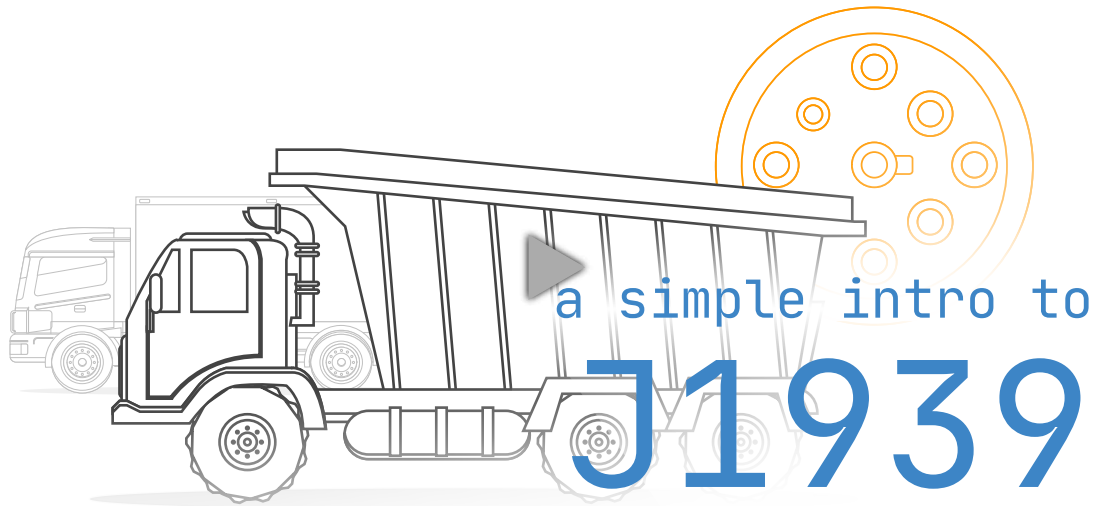


J1939 Explained - A Simple Intro [2025]



Need a simple intro to SAE J1939?

In this guide we introduce the J1939 protocol basics incl. PGNs and SPNs, key characteristics, request messages and the transport protocol.

Note: This is a **practical intro** so you will also learn how to decode raw J1939 data with real-life example data.

Learn below why this has become the **#1 introduction to J1939**.



You can also watch our J1939 intro above - or get the PDF.

In this article

1. What is J1939?
2. J1939 standards
3. The J1939 connector
4. The J1939 PGN & SPN
5. How to decode J1939
6. Requests messages
7. J1939 transport protocol (TP)
8. J1939 logging use cases

Author: Martin Falch  (updated January 2025)



Download as PDF

What is J1939?

