing to be sent. It is possible to immediately verify the controller connection and refresh the communication state by sending a *Ping*, accessed from the communication window pop-up (only available in *ETH/GPRS* link mode).

Clicking  will send a Ping to the controller. The  icon will spin while the Ping is being sent. If the Ping succeeds, the communication state indicator should return to the  *A-OK* state.

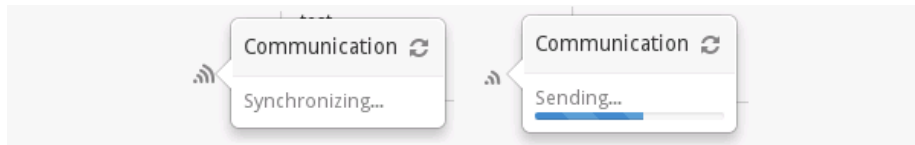
After 10 seconds we still haven't received the controller response, the ping fails and it switches to the *Unknown*  state.



7.2.2 Synchronization

When in *ETH/GPRS* link mode and data has been changed by the GIR W250 server (such as a new vehicle added) or if the controller has transactions to send, then the controller enters the  *Not synchronized* communication state. Clicking  will show details in the communication window pop-up.

If there are fewer than 5 records needing to be synchronized, the communication window pop-up will display the message *Synchronizing...* Otherwise, it will display the message *Sending...* or *Receiving...* along with a progress bar tracking the synchronization process.



7.2.3 Diagnostic tools

Diagnostic is an action available in the *Communication* tab of the controller editor. It allows to analyze what's going on a controller when it doesn't work as expected. The *Diagnostic* action requires *kvgca 2.0.27* or above.

Clicking on this action opens a new window. If the controller is online, a form will be displayed containing three sliders:

- **Modules test:** defines whether to test and report module statuses. Possible values are:
 - *None:* don't display module statuses.
 - *Soft:* displays current module statuses. This is a read-only operation which doesn't interrupt any controller operation, and may report cached information.
 - *Hard:* force a full modules refresh before displaying statuses. This operation is longer than *Soft:* it ensures that all module statuses are refreshed before reporting them. If a user prompt is in progress, it will be reset. Fuel transactions in progress are not impacted and can continue normally.
- **Prompt logs:** this log contains interactions between the controller and the users. It displays data entered by users on the terminal keyboard, and badges used for identification. It also shows what was displayed on the terminal screen. If different than *None*, the slider defines how much data should be fetched from the controller (more data takes longer to retrieve).
- **Devices logs:** this log contains all the technical data exchanged between the controller and devices on serial links. If different than *None*, the slider defines how much data should be fetched from the controller (more data takes longer to retrieve).

Below the form, or instead it if the controller is offline, a *Last operations* table allows to review the last diagnostic operations done on this controller. The *Show* button displays the results for this previous operation.

7.2.4 Other actions

The following actions are also available in the controller *Communication* tab:

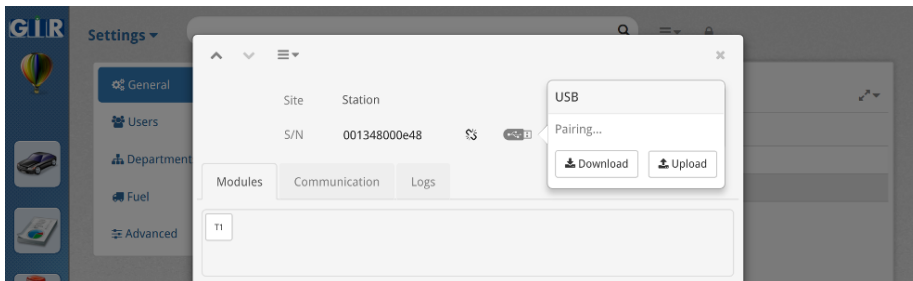
- **Upgrade:** available if the controller is online and if its version is not up to date. It upgrades the controller to the latest version. Controller operation continues normally during almost all the upgrade operation, while transferring the new firmware file: only a small interruption occurs when the controller restarts on the new firmware. Fuel transactions in progress continue normally during and after an upgrade.
- **Reboot:** available if the controller is online. This forces a full software system reboot on the controller. Fuel transactions in progress continue normally during a controller reboot, as long as pump modules are powered on.

7.3 USB Communication

If the *USB* link mode is selected during the pairing, the controller will communicate with the GIR W250 server through files on a USB device.

7.3.1 Pairing

Select *USB* link mode in the pairing window, enter your controller serial number and click “Pair”. The USB communication window pop-up will then show up and display *Pairing...* alongside two buttons: *Download* and *Upload*.




To complete the pairing, you'll have to:

1. *Download* the synchronization file
2. Put the file on a USB device.
3. Insert the USB device in one of the controller's USB ports.
4. Follow the instructions on the terminal.
5. *Upload* the file created by the controller on the USB device.

Once the first *upload* is done, the USB communication window pop-up will display *Last synchronization:* and the date when the upload was done.

7.3.2 Communication

On *USB* link mode, the communication state indicator is replaced by an USB device icon: . A click on this icon displays the USB communication window pop-up which contains the following elements:

- **Last synchronization:** the date of the last upload for this controller.
- **Download:** button that generates and download a USB synchronization file to send the controller.
- **Upload:** button that sends a USB synchronization file from the controller to the GIR W250 server. It is also possible to drag & drop the file directly into the window pop-up to trigger an upload.

A click on *Download* generates a file containing the GIR W250 server data in order to send it to the controller. The file should be named `gir-usbsync-<S/N>-server_to_vatersay.` (where `<S/N>` is the controller serial number). Copy this file directly to the root directory of a USB device, and then insert this USB device to one of the controller's USB ports. The USB device will then be automatically detected and the terminal will display:

```
PRESS 9 FOR USB SYNC
```

Press the **9** key to start the USB synchronization scenario. Press **Cancel** to return to the controller's idle state.

```
26/01/2018 10:28
DATE OK?
```

Press the **Validate** key to confirm that the date displayed is correct. If the date is not correct, press the **Cancel** key and enter the correct date. Once the date is validated, the USB synchronization will start:

```
USB SYNC 0.0%
PLEASE WAIT
```

This operation can take several minutes.

```
USB SYNC 100.0%
PLEASE WAIT
```

Once the following message is displayed, it is safe to remove the USB device:

```
USB SYNC OK
PLEASE REMOVE DEVICE
```

The controller will then reload its configuration with the new server data and then return to the idle state. During the USB synchronization scenario, the controller generates a file on the USB device (in its root directory). The file should be named `gir-usbsync-<S/N>-vatersay_to_server.dat`. This file contains the controller transactions to send to the GIR W250 server. This is the file that should be selected when the *Upload* button is clicked.

Once the upload is completed, the transactions are added to the application. A *+<N> Fuel transactions* notification is displayed (<N> is the number of transactions fetched from the controller), and the *Last synchronization* date is updated.

7.3.3 Special cases

During the USB synchronization scenario, if the USB file generated by the GIR W250 server contains a new version of the controller's firmware, the scenario will upgrade the controller firmware and reboot after displaying the following message:

```
REBOOTING...
PLEASE WAIT
```

If the controller was already paired to another GIR W250 server, it will ask the user to confirm the pairing with the new server:

```
NEW SERVER DETECTED.
PAIR W/ NEW SERVER?
```

Press the **Validate** key to confirm, or the **Cancel** key to quit the USB synchronization scenario:

```
PAIRING WILL ERASE
LOCAL DATA. OK?
```

Press the **Validate** key again to confirm, or the **Cancel** key to quit the USB synchronization scenario:

```
PAIR W/ NEW SERVER?
TYPE 255 TO CONFIRM
```

Press the **2**, **5** and **5** keys to continue the pairing and erase the existing controller data, or the **Cancel** key to quit the USB synchronization scenario.

7.4 Controller usage

When a controller is idle, it will display one of the following messages on its onboard display:

- **PUMP 1 AVAILABLE, SEL.PUMP 1 2:** messages referencing pump numbers indicate that one or more pumps are available. Other pumps numbers not mentioned in the message are not available.
- **TRANSAC. IN PROGRESS:** all available pumps are currently in use.
- **OUT OF ORDER:** no pump is available.

A typical refueling process will consist of the following steps:

1. **Pump selection:** select a pump number on the keyboard. This step is skipped when only one pump is defined on a controller.
2. **Identification:** vehicle and (optionally) driver identification, as defined in the configuration (see: Settings / Ident. modes).
3. **Optional entries:** Meter and NCE code entries, as defined in the configuration (see: Settings / Models).

During data entry, the process can be cancelled at any time by pressing the cancel key.

The message **TRANSAC. CANCELLED** will appear and the controller will return to the idle state.

A detailed diagram of the refueling process is available in the appendix (see: Refueling process).

7.4.1 Pump selection

```
Mo 03/09/16 08:30 .+
PUMP 1 AVAILABLE
```

or

```
Mo 03/09/16 08:30 .+
SEL.PUMP 1 2
```

Select the desired pump number.

```
P1 DIESEL
Validate your choice
```

The pump and the product name appears. Press the **Validate** key to continue.

Note:

- When only a single pump is defined on the controller, the pump selection step is skipped.
- When a pump is not available, pressing its number shows the reason.

A hypothetical terminal with three pumps defined might show the following:

```
Mo 03/09/16 08:30 .+  
SEL.PUMP 1 2
```

In this situation, pumps 1 and 2 are available for use. Pressing 3 would display the reason why pump 3 is unavailable:

```
Pump 3 DIESEL  
BLOCKED/SUPERV.
```

7.4.2 Identification

In the next step, a controller waiting for a driver or vehicle identification will display one of the following messages:

Controller waiting for a vehicle badge:

```
P1 DIESEL  
VEHIC. BADGE
```

Controller waiting for a vehicle code:

```
P1 DIESEL  
VEHIC. CODE
```

Controller waiting for a driver badge:

```
P1 DIESEL  
DRIVER BADGE
```

Controller waiting for a driver code:

```
P1 DIESEL  
DRIVER CODE
```

After a successful identification, the vehicle or driver label may be displayed on the terminal, but only if all vehicles/drivers identify by badge.

7.4.3 Meter entry

```
P1 DIESEL
Meter km:
```

Enter the odometer value. Units may vary.
Possible error messages:

- **INCORRECT VALUE:** The entered meter value is outside of the valid range. See Settings / Models.

An incorrect value entered for a meter can be forced, in the following cases:

- Operator-initiated transaction (see: Settings / Ident. modes). A confirmation message to accept the forced value will appear after the first incorrect entry.
- The meter setting is not set to *range (strict)*. A confirmation message to accept the forced value will appear after the same incorrect value is entered twice in a row.

7.4.4 Activity code

```
P1 DIESEL
Activity:
```

Enter the activity code. Use the **Code** field in the activity editor (See Settings / Activities)

7.4.5 NCE code entry

```
P1 DIESEL
NCE code:
```

Enter the NCE code. Any non-empty value will be accepted.

7.4.6 Pump timeouts

Once the previous steps are completed, the controller will display *READY TO PUMP*, the pump will engage and dispensing can take place.

During product dispensing, the following timeouts apply:

- If dispensing is not started within T_{begin} seconds, the transaction is stopped.
- Once dispensing is started, if it idles for more than T_{end} seconds, the transaction is stopped.
- If the pump nozzle is hung up, the transaction is stopped.

- Once the transaction has been stopped, the volume output is counted for an additional T_{after} seconds and included in the transaction. This accounts for compression of the pump hose.

T_{begin} and T_{end} are defined in the pump configuration (See Settings / Sites). T_{after} is fixed to two seconds.

7.5 Operator menu

The operator menu can be accessed by pressing “0” on idle:

- when the idle prompt is pump selection, just pressing “0” enters the operator menu
- when the idle prompt is vehicle or driver identification, pressing “0”+validate enters the operator menu

This menu has two entries:

1 : PUMPS
2 : GAUGES

- “1:PUMPS” asks for operator authentication, then runs the pumps unblocking scenario
- “2:GAUGES” doesn’t ask for authentication, and shows the current tanks stocks reported by gauges

7.5.1 Pumps unblocking

The pumps unblocking scenario first asks for operator authentication, using an operator badge or code defined in the identification modes (see: Settings / Ident. modes).

Then, for each pump currently blocked on the controller, it prompts for unblocking:

P1 DIESEL UNBLOCK?

Pressing the Validate key unblocks the pump, which then becomes available for normal use.

Pressing the Cancel key leaves the pump blocked, and moves on to the next pump blocked if there is one.

Hardware-based pump unblocking

It is also possible to unblock a pump without an operator badge or code by accessing the internal case of a controller.

This can be useful when a pump is blocked, no operator badge or code has been defined, and no server connection is available to perform the unblocking from the supervision menu.

The hardware-based pump unblocking method is as follows:

- Open the controller case
- On the pump module controlling the pump you want to unblock, toggle the auto/manual (A/M) switch to “manual” (LED on)
- The terminal prompts to unblock the pump, press the Validate key to confirm
- Set the A/M switch back to “auto” (LED off)

7.5.2 Gauges

The gauges menu shows the current volume of all gauges on a controller.

It is read-only, and can be accessed without authentication.

It can be disabled in the controllers settings (see: Settings / Sites).

For each gauge, this menu shows the current volume. The gauge device is actively polled and the value is updated in real-time when it changes.

T1 DIESEL	.	/+
000000 / 000000 L	*	

The indicators on the screen are as follows:

- at the top left, the tanker number and product
- at the top right, a navigation indicator showing if there are gauges before or after the current one. Pressing 7/9 or -/+ browses to the next or previous gauge
- at the bottom left, the current volume value, followed by the tank total capacity as defined in the settings
- at the bottom right, a “*” indicator is shown when there is a communication error with the gauge device. This tells that the volume reported may be out of date

When there are multiple compartments, the tank capacity is replaced by the compartment number and total compartments count on the tank:

T1 DIESEL	.	/+
000000 L (1/N)	*	

7.6 Operator identification

Operator identification mode in controllers is accessed by identifying with specific badges and codes defined in the identification modes (see: Settings / Ident. modes). Operator mode provides access to special menu options that are useful in two particular cases:

- To bypass normal refueling restrictions on vehicles, drivers, products, etc. — typically by “forcing” a particular vehicle or driver for the transaction.
- As a wildcard badge when a badge has been lost or forgotten.

Depending on the application configuration, Operator mode should be used in the following manner:

7.6.1 Vehicle code only

Forcing a vehicle code:

- VEHIC. CODE: Enter the operator code
- VEHIC. CODE: Enter the vehicle code

7.6.2 Vehicle badge only

Forcing a vehicle, vehicle badge lost

- VEHIC. BADGE: Use the operator badge
- VEHIC. CODE: Enter the vehicle code

7.6.3 Driver badge then vehicle code

Forcing a vehicle, forcing a driver, driver badge lost:

- DRIVER BADGE: Use the operator badge
- VEHIC. CODE: Enter the vehicle code

7.6.4 Driver badge then vehicle badge

Forcing a driver, driver badge lost:

- DRIVER BADGE: Use the operator badge
- VEHIC. BADGE: Use the vehicle badge

Forcing a vehicle, vehicle badge lost:

- DRIVER BADGE: Use the driver badge
- VEHIC. BADGE: Use the operator badge
- VEHIC. CODE: Enter the vehicle code

7.6.5 Driver code then vehicle badge

Forcing a driver: not possible.

If you are concerned that you may need to perform this action, it may be better to use the “vehicle badge then driver code” identification mode.

Forcing a vehicle, vehicle badge lost:

- DRIVER CODE: Enter the driver code
- VEHIC. BADGE: Use the operator badge
- VEHIC. CODE: Enter the vehicle code

7.6.6 Driver code then vehicle code

Forcing a driver:

- DRIVER CODE: Enter the operator code
- VEHIC. CODE: Enter the vehicle code

Forcing a vehicle:

- DRIVER CODE: Enter the driver code
- VEHIC. CODE: Enter the operator code
- VEHIC. CODE: Enter the vehicle code

7.6.7 Vehicle badge then driver code

Forcing a vehicle, forcing a driver, vehicle badge lost:

- VEHIC. BADGE: Use the operator badge
- VEHIC. CODE: Enter the vehicle code
- DRIVER CODE: Enter the driver code

7.6.8 Vehicle badge then driver badge

Forcing a vehicle, vehicle badge lost:

- VEHIC. BADGE: Use the operator badge
- VEHIC. CODE: Enter the vehicle code
- DRIVER BADGE: Use the driver badge

Forcing a driver, driver badge lost:

- VEHIC. BADGE: Use the vehicle badge
- DRIVER BADGE: Use the operator badge

7.6.9 Vehicle code then driver badge

Forcing a vehicle: not possible.

