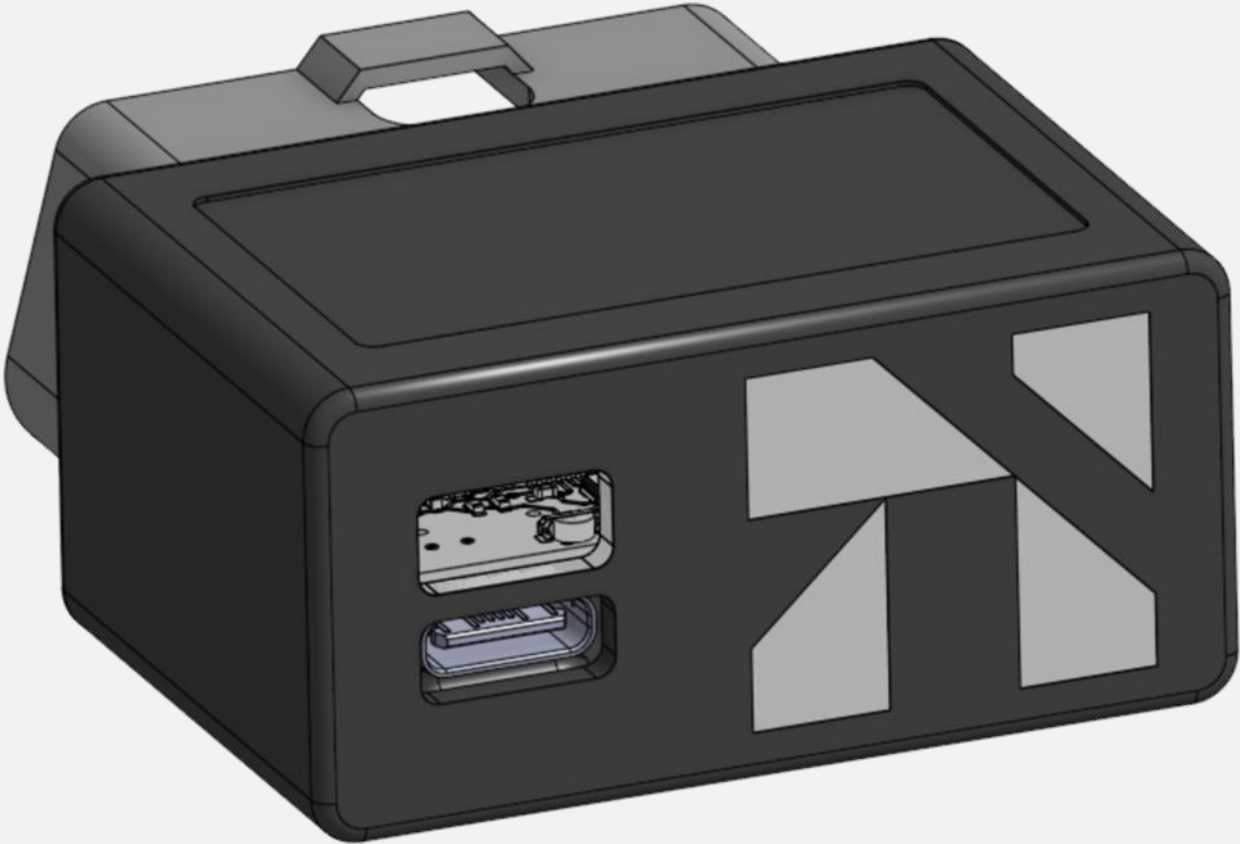




UDS Whisperer

Introducing UDS Whisperer

Compact, Wireless, Powerful.





THE CHALLENGES

Six critical challenges shaping the future of Vehicle Diagnostics

Remote Accessibility

Diagnostic tools must support remote and mobile access - across continents, not just across garages.

Automation & Scalability

Manual testing doesn't scale. Development and testing demand automated, scriptable diagnostics.

Early-Stage Compatibility

Diagnostic access is often blocked or incomplete during early development or pre-series stages.

Data Logging & Analysis

Real-time access to UDS and CAN logs is crucial for debugging, validation, and post-analysis.

OTA Updates & Maintenance

Firmware and configuration updates must be seamless, secure, and over-the-air - especially in large-scale deployments.

Cost & Hardware Limitations

Existing solutions are often bulky, expensive, or closed-source - limiting flexibility and innovation.

**PROBLEM**

Technicians and developers are often not co-located with the vehicle under investigation. However, diagnostic routines must still be executed - remotely, repeatedly, and in some cases fully automated during development, testing, or pre-delivery validation. Traditional diagnostic tools are costly, stationary, and rarely offer remote or scriptable interfaces. Furthermore, built-in remote diagnostics services are often unavailable or not yet activated in early vehicle stages, such as during manufacturing or before delivery.

SOLUTION

UDS Whisperer is a compact, wireless, and fully programmable diagnostics tool. It bridges the gap between vehicle and engineer - on-site or remotely - by enabling automated UDS communication, flashing, data logging, and remote control over WiFi, BLE, or Zigbee.

Whether used in the field, during fleet commissioning, or in a development lab, UDS Whisperer simplifies diagnostics and accelerates development.



SOLUTION OVERVIEW

Detailed Description

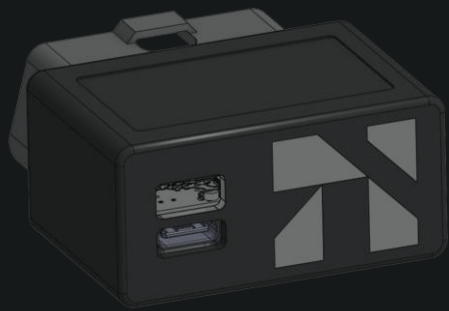




Compact, Wireless, Powerful.

**Core
Capabilities**

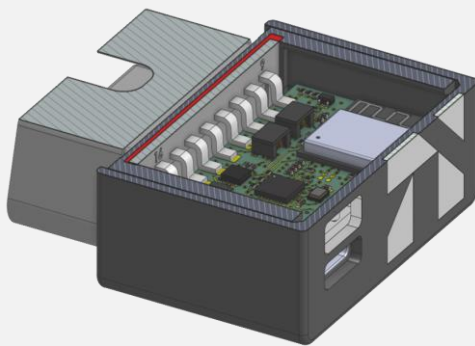
- UDS-Stack, ISO-TP, Raw CAN
- WiFi (Station / AP / Mesh), BLE, Zigbee
- UDP / TCP Communication
- Gigabytes of Storage (microSD card)
- Customer Code
- Rich Feature set (flashing, logging, services, ...)