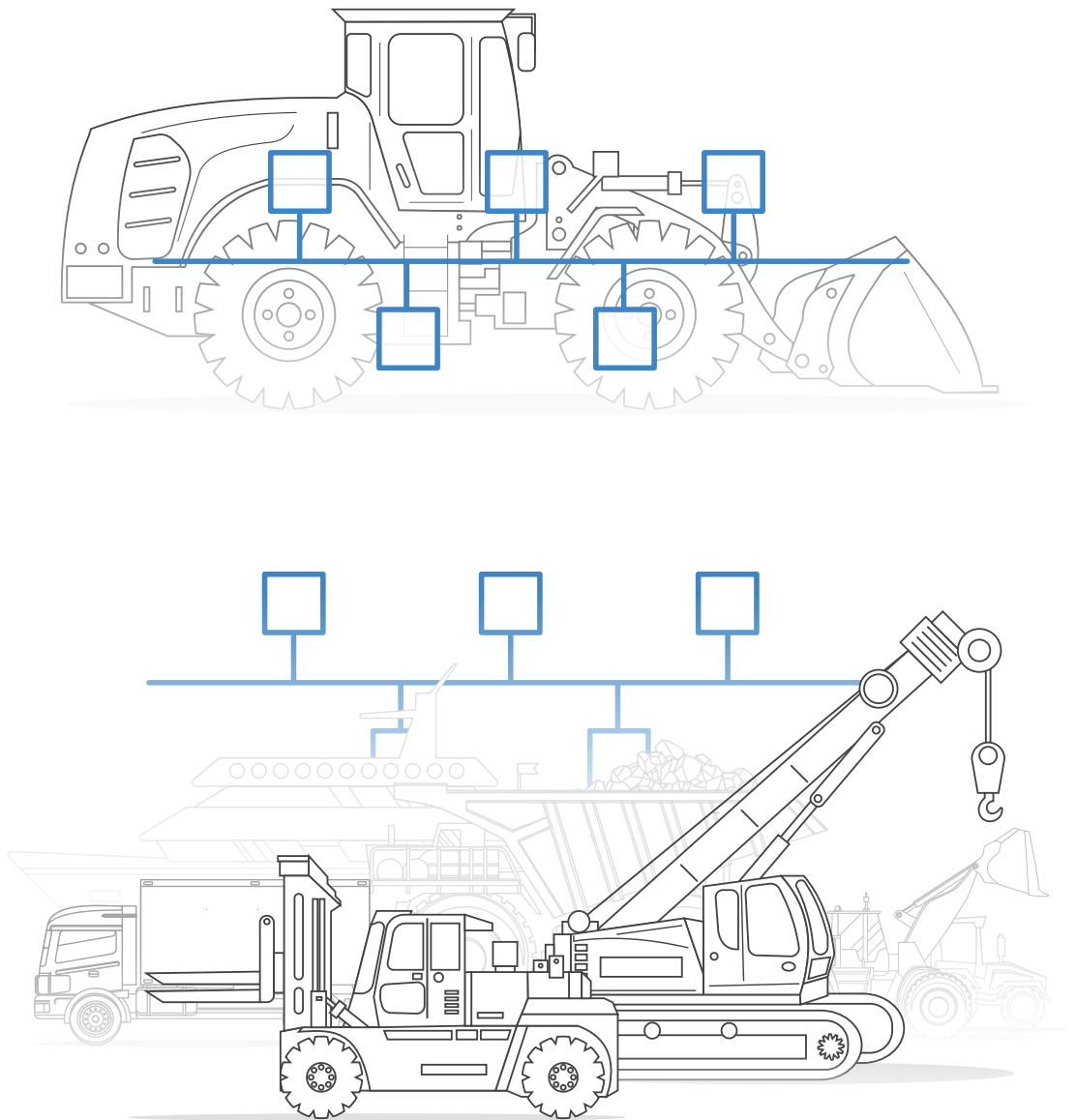
SAE J1939 is a set of standards that define how ECUs communicate via CAN bus in heavy-duty vehicles.

As explained in our CAN bus intro, most vehicles today use the Controller Area Network (CAN) for ECU communication. However, CAN bus only provides a "basis" for communication - not a "language" for conversation.

In most heavy-duty vehicles, this language is the SAE J1939 standard defined by SAE International.

In more technical terms, J1939 provides a higher layer protocol based on CAN (more on this later).

What does that mean, though?



One standard across heavy-duty vehicles

In simple terms, J1939 offers a **standardized** method for communication across ECUs, or in other words:

J1939 provides a common language across manufacturers.

