Plugin-Daten	
Autor	Jan Wachsmuth
Logo	
Status	BETA
Version	0.7.1
Min. LB Version	2.0
Pre-Release Download	https://github.com/Jan21493/LoxBerry-Plugin-TeslaCmd/archive/refs/tags/0.7.1.zip
Beschreibung	Plugin für das von Tesla in 2024 veröffentlichte Tesla Vehicle Command SDK. Über das Plugin können Informationen zu den verwendeten Tesla Produkten abgerufen oder Befehle an die Tesla Produkte gesendet werden. Nachfolger des TeslaConnect Plugins.
Sprachen	EN
Diskussion	https://www.loxforum.com/forum/projektforen/loxberry/plugins/437633-neues-plugin-f%C3%BCr-tesla-mit-vehicle-command-api

TeslaCommand

Version history ...

Version 0.7.1:

- Added commands to retrieve status and control battery for Tesla powerwall.

Version 0.7.0:

- Settings page enhanced: now you can adjust timeouts, debugging and retries directly (custom BLE command removed).
- Debug messages for plugin are more descriptive and comprehensive.
- Process handling for generating and verifying keys is improved: you may stop the countdown of 30 seconds now if you are done by clicking the button.
- Enhancement for send.php: now VIN is used for MQTT messages if it was provided.

Version 0.6.0:

- added queuing system on Loxberry for BLE commands (only one binary is able to use BLE device at a time)
- improved performance for settings screen - RSSI is fetched on request only
- updated tesla-control binary (RSSI is added in JSON output)
- added verification, if a public key was installed properly

Version 0.5.1:

- Minor corrections in function to send keys to car
- Added verification if public key was properly installed in car
- Added/updated binary images for Tesla Vehicle Command SDK utilities for 32-bit Raspberry OS

Version 0.5.0:

- Automatically select the proper SDK/API from VIN of the vehicle (only pre 2021 S and Y models use old API).
- Create and delete a key pair via GUI for each vehicle with explanations
- Send public key to car via GUI with explanations

- Show RSSI (received signal strength) and meaning
- Show function blocks for outputs and output commands that can be copied to Loxone config
- **NOTE**: since this version, special builds of Tesla utlities that produce JSON output are required! See below for more information.
- Updated binary for tool '**tesla-contol**' that has a new category to get location information
- Added binary for new tool '**tesla-scan**' that scans for vehicles nearby without using any keys

Version 0.2.4:

- Fix for version 0.2.2 - state command was missing in json file.

Version 0.2.2:

- Updated and enhanced to reflect changes from Tesla vehicle-command SDK, PR #330 - Support vehicle data over BLE

Statement from Tesla regarding the updated SDK: adds support for fetching and decrypting vehicle data over BLE. Some types of vehicle data are not available prior to 2024.38.

You are able to retrieve detailed status information from the vehicle via BLE. The tool '**tesla-contol**' has a new command 'state <category>', with following values for category: charge-schedule, precondition-schedule, media-detail, software-update, climate, drive, tire-pressure, media, parental-controls, charge, closures

The TeslaCmd plugin supports all new commands and options for 'tesla-control' and 'tesla-keygen'. Updated binaries for 64-bit Raspberry are included.

Version 0.2.1:

- Minor bug fixes.
- Fixed problem with 'wake up' option via force=true option.

Version 0.1.9:

- Minor bug fixes.
- Added 'wake up' to commands via force=true option.

Version 0.1.8:

- Minor bug fixes.
- Updated binary for Raspberry PI 64-bit - including patches from https://github.com/teslamotors/vehicle-command/issues/272#issuecomment-2286601142

Version 0.1.7:

- Enhancements for Queries: bluetooth icon for all commands that use the new API via BLE, all parameters are explained
- Enhancement for BLE commands: vehicleNearby added, MQTT modifications, bug fixes and enhancements
- Updates binaries for Raspberry PI 64-bit - Tesla has modified source code of SDK to improve BLE reliability

Version 0.1.6:

- Bug fixing and modifications.

Version 0.1.2 to 0.1.5:

- Fixed some bugs

Version 0.1.1:

- First Beta

Warnung

Dieses Plugin speichert ein Token, um auf die Tesla API zuzugreifen. Wenn Befehle über Bluetooth Low Energy (BLE) an das Fahrzeug gesendet werden, wird außerdem ein privater Schlüssel gespeichert, über den die Befehle an das Fahrzeug signiert werden. Ich empfehle das Plugin nur zu installieren, wenn der Loxberry gut geschützt und nicht direkt aus dem Internet erreichbar ist.

Gelingt es jemand dieses Token und/oder den privaten Schlüssel auszulesen, kann dieser alle Informationen über das Fahrzeug abrufen und Befehle an das Fahrzeug senden.

Download

Direkter Download-Link: Siehe *Tabelle oben*

Letzter Entwicklungsstand im Repo: https://github.com/Jan21493/LoxBerry-Plugin-TeslaCmd

Funktion des Plugins

Dieses Plugin ist ein Nachfolger des von Marius H. erstellten TeslaConnect Plugins. Da Marius H. die Arbeit an diesem älteren Plugin eingestellt hat, habe ich die Herausforderung angenommen und Unterstützung für die neue API hinzugefügt sowie einige Verbesserungen innerhalb der Benutzeroberfläche zum Senden von Testabfragen vorgenommen. Die neue Version hat einen neuen Namen bekommen (und beginnt mit Version 0.1.1), da es nach der Erstveröffentlichung nicht mehr erlaubt ist, den Namen des Autors eines LoxBerry-Plugins zu ändern.

Tesla hat eine neue API, die [Vehicle Command API](https://github.com/teslamotors/vehicle-command/blob/main/README.md) im Oktober 2023 als Nachfolger der [(inoffiziellen) Owner's API](https://tesla-api.timdorr.com/) veröffentlicht. Fahrzeuge der Modelle S und X, die vor 2021 hergestellt wurden, unterstützen nicht das neue Protokoll, aber alle anderen Fahrzeuge werden in 2024 auf das neue Protokoll umgestellt.

Das Tesla Vehicle Command SDK enthält das [Tesla Control Utility](https://github.com/teslamotors/vehicle-command/tree/main/cmd/tesla-control#tesla-control-utility) über welches Befehle entweder über Bluetooth Low Energy (BLE) oder das Internet über die

Tesla Fleet API an das Fahrzeug gesendet werden können. Derzeit ist für das Plugin nur BLE implementiert und es gibt derzeit keine Pläne die Fleet API zu unterstützen.

Alle Befehle zur Steuerung des Fahrzeugs, z.B. Das Ändern der Klimatisierung, das Ändern der Einstellungen zum Laden des Fahrzeugs oder das Öffnen oder Schließen des Kofferraums sind über die neue Vehicle Command API sowohl über BLE, als auch über die Fleet API verfügbar. Mittlerweile unterstützt die neue API zwei Befehle zum Abrufen von Statusinformationen über BLE. Der erste Befehl „***body-controller-state***“ ruft grundlegende Informationen über Schließzustände, Schlafstatus und Benutzeranwesenheitsstatus ab und funktioniert auch, wenn das Fahrzeug schläft. Der zweite Befehl „***state***“ wird verwendet, um detaillierte Statusinformationen abzurufen.

Das Plugin unterstützt ebenfalls die alten Befehle der (inoffiziellen) Owner's API, die noch funktionieren und z.B. für S- und Y-Modelle vor 2021und Powerwalls weiterhin verwendet werden. Bei neueren Fahrzeugen können einige Statusinformationen weiterhin ebenfalls über die (inoffizielle) Owner's-API (https://owner-api.teslamotors.com/api/1/vehicles/...) abgerufen werden. Seit Version 0.5 dieses Plugins wird die richtige API automatisch ausgewählt.

Alle Daten (Statusinformationen) werden vom Fahrzeug über MQTT zurück an den Loxone Miniserver übertragen. Die Subskription dafür lautet `teslacmd/#` und wird im MQTT Gateway Plugin automatisch eingetragen. Beim ersten Öffnen des Plugins muss die Anmeldung bei Tesla durchgeführt werden. Dazu wird bei Tesla ein Token generiert. Das Token wird anschliessend automatisch verlängert. Nach erfolgreicher Anmeldung werden möglichen Abfragen aufgelistet, welche am Miniserver als Virtuelle Ausgänge definiert werden können.