Onboard Diagnostics

**Smart
Connectivity**

- Mobile hotspot compatible
- Seamless link to e:fs & customer infrastructure
- OTA firmware updates
- Mixed Standalone / Cloud Operation





Connectivity Services

- Bluetooth Low Energy (BLE)
 - Configuration Interface
 - PID mirroring
 - Triggering
- WiFi (Station, Access Point)
 - Direct mode
 - Cloud mode
- Zigbee*
- 2.4 GHz Mesh
- USB



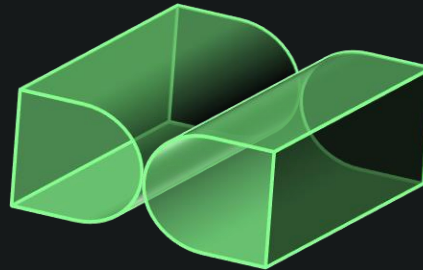
CAN and Diagnostic Services

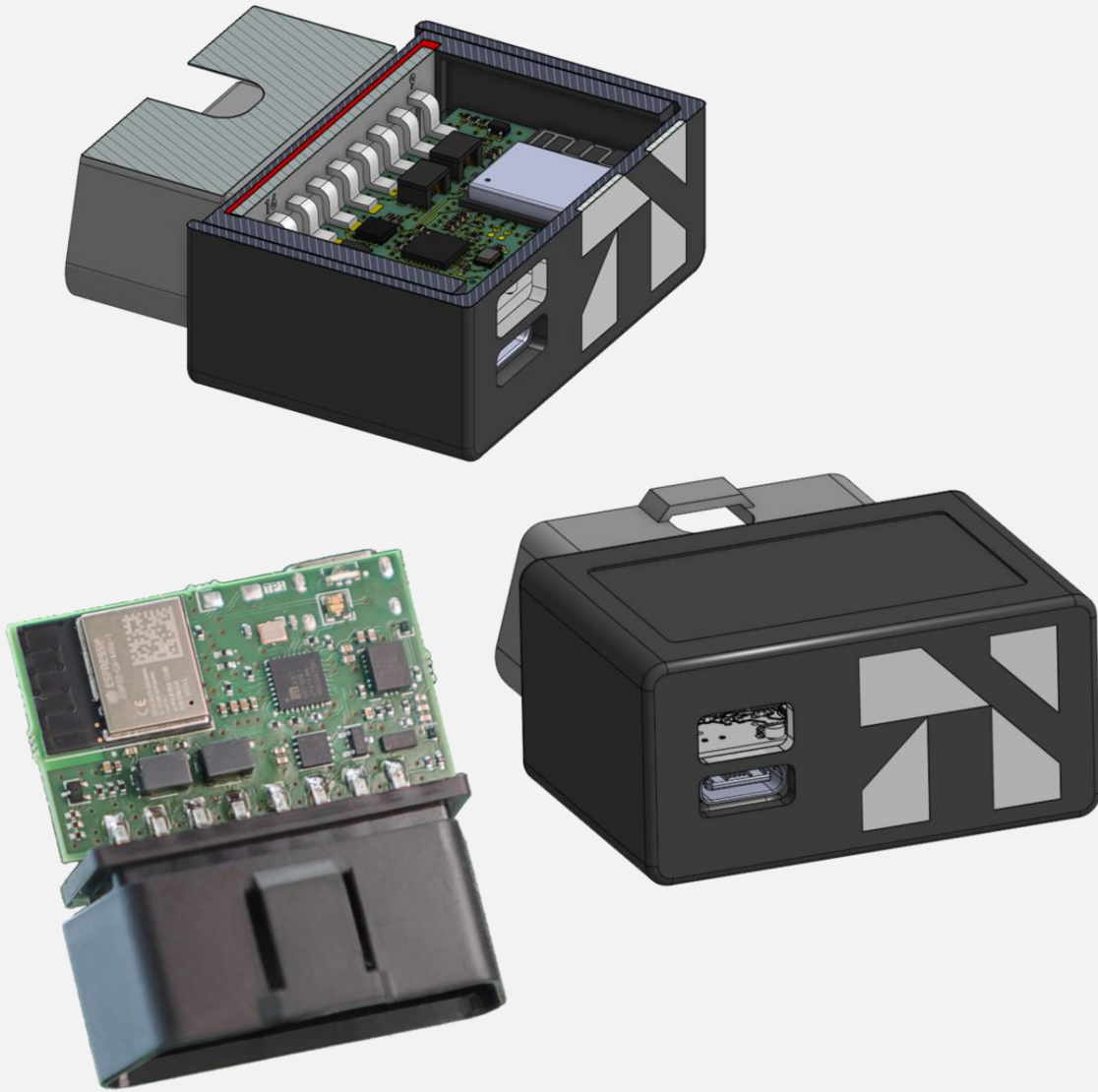
- OBD/UDS
- ISO-TP
- Raw CAN
- TCP/IP
- DoIP*



Using mobile hotspot

- Connection with efs infrastructure
- Connection with customer infrastructure
- Firmware Updates OTA





Hardware V2.0

v2.0: More Power, Same Size

- 5-40 V input, full ESD & fault protection
- Automotive-grade (excl. μ C), -40°C to 105°C
- microSD, Flash Encryption, Secure Boot
- Crypto HW: AES, SHA, RSA, HMAC
- Wireless: WiFi, BLE, ESP-NOW, Zigbee
- Wired: USB C, CAN, Ethernet
- RGB Indicator

→ Enhanced features, unchanged form factor



Cloud Server



Stack

Editor

This stack will be deployed using `docker compose`
You can get more information about Compose file

Define or paste the content of your docker con

```
1 version: '3.8'
2
3 services:
4   router-alpine:
5     build:
6       context: .
7       dockerfile_inline: |
8         FROM alpine:latest
9         RUN apk -Uuv add make b
10      command: >
11        /bin/sh -c "cd /app &&
12      ports:
13        - "2700:2700"
14        - "2702:2702"
15      volumes:
16        - /home/g3gg0/efs-obd/:/a
17      cap_add:
18        - SYS_PTRACE
19      security_opt:
20        - seccomp:unconfined
21
```

dev.udswhisperer.de Info Display - Info Display

Logins: 143
Packets: 0
Uptime: 18d 5h 23m 51s

Info

Name: DPDU: EDF76FG
Version: 1.5.0.15-80c28b2
Latency: 46.183 ms
Login time: 2025-06-06 08:42:44
Last data: 2025-06-06 09:13:04
Address: 10.233.75.49:35122
MAC: 00:00:00:00:00:00
Encrypted: AES256

1

Options

Channel: 0
Packets Sent: 733
Payload Tx: 0 pkt / 0 byte
Payload Rx: 0 pkt / 0 byte
Last Packet: 0