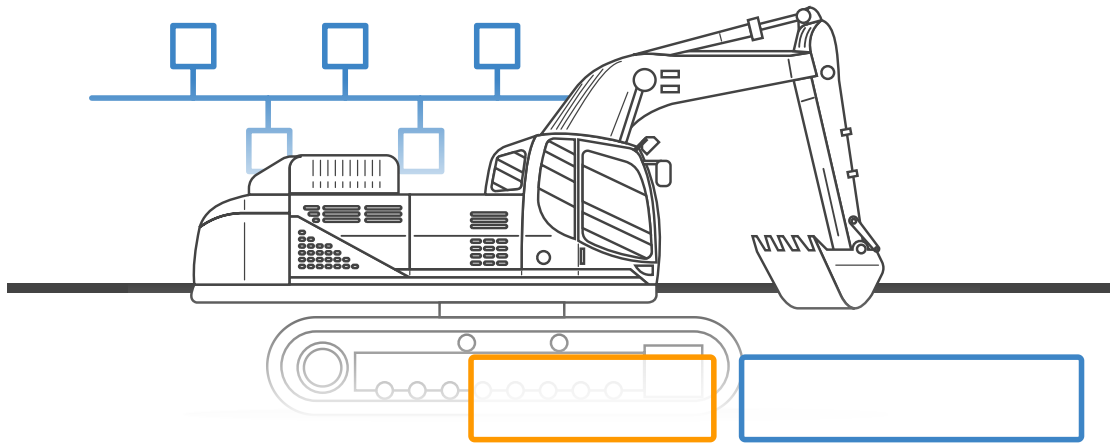
In contrast, e.g. cars use proprietary OEM specific protocols.

The J1939 standardization is a key enabler to data logging use cases across heavy-duty vehicles - more on this below.

J1939 application examples	+
The role of SAE International	+

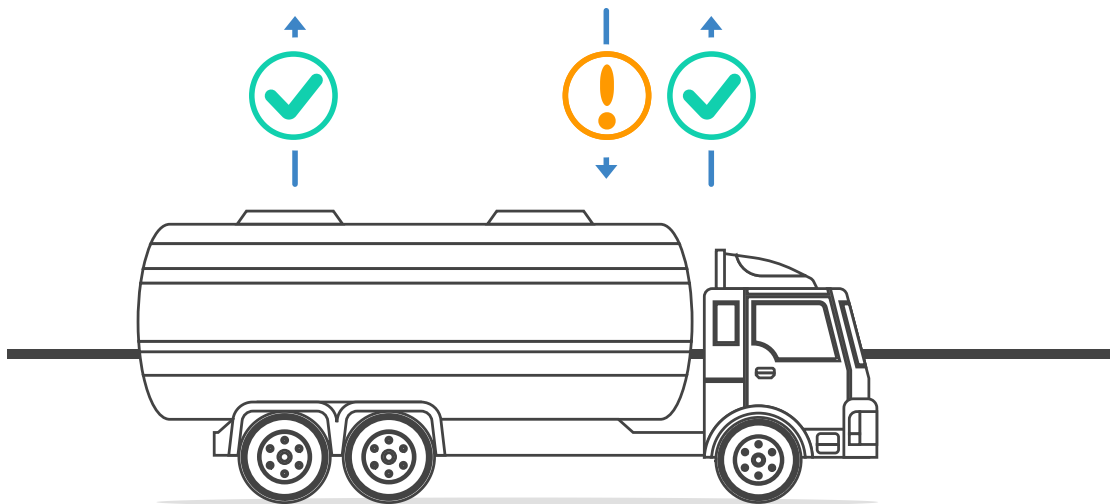
4 key characteristics of J1939

The J1939 protocol has a set of defining characteristics outlined below:



250K baud rate & 29-bit extended ID

The J1939 baud rate is typically 250K (though recently with support for 500K) - and the identifier is extended 29-bit (CAN 2.0B)