If you are concerned that you may need to perform this action, it may be better to use the “driver badge then vehicle code” identification mode.

Forcing a driver, driver badge lost:

- VEHIC. CODE: Enter the vehicle code
- DRIVER BADGE: Use the operator badge

7.6.10 Vehicle code then driver code

Forcing a vehicle, forcing a driver

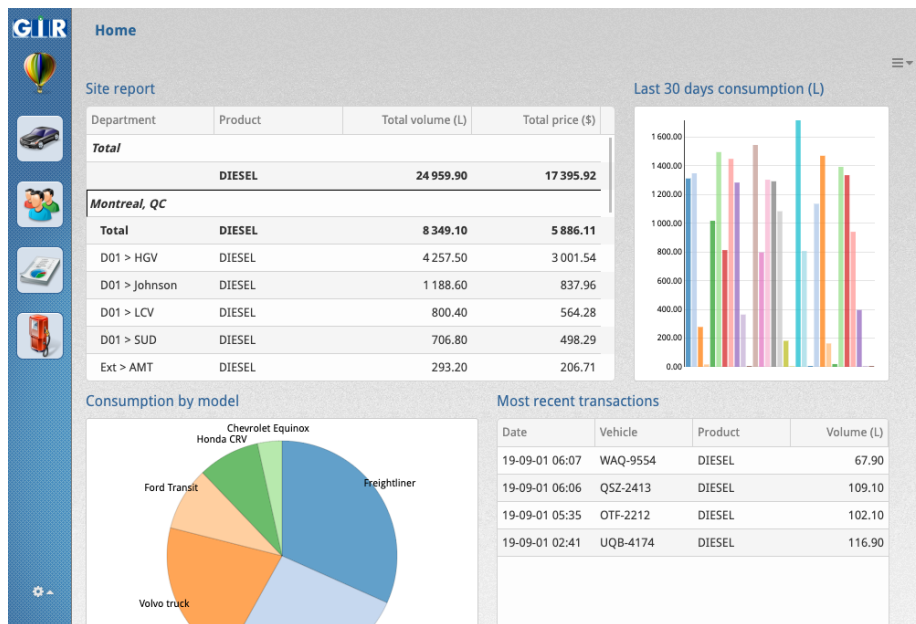
- VEHIC. CODE: Enter the operator code
- VEHIC. CODE: Enter the vehicle code
- DRIVER CODE: Enter the driver code

Chapter 8

Dashboard

GIR W250 supports a customizable dashboard for displaying saved queries from across different sections of the application in one place, on the home page. The dashboard can be customized to display relevant reporting and operational information.



The dashboard is automatically enabled once a saved query is added to the dashboard.

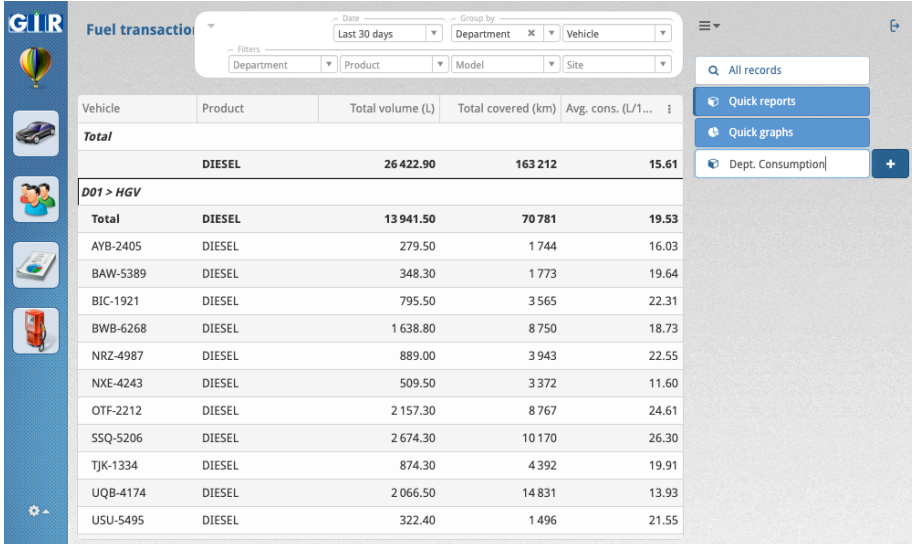


8.1 Adding a sample report to the dashboard


The basic process to add a report to the dashboard is to create the relevant report in its collection view (see: Common user interface elements), save it using a saved query, and then add the saved query to the dashboard.

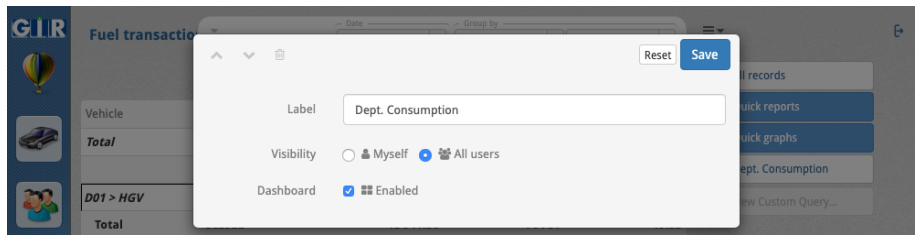
The following steps illustrate in detail how to add an example report to the dashboard.

- Go to the Fuel transactions collection by selecting the Fuel transactions link in the main menu
- Expand the saved queries panel if it is not already visible by clicking 
- Click *Quick reports*
- Using the Quick reports query form, select Date: “Last 30 days”. Using the *Group by* inputs, select “Department” and then “Vehicle”: a table summarizing aggregate totals will be shown
- Save the current report by creating a Saved query:
 - Click the text input box at the bottom right of the screen
 - Enter a label for this saved query: “Dept. Consumption”
 - Click 



Vehicle	Product	Total volume (L)	Total covered (km)	Avg. cons. (L/100km)
Total				
	DIESEL	26 422.90	163 212	15.61
D01 > HGV				
Total				
	DIESEL	13 941.50	70 781	19.53
AYB-2405	DIESEL	279.50	1 744	16.03
BAW-5389	DIESEL	348.30	1 773	19.64
BIC-1921	DIESEL	795.50	3 565	22.31
BWB-6268	DIESEL	1 638.80	8 750	18.73
NRZ-4987	DIESEL	889.00	3 943	22.55
NXE-4243	DIESEL	509.50	3 372	11.60
OTF-2212	DIESEL	2 157.30	8 767	24.61
SSQ-5206	DIESEL	2 674.30	10 170	26.30
TJK-1334	DIESEL	874.30	4 392	19.91
UQB-4174	DIESEL	2 066.50	14 831	13.93
USU-5495	DIESEL	322.40	1 496	21.55

- Access the editor for the new saved query:
 - Place the mouse over the newly-created “Dept. Consumption” item. This will make a  icon appear.
 - Click it to open the saved query editor.
- In the *Visibility* field, select “All users”
- In the *Dashboard* field, click the “Enabled” checkbox
- Click “Save”.
Note that only users with the “Edit other settings” permission (see: Settings / Users) can change a saved query’s visibility to “All users”, and add it to the dashboard.







- Click the Home page link in the main menu. The saved query will now appear on the home page for all users.


Vehicle	Product	Total vol...	Total cov...
Total			
	DIESEL	26 422.90	163 2
D01 > HGV			
Total	DIESEL	13 941.50	70 7
AYB-2405	DIESEL	279.50	1 7
BAW-5389	DIESEL	348.30	1 7
BIC-1921	DIESEL	795.50	3 5
BWB-6268	DIESEL	1 638.80	8 7
NRZ-4987	DIESEL	889.00	3 5

- Repeat this process with other reports to add them to the dashboard as well.

8.2 Editing the dashboard


To edit the dashboard, open the action menu drop-down button  on the home page and click on “Edit dashboard”. This will enable dashboard edit mode, where it is possible to:

- resize a tile: a tile can be resized to a few predefined widths by clicking the  icon in the bottom-right corner, and dragging it to the desired width.
- reorder dashboard tiles: tiles can be reordered by clicking the  icon in the top-left corner, and then dragging it to the desired position.
- rename or remove a tile: click the  icon next to a dashboard tile to open the saved query editor. From there, a tile can be renamed by changing its *Label*, or removed from the Dashboard by unchecking the “Dashboard: Enabled” checkbox. Note that a tile removed from the dashboard will still be available as a shared query in its collection view.

To exit dashboard edit mode, click the “Save” button to save the changes that have been made, or click  or “Discard” to ignore changes and restore the original dashboard.

Chapter 9

Specific features

GIR W250 has a number of additional features that are available in the Settings / Advanced / Features page. To make this tab visible, a valid QSC/RSC code must be entered via the  button (see: Settings / QSC/RSC protection).

9.1 KM+Miles odometers (imperial)

This feature allows for vehicles with odometers in imperial miles to be used alongside others with odometers in kilometers.

When enabled, the *mi* and *mi + h* values is added as choices for the *Meter(s)* option (either for models or for vehicles, depending on the *Advanced vehicle options* setting) .

Once *mi* or *mi + h* is selected for a product, the vehicle is considered to have an odometer in miles. When the vehicle is identified to a controller, the controller will prompt for its meter in miles. The odometer in the vehicle editor will also be shown in miles.

On the Vehicles, Drivers and Fuel transactions pages, a *km / mi* switch will be added on the left side of the page, below the *delete* icon. This switch can be used to dynamically change the units for all odometer values shown in all lists on the page.



9.2 Third-party services

The *Third-party services* feature allows the GIR W250 server to communicate with external web services. A *Third party services* tab appears in the Settings / Advanced section.

Third-party services fields:

- **Type:** web service type. Currently, only *Meter collection* is available.

- **Vendor:** web service provider. Available values:
 - *Ctrack*
 - *Eliot/OMP* (also allows to resolve vehicles if the vehicle of a transaction has the “generic” third-party ID, and GPS coordinates)
 - *Fleetmatics* (also allows to retrieve hour meter)
 - *Geotab* (also allows to retrieve hour meters and GPS coordinates)
 - *Geothentics*
 - *GPS Insight* (also allows to retrieve GPS coordinates)
 - *Groeneveld* (also allows to resolve vehicles if the vehicle of a transaction has the “generic” third-party ID)
 - *INGTECH*
 - *Isotrak*
 - *Matrix telematics* (also allows to retrieve GPS coordinates)
 - *NIBC Corcra* (also allows to retrieve hour meter)
 - *Ocean*
 - *Transics* (also allows to retrieve GPS coordinates)
 - *Trimble* (also allows to retrieve GPS coordinates)
 - *Truckonline* (also allows to retrieve GPS coordinates)
 - *Verizon Connect Fleet (TDE)*
 - *Verizon Networkfleet* (also allows to retrieve hour meter and GPS coordinates)
- **Details:** customizes the web service URL and access credentials.
- **Label:** optional label. If set, this value is appended to the third-party service label when the service is referenced elsewhere.

Meter collection services allow for vehicle meters to be retrieved automatically from third-party services so that they need not be entered by drivers during transactions.

If there is a *Meter collection* third-party service defined, a new field is added to the vehicle editor: *Meter collection*. Clicking this field opens a new window with the following options:

- **Meter collection:** defines how the vehicle meter is retrieved for future transactions. Possible values:
 - *Manual:* the vehicle meter is input manually on the controller terminal (classic behaviour).
 - *Auto:* the vehicle meter is retrieved automatically from a third-party service.
- **Meter collection service:** third-party service to use to retrieve the vehicle meter. Displayed if *Auto* Meter collection is selected.
- **Third-party ID:** the vehicle ID as configured in the third-party service. Displayed if a Meter collection service is selected. A *Test* button will be visible to test whether the vehicle ID is correctly recognized by the third party service.

If a *Meter collection* third-party service with vehicle resolution (*Eliot/OMP*) is defined, a new field is added to the driver editor: *__Third-party ID*. Clicking this field opens a new window allowing to configure the driver ID as configured in

the third-party service. A *Test* button will be visible to test whether the driver ID is correctly recognized by the third party service, with an optional *Date* parameter which allows to simulate the vehicle resolution for a transaction.

9.2.1 Geofencing

Along with odometers and hour meters, some third-party services allow to fetch the GPS coordinates of a vehicle at the date of the fuel transaction. When such a service is defined in the application, this triggers some changes in the “Fuel transactions” page.

When a fuel transaction has GPS coordinates, this adds a *Map* tab in the fuel transaction editor, which contains the following markers:

- A green marker displaying the position of the vehicle for this transaction
- A red marker displaying site GPS position of the fuel transaction (if defined in the site settings)
- Blue markers displaying the GPS positions of all the other sites (if any)

Still in the fuel transaction editor, when both the transaction and its site have GPS coordinates, this adds the following field in the “Details” tab:

- ***Geofencing***: displays the distance between the transaction and its site, in km (or miles in US units)

When a third-party service defined in the application can resolve GPS coordinates, this adds a *Geofencing map* item in the *Fuel transactions* page action menu. This action is only enabled when the filtered collection results total count is less than 2000. When clicked, this shows a new window containing a map that displays:

- The GPS positions of all filtered transactions as green markers
- The GPS positions of all sites as blue markers

In the third-party services editors that can resolve GPS coordinates, this adds the following field:

- ***Geofencing radius***: threshold defining the maximum distance in km (or miles in US units) for the fuel transactions *Geofencing* values. When defined, the icon at the right of the *Geofencing* field in fuel transactions becomes red when its value exceeds this threshold

The *Geofencing radius* threshold also allows to trigger alerts:

- When there are transactions in the last 24 hours with an exceeding *Geofencing* distance, this displays a *Geofencing alert* item in the alert panel. The (*View*) link for this alert allows to see all the transactions that triggered this alert
- When a user has enabled the *Unauthorized refueling transactions* notification, this user will now receive an email alert when the resolved *Geofencing* distance of a new fuel transaction exceeds the third-party service *Geofencing radius* threshold

9.3 Generic badges

This feature enables multi-technologies badges for the *TLG* reader. It adds the following values to the Vehicles and Drivers identification modes:

- *Generic TLG badge*
- *Advanced badge*

With the TLG reader, more than 40 different RFID technologies are supported, in 125kHz and 13.56MHz ranges, including EM, HID, HITAG, ISO14443A/Mifare, ISO14443B, LEGIC, ISO15693. . .

The TLG reader can also be used to read EMG and MFG badges.

9.4 MFG badges

This feature enables the MFG badge technology. It adds the following value to the Vehicles and Drivers identification modes:

- *MFG badge*

It also adds the *GemProx* reader device in the controller modules configuration.

9.5 BS125 badges

This feature enables the BS125 badge technology. It adds the following value to the Vehicles and Drivers identification modes:

- *Advanced badge*

It also adds a *BS125* pump reader device in the controller modules configuration.

9.6 RPK badges

This feature enables the RPK badge technology. It adds the following values to the Vehicles and Drivers identification modes:

- *RPK badge*
- *Advanced badge*

It also adds a *RPK* pump reader device in the controller modules configuration.

9.7 ISO2 badges

This feature enables the ISO2 badge technology. It adds the following value to the Vehicles and Drivers identification modes:

- *Advanced badge*

It also adds a *ISO2* reader device in the controller modules configuration.

9.8 iButton badges

This feature enables the iButton badge technology. It adds the following value to the Vehicles and Drivers identification modes:

- ***iButton S/N badge***

It also adds a *iButton* reader device in the controller modules configuration.

9.9 Tacho badges

This feature enables the Tacho badge technology. It adds the following value to the Drivers identification modes:

- ***Tacho badge***

It adds the following value to the Vehicles identification modes if *Driver + Vehicle* is selected in the *Identification* field:

- ***Tacho – Auto ID badge***

If drivers use *Tacho badge* and vehicles *Tacho – Auto ID badge*, then a second identification mode is allowed for vehicles even if the *Multiple Ident. Modes* feature is disabled.

It also adds a *Tacho* reader device in the controller modules configuration. In this module editor, the following parameters are available:

- ***Tacho. download***: if set to something else than *None*, enables the Tacho. download on this controller. Currently only works with *ETH/GPRS* communication mode, and not in *USB* mode. Possible values:
 - *None*: Tacho. download is disabled for this reader
 - *C1B (Daily)*: Tacho. download is enabled for this reader, C1B files will be downloaded at most once a day for a given Tacho card
 - *C1B (Forced)*: Tacho. download is enabled for this reader, C1B files will be downloaded more than once a day for a given Tacho card

When the Tacho. download is enabled on a controller, it adds a new event type: *Tacho. download*. This event allows to download the file if it succeeded, but if the Tacho. download stopped it contains the reason why it failed. The tacho. download is performed automatically on the controller when a Tacho. card is inserted.

9.10 RS-232 / Wiegand badges

This feature enables the RS-232 / Wiegand badge technology. It adds the following value to the Vehicles and Drivers identification modes:

- ***Advanced badge***

It also adds the *Passive RS-232* and *MR-Access* reader devices in the controller modules configuration.

9.11 Legacy Mifare badges

This feature enables the Legacy Mifare badge technology. It adds the following value to the Vehicles and Drivers identification modes:

- **Legacy Mifare badge**

It also adds the *GemProx* reader device in the controller modules configuration.

9.12 PIN-protected badge or code

This feature allows to use PIN security codes during the Vehicles and Drivers identification. For all the available identification modes, it adds a “+ PIN code” variant, which prompts for a PIN code after the badge or code identification.

9.13 US units

This feature enables United States customary units throughout the application. Kilometer units are replaced by miles, liters are replaced by gallons, L/100km is replaced by mpg, L/h is replaced by gal/h and millimeters are replaced by inches.

9.14 Cons. km/L

This feature displays all fuel consumptions in kilometer per liter (km/L) instead of L/100km.

9.15 AEAT exportación (AEAT export)

Activates the AEAT export, available for ES licenses only. Adds a *AEAT exportación* tab in the advanced parameters of the application, containing the following fields:

- **Formato** (Format): selects the formats used for the export. Available values: *CAE (.csv)*, *CIM (.xml)*, *CIM (.csv)*, *ATC Canaria – CAE (.xml)* and *ATC Canaria – CIM (.xml)*.
- **Posición de palabra de la matrícula** (registration word position): selects the position of the word in the *ID* field in the vehicle editor which allows to fetch the vehicle registration number.
- **NIF por vehículo** (NIF per vehicle): if enabled, allows to define a NIF code per vehicle.

This feature adds the *AEAT* tabs in the product, site and vehicle editors containing fields used by the export, depending on the format selected (see below). It also adds the *AEAT – NIF* field in the department editor.

This feature adds the *AEAT exportación* action in the *Fuel transactions* page which allows to generate a new export or to download the previous exported files.

9.15.1 CAE (.csv)

This format enables the following fields:

- Vehicles: *Elegible AEAT*, *NIF* (if the *NIF por vehículo__* option is enabled).
- Departments: *NIF*.
- Sites: *Elegible AEAT*, *NIF*, *CAE*, *Razón social*.
- Products: *AEAT – Código*.

The *AEAT exportación* export form for this format requires to select a *NIF* value, and exports transaction with the following filter:

- Product *Código*: not empty.
- Vehicle *AEAT eligible*: enabled.
- Site *AEAT eligible*: enabled.
- Site *CAE*: not empty.
- Vehicle / Department *NIF*: selected *NIF*.
- Site *NIF*: selected *NIF*.

The exported filename is “*aeat-[NIF]-[date].csv*”.

9.15.2 CIM (.xml)

This format enables the following fields:

- Vehicles: *Elegible AEAT*, *NIF* (if the *NIF por vehículo__* option is enabled).
- Departments: *NIF*.
- Sites: *Elegible AEAT*, *NIF*, *CIM*, *CodEE*, *Razón social*.
- Products: *AEAT – Código*.

The *AEAT exportación* export form for this format requires to select a *CodEE* value, and exports transaction with the following filter:

- Product *Código*: not empty.
- Vehicle *AEAT eligible*: enabled.
- Site *AEAT eligible*: enabled.
- Site *CIM*: not empty.
- Vehicle / Department *NIF*: not empty.
- Site *CodEE*: selected *CodEE*.

The exported filename is “*aeat-[CodEE]-[date].xml*”.

9.15.3 CIM (.csv)

This format enables the following fields:

- Vehicles: *Elegible AEAT, NIF* (if the *NIF por vehículo_* option is enabled).
- Departments: *NIF*.
- Sites: *Elegible AEAT, NIF, CIM, Razón social*.
- Products: *AEAT – Código*.

The *AEAT exportación* export form for this format requires to select a *Razón social* value, and exports transaction with the following filter:

- Product *Código*: not empty.
- Vehicle *AEAT eligible*: enabled.
- Site *AEAT eligible*: enabled.
- Site *CIM*: not empty.
- Vehicle / Department *NIF*: not empty.
- Site *Razón social*: selected *Razón social*.

The exported filename is “*aeat-[Razón social]-[date].csv*”.

9.15.4 ATC Canaria – CAE (.xml)

This format enables the following fields:

- Vehicles: *Elegible ATC Canaria, NIF* (if the *NIF por vehículo_* option is enabled).
- Departments: *NIF*.
- Sites: *Elegible ATC Canaria, NIF, CAE, Razón social*.
- Products: *ATC Canaria – Tipo*.

The *AEAT exportación* export form for this format requires to select a *NIF* value, and exports transaction with the following filter:

- Product *ATC Canaria – Tipo*: not empty.
- Vehicle *Elegible ATC Canaria*: enabled.
- Site *Elegible ATC Canaria*: enabled.
- Site *CAE*: not empty.
- Vehicle / Department *NIF*: not empty.
- Site *NIF*: selected *NIF*.

The exported filename is “*atc-[NIF]-[date].xml*”.

9.15.5 ATC Canaria – CIM (.xml)

This format enables the following fields:

- Vehicles: *Elegible ATC Canaria, NIF* (if the *NIF por vehículo_* option is enabled).
- Departments: *NIF*.
- Sites: *Elegible ATC Canaria, NIF, CIM, CodEE, Razón social*.
- Products: *ATC Canaria – Tipo*.

The *AEAT exportación* export form for this format requires to select a *CodEE* value, and exports transaction with the following filter:

- Product *ATC Canaria – Tipo*: not empty.
- Vehicle *Elegible ATC Canaria*: enabled.
- Site *Elegible ATC Canaria*: enabled.
- Site *CIM*: not empty.
- Vehicle / Department *NIF*: not empty.
- Site *CodEE*: selected *CodEE*.

The exported filename is “atc-[CodEE]-[date].xml”.

9.16 Excises export

Activates the Excises export, available for BE licenses only. Adds a *Excises export* tab in the advanced parameters of the application, containing the following field:

- **Product**: product on which the excises export apply.

This tabs also allows to define periods. Each period contains the following fields:

- **From (start date)**: start date of the period.
- **To (end date)**: end date of the period.
- **Rate**: refund rate used for this period.

This feature adds the *Excise* tabs in vehicles and in manual deliveries, both containing the same *Excises eligible* field.

This feature adds the *Excises export* action in the *Fuel transactions* page which allows to generate the export for the site and dates provided. Once the export is generated, it is possible to print it.

9.17 Name + First name

This feature adds the *First name* field in drivers collection and editor.

9.18 GIR mobile ID badges

This feature allows to use a smartphone to identify on controllers just like a RFID tag, using the “GIR Mobile ID” application available on Android and iOS. Requires a compatible terminal hardware.

It adds the following value to the Vehicles and Drivers identification modes:

- *GIR mobile ID badge*

9.19 Custom properties

This features allows to customize additional properties for vehicles and drivers.

It adds a new *Custom properties* tab in the *Advanced* section of the *Settings* page, containing the following fields:

- **Name:** property name displayed in the application.
- **Identifier:** technical identifier, used in transaction import, bulk import, bulk export and web services.
- **Entity:** *Vehicle*, *Driver*, *Fuel transaction* or *Department* (one value per department level)
- **Visibility:** if the selected *Entity* is not *Department*, defines how the custom property is displayed on the user interface. Possible values:
 - “*Properties*” tab: the custom property is placed into the *Properties* tab of the vehicle or driver editor (default value).
 - *Main tab* (“*Details*”): selecting this value allows to move this custom property into the *Details* tab of the vehicle or driver editor. It will also be part of the default columns for the *Vehicles* or *Drivers* page, and it will be available as a filter in the quick reports and quick graphs in the *Fuel transactions* page (only for properties of type *List*).
- **Type:** *Text*, *List*, *Decimal* or *Boolean* (boolean: selection between “true” and “false” values).
- **Values:** if the property *Type* is *List*, defines all the values available for this property.
- **Precision:** if the property *Type* is *Decimal*, defined the number of decimals digits for this property (default value is 2).

Once a property is added for vehicles or drivers, it is possible to edit it in the vehicle or driver editor. It is also possible to query it or display it in the list columns.

Custom properties can be updated in bulk import or web services. The fields name is generated using the properties *Identifier*: `xfields.<identifier>`

9.20 File-based import

This feature allows to automatically import vehicles and drivers using files.

It adds a *File-based import* field inside the *Impexp* tab in the *Advanced* section of the *Settings* page. This field contains the following options:

- **Vehicles:** enables vehicles file-based import
- **Drivers:** enables drivers file-based import

When the file-based import is enabled for vehicles or drivers, this adds a *Sync.* option in the vehicle/driver editor. This option is automatically enabled when a vehicle/driver is created from a file-based import, but can be manually updated later. The file-based import will only update or delete vehicle/driver records if their *Sync.* option is enabled.

9.21 Transaction import

This feature allows to import transactions manually or automatically from multiple sources.

It adds a *Transaction import* tab in the *Advanced* section of the *Settings* page. This tab manages the different import format providers used in the application. An import format provider contains the following fields:

- **Name:** provider name displayed in the application.
- **Identifier:** technical identifier, used in auto import.
- **Email:** for SaaS applications, it is possible to import a file automatically by sending an email to this address with the file attached. Only displayed if the *Script* field is not empty.
- **Directory:** for On-Prem applications, it is possible to import a file automatically by placing it the *impexp/import/todo* directory and respecting the pattern displayed in this field. Only displayed if the *Script* field is not empty.
- **Script :** allows to configure the script in lua. A click on *Test* allows to select a file in order to test the script currently being edited. It then displays warning / errors and created / duplicated transactions in the *Result* tab and more technical logs are listed in the *Console logs* tab.

This features adds a *Transaction import* action in the *Fuel transactions* page. This action opens a new window to import a transactions file manually by selecting the file and an import format provider. The manual file import first previews how the file will be imported by displaying warning/errors and created / duplicated transactions. Another click on *Import* really imports the file. The latest imported file are listed in the *Recent files* collapsible list.

This feature also adds a new *Provider* field in the *Fuel transactions* page, allowing to know from which provider the transaction was imported. In the *Quick reports* and *Quick graphs* views of the same page, it replaces the *Site* field

by a *Site/Provider* field, which allows to group (or filter) transactions by site or by provider.

The *Recent files* list is also available in the import format provider editor. From this list it is possible to download the file, to visualize the transactions created with this file and to re-import the file.

This feature adds the *Transaction import* notification in the user editor, which sends an email alert to all the users that have enabled the notification when a transaction file is automatically imported.

9.22 Editable prices

This feature adds the *Editable prices* tab in the settings *Advanced* section. This tab contains the following fields:

- ***Editable prices***: defines how editable prices are managed. Possible values:
 - *Disabled*: editable prices are disabled on this application.
 - *By department*: editable prices are defined by department (prices can be defined on each department level).
 - *By site*: editable prices are defined by site, or by regions if the *Regions* feature is enabled.
- ***Prevent price change after. . .***: displayed if the *Editable prices* field is not *Disabled*. If this field is not *None*, this enables the export of editable prices in fuel transaction (impexp API, custom and C4 formats) in the *custom_price* field. Possible values:
 - *None*: editable prices are not exported in *custom_price*.
 - *30 days*: editable prices are exported in *custom_price*, but is is not possible to edit fuel transactions prices if the transaction is older than 30 days.
 - *60 days*: editable prices are exported in *custom_price*, but is is not possible to edit fuel transactions prices if the transaction is older than 60 days.
- ***Scriptable business logic***: this field customizes the application behavior with the following scripts:
 - *on_transac_fuel_change()*: called each time a transaction is created or updated. Allows to customize the transactions *Notes*, editable price (*Unit. price*) and the transaction custom properties.
 - *on_auto_delivery_price()*: called each time an auto. delivery *Unit. price* field is modified. Allows to customize the computation of the *After* value, which updates the transactions *Tank unit. price*.

When the *Editable prices* field is not *Disabled*, this feature adds an *Edit price* action in the *Fuel transactions* page. This window allows to customize the prices set for all transactions, by editing a three-dimensional table:

- Each line of the table define the department (or the site if *By site* is selected in the *Editable prices* settings field) on which the price applies (or *Default price*, which matches all other departments/sites).

- The columns define the prices for each product. Clicking on a cell displays the prices for that department / product (or site / product if *By site* is selected) on a new window.
- For each department / product (or site / product if *By site* is selected), you'll find a list of *Date* and *Price* values which defines the periods used to apply the prices.

After updating prices, a click on *Save* will display a confirmation prompt if the price change impacts at least one transaction. The prompt displays an estimation of the number of impacted transactions. When confirmed, this will update the price of all the impacted transactions. The price of a transaction is defined by:

- The price set on that table for the transaction date, product and department (or site if *By site* is selected).
- If no price is set for a department (or site if *By site* is selected), the system uses the department's parent(s) (or the site region if *By site* is selected and the *Regions* feature is enabled).
- If no price is set in a whole department (or site if *By site* is selected) hierarchy, the system uses the *Default price* line.

Please note that saving prices may need to update a large number of transactions, so this operation can take up to several minutes.

This feature renames all the existing *Unit price* field in fuel transactions by *Tank unit price*. The new *Unit price* field with this feature enabled is the editable price.

9.23 Reminders

This feature allows to define periodic maintenance reminders for vehicles or drivers.

It adds the *Reminders* tabs in vehicle and driver editors.

A vehicle or driver reminder is defined by:


- **Name:** reminder name displayed in the application.
- **Type:** one of:
 - *Date:* reminder is due when the *Date* defined is reached
 - *Meter:* reminder is due when the vehicle *Odometer* or *Hour meter* has reached the value defined. When a vehicle has two meters and a reminder has two meters as well, the reminder is due when one of the meters is reached.
 - *Volume:* reminder is due when the vehicle has consumed the fuel *Volume* defined. Note that the reminder volume is decreased on each fuel transactions, and that all products are counted together.
Date reminders are available for both vehicles and drivers, *Meter* and *Volume* reminders are available for vehicles only.

When a reminder is due:

- an alert is shown in the Alert panel.
- an email is sent to all users who have the *Reminders* notification enabled.

A reminder is typically set back from due to not due by changing Date, Meter, or Volume to the next value.

An informal history of past reminders can be viewed in *Audit logs*.

Clicking the expand icon  in the *Reminders* tab, or the link in the Alert panel, opens the global reminders list. This view can be used to perform reports on all vehicles or drivers reminders, especially with the following fields:

- ***Due***: Whether a reminder is due or not.
- ***Remaining (days)***: Number of days left for a reminder of type *Date*.
- ***Remaining (km) or Remaining (mi)***: Distance left for a reminder of type *Odometer*.
- ***Remaining (h)***: Number of hours left for a reminder of type *Hour meter*.
- ***Current (km) or Current (mi)***: current vehicle *Odometer*.
- ***Current (h)***: current vehicle *Hour meter*.

Note that the *Remaining* fields can be used to sort reminders, and see which ones are next to be due.

Also note that negative *Remaining* values are possible when a reminder is due.

9.24 Regions

This feature creates an extra level in the sites hierarchy, allowing to group sites by region. Enabling this feature will create a default region named “-” which will contains all the existing sites.

When enabled, this feature shows the complete hierarchy including regions in the *Sites* tab of the application settings. In the site editor, a new *Region* field is added.

On all pages of the application, when a site is referenced its label includes the region name (ex: “Region > Site”). In query builders and columns, wherever the *Site* field is present, a *Region* field is also be in order to do reporting on that level.

In fuel transactions *Quick reports* and *Quick graphs*, this feature adds *Region* to the *Group by* fields and the *Site* filter will now allow to select regions as well.

The feature also allows to define user permissions and notifications on regions and not only on sites.

9.25 Zones

This feature allows to restrict access to some Fuelsets and some Accesses by defining zones, possibly with restricted time ranges.

When this feature is enabled, two zones types are available:

- **Predefined zones:** those zones are automatically created by the application when a Fuelset or an Access is added.
- **Custom zones:** those zones are created manually and can contain multiple Fuelsets or Accesses.

This feature adds the *Zones* tab in the *General* section of the application settings. A zone contains the following tabs:

- ***Details*:** lists the Fuelsets and Accesses contained in the zone. It can be only modified in custom zones.
- ***Restricted time ranges*:** allows to define authorized time ranges for that zone.
- ***Stay open schedule*:** allows to update *Stay open* time ranges if they were enabled in the controller settings (see below). It is only available for predefined Access zones.

This feature adds the following tabs in Fuelsets and Accesses windows in the controller settings:

- ***Zone “...”:*** allows to update the restricted time ranges for the predefined zone created for this Fuelset or Access.
- ***Stay open schedule*:** allows to enable and update the *Stay open* time ranges for this Access (see below).
- ***Custom zones*:** allows to update the list of custom zones containing that Fuelset / Access.

This feature adds the *Zones* tabs in vehicle and driver editors. It allows to define if a vehicle or driver is authorized everywhere, or limited to some zones (predefined or custom). If there are time restrictions, a *24/24* checkbox allows to bypass them.

When a restricted vehicle or a driver tries to access to an unauthorized site or time range, a **VEHICLE OUT OF ZONE** or **DRIVER OUT OF ZONE** message is displayed on the controller. For a Fuelset, a *Denied identification* event is stored, with the mention: *Out of zone*. For an Access, a transaction is stored with the *Denied (Out of zone)* status.

9.25.1 Stay open schedule

When a *Stay open schedule* is defined on an Access, it is possible to define time ranges when the access remains open. For example, it allows to schedule that a barrier should be kept opened during business hours. Four different time ranges can be set on each day, from Monday to Sunday.

The *Stay open schedule* tab also contains an *Exceptions* field. This field allows to select, create or update a calendar that defines calendar dates on which the stay open schedule doesn't apply. For example, this allows to avoid opening a barrier during business hours on holidays. To create a new calendar, select the *Add: Calendar* value. This will open a new window containing the following fields:

- **Name:** name of the calendar. A calendar created can be reused between multiple accesses.
- **Notes:** additional notes.
- **Details:** contains the list of rules defining the dates or the date ranges for this calendar. The order of the rules doesn't matter. Each rule contains a *Type* field and two dates fields. If only the first date field is defined, then the rule will only apply on this day. If both dates are defined, then the rule will apply on a date range. The *Type* field can have the following values:
 - *YYYY-MM-DD*: defines a unique date (ex: "2020-10-25").
 - *MM-DD*: defines a year-independent date (ex: "10-25").

In the supervision, if an Access is in the *Stay open* mode, its icon changes and an indicator is displayed in its popover.

The *Stay open schedule* option also adds the following access supervision actions (when the controller is online):

- **Stay open:** allows to force the stay open state for this access. The access will be opened as long as this state is enforced
- **Stay closed:** allows to force the stay closed state for this access. As long as the access is on this state, the access will be closed even
- **Auto:** stops the *Stay open* or *Stay closed* state and returns to the default stay open schedule

Using those supervision actions adds the following access transactions:

- *Stay open – Start* and *Stay open – End*
- *Stay closed – Start* and *Stay closed – End*

9.26 Custom zones only

Enabling this feature requires to enable the *Zones* feature first. When enabled, this feature hides predefined zones in *Vehicles* and *Drivers* pages. It also hides the predefined zones without stay open time ranges in the *Settings* page or the *Zones* dedicated page.

This feature allows to simplify the zones configuration in vehicles and drivers when there is a large number of controllers.

9.27 Advanced Vehicle Options

This feature enables finer-grained customization of certain characteristics for each individual vehicle rather than per model. All of the options such as the *NCE* and *Activity* code prompts as well as the product capacity and metering configuration that normally appear in the *Model* editor will instead appear in the *Vehicle* editor.

This feature also adds a *Vehicle tanks* page accessible from the menu in the top left corner of the *Vehicles* page. This page lists all the vehicle and product associations, and contains all the product fields accessible in the *Vehicles* page *Details* tab.

9.27.1 Default vehicle tank parameters

This feature adjusts the product editor: enabling the *Authorize by default* option will display default vehicle tank parameters in the *Default for new vehicles* tab. When this option is enabled, this product will be added by default to new vehicles, and the vehicle tank options for this product will have the same parameters as the ones defined in the product editor.

This feature also adds the *Default values* option below *Authorize by default*. This option also displays default vehicle tank parameters, but they are be used as default values when this product is explicitly added to a vehicle.

This feature also adds a new product *Type* value: *Fuel main engine, with unified gas mileage*. When selected, the product shares its mileage with other products of the same *Fuel main engine* type.

9.27.2 Consumption reporting

This feature also adds a *Cons.* tab next to *Meters* in the vehicle tank editor if at least one meter prompt is enabled. This tab is not available in the *Model* editor when the *Advanced Vehicle Options* feature is disabled. Allows to define minimum and maximum consumption values for the odometer or/and the hour meter for this product. The same fields are also available in the *Product* editor, when the *Authorize by default* option is enabled. When a minimum and maximum meters are defined in the *Cons.* tab, a small graph is displayed next to those values, representing the scale that will be used in fuel transactions.

When a vehicle tank has a consumption range defined, this adds a consumption graph on the vehicle's fuel transaction editor next to the *Cons.* fields. The consumption graph allows to determine visually if the vehicle consumption for this transaction is in the "normal" range by placing an arrow on a scale defined by the minimum and maximum values.

The color of the arrow in the graph allows to show if the consumption is normal (black), has moderate under/over consumption (orange) or high under/over consumption (red).

The consumption graph is also displayed in the fuel transactions quick reports *Avg. cons.* column if the *Vehicle* group by criterion is elected in the form.

9.28 Group authorizations

This features allows to restrict drivers to some vehicles or some vehicle groups. It adds a new *Groups* tab in the *Fuel* section of the *Settings* page. This collection allows to define vehicle groups.

It also adds the *Group* field in the *Vehicles* page. This field is used to select a group for this vehicle.

In the *Drivers* editor, this feature adds the *Authorization* field, which allows 3 values:

- **All groups & vehicles** (default value): the driver is authorized on all vehicles.
- **By group**: the driver is only authorized on some vehicles group. If selected, the user has to select which groups the driver is authorized on.
- **By vehicle**: the driver is only authorized on some vehicles. If selected, the user has to add which vehicles the driver is authorized on.

In the *Vehicles* editor, this feature adds the *Authorized drivers* tab. This tab displays the drivers authorized with this vehicle, regrouped in 3 sections. Each section contain an expand icon which allows to browse all the drivers in this section:

- The *Everywhere* section lists drivers authorized on all vehicles.
- The *Group: ...* section lists driver authorized on the vehicle group.
- The *Vehicle: ...* section lists drivers authorized on the vehicle specifically.

9.29 Quotas

This feature allows to limit the quantity of fuel distributed per vehicle or per department in a given time range. Enabling this feature requires to enable the *Advanced vehicle options* feature first.

This feature adds a *Quota* tab in the vehicle tank editor (available when clicking on a line in the *Products* list in the vehicle *Details* tab). This tab contains the following fields:

- **Type**: quota type for this vehicle and product. Possible values:
 - *None*: there is no quota for this product on the vehicle, there are no restrictions on the quantity of fuel distributed.
 - *Manual*: allows to manually set the remaining quantity of fuel that can be distributed to this vehicle and product. Once this quantity of fuel has been distributed, the quota has to be manually updated again to allow fuel distribution again on this vehicle and product.
 - *Recurring*: sets a recurring quota for this vehicle and product. Reinitializes automatically the remaining quota value every week or every month (see below).
 - *Inherited - Department*: uses the vehicle's department quota for this vehicle and product (if available, see below).
 - *Inherited - Direction*: uses the vehicle's direction quota for this vehicle and product (if available, see below).
 - *Inherited - Division*: uses the vehicle's division quota for this vehicle and product (if available, see below).
- **Remaining**: current remaining value of the quota for this vehicle and product. This value can be automatically updated when a transaction is received, or when the quota is reinitialized. Displayed when type *Type* field is not *None* or *Inherited*.

- **Recurring** : section displayed when the *Type* field is set to *Recurring*. Contains the following fields:
 - **Reinitialize to**: initial value of the quota (maximum quantity of fuel to distribute on the recurring period).
 - **the**: next date when the quota should be reinitialized.
 - **every**: defines the recurring period. Possible values: *week* or *month*.
 - **Rollover**: if set to *Yes* and if the remaining value of the quota is more than zero when it is reinitialized, then this remaining value is added to the initial value. Otherwise, if set to *No*, then the quota is set to the initial value in all cases.

When a quota is set on a vehicle product, a (*) indicator is added to the *Capacity* column of the *Products* list in the *Details* tab of a vehicle.

Quotas only apply to internal fuel transactions.

This feature also adds a *Quota* field in the fuel transaction editor which reports which quota has been decremented by the transaction. Possible values:

- *None*: no quota was used.
- *Vehicle*: vehicle tank quota was used.
- *Inherited – Department*: the department's quota was used.
- *Inherited – Direction*: the direction's quota was used.
- *Inherited – Division*: the division's quota was used.

When a fuel transaction is modified, quota is checked again. If a different quota should apply, the previous quota is re-incremented, the new one is decremented and *Quota* field is updated accordingly.

9.29.1 Departments quotas

Department quotas can be shared between multiple vehicles. The *Quotas* feature also adds a *Quota* field in the department editor. Possible values:

- *None*: no quota is defined for this department.
- *Volume*: a quota may be defined for each product (in L), similarly to vehicle tanks quotas. A *Quota – Volume* tab appears in the department editor.
- *Monetary*: only available when the *Editable price* feature is enabled. Allows to define monetary quota instead of a volume quota. Only one monetary quota can be defined by department, and it is applied to all products. In addition to the transaction volume, monetary quotas use the transaction editable price (and not the tank unit price). A *Quota – Monetary* tab appears in the department editor.

When a vehicle has a department and at least one department quota is configured in the app, the vehicle tanks quota *Type* field contains inherited department quotas to define which department level it uses (if a quota is defined on this department level).

Volume and *Monetary* quotas share common fields:

- **Type:** quota type for this department (and product if *Volume* quota). Possible values:
 - *None:* there is no quota for this department (and product if *Volume* quota), there are no restrictions on the quantity of fuel distributed.
 - *Manual:* allows to manually set the remaining quantity of fuel that can be distributed to this department (and product if *Volume* quota). Once this quantity of fuel has been distributed, the quota has to be manually updated again to allow fuel distribution again on this department (and product if *Volume* quota).
 - *Recurring:* sets a recurring quota for this department (and product if *Volume* quota). Reinitializes automatically the remaining quota value every week or every month (see below).
- **Remaining:** current remaining value of the quota for this department (and product if *Volume* quota). This value can be automatically updated when a transaction is received, or when the quota is reinitialized. Displayed when type *Type* field is not *None*.
- **Recurring :** section displayed when the *Type* field is set to *Recurring*. Contains the following fields:
 - **Reinitialize to:** initial value of the quota (maximum quantity of fuel or total monetary amount to distribute on the recurring period).
 - **the:** next date when the quota should be reinitialized.
 - **every:** defines the recurring period. Possible values: *week* or *month*.
 - **Rollover:** if set to *Yes* and if the remaining value of the quota is more than zero when it is reinitialized, then this remaining value is added to the initial value. Otherwise, if set to *No*, then the quota is set to the initial value in all cases.


When the department *Quota* field is *Volume* and if a product quota type is *Recurring*, a (*) indicator is added to the *Quota* column in the *Quota – Volume* tab.

9.30 Multiple Ident. Modes

This feature allows for multiple identification modes (EMG badge, Code, ID, etc) to be supported in a single GIR W250 instance. This is useful, for example, if different identification mode are used for vehicles at different sites. More than one *Ident. modes* option may then be enabled under *Settings / General / Ident. modes*. A specific *Ident. mode* must then be selected for each individual vehicle and driver in the corresponding editor.

If at least two identifications modes are defined for an entity (vehicle or driver), then the following additional features can be enabled from the *Ident. modes* tab of the application settings:

- *2 identifications per vehicle*
- *2 identifications per driver*

Enabling one of these features allows to define up to two identification methods for a given vehicle or driver, for example two different badges, or a badge and a code. Clicking on the  icon at the right of the *Identification* field in the vehicle or driver editor adds a new identification method.

Note that the *2 identifications per vehicle/driver* features requires all controllers to be upgraded with version 2.0.25 or higher, and can't be used at the same time as the "+ PIN code" feature.

See also the Settings / General / Ident. modes of this manual.

9.31 Tankers

This feature allows to configure some vehicles as tankers, which are vehicles that can deliver fuel to other vehicles.

This feature adds the *Tanker* field in the vehicle *Options* window. Possible values:

- **No**: this vehicle is not a tanker.
- **Yes (Manual)**: this vehicle is a tanker. The transactions delivered by this tanker are only entered manually.
- **Yes (Auto)**: this vehicle is a tanker. The transactions delivered by this tanker are retrieved from a controller on the tanker.

If the *Yes (Auto)* value is selected, a field requiring to select a site is displayed below. Selecting a site will associate this site with the tanker and the transactions fetched from the site controller will be considered delivered by this tanker. Only sites without tanks can be selected for a tanker. Once a site is associated to a tanker, it is no longer possible to add tanks to this site, and in the controller editor, the tanks area is replaced with a *Tanker* field, pointing to the tanker vehicle.

When a vehicle is configured as a tanker, the *Last transactions* tab is replaced by three tabs:

- **Last in**: last input transactions which have refueled the tanker.
- **Last out**: last output transactions delivered by this tanker to other vehicles.
- **Tanker graph**: represents the evolution of the tanker volume over time (see below).

The *Tanker graph* tab represents all the transactions inputs and outputs over time, providing an overview of fuel transactions to and from a tanker, and allowing to check if both streams are equivalent. If the vehicle has multiple products, it is possible to switch products on the top left corner. When enough data is available, it is possible to visualize a linear regression of the graph using

the icon at the top right corner: a well-behaved system where outputs match inputs should have a reasonably flat line.

This feature also adds a new *Tanker* type to the fuel transactions creation form. Transactions of type *Tanker* require the selection of a tanker vehicle in the *Tanker* field. Transactions fetched from controllers associated with *Auto* tankers automatically have the *Tanker* type, and the associated vehicle in the *Tanker* field.

9.32 Expiry management

Enabling this feature allows to forbid or authorize vehicles or drivers, or to set them an expiration date, after which the vehicle or driver will be forbidden. When a vehicle or driver is forbidden, it is no longer synchronized with controllers. If a forbidden vehicle or driver badge/code is entered on controllers, the terminal will then display “Unknown badge/code”.

This feature adds an *Authorization* field in the vehicles and drivers *Options* window. Possible values:

- ***Authorized***: the vehicle or driver is authorized on controllers
- ***Forbidden***: the vehicle or driver is forbidden, it will no longer be synchronized with controllers
- ***Authorized until***: the vehicle or driver is authorized until some date. Once the date is past, i.e. the next day at midnight, it will no longer be synchronized with controllers

The *Authorization* field value can also be imported in vehicles and drivers impexp API, bulk import and file-based import using the `active` field, available when this feature is enabled. Possible values:

- `"yes"`: authorizes the vehicle or driver
- `"no"`: forbids the vehicle or driver
- `"until:YYYY-MM-DD"`: authorizes the vehicle or driver until the date specified

9.33 Multiple timezones & languages

Enabling this feature adds a *Settings* tab in the controller pairing modal. This tab allows to customize the language and the timezone for this specific controller. The default value for those parameters is *Auto* which means they use the same language and timezone as the server.

This feature also adds a *Lang* field in the user editor, which defines the language used for this specific user.

9.34 Advanced MR-Pompe options

This feature adds advanced options in the MR-Pompe module editor:

- **Type:** Defines the pump module role:
 - *Standard:* default value, module is used for both counting and command.
 - *Command only:* module is used for command only. This adds the *MR-Pompe Count only* field in the pump module editor, which allows to set a related counting module by selecting its address.
 - *Count only:* module is used for counting only.
- **Options:**
 - *Negative volume:* indicates that fuel transactions on this pump should increase the tank stock instead of decreasing it.
- **Label:** custom optional label for this pump. When defined, this label replaces the “P<num> <product>” label on the application and on the controller, which can be useful when multiple fuel controllers are defined on the same site.


9.35 Prompt max volume

This feature adds the “Prompt max volume” option to models options (when *Advanced vehicle options* feature is disabled) or vehicles tanks options (when *Advanced vehicle options* feature is enabled). If this option is enabled for a vehicle, it adds a MAX VOLUME? prompt on the controller before each transaction is made with this vehicle. This will set the maximum volume for this transaction only.

9.36 Electric vehicles

This feature is active by default for GIR W250 SaaS applications. It adds management of electric vehicles (“EV”) and EV chargepoints. To activate the rest of the features, at least one product with the *Electric* type must be added to the application. Electrical products can be created from the *Products* tab in the *Settings* page. This new product type can be declared on vehicles, but also in the EV chargepoints settings.

9.36.1 EV Chargepoints

Chargepoints use the OCPP protocol to communicate with GIR W250 . The *Chargepoints* collection can be accessed from the dropdown menu in the top right corner of the *Sites* tab in the *Settings* page (by clicking on the  icon). New chargepoint can be only created when at least one product of type *Electric* is added to the application.

The chargepoint editor contains the following fields:

- **Site:** site where the chargepoint is located.
- **S/N:** chargepoint serial number. This parameter is used to pair the chargepoint with GIR W250 . If left empty, the chargepoint is not paired. When the chargepoint is paired, a communication icon is displayed to the right of this field to indicate whether the chargepoint is currently online or not.

The *Details* tab of the chargepoint editor contains the following fields:


- **Connectors:** list of chargepoint connectors. Connectors are the sockets on which a vehicle can be charged at the chargepoint. Each connector must be associated with a product of type *Electric*. Charges carried out on connectors not declared in this list will not be retrieved by the application.

The *Advanced* tab contains the following information:

- **Last synchronization:** date of the last OCPP message exchanged with the chargepoint.
- Under the *Last synchronization* field, the *Advanced* tab displays data from the last *BootNotification* message received from the chargepoint. This data contains detailed information on the chargepoint hardware.

The *Communication* tab displays events related to communication with the chargepoint (*Online*, *Offline*). Clicking on the icon at the top right of this tab displays the communication graph for this chargepoint in a new window.

9.36.2 Supervision

Chargepoints are also displayed in the *Supervision* page of their site. The supervision page for such a site contains a *Chargepoints* section below the fuel controllers, which displays the chargepoints and their connectors for that site. Clicking on a chargepoint icon opens a pop-up window displaying its serial number and its last synchronization date with GIR W250 if it is paired. Clicking on  next to the chargepoint's S/N redirects you to the chargepoint configuration page. The current status of the chargepoint is displayed in a *Status* field. This field can have the following values:

- **Available:** the chargepoint is ready to start new charges.
- **Error:** the chargepoint has reported an error and is not ready to start new charges.

The chargepoint's connectors are displayed to the right of its icon. Clicking on the connector icon opens a pop-up window containing the associated electrical product and the status of this connector in a *Status* field. This field can have the following values:

- **Available:** the connector is ready to start new charges.
- **Error:** the connector has reported an error and is not ready to start new charges.

- *Preparing*: the connector is about to start a new charge.
- *Charging*: the connector is currently charging.
- *Suspended EVSE*: charging is suspended at the EVSE (Electric Vehicle Supply Equipment) initiative.
- *Suspended EV*: charging is suspended because the Electric Vehicle is not ready.
- *Finishing*: charging has stopped, but the EV is still connected.
- *Reserved*: the connector is reserved.
- *Unavailable*: the connector has been blocked from supervision.

When a charge is not in progress on a connector, the pop-up window contains the following actions:

- *Block*: (if the connector is *Available*) blocks the connector, new charges won't be authorized.
- *Unblock*: (if the connector is *Unavailable*) unblocks the connector, new charges will be authorized again.

When a charge is in progress on a connector, its icon changes in the supervision page. Clicking on a connector in this state displays a window containing the details of the ongoing charge on that connector.

The supervision page also contains an additional tab when chargepoints are configured for the site: *Chargepoints history*. This collection contains the history of chargepoint and connector *Status* changes, and contains the following fields:

- *Date*: date when the status was stored by the chargepoint.
- *Site*: the chargepoint's site.
- *Chargepoint*: the chargepoint which stored the status change.
- *Connector*: the connector number (empty if it is a chargepoint status change).
- *Status*: the chargepoint or connector status stored (see above).

9.36.3 Ongoing charges

Ongoing charges are electrical transactions that have not yet been completed. The *Ongoing charges* collection can be accessed from the drop-down menu located in the top-left corner of the *Chargepoints* page.

It provides the following fields:

- *Date*: charge start date.
- *Product*: associated product.
- *Connector*: connector that started the charge.

- **Vehicle:** identified vehicle.
- **User:** identified user.

It also provides the following remote action (requires the chargepoint to be online):

- **Unlock:** stops the charge and physically unlocks the connector.

The *Advanced* tab contains additional information on the charge:

- **Meter start:** value of the connector counter at the start of the charge in wH.
- **Last meter:** last connector counter value received in wH.
- **Transaction ID:** identifier of the OCPP transaction associated with this charge.
- **ID Tag:** raw badge code sent by the chargepoint to start this charge.

Meters tab:

- Displays physical measurements retrieved from the charging point during the charge. This data can be displayed in graphical or tabular form.

9.36.4 Electrical transactions

Electrical transactions are stored when the charging on a chargepoint connector is complete. They are available in the *Fuel transactions* page. It is possible to differentiate a fuel transaction from an electrical transaction by using the *Product* field or the *Volume* column, which adds the watt-hour unit “(Wh)” in the case of an electrical transaction.

The editor for an electric transaction is slightly different from that for a fuel transaction. For an electric transaction, the *Tank*, *Pump* and *Volume (L)* (also shown *Volume (gal)*) fields are not displayed, while the following fields are added:

- **Connector:** contains the connector number used for this transaction. This connector number can be found in the connector configuration in the chargepoint editor.
- **Chargepoint:** displays the chargepoint that was used for this transaction.
- **Volume (Wh):** the energy used for this charge in Watt-hours.

9.36.5 OCPP logs

Details of OCPP messages exchanged between paired chargepoints and GIR W250 can be accessed from the *Chargepoints* page by clicking on the drop-down menu in the top-left corner, then by selecting the *OCPP logs* item.

OCPP logs contain the following fields:

- **Date:** date on which the OCPP message was received or sent.

- **Message type:** name of the associated OCPP operation.
- **Chargepoint:** label of the chargepoint.
- **Direction:** direction of the OCPP message. Possible values:
 - *CP* → *CS*: the message was initiated by the chargepoint (ChargePoint → CentralSystem).
 - *CS* → *CP*: the message was initiated by GIR W250 (CentralSystem → ChargePoint).
- **Request:** message request data.
- **Response:** message response data.

9.37 Other features

9.37.1 Hide volume units

This feature removes the *L* or *gal* units from the GIR W250 server application and from the controller terminal screens.

9.37.2 Show vehicles impexp IDs

This feature adds an *Impexp ID* field to the vehicle editor. This is used to provide compatibility with the ID requirements of external third-party applications. The *Impexp ID* value is then be used as the *vehicle.id* value when exporting data.

9.37.3 Show drivers impexp IDs

This feature adds an *Impexp ID* for drivers.

9.37.4 Show settings impexp IDs

This feature adds an *Impexp ID* for entities in the settings.

Appendix A

System prerequisites

A.1 Server machine

For an On-Prem installation with a few controllers, the machine hosting the GIR W250 server must meet the following requirements:

- Operating system: Windows 7, Windows Server 2008 R2 or later
- Memory: at least 2 GB
- Available disk space: the disk space used is very dependent on configuration and usage. Generally, it is recommended to have at least 10 GB available for the application.

For an On-Prem installation with many controllers (20 or more), the machine capabilities must be adapted to the expected usage. Contact GIR for more information.

A.2 Client machine

The client machine must run one of the supported web browsers:

- Microsoft Internet Explorer: version 11 or later. An up-to-date version of Microsoft Edge is recommended.
- Google Chrome, Mozilla Firefox: An up-to-date version of the consumer or enterprise releases. In practice, any reasonably recent version (less than one or two years old) should work.
- Apple Safari: version 9 or later
- Android Browser: version 4.4 or later

A.3 Storage capacities

GIR W250 server and controllers can support up to:

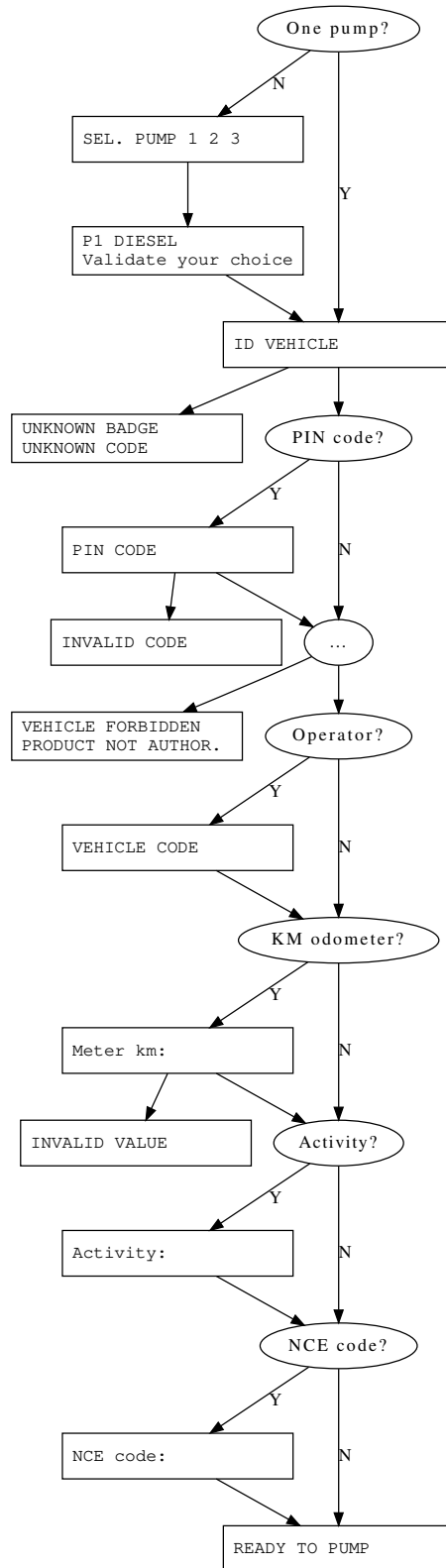
- 16 products
- 10000 vehicles
- 100000 vehicle tanks
- 10000 drivers
- 10000 vehicle/group authorizations in drivers (When the *Group authorizations* feature is enabled), with a maximum of 50 authorizations per driver
- 100 controllers
- 1000 activities
- 1000 models
- 1000 custom zones (when the *Zones* feature is enabled)
- 1000000 transactions (Controller autonomy: 2500 transactions)

Support for larger capacities can be studied on demand. Contact GIR for more information.

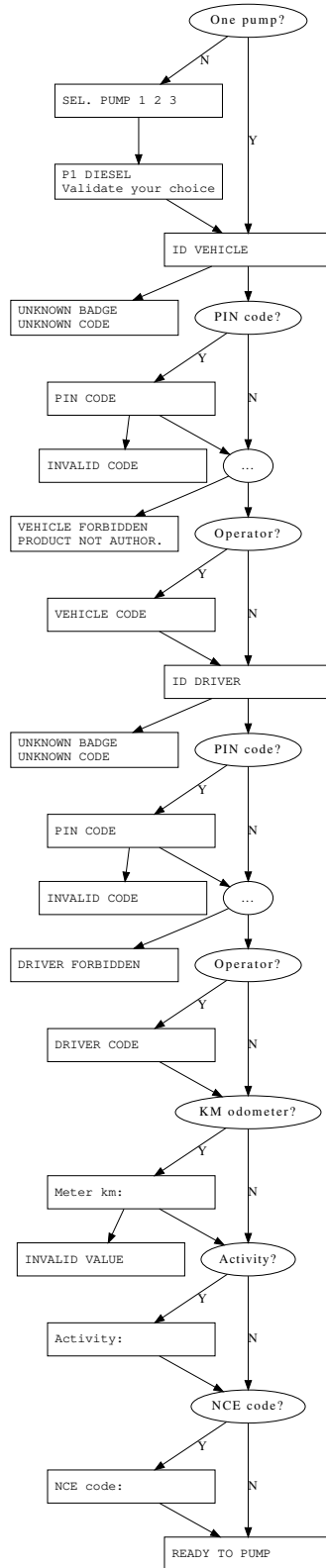
Appendix B

Fueling scenario

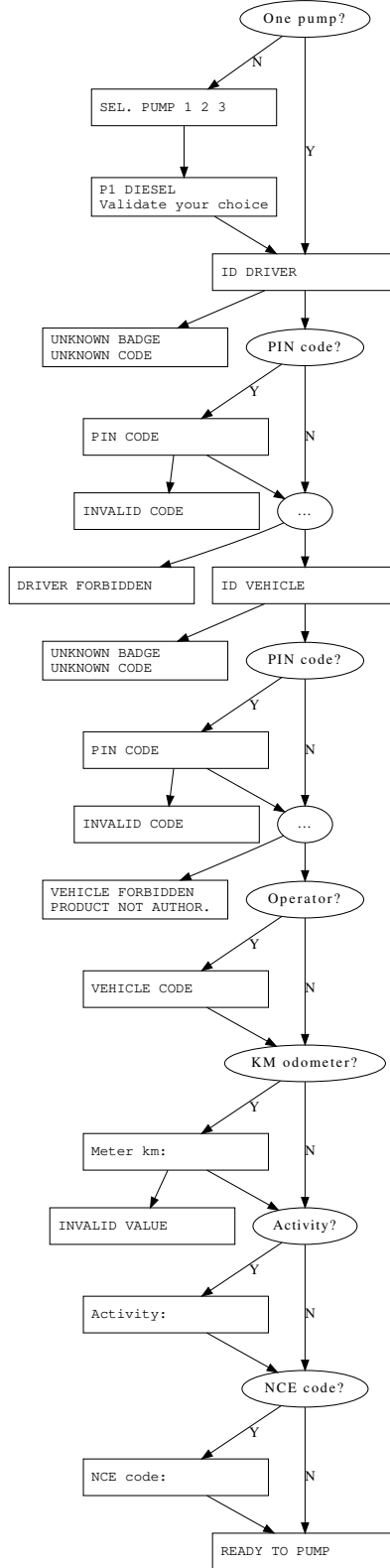
B.1 Vehicle only



B.2 Vehicle + driver



B.3 Driver + vehicle



Appendix C

Vehicles and drivers import and export

This chapter describes the methods available for importing or exporting vehicles and drivers.

These are mechanisms to automate the reading, creation, modification or deletion of vehicles or drivers, in order to synchronize this data between GIR W250 and a third-party system (for instance an employee directory, a fleet management software, ...).

Vehicle and driver import and export can be performed:

- programmatically, with web services
- manually, using “Bulk import” to copy/paste from a spreadsheet

Web services documentation can also be found online at <https://apidoc-gir-w250.klervi.com/>.

The following sections, Vehicles, Drivers, and Common mechanisms, describe the data model as well as a number of mechanisms that are common between all methods. The specifics of each import/export method are then discussed in the sections Web services and Bulk import.

C.1 Vehicles

This table describes the data model of vehicles, available in the `/api-impexp/vehicles` web service.

Name	Type	Description
<code>id</code>	ID	Vehicle identifier
<code>name</code>	Text	ID (label) of the vehicle (mandatory)
<code>ident_mode.id</code>	ID	Ident. mode identifier (6)
<code>ident_mode.name</code>	Text	Ident. mode name (6)
<code>badge</code>	Text	Badge used for identification
<code>pubsn_badge</code>	Text	Public S/N badge (read-only)
<code>code</code>	Text	Code used for identification
<code>idents[].ident_mode.id</code>	ID	Ident. mode identifier (7)
<code>idents[].ident_mode.name</code>	Text	Ident. mode name (7)
<code>idents[].badge</code>	Text	Badge used for identification (7)
<code>idents[].pubsn_badge</code>	Text	Public S/N badge (read-only) (7)
<code>idents[].code</code>	Text	Code used for identification (7)
<code>pin_code</code>	Text	PIN code
<code>driver_ident</code>	Boolean	Driver identification is enabled (10)
<code>model.id</code>	ID	Model identifier
<code>model.name</code>	Text	Model name
<code>department.id</code>	ID	Department identifier
<code>department.name</code>	Text	Department name
<code>department.parent.id</code>	ID	Direction identifier (1)
<code>department.parent.name</code>	Text	Direction name (1)
<code>department.parent.parent.id</code>	ID	Division identifier (1)
<code>department.parent.parent.name</code>	Text	Division name (1)
<code>kmeter</code>	Integer	Odometer (in km or miles)
<code>hmeter</code>	Decimal	Hour meter (in hours)
<code>activity_prompt</code>	Boolean	Activity code prompt (2)
<code>nce_prompt</code>	Boolean	NCE code prompt (2)
<code>notes</code>	Text	Additional notes for this vehicle
<code>vtanks[].product.id</code>	ID	Product identifier
<code>vtanks[].product.name</code>	Text	Product name

<code>vtanks[] .capacity</code>	Integer	Capacity (in L) (2)
<code>vtanks[] .capacity_duration</code>	Integer	Vol. max every (in hours) (2)
<code>vtanks[] .kometer_prompt</code>	Boolean	Prompt for odometer (2)
<code>vtanks[] .krange</code>	Integer	Odometer control range (in km or miles) (2)
<code>vtanks[] .kometer_force</code>	Boolean	Odometer control forceable or strict (2)
<code>vtanks[] .hometer_prompt</code>	Boolean	Prompt for hour meter (2)
<code>vtanks[] .hrange</code>	Integer	Hour meter control range (in hours) (2)
<code>vtanks[] .hometer_force</code>	Boolean	Hour meter control forceable or strict (2)
<code>vtanks[] .quota</code>	Integer	Quota left for that product (3)
<code>vtanks[] .vmax_prompt</code>	Boolean	If set to <code>true</code> , prompts for the max. volume before distribution (2) (9)
<code>vtanks[] .kcons_min</code>	Decimal	Minimum value for odometer consumption (in L/100km or MPG) (2)
<code>vtanks[] .kcons_max</code>	Decimal	Maximum value for odometer consumption (in L/100km or MPG) (2)
<code>vtanks[] .hcons_min</code>	Decimal	Minimum value for hour meter consumption (in L/H) (2)
<code>vtanks[] .hcons_max</code>	Decimal	Maximum value for hour meter consumption (in L/H) (2)
<code>xfields.example_identififier</code>	Text	Value for the <code>example_identififier</code> custom property (4)
<code>zone_limited</code>	Boolean	Set to <code>true</code> if the vehicle is restricted by zones (5)
<code>zones[] .id</code>	ID	Zone identifier (5)
<code>zones[] .name</code>	Text	Zone name (5)
<code>zones[] .access_24_7</code>	Boolean	Set to <code>true</code> if the vehicle bypasses the zone time restrictions (5)
<code>active</code>	Text	Authorizes or forbid the vehicle. Possible values: <code>"yes"</code> / <code>"no"</code> / <code>"until:YYYY-MM-DD"</code> (8)

- (1): depending on the number of hierarchy levels
- (2): only when the *Advanced vehicles options* feature is enabled
- (3): only when the *Quotas* feature is enabled
- (4): only when the *Custom properties* feature is enabled and if the corresponding identifier is defined in the settings
- (5): only when the *Zones* feature is enabled
- (6): only when multiple identification modes are defined for vehicles
- (7): only when the *2 identifications per vehicle* advanced identification parameter is enabled
- (8): only when the *Expiry management* feature is enabled
- (9): only when the *Prompt max. volume* feature is enabled
- (10): only when the identification mode is *Vehicle + Optional driver*

C.2 Drivers

This table describes the data model of drivers, available in the `/api-impexp/drivers` web service.

Name	Type	Description
<code>id</code>	ID	Driver identifier
<code>name</code>	Text	Driver name (mandatory)
<code>first_name</code>	Text	Driver first name (1)
<code>ident_mode.id</code>	ID	Ident. mode identifier (5)
<code>ident_mode.name</code>	Text	Ident. mode name (5)
<code>badge</code>	Text	Badge used for identification
<code>pubsn_badge</code>	Text	Public S/N badge (read-only)
<code>code</code>	Text	Code used for identification
<code>idents[].ident_mode.id</code>	ID	Ident. mode identifier (6)
<code>idents[].ident_mode.name</code>	Text	Ident. mode name (6)
<code>idents[].badge</code>	Text	Badge used for identification (6)
<code>idents[].pubsn_badge</code>	Text	Public S/N badge (read-only) (6)
<code>idents[].code</code>	Text	Code used for identification (6)
<code>pin_code</code>	Text	PIN code
<code>department.id</code>	ID	Department identifier
<code>department.name</code>	Text	Department name
<code>department.parent.id</code>	ID	Direction identifier (2)
<code>department.parent.name</code>	Text	Direction name (2)
<code>department.parent.parent.id</code>	ID	Division identifier (2)
<code>department.parent.parent.name</code>	Text	Division name (2)

<code>activity_prompt</code>	Boolean	Activity code prompt
<code>nce_prompt</code>	Boolean	NCE code prompt
<code>notes</code>	Text	Additional notes for this driver
<code>xfields.example_identifier</code>	Text	Value for the <code>example_identifier</code> custom property (3)
<code>zone_limited</code>	Boolean	Set to <code>true</code> if the driver is restricted by zones (4)
<code>zones[].id</code>	ID	Zone identifier (4)
<code>zones[].name</code>	Text	Zone name (4)
<code>zones[].access_24_7</code>	Boolean	Set to <code>true</code> if the driver bypasses the zone time restrictions (4)
<code>active</code>	Text	Authorizes or forbid the driver. Possible values: <code>"yes"</code> / <code>"no"</code> / <code>"until:<date in YYYY-MM-DD format>"</code> (7)

- (1): only when the *Name + First name* feature is enabled
(2): depending on the number of hierarchy levels
(3): only when the *Custom properties* feature is enabled and if the corresponding identifier is defined in the settings
(4): only when the *Zones* feature is enabled
(5): only when multiple identification modes are defined for drivers
(6): only when the *2 identifications per driver* advanced identification parameter is enabled
(7): only when the *Expiry management* feature is enabled

C.3 Common mechanisms

C.3.1 Identifiers

Each object (vehicles, drivers, departments, models, products...) is uniquely identified by its `id` field. This field must be unique, to avoid ambiguities when referencing an object.

GIR W250 automatically generates a unique identifier for all objects, as a UUID. It is recommended to use this auto-generated ID in most cases, as the guarantee of uniqueness is then assured by GIR W250 .

In some cases, it can be relevant to use external identifiers, typically coming from a third-party system. This is possible by enabling the “Impexp ID” features in the *Advanced / Features* tab of the application settings:

- *Show vehicle impexp IDs*: enables external IDs for vehicles.
- *Show driver impexp IDs*: enables external IDs for drivers.
- *Show settings impexp IDs*: enables external IDs for departments, models, products, sites and activities.

When impexp IDs are enabled for a given object type:

- An *Impexp ID* field is displayed in the application, where the external ID can be entered manually.
- An *Impexp ID* value must be defined in the `id` field when a new record is created with the API.
- An *Impexp ID* value must be set in the `id` field to reference an existing object.

When impexp IDs are enabled, the third-party system is fully responsible for setting the `id` field in the API, when creating new objects or referencing existing ones. If impexp IDs are enabled on an entity that doesn't have an *Impexp ID* value defined, GIR W250 falls back to the default auto-generated UUID. This situation should be considered as exceptional: for instance, it can happen just after the impexp IDs feature is enabled on a non-empty database, when all impexp ID values haven't been set yet. It is highly discouraged to mix Impexp IDs and UUIDs usage for a given entity: if you decide to use impexp IDs for vehicles, then all vehicles should have an *Impexp ID* value defined.

Caution: while using external IDs can sometimes simplify the development of an interface between GIR W250 and a third-party system, that system must then assume full responsibility for generating and maintaining unique IDs. We recommend enabling this feature only when the detailed technical implications are well understood. Also note that *Impexp IDs* are visible and editable manually in the application. If two IDs are identical, this can cause conflict errors.

C.3.2 References

Some fields in the data model are references to other entities, such as:

- `department`, `model`, and `product` in `vehicles`
- `department` in `drivers`

The behaviour when reading and writing *reference* fields is different than when dealing other fields. The properties within these fields (such as the `name` of a `department`) are shared between other records with a matching `id` property in the reference field. Changing any properties within the reference field of a record will also change the properties in the reference field of all other records that contain a matching `id` property.

When writing data (via POST or PUT request), if no `id` is provided, then an attempt is made to resolve the `id` value to an existing reference field value with the same `name` value. If no matching value is found, then a new `id` is assigned.

For instance, a PUT request on a vehicle containing `"model":{"name":"m1"}` will:

- Search if there is a model whose name is “m1”.
- If the “m1” model is found, assign it to the vehicle.
- If the “m1” model is not found, create a new “m1” model and assign it to the vehicle.

Resolving an entity uses the `id` field when it is set in a request, and the `name` field otherwise.

If both `id` and `name` are set, the search is performed by `id`, and the `name` field of the referenced entity is modified if needed, to match the request content.

Constraints on references

In `vehicles`, tanks are identified by their product (`vtanks[].product` field). As products are part of the protected settings, it is not possible to create a new product through a vehicle creation or update, but only to reference an existing product, by ID or by name. A vehicle can't have two tanks referencing the same product.

If several hierarchical levels of departments are configured, the `parent` field in the `department` object is a reference to the upper-level department. In such cases, references to a department in the `department` field of `vehicles` or `drivers` as well as in the `parent` fields of departments must be non-ambiguous and consistent with hierarchical levels. By specifying all of the `parent` department values when importing `vehicles` and/or `drivers`, it is possible to create a full hierarchical tree of departments.

C.4 Web services

Web services expose a REST API in JSON format.

They are available at `<app_url>/api-impexp/vehicles` and `<app_url>/api-impexp/drivers`, where `<app_url>` is the same URL you use to access the application web interface.

HTTP methods GET, POST, PUT, and DELETE are used to perform the different operations.

Vehicles are available at `/api-impexp/vehicles`:

- GET `/api-impexp/vehicles`: Read the vehicles list
- GET `/api-impexp/vehicles/<id>`: Read a vehicle
- POST `/api-impexp/vehicles`: Create a vehicle
- PUT `/api-impexp/vehicles/<id>`: Update a vehicle
- DELETE `/api-impexp/vehicles/<id>`: Delete a vehicle

Drivers are available at `/api-impexp/drivers`:

- GET `/api-impexp/drivers`: Read the drivers list
- GET `/api-impexp/drivers/<id>`: Read a driver
- POST `/api-impexp/drivers`: Create a driver
- PUT `/api-impexp/drivers/<id>`: Update a driver
- DELETE `/api-impexp/drivers/<id>`: Delete a driver

C.4.1 Authentication

Accessing web services requires authentication. An HTTP header "X-Klervi-API-Key: `<apikey>`" must be added to the request, where `<apikey>` is the value of the *Impexp API key* field in the *Advanced / Impexp* tab of application settings. If this header is missing or invalid, an authentication error is returned (401).

C.4.2 HTTP status codes

The API uses HTTP status codes to indicate the success or failure of an operation:

- a 2XX status code indicates a successful operation.
- a 4XX status code indicates an error due to request content. Retrying the same request will cause the same error.
- a 5XX status code indicates an internal or temporary error. Retrying the same request may succeed.

When an error occurs, a message in the response body contains a detailed description of the problem.

In the future, new fields may be added to the JSON body of server responses (both success and failure). The caller must be prepared for this, by not assuming anything about field names that are not presently documented in this API. The HTTP status code alone should be used to determine whether an operation is successful – not the fact that the response body is empty (as this behaviour may change in the future). Certain API requests do currently return empty objects upon successful completion.

The table below lists some common responses and their HTTP status code:

Code	Description	Cause
200	OK	Success
400	Bad Request	Request content is invalid (for instance: bad JSON, mandatory name field not set, inconsistent data...)
401	Unauthorized	The X-Klervi-API-Key header is missing or incorrect
404	Not found	GET request on an object whose id is unknown
405	Method Not Allowed	HTTP method is not supported on this resource
409	Conflict	The object identifier (id) is not unique
500	Internal error	Temporary error due to some server condition

C.4.3 Requests/Reponses content

This section describes the general structure of HTTP requests and responses. It uses URLs with the vehicles web services as an example, but all information here is generic and also applies to the drivers web services.

POST /api-impexp/vehicles

A POST request creates a new object:

- The request body must be a JSON object, matching the data model.
- Fields indicated as mandatory in the data model (for instance, for vehicles, the **name** field) must be defined.
- Other fields are optional and may be omitted, in which case the object is created with default values (for instance, if the **vtanks** field is omitted when creating a vehicle, it will be created with products defined as *Authorized by default* in the settings).

The response to a POST returns the created object in the **<result>** field.

PUT /api-impexp/vehicles/<id>

A PUT request modifies an existing object:

- The request body must be a JSON object, matching the data model.
- Omitted fields are left unchanged.

The response to a PUT returns the modified object in the `<result>` field.

DELETE /api-impexp/vehicles/<id>

A DELETE request deletes an existing object.

The response to a DELETE is an empty JSON object.

Note: a DELETE on a non-existing ID is considered as successful, and returns “200 OK”. See the Idempotency section for the reason of this behaviour.

GET /api-impexp/vehicles/<id>

A GET request on an individual object gets the data for this object.

The response to a GET returns the requested object in the `<result>` field.

GET /api-impexp/vehicles

A GET request on a collection lists all object of a given type (for instance all vehicles). This call accepts optional parameters in the “Query string”:

- `<limit>` defines the maximum number of objects to return in the response (by default: 500).
- `<offset>` defines the position of the first object to return in the `<result>` array (by default: 0, i.e. return the first object and its successors).

The response to a GET returns the following fields:

- `<result>`: an array containing at most `<limit>` objects, starting from position `<offset>`.
- `<offset>`: number of objects ignored before the first object returned. This value always matches the `<offset>` value set in the request.
- `<more>`: a boolean indicating if the limit of the `<result>` array is reached. When `<more>` is `true`, there remains objects to read, and a new GET request has to be performed, by incrementing the offset to get the rest of the data. When `<more>` is `false`, the end of the list has been reached and all objects have been read.

The `<limit>` and `<offset>` parameters coupled with the `<more>` field provide a paging mechanism, allowing to browse a full collection with several GET requests, even when the collection contains a lot of elements. In order to keep reasonable sizes for HTTP responses, it is recommended to not use “pages” of more than a few hundred elements. The server may restrict the maximum value of the `<limit>` parameter.

C.4.4 Idempotency

When a request is sent to the server and a network problem occurs, the caller may not get a response. In such a case, the caller will have to re-emit the request.

However, the caller may not always be able to know if the network error occurred when transmitting the request to the server, or when receiving the response. In the first case, there is nothing special to watch out for: the caller simply re-emits the request, and the server processes it. The second case is more complex as it requires that all requests are idempotent to avoid undesired duplicates. In other words, sending a request once or several times must produce the same result server-side.

- GET requests are idempotent by nature, as they do not change the server state.
- PUT requests are idempotent as well: if the server receives a PUT request for the second time, all fields will already have the correct value, and the server state will remain unchanged.
- DELETE requests are made idempotent by considering that deleting an unknown ID is always successful. This way, if the caller doesn't receive the response to a first DELETE request and re-emits it, it will still receive a "200 OK" response, indicating that the object is indeed no longer in the database.
- POST requests are the most problematic: as they do not contain an ID (except when using Impexp ID: see below), the server has no way to know if a POST request to create a new object has already been processed or not. This can be solved by the `X-Klervi-Idempotency-Key` HTTP header, which does the following:
 - The value defined by this header is called an "idempotency key"
 - When a POST request contains an idempotency key, the key is associated to the object created, for 24 hours after the first request has been processed.
 - If an idempotency key has already been processed, the rest of the POST request is ignored (as the creation already occurred), and the associated object is simply returned. In such a case, the request is equivalent to a GET request with the object identifier.

For POST requests without an Impexp ID, it is recommended to use the `X-Klervi-Idempotency-Key` header, with a random or content-related idempotency key, which is re-transmitted as is when there are retries. Without this header, a network error at the wrong time may cause duplicates to be created when there are retries.

Idempotency and Impexp ID

When using the *Impexp ID* feature (see the Identifiers section), POST requests can contain an external ID, and are then naturally idempotent. The `X-Klervi-Idempotency-Key` HTTP header remains operative if it is set, but it is no longer required.

For convenience, when external IDs are enabled, a PUT request on an unknown ID is interpreted as a creation operation, and is naturally idempotent as well.

C.4.5 Examples**Create a vehicle with a license plate and a badge**

Request:

```
POST /api-impexp/vehicles
```

```
{
  "name": "AA-001-AA",
  "badge": "20000001"
}
```

Response:

```
{
  "result": {
    "id": "bb1fe6ad-c986-4e4e-be2e-78fa5bf828e7",
    "name": "AA-001-AA",
    "badge": "20000001",
    ...
    "vtanks": [ {
      "product": {
        "id": "3e565959-e6b5-4de3-8c69-dbe3d476d893",
        "name": "DIESEL"
      }
    } ]
  }
}
```

Modify the badge of a vehicle

Request:

```
PUT /api-impexp/vehicles/bb1fe6ad-c986-4e4e-be2e-78fa5bf828e7
```

```
{
  "badge": "20000002"
}
```

Response:

```
{
  "result": {
    "id": "bb1fe6ad-c986-4e4e-be2e-78fa5bf828e7",
    "name": "AA-001-AA",
    "badge": "20000002",
    ...
  }
}
```

List all vehicles, in “pages” of ten

We assume the database contains 14 vehicles, from AA-001-AA to AA-014-AA.

Request 1: the `offset` parameter is implicitly zero.

```
GET /api-impexp/vehicles?limit=10
```

Response 1: the `result` array contains 10 vehicles, the `more` boolean to `true` indicates that there are more vehicles.

```
{
  "result": [
    { "name": "AA-001-AA", ... },
    { "name": "AA-002-AA", ... },
    { "name": "AA-003-AA", ... },
    ...
    { "name": "AA-010-AA", ... }
  ],
  "offset": 0,
  "more": true
}
```

Request 2: to continue listing vehicles, we set the `offset` parameter to 10.

```
GET /api-impexp/vehicles?offset=10&limit=10
```

Response 2: the `result` array contains 4 vehicles, the `more` boolean to `false` indicates that this is the end of the list.

```
{
  "result": [
    { "name": "AA-011-AA", ... },
    { "name": "AA-012-AA", ... },
    { "name": "AA-013-AA", ... },
    { "name": "AA-014-AA", ... }
  ],
  "offset": 10,
  "more": false
}
```

Read a vehicle

Request:

GET /api-impexp/vehicles/bb1fe6ad-c986-4e4e-be2e-78fa5bf828e7

Response:

```
{
  "result": {
    "id": "bb1fe6ad-c986-4e4e-be2e-78fa5bf828e7",
    "name": "AA-001-AA",
    "badge": "20000002",
    "code": "",
    "pin_code": "",
    "model": null,
    "department": null,
    "kmeter": 0,
    "hmeter": 0,
    "vtanks": [ {
      "product": {
        "id": "3e565959-e6b5-4de3-8c69-dbe3d476d893",
        "name": "DIESEL"
      }
    } ]
  }
}
```

Delete a vehicle

Request:

DELETE /api-impexp/vehicles/bb1fe6ad-c986-4e4e-be2e-78fa5bf828e7

Response:

{}

The deleted vehicle is no longer in the vehicles list, and a GET on its id returns a 404 error.

Create a driver with a name and a code

Request:

POST /api-impexp/drivers

```
{
  "name": "Martin",
  "code": "1234"
}
```

Response:

```
{
  "result": {
    "id": "a3cffa32-f795-4a21-9131-1127f7c8efcf",
    "name": "Martin",
    "code": "1234",
    ...
  }
}
```

Read a driver

Request:

```
GET /api-impexp/drivers/a3cffa32-f795-4a21-9131-1127f7c8efcf
```

Response:

```
{
  "result": {
    "id": "a3cffa32-f795-4a21-9131-1127f7c8efcf",
    "name": "Martin",
    "first_name": "",
    "badge": "",
    "code": "1234",
    "pin_code": "",
    "department": null,
    "activity_prompt": false,
    "nce_prompt": false
  }
}
```

Modify the department of a driver

Request:

```
PUT /api-impexp/drivers/a3cffa32-f795-4a21-9131-1127f7c8efcf
```

```
{
  "department": { "name" : "D1" }
}
```

Response: the driver has been assigned to the “D1” department, which has been created if needed.

```
{
  "result": {
    "id": "a3cffa32-f795-4a21-9131-1127f7c8efcf",
    "name": "Martin",
    "code": "1234",
    "department": {
      "id": "702f6726-bd13-4916-adb9-9840064cd7bf",
      "name": "D1"
    }
  }
}
```

```

    },
    ...
  }
}

```

C.5 Bulk import

The bulk import mechanism allows to quickly import into the application vehicles or drivers from a spreadsheet.

The imported or exported data is constituted by a first header line defining the fields for each column, followed by N lines of vehicles or drivers. The fields in a line are separated by tab characters.

To access to that feature, navigate to the *Vehicles* or the *Drivers* page. Then click directly on the info ⓘ icon in the action menu, without selecting any vehicle or driver to the list. This will open a new window containing the *Bulk operations* tab. When you select this tab, the following actions are available:

- **Bulk update:** allows to update or delete all the records currently displayed in the collection.
- **Bulk import:** allows to import multiple drivers or vehicles at the same time.
- **Bulk export:** allows to export all the drivers or vehicles currently displayed in the collection.

C.5.1 Basic usage

To correctly import vehicles or drivers, this is the recommended workflow:

1. First, configure the application settings to your needs (features, products).
2. Create a sample vehicle or driver in the application. If needed, create also a sample department, model or activity.
3. Click on *Bulk import* in the *Bulk operations* tab of the info ⓘ window.
4. On the top half of the window, you'll find your sample vehicle or driver in a tab-separated text format. Copy/paste the contents of this data to an empty spreadsheet to get the import columns, and your sample vehicle or driver as an example.
5. Fill your spreadsheet with all the vehicles or drivers to import. The first column should be the primary key used to resolve existing vehicle or drivers (example: code, badge, name).
6. Copy/paste your spreadsheet cells (**including the headers**) to the bottom half text area of the import window.
7. Click on *Preview* and check that there are no errors.
8. Finally, click on *Import* to save the vehicles or driver imported.

C.5.2 Bulk import fields

All fields defined in the Vehicles and Drivers sections can be used. For example:

- `name`: vehicle/driver name.
- `department.parent.name`: direction name.
- `vtanks[0].product.name`: 1st product name.

Those field names can be a bit complicated, as they are intended for programmatic usage. To make manual usage easier, aliases are used automatically in *Bulk export* and *Bulk import*, for the most common cases.

For instance, the following aliases are available for vehicles:

Alias	Original field name	Type	Description
<code>model</code>	<code>model.name</code>	Text	Model name
<code>department</code>	<code>department.name</code>	Text	Department name
<code>department.id</code>	<code>department.id</code>	ID	Department identifier
<code>department.name</code>	<code>department.name</code>	Text	Department name
<code>direction</code>	<code>department.parent.name</code>	Text	Direction name (1)
<code>direction.id</code>	<code>department.parent.id</code>	ID	Direction identifier (1)
<code>direction.name</code>	<code>department.parent.name</code>	Text	Direction name (1)
<code>division</code>	<code>department.parent.parent.name</code>	Text	Division name (1)
<code>product1</code>	<code>vtanks[0].product.name</code>	Text	1st product name
<code>product1.id</code>	<code>vtanks[0].product.id</code>	ID	1st product identifier
<code>product1.name</code>	<code>vtanks[0].product.name</code>	Text	1st product name
<code>product1.capacity</code>	<code>vtanks[0].capacity</code>	Integer	1st product capacity (in L) (2)
<code>product2</code>	<code>vtanks[1].product.name</code>	Text	2nd product name
<code>zone1</code>	<code>zone[0].name</code>	Text	1st zone name (3)
<code>zone1.access_24_7</code>	<code>zone[0].access_24_7</code>	Text	1st zone access_24_7 (3)
<code>ident</code>	<code>ident_mode.name</code>	Text	Ident. mode name (4)

(1): depending on the number of hierarchy levels

(2): only when the *Advanced vehicles options* feature is enabled

(3): only when the *Zones* feature is enabled

(4): only when multiple identification modes are defined for vehicles

The following fields are also available for vehicles and drivers bulk import (they are however not exported):

Name	Type	Description
<code>tps_id</code>	Text	Third-party ID used by Third-party services (1)

(1): only when the *Third-party services* feature is enabled

C.5.3 Bulk import preview

In the *Bulk import* window, after you click on *Preview*, you can review the changes that will be applied to the database before clicking on *Import*:

- **Pending changes:** summary of the changes that will be applied to the database once the import is validated.
- **Errors:** list of warnings or errors ordered by line number.
- **Preview:** list of vehicles or drivers as they would be seen in the application once the import is validated. A click on a line opens a window showing the details for that record.

Click on *Import* to validate those changes or on *Back* to go back to the previous screen.

Appendix D

Transactions export

This appendix describes the different transactions export methods available:

- The HLF1 file format export
- The C4 file format export
- The transactions export web service

D.1 Export mechanisms

D.1.1 File export

The file export format and parameters can be defined in the application settings, in the *Advanced / Impexp* tab. When enabled, it exports files in the `impexp/export/todo` directory (on Windows it's typically `C:\girw250\data\impexp\export\todo`). When you are done with an export file, we recommend you to move this file to the `impexp/export/done` directory.

One parameter in particular (*Transac. change*) defines how transactions changes are tracked:

- *None (do not export changes)*: only transactions creations are exported.
- *Update (one new line per change)*: a transaction modification is exported as a new line which updates a previous transaction.
- *Diff (two new lines per change)*: a transaction modification is exported as two lines: the first line cancels the previous transaction, and the second line creates a new transaction with updated values.
- *Full diff (multiple new lines per change)*: a transaction modification is exported as at least two lines, and sometimes more, in case of cascading changes on the following fields: `kdelta`, `kcons`, `hdelta`, `hcons`.

D.1.2 Web service export

The web service exports uses HTTP calls and returns the exported data in the JSON format.

In order to access those web services, you'll have to add the "X-Klervi-API-Key: <apikey>" HTTP header where <apikey> is the value of the *Impepx API key* field in the *Advanced / Impepx* tab of your application settings. You also have to allow the API key to read the application data: the *Authorization* field in the advanced window of this setting needs to be set to *Read-only* or *Read/Write*.

The JSON response is structured like this:

```
{
  "result": [
    {
      "id": "05008e00-5d00-0000-005a-005500e336f1",
      ...
    },
    ...
    {
      "id": "0f000d00-7800-4800-005a-005500e43745",
      ...
    }
  ],
  "more": true
}
```

- <result> is the array containing the exported data. Please note that if the size of this array hits a predetermined limit, we'll stop sending you data and you'll have to call the web service once again with the right parameters to get the rest of the data.
- <more> is a boolean set to true if the limit of <result> is hit and there is some data left.
- <id> is an unique identifier for an exported element.

If <more> is set to true and you'd like to fetch more data, just add the `?last_id=lastidfetched` query-string parameter to the URL where <lastidfetched> is the last exported element <id> you fetched with this web service. The response will then contain the following elements. For the example above, you would use `?last_id=0f000d00-7800-4800-005a-005500e43745`.

`?change=mode` is another useful query-string parameter defining how transactions changes are tracked, where <mode> can take the following values:

- **none**: only transactions creations are exported.
- **update**: a transaction modification is exported as a new element which updates a previous transaction.
- **diff**: a transaction modification is exported as two elements: the first one cancels the previous transaction, and the second one creates a new transaction with updated values.
- **fulldiff**: a transaction modification is exported as at least two elements, and sometimes more, in case of cascading changes on the following fields: `kdelta`, `kcons`, `hdelta`, `hcons`.

If no <change> parameter is added, the default behavior of the web service is the same as `?change=diff`.

D.1.3 Recommendations

- Don't make any assumption about the order of the exported transactions. For instance, they are not always ordered by date, e.g. when there are modifications, or when some transactions were not received by the server just after being made. The export mechanism with the `<last_id>` parameter guarantees that all transactions are exported.
- New columns or fields can be added to the export formats anytime.

D.1.4 Fuel transactions ledger

The lines exposed by the web service and the file export when the *Transac.change* parameter is set to *Diff* can be visualized in the application web interface by selecting a record in the *Fuel transactions* page, then by opening the *Info* window and displaying the *Ledger* tab. The *id* column in that tab match the *id* field in the web service response or C4 format, which is the export line UUID and also the value to pass to the `<last_id>` parameter of the web service.

The detailed view of each record in this tab contains all the exported fields using the C4 format below.

At the top right corner of the tab, a link allows to access to the dedicated *Fuel transactions ledger* page.

D.2 HLF1 format

D.2.1 General

The HLF1 export format exported by GIR W250 is fully compatible with GIR W100 and GIR W200 servers HLF1 export format.

For a full description of the *GIR HLF1* format please refer to the *GIR Titan-Hykkoris* (W100) documentation.

NB.: when creating a new file-based interface please consider using the newer C4 export format described below.

D.2.2 Export IDs (Identifiers)

Each exported HLF1 row is identified by its unique *id*. An exported row may handle 3 different cases:

- Creation of a fuel transaction: exported *id* is a new unique identifier.
- Deletion of a fuel transaction: the exported *id* is the same as the initial transaction creation.
- Update of a fuel transaction: exported *id* is a new unique identifier.

WARNING: The HLF1 format doesn't provide any trivial way to know which transaction has been updated when using the *Update (one line per change)* option. Therefore the *Diff (two new lines per change)* should generally be used when tracking changes with the HLF1 export format.

D.3 C4 format

D.3.1 General

The C4 exported files are text files containing fuel transactions. Each line ends with CRLF (ASCII hex codes 0x0D & 0x0A) and represents a transaction, except the first one if there's a header. The file is encoded in UTF-8.

All fields are separated by a comma (, - ASCII hex code 0X2C) and can take an unlimited number of characters.

Field types:

- Text values are delimited by a double quote sign (" - ASCII hex code 0x22)
- ID values are also delimited by double quotes
- Datetime values are in UTC in the ISO 8601 format (i.e.: YYYY-MM-DDTHH:mm:ssZ), in double quotes as well
- Decimal values use the dot notation (. - ASCII hex code 0x2E) with a possible minus sign in front of it (e.g.: -3.14), without space.
- Boolean values are either `false` or `true` without quotes

D.3.2 Export IDs (Identifiers)

Each exported C4 row is identified by its unique `id`. An exported row may handle 3 different cases:

- Creation of a fuel transaction: exported `id` is a new unique identifier. Actual transaction is identified by the `transac_id` field.
- Deletion of a fuel transaction: exported `id` is a new unique identifier. Actual transaction is identified by the `transac_id` field. NB: in this case the `_deleted` field is set to `true`.
- Update of a fuel transaction: exported `id` is a new unique identifier. Actual transaction is identified by the `transac_id` field.

D.3.3 Fuel transaction URL

A link to a fuel transaction in GIR W250 can be built with:

`https://<domain>/r/transac_fuels/<transac_id>`

- `<domain>`: the domain name you use to access your application
- `<transac_id>`: the `transac_id` field (ex: 338d031b-cef7-43f0-afba-dc0f1b5b76f0)

D.3.4 Columns

The table below lists the CSV columns for the GIR C4 format. Additional columns may be added after those described here, but the position of the following columns is guaranteed not to change.

Position	Name	Type	Description
01	<code>id</code>	ID	Export line UUID (unique)
02	<code>transac_id</code>	ID	Transaction public ID
03	<code>_deleted</code>	Boolean	<code>true</code> if the transaction was cancelled
04	<code>type</code>	Text	<code>int</code> (internal) or <code>ext</code> (external). Other possible value: <code>tanker</code> (3)
05	<code>date</code>	Datetime	Date + time in UTC
06	<code>vehicle.id</code>	ID	Vehicle public ID
07	<code>vehicle.name</code>	Text	Vehicle name (labeled as “ID” in the UI)
08	<code>vehicle.badge</code>	Text	Vehicle badge (4)
09	<code>vehicle.code</code>	Text	Vehicle code (4)
10	<code>driver.id</code>	ID	Driver public ID
11	<code>driver.fstname</code>	Text	Driver first name
12	<code>driver.name</code>	Text	Driver name
13	<code>driver.badge</code>	Text	Driver badge (4)
14	<code>driver.code</code>	Text	Driver code (4)
15	<code>department.id</code>	ID	Department public ID
16	<code>department.name</code>	Text	Department name
17	<code>product.id</code>	ID	Product public ID
18	<code>product.name</code>	Text	Product name
19	<code>volume</code>	Decimal	Distributed volume (L or gal) (1)
20	<code>unit_price</code>	Decimal	Unit price (1)
21	<code>kmeter</code>	Integer	Odometer (km or Miles) (1)
22	<code>hmeter</code>	Decimal	Meter in hours
23	<code>activity.id</code>	ID	Activity public ID
24	<code>activity.name</code>	Text	Activity name
25	<code>nce_code</code>	Text	NCE code
26	<code>site.id</code>	ID	Site public ID
27	<code>site.name</code>	Text	Site name
28	<code>pump</code>	ID	Pump number
29	<code>manual</code>	Boolean	<code>true</code> if created manually
30	<code>mtr_forced</code>	Boolean	<code>true</code> if the meter was forced
31	<code>vol_max</code>	Boolean	<code>true</code> if the maximum capacity was reached

32	<code>new_kmeter</code>	Boolean	<code>true</code> if the odometer was initialized
33	<code>new_hmeter</code>	Boolean	<code>true</code> if the hour meter was initialized
34	<code>local_date</code>	Text	Local date in the YYYY-MM-DD format
35	<code>local_time</code>	Text	Local time in the HH:MM:SS format
36	<code>line_id</code>	Integer	Consecutive integer between 0 and 999999999 replacing <code>id</code> for legacy systems
37	<code>kdelta</code>	Integer	Covered distance (km or Miles) (1)
38	<code>kcons</code>	Decimal	Fuel consumption (km/L or MPG) (1)
39	<code>hdelta</code>	Decimal	Covered time (h)
40	<code>hcons</code>	Decimal	Fuel consumption (L/h)
41	<code>custom_price</code>	Decimal	Unit price assigned using the “Edit prices” window (2)
42	<code>tanker.id</code>	ID	Tanker vehicle public ID (3)
43	<code>tanker.name</code>	Text	Tanker vehicle name (labeled as “ID” in the UI) (3)
44	<code>tanker.badge</code>	Text	Tanker vehicle badge (3)
45	<code>tanker.code</code>	Text	Tanker vehicle code (3)
46	<code>dest_type</code>	Text	<code>tanker</code> if the transaction was delivered to a tanker vehicle, or <code>vehicle</code> otherwise (3)
47	<code>tank.id</code>	ID	Tank public ID
48	<code>tank.name</code>	Text	Tank name

Fields of type “ID” contain the “Public ID” for a given entity, using the rules defined in the Identifiers section of the “Vehicles and drivers import and export” chapter.

Those fields are either UUIDs or “Impexp IDs” depending on the settings for the entity.

Especially, for vehicles, the `vehicle.id` field is the “Public ID” of a vehicle, and is not to be confounded with the `vehicle.name` field. `vehicle.name` is the main field used to name a vehicle, and typically appears as the first column in the vehicles menu UI, with the label “ID” in the English version of GIR W250 .

(1) Unit depends on settings or area

(2) Only set if the “Editable price” W250 feature is enabled and if the “Editable prices” field in the Advanced / Impexp settings is not “None”

(3) Only set if the “Tanker vehicles” W250 feature is enabled

(4) If the “2 identifications per vehicle/driver” option is enabled for this vehicle/driver record, this field uses the first badge/code displayed in the vehicle/driver record

D.4 Web service /api-impexp/transac_fuels

D.4.1 URL

This web service provides access to fuel transactions on the following URL:

GET <app_url>/api-impexp/transac_fuels

<app_url> is the same URL you use to access the application web interface.

D.4.2 Fields

This export format use the same fields as the C4 file export, but in JSON. Additional fields are also available, the detail is available on our online documentation for the impexp API: <https://apidoc-gir-w250.klervi.com/>

Example of JSON data:

```
{
  "result": [
    {
      "id": "05008e00-5d00-0000-005a-005500e336f1",
      "transac_id": "338d031b-cef7-43f0-afba-dc0f1b5b76f0",
      "_deleted": false,
      "type": "int",
      "date": "2018-01-10T09:57:52Z",
      "vehicle": {
        "id": "120",
        "name": "HDW-7034",
        "badge": "",
        "code": "2324"
      },
      "driver": {
        "id": "211",
        "fstname": "Andrew",
        "name": "Nelson",
        "badge": "20000047",
        "code": ""
      },
      "department": null,
      "product": {
        "id": "1",
        "name": "GOI"
      },
      "volume": 6.70,
      "unit_price": 0.00,
      "kmeter": null,
      "hmeter": null,
      "activity": null,
    }
  ]
}
```

```
    "nce_code": "",
    "site": {
      "id": "2",
      "name": "Station"
    },
    "pump": "1",
    "manual": false,
    "mtr_forced": false,
    "vol_max": false,
    "new_kmeter": false,
    "new_hmeter": false,
    "local_date": "2018-01-10",
    "local_time": "10:57:52",
    "line_id": 156,
    "kdelta": null,
    "kcons": null,
    "hdelta": null,
    "hcons": null
  },
  ...
],
"more": false
}
```

Appendix E

Remote Control and Monitoring API

This chapter describes the web services available to perform remote control and monitoring on controllers.

E.1 Web services

Web services expose a REST API in JSON format.

They are available at `<app_url>/api-impexp/`, where `<app_url>` is the same URL you use to access the application web interface.

HTTP methods GET, POST, PUT, and DELETE are used to perform the different operations.

Sites are available at `/api-impexp/rcm/sites`:

- GET `/api-impexp/rcm/sites`: Read the list of sites list
- GET `/api-impexp/rcm/sites/<id>`: Read a site

Supervision actions are available at `/api-impexp/rcm`:

- POST `/api-impexp/rcm/block_pump`: block a pump
- POST `/api-impexp/rcm/unblock_pump`: unblock a pump
- POST `/api-impexp/rcm/start_pump`: start a remote transaction
- POST `/api-impexp/rcm/force_refresh`: triggers a new gauging

Inventories are available at `/api-impexp/inventories`:

- GET `/api-impexp/inventories`: Read the inventories list (works the same as GET `/api-impexp/transac_fuels`)

E.2 Sites

This table describes the data model of sites in the `/api-impexp/rcm/sites` web service.

Name	Type	Description
<code>id</code>	ID	Site public ID
<code>name</code>	Text	Site name
<code>controllers[] .sn</code>	Text	Controller S/N
<code>controllers[] .online</code>	Boolean	true if the controller is online
<code>controllers[] .date</code>	Datetime	Last time the system communicated with this controller
<code>tanks[] .id</code>	ID	Tank public ID
<code>tanks[] .name</code>	Text	Tank name
<code>tanks[] .product.id</code>	ID	Product public ID
<code>tanks[] .product.name</code>	ID	Product name
<code>tanks[] .capacity</code>	Integer	Tank capacity (L or gal)
<code>tanks[] .volume</code>	Decimal	Gauged volume (L or gal)
<code>tanks[] .volume_date</code>	Datetime	Date and time when gauge was last read
<code>pumps[] .id</code>	ID	Pump public ID
<code>pumps[] .name</code>	Text	Pump name
<code>pumps[] .num</code>	Text	Pump number
<code>pumps[] .product.id</code>	ID	Product public ID
<code>pumps[] .product.name</code>	Text	Product name
<code>pumps[] .tank.id</code>	ID	Tank public ID
<code>pumps[] .tank.name</code>	Text	Tank name
<code>pumps[] .blocked</code>	Boolean	true if the pump is currently blocked
<code>pumps[] .blocked_reason</code>	Text	Pump blocking reason: supervision , zero_volumes , error or tank_block
<code>pumps[] .manual</code>	Boolean	true if the pump is currently in manual mode
<code>pumps[] .pumping</code>	Boolean	true if the pump is currently delivering fuel, either when a transaction is in progress, or when it's in manual mode
<code>pumps[] .transaction.date</code>	Datetime	Date + time when the transaction on this pump started
<code>pumps[] .transaction.vehicle.id</code>	ID	Vehicle public ID
<code>pumps[] .transaction.vehicle.name</code>	Text	Vehicle name or registration
<code>pumps[] .transaction.driver.id</code>	ID	Driver public ID
<code>pumps[] .transaction.driver.name</code>	Text	Driver name

<code>pumps[].transaction.activity_id</code>	ID	Activity public ID
<code>pumps[].transaction.activity_name</code>	Text	Activity name
<code>pumps[].transaction.nce_code</code>	Text	NCE code
<code>pumps[].transaction.kmeter</code>	Integer	Odometer (km or miles)
<code>pumps[].transaction.hmeter</code>	Decimal	Meter in hours
<code>accesses[].id</code>	ID	Access public ID
<code>accesses[].name</code>	Text	Access name
<code>accesses[].stay_open</code>	Boolean	<code>true</code> if the access is currently in “stay open” mode (1)

(1): only when the *Zones* feature is enabled

E.3 Inventories

This table describes the data model of inventories in the `/api-impexp/inventories` web service.

Name	Type	Description
<code>id</code>	ID	Inventory public ID
<code>date</code>	Datetime	Date + time in UTC
<code>site.id</code>	ID	Site public ID
<code>site.name</code>	Text	Site name
<code>tank.id</code>	ID	Tank public ID
<code>tank.name</code>	Text	Tank name
<code>volume</code>	Decimal	Gauged volume (L or gal)
<code>temperature</code>	Decimal	Gauged temperature (°C)

E.4 Supervision actions

This section describes the detail of the requests to perform supervision actions. All supervision actions web services are POST requests. On success, the HTTP status code is 200, and the response of those actions is just `{}`.

E.4.1 POST `/api-impexp/rcm/block_pump`

Name	Type	Description
<code>pump</code>	ID	Pump public ID

E.4.2 POST `/api-impexp/rcm/unblock_pump`

Name	Type	Description
pump	ID	Pump public ID

E.4.3 POST /api-impexp/rcm/start_pump

Name	Type	Description
pump	ID	Pump public ID
vehicle	ID	Vehicle public ID
driver	ID	Driver public ID
kmeter	Integer	Odometer (km or Miles)
hmeter	Decimal	Meter in hours
activity	ID	Activity public ID
nce_code	Text	NCE code

E.4.4 POST /api-impexp/rcm/force_refresh

Name	Type	Description
pump	ID	Site public ID

E.5 Examples

E.5.1 Get the status of a site

Request:

```
GET /api-impexp/rcm/sites/bfb3505b-9426-4c21-98e0-2803802d442e
```

Response:

```
{
  "result": {
    "id": "bfb3505b-9426-4c21-98e0-2803802d442e",
    "name": "Station",
    "controllers": [
      {
```

```
    "sn": "001348000000",
    "online": false,
    "date": "2020-09-04T09:07:10.941871Z"
  }
],
"tanks": [
  {
    "id": "019b1733-078e-466d-a777-4bf598905272",
    "name": "Station T1 DIESEL",
    "product": {
      "id": "141da8b9-5755-4467-8c4b-ecadd7126c5",
      "name": "DIESEL"
    },
    "capacity": 10000,
    "volume": 4400,
    "volume_date": "2020-08-28T12:02:42Z"
  }
],
"pumps": [
  {
    "id": "b41024e5-7a6b-0b64-dde3-1f03ccb1cd5b",
    "name": "P1",
    "num": "1",
    "product": {
      "id": "141da8b9-5755-4467-8c4b-ecadd7126c5",
```

```
    "name": "DIESEL"
  },
  "tank": {
    "id": "019b1733-078e-466d-a777-4bf598905272",
    "name": "Station T1 DIESEL"
  },
  "blocked": false,
  "manual": false,
  "pumping": false,
  "blocked_reason": "",
  "transaction": null
}
]
}
}
```

E.5.2 Block a pump

Request:

```
POST /api-impexp/rcm/block_pump
{
  "pump": "b41024e5-7a6b-0b64-dde3-1f03ccb1cd5b"
}
```

Response:

```
{}
```

E.5.3 Unblock a pump

Request:

```
POST /api-impexp/rcm/unblock_pump
```

```
{  
  "pump": "b41024e5-7a6b-0b64-dde3-1f03ccb1cd5b"  
}
```

Response:

```
{}
```

E.5.4 Start a remote transaction

Request:

```
POST /api-impexp/rcm/start_pump
```

```
{  
  "pump": "b41024e5-7a6b-0b64-dde3-1f03ccb1cd5b",  
  "vehicle": "fb7e78b1-9005-49b5-96cf-a5071177505a",  
  "driver": "5d49d014-1bd7-49f8-851d-92eaa44ad112"  
}
```

Response:

```
{}
```

E.5.5 Trigger a new gauging

Request:

```
POST /api-impexp/rcm/force_refresh
```

```
{  
  "site": "bfb3505b-9426-4c21-98e0-2803802d442e"  
}
```

Response:

```
{}
```

E.5.6 Fetch the first inventories

Request:

```
GET /api-impexp/inventories
```

Response:

```
{  
  "result": [  
    {  
      "id": "22d8c15f-e4af-4a58-9178-eabafd6d1fb0",  
      "date": "2018-06-14T07:40:05Z",  
      "site": {  
        "id": "bfb3505b-9426-4c21-98e0-2803802d442e",  
        "name": "Station"  
      },  
      "tank": {  
        "id": "141da8b9-5755-4467-8c4b-ecaddd7126c5",  
        "name": "Station T1 DIESEL"  
      },  
      "volume": 20238,  
    },  
  ],  
}
```

```
    "temperature": 16.3
  },
  ...
  {
    "id": c58709c9-d4f2-49e5-ac1e-da9d35652722,
    ...
  }
],
"more": true
}
```

E.5.7 Get the next inventories (after c58709c9-d4f2-49e5-ac1e-da9d35652722)

Request:

```
GET /api-impexp/inventories?last_id=c58709c9-d4f2-49e5-ac1e-da9d35652722
```

Response:

```
{
  "result": [
    ...
  ],
  "more": true
}
```