Open WebClient

Name: #999999 Dev-Flashbrett
Version: v0_test-27-g4a5a771
Latency: 223.798 ms
Login time: 2025-06-06 08:42:45
Last data: 2025-06-06 09:13:00
Address: 10.233.98.183:5363
MAC: 54:32:04:7e:ae:f8
Encrypted: AES256

0

Options

Channel: 1
Packets Sent: 17803
Payload Tx: 0 pkt / 0 byte
Payload Rx: 209 pkt / 2975 byte
Last Packet: 0

Options

Name: #200000 EI-EF2211
Version: 913edfd-dirty
Latency: 70.076 ms
Login time: 2025-06-06 08:42:44
Last data: 2025-06-06 09:13:04
Address: 10.233.98.183:15919
MAC: 8c:bf:ea:ff:fe:b4
Encrypted: AES256

2

Options

Channel: 2
Packets Sent: 2451
Payload Tx: 0 pkt / 0 byte
Payload Rx: 0 pkt / 0 byte
Last Packet: 0

Options

Actions for '#999999 Dev-Flashbrett'

Trigger firmware update

Change Channel

Set Identification

Set Cloud Server

Disconnect

Reboot

Admin interface

Pitch Deck | UDS Whisperer



UDS Connectivity



ISO 22900

D-PDU API



Start

Startseite (F1)

Einstellungen (F2)

Protokolle verwalten (F3)

Flashen (F5)

Diagnosefunktionen (F6)

Gateway Codier-Liste (F7)

Special Tools (F8)

Sammeldienste (F9)

Service Key (F10)

SFD (F12)

SWaP/FoD

SOD (F4)

Macro (F11)

Einstellungen und Protokolle

Flashen

Diagnose

Gesamtfahrzeug

Script

Projekt: AU536/3 Q8

↺

↻

+

PDX Import

VIS: VINFO_AU58XCAN

VIN: -

KM-Stand: -

Interface: #999999 Dev-Flashbrett

SN: 1

ID: #999999 Dev-Flashbrett

⚠

VCI Auswählen

VAS6154

DoIP/Ethernet-Kabelverbindung

Sonstige VCI

e.fs OBD Online

VAS6154 ODIS

Textfilter (Typ, Seriennummer, Hersteller) zur gezielten Suche

Enter text to search...

Typ	Seriennummer	Verbindungsart	Hersteller	Protokoll-Typ
#200000 EI-EF2211	8C:8F:EA:FF:FE:B4	Cloud	e:fs TechHub GmbH	CAN
#999999 Dev-Flashbrett	54:32:04:7E:AE:F8	Cloud	e:fs TechHub GmbH	CAN
#000000 UDS-Whisperer	8C:8F:EA:FF:FE:B4	USB	e:fs TechHub GmbH	CAN

efs_DPDU_API:

i

Konfiguration

⚙

OK

✓

iDEX Ordner

iDEX@audi.de

iDEX Group Wiki

Support Info

iDEX Handbuch

iDEX Sharepoint

+49 841 / 89 43999

iDEX Neo

Version: 3.0.3.3

User Lizenziert bis: 28.04.2026

iDEX Neo



DiagRA D Professional 7.44.40.44670 - 33 Scan-Tool

File Trigger Functions Extras View Options Help

33 Scan-Tool

ISO 15765-4 (CAN)
ISO 9141-2, ISO 14230-4 (K-Line)
ISO 15765-4 (CAN)
SAE J1850 VPW/PWM
SAE J1939-03 (CAN)
ISO 27145 (ISO-CAN)

Auto FD ONLINE

Standard Extended Memory

J1979 J1939/ISO27145

E8 ECM-EngineControl E9 HPC1-HybridPtCtrl1 EA TCM-TransmisCtrl EB FPCM-FuelPumpCtrl EC HPC2-HybridPtCtrl2 ED BSCM-BrakeSystem EF BECM-B+EnergyCtrl

PID	Value	Unit	Comment
01	0000 0000 0000 0111 1110 0101 0000 0000	Bit	Monitor status since DTCs cleared • MIL off, 0 fault code entries • Misfire monitoring supported and complete • Fuel system monitoring supported and complete • Comprehensive component monitoring supported and complete • Catalyst monitoring supported and complete • Heated catalyst monitoring not supported • Evaporative system monitoring supported and complete • Secondary air system monitoring not supported • PM filter monitoring not supported • Oxygen sensor monitoring supported and complete • Oxygen sensor heater monitoring supported and complete • EGR and/or VVT system monitoring supported and complete
03	0000 0000 0000 0000	Bit Bit	Fuel system A status Fuel system B status
04	0.0	%	Calculated load value
05	21	°C	Engine coolant temperature
06	0.0	%	Short term fuel trim - Bank 1
07	3.9	%	Long term fuel trim - Bank 1
0A	435	kPa	Fuel pressure (gauge)
0B	98	kPa	Intake manifold absolute pressure
0C	0	1/min	Engine RPM
0D	0	km/h	Vehicle speed sensor
0E	0.0	°	Ignition timing advance for #1 cylinder
0F	27	°C	Intake air temperature
10	0.00	g/s	Air flow rate from mass air flow sensor

Mode 1 Mode 2 Mode 3 Mode 4 Mode 5 Mode 6 Mode 7 Mode 8 Mode 9 Mode A

☒ Cyclical ☐ Once

Overview

Selection

138

LOG FILE ASSET WORKSPACE

default-

DiagRA S / PassThru 04.04 500 kbps

DiagRA



SAE J2534

PassThru API

On Request



Open Design, Custom Code



Simple scripts using JSON

Ideal for automation of tedious tasks:

Clear Fault Memory

Set Coding Word

Calibration Routine

...

```
    "payload": ["0x01", "0x02", "0x03"]
  }, {
    "type": "ELEM_LOG",
    "log": {
      "type": "LOG_STATUS",
      "message": "Enter programming session"
    }
  }, {
    "type": "ELEM_XMIT",
    "xmit": {
      "sid": "0x10",
      "payload": ["0x02"]
    }
  }, {
    "type": "ELEM_LOG",
    "log": {
      "type": "LOG_STATUS",
      "message": "Security access"
    }
  }, {
    "type": "ELEM_SAK",
    "sak": {
      "key": "0x0B231389"
    }
  }, {
    "type": "ELEM_LOG",
    "log": {
      "type": "LOG_STATUS",
      "message": "Erase memory"
    }
  }, {
    "type": "ELEM_XMIT",
    "xmit": {
      "sid": "0x31",
      "payload": ["0x01", "0xFF", "0x00", "0x02", "0x71", "0x10"]
    }
  }, {
    "type": "ELEM_LOG",
    "log": {
      "type": "LOG_STATUS",
      "message": "Download"
    }
  }, {
    "type": "ELEM_DOWNLOAD",
    "download": {
      "address": "0x00007110",
```



Flash ECUs

Fire-and-forget flashing

Dataset

Application

Bootloader

Support .odx, .bin and .hex

Page Title | 18

```
[
  {
    "type": "ELEM_COMMPARM",
    "commparm": {
      "tx_id": "0x17FC0080",
      "rx_id": "0x17FE0080",
      "timeout_ms": 0
    }
  },
  {
    "type": "ELEM_LOG",
    "log": {
      "type": "LOG_STATUS",
      "message": "Erase memory"
    }
  },
  {
    "type": "ELEM_XMIT",
    "xmit": {
      "sid": "0x31",
      "payload": [
        "0x01", "0xFF", "0x00", "0x02", "0x71", "0x05"
      ]
    }
  },
  {
    "type": "ELEM_LOG",
    "log": {
      "type": "LOG_STATUS",
      "message": "Download"
    }
  },
  {
    "type": "ELEM_DOWNLOAD",
    "download": {
      "address": "0x00007105",
      "decompressed_size": "0x3E00",
      "address_len": 4
    }
  },
  {
    "type": "ELEM_LOG",
    "log": {
      "type": "LOG_STATUS",
      "message": "Check Signature"
    }
  },
  {
    "type": "ELEM_SIGNATURE",
    "signature": {
      "filename": "/int/CKE6.odx",
      "section": "SES_074_7105_4MF_CKE6_A4M"
    }
  }
]
```

```
</DATABLOCK>
<FLASHDATAS>
  <FLASHDATA ID="EMEM_074_7105_4MF_CKE6_A4MF1BK0000R_LCEFP20V034.FD_7105FLASHDATA">
    <SHORT-NAME>FD_7105FLASHDATA</SHORT-NAME>
    <LONG-NAME>7105 FLASH DATA</LONG-NAME>
    <DATAFORMAT SELECTION="BINARY"/>
    <ENCRYPT-COMPRESS-METHOD TYPE="A_BYTEFILE">PT-COMPRESS-METHOD</ENCRYPT-COMPRESS-METHOD>
    <DATA>0E9A909941100420081004BE009F9A4160E59041F000A03641FED478411D005A64413BDF</DATA>
  </FLASHDATA>
</FLASHDATAS>
</MEM>
</ECU-MEM>
</ECU-MEMS>
<ECU-MEM-CONNECTORS>
  <ECU-MEM-CONNECTOR ID="EMC_074_7105_4MF_CKE6_A4MF1BK0000R_LCEFP20V034">
    <SHORT-NAME>EMC_074_7105_4MF_CKE6_A4MF1BK0000R_LCEFP20V034</SHORT-NAME>
    <LONG-NAME>074_7105_4MF_CKE6_A4MF1BK0000R_LCEFP20V034</LONG-NAME>
    <SESSION-DESCS>
      <SESSION-DESC DIRECTION="DOWNLOAD">
        <SHORT-NAME>SD_074_7105_4MF_CKE6_A4MF1BK0000R_LCEFP20V034</SHORT-NAME>
        <LONG-NAME>074_7105_4MF_CKE6_A4MF1BK0000R_LCEFP20V034</LONG-NAME>
        <SESSION-SNR SHORT-NAME="SES_074_7105_4MF_CKE6_A4MF1BK0000R_LCEFP20V034"/>
        <DIAG-COMM-SNR SHORT-NAME="SinglJob_FlashJobUDS"/>
      </SESSION-DESC>
    </SESSION-DESCS>
    <ECU-MEM-REF ID-REF="EMEM_074_7105_4MF_CKE6_A4MF1BK0000R_LCEFP20V034"/>
    <LAYER-REFS>
      <LAYER-REF DOCTREF="BV_ChassContrUDS" DOCTYPE="LAYER" ID-REF="BV_ChassContrUDS"/>
    </LAYER-REFS>
  </ECU-MEM-CONNECTOR>
</ECU-MEM-CONNECTORS>
</FLASH>
</ODX>
```





Custom Code

Open for custom modifications via customization options.

32 Bit MCU, 4 MiB Flash, FPU, FreeRTOS

Optionally:
Source Code access

```
static int isotp_send_flow_control(isotp_link_t *link, uint8_t flow_status, uint8_t
{

    isotp_can_message message;
    int ret;

    /* setup message */
    message.as.flow_control.type = ISOTP_PCI_TYPE_FLOW_CONTROL_FRAME;
    message.as.flow_control.FS = flow_status;
    message.as.flow_control.BS = block_size;
    message.as.flow_control.STmin = isotp_ms_to_ (st_min_ms);

    /* send message */
#ifdef ISO_TP_FRAME_PADDING
    (void)memset(message.as.flow_control.data, 0x55, sizeof(message.as.flow_control.data));
    ret = isotp_user_send_can(link, message.as.data_array.ptr, message.as.data_array.size);
#else
    ret = isotp_user_send_can(link, message.as.data_array.ptr, message.as.data_array.size);
#endif

    return ret;
}

static int isotp_send_single_frame(isotp_link_t *link, uint32_t id)
{

    isotp_can_message message;
    int ret;

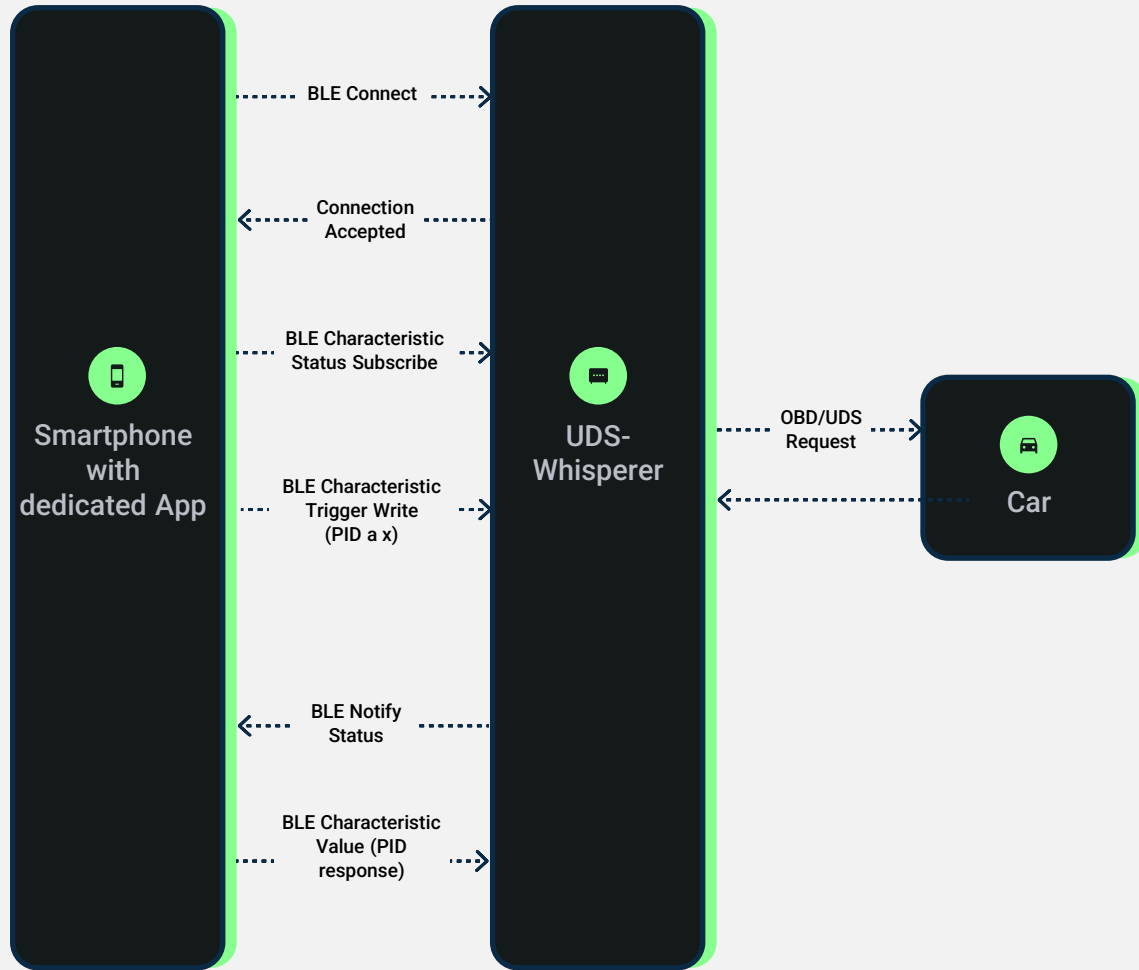
    /* multi frame message length must greater than 7 */
    assert(link->send_size <= 7);

    /* setup message */
    message.as.single_frame.type = ISOTP_PCI_TYPE_SINGLE;
```