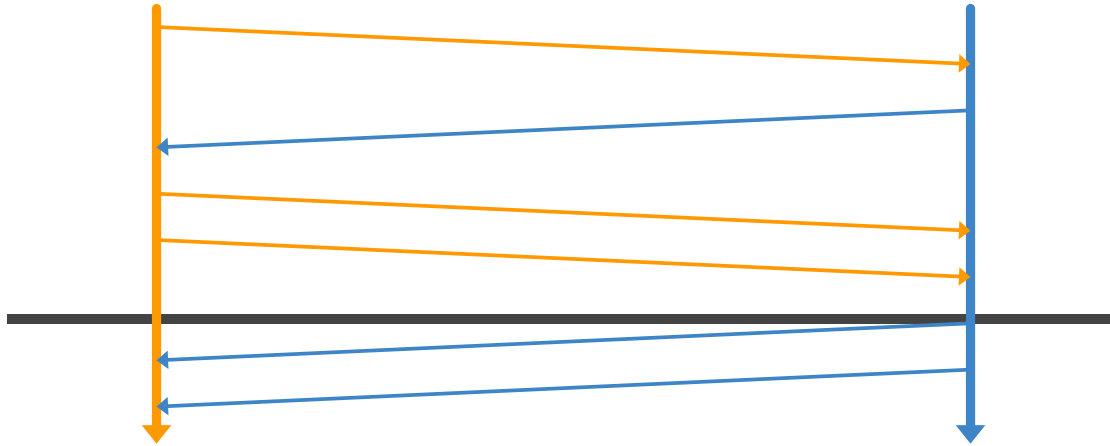
Broadcast + on-request data

Most J1939 messages are broadcast on the CAN-bus, though some data is only available by requesting the data via the CAN bus



PGN identifiers & SPN parameters

J1939 messages are identified by 18-bit Parameter Group Numbers (PGN), while J1939 signals are called Suspect Parameter Numbers (SPN)



Intel byte order & multi-packets

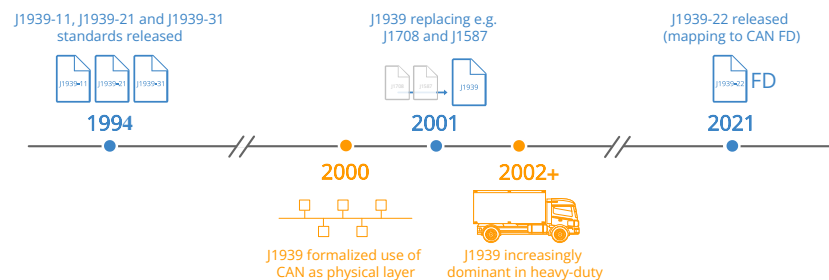
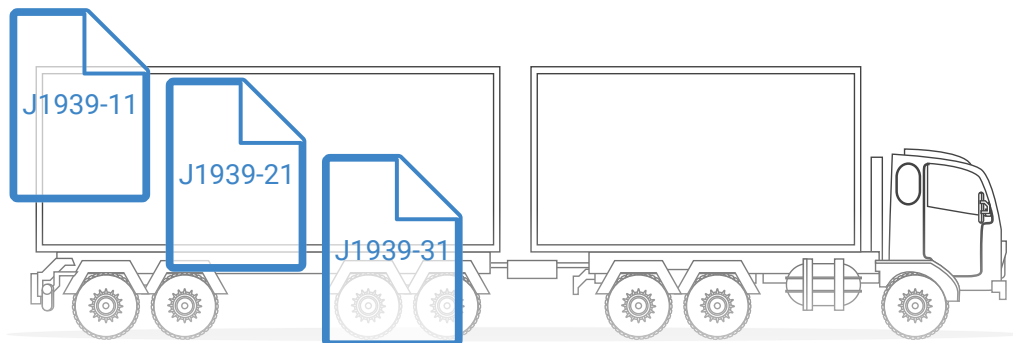
Multibyte signals are sent least significant byte first (Intel byte order). PGNs with up to 1785 bytes are supported via J1939 transport protocol

