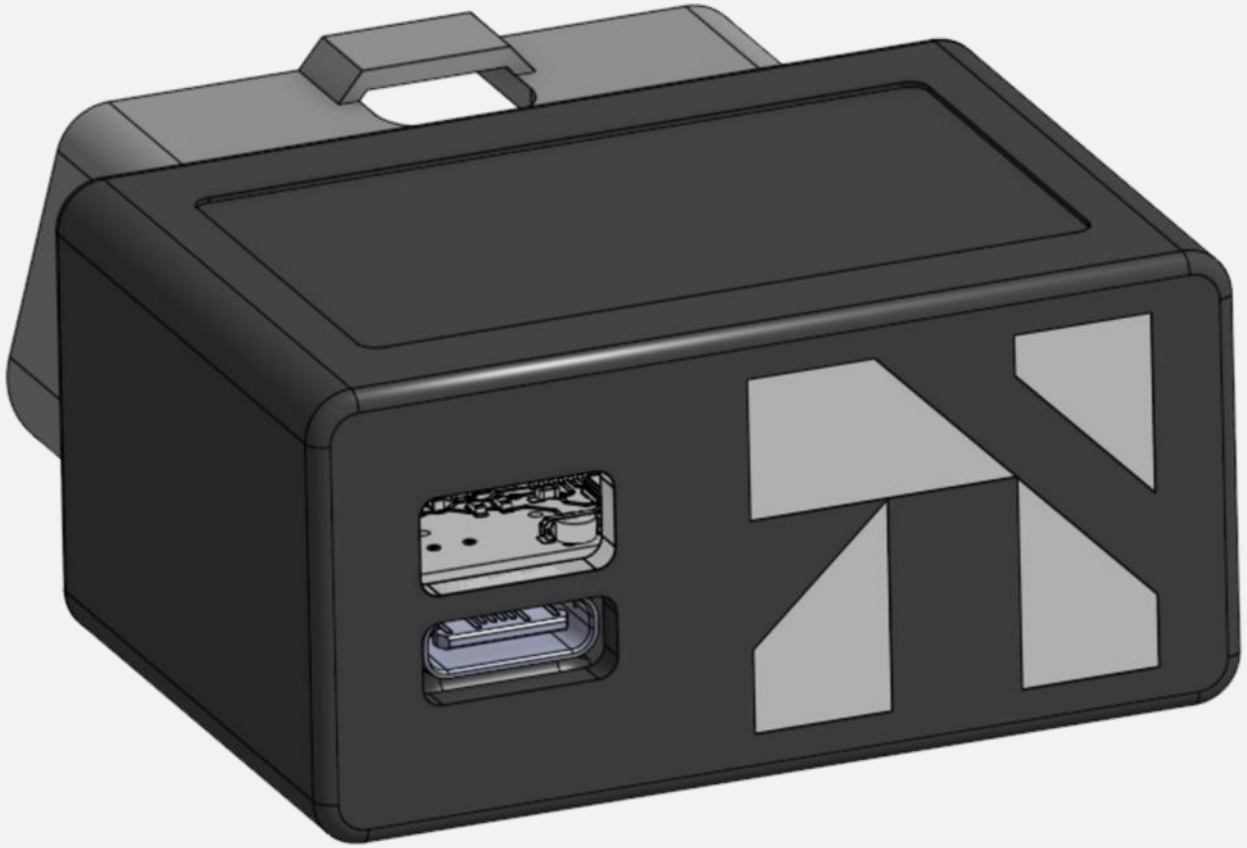
UDS Whisperer

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.

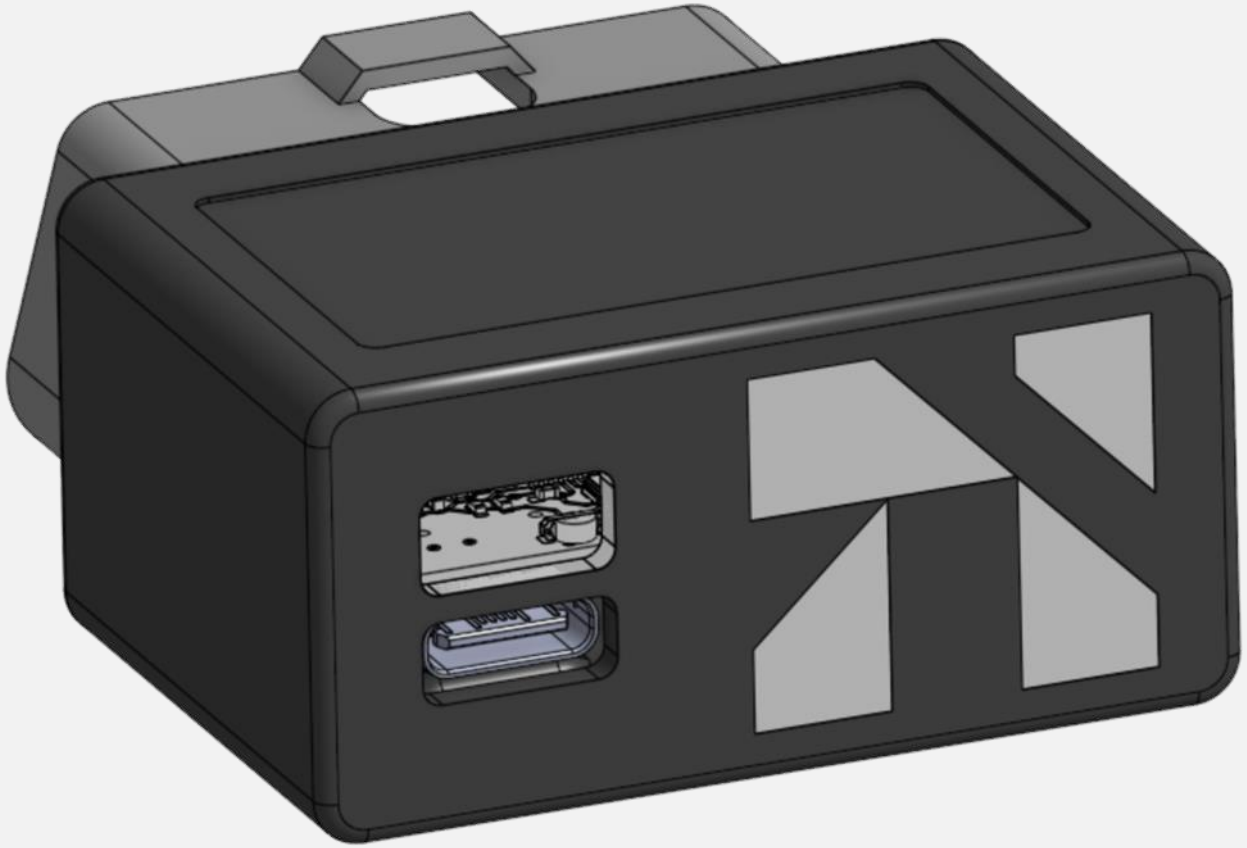
SOLUTION

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.