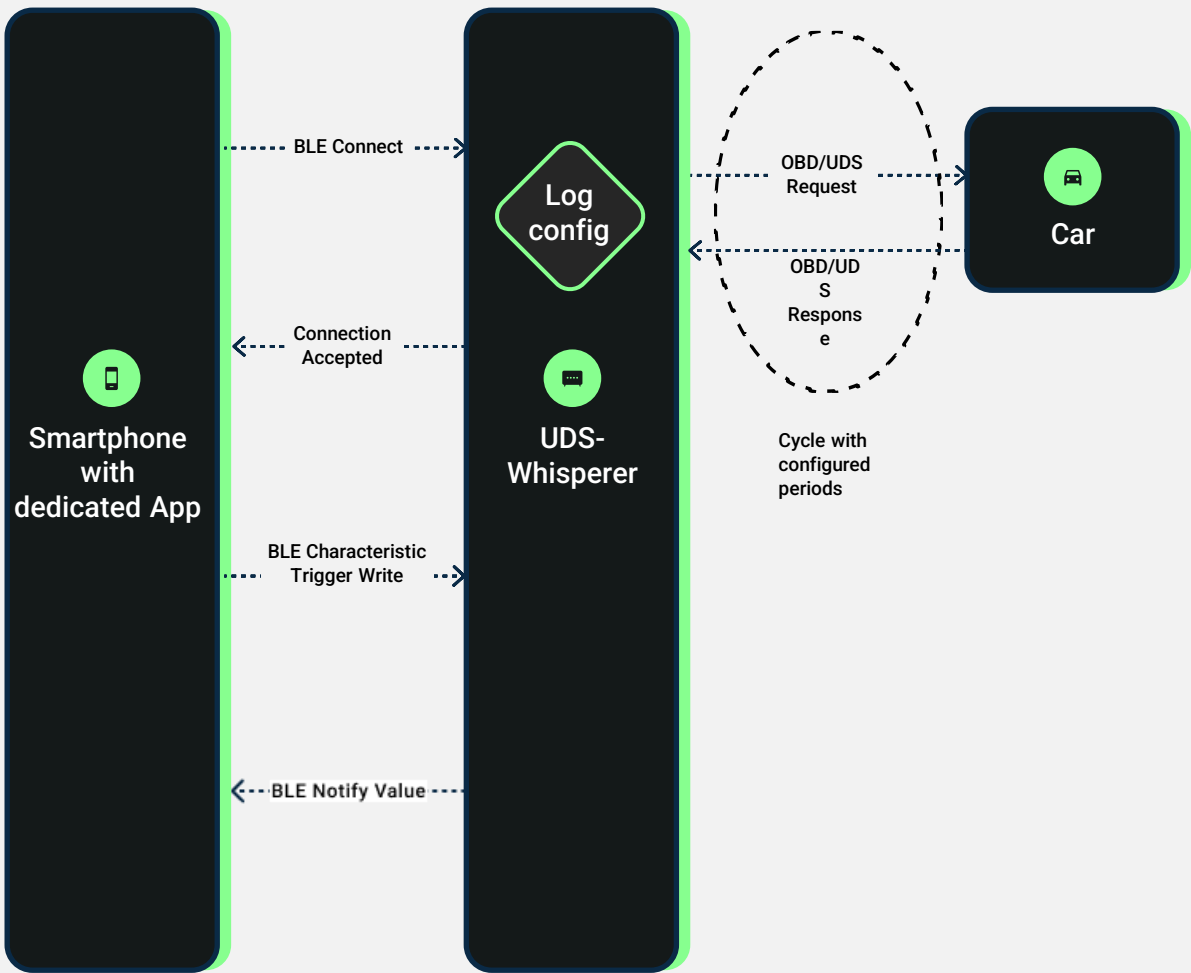
Focus:
Use Cases

01	OBD-Snapshot Mode
02	OBD-Dashcam Mode
03	Raw CAN Bridge
04	UDS Bridge
05	Car Commissioning (DogTag)
06	Automated Flasher
07	World Wide UDS Debugger



OBD-Snapshot Mode

- UDS Whisperer gets plugged into the car's OBD port
- Peer Device (Smartphone, PC...) connects using BLE
- Via BLE the Vehicle Identification Number is advertised and other configurable parameters are provided
- By writing a parameter identifier PID to a characteristic or by directly using READ access to a characteristic a query can be triggered
- After a successful query the UDS Whisperer provides the data to the peer



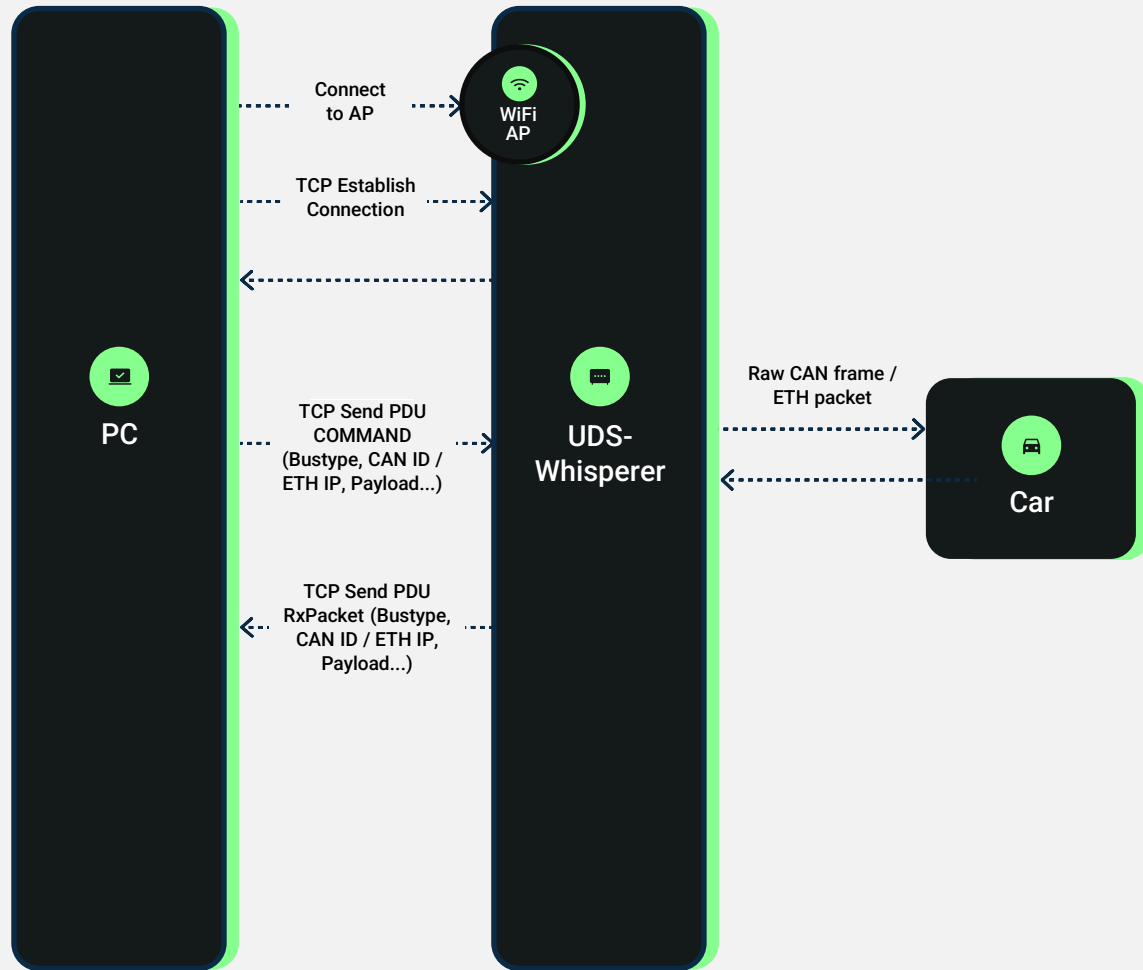
OBD-Dashcam Mode

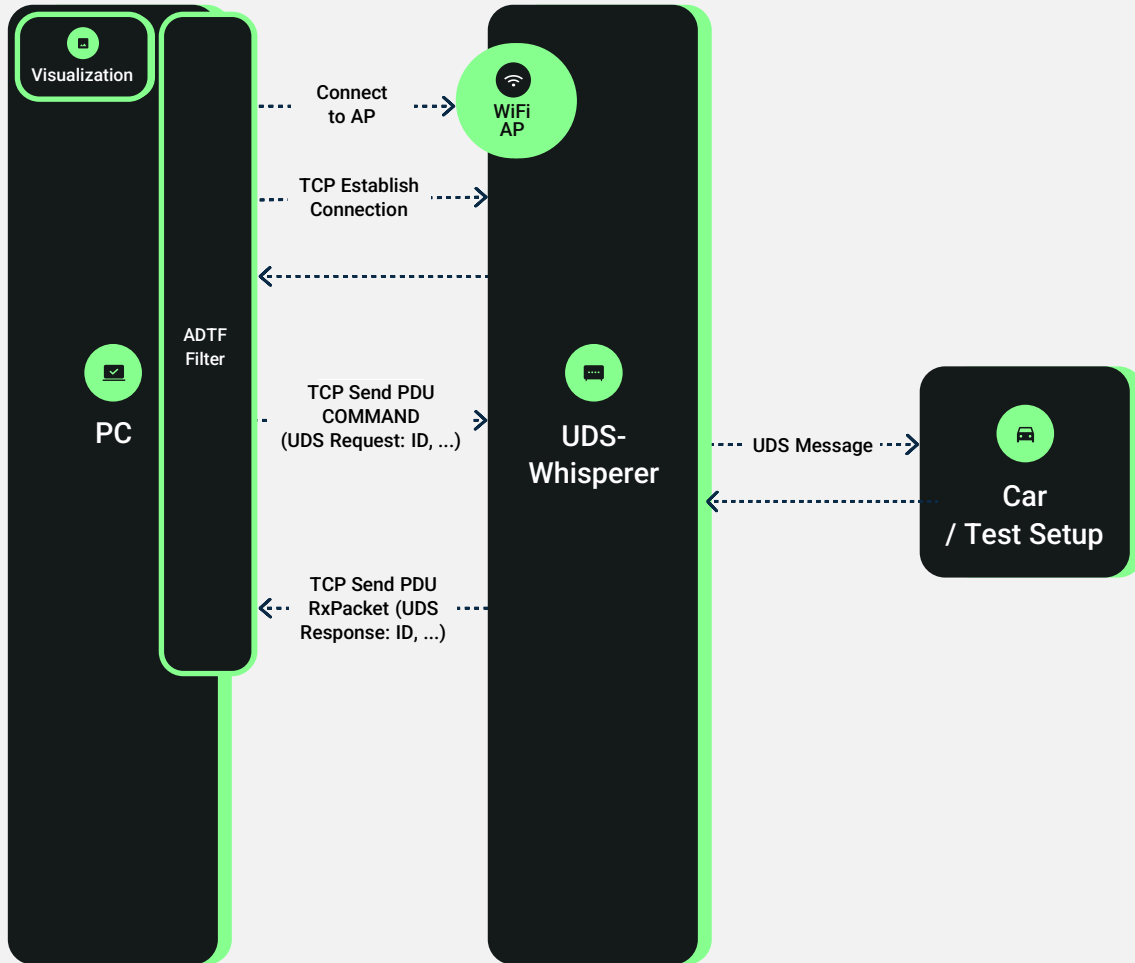
- UDS Whisperer contains a standardized log config
 - This config contains the parameters to query and the cycle time
- The device periodically queries the configured parameters (DIDs or PIDs) from the car and logs them to the non volatile memory (internal or SD card)
- Peer device can set a trigger via a Write characteristic
 - Via a NOTIFY characteristic the peer device receives the data captured from within a specified period
 - Larger data can also be transferred by WiFi