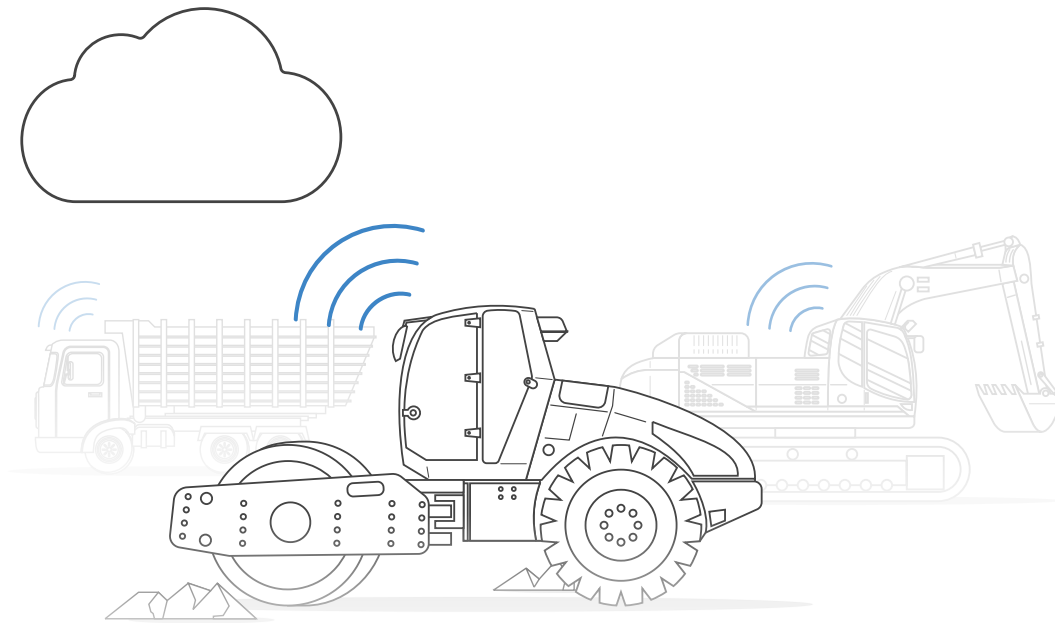
Additional J1939 characteristics

+

J1939 history

- 1985: SAE initiated development of the J1939 protocol
- 1994: First docs were released (J1939-11, J1939-21, J1939-31)
- 2000: The initial top level document was published
- 2000: CAN formally included as part of J1939 standard
- 2001: J1939 starts replacing former standards SAE J1708/J1587
- 2013: J1939 Digital Annex released, digitizing PGN/SPN data
- 2020-21: J1939-17 and J1939-22 released (J1939 on CAN FD)





J1939 future

We see a number of trends affecting the protocol:

- **Bandwidth challenge:** The need for more bandwidth may drive a transition towards J1939-22 (J1939 on CAN FD), increasing use of separate J1939 networks per vehicle and/or a potential transition towards Automotive Ethernet
- **Right to Repair:** The 'Right to Repair' movement is particularly relevant in expensive heavy-duty vehicles, incl. e.g. famously John Deere equipment and military vehicles. At the same time, OEMs are commercially motivated to increasingly offer closed J1939 telematics systems, which may drive a push towards the use of increasingly proprietary PGN/SPN encoding
- **J1939 EVs:** The increase in electric heavy-duty vehicles poses a risk to the J1939 standardization and thus e.g. mixed fleet telematics. This is both due to the absence of legal requirements for emissions measurements and the fact that OEM EV development may sometimes precede the introduction of new standardized J1939 PGN/SPN encoding