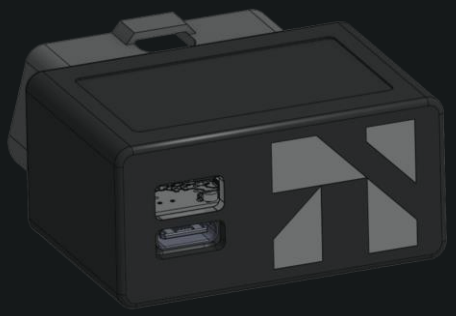
SOLUTION OVERVIEW

Detailed Description



Core Capabilities

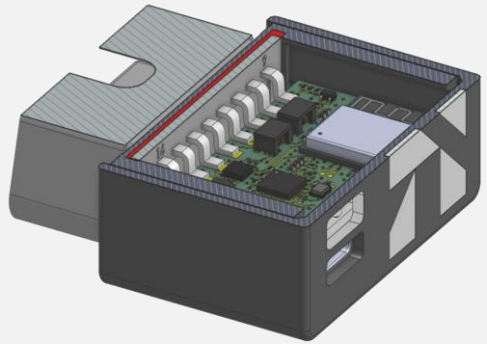
- UDS-Stack, ISO-TP, Raw CAN
- WiFi (Station / AP / Mesh), BLE, Zigbee
- UDP / TCP Communication
- Gigabytes of Storage (microSD card)
- Customer Code
- Rich Feature set (flashing, logging, services, …)



Onboard Diagnostics

Smart Connectivity

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



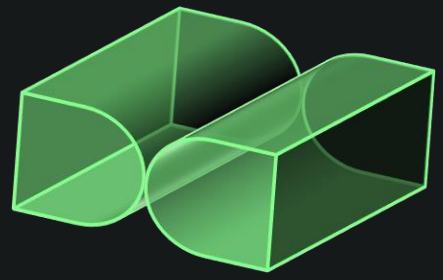
Modular Firmware in Detail

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

* in development

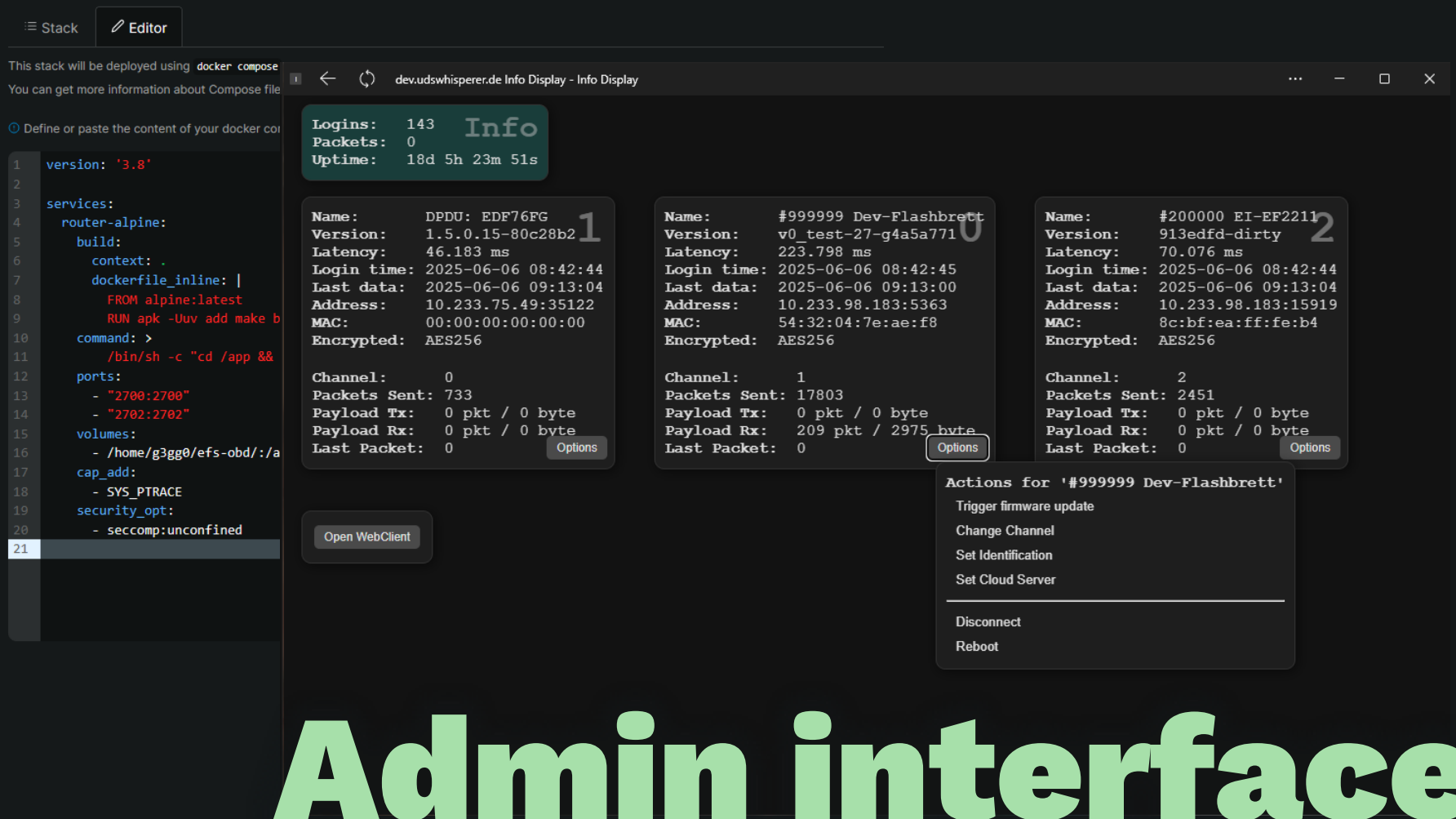
Hardware V2.0

v2.0: More Power, Same Size

- 5-40 V input, full ESD & fault protection
- Automotive-grade (excl. µC), –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