Raw CAN Bridge

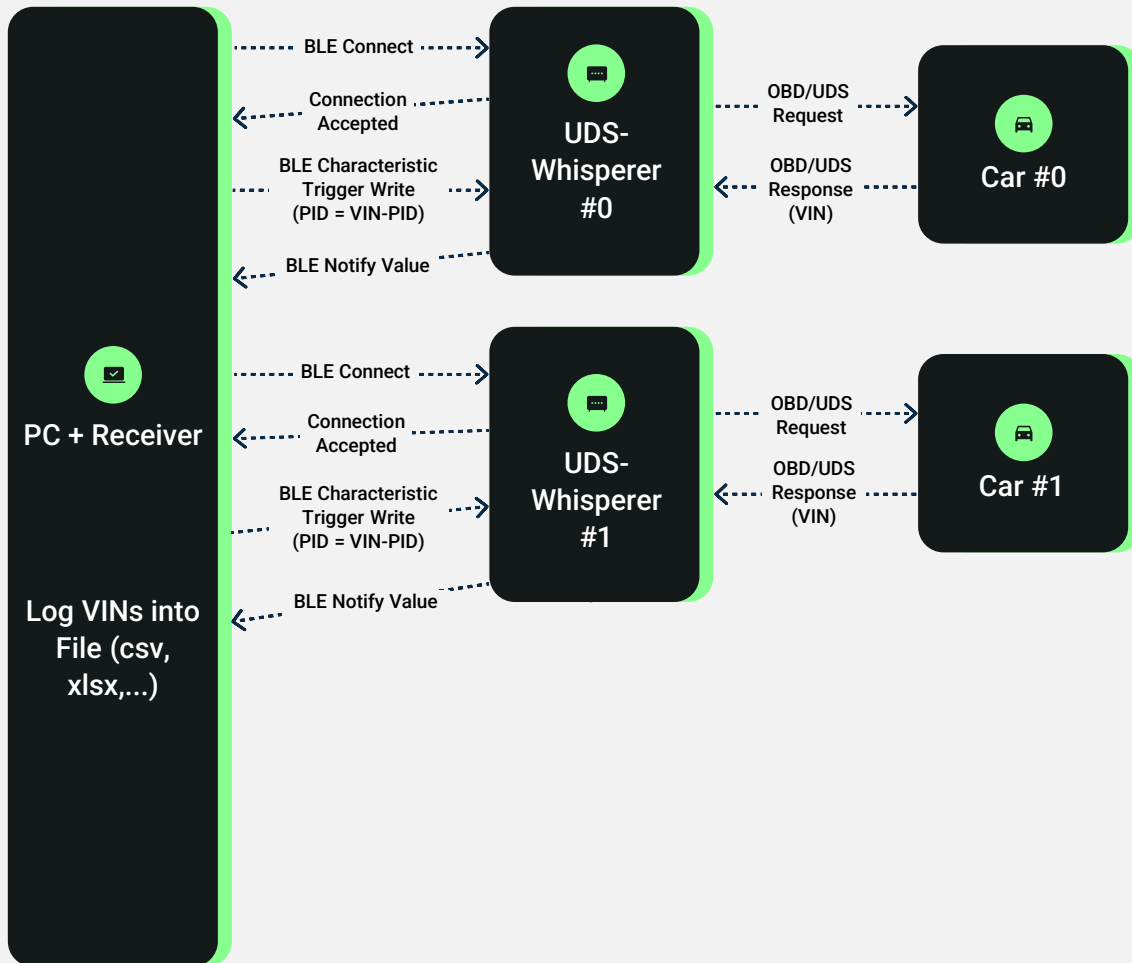
- UDS Whisperer is connected to a car or development setup
- A wireless access point (AP) is created by the device
- The user's PC connects to AP and receives an IP address via DHCP
- UDS Whisperer listens on e.g. port 20000 for incoming TCP traffic
- This traffic is converted to CAN frames
- In the opposite direction CAN frames are converted to TCP packets and sent to the PC





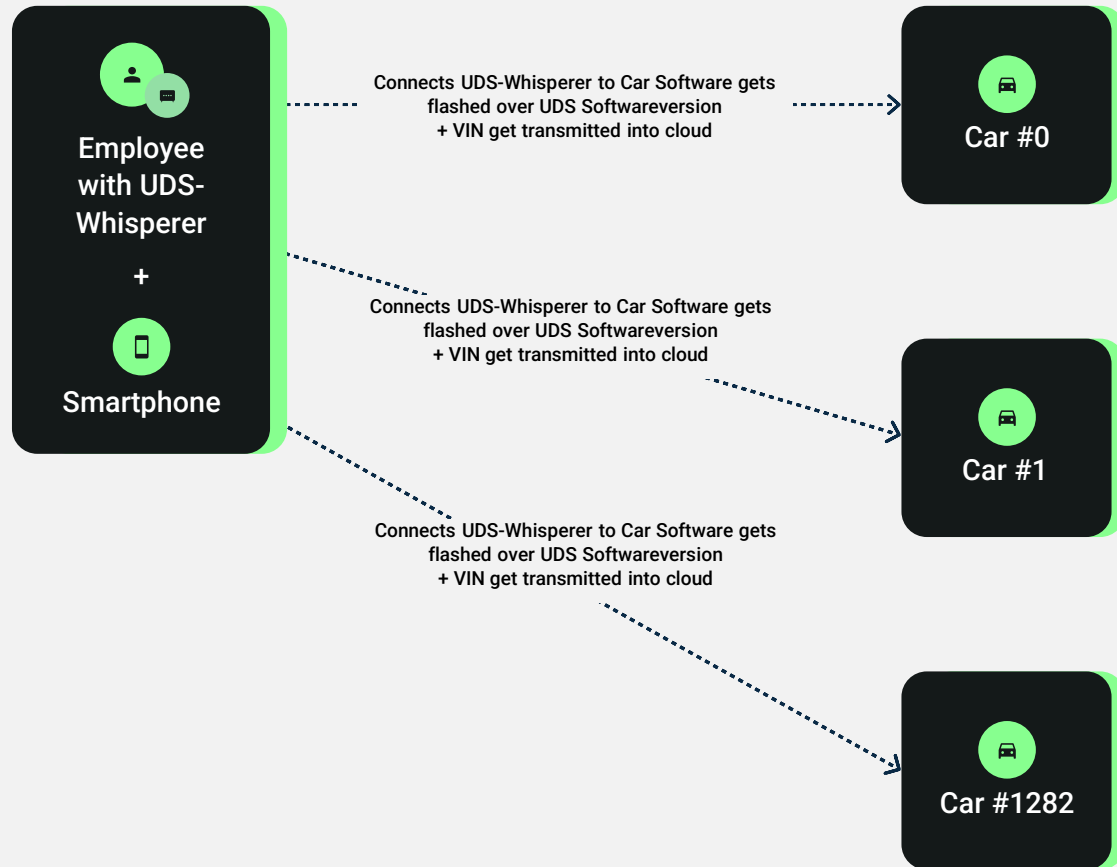
UDS Bridge

- Like the CAN bridge the UDS Whisperer and PC connect via WiFi
- Communication uses TCP
- Difference is that now instead of the CAN layer the UDS layer is used
- The created TCP stream towards the PC can be integrated as filters in CASSANDRA/ADTF or other tooling



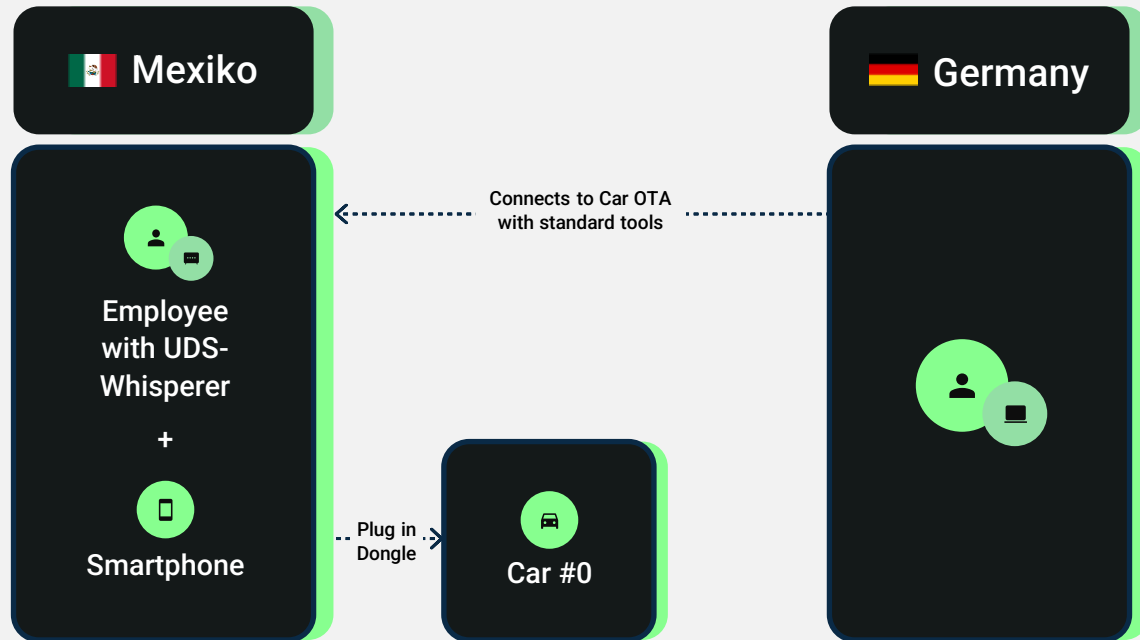
Car Commissioning (Dog Tag)

- When loading a large number of vehicles, it is cumbersome to document which vehicle with which VIN was loaded onto or unloaded from the ship and when
- UDS Whisperer can help
 - A driver plugs in the UDS Whisperer every time they get into a vehicle
 - The UDS Whisperer reads the VIN and optionally other parameters such as the software version
 - The driver drives off the ship and passes a nearby base station
 - The UDS Whisperer connects to the base station, which logs the transmitted VIN in any desired format
 - Afterwards, the driver unplugs the dongle and plugs it into the next vehicle



Automated Flasher

- **Scenario:** A large number of vehicles are to be flashed with a new software version or dataset.
- **Problem:** It is tedious and time-consuming to service each vehicle with a laptop (iDEX/ODIS/...) and navigate through menus. Even with scripting there is a significant delay
- The UDS Whisperer receives the package to be flashed via Over-The-Air (OTA)
- The on-site staff use the preconfigured dongle by plugging it into each vehicle and waiting for the flashing process to complete.
- During the flashing process, the dongle blinks blue; afterwards, it lights up purple. In case of an error, it blinks red
- After flashing, the VIN and software version are transmitted to the employee's device
- This information can then be forwarded to a backend system
- Other routines can also be automated, such as reading or clearing fault memory entries

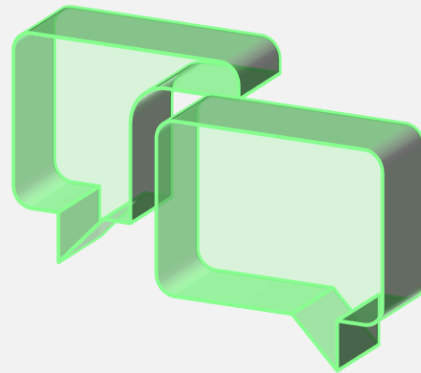


Remote Diagnosis

- Scenario: A large number of vehicles are to be flashed with a new software version or dataset.
- Problem: It is tedious and time-consuming to service each vehicle with a laptop (iDEX/ODIS/...) and navigate through menus. Even with scripting there is a significant delay
- The UDS Whisperer receives the package to be flashed via Over-The-Air (OTA)
- The on-site staff use the preconfigured dongle by plugging it into each vehicle and waiting for the flashing process to complete.
- During the flashing process, the dongle blinks blue; afterwards, it lights up purple. In case of an error, it blinks red
- After flashing, the VIN and software version are transmitted to the employee's device
- This information can then be forwarded to a backend system
- Other routines can also be automated, such as reading or clearing fault memory entries



Questions?
Contact us



People Lead and Developer

Georg Hofstetter

georg.hofstetter@efs-techhub.com

+49 845839730042



People Lead

Lars Biermanski

lars.biermanski@efs-techhub.com

+49 1622844958