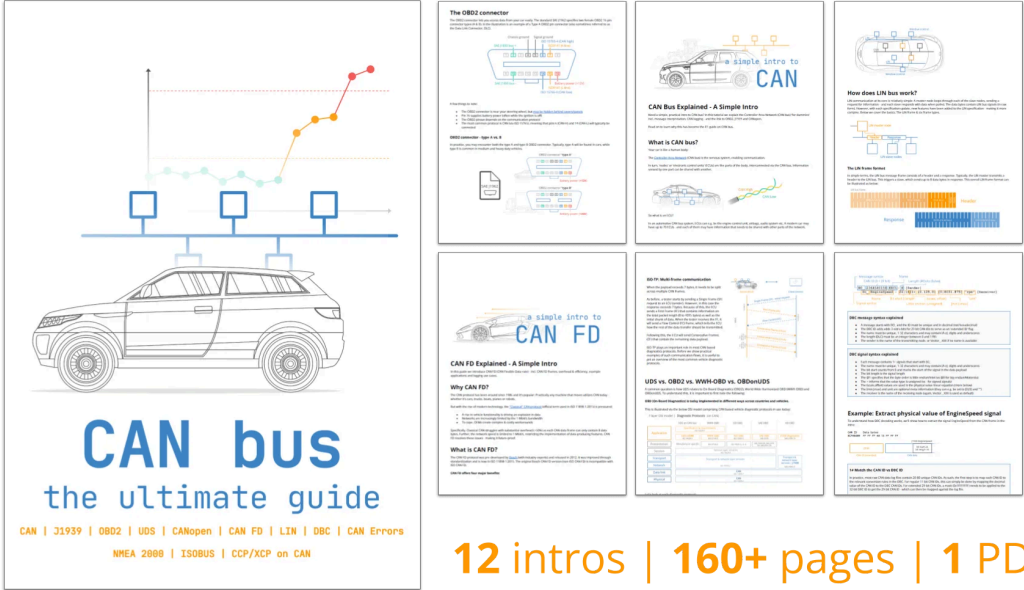
Get our 'Ultimate CAN Guide'

Want to become a CAN bus expert?

Get our 12 'simple intros' in one **160+ page PDF**:

- | | | |
|-----------|-------------|--------------|
| • CAN bus | • CANopen | • CAN FD |
| • J1939 | • NMEA 2000 | • LIN bus |
| • OBD2 | • CCP/XCP | • CAN errors |
| • UDS | • ISOBUS | • DBC files |

[download now](#)



12 intros | 160+ pages | 1 PDF

J1939 standards (higher-layer protocol)

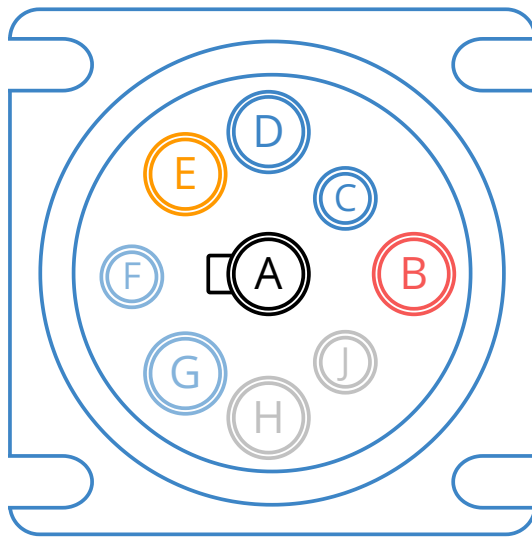
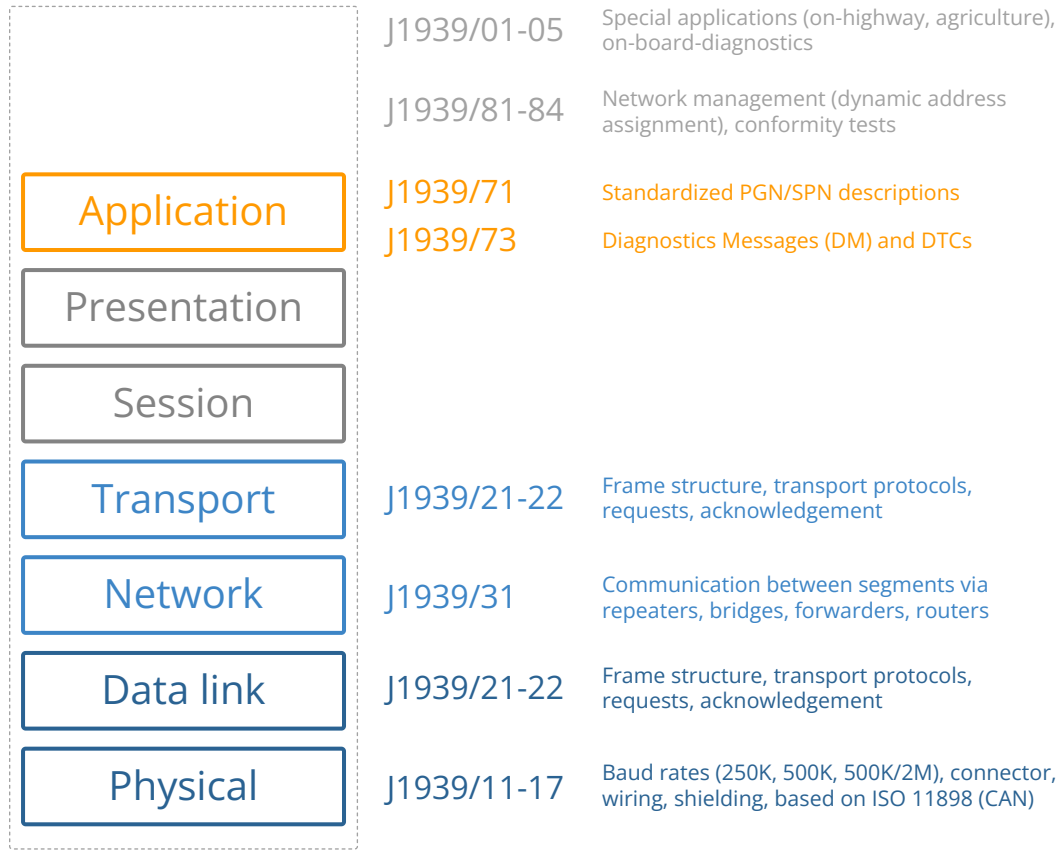
J1939 is based on CAN, which specifies the physical layer (ISO 11898-2) and data link layer (ISO 11898-1) of the OSI model.

Here, CAN is a 'lower-layer protocol' that specifies means of communication like wires and CAN frames - but not a lot more.

J1939 is a 'higher-layer protocol' that adds a specific language to enable more advanced communication. Other CAN based protocols exist like OBD2, UDS and CANopen.

To better understand J1939, we will explore some of the sub standards (or chapters) in the sections below.

7 layer OSI model | J1939 standards



- A** Ground
- B** Battery power
- C** CAN 1 H
- D** CAN 1 L
- E** CAN shield
- F** J1708 (+) / CAN 2 H
- G** J1708 (-) / CAN 2 L
- H** OEM specific
- J** OEM specific

Type 1 (black): CAN 1 = 250K

Type 2 (green): CAN 1 = 500K

The J1939 connector [J1939-13]

The J1939-13 standard specifies the 'off-board diagnostic connector' - also known as the J1939 connector or 9-pin Deutsch connector. This is a standardized method for interfacing with the J1939 network of most heavy duty vehicles - see the illustration for the J1939 connector pinout.

Multiple J1939 networks	+
Other heavy duty connectors	+

Black type 1 vs green type 2

Note that the J1939 deutsch connector comes in two variants: The original black connector (type 1) and the newer green connector (type 2), which started getting rolled out in 2013-14. Green type 2 J1939 adapter cables can be used on both black and green connectors as they are backwards compatible.

Why two connector types?

J1939 connector (9-pin deutsch)



The J1939 PGN and SPN [J1939-21/71]

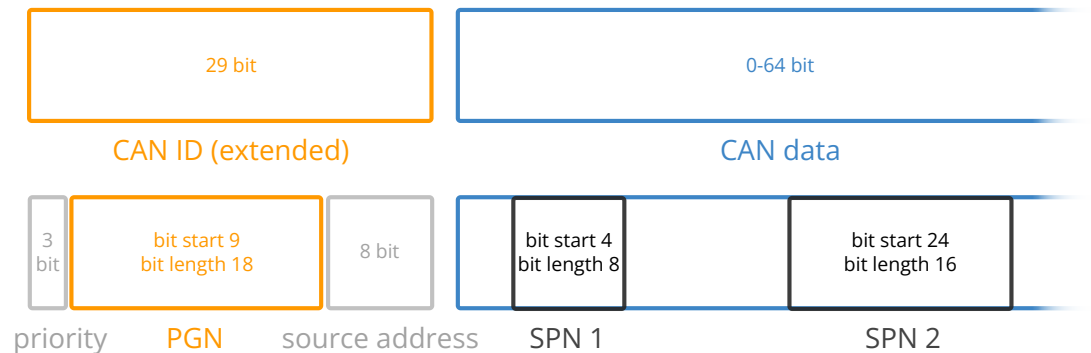
In the following section we explain the J1939 PGNs and SPNs.

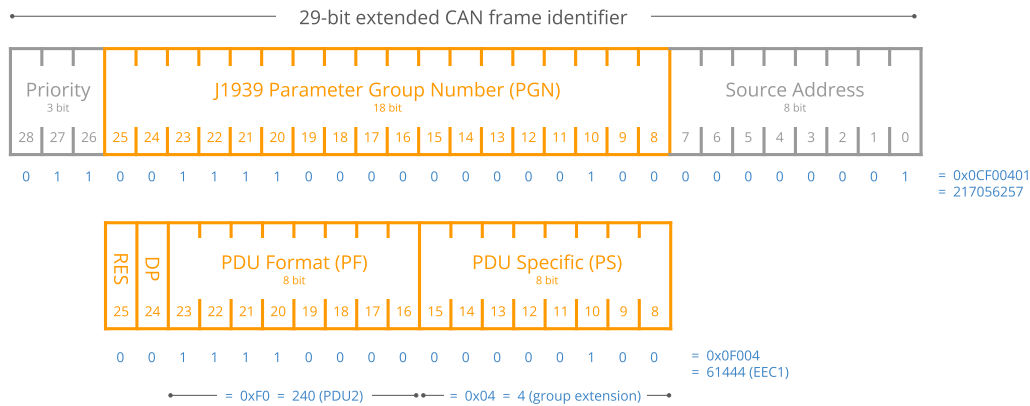
Parameter Group Number (PGN)

The **J1939 PGN** comprises an 18-bit subset of the 29-bit extended CAN ID. The PGN serves as the unique frame identifier within the J1939 standard - meaning that the rules for decoding raw J1939 data are specified at PGN level, rather than 29-bit ID level.

As a result, multiple CAN messages with unique CAN IDs can map to the same PGN - and be interpreted identically.

J1939 message (PGN & SPNs)





blue: Example values

RES: Reserved | DP: Data Page | PDU: Protocol Data Unit (message format)

PF < 240: Message is PDU1 (addressable message, PS contains destination address)
PF >= 240: Message is PDU2 (broadcast message, PS contains group extension)

Detailed breakdown of the J1939 PGN

Let's look at the CAN ID to PGN transition in detail.

Specifically, the 29 bit CAN ID comprises the Priority (3 bits), the J1939 PGN (18 bits) and the Source Address (8 bits). In turn, the PGN can be split into the Reserved Bit (1 bit), Data Page (1 bit), PDU format (8 bit) and PDU Specific (8 bit). PDU refers to Protocol Data Unit.

The detailed PGN illustration also includes example values for each field in binary, decimal and hexadecimal form.

Tip: CAN ID to J1939 PGN converter

Our online J1939 PGN converter lets you convert 29-bit CAN IDs to J1939 PGNs - and vice versa. The tool also lets you check if the J1939 PGN is included in our J1939 DBC file.

j1939 pgn converter →

number format: <input type="text" value="HEX"/>		CAN ID to PGN					PGN to CAN ID		PGN List	
CