a simple intro to UDS

In this article

1. UDS message structure
2. UDS diagnostic services
3. UDS vs. OBD2 vs. WWH-OBD vs. ODBonUDS
4. CAN bus TP – Transport Protocol (ISO 15765-2)
5. Wireshark dissectors for UDS
6. Wireshark dissectors for OBD2
7. Wireshark dissectors for WWH-OBD
8. Wireshark dissectors for ODBonUDS
9. Wireshark dissectors for UDS
10. UDS diagnostic applications
11. Vehicle Diagnostic Emulator (VDE)
12. Downloads (VI)

Example: Nissan Leaf SoC

UDS and CAN OBD are complex topics, so to explain, we've done a case study to show how to read a Nissan Leaf's State of Charge (SoC) using UDS and CAN OBD. Both protocols are used by battery manufacturers from Nissan and LG.

In the example, we also use a CANdongle and CANtrack(Demo) to connect to the car, and we use Wireshark with our free plugins and CANdongle for ODB2 to add the ODB2 frames.

The modifications for TP, sequences and CANtrack signals are PRO decoders as our Kaitai Struct and Lua dissectors are not capable of handling such complex protocols.

DOWNLOADS CODE STARTS →

UDS message structure

UDS is a request/response protocol. In the illustration and in a pcap(ng) capture of an **UDS request frame** (using CAN as the bus):

A diagnostic UDS request on CAN contains various fields like:

#1 Protocol Control Information (PCI)

The first byte of the payload is the Protocol Control Information (PCI) byte. It specifies the type of diagnostic UDS request made on CAN bus. In short, the first nibble of the PCI byte defines the type of the message. In the example, we have a single frame request that do not fill within a single CAN frame. We will detail this more in the section on the CAN bus transport protocol (ISO TP).

#2 UDS Service ID (SID)

The one-byte unsigned service identifier (SID) specifies the type of diagnostic action or command. It also defines a specific UDS service. We refer to the UDS request message as a **service request**. The **service identifier** (SID) is the first byte of the payload. Note that the identifiers are split between ISO standard services (ISO 14229-1) and legacy OBD services. In the UDS, the SID is an unsigned byte. The response SID is the request SID+0x40.

You can find a summary of all standardized UDS services and SIDs. We will mainly focus on the UDS services defined in ISO 14229-1 and related to the ISO 27145 standard (WWH-OBD).

SID service identifier	Hex	Description
10	0x10	Diagnostic session control
11	0x11	ECU reset
14	0x14	Clear diagnostic information
19	0x19	Read DTC information
22	0x22	Read data by identifier
23	0x23	Read memory by address
24	0x24	Read scaling data by identifier
27	0x27	Security access
28	0x28	Communication control
2A	0x2A	Read data by periodic identifier
2C	0x2C	Dynamically define data identifier
2E	0x2E	Write data by identifier
2F	0x2F	Input output control by identifier
31	0x31	Routine control
34	0x34	Request download
35	0x35	Request upload
36	0x36	Transfer data
37	0x37	Request transfer exit
38	0x38	Request file transfer
3D	0x3D	Write memory by address
3E	0x3E	Tester present
83	0x83	Access timing parameter
84	0x84	Secured data transmission
85	0x85	Control DTC setting
86	0x86	Response on event
87	0x87	Link control
B9	0xB9	Diagnostic services 0xB9
BA	0xBA	Diagnostic services 0xBA
BB	0xBB	Diagnostic services 0xBB
BC	0xBC	Diagnostic services 0xBC
BD	0xBD	Diagnostic services 0xBD
BE	0xBE	Diagnostic services 0xBE
C3	0xC3	Diagnostic services 0xC3
C4	0xC4	Diagnostic services 0xC4
C5	0xC5	Diagnostic services 0xC5
C6	0xC6	Diagnostic services 0xC6
C7	0xC7	Diagnostic services 0xC7
C8	0xC8	Diagnostic services 0xC8
C9	0xC9	Diagnostic services 0xC9
CA	0xCA	Diagnostic services 0xCA
CB	0xCB	Diagnostic services 0xCB
CC	0xCC	Diagnostic services 0xCC
CD	0xCD	Diagnostic services 0xCD
CE	0xCE	Diagnostic services 0xCE
CF	0xCF	Diagnostic services 0xCF
D0	0xD0	Diagnostic services 0xD0
D1	0xD1	Diagnostic services 0xD1
D2	0xD2	Diagnostic services 0xD2
D3	0xD3	Diagnostic services 0xD3
D4	0xD4	Diagnostic services 0xD4
D5	0xD5	Diagnostic services 0xD5
D6	0xD6	Diagnostic services 0xD6
D7	0xD7	Diagnostic services 0xD7
D8	0xD8	Diagnostic services 0xD8
D9	0xD9	Diagnostic services 0xD9
DA	0xDA	Diagnostic services 0xDA
DB	0xDB	Diagnostic services 0xDB
DC	0xDC	Diagnostic services 0xDC
DD	0xDD	Diagnostic services 0xDD
DE	0xDE	Diagnostic services 0xDE
DF	0xDF	Diagnostic services 0xDF
E0	0xE0	Diagnostic services 0xE0
E1	0xE1	Diagnostic services 0xE1
E2	0xE2	Diagnostic services 0xE2
E3	0xE3	Diagnostic services 0xE3
E4	0xE4	Diagnostic services 0xE4
E5	0xE5	Diagnostic services 0xE5
E6	0xE6	Diagnostic services 0xE6
E7	0xE7	Diagnostic services 0xE7
E8	0xE8	Diagnostic services 0xE8
E9	0xE9	Diagnostic services 0xE9
EA	0xEA	Diagnostic services 0xEA
EB	0xEB	Diagnostic services 0xEB
EC	0xEC	Diagnostic services 0xEC
ED	0xED	Diagnostic services 0xED
EE	0xEE	Diagnostic services 0xEE
EF	0xEF	Diagnostic services 0xEF
F0	0xF0	Diagnostic services 0xF0
F1	0xF1	Diagnostic services 0xF1
F2	0xF2	Diagnostic services 0xF2
F3	0xF3	Diagnostic services 0xF3
F4	0xF4	Diagnostic services 0xF4
F5	0xF5	Diagnostic services 0xF5
F6	0xF6	Diagnostic services 0xF6
F7	0xF7	Diagnostic services 0xF7
F8	0xF8	Diagnostic services 0xF8
F9	0xF9	Diagnostic services 0xF9
FA	0xFA	Diagnostic services 0xFA
FB	0xFB	Diagnostic services 0xFB
FC	0xFC	Diagnostic services 0xFC
FD	0xFD	Diagnostic services 0xFD
FE	0xFE	Diagnostic services 0xFE
FF	0xFF	Diagnostic services 0xFF

#3 UDS Sub-function Byte

The **sub-function byte** is used in some UDS request frames as the second byte. It specifies the type of request within a service. Note that the sub-function byte is not used.

Generally, when a response is sent back to an ECU, the ECU may respond positively or negatively. In case the response is positive, the ECU response with a positive response message. In case the response is negative, the ECU sends a negative response message. The response SID is the request SID + 0x40.

The remaining 7 bits can be used to define up to 128 sub-function values. For example, when reading DTC information on the ISO 14229-1 Report Server Information, the sub-function can be `reportNumberOfDTCBySeverityMaskRecord` or `reportSeverityInformationOfDTC`.

If an ECU can positively or negatively ACK an request, we can provide a sub-function value.

#4 UDS Request Data Parameters – Incl. Data Identifier (DID)

An least UDS request frame's services, such as **Read Data by Identifier** (0x22), **Write Data by Identifier** (0x2E), and **Routine Control** (0x31), expect a parameter called **DIDs**. The DID serves as a predefined identifier for both requested/responses (similar to how the sub-function byte functions for UDS services).

For example, a request for reading data via a single DID would use one CAN-sized frame (8 bytes) with the PCI byte, the UDS service byte (0x22), and the DID's two bytes. The response would then include additional UDS data, adhering to the CAN bus's 8-byte limit.

We will look further into this in the section on how to record UDS diagnostic services.

Data identifiers can be proprietary and only known by OEMs, though some DIDs are standardized. This is for example the case for the Vehicle Information DIDs (0x2000-0x2FFF) and the Vehicle Identification Number (VIN) (0xF190). Note that requests refer to one of standardized DIDs only (0x01-0xFF).

Diagnostic service (Read Data by Identifier) - Data identifiers

Another example is service **Read Data by Identifier** (0x22). The request message for this service (0x22) expects a parameter (DID/PID). The DID serves as a predefined identifier for both requests/responses (similar to how the sub-function byte functions for UDS services).

DID - standardized data identifiers (0x01-0xFF)	
0x01	Boot software identification
0x02	application software identification
0x03	application data identification
0x04	boot fingerprint identification
0x05	programming fingerprint identification
0x06	application fingerprint identification
0x07	system supplier ECU software number
0x08	system supplier ECU software version
0x09	system supplier ECU application software number
0x0A	system supplier ECU application software version
0x0B	system supplier ECU application data number
0x0C	system supplier ECU application data version
0x0D	boot software fingerprint
0x0E	application software fingerprint
0x0F	application data fingerprint
0x10	active diagnostic session
0x11	vehicle manufacturer spare part number
0x12	vehicle manufacturer ECU software number
0x13	vehicle manufacturer ECU software version
0x14	system supplier identifier
0x15	ECU manufacturing date
0x16	ECU serial number
0x17	supported functional units
0x18	VMS master ECU software number
0x19	VMS master ECU software version
0x1A	boot software identification
0x1B	application software identification
0x1C	application data identification
0x1D	boot fingerprint identification
0x1E	programming fingerprint identification
0x1F	application fingerprint identification
0x20	boot software fingerprint
0x21	application software fingerprint
0x22	application data fingerprint
0x23	active diagnostic session
0x24	vehicle manufacturer spare part number
0x25	vehicle manufacturer ECU software number
0x26	vehicle manufacturer ECU software version
0x27	system supplier identifier
0x28	ECU manufacturing date
0x29	ECU serial number
0x2A	supported functional units
0x2B	VMS master ECU software number
0x2C	VMS master ECU software version
0x2D	boot software identification
0x2E	application software identification
0x2F	application data identification
0x30	boot fingerprint identification
0x31	programming fingerprint identification
0x32	application fingerprint identification
0x33	boot software fingerprint
0x34	application software fingerprint
0x35	application data fingerprint
0x36	active diagnostic session
0x37	vehicle manufacturer spare part number
0x38	vehicle manufacturer ECU software number
0x39	vehicle manufacturer ECU software version
0x3A	system supplier identifier
0x3B	ECU manufacturing date
0x3C	ECU serial number
0x3D	supported functional units
0x3E	VMS master ECU software number
0x3F	VMS master ECU software version
0x40	boot software identification
0x41	application software identification
0x42	application data identification
0x43	boot fingerprint identification
0x44	programming fingerprint identification
0x45	application fingerprint identification
0x46	boot software fingerprint
0x47	application software fingerprint
0x48	application data fingerprint
0x49	active diagnostic session
0x4A	vehicle manufacturer spare part number
0x4B	vehicle manufacturer ECU software number
0x4C	vehicle manufacturer ECU software version
0x4D	system supplier identifier
0x4E	ECU manufacturing date
0x4F	ECU serial number
0x50	supported functional units
0x51	VMS master ECU software number
0x52	VMS master ECU software version
0x53	boot software identification
0x54	application software identification
0x55	application data identification
0x56	boot fingerprint identification
0x57	programming fingerprint identification
0x58	application fingerprint identification
0x59	boot software fingerprint
0x5A	application software fingerprint
0x5B	application data fingerprint
0x5C	active diagnostic session
0x5D	vehicle manufacturer spare part number
0x5E	vehicle manufacturer ECU software number
0x5F	vehicle manufacturer ECU software version
0x60	system supplier identifier
0x61	ECU manufacturing date
0x62	ECU serial number
0x63	supported functional units
0x64	VMS master ECU software number
0x65	VMS master ECU software version
0x66	boot software identification
0x67	application software identification
0x68	application data identification
0x69	boot fingerprint identification
0x6A	programming fingerprint identification
0x6B	application fingerprint identification
0x6C	boot software fingerprint
0x6D	application software fingerprint
0x6E	application data fingerprint
0x6F	active diagnostic session
0x70	vehicle manufacturer spare part number
0x71	vehicle manufacturer ECU software number
0x72	vehicle manufacturer ECU software version
0x73	system supplier identifier
0x74	ECU manufacturing date
0x75	ECU serial number
0x76	supported functional units
0x77	VMS master ECU software number
0x78	VMS master ECU software version
0x79	boot software identification
0x7A	application software identification
0x7B	application data identification
0x7C	boot fingerprint identification
0x7D	programming fingerprint identification
0x7E	application fingerprint identification
0x7F	boot software fingerprint
0x80	application software fingerprint
0x81	application data fingerprint
0x82	active diagnostic session
0x83	vehicle manufacturer spare part number
0x84	vehicle manufacturer ECU software number
0x85	vehicle manufacturer ECU software version
0x86	system supplier identifier
0x87	ECU manufacturing date
0x88	ECU serial number
0x89	supported functional units
0x8A	VMS master ECU software number
0x8B	VMS master ECU software version
0x8C	boot software identification
0x8D	application software identification
0x8E	application data identification
0x8F	boot fingerprint identification
0x90	programming fingerprint identification
0x91	application fingerprint identification
0x92	boot software fingerprint
0x93	application software fingerprint
0x94	application data fingerprint
0x95	active diagnostic session
0x96	vehicle manufacturer spare part number
0x97	vehicle manufacturer ECU software number
0x98	vehicle manufacturer ECU software version
0x99	system supplier identifier
0x9A	ECU manufacturing date
0x9B	ECU serial number
0x9C	supported functional units
0x9D	VMS master ECU software number
0x9E	VMS master ECU software version
0x9F	boot software identification
0xA0	application software identification
0xA1	application data identification
0xA2	boot fingerprint identification
0xA3	programming fingerprint identification
0xA4	application fingerprint identification
0xA5	boot software fingerprint
0xA6	application software fingerprint
0xA7	application data fingerprint
0xA8	active diagnostic session
0xA9	vehicle manufacturer spare part number
0xAA	vehicle manufacturer ECU software number
0xAB	vehicle manufacturer ECU software version
0xAC	system supplier identifier
0xAD	ECU manufacturing date
0xAE	ECU serial number
0xAF	supported functional units
0xB0	VMS master ECU software number
0xB1	VMS master ECU software version
0xB2	boot software identification
0xB3	application software identification
0xB4	application data identification
0xB5	boot fingerprint identification
0xB6	programming fingerprint identification
0xB7	application fingerprint identification
0xB8	boot software fingerprint
0xB9	application software fingerprint
0xBA	application data fingerprint
0xBB	active diagnostic session
0xBC	vehicle manufacturer spare part number
0xBD	vehicle manufacturer ECU software number
0xBE	vehicle manufacturer ECU software version
0xBF	system supplier identifier
0xC0	ECU manufacturing date
0xC1	ECU serial number
0xC2	supported functional units
0xC3	VMS master ECU software number
0xC4	VMS master ECU software version
0xC5	boot software identification
0xC6	application software identification
0xC7	application data identification
0xC8	boot fingerprint identification
0xC9	programming fingerprint identification
0xCA	application fingerprint identification
0xCB	boot software fingerprint
0xCC	application software fingerprint
0xCD	application data fingerprint
0xCE	active diagnostic session
0xCF	vehicle manufacturer spare part number
0xD0	vehicle manufacturer ECU software number
0xD1	vehicle manufacturer ECU software version
0xD2	system supplier identifier
0xD3	ECU manufacturing date
0xD4	ECU serial number
0xD5	supported functional units
0xD6	VMS master ECU software number
0xD7	VMS master ECU software version
0xD8	boot software identification
0xD9	application software identification
0xDA	application data identification
0xDB	boot fingerprint identification
0xDC	programming fingerprint identification
0xDD	application fingerprint identification
0xDE	boot software fingerprint
0xDF	application software fingerprint
0xE0	application data fingerprint
0xE1	active diagnostic session
0xE2	vehicle manufacturer spare part number
0xE3	vehicle manufacturer ECU software number
0xE4	vehicle manufacturer ECU software version
0xE5	system supplier identifier
0xE6	ECU manufacturing date
0xE7	ECU serial number
0xE8	supported functional units
0xE9	VMS master ECU software number
0xEA	VMS master ECU software version
0xEB	boot software identification
0xEC	application software identification
0xED	application data identification
0xEE	boot fingerprint identification
0xEF	programming fingerprint identification
0xF0	application fingerprint identification
0xF1	boot software fingerprint
0xF2	application software fingerprint
0xF3	application data fingerprint
0xF4	active diagnostic session
0xF5	vehicle manufacturer spare part number
0xF6	vehicle manufacturer ECU software number
0xF7	vehicle manufacturer ECU software version
0xF8	system supplier identifier
0xF9	ECU manufacturing date
0xFA	ECU serial number
0xFB	supported functional units
0xFC	VMS master ECU software number
0xFD	VMS master ECU software version
0xFE	boot software identification
0xFF	application software identification

Positive vs. negative UDS responses

When an ECU responds positively to an UDS request, the response frame is structured with similar elements as the request frame. For instance, the response SID is the request SID plus 0x40, and the sub-function byte and data identifiers (if applicable) are included in the response to ensure clarity. This structure mirrors the request format, facilitating easy interpretation of the response by the requesting tester.

However, in some cases an ECU may provide a negative response to an UDS request. For example if the service is not supported. A negative response is structured as follows (CAN bus example):

Positive / Negative Response Code (NRC)

Below are a few detail the negative response frame with focus on the NRC:

- The 1st byte is the SID field
- The 2nd byte is the Negative Response Code (NC, 0x7F)
- The 3rd byte is the SID of the rejected request
- The 4th byte is the Negative Response Code (NRC)

In the negative UDS response the NRC provides information regarding the cause of the rejection as per the table below.

Overview of UDS standards & concepts

UDS refers to a large number of standard documents, meaning it can be a bit overwhelming. To give you an overview, we provide a high-level explanation of the most relevant terms/standards/concepts that fall under the UDS umbrella.

What is the UDS protocol?

The roots of UDS (UDS) lie in an old, a great standard, known as ISO 14229-1 and ISO 14229-2 (UDS on CAN). Its first edition was published in 2006 and it was developed by the International Organization for Standardization (ISO).

In practice, WWH-OBD has been required in EU since 2014 for all new type approved vehicles and since 2015 for all new registrations. Some US OEMs have also adopted it, but it is not mandatory in the US.

Beyond this, mandated, the roll-out of WWH-OBD has been limited. That challenge is about to change. In US, it is now accepted by CFR/EPA in 40 CFR Part 86. Starting 2026 model year, OEMs can optionally use WWH-OBD (UDS) instead of legacy OBD (J1979).

What is UNECE R155 / ISO 21434?

Similar to the UNECE R156 (see later) UNECE R155 (UNECE R155 CS and ISO/SAE DIS 21434) is a mandatory standard for the EU. The next generation of OBD/ Data regime for type approvals.

Specifically, WWH-OBD (ISO 27145) has been adopted in the EU as the mandatory way to fulfill the requirements of UNECE R155 CS. Vehicle type approvals (after 2024-01-01) in EU vehicles in the scope of UNECE R155 CS must be able to support WWH-OBD (ISO 27145) and the UDS (ISO 14229). The new OBD (J1979-2 standard is also referred to as "UDS for OBD") is not required for EU type approvals (after 2024-01-01), where ECUs are allowed to support either UDS or OBD/KWP2000. From 2027, OBD/UDS will be a mandatory requirement for all new EU type approvals.

Looking ahead: OBD/UDS vs. DiMoLinO

Data is the lifeblood of modern transportation. In particular, it is generally not known what data decoders OEMs use in order to interpret raw data of diagnostic data stored within, where and how the data is transferred. OEMs, Tier1s and others are aligned on the next section. It is not known how to decode the response payload.

Although, some standards are more open than others, most of them do not provide open access databases for specific parameters. Instead, access is based on licensing agreements. The information can be very lucrative and the commercialization is taken from non-audit perspective.

Regarding the CAN DLC used

In this case we use the CAN DLC(8 bytes). Legacy data field is capped to 8 bytes, which is equal with CAN DLC(8). This is known as a physically addressed request.

In contrast, functionally addressed request uses a CAN DLC(8) for OBD/UDS* in heavy duty vehicles.

Generally, requests/responses CAN DLC are governed as per the table below and you can identify the physical request (0 x7DF) and functional request (0 x7E0). Physical request is typically used when a tester is talking to a single ECU (0 x7E8) response. Up to 8 ECUs, the physical request & supporting ECU response (0 x7E8 to 0 x7E8 + 7) are supported.

Since you can receive what seems to be a single response, but it could be something not a functional response only the first response is valid. Functional request is typically used when a tester is talking to several ECUs. If you, say perform an analysis of functional response, you should, in theory, be able to see the response from all ECUs supporting functional request and using the information to troubleshoot your configuration.

On a separate note, ISO 15765-4 states that extended addressing is optional for vehicles. Extended addressing (29-bit CAN) OBD/UDS is no longer in use, but it is still mandatory for all vehicles that are sold in the specific Keyword Protocol 2000 variant (ISO 14230-4) and Keyword Protocol 2000 (ISO 14230-4).

Yes, CAN (Protocols) the Standardized CAN frame (normally) is a big deal. However, we generally recommend to use UDS (Unified Diagnostic Services) instead of OBD.

CAN (Controller Area Network) is a bus and not the Standardized CAN frame is a case of **standard multiplexing**. We provide an example using ISO-TP, the first Transport Protocol, on the Data of Standard CAN frame. Yes, the Data of Standard CAN frame is capped to 8 bytes.

Yes, ISO-TP (ISO 15765-2) allows transmitting PDU (Protocol Data Units) of multi-frame UDS responses - and our UDS examples responding to these blocks use the Standard frame payloads here.

Regarding Vehicle Identification Number (VIN)

Vehicle Identification Number (VIN) is a unique identifier for a vehicle. The VIN was introduced in 1954 and has been used in various forms since then. It is now standardized worldwide and is required in most vehicles manufactured after 1980. The VIN is used for various purposes, including tracking vehicle ownership, registration, and insurance, and for identifying specific vehicles in recalls and other safety-related issues. The VIN can also be used to access vehicle-specific information, such as its manufacturer, model, and year of manufacture, and to access other information, such as its warranty and maintenance history.

The VIN is a 17-character alphanumeric code that can be retrieved via various methods, including OBD-II, UDS, and proprietary protocols. The specific method used to retrieve the VIN may vary depending on the vehicle manufacturer and the specific diagnostic tool or software being used. The VIN can be a useful tool for technicians and other professionals who need to quickly and accurately identify a vehicle and access its relevant information.

Regarding Diagnostic Trouble Codes (DTC)

Diagnostic Trouble Codes (DTC) are a standardized set of codes used in the automotive industry to diagnose and troubleshoot issues with a vehicle's engine, transmission, and other systems. These codes are stored in the vehicle's ECU (Engine Control Unit) and can be accessed using a diagnostic tool or software that supports the OBD-II protocol. The DTC consists of a five-character code that includes a letter and four numbers, which provides information about the specific fault or issue that has been detected by the vehicle's sensors and other diagnostic systems. The first character of the DTC indicates the system or module that is affected by the fault, such as the engine, transmission, or chassis. The remaining four characters provide more specific information about the fault, such as the specific sensor or component that is malfunctioning or the type of fault that has been detected. The DTC can be used by technicians and other professionals to quickly identify the root cause of a vehicle's issue and to develop a repair plan to address the fault and restore the vehicle to its original condition.

Regarding Data Logging

Data logging can be a powerful tool for monitoring and analyzing the performance of a vehicle's various systems and components. By collecting data on various parameters such as engine speed, temperature, fuel consumption, and emissions, data logging can provide valuable insights into the vehicle's performance and identify any issues or areas for improvement.

There are many different types of data that can be logged, including sensor data, diagnostic trouble codes, and GPS data. The specific data that is logged will depend on the specific goals and objectives of the data logging project, and the capabilities of the data logging tool or software being used.

Data logging can be performed using a variety of tools and methods, including OBD-II scanners, proprietary diagnostic tools, and custom-built data logging devices. The choice of tool or method will depend on the specific requirements of the data logging project, such as the type of data to be logged, the number of vehicles to be monitored, and the budget available for the project.

Once the data has been collected, it can be analyzed using various software tools and techniques to identify patterns and trends in the data, and to draw conclusions about the performance of the vehicle and its various systems and components. This analysis can be used to inform decisions about maintenance and repair, to optimize the vehicle's performance, and to develop new products and services that meet the needs of customers.

Regarding Security

Security is a critical consideration when it comes to data logging and analysis, especially when dealing with sensitive or proprietary information. There are several steps that can be taken to ensure the security of data logging and analysis, including encryption, access control, and data anonymization. Encryption can be used to protect the data while it is being stored or transmitted, access control can be used to limit the number of people who have access to the data, and data anonymization can be used to remove any personally identifiable information from the data before it is analyzed.

It is important to consider the specific requirements and vulnerabilities of the data logging and analysis system when implementing security measures, and to regularly review and update the security measures as new threats and vulnerabilities emerge.

Regarding Cybersecurity

Cybersecurity is a critical consideration when it comes to data logging and analysis, especially when dealing with sensitive or proprietary information. There are several steps that can be taken to ensure the cybersecurity of data logging and analysis, including encryption, access control, and data anonymization. Encryption can be used to protect the data while it is being stored or transmitted, access control can be used to limit the number of people who have access to the data, and data anonymization can be used to remove any personally identifiable information from the data before it is analyzed.

It is important to consider the specific cybersecurity threats and vulnerabilities that may be relevant to the data logging and analysis system, and to implement appropriate cybersecurity measures to mitigate these risks. This may include conducting regular cybersecurity assessments, implementing intrusion detection and prevention systems, and training employees on cybersecurity best practices.

Regarding Telematics

Telematics is a technology that combines telecommunications and vehicle diagnostics to provide real-time data on a vehicle's location, performance, and maintenance needs. This data can be used for various purposes, such as fleet management, insurance, and vehicle maintenance. Telematics can be used to track the location of vehicles in real-time, monitor their performance and fuel consumption, and provide alerts for maintenance and repairs. The data collected by telematics can be used to improve the efficiency and safety of vehicles, reduce costs for fleet management companies, and provide valuable insights for insurance companies and vehicle manufacturers.

Regarding Reverse Engineering

Reverse engineering is the process of analyzing and deconstructing a product or system to understand its design, functionality, and underlying technology. In the context of data logging and analysis, reverse engineering can be used to understand the proprietary protocols and data formats used by vehicle manufacturers and other companies, and to develop tools and software that can interpret and analyze the data collected by these systems.

However, reverse engineering can also be a legally and ethically complex issue, depending on the specific laws and regulations in the jurisdiction where the reverse engineering is being performed, and the intellectual property rights of the manufacturer or other companies whose products or systems are being reverse engineered. It is important to consider the legal and ethical implications of reverse engineering before engaging in any such activity, and to seek advice from qualified legal and ethical experts if necessary.

Regarding Machine Learning (ML) and Artificial Intelligence (AI)

Machine Learning (ML) and Artificial Intelligence (AI) are powerful technologies that can be used to analyze and interpret data from various sources, including data logging and analysis. ML and AI can be used to identify patterns and trends in the data, predict future outcomes, and provide recommendations for optimizing vehicle performance, reducing costs, and improving safety and reliability.

However, ML and AI also raise ethical and legal concerns, such as bias in the data and algorithms, lack of transparency and accountability in the decision-making process, and potential job displacement. It is important to consider these issues and to develop and implement ML and AI systems in a responsible and ethical manner, with appropriate oversight and safeguards to ensure that the benefits of these technologies are shared fairly and do not exacerbate existing inequalities or create new risks and vulnerabilities.

Regarding Cloud Computing

Cloud computing is a technology that allows data to be stored and accessed over the internet, rather than on local servers or devices. In the context of data logging and analysis, cloud computing can be used to store and process large amounts of data collected from vehicles and other sources, and to provide access to the data and analysis tools from anywhere with an internet connection.

Cloud computing can provide many benefits for data logging and analysis, including scalability, flexibility, and cost-effectiveness. It can also provide access to advanced analytics and machine learning tools that may be difficult or expensive to implement on-premise. However, it also raises concerns about data security and privacy, and it is important to carefully evaluate the risks and benefits of using cloud computing for data logging and analysis, and to implement appropriate safeguards and controls to protect the data and ensure compliance with relevant regulations and standards.

Regarding Over-the-Air (OTA) Updates

Over-the-Air (OTA) updates are a method of wirelessly delivering software updates and data to vehicles and other connected devices, such as smartphones and tablets. In the context of data logging and analysis, OTA updates can be used to deploy new data collection and analysis tools, update existing software and firmware, and provide access to new features and functionality.

OTA updates can be a powerful tool for improving the efficiency and effectiveness of data logging and analysis, by allowing for the rapid deployment of new tools and updates to the data collection and analysis infrastructure, and by enabling the collection of data from a wide range of vehicles and other devices in real-time. However, OTA updates also raise concerns about data security and privacy, and it is important to carefully evaluate the risks and benefits of using OTA updates for data logging and analysis, and to implement appropriate safeguards and controls to protect the data and ensure compliance with relevant regulations and standards.

Regarding Telematics Control Unit (TCU)

A Telematics Control Unit (TCU) is a hardware component that is installed in a vehicle to provide telematics services, such as GPS tracking, cellular connectivity, and remote diagnostics. The TCU is typically connected to the vehicle's CAN bus and other electronic systems, and can collect data from various sensors and components in the vehicle, such as the engine, transmission, and brakes. The data collected by the TCU can be transmitted to a cloud-based server or platform for analysis and storage, and can be accessed by authorized users through a web interface or mobile app.

The TCU can be used for various purposes, such as fleet management, insurance telematics, and vehicle diagnostics and maintenance. In the context of data logging and analysis, the TCU can be used to collect data on various parameters, such as engine speed, temperature, fuel consumption, and emissions, and to provide real-time alerts and notifications to fleet managers or drivers when issues or anomalies are detected. The data collected by the TCU can also be used to develop predictive maintenance models and to optimize vehicle performance and reduce costs for fleet management companies.

Regarding Vehicle Speed Limiter (VSL)

Vehicle Speed Limiter (VSL) is a technology that restricts the maximum speed at which a vehicle can travel. In the context of data logging and analysis, VSL can be used to collect data on vehicle speed and to enforce speed limits in real-time. The data collected by VSL can be used to monitor driver behavior and to identify instances of speeding or other unsafe driving behaviors.

However, the use of VSL also raises concerns about privacy and civil liberties, as it can be used to track the movements and behavior of individuals and to enforce speed limits in a manner that may be perceived as intrusive or discriminatory. It is important to consider the legal and ethical implications of VSL and to ensure that its use is transparent, fair, and respectful of individuals' rights and freedoms.

Regarding Electronic Stability Control (ESC)

Electronic Stability Control (ESC) is a technology that uses sensors and actuators to monitor and control the vehicle's stability and traction, and to prevent skidding and loss of control in dangerous situations. The data collected by ESC can be used in data logging and analysis to identify instances of loss of control or near-misses, and to develop predictive models and algorithms that can anticipate and prevent such incidents.

However, the use of ESC also raises concerns about privacy and civil liberties, as it can be used to track the movements and behavior of individuals and to enforce speed limits in a manner that may be perceived as intrusive or discriminatory. It is important to consider the legal and ethical implications of ESC and to ensure that its use is transparent, fair, and respectful of individuals' rights and freedoms.

Regarding Advanced Driver Assistance Systems (ADAS)

Advanced Driver Assistance Systems (ADAS) are a set of technologies that use sensors, cameras, and other data collection devices to provide drivers with real-time information and alerts about the vehicle's surroundings and potential hazards on the road. The data collected by ADAS can be used in data logging and analysis to identify patterns and trends in driver behavior and vehicle performance, and to develop predictive models and algorithms that can anticipate and prevent accidents and other incidents.

However, the use of ADAS also raises concerns about privacy and civil liberties, as it can be used to track the movements and behavior of individuals and to collect sensitive personal information. It is important to consider the legal and ethical implications of ADAS and to ensure that its use is transparent, fair, and respectful of individuals' rights and freedoms.

Regarding Artificial Intelligence of Things (AIoT)

Artificial Intelligence of Things (AIoT) is a technology that combines artificial intelligence (AI) and the Internet of Things (IoT) to create smart, connected devices that can collect, analyze, and act on data in real-time. The data collected by AIoT devices can be used in data logging and analysis to identify patterns and trends in vehicle performance and driver behavior, and to develop predictive models and algorithms that can anticipate and prevent accidents and other incidents.

However, the use of AIoT also raises concerns about privacy and civil liberties, as it can be used to track the movements and behavior of individuals and to collect sensitive personal information. It is important to consider the legal and ethical implications of AIoT and to ensure that its use is transparent, fair, and respectful of individuals' rights and freedoms.

Regarding Vehicle-to-Everything (V2X)

Vehicle-to-Everything (V2X) is a technology that allows vehicles to communicate with other vehicles, infrastructure, and pedestrians in real-time, using various communication protocols and networks. The data collected by V2X can be used in data logging and analysis to identify patterns and trends in traffic flow and congestion, and to develop predictive models and algorithms that can anticipate and prevent accidents and other incidents.

However, the use of V2X also raises concerns about privacy and civil liberties, as it can be used to track the movements and behavior of individuals and to collect sensitive personal information. It is important to consider the legal and ethical implications of V2X and to ensure that its use is transparent, fair, and respectful of individuals' rights and freedoms.

Regarding Electronic Data Recorder (EDR)

Electronic Data Recorder (EDR) is a device that records data from various sensors and components in a vehicle, such as the engine, transmission, and brakes, in the event of a crash or other incident. The data collected by EDR can be used in data logging and analysis to reconstruct the events leading up to the incident and to identify the root cause of the issue.

However, the use of EDR also raises concerns about privacy and civil liberties, as it can be used to collect sensitive personal information and to track the movements and behavior of individuals. It is important to consider the legal and ethical implications of EDR and to ensure that its use is transparent, fair, and respectful of individuals' rights and freedoms.

Regarding Digital Twin

Digital Twin is a virtual representation of a physical asset or system, such as a vehicle or a manufacturing plant, that can be used to simulate and analyze the performance and behavior of the asset or system in real-time. The data collected by Digital Twin can be used in data logging and analysis to identify patterns and trends in performance and behavior, and to develop predictive models and algorithms that can anticipate and prevent issues and optimize performance.

However, the use of Digital Twin also raises concerns about data security and privacy, and it is important to carefully evaluate the risks and benefits of using Digital Twin for data logging and analysis, and to implement appropriate safeguards and controls to protect the data and ensure compliance with relevant regulations and standards.

How to remove/add one decoder

Will be once shared how you can remove/add one decoder to/from a given configuration.

How to remove/add one decoder multi-frame UDS data?

Will be once shared how you can remove/add one decoder to/from a given configuration multi-frame UDS data.

Example 1: Record single frame UDS data (Speed via WWH-OBD)

To keep it simple we're using a simple and well-used value - speed. For example, to extract data in WWH-OBD is based on UDS - and is specified in ISO 27145-3 and ISO 14229-1.

As part of this example, we're using the most well-adopted parameters (the speed). While, fuel level etc in a way similar to how we can use PIDs in legacy OBD (SAE J1979 standard). In UDS, we use DIDs (and not OBD/SAE J1979 standard). For details, however, some WWH-OBD DIDs (J1979-2), the OBD/PIDs are based on SAE J1979 and SAE J1979-2.

For the UDS/WWH-OBD DIDs (Data Identifiers) **the Fixed Length** DIDs are:

- The speed DID has the Fixed Length of 0x F405
- The Fuel level DID has the Fixed Length of 0x F419 (Fuel All Types DATA)
- The Fuel type is the requested PID for Short Data (Is identified by the 0x 51)
- The Min and Max values are Data dependent (DID/SAE)
- The Fuel type is the requested PID for Short Data (Is identified by the 0x 51)

Following the PID, the actual data is provided. For example, the Fuel type is:

- The initial 4 bits equal the Frame type (0x for FF)
- The next 4 bits specifies the length of the following Data (0x 1 for SF)
- The 8 bits specifies the Fuel type (0x 52 for Electric Fuel)

Since we are using a single frame response, the Data is identified by the 0x 04 (0x04).

Following the PID, the actual data is provided. For example, the Fuel type is:

- The initial 4 bits equal the Frame type (0x for FF)
- The next 12 bits specifies the length of the following Data (0x 007 for SF)
- The remaining 7 bytes of each CF contain the rest of the payload (0x 62)

Regarding proprietary UDS data

The vehicle manufacturers (OEMs) are proprietary. In particular, it is generally not known what data decoders OEMs use in order to interpret raw data of diagnostic data stored within, where and how the data is transferred. OEMs, Tier1s and others are aligned on the next section. It is not known how to decode the response payload.

Although, some standards are more open than others, most of them do not provide open access databases for specific parameters. Instead, access is based on licensing agreements. The information can be very lucrative and the commercialization is taken from non-audit perspective.

Regarding the CAN DLC used

In this case we use the CAN DLC(8 bytes). Legacy data field is capped to 8 bytes, which is equal with CAN DLC(8). This is known as a physically addressed request.

In contrast, functionally addressed request uses a CAN DLC(8) for OBD/UDS* in heavy duty vehicles.

Generally, requests/response CAN DLC are governed as per the table below and you can identify the physical request (0 x7DF) and functional request (0 x7E0). Physical request is typically used when a tester is talking to a single ECU (0 x7E8) response. Up to 8 ECUs, the physical request & supporting ECU response (0 x7E8 to 0 x7E8 + 7) are supported.

Since you can receive what seems to be a single response, but it could be something not a functional response only the first response is valid. Functional request is typically used when a tester is talking to several ECUs. If you, say perform an analysis of functional response, you should, in theory, be able to see the response from all ECUs supporting functional request and using the information to troubleshoot your configuration.

On a separate note, ISO 15765-4 states that extended addressing is optional for vehicles. Extended addressing (29-bit CAN) OBD/UDS is no longer in use, but it is still mandatory for all vehicles that are sold in the specific Keyword Protocol 2000 variant (ISO 14230-4) and Keyword Protocol 2000 (ISO 14230-4).

Yes, CAN (Protocols) the Standardized CAN frame (normally) is a big deal. However, we generally recommend to use UDS (Unified Diagnostic Services) instead of OBD.

CAN (Controller Area Network) is a bus and not the Standardized CAN frame is a case of **standard multiplexing**. We provide an example using ISO-TP, the first Transport Protocol, on the Data of Standard CAN frame. Yes, the Data of Standard CAN frame is capped to 8 bytes.

Yes, ISO-TP (ISO 15765-2) allows transmitting PDU (Protocol Data Units) of multi-frame UDS responses - and our UDS examples responding to these blocks use the Standard frame payloads here.

Regarding Vehicle Identification Number (VIN)

Vehicle Identification Number (VIN) is a unique identifier for a vehicle. The VIN was introduced in 1954 and has been used in various forms since then. It is now standardized worldwide and is required in most vehicles manufactured after 1980. The VIN is used for various purposes, including tracking vehicle ownership, registration, and insurance, and for identifying specific vehicles in recalls and other safety-related issues. The VIN can also be used to access vehicle-specific information, such as its manufacturer, model, and year of manufacture, and to access other information, such as its warranty and maintenance history.

The VIN is a 17-character alphanumeric code that can be retrieved via various methods, including OBD-II, UDS, and proprietary protocols. The specific method used to retrieve the VIN may vary depending on the vehicle manufacturer and the specific diagnostic tool or software being used. The VIN can be a useful tool for technicians and other professionals who need to quickly and accurately identify a vehicle and access its relevant information.

Example 2: Record the Vehicle Identification Number (VIN)

The **Vehicle Identification Number** (VIN) is a unique identifier for a vehicle. The VIN was introduced in 1954 and has been used in various forms since then. It is now standardized worldwide and is required in most vehicles manufactured after 1980. The VIN is used for various purposes, including tracking vehicle ownership, registration, and insurance, and for identifying specific vehicles in recalls and other safety-related issues. The VIN can also be used to access vehicle-specific information, such as its manufacturer, model, and year of manufacture, and to access other information, such as its warranty and maintenance history.

VIN: Vehicle Identification Number

2T3RFREV7DW108177

A VIN consists of 17 digits (characters and can be retrieved on request from a vehicle. This is useful e.g. into logging or diagnostic applications. Some manufacturers provide it in an easy-to-read way. OBD-II is one way.

Some OEMs can be somewhat from VDI or ACES for tables, unless OEM is ISO/SAE Partner/Leader etc.

For example, the type 9a01 corresponds to the Note "2".

Note 2 - Vehicle Identification Number (VIN) - Service $09 Data Identifier (DID) $F190 - Diagnostic Communication Standards - Diagnostic Services - Diagnostic Response Interpretation in Standards Further, the VIN is stored differently depending on the protocol used.

Below we provide three examples on how to recover the VIN.

3.1: How to recover the VIN via OBD-II (SAE J1979):

To recover the Vehicle Identification Number from e.g. a passenger car (OBD-II), requires you, use Service $09 and the PID $02.

```
VIN - Vehicle Identification Number (request/response (OBD)
Tx: 7DF 02 09 02 55 55 55 55 55 55 55 55 Broadcast (OBD) Tx: 7DF
Rx: 7E8 10 14 49 02 01 31 44 34 47 50 30 Rx: 7E8 Single frame (SF)
Rx: 7E9 10 14 49 02 01 57 57 57 57 57 57 Rx: 7E9 Single frame (SF)
Rx: 7EA 10 14 49 02 01 57 57 57 57 57 57 Rx: 7EA Single frame (SF)
Rx: 7EB 10 14 49 02 01 57 57 57 57 57 57 Rx: 7EB Single frame (SF)
Rx: 7EC 10 14 49 02 01 57 57 57 57 57 57 Rx: 7EC Single frame (SF)
Rx: 7ED 10 14 49 02 01 57 57 57 57 57 57 Rx: 7ED Single frame (SF)
Rx: 7EE 10 14 49 02 01 57 57 57 57 57 57 Rx: 7EE Single frame (SF)
Rx: 7EF 10 14 49 02 01 57 57 57 57 57 57 Rx: 7EF Single frame (SF)
```

3.2: How to recover the VIN via UDS (ISO 14229-2):

To read the Vehicle Identification Number via UDS, you can use the UDS SID 0x22 and the DID 0xF190.

```
VIN - Vehicle Identification Number (request/response (UDS)
Tx: 7DF 02 22 F1 90 55 55 55 55 55 55 55 55 Broadcast (UDS) Tx: 7DF
Rx: 7E8 10 14 62 F1 90 57 57 57 57 57 57 Rx: 7E8 Single frame (SF)
Rx: 7E9 10 14 62 F1 90 57 57 57 57 57 57 Rx: 7E9 Single frame (SF)
Rx: 7EA 10 14 62 F1 90 57 57 57 57 57 57 Rx: 7EA Single frame (SF)
Rx: 7EB 10 14 62 F1 90 57 57 57 57 57 57 Rx: 7EB Single frame (SF)
Rx: 7EC 10 14 62 F1 90 57 57 57 57 57 57 Rx: 7EC Single frame (SF)
Rx: 7ED 10 14 62 F1 90 57 57 57 57 57 57 Rx: 7ED Single frame (SF)
Rx: 7EE 10 14 62 F1 90 57 57 57 57 57 57 Rx: 7EE Single frame (SF)
Rx: 7EF 10 14 62 F1 90 57 57 57 57 57 57 Rx: 7EF Single frame (SF)
```

3.3: How to recover the VIN via OBD-II (ISO 27145-3):

To recover the Vehicle Identification Number from e.g. a passenger car (WWH-OBD), requires you, use Service $09 and the PID $02.

```
VIN - Vehicle Identification Number (request/response (WWH-OBD) Tx: 7DF
Rx: 7E8 10 14 49 02 01 31 44 34 47 50 30 Rx: 7E8 Single frame (SF)
Rx: 7E9 10 14 49 02 01 57 57 57 57 57 57 Rx: 7E9 Single frame (SF)
Rx: 7EA 10 14 49 02 01 57 57 57 57 57 57 Rx: 7EA Single frame (SF)
Rx: 7EB 10 14 49 02 01 57 57 57 57 57 57 Rx: 7EB Single frame (SF)
Rx: 7EC 10 14 49 02 01 57 57 57 57 57 57 Rx: 7EC Single frame (SF)
Rx: 7ED 10 14 49 02 01 57 57 57 57 57 57 Rx: 7ED Single frame (SF)
Rx: 7EE 10 14 49 02 01 57 57 57 57 57 57 Rx: 7EE Single frame (SF)
Rx: 7EF 10 14 49 02 01 57 57 57 57 57 57 Rx: 7EF Single frame (SF)
```

UDS data logging - applications

In this section, we outline example use cases for specific UDS data logging.

UDS telematics for prototype electric vehicles

As an OEM, you may want to get data on various sensor parameters from prototype electric vehicles. Some of the data may be used for e.g. predictive maintenance, required battery usage, amount of charge, health, temperature and more by controlling UDS request frames and sensor frames parameters. The data can be used to develop a new product or improve an existing one. It is also a great way to gain a deep-dive into the vehicle's performance.

Training a predictive maintenance model

If you're looking to implement predictive maintenance across fleets of heavy duty vehicles, you need to collect data from the ECUs. The data could be anything ranging from to be relevant, including both sensor data (speed, RPM, available pressure, tire pressures and air) "roadside" data (fault, hot fault, time-to-return-to-service, etc) and other data points. Some of the data points are available only via UDS, e.g. the Fuel consumed, the Fuel type, the Fuel level, the Fuel type of a given ECU, the Fuel type of a given ECU, the Fuel type of a given ECU

CANEDGE1 - PRO BLACK BOX LOGGER

CANEDGE2 - PRO CAN IoT LOGGER

PYTHON API - PROCESS YOUR CAN DATA

Contact

CSS Electronics | VAT ID: DK36711949
Soeren Frichs Vej 38K (Office 35)
8230 Aabyhoej, Denmark

contact@csselectronics.com
+45 91 25 25 63

Terms of Service
Returns and Refunds
Privacy Policy
About Us
CO2 Neutral Since 2015

Newsletter

Email address

subscribe

© 2023, CSS Electronics