Aktuell sind folgende Abfragen und Befehle möglich:

Allgemein

Befehl	Beschreibung
status	Status of the Tesla API.
product_list	Returns all products including vehicles, powerwalls, and energy sites.
vehicles	Returns a list of all vehicle registered for the authenticated user.

Abfragen für ein Fahrzeug

Alle Befehle, die das neue Vehicle Command API via BLE verwenden, sind mit dem Bluetooth Symbol gekennzeichnet.

Befehl	Type	Beschreibung
vehicle_summary		Summary information of the vehicle.
vehicle_data		All information and states of the vehicle.
list_keys		List public keys enrolled on vehicle.
charge_state		Charge state information including battery limit, charge miles, charge voltage, charge phases, current, charge management, and battery heater status.
climate_state		Climate settings including seats, vents battery, steering wheel, and preconditioning state.
closures_state		Closure state informations - may be empty
drive_state		Drive state including latitude, longitude, and heading of the vehicle.

Befehl	Type	Beschreibung
gui_settings		GUI settings of the car, such as unit format and range display
location_data		Location information including position, heading, and speed.
vehicle_config		Vehicle configuration information including model, color, badging and wheels
vehicle_state		Vehicle state, such as which doors are open, tpms pressure
body_controller_state		Fetch limited vehicle state information. Works over BLE even when infotainment / vehicle is asleep.
state		Fetch vehicle state over BLE for following categories: 'tire-pressure', 'parental-controls', 'closures', 'charge-schedule', 'drive', 'location', 'precondition-schedule', 'media', 'media-detail', 'software-update', 'charge', or 'climate'.

Befehle für ein Fahrzeug für die alte Owner's API

Die nachfolgenden Befehle sind über die alte Owner's API über das Internet verfügbar. Neuere Fahrzeuge unterstützen diese API nicht mehr, sondern aus Sicherheitsgründen für Befehle an das Fahrzeug nur noch die neue Vehicle Command API.

Befehl	Beschreibung
wake_up	Wakes up the vehicle from a sleeping state.
sun_roof_control	Controls the panoramic sunroof on the Model S.
window_control	Controls the windows. Will vent or close all windows simultaneously.
actuate_trunk	Opens or close either the front or rear trunk.
honk_horn	Honks the horn of the vehicle once.
flash_lights	Flashes the headlights once.
auto_conditioning_start	Start the climate control (HVAC) system. Will cool or heat automatically, depending on set temperature.
auto_conditioning_stop	Stop the climate control (HVAC) system.
set_temps	Sets the target temperature for the climate control (HVAC) system. (e.g. value 20)
set_preconditioning_max	Toggles the climate controls between Max Defrost and the previous setting.
remote_seat_heater_request	Sets the specified seat's heater level.
remote_steering_wheel_heater_request	Turn steering wheel heater on or off.
charge_port_door_open	Opens the charge port or unlocks the cable.
charge_port_door_close	Closes the charge port.
charge_start	If the car is plugged in but not currently charging, this will start it charging.
charge_stop	If the car is currently charging, this will stop it.
charge_standard	Set vehicle to standard charge limit or ~90%.
charge_max_range	Set the vehicle to max charge limit.
set_charge_limit	Set the vehicle to defined charge limit. (e.g. value 80)
set_charging_amps	Sets the charge amps limit to a custom value. (e.g. value 16)
set_scheduled_departure	Set the scheduled departure.
set_scheduled_charging	Set the scheduled charge.

Befehl	Beschreibung
set_valet_mode	Activates or deactivates Valet Mode.
reset_valet_pi	Clears the currently set PIN for Valet Mode when deactivated. A new PIN will be required when activating from the car screen.
set_sentry_mode	Turns sentry mode on or off.

Neue oder angepasste Befehle für die Vehicle Control API via BLE

Die nachfolgenden Befehle sind in der neuen Vehicle Command API enthalten. Das Plugin unterstützt derzeit nur das Senden über BLE.

Befehl	Beschreibung
ble_wake	Wakes up the vehicle from a sleeping state via Bluetooth Low Energy (BLE).
ping	Ping the vehicle.
remote_start_drive	Enables keyless driving. There is a two minute window after issuing the command to start driving the car.
door_unlock	Unlocks the doors to the vehicle.
door_lock	Locks the doors to the vehicle.
autosecure_modelx	Close falcon-wing doors and lock vehicle. Model X only.
windows_close	Closes all windows simultaneously.
windows_vent	Vent all windows simultaneously.
trunk_close	Closes vehicle trunk. Only available on certain vehicle types.
trunk_open	Open vehicle trunk. Note that trunk-close only works on certain vehicle types.
trunk_move	Toggle trunk open/closed. Closing is only available on certain vehicle types.
frunk_open	Open vehicle frunk. Note that there's no frunk-close command!
honk_horn	Honks the horn of the vehicle once.
flash_lights	Flashes the headlights once.
auto_conditioning_start	Start the climate control (HVAC) system. Will cool or heat automatically, depending on set temperature.
auto_conditioning_stop	Stop the climate control (HVAC) system.
auto_seat_and_climate	Turn on automatic seat heating and climate control (HVAC).
climate_set_temp	Desired temperature for climate control (HVAC) system.
seat_heater	Sets the specified seat's heater level.
steering_wheel_heater	Turn steering wheel heater on or off.
charge_port_door_open	Opens the charge port or unlocks the cable.
charge_port_door_close	Closes the charge port.
charge_start	If the car is plugged in but not currently charging, this will start it charging.
charge_stop	If the car is currently charging, this will stop it.
set_charge_limit	Set the vehicle to defined charge limit. (e.g. value 80)
set_charging_amps	Sets the charge amps limit to a custom value. (e.g. value 16)
charging_schedule	Schedule charging to MINS minutes after midnight and enable daily scheduling.
charging_schedule_cancel	Cancel scheduled charge start.
charging_schedule_add	Schedule charge for DAYS START_TIME-END_TIME at LATITUDE LONGITUDE for [ID].
charging_schedule_remove	Removes charging schedule of TYPE [ID].
precondition_schedule_add	Schedule precondition for DAYS TIME at LATITUDE LONGITUDE.
precondition_schedule_remove	Removes precondition schedule of TYPE [ID]
valet_mode_off	Disable valet mode.
valet_mode_on	Enable valet mode and set pin.
sentry_mode	Turns sentry mode on or off.
tonneau_close	Close Cybertruck tonneau.
tonneau_open	Open Cybertruck tonneau.
tonneau_stop	Stop moving Cybertruck tonneau.

Befehl	Beschreibung
media_set_volume	Set volume.
media_toggle_playback	Toggle between play/pause.
software_update_start	Start software update after DELAY.
software_update_cancel	Cancel a pending software update.
guest_mode_off	Disable Guest Mode.
guest_mode_on	Enable Guest Mode. This mode is used by e.g. rental car companies to restricts certain vehicle UI functionality like renaming the car or deleting keys. See https://developer.tesla.com/docs/fleet-api/endpoints/vehicle-commands#guest-mode.
erase_guest_data	Erase Guest Mode user data.
add_key	Add PUBLIC_KEY to vehicle whitelist with ROLE and FORM_FACTOR.
add_key_request	Request NFC-card approval for a enrolling PUBLIC_KEY with ROLE and FORM_FACTOR.

Abfragen für Powerwall / Energy Site

Befehl	Beschreibung
LIVE_STATUS	Retrieves current system information (e.g. solar production, grid export/import, home consumption, etc.).
SITE_INFO	Retrieves general system information.
SITE_STATUS	Retrieves general system information.
SITE_PROGRAMS	Retrieves energy site program information.
SITE_TARIFF_RATE	Retrieves the user defined Utility Rate Plan used for Time-Based Control mode. It looks like this endpoint is updated every 30 minutes.
BACKUP_HISTORY	Returns the backup (off-grid) event history of the site in duration of seconds.
CHARGE_HISTORY	Returns the charging history of a wall connector.
ENERGY_HISTORY	Returns the energy measurements of the site, aggregated to the requested period.

Befehle für Powerwall / Energy Site

Befehl	Beschreibung
BACKUP	Adjust the site's backup reserve.
GRID_IMPORT_EXPORT	Allow/disallow charging from the grid and exporting energy to the grid.
OFF_GRID_VEHICLE_CHARGING_RESERVE	Adjust the site's off-grid vehicle charging backup reserve.
OPERATION	Use autonomous for time-based control and self_consumption for self-powered mode.
STORM_MODE	Update storm watch participation.
TIME_OF_USE_SETTINGS	Update the time of use settings for the energy site.

Voraussetzungen

Damit der Loxberry Befehle via BLE (Bluetooth Low Energy) an ein Fahrzeug senden kann, muss er sich in der Nähe des Fahrzeugs befinden. Eine maximal mögliche Entfernung kann ich nicht nennen, aber bei mir funktioniert es mit der (nicht besonders guten) kombinierten WLAN/BLE Antenne, die sich auf der Platine des Raspberry PI 3B+ befindet, durch eine Außenwand des Gebäudes (ohne Fenster auf der Seite zum Fahrzeug) über eine Distanz von ca. 7m ziemlich zuverlässig. Wer seinen Loxberry im Verteilerschrank aus Metall eingebaut hat, der sollte die Antenne auf jeden Fall nach außen verlegen, wofür die Platine bearbeitet werden muss. Anleitungen für so eine Modifikation gibt es sehr viele im Internet, u.a. hier. Falls der Raspberry in ein Gehäuse aus Metall eingebaut wurde, ist eine

externe Antenne ebenfalls dringend empfohlen. Man kann das Plugin sicherlich auch auf einem eigenen Loxberry installieren, der an einem sicheren Ort in der Nähe des Fahrzeugs installiert wird. Im SDK von Tesla werden Linux und MacOS als mögliche Betriebssysteme bei Verwendung von Bluetooth genannt. Windows wird offiziell nicht unterstützt.

Golang Bluetooth Library

Leider wird die Golang Bluetooth Library ***go-ble*** derzeit nicht gepflegt. Auf einem Raspberry PI kommt es bei Ausführung des Tools '*tesla-control*' aus dem SDK, welches das Plugin verwendet, zu einem Problem, wenn z.B. das Fahrzeug nicht in der Nähe über BLE erreichbar ist. Das Tool kann dann in einer Endlosschleife festhängen und gibt die verwendeten Bluetooth Ressourcen nicht mehr frei. Ein weiteres Ausführungen des Tools wird mit einer Fehlermeldung abgebrochen. Dieses Problem ist von einem Anwender erkannt worden und in einem Pull Request (PR) bereits ein Code vorbereitet worden, der den Fehler beseitigt, aber der PR wird nicht von den Maintainern des Git Repositories bestätigt. Dadurch ist es deutlich umständlicher, aus den 'offiziellen' Quellen eine Version des Tools ohne den Fehler zu erstellen.

Da die Meldungen im Debug-Modus im o.a. Fehlerbild zu mehreren 1000 Zeilen Output führen, habe ich die Tools aus dem Tesla Vehicle Command SDK auf eine andere Golang Bluetooth Library von Rigado umgestellt, die deutlich bessere Meldungen erzeugt. Der o.a. Fehler ist dort leider ebenfalls noch vorhanden, siehe Pull Request #76. Daher muss eine angepasste Version des Tesla Vehicle Command SDK verwendet werden, die eine angepasste Version der BLE Library inklusive Fix verwendet.

Installation

Für das Plugin empfehle ich eine frische Installation eines Loxberries ab Version 3.0, wo das MQTT Gateway bereits enthalten ist. Wenn ein Raspberry 3, 4 oder 5 verwendet wird, empfiehlt sich die Installation mit OS Bookworm 64-Bit auf Basis eines DietPi.

1. Aktivierung von Bluetooth, sofern erforderlich. **Bei einem DietPi ist Bluetooth per Default ausgeschaltet** und wird mit Hilfe des Tools '**//dietpi-config//**' im Menü 'advanced options' eingeschaltet.
2. Installation des Plugin MQTT Gateway. Sollte dieses bereits auf dem Loxberry vorhanden sein, kann dieser Punkt übersprungen werden. Ab LoxBerry 3.0 ist das MQTT Gateway ein Bestandteil des LoxBerrys und daher bereits automatisch installiert.
3. Installation des Tesla-Command Plugins über die Plugin Verwaltung.
4. Installation bzw. Erstellung der beiden Kommandozeilentools aus dem Tesla Vehicle Command SDK, siehe zugehörige README.md auf Github oder nachfolgende Hinweise.
5. Erstellung eines privaten und öffentlichen Schlüssels bzw. Schlüsselpaares, Installation des öffentlichen Schlüssels im Fahrzeug, siehe zugehöriges README.md vom Tesla Vehicle Command SDK auf Github oder nachfolgende Hinweise.

Installation der Tools aus dem Tesla Vehicle Command SDK

Dieser Schritt kann bei einem Loxberry auf Basis eines Raspberry PI mit **64-Bit** (z.B. bei einem Diet-PI

mit Version Bookworm) übersprungen werden, weil die beiden benötigten Tools bereits im Binärformat im Plugin enthalten sind. Man benötigt daher weder Golang noch das SDK. Für alle anderen Betriebssysteme und CPUs ist dieser Schritt erforderlich. Die Installation der Tools ist in der README.md des SDKs beschrieben und hier nur ergänzt. Windows wird offiziell nicht unterstützt. Die Installation hat auf einem MacBook mit Sonoma und einem Raspberry PI mit 32 als auch 64-Bit funktioniert. Wenn man die Tools lieber selber erstellen möchte oder Tesla das SDK aktualisiert hat, dann kann man alternativ die nachfolgenden Schritte auch auf einem Raspberry PI ausführen.

Installation von Golang

Die aktuelle Version von Golang findet man unter https://go.dev/doc/install Für einen Raspberry PI mit 32-Bit sind ergänzend die folgenden Befehle exemplarisch für Version 1.23.2 dargestellt. Für einen Raspberry PI mit 64-Bit muss das Paket ***go1.23.2.linux-arm64.tar.gz*** verwendet werden und die beiden Befehle sind entsprechend anzupassen. Für andere Platformen muss das entsprechende Go Paket verwendet werden. Die Version sollte durch die jeweils aktuelle Version ersetzt werden!

```
mkdir ~/golang
cd ~/golang
# on 32-Bit Linux ARM
wget https://go.dev/dl/go1.23.2.linux-armv6l.tar.gz
tar -xzf go1.23.2.linux-armv6l.tar.gz
# on 64-Bit Linux ARM
wget https://go.dev/dl/go1.23.2.linux-arm64.tar.gz
tar -xzf go1.23.2.linux-arm64.tar.gz
#
export PATH=$PATH:/opt/loxberry/golang/go/bin
# Verify that Go is working
go version
```

Installation des SDKs über Git

Die Installation des ***Tesla Vehicle Command SDK*** kann über ***Git*** erfolgen. Alternativ kann das SDK auch im ***.zip*** Format heruntergeladen und ausgepackt werden. Der entscheidende Schritt ist der Befehl 'build' mit dem die beiden benötigten Tools ***tesla-keygen*** und ***tesla-control*** erstellt werden. Diese Befehle sollten als User 'loxberry' ausgeführt werden.

Eine modifizierte Version der Golang Bluetooth Library von Rigado ist hier zu finden. Eine angepasste Version des Tesla Command SDKs für diese Golang BLE Library, die zusätzlich Versionsinformationen anzeigt und Ausgaben kompakt im JSON Format liefert, ist dort ebenfalls zu finden. Für das Erstellen der Tools mit Go sind mehrere Umgebungsvariablen zu setzen:

```
# install GIT as superuser - only if required
# su
# apt-get install git
# exit
cd ~
# git clone https://github.com/teslamotors/vehicle-command.git
git clone https://github.com/jan21493/vehicle-command.git
```

```
# git clone https://github.com/rigado/ble.git
git clone --single-branch --branch local-enhancements
https://github.com/Jan21493/ble
cd vehicle-command/
# if not set yet: export PATH=$PATH:/opt/loxberry/golang/go/bin
go get ./...

# set environment variables that are used for version information
HWINFO=$(cat /sys/firmware/devicetree/base/model|xargs --null printf "%s")
HWARCH=$(uname -m)
VCVERSION=$(git rev-parse --short HEAD)
TODAY=$(date "+%a, %d %b %Y %T")
echo "Hardware info: "$HWINFO
echo "Hardware architecure: "$HWARCH
echo "Vehicle-Command SDK version: "$VCVERSION
echo "Date: "$TODAY

# build tesla-control utility - may take a while
cd cmd/tesla-control
go build -ldflags "-X 'main.version=$VCVERSION' -X 'main.hwinfo=$HWINFO' -X
'main.hwarch=$HWARCH' -X 'main.today=$TODAY'"  ./...

# build tesla-scan utility - may take a while
cd ../tesla-scan
go build -ldflags "-X 'main.version=$VCVERSION' -X 'main.hwinfo=$HWINFO' -X
'main.hwarch=$HWARCH' -X 'main.today=$TODAY'"  ./...

# build tesla-keygen utility - may take a while
cd ../tesla-keygen
go build ./...
cd ..
```

Anschließend werden die Tools in das Verzeichnis ***/usr/local/bin*** kopiert, welches für zusätzliche Befehle bereits vorgesehen und im Suchpfad eingetragen ist. Da der Befehl ***sudo*** auf dem Loxberry aus Sicherheitsgründen eingeschränkt ist, muss man sich mit dem Befehl ***su*** als Superuser anmelden, um die entsprechenden Tools in die Systemverzeichnisse kopieren zu können. Der Zugriff auf Bluetooth erfordert unter Linux ein entsprechendes Recht, welches man den Tools tesla-control und tesla-scan explizit geben muss, damit es von einem 'normalen' User ausgeführt werden kann. Dafür ist ebenfalls eine Anmeldung als Superuser erforderlich.

```
# als Superuser ausführen
su
cd /opt/loxberry
mv ./vehicle-command/cmd/tesla-control/tesla-control /usr/local/bin/
setcap 'cap_net_admin=eip' /usr/local/bin/tesla-control
mv ./vehicle-command/cmd/tesla-keygen/tesla-keygen /usr/local/bin/
mv ./vehicle-command/cmd/tesla-scan/tesla-scan /usr/local/bin/
setcap 'cap_net_admin=eip' /usr/local/bin/tesla-scan
exit
```

Als Ergebnis der oben dargestellen Schritte sind jetzt die Tools aus dem Tesla Vehicle Command SDK

auf dem Loxberry installiert, die vom User Loxberry ausgeführt werden können. Zur ersten Überprüfung kann man die Tools mit der Hilfe-Funktion aufrufen.

```
tesla-control -h
tesla-keygen -h
tesla-scan -h
```

Erstellung eines Schlüsselpaares

Um Befehle sicher via BLE an das Fahrzeug senden zu können, wird ein Schlüsselpaar benötigt, welches aus einem öffentlichen und einem privaten Schlüssel besteht. Der private Schlüssel wird zum Signieren der Befehle verwendet die an das Fahrzeug gesendet werden. Der öffentliche Schlüssel wird im Fahrzeug zur Liste der autorisierten Schlüssel hinzugefügt, damit die empfangenen Befehle auf Echtheit überprüft werden können. Dieses Verfahren im Gegensatz zur alten Owner's API den großen Vorteil, dass Schlüssel individuell zurückgezogen werden können, ohne das der Zugriff für andere Geräte oder Benutzer unterbrochen wird. Ein weiterer Vorteil ist der, dass die Schlüssel nur lokal im Loxberry (der private Schlüssel) und im Fahrzeug (der öffentliche Schlüssel) gespeichert werden und nicht mal in der Cloud von Tesla hinterlegt oder durchgeleitet werden müssen.

Zunächst wird dafür ein Schlüsselpaar erstellt. Damit im Plugin mehrere Fahrzeuge unterschieden werden können, muss das Schlüsselpaar die VIN des Fahrzeugs im Namen enthalten. Diese wird z.B. im 'Settings Menu' für jedes Fahrzeug angezeigt.

Mittlerweile kann das Schlüsselpaar über die GUI im Menü 'Settings' erstellt werden, indem man auf das Plus-Symbol klickt.

Um manuell ein Schlüsselpaar zu erstellen, ersetzt man die VIN durch die VIN des eigenen Fahrzeugs und führt den nachfolgend angegebenen Befehl aus. Der öffentliche Schlüssel wird nach der Installation im Fahrzeug nicht mehr benötigt und kann dann gelöscht werden.

```
cd ~/config/plugins/teslacmd
tesla-keygen -f -key-file LRW31234567890123-private.pem create >
LRW31234567890123-public.pem
```

Der gleiche o.a. Befehl mit 'create' zeigt den öffentlichen Schlüssel an, sofern der private Schlüssel bereits vorhanden ist.

Die Schlüssel, die mit Hilfe des Tools ***tesla-keygen*** erstellt wurden, können auch mit OpenSSL Tools verarbeitet werden, um sich z.B. den öffentlichen Schlüssel anzeigen zu lassen `openssl ec -in LRW31234567890123-private.pem -pubout` bzw. diesen zu speichern `openssl ec -in LRW31234567890123-private.pem -pubout -out LRW31234567890123-public.pem`.

Installation des öffentlichen Schlüssels im Fahrzeug

Erst mit der Installation des öffentlichen Schlüssels eines Schlüsselpaares im Fahrzeug wird der zugehörige private Schlüssel autorisiert und damit schützenswert. Um den Schlüssel zu installieren, empfiehlt es sich ein Laptop zu verwenden und mit diesem eine ssh Verbindung zum Loxberry aufzubauen. So kann man einfach und relativ schnell die notwendigen Schritte im Fahrzeug ausführen, um den öffentlichen Schlüssel zu installieren.

Senden des Befehls an das Fahrzeug

Im ersten Schritt wird der Befehl zum Installieren des öffentlichen Schlüssels an das Fahrzeug gesendet. Dieser Schritt ist mittlerweile ebenfalls in der GUI implementiert und kann über das blaue Auto-Symbol im Menü 'Settings' ausgewählt werden.

Der Befehl ***tesla-control add-key-request*** erfordert dabei die Angabe der VIN, des öffentlichen Schlüssels, der Rolle (entweder ***owner*** oder ***driver***) und eines Form-Faktors (entweder '**//nfc_card//**', '**//ios_device//**', '**//android_device//**' oder '**//cloud_key//**'). Nachfolgend ist ein Beispiel angegeben.

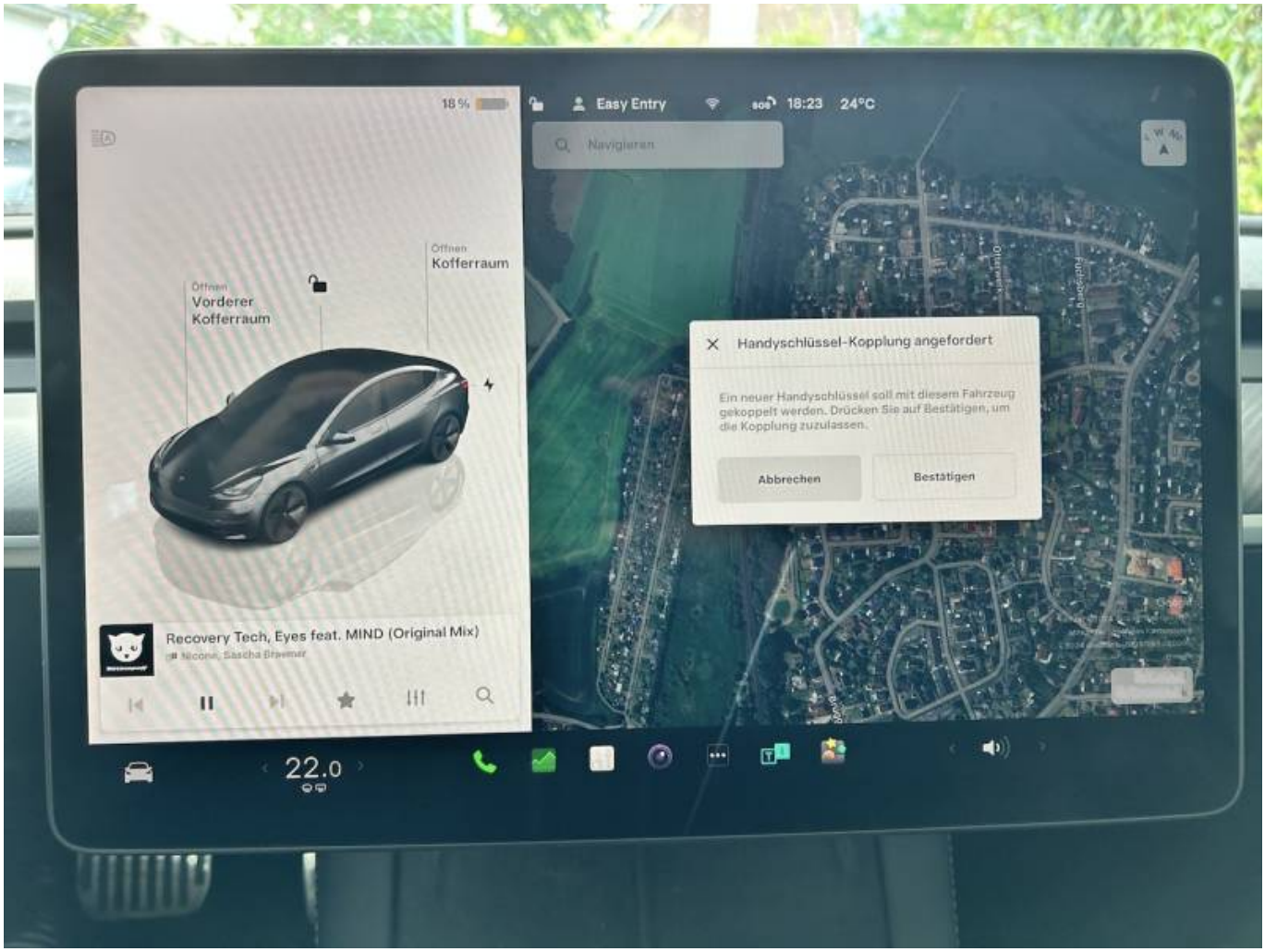
```
cd ~/config/plugins/teslacmd/
tesla-control -vin LRW31234567890123 -ble add-key-request
./LRW31234567890123-public.pem owner cloud_key
Sent add-key request to LRW31234567890123. Confirm by tapping NFC card on
center console.
```