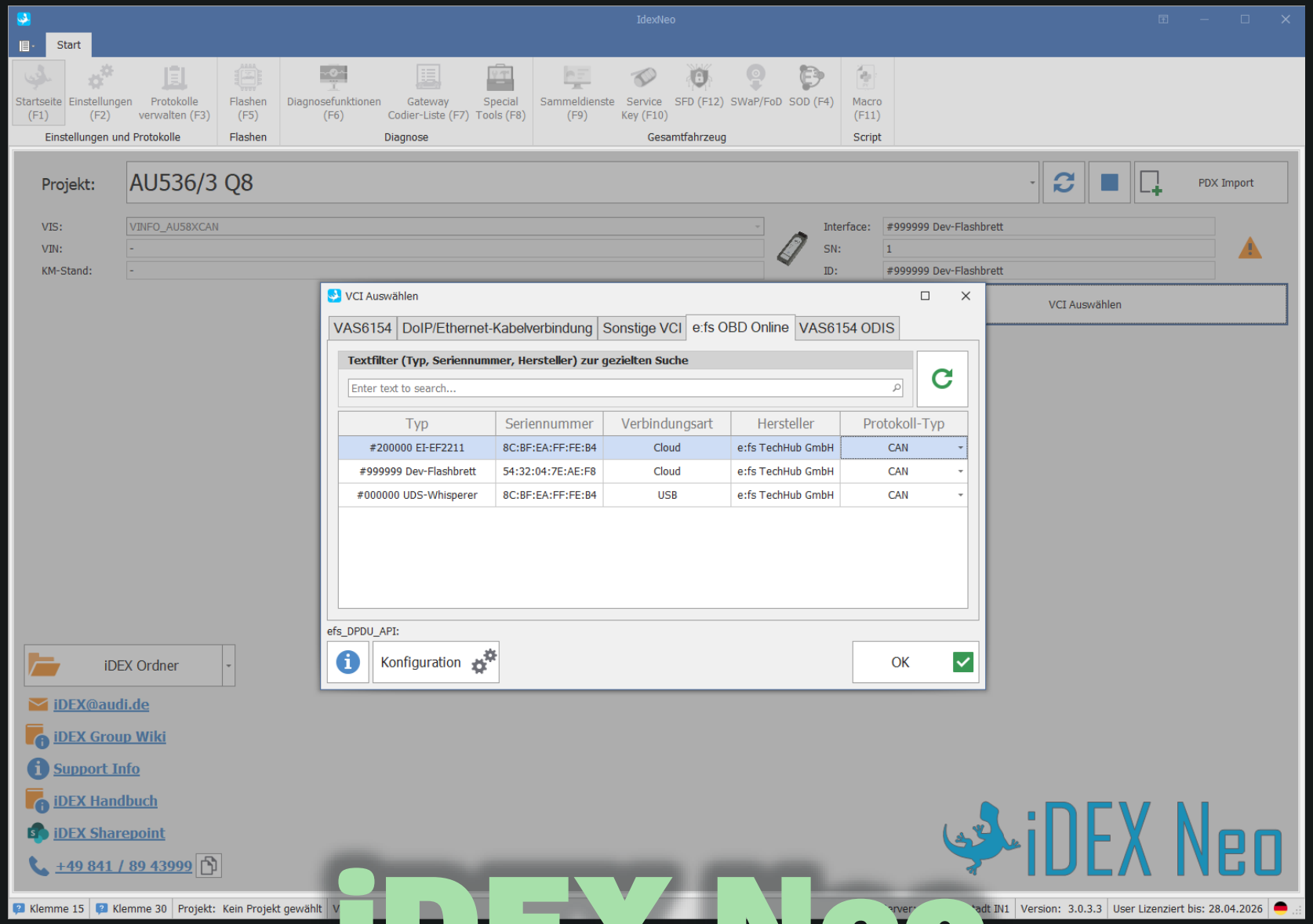
Cloud Server



Admin interface

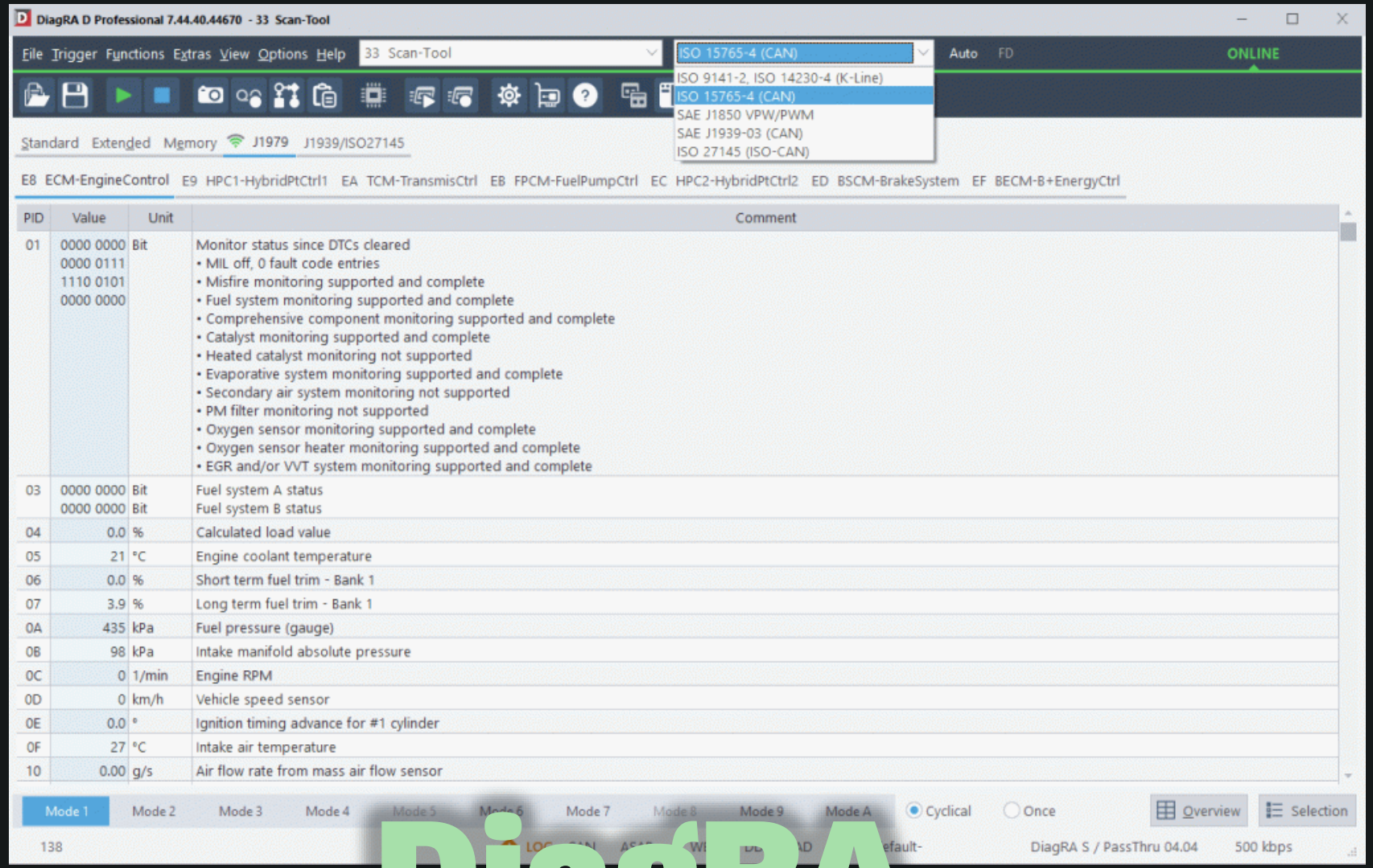

UDS Connectivity

ISO 22900
D-PDU API



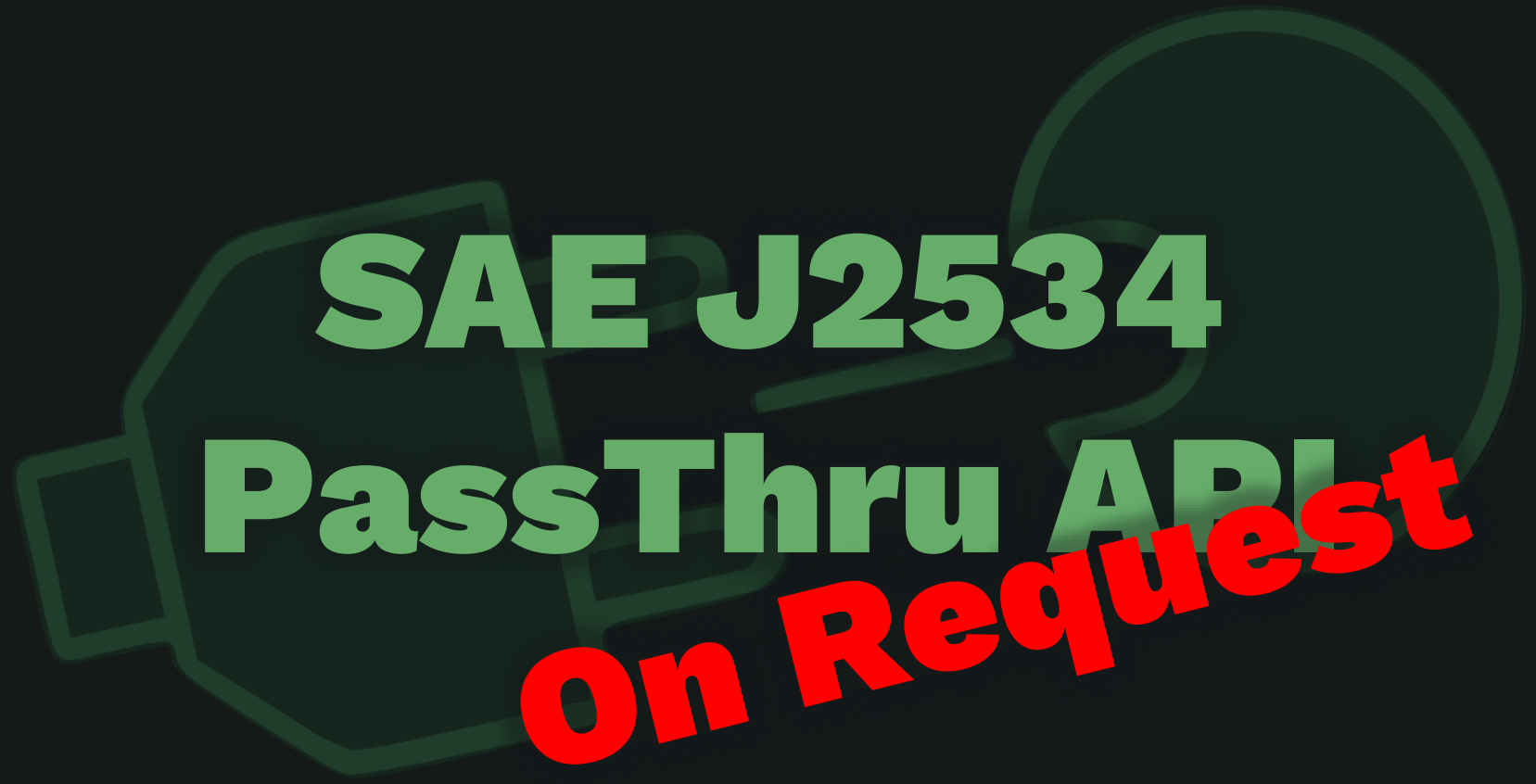
iDEX Neo

Pitch Deck | UDS Whisperer



DiagRA

(USB only)


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`

Custom Code

Open for custom modifications via customization options.

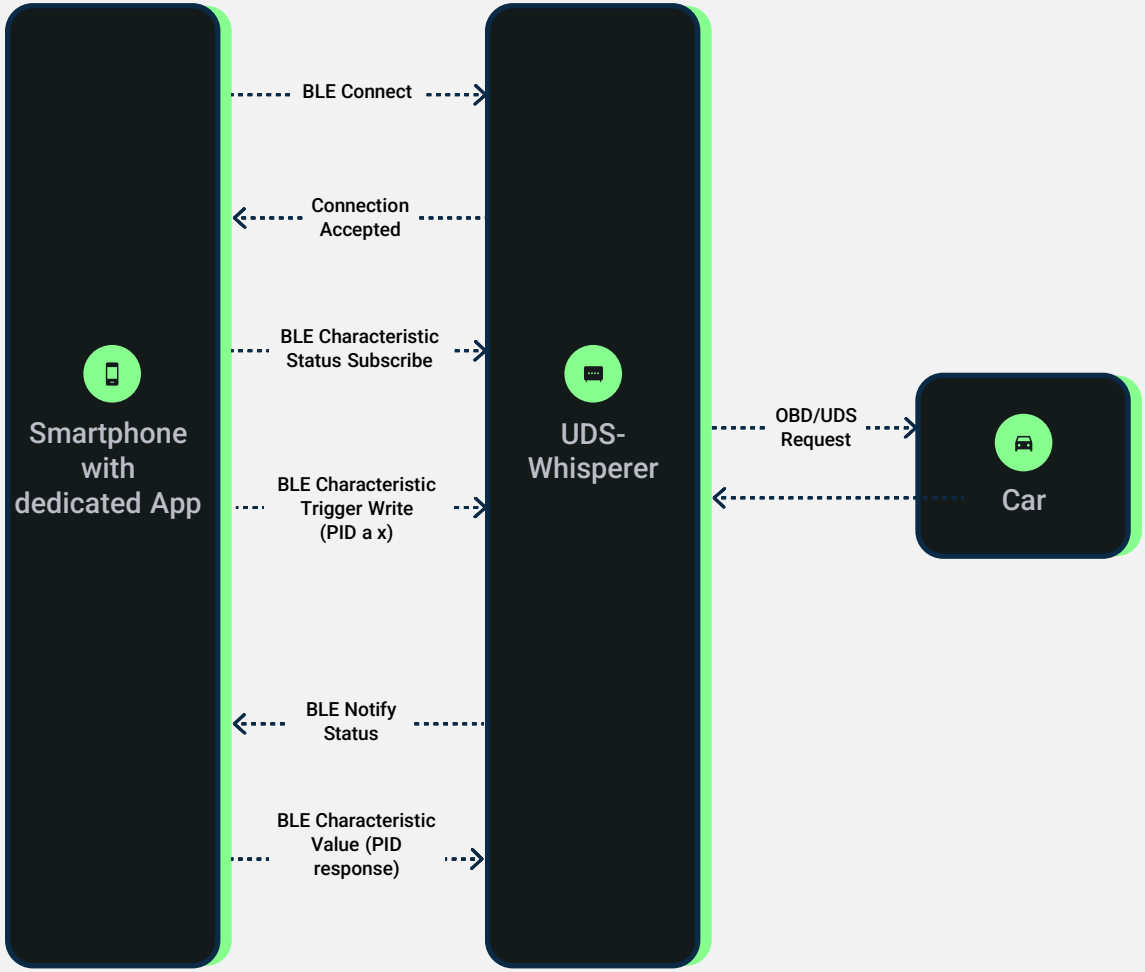
32 Bit MCU, 4 MiB Flash, FPU, FreeRTOS

Optionally:
Source Code access

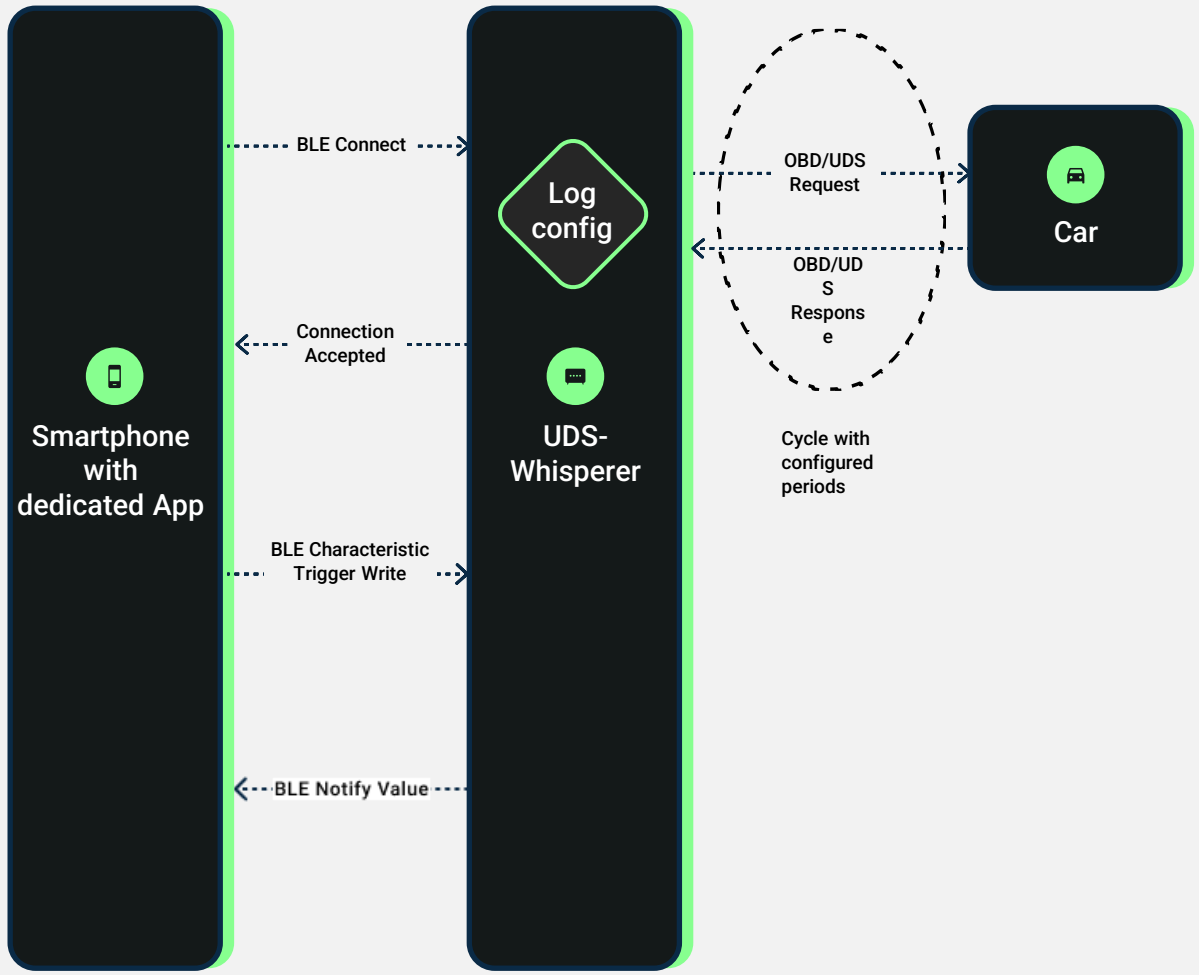
Focus:
Use Cases

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



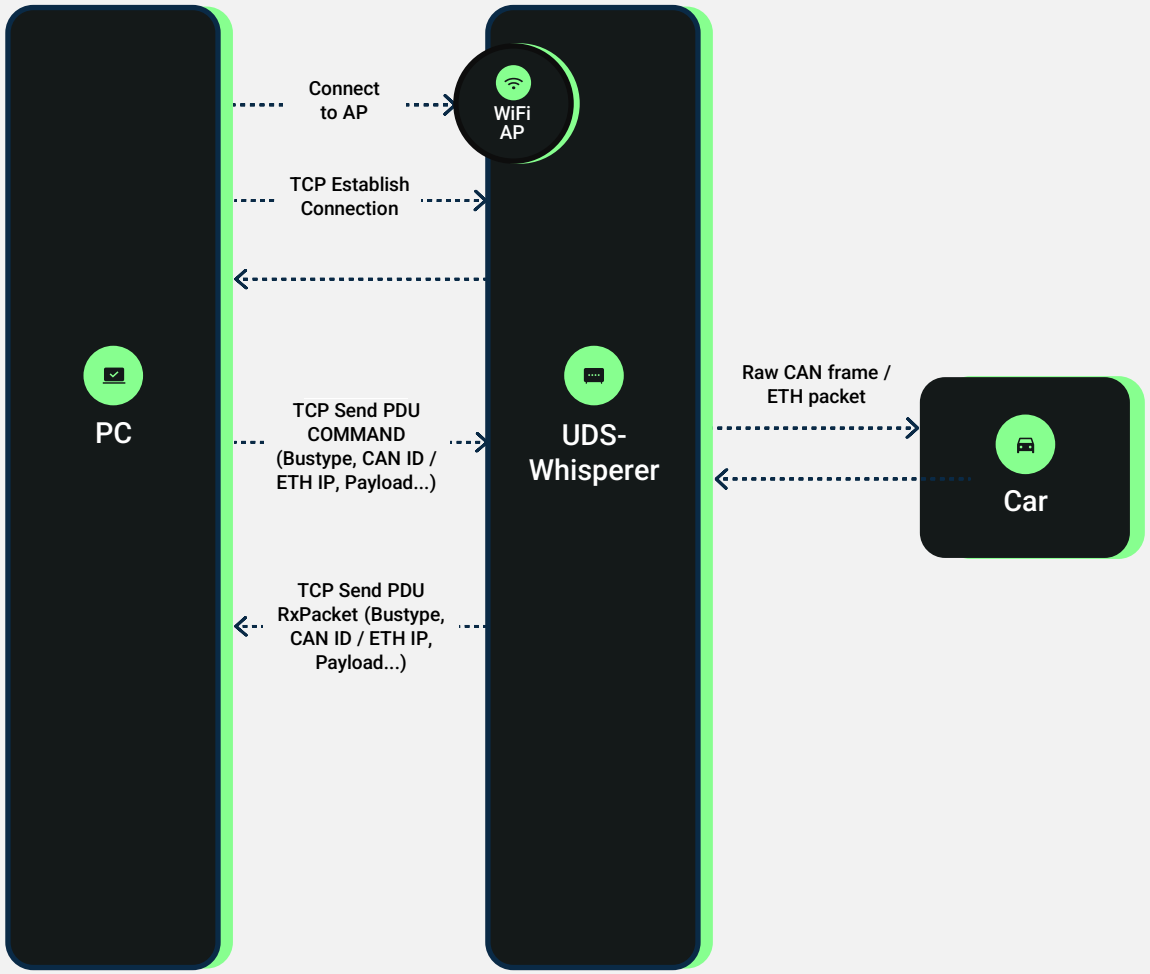
- UDS Whisperer gets plugged into the car‘s OBD port
- Peer Device (Smartphone, PC…) connects using BLE
- Via BLE the Vehicle Identification Number is advertised and other configurable parameters are provided
- By writing a parameter identifier PID to a characteristic or by directly using READ access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



OBD-Dashcam Mode

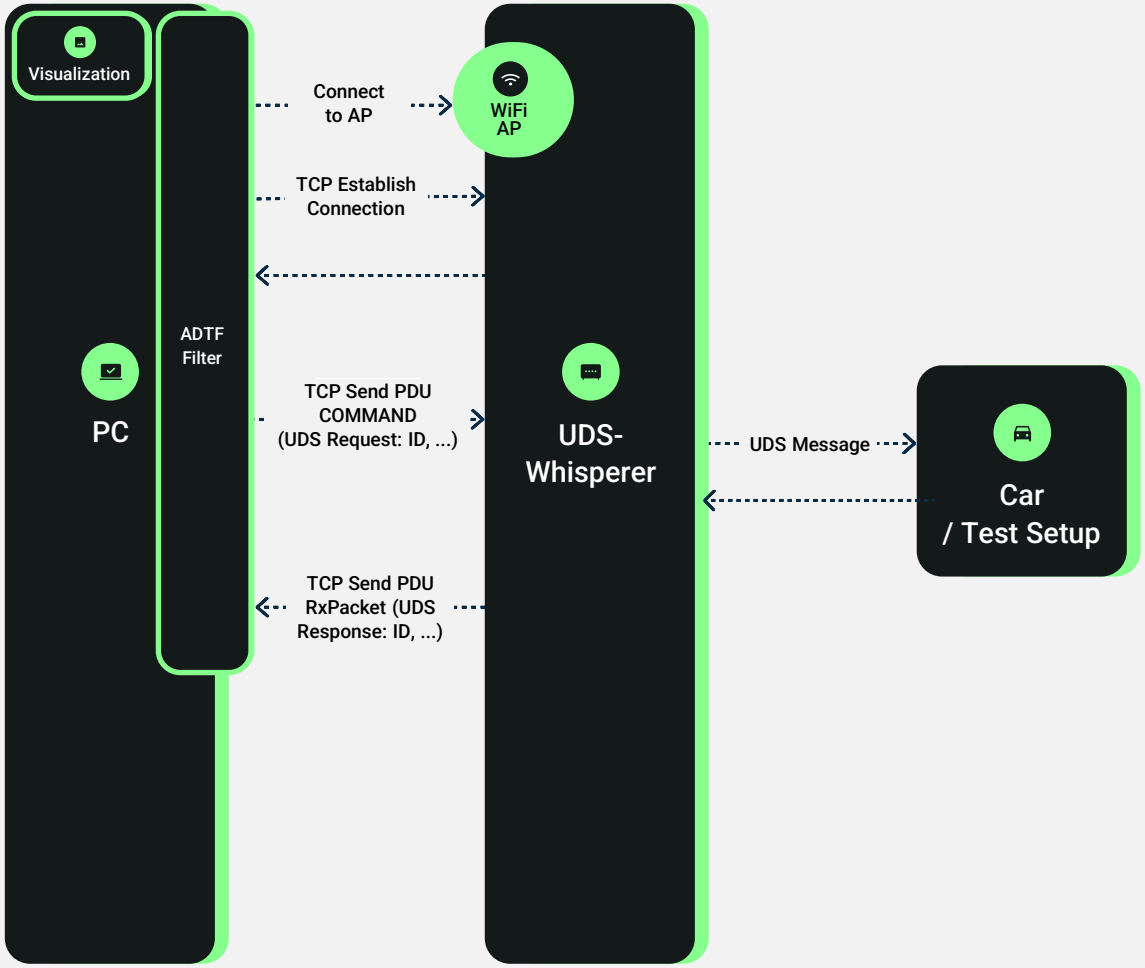
- UDS Whisperer contains a standardized log config
  - This config contains the parameters to query and the cycle time
- The device periodically queries the configured parameters (DIDs or PIDs) from the car and logs them to the non volatile memory (internal or SD card)
- Peer device can set a trigger 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



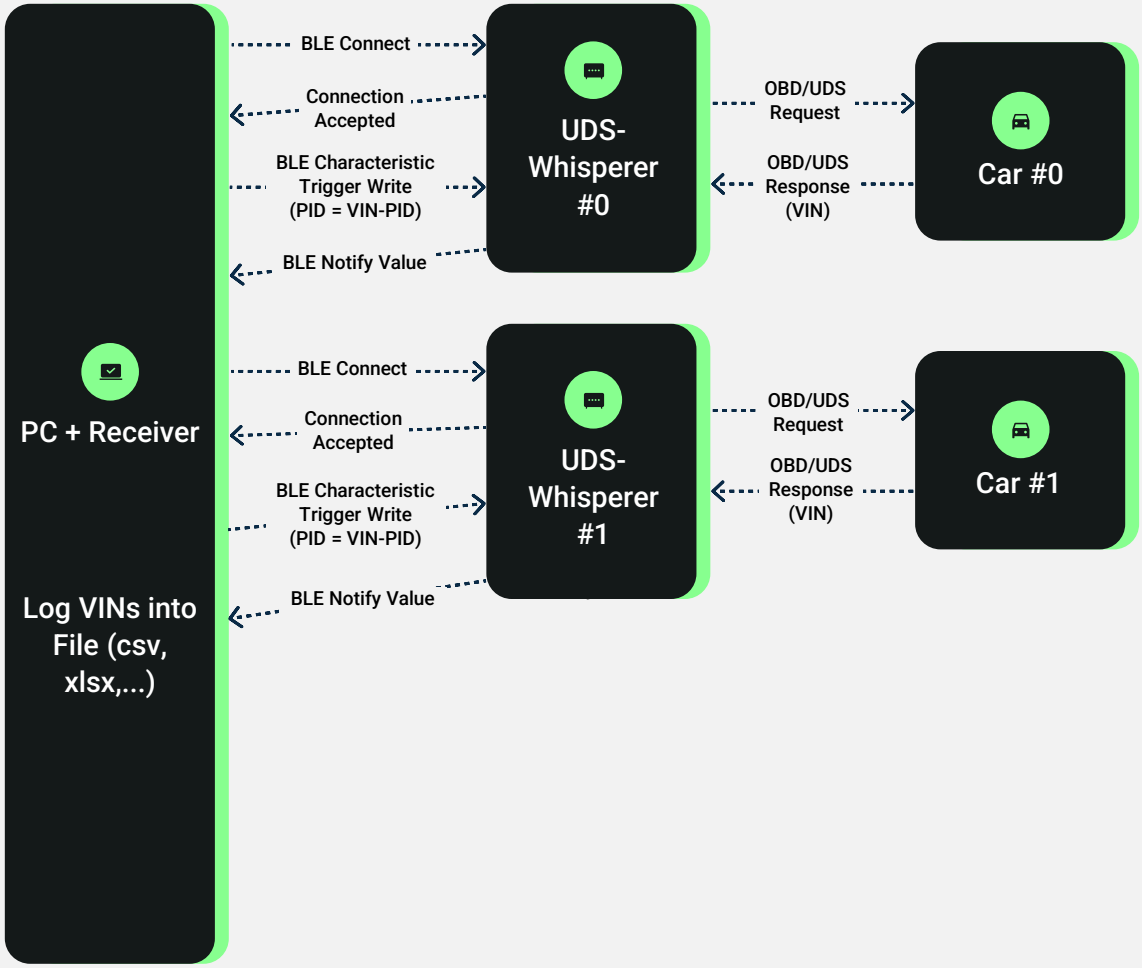
- UDS Whisperer is connected to a car or development setup
- A wireless access point (AP) is created by the device
- The user‘s PC connects to AP and receives an IP address via DHCP
- UDS Whisperer listens on e.g. port 20000 for incoming TCP traffic
- This traffic is converted to CAN frames
- In the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

