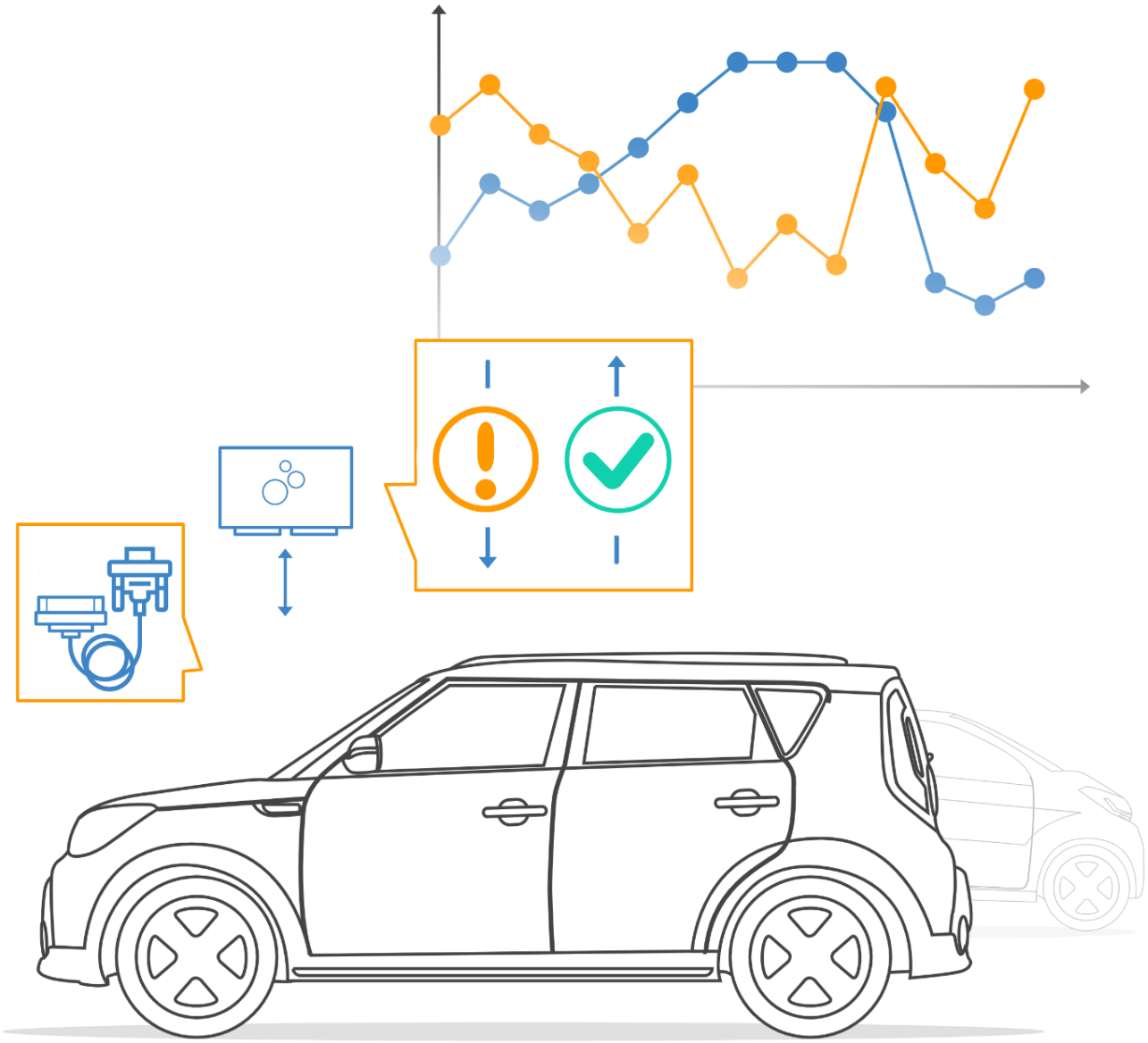
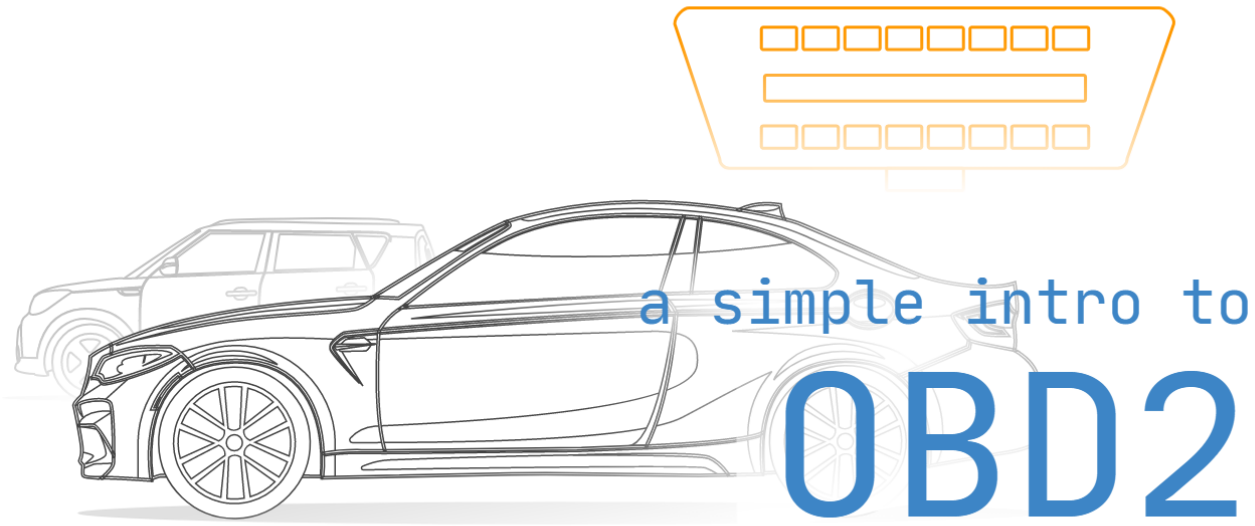


OBD2 data

starter pack

CSS Electronics | www.csselectronics.com | contact@csselectronics.com
Last modified: April 20, 2022

Table of Contents

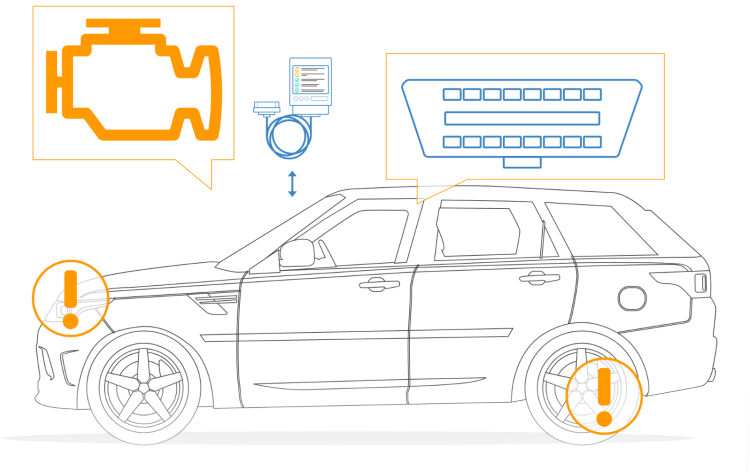


OBD2 Explained - A Simple Intro

In this guide we introduce the On Board Diagnostic (OBD2) protocol incl. the OBD2 connector, OBD2 parameter IDs (PID) and the link to CAN bus. This is a practical intro so you will also learn how to request and decode OBD2 data, key logging use cases and practical tips.

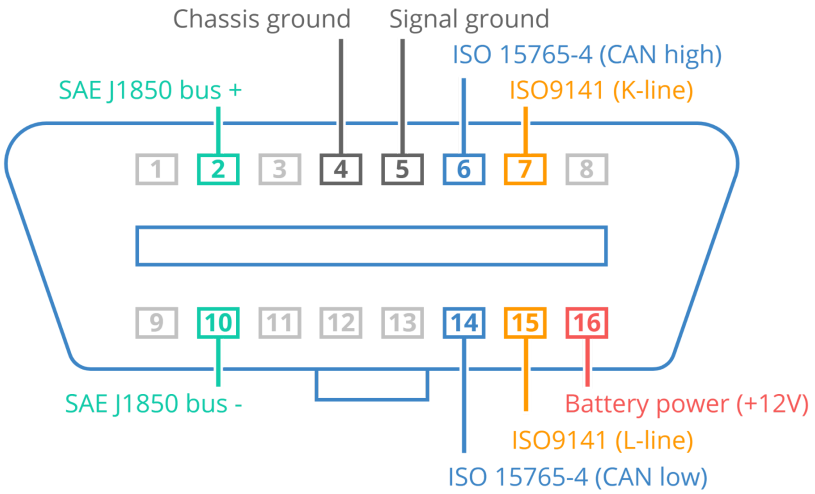
What is OBD2?

In short, OBD2 is your vehicle's built-in self-diagnostic system. You've probably encountered OBD2 already: Ever noticed the malfunction indicator light on your dashboard? That is your car telling you there is an issue. If you visit a mechanic, he will use an OBD2 scanner to diagnose the issue. To do so, he will connect the OBD2 reader to the OBD2 16 pin connector near the steering wheel. This lets him read OBD2 codes aka Diagnostic Trouble Codes (DTCs) to review and troubleshoot the issue.



The OBD2 connector

The OBD2 connector lets you access data from your car easily. The standard SAE J1962 specifies two female OBD2 16-pin connector types (A & B). In the illustration is an example of a Type A OBD2 pin connector (also sometimes referred to as the Data Link Connector, DLC).

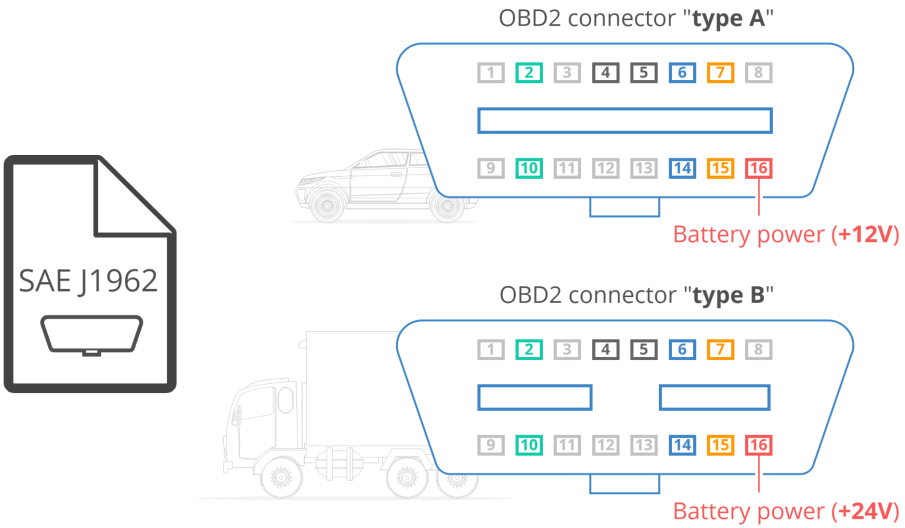


A few things to note:

- The OBD2 connector is near your steering wheel, but may be hidden behind covers/panels
- Pin 16 supplies battery power (often while the ignition is off)
- The OBD2 pinout depends on the communication protocol
- The most common protocol is CAN (via ISO 15765), meaning that pins 6 (CAN-H) and 14 (CAN-L) will typically be connected

OBD2 connector - type A vs. B

In practice, you may encounter both the type A and type B OBD2 connector. Typically, type A will be found in cars, while type B is common in medium and heavy duty vehicles.



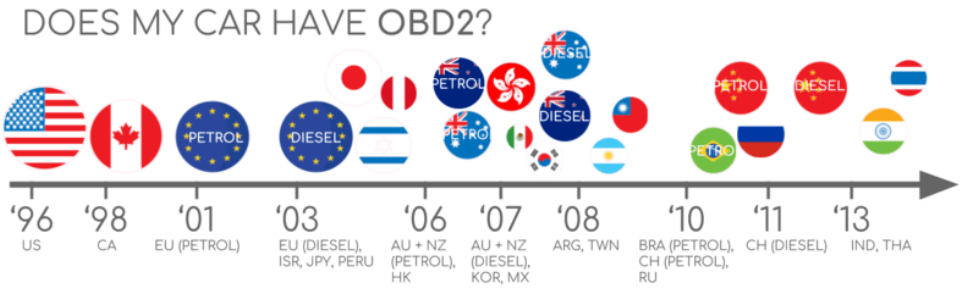
Type A & B connector differences

As evident from the illustration, the two types share similar OBD2 pinouts, but provide two different power supply outputs (12V for type A and 24V for type B). Often the baud rate will differ as well, with cars typically using 500K, while most heavy duty vehicles use 250K (more recently with support for 500K).

To help physically distinguish between the two types of OBD2 sockets, note that the type B OBD2 connector has an interrupted groove in the middle. As a result, a type B OBD2 adapter cable will be compatible with both types A and B, while a type A will not fit into a type B socket.

Does my car have OBD2?

In short: Probably! Almost all newer cars support OBD2 and most run on CAN (ISO 15765). For older cars, be aware that even if a 16 pin OBD2 connector is present, it may still not support OBD2. One way to determine compliance is to identify where & when it was bought new:



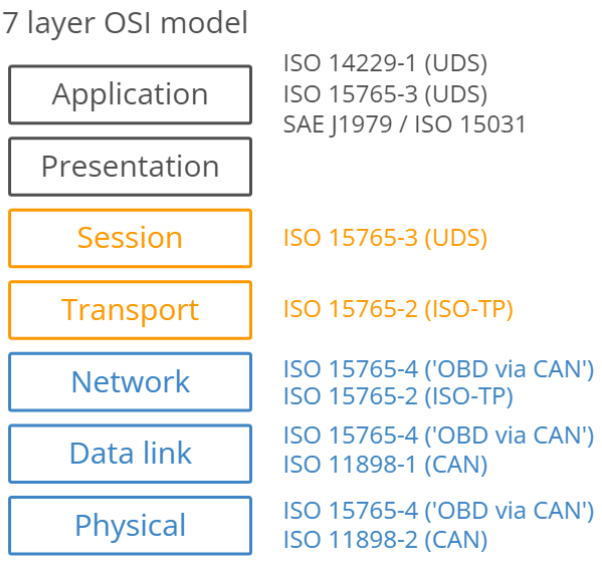
Link between OBD2 and CAN bus

On board diagnostics, OBD2, is a 'higher layer protocol' (like a language). CAN is a method for communication (like a phone). In particular, the OBD2 standard specifies the OBD2 connector, incl. a set of five protocols that it can run on (see below). Further, since 2008, CAN bus (ISO 15765) has been the mandatory protocol for OBD2 in all cars sold in the US.

What is the ISO 15765 standard?

ISO 15765 refers to a set of restrictions applied to the CAN standard (which is itself defined in ISO 11898). One might say that ISO 15765 is like "CAN for cars". In particular, ISO 15765-4 describes the physical, data link layer and network layers, seeking to standardize the CAN bus interface for external test equipment.

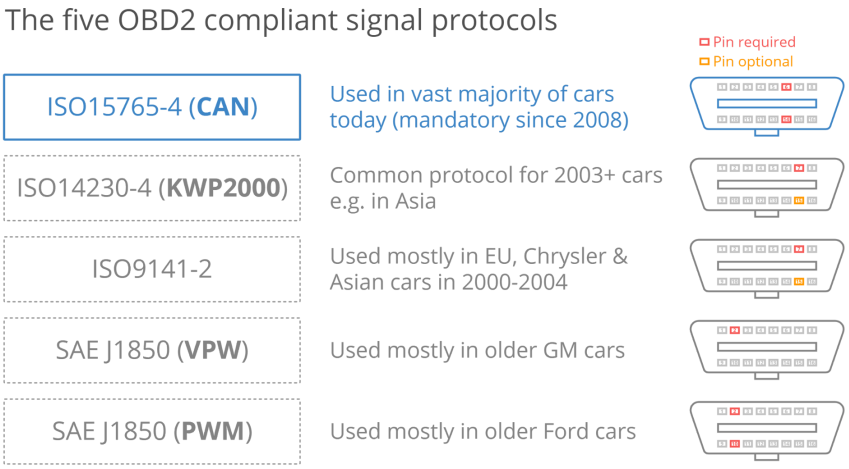
ISO 15765-2 in turn describes the transport layer (ISO TP) for sending CAN frames with payloads that exceed 8 bytes. This sub standard is also sometimes referred to as Diagnostic Communication over CAN (or DoCAN). See also the 7 layer OSI model illustration. OBD2 can also be compared to other higher layer protocols (e.g. J1939, CANopen).



The five OBD2 protocols

As explained above, CAN bus today serves as the basis for OBD2 communication in the vast majority of cars through ISO 15765. However, if you're inspecting an older car (pre 2008), it is useful to know the other four protocols that have been used as basis for OBD2. Note also the pinouts, which can be used to determine which protocol may be used in your car.

- ISO 15765 (CAN bus): Mandatory in US cars since 2008 and is today used in the vast majority of cars
- ISO14230-4 (KWP2000): The Keyword Protocol 2000 was a common protocol for 2003+ cars in e.g. Asia
- ISO9141-2: Used in EU, Chrysler & Asian cars in 2000-04
- SAE J1850 (VPW): Used mostly in older GM cars
- SAE J1850 (PWM): Used mostly in older Ford cars

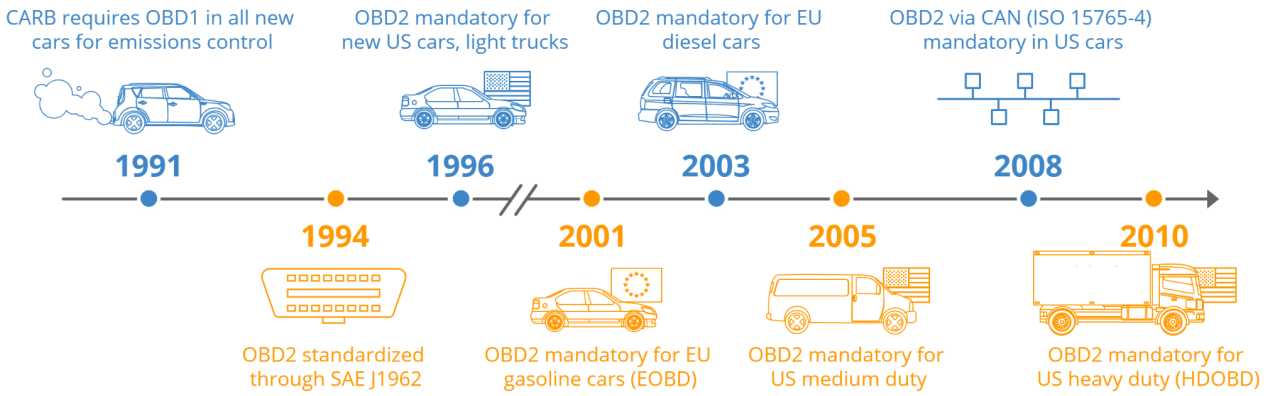


OBD2 history & future

History

OBD2 originates from California where the California Air Resources Board (CARB) required OBD in all new cars from 1991+ for emission control purposes. The OBD2 standard was recommended by the Society of Automotive Engineers (SAE) and standardized DTCs and the OBD connector across manufacturers (SAE J1962). From there, the OBD2 standard was rolled out step-by-step:

- 1996: OBD2 made mandatory in USA for cars/light trucks
- 2001: Required in EU for gasoline cars
- 2003: Required in EU also for diesel cars (EOBD)
- 2005: OBD2 was required in US for medium duty vehicles
- 2008: US cars must use ISO 15765-4 (CAN) as OBD2 basis
- 2010: Finally, OBD2 was required in US heavy duty vehicles



Future

OBD2 is here to stay - but in what form? Two potential routes may radically change OBD2:

OBD3/OBD-III - wireless emission testing

In today's world of connected cars, OBD2 tests can seem cumbersome: Manually doing emission control checks is time-consuming and expensive. The solution? OBD3 - adding telematics to all cars. Basically, OBD3 adds a small radio transponder (as in e.g. bridge tolls) to all cars. Using this, the car vehicle identification number (VIN) and DTCs can be sent via WiFi to a central server for checks.

Many devices today already facilitate transfer of CAN or OBD2 data via WiFi/cellular - e.g. the CANedge2 WiFi CAN logger. This saves cost and is convenient, but it is also politically a challenge due to surveillance concerns.

Eliminating 3rd party OBD2 services

The OBD2 protocol was originally designed for stationary emission controls. Yet, today OBD2 is used extensively for generating real-time data by 3rd parties - via OBD2 dongles, CAN loggers etc. However, the German car industry is looking to change this:

"OBD has been designed to service cars in repair shops. In no way has it been intended to allow third parties to build a form of data-driven economy on the access through this interface"

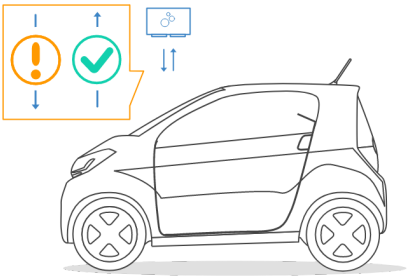
- Christoph Grote, SVP Electronics, BMW (2017)

The proposal is to "turn off" the OBD2 functionality while driving - and instead collect the data in a central server. This would effectively put the manufacturers in control of the automotive 'big data'. The argumentation is based in security (e.g. removing the risk of car hacking), though many see it as a commercial move. Whether this becomes a real trend is to be seen - but it may truly disrupt the market for OBD2 3rd party services.

OBD2 parameter IDs (PID)

Why should you care about OBD2 data? Mechanics obviously care about OBD2 DTCs (maybe you do too), while regulatory entities need OBD2 to control emission. But the OBD2 protocol also supports a broad range of standard parameter IDs (PIDs) that can be logged across most cars.

This means that you can easily get human-readable OBD2 data from your car on speed, RPM, throttle position and more. In other words, OBD2 lets you analyze data from you car easily - in contrast to the OEM specific proprietary raw CAN data.



Decoding OBD2 vs CAN bus data

In principle it is simple to log the raw CAN frames from your car. If you e.g. connect a CAN logger to the OBD2 connector, you'll start logging broadcasted CAN bus data out-the-box. However, the raw CAN messages need to be decoded via a database of conversion rules (DBC) and a suitable CAN software that supports DBC decoding (like e.g. asammdf). The challenge is that these CAN DBC files are typically proprietary, making the raw CAN data unreadable unless you're the automotive OEM.

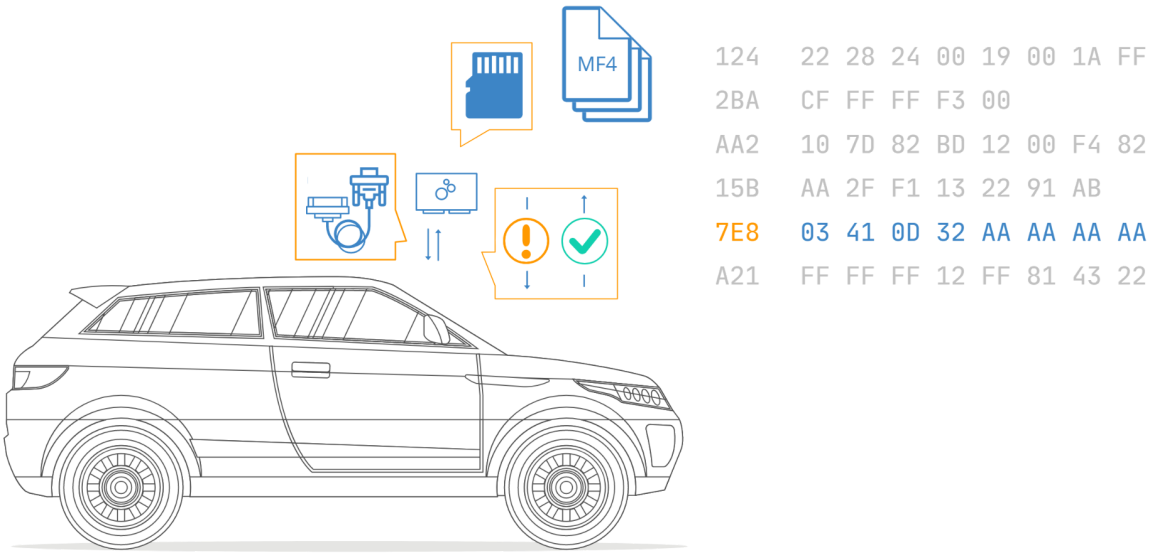
Car hackers may try to reverse engineer the rules, though this can be difficult. CAN is, however, still the only method to get "full access" to your car data - while OBD2 only provides access to a limited subset of data.

How to log OBD2 data?

OBD2 data logging works as follows:

- You connect an OBD2 logger to the OBD2 connector
- Using the tool, you send 'request frames' via CAN
- The relevant ECUs send 'response frames' via CAN
- Decode the raw OBD2 responses via e.g. an OBD2 DBC

In other words, a CAN logger that is able to transmit custom CAN frames can also be used as an OBD2 logger.
Note that cars differ by model/year in what OBD2 PIDs they support. For details, see our OBD2 data logger guide.

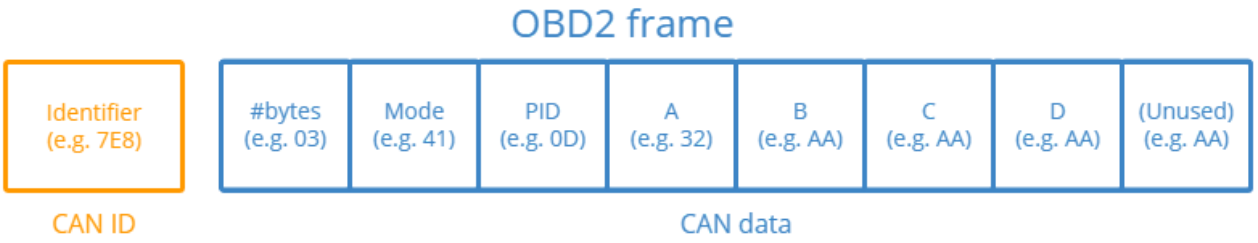


CANedge OBD2 data logger

The CANedge lets you easily log OBD2 data to an 8-32 GB SD card. Simply specify what OBD2 PIDs you wish to request, then connect it to your car via an OBD2 adapter to start logging. Process the data via free software/APIs and our OBD2 DBC.

Raw OBD2 frame details

To get started recording OBD2 data, it is helpful to understand the basics of the raw OBD2 message structure. In simplified terms, an OBD2 message consists of an identifier and data. Further, the data is split in Mode, PID and data bytes (A, B, C, D) as below.



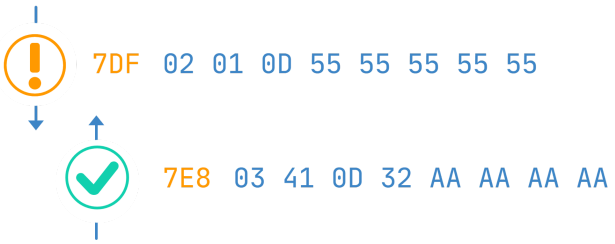
OBD2 message fields explained

- **Identifier**: For OBD2 messages, the identifier is standard 11-bit and used to distinguish between "request messages" (ID 7DF) and "response messages" (ID 7E8 to 7EF). Note that 7E8 will typically be where the main engine or ECU responds at.
- **Length**: This simply reflects the length in number of bytes of the remaining data (03 to 06). For the Vehicle Speed example, it is 02 for the request (since only 01 and 0D follow), while for the response it is 03 as both 41, 0D and 32 follow.
- **Mode**: For requests, this will be between 01-0A. For responses the 0 is replaced by 4 (i.e. 41, 42, ... , 4A). There are 10 modes as described in the SAE J1979 OBD2 standard. Mode 1 shows Current Data and is e.g. used for looking at real-time vehicle speed, RPM etc. Other modes are used to e.g. show or clear stored diagnostic trouble codes and show freeze frame data.

- **PID**: For each mode, a list of standard OBD2 PIDs exist - e.g. in Mode 01, PID 0D is Vehicle Speed. For the full list, check out our OBD2 PID overview. Each PID has a description and some have a specified min/max and conversion formula. The formula for speed is e.g. simply A, meaning that the A data byte (which is in HEX) is converted to decimal to get the km/h converted value (i.e. 32 becomes 50 km/h above). For e.g. RPM (PID 0C), the formula is (256*A + B) / 4.
- **A, B, C, D**: These are the data bytes in HEX, which need to be converted to decimal form before they are used in the PID formula calculations. Note that the last data byte (after Dh) is not used.

OBD2 request/response example

An example of a request/response CAN message for the PID 'Vehicle Speed' with a value of 50 km/h can be seen in the illustration. Note in particular how the formula for the OBD2 PID 0D (Vehicle Speed) simply involves taking the 4th byte (0x32) and converting it to decimal form (50).



Extended OBD2 PID request/response

In some vehicles (e.g. vans and light/medium/heavy duty vehicles), you may find that the raw CAN data uses extended 29-bit CAN identifiers instead of 11-bit CAN identifiers.

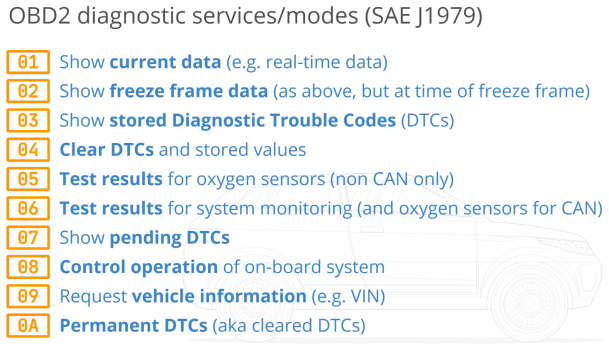
In this case, you will typically need to modify the OBD2 PID requests to use the CAN ID 18DB33F1 instead of 7DF. The data payload structure is kept identical to the examples for 11-bit CAN IDs.

If the vehicle responds to the requests, you'll typically see responses with CAN IDs 18DAF100 to 18DAF1FF (in practice, typically 18DAF110 and 18DAF11E). The response identifier is also sometimes shown in the 'J1939 PGN' form, specifically the PGN 0xDA00 (55808), which in the J1939-71 standard is marked as 'Reserved for ISO 15765-2'.

We provide an OBD2 DBC file for both the 11-bit and 29-bit responses, enabling simple decoding of the data in most CAN software tools.

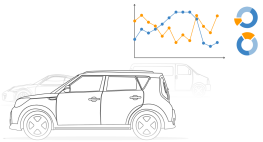
The 10 OBD2 services (aka modes)

There are 10 OBD2 diagnostic services (or modes) as described in the SAE J1979 OBD2 standard. Mode 1 shows Current Data and is used for looking at real-time parameters like vehicle speed, RPM, throttle position etc. Other modes are e.g. used to show/clear diagnostic trouble codes (DTCs) and show freeze frame data. Manufacturers do not have to support all diagnostic services - and they may support modes outside these 10 services (i.e. manufacturer specific OBD2 services).



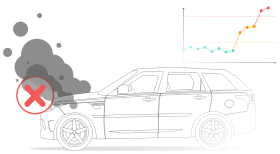
OBD2 data logging - use case examples

OBD2 data from cars and light trucks can be used in various use cases:

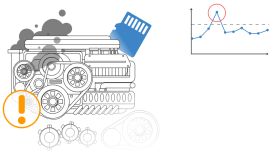


Logging data from cars
OBD2 data from cars can e.g. be used to reduce fuel costs, improve driving, test prototype parts and insurance
learn more

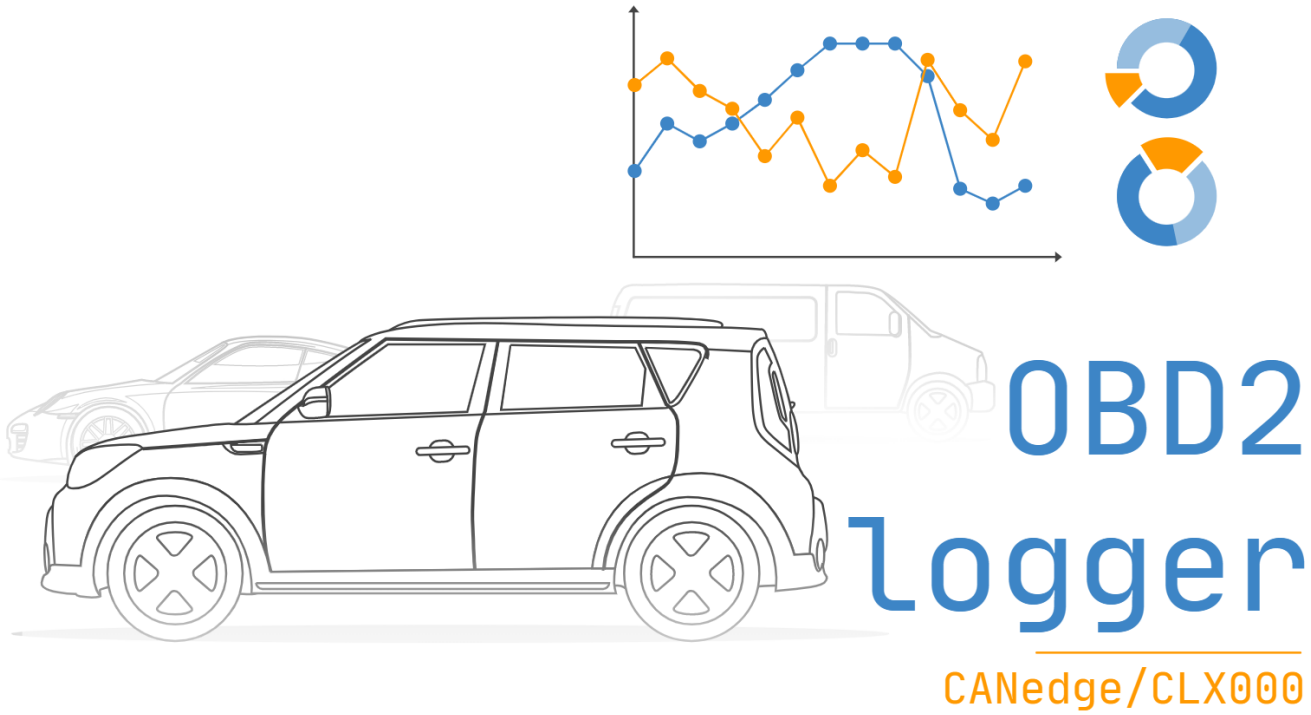
Real-time car diagnostics
OBD2 interfaces can be used to stream human-readable OBD2 data in real-time, e.g. for diagnosing vehicle issues
learn more



Predictive maintenance
Cars and light trucks can be monitored via IoT OBD2 loggers in the cloud to predict and avoid breakdowns
learn more



Vehicle blackbox logger
An OBD2 logger can serve as a 'blackbox' for vehicles or equipment, providing data for e.g. disputes or diagnostics
learn more



OBD2 Data Logger - Easily Record & Visualize Your Car Data

This intro recaps the basics of OBD2 logging, the top 4 benefits and example use cases. You can also download sample OBD2 data from an Audi A4 (speed, RPM, ...).

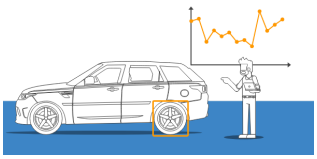
How does OBD2 data logging work?

First, let us briefly recap the basics of OBD2: OBD2 offers a standardized set of parameters (OBD2 PIDs) that you can record - and easily decode - across most cars.

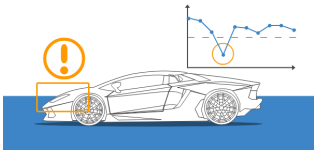
To log OBD2 data involves 3 simple steps:

1. Configure your OBD2 logger with a list of OBD2 PIDs
2. Connect it in your car via an OBD2 adapter to start logging
3. Extract the SD and decode the data via the free software/API
4. For details, see the FAQ further below - or our docs

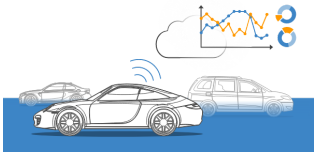
Top 4 benefits of OBD2 data logging



Driver/vehicle/part optimization
OBD2 data lets you e.g. monitor/optimize driving behavior and tune your car. OEMs can use the data to analyze the field performance of new prototype parts



Rare issue diagnostics
Rare issues in cars may occur briefly while driving, yet not during repairs. Logging the OBD2 data lets you analyze the period around an event to troubleshoot the problem



Car fleet management
OBD2 WiFi telematics at fleet level enables e.g. driver behavior research, fuel cost reductions, fewer breakdowns, compliance, dispute handling and predictive maintenance

Data control & custom integration
With an OBD2 WiFi logger, you record the raw time series data, which can be extracted via SD or uploaded to your own server - for easy custom integration via open APIs

The CANedge OBD2 data logger

PLUG & PLAY
Log data out-the-box. Standalone - no PC required. Power via CAN / OBD2 connector

PRO SPECS
Extractable 8-32 GB SD. 2xCAN/LIN. CAN FD. Zero data loss. 50 µs RTC resolution.

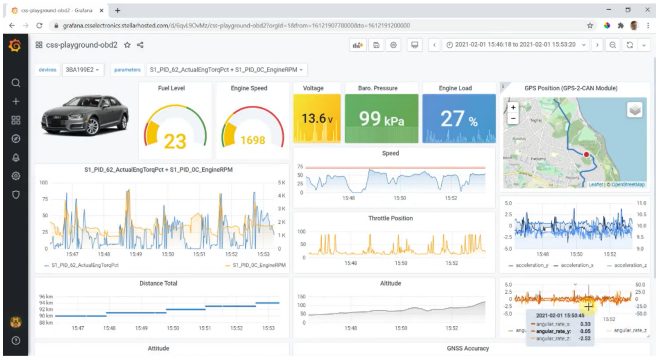
SO SMALL
Only 8 x 5 x 2 CM. 100G. Aluminium enclosure. 4 LEDs. Modular (add e.g. GPS-to-CAN)

CONFIGURABLE
Advanced filters. Transmit lists. Triggers. Cyclic logging. Compression

FAST & SECURE
Industrial SD card. Read data at 80 MB/s. Data encryption for GDPR/CCPA

OPEN SOURCE
Popular data format (MF4). Free open source GUI/API. Easily convert via DBC

Software example: OBD2 dashboard for your cars



With the CANedge, you can easily set up free, custom browser dashboards for visualizing your OBD2 data and setting alerts.

You can also combine your OBD2 data with GPS/IMU data by using the CANmod.gps as an add-on module for the CANedge.

Check out the online playground - or learn more in our intro!

Use case examples

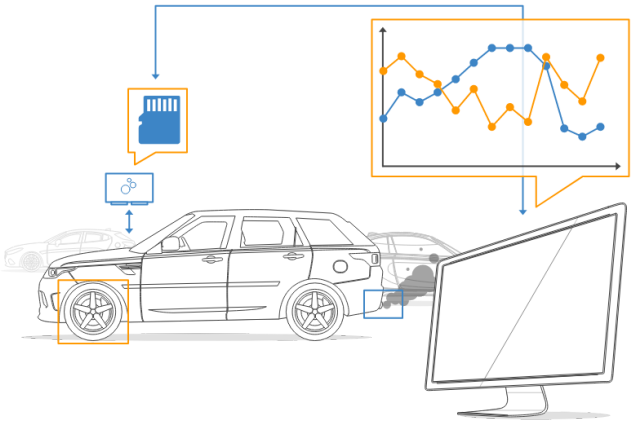
Below we provide practical examples of how the CANedge can be used for logging OBD2 data.

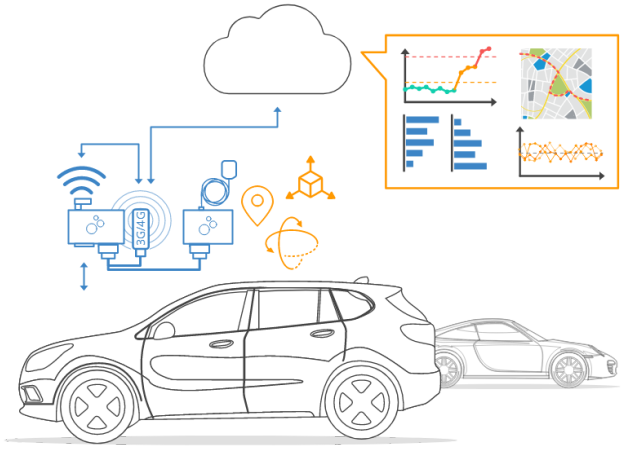
OEM field testing of vehicle parts

Need to log CAN/OBD2 field data from vehicles in the field?

As an OEM, you may need to do late-stage field tests of prototype equipment. This often requires OBD2 & CAN data from multiple vehicles over e.g. months. The CANedge1 is ideal as it's extremely compact, plug & play and can be pre-configured easily. Data can be collected periodically and analyzed in your favorite CAN tools or the free asammdf GUI/API.

Learn about the CANedge1.





Vehicle telematics (OBD2 + GNSS/IMU + 3G/4G)

Need to setup OBD2 telematics for on-road vehicle fleets?

The CANedge2 can upload recorded OBD2 data through a WiFi access point, e.g. a 3G/4G hotspot. This enables near real-time wireless OBD2 data transfer from e.g. on-road vehicles to your own cloud server. The OBD2 data can be auto-processed via the open APIs (incl. OBD2 DBC decoding support), while the CANedge2 devices can be updated over-the-air. If you need to add GNSS position or 3D inertial sensor data, you can use the CANmod.gps add-on module.

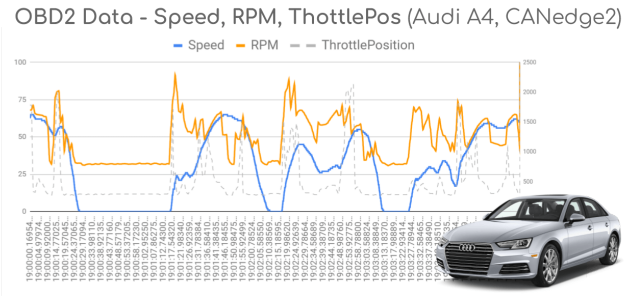
Learn about the CANedge2.

Case study: OBD2/CAN telematics

Learn how Volkswagen uses the CANedge2 to log OBD2 and raw CAN data to an SD card - as well as auto-push the data to their own self-hosted server for analysis.

"The CANedge2 got us up-and-running at a rapid pace with robust config options - and the support was outstanding!"

Read case study
40+ other case studies



OBD2 data from an Audi A4

Below you can download OBD2 samples from the CANedge.

You can also download the free open source OBD2 software and try out the process of decoding the raw OBD2 data.

raw obd2 decoded obd2 software

FAQ

What data can I log via OBD2?

The OBD2 protocol (SAE J1979) specifies a range of standardized vehicle data that you can log from your car. Note, however, that each car differs in what OBD2 data is supported - and in particular older cars often support fewer parameters.

With that said, we've listed some of the standard OBD2 parameters that are often available:

Fuel system status
Engine load
Coolant temperature
Fuel trim
Fuel pressure
Intake manifold pressure
Engine RPM
Vehicle speed

Intake air temperature
MAF air flow rate
Throttle position
Air status
Oxygen sensor status
Runtime since engine start
Distance with MIL on
Fuel tank level input
System vapor pressure
Absolute load value
Hybrid battery pack life
Engine oil temperature
Engine fuel rate
Torque
VIN
Various DTCs

For further details, see the OBD2 PID Wiki page or the SAE J1979 standard.

How do I decode raw OBD2 data?

To decode raw OBD2 data from a CANedge OBD2 data logger into physical values (km/h, rpm, ...), you need a database of decoding rules and suitable OBD2 software.

For this purpose, we provide a 100% free OBD2 DBC file, which contains the majority of the standardized Mode 01 (aka Service 01) OBD2 PID decoding rules as found on e.g. the OBD2 PID Wiki page.

The OBD2 DBC file uses extended multiplexing to enable OBD2 decoding. For more on this, see our DBC intro and our OBD2 intro (where we explain how to interpret raw CAN frames with OBD2 responses).

You can load your raw OBD2 data and the OBD2 DBC file in one of our free software tools (e.g. asammdf or our OBD2 dashboard integrations). This lets you visualize your decoded OBD2 data such as Speed, Engine Speed, MAF, Fuel Level etc.

One of the benefits of using this approach is that you can easily modify the OBD2 DBC to include additional proprietary OBD2 PIDs. You can also combine the OBD2 DBC with e.g. proprietary CAN DBC files to enable both CAN/OBD2 car data logging.

How to log UDS data?

What is UDS?
The Unified Diagnostic Services protocol (UDS, ISO 14229-1) is a communication protocol used within automotive ECU communication. An UDS diagnostic tool can be used to send requests messages into the CAN bus, with the purpose of e.g. retrieving information from specific ECUs. While OBD2 is intended as an on-board diagnostics protocol (for while the vehicle is moving), UDS is intended as an off-board diagnostic protocol (for when the vehicle is standing still).

How to make UDS requests over ISO-TP (ISO 15765-2)
Requesting OBD2 PIDs is relatively simple: An OBD2 scan tool or OBD2 data logger sends a specific CAN frame (the OBD2 request) and if the car supports the OBD2 PID, it responds with a single CAN frame. In contrast, UDS requests may require performing transport protocol requests. For example, you can use the UDS service 0x22 to request data by identifier. In this case, the communication flow may be as follows:

An 'UDS data logger' sends a request frame specifying the service ID (SID) and data identifier (DID) of interest
The car responds with a first frame, which contains the SID, DID, total message length and initial payload bytes
Shortly after the first frame, the UDS logger acknowledges the first frame by sending a flow control frame

After this, the ECU sends consequtive frames with the remaining payload of the message
In other words, to log UDS data requires that the UDS tool can send custom CAN frames and flow control frames. Further, the software tools must be able to re-construct multiframe UDS responses in order to extract the payload and decode it.

UDS and OBD2 extended PIDs for car data logging
The UDS service ID (SID) and the data identifiers (DID) are sometimes combined into one ID and referred to as an 'extended OBD2 PID' - for example 0x220101. Recording UDS data via service 0x22 requests is sometimes used to extract car data beyond what is available through service 01 OBD2 PID requests. For example, some electric cars provide access to State of Charge (SoC%) via UDS requests under service 0x22.

Using the CANedge as an UDS data logger
The CANedge can be configured to send UDS requests. This is done by sending a request frame and adding a flow control frame within X ms after the request. When done right, this triggers the full sequence of UDS responses. The resulting log files with UDS responses can e.g. be loaded in tools like CANalyzer (by converting the MF4 data to Vector ASC) for decoding. Alternatively, the multiframe UDS response data can be processed via our free Python CAN bus API e.g. to push parameters to a Grafana UDS dashboard. Our github API examples library contains UDS response data and an UDS DBC file for decoding State of Charge (SoC%) from a Hyundai Kona EV. To learn more about this, see our EV data logger article or contact us.

Does my car support OBD2?

Most likely. The majority of cars and light trucks use the OBD2 standard as their on-board diagnostics methodology. In particular, OBD2 has been mandatory in USA since 1996 and in EU since 2003 (here it is denoted EOBD, but it is basically the same).

However, even if your vehicle supports OBD2, you may be unable to log the data you want. First, each vehicle model (brand/year) differs in terms of what OBD2 data parameters are supported. In particular, older cars often have more limited support for e.g. real-time parameters like speed, RPM etc. Further, some car manufacturers have begun restricting access to the OBD2 data to better control their vehicle data. And finally, while the vast majority of cars use CAN as the signal protocol for OBD2, you may encounter cases for old US cars (pre 2008) or e.g. some EU brands where another protocol is used instead.

Note: A useful check is to review your OBD2 connector in your car. To use a CAN logger to record OBD2 data, it's necessary that you have "metal pins" in the CAN High (pin 6) and CAN Low (pin 14) pins of the OBD2 connector - see our OBD2 connector illustration (the red pins). If in doubt, send us a picture for review.

There are 5 OBD2 signal protocols in total:

- ISO 15765 (CAN): Dominant, as it's required in all vehicles sold in the USA since 2008
- SAE J1850: Standard of the Ford Motor Company
- SAE J1850: Standard of General Motors
- ISO 9141-2: Used by Chrysler and some EU/Asian vehicles
- ISO 14230 (KWP2000): Mainly used by EU manufacturers

The CANedge/CLX000 supports CAN based OBD2 - if in doubt whether your car is supported, contact us.

If you do not have access to your car to visually screen the OBD2 connector, you can do a rough protocol check for the specific car here: OBD2 compatibility (cars). For further guidance on the basics, check our simple intro to OBD2.

Note: With an OBD2 logger, you can also check what Mode 01 OBD2 parameter IDs are supported in your car. To do so, you request 'Supported PIDs' parameters (ID 00, 20, 40, 60, 80, A0, C0). Once you've logged these, you can review the response data bytes bit-by-bit to see if a PID is supported not (see the Wikipedia OBD2 PID article for details).

Can I log OBD2 data from heavy duty trucks?

Generally speaking, you'll be logging OBD2 data from cars and light trucks. In contrast, if you're aiming to log data from heavy-duty vehicles like trucks, tractors, excavators etc, you will typically need to record J1939 data. The J1939 protocol is a standardized protocol used in most heavy-duty vehicles today, meaning that data can be decoded across vehicle brands (similar to the case of OBD2). To decode J1939 data, a J1939 DBC file is required - and you can use the CANedge/CLX000 as a J1939 data logger as well.

Which CANedge/CLX000 should I choose?

Both the CANedge and CLX000 data logger series can be used as OBD2 data loggers.

If your main goal is to log data to an SD card, we recommend the CANedge series - it is the 2nd generation of the CLX000 and optimized for logging. Further, if you wish to be able to auto-upload your log files to your own server we recommend the CANedge2. This device is particularly useful if you aim to set up OBD2 telematics workflows and OBD2 dashboards.

If you wish to also be able to stream OBD2 data in real-time via USB to your PC, we recommend the CLX000 series, e.g. the CL2000.

If in doubt, please contact us - we'll be able to help find the best fit quickly based on a few details on your use case.

Can I stream OBD2 data in real-time?

Yes, the CLX000 series enables real-time streaming of raw CAN data and OBD2 data via USB - see our OBD2 streaming intro.

OBD2 vs CAN logging - what is the difference?

If you connect a CAN logger like the CANedge or CLX000 to your car via the OBD2 connector, it will by default start recording raw CAN bus data (in most cars). This raw CAN data is "broadcasted" by the car sensors and used by the car itself to communicate.

In some cases, you may want to log this raw CAN data - e.g. if you are the Original Equipment Manufacturer (OEM) of the car. In this case you will know what each CAN ID and data bytes represent and you'll have a CAN database (DBC file) that you can use to decode the raw CAN data. However, if you're not the vehicle OEM, the only way to decode the raw CAN data will be by hacking your car and reverse engineer the data. In some cases, you can be lucky and find partial databases for your specific car model/year online - e.g. from projects like opendbc.

In most cases, if you're not the car OEM, your main option for collecting data will be via the OBD2 protocol. Today, almost all cars base their OBD2 communication on CAN bus. OBD2 is available only "on-request", in contrast to the raw CAN bus data. To log OBD2 data, you're basically sending a specific custom CAN frame into the vehicle CAN bus. Essentially you're using the CAN bus to send a command to the car to respond with the requested data. The car may respond to your request, assuming the specific OBD2 PID is supported by the car (which is to some extent up to the OEM to decide).

OBD2 loggers, dongles, scanners - what is the difference?

There is a massive number of OBD2 device types on the market - below we outline the main groups:

OBD2 Scanners: These are typically used by mechanics/technicians for diagnostic purposes - e.g. to identify what causes your malfunction indicator lamp (MIL) to be turned on. The OBD2 scanners typically include built-in databases for diagnostic trouble codes (DTCs) and functionality for clearing these. OBD2 bluetooth scanners and OBD2 WiFi scanners also exist for more convenient access to OBD2 diagnostic codes via smartphone.

OBD2 Dongles: While not a "formal" definition, OBD dongles typically refer to small, low cost and simple-to-use consumer oriented bluetooth OBD2 readers. They typically provide data via e.g. a smartphone app, allowing you to get a real-time view of your vehicle's performance. They are great for plug-and-play consumer purposes, but offer very limited flexibility in terms of use cases. Typically these devices use an ELM327 microcontroller.

OBD2 Data Loggers: An OBD logger can be used to record OBD2 timeseries data to an SD card in "standalone mode" (i.e. no PC or app required). The data can be extracted via USB or an extractable SD card for later analysis. The CANedge1 is an example of a CAN bus data logger that can be used as an OBD2 data logger.

OBD2 WiFi Loggers: Some OBD2 data loggers also support WiFi data transfer. For example, the CANedge2 can log OBD2 data to an SD card and auto-transfer the OBD2 data to a server (cloud, self-hosted) via a WiFi access point (incl. 3G/4G hotspots). This makes OBD2 WiFi loggers ideal for OBD2 telematics - e.g. to enable wireless access to car codes and parameters like speed, MAF, RPM etc. This type of solution is also ideal for creating OBD2 dashboards to visualize data across car fleets.

OBD2 Interfaces: Some CAN interfaces can also serve as OBD2 interfaces, allowing for real-time streaming of OBD2 data to a PC via USB. For example, the CLX000 enables USB streaming of OBD2 data to a PC via Wireshark.

Can I leave an OBD2 logger in my car?

In most cases, yes.

Typically, when you connect e.g. a CANedge to your vehicle, it'll turn on/off with the ignition, since the OBD2 connector typically uses the IGN power supply. This means that the CANedge will not drain any power from the vehicle battery when turned off.

However, in some vehicles the OBD2 connector power supply will be directly wired to the battery, meaning that the CANedge may still be turned on when the car is off. Normally this is not an issue as the power drain from the logger itself is minimal (<1W). However, if you're polling OBD2 data from the ECUs, this may "wake up" the ECUs while the car is off. This can result in some power drain if the car is off for a longer duration.

You can quickly verify if your logger turns on/off with your vehicle by evaluating the LEDs after the car has been turned off for 15-20 minutes - if the LEDs are not lit, the CANedge is turned off.

In case the CANedge/CLX000 does not turn off with the vehicle - and you know the vehicle will be turned off for longer durations - you can disconnect the device during this period. Alternatively, you can configure the CANedge to start/stop transmitting based on broadcasted CAN data patterns. For example, if your car emits a specific CAN ID or data byte pattern when the ignition is turned on/off, this can be used to toggle the transmit functionality of the CANedge. Finally, you can use a DB9-DC splitter cable and a DC-cigarette receptacle adapter to power the CANedge car data logger via your cigarette power supply. This is typically linked to the ignition, forcing the device to turn off with the vehicle. For details, see the CANedge Docs.

Can I record my car's GPS position?

Your car may have a built-in GPS, though it's rarely possible to extract the data via OBD2 or the proprietary CAN data. For practical purposes, we recommend using a GPS-to-CAN module like the CANmod.gps as an add-on for your CANedge. This lets you log both GNSS position and 3D inertial sensor data in a consistent way via Channel 2 of the CANedge. You can then combine this data with the timesynced CAN/OBD2 data that you record from your car via Channel 1.

Thank you for reading our guide - we hope you found it useful!

CSS Electronics | DK36711949 | Soeren Frichs Vej 38K, 8230 Aabyhoej, Denmark

www.csselectronics.com | contact@csselectronics.com | +45 91 25 25 63 | LinkedIn

Products | Software | Guides | Case studies