Nur wenn die Ausgabe *`Sent add-key request to LRW31234567890123. Confirm by tapping NFC card on center console.`* nach einigen Sekunden angezeigt wird, war der Befehl erfolgreich. Eine Meldung wird auf dem Bildschirm im Fahrzeug zunächst **nicht** angezeigt.

Man muss beim Absenden des Befehls nicht im Fahrzeug sein. Bei einigen Modellen kann es erforderlich sein, dass sich das Fahrzeug nicht im Schlafmodus befindet.

Autorisierung des neuen Schlüssels

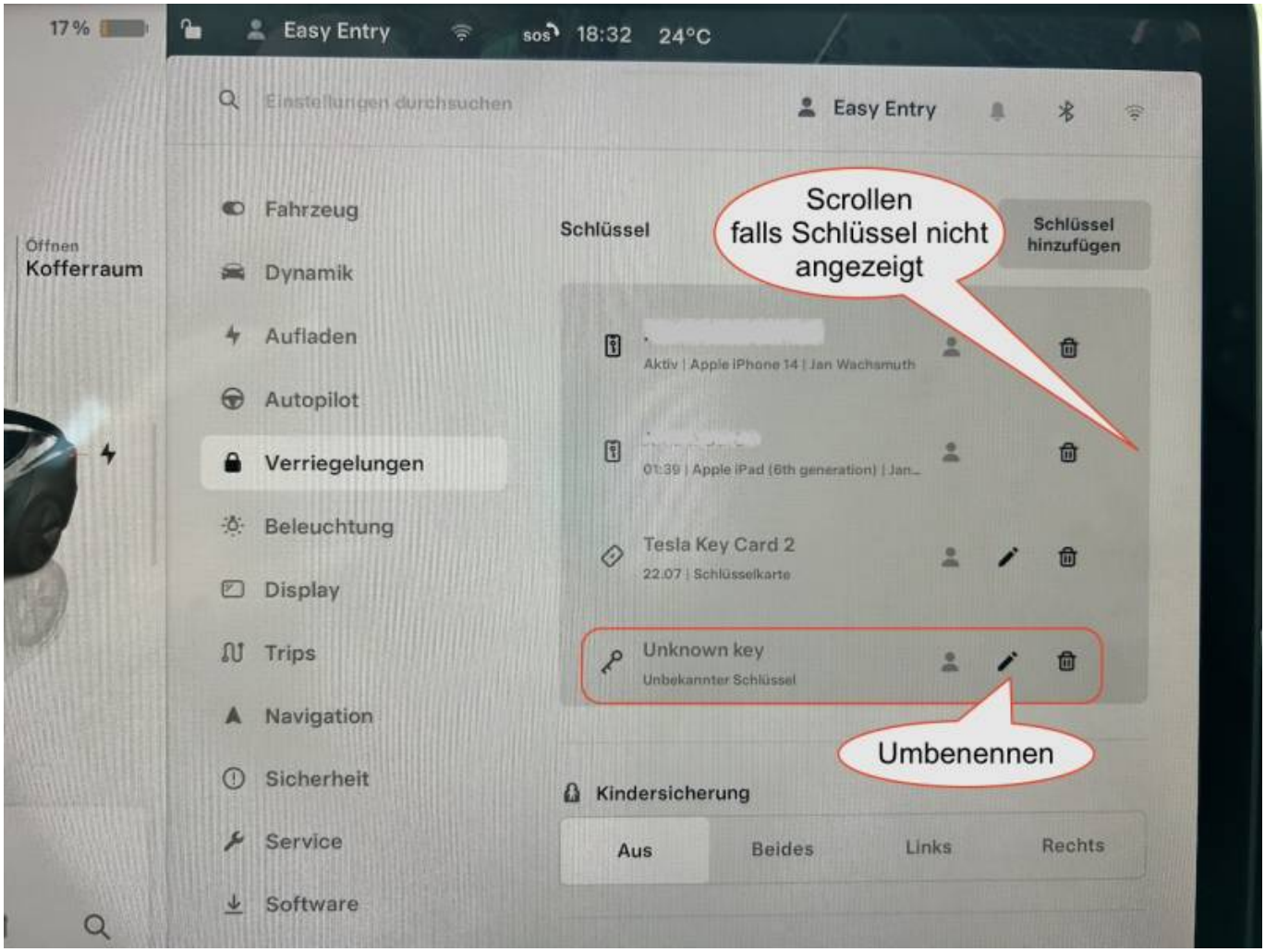
Der neue Schlüssel wird autorisiert, indem eine der beiden schwarzen NFC Karten, die mit dem Fahrzeug ausgeliefert wurden, auf die Mittelkonsole zwischen Getränkehalter und Armlehne gelegt wird. Dieser Schritt muss **innerhalb von 30 Sekunden** nach dem o.a. Befehl erfolgen! Erst danach wird eine Meldung angezeigt, die innerhalb von 10 Sekunden bestätigt werden muss.



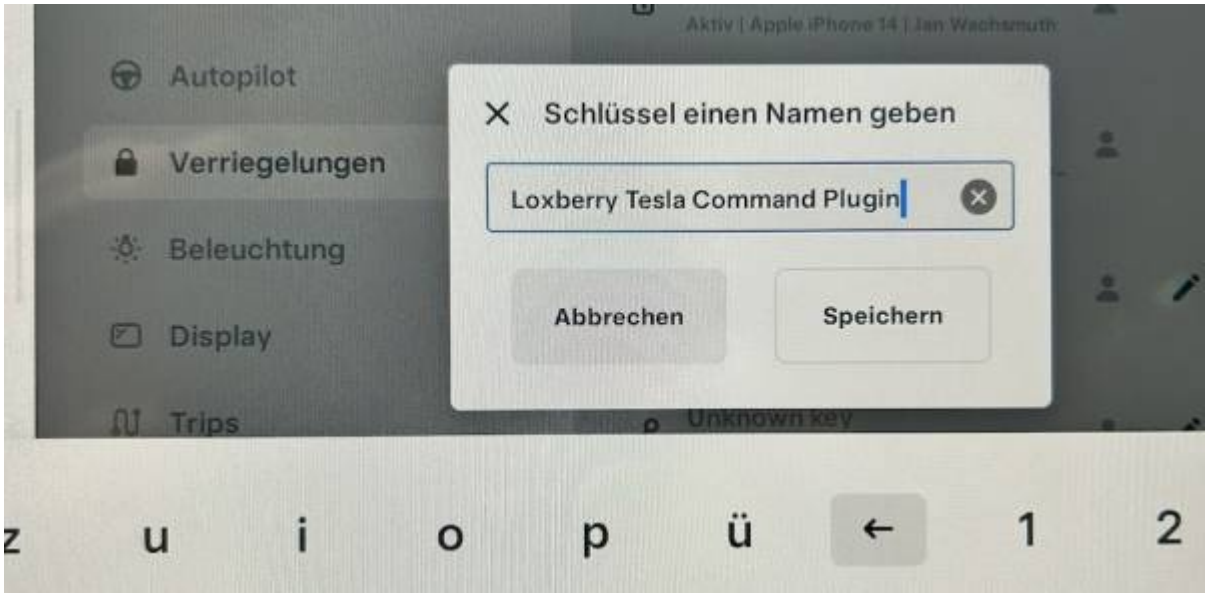
Der Text der Meldung "***Handy****schlüssel*-Koppelung" ist nicht ganz korrekt, aber wahrscheinlich hat Tesla sich nicht die Mühe gemacht, eine eigene Meldung für API Schlüssel zu erstellen.

Nach der Bestätigung wird der neue Schlüssel in der Liste der autorisierten Schlüssel angezeigt. Das entsprechende Menü heisst ***Verriegelungen*** und ist durch ein Schloss aus Symbol gekennzeichnet. Falls der Schlüssel nicht sichtbar ist, kann man auch nach unten scrollen, was grafisch nicht angezeigt

durch einen Balken wird.



Der neue Schlüssel sollte passend zum Verwendungszweck umbenannt werden. Dafür drückt man auf das Bleistiftsymbol des entsprechenden Schlüssels. Der Typ bleibt "*Unbekannter Schlüssel*" und kann leider nicht geändert werden.

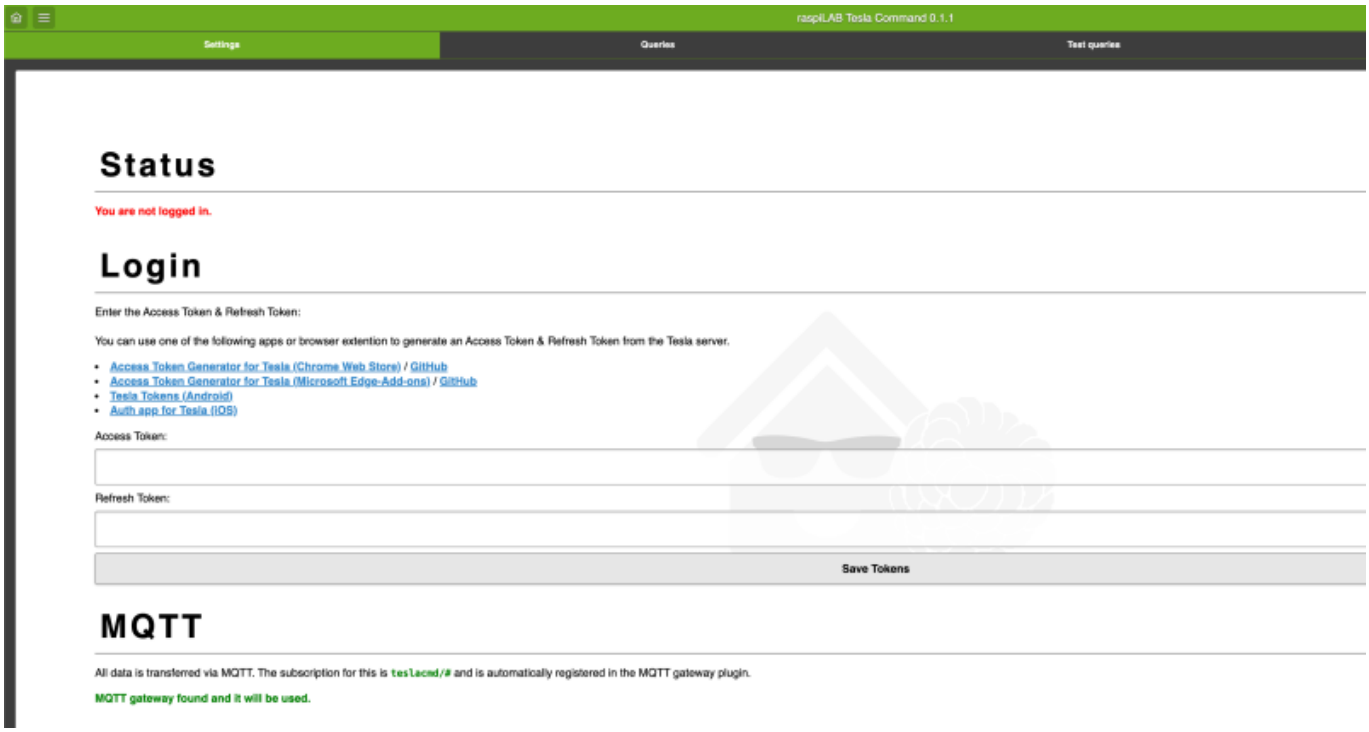


Zur Sicherheit wird in der Meldungszentrale auf dem Handy und der Tesla App eine Meldung angezeigt, dass ein neuer Schlüssel hinzugefügt wurde. Über den erwähnten Menüpunkt

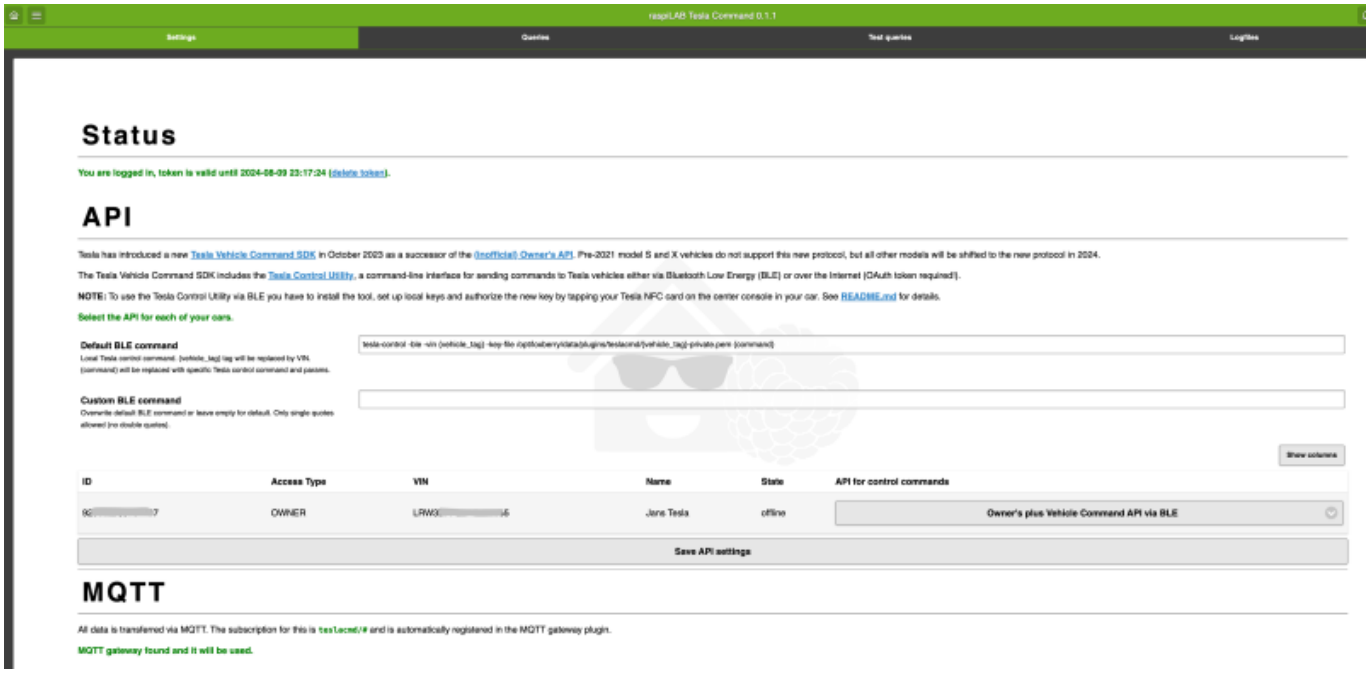
Verriegelungen können Schlüssel nach Bedarf auch wieder gelöscht werden.

Konfigurationsoptionen

Sofern das Plugin keinen gültigen Access- und Renewal-Token hat, müssen zuerst diesen Tokens zuerst im Menü ***Settings*** eingegeben werden.



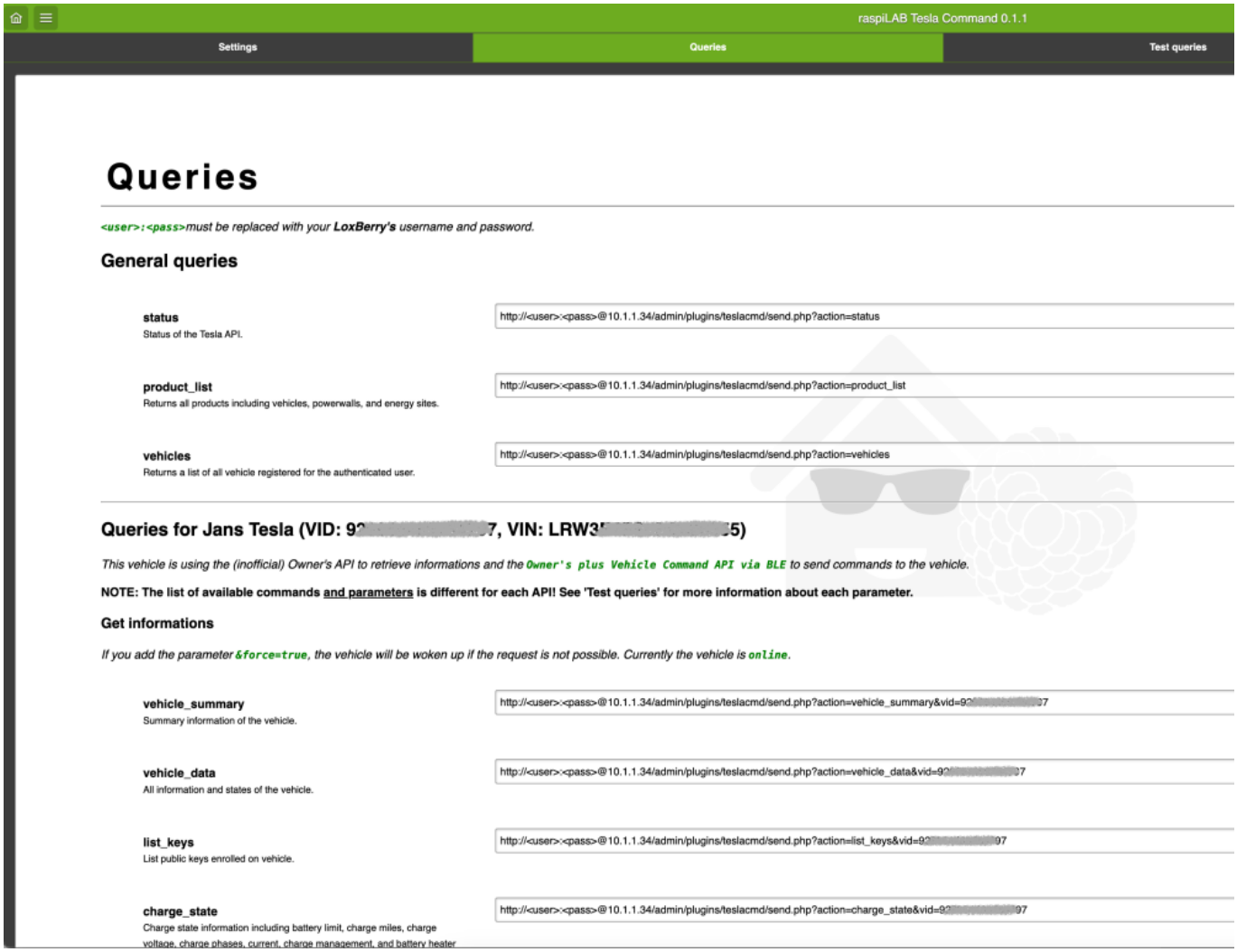
Außerdem kann man für jedes Fahrzeug bzw. Powerwall festlegen, welche API verwendet werden soll. Wichtig ist das Speichern der Einstellungen über den Button ***Save API settings***. Bei Bedarf kann optional der Befehl zum Senden von Befehlen über BLE angepasst werden.



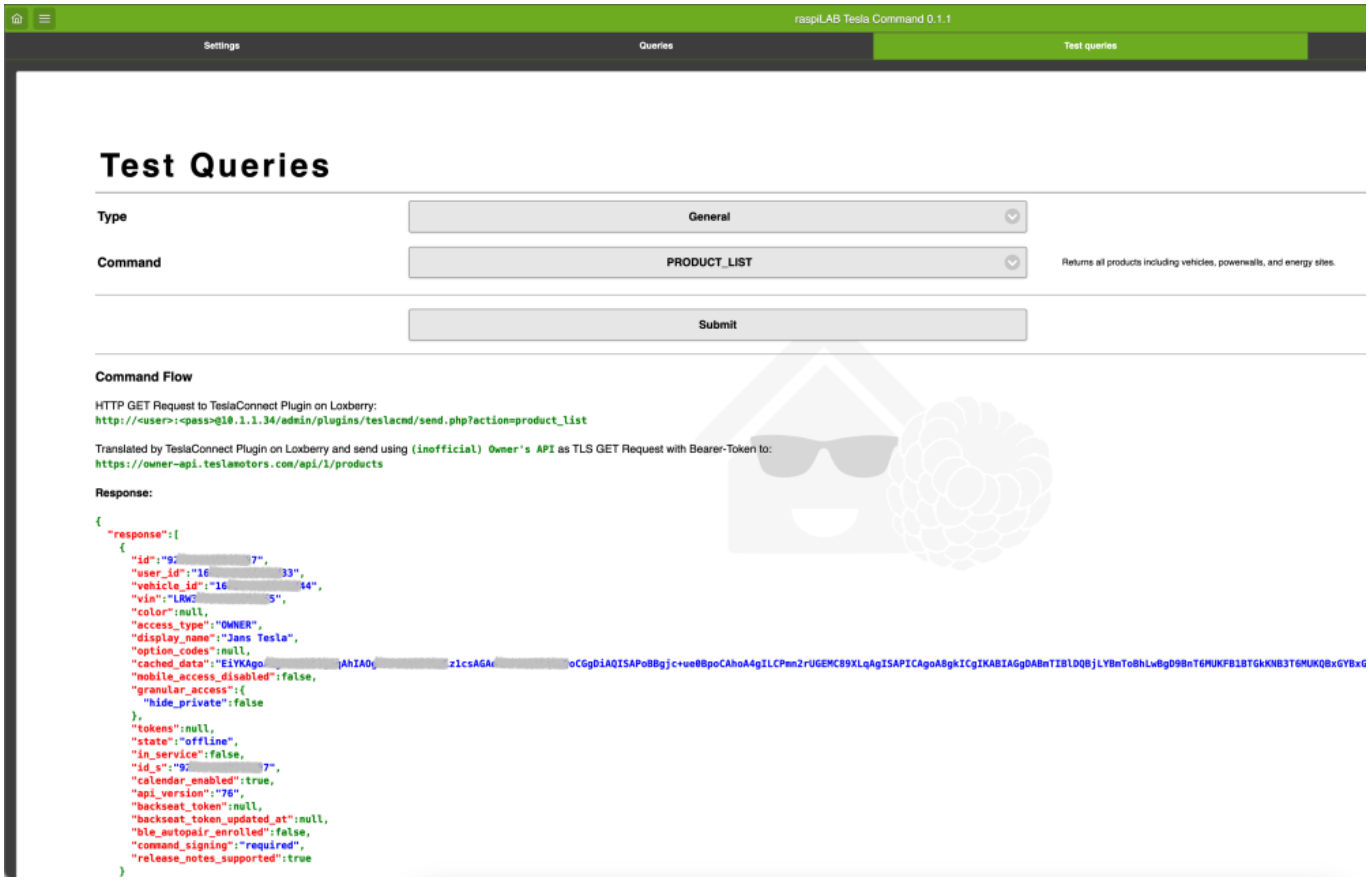
Einrichtung in der Loxone Config Software

Im Menü ***Queries*** werden alle verwendenden URLs dargestellt. Sämtliche Funktionen müssen vom Loxone Miniserver aus getriggert werden. Hierfür werden **Virtuelle Ausgänge** benötigt. Die Antwort eines Befehls, z.B. um Statusinformation abzufragen, wird immer per MQTT übertragen. Die einzelnen Variablen werden im MQTT Gateway Plugin unter dem Menü ***Incoming Overview*** unter ***HTTP Virtual Inputs*** angezeigt. Bitte folge der Anleitung des MQTT Gateway Plugins, wie du diese Daten in den Loxone Miniserver bekommst.

Die Liste der Befehle enthält 3 allgemeine Befehle und weitere spezifische Befehle für jedes Fahrzeug bzw. Tesla Powerwall:



Einzelne Befehle können im Menü ***Test queries*** ausgetestet werden, d.h. man wählt den gewünschten Befehl aus der LIste aus und kann sich nach Drücken auf den Submit Button wird das Ergebnis des Befehls angezeigt.

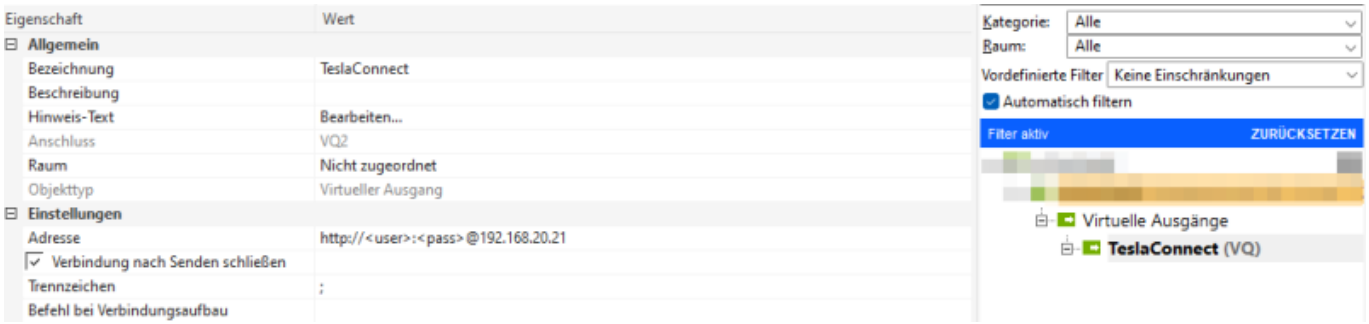



```
{
  "response":[
    {
      "id":"92[redacted]7",
      "user_id":"16[redacted]33",
      "vehicle_id":"16[redacted]14",
      "vin":"LRW2[redacted]5",
      "color":null,
      "access_type":"OWNER",
      "display_name":"Jans Tesla",
      "option_codes":null,
      "cached_data":"EiYKAgo[redacted]jAhIAO[redacted]z1csAGA[redacted]oCGgDiAQISAPoBBgjc+ue0BpoCAhoA4gILCPmn2rUGEMC89XLqAgISAPICAgoA8gkICgIKABIAGgDABnTIBlDQBjLYBnToBhLwBgD9BnT6MUKFB1BTGkKNB3T6MUKQBxGYBxG
      "mobile_access_disabled":false,
      "granular_access":{
        "hide_private":false
      },
      "tokens":null,
      "state":"offline",
      "in_service":false,
      "id_s":"92[redacted]7",
      "calendar_enabled":true,
      "api_version":"76",
      "backseat_token":null,
      "backseat_token_updated_at":null,
      "ble_autopair_enrolled":false,
      "command_signing":"required",
      "release_notes_supported":true
    }
```


Die Befehle müssen in der Loxone Config als Virtuelle Ausgänge definiert werden. Wenn diese Ausgänge getriggert werden, führt das Plugin die gewünschte Funktion aus. Der Aufruf der virtuellen Ausgänge kann über die Standard Bausteine von Loxone erfolgen.

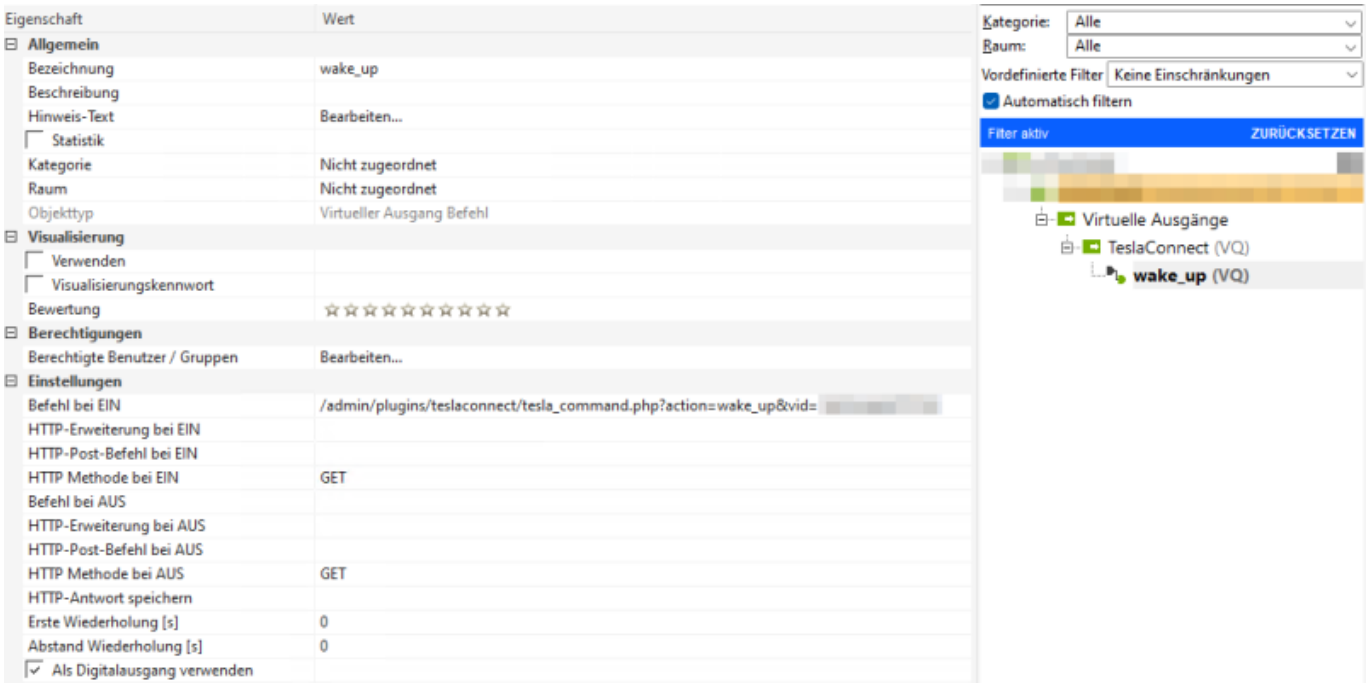
Erstelle zuerst einen Virtuellen Ausgang:

```
http://<user>:<pass>@10.1.1.34
```



Als nächstes erstelle pro Befehl ein Virtueller Ausgang Befehl. Exemplarisch ist nachfolgend der Befehl zum Abschließen aller Türen aufgeführt:

```
/admin/plugins/teslacmd/send.php?action=door_lock&vid=<vehicle-id>
```



Zum Abrufen der gesamten Fahrzeugdaten kann folgender Befehl erstellt werden.

```
/admin/plugins/teslacmd/send.php?action=vehicle_data&vid=<vehicle-id>
```

Die Daten werden nur abgerufen, wenn sich das Fahrzeug nicht im Schlafmodus befindet. Soll das Fahrzeug auch abgefragt werden, wenn es schläft, kann der Parameter ***force=true*** hinzugefügt werden. Dann wird das Fahrzeug geweckt und anschliessend die Abfrage gestartet.

```
/admin/plugins/teslacmd/send.php?action=vehicle_data&vid=<vehicle-id>&force=true
```

In der folgenden Tabelle werden Parameter aufgeführt, welche bei jeder Abfrage übertragen werden.

Parameter	**Beschreibung**	**Beispieldaten**
teslacmd_mqtt_timestamp	Zeitstempel der letzten Übertragung über MQTT in Loxonezeit	422377202
teslacmd_token_expires	Zeitstempel, wann das Token abläuft in Loxonezeit	422398802
teslacmd_token_valid	Ausgabe, ob das aktuelle Teslatoken noch gültig ist 1=gültig, 0=ungültig	1

Roadmap

Dieses Plugin ist im Entwicklungsstand. Ich verwende das Plugin derzeit, um zu prüfen, ob das Fahrzeug in der Nähe (i.d.R. im Carport) ist und zum Senden einiger Befehle an das Auto, wie z.B. Ladeklappe öffnen, Vorklimatisierung. Ich freue mich, falls jemand Ideen hat und das Projekt weiter mit entwickeln möchte.

Mögliche weiter Entwicklungsschritte:

- Bisher keine Feature Requests gestellt.

Fragen stellen und Fehler melden

Im
Loxforum: https://www.loxforum.com/forum/projektforen/loxberry/plugins/437633-neues-plugin-f%C3%BCr-tesla-mit-vehicle-command-api

From:
https://wiki.loxberry.de/ - **LoxBerry Wiki - BEYOND THE LIMITS**

Permanent link:
https://wiki.loxberry.de/plugins/teslacmd/start?rev=1738788257

Last update: **2025/02/05 21:44**