UDS Bridge



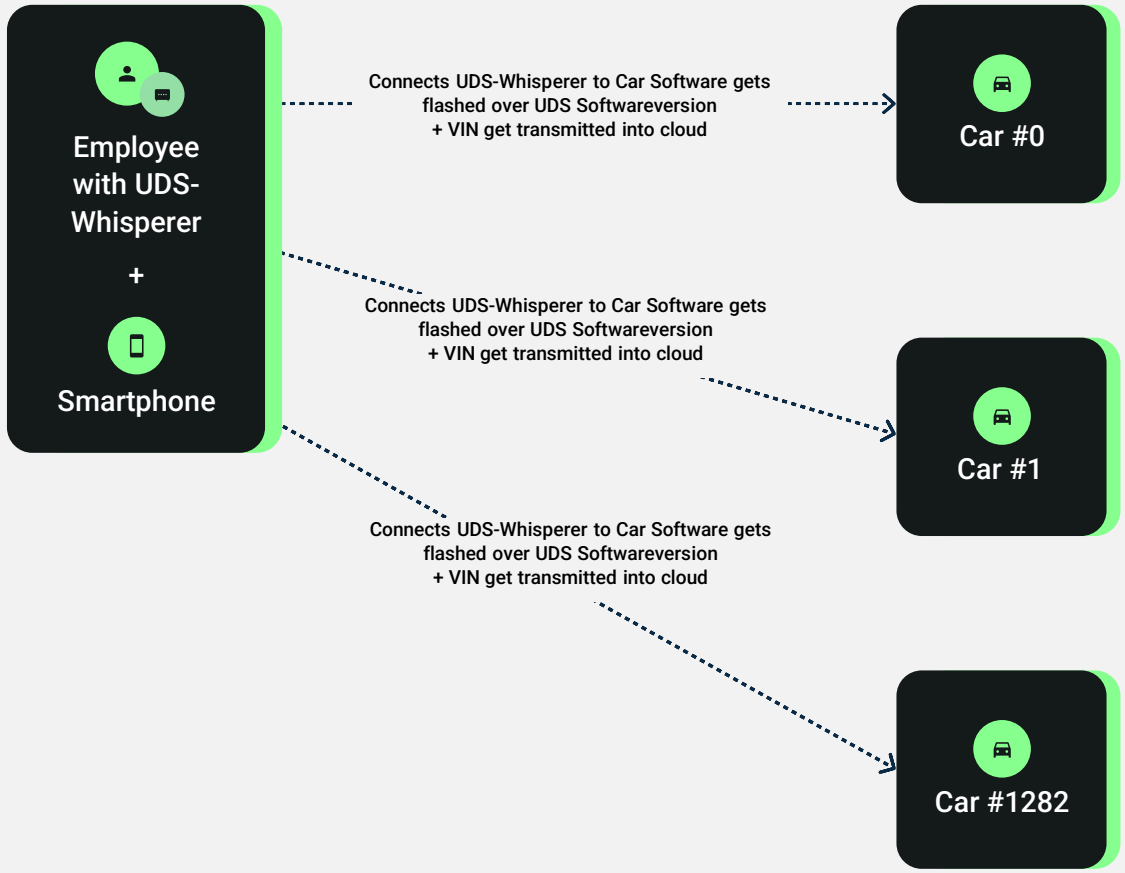
- Like the CAN bridge the UDS Whisperer and PC connect via WiFi
- Communication uses TCP
- Difference is that now instead of the CAN layer the UDS layer is used
- The created TCP stream towards the PC can be integrated as filters in CASSANDRA/ADTF or other tooling

Car Commissioning (Dog Tag)



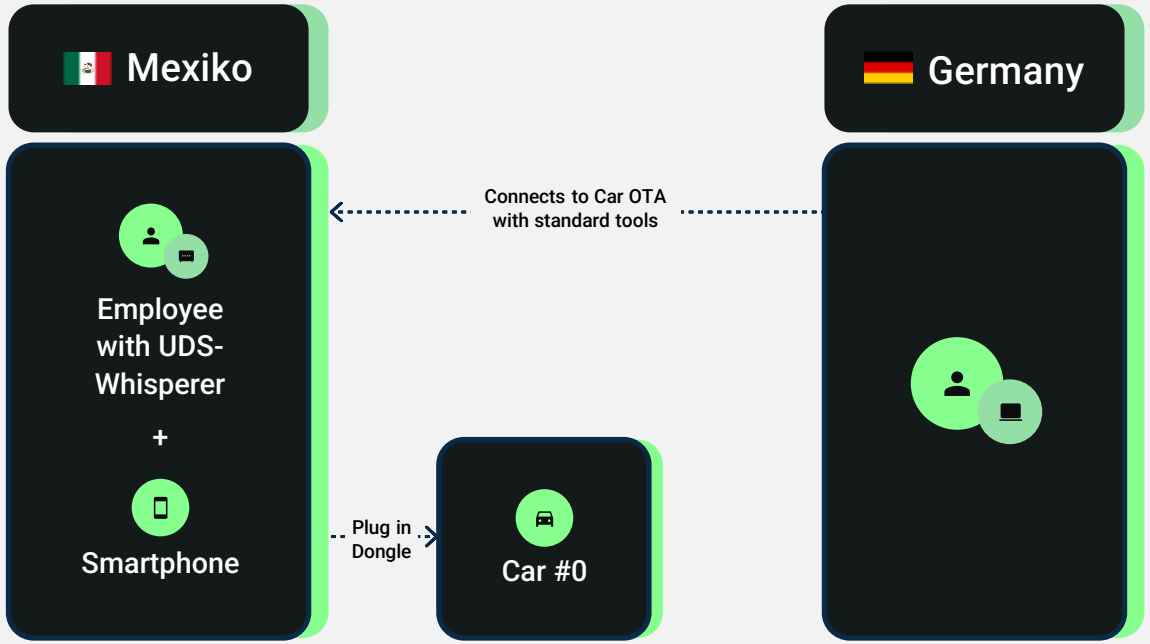
- When loading a large number of vehicles, it is cumbersome to document which vehicle with which VIN was loaded onto or unloaded from the ship and when
- UDS Whisperer can help
  - A driver plugs in the UDS Whisperer every time they get into a vehicle
  - The UDS Whisperer reads the VIN and optionally 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



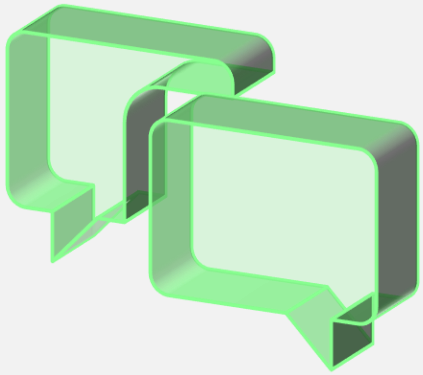
- Scenario: A large number of vehicles are to be flashed with a new software version or dataset.
- Problem: It is tedious and time-consuming to service each vehicle with a laptop (iDEX/ODIS/...) and navigate through menus. Even with scripting there is a significant delay
- The UDS Whisperer receives 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

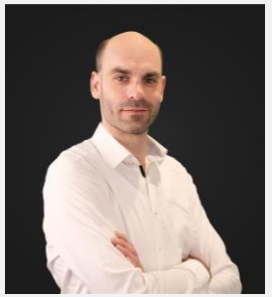


People Lead and Developer

Georg Hofstetter

georg.hofstetter@efs-techhub.com

+49 845839730042



People Lead

Lars Biermanski

lars.biermanski@efs-techhub.com

+49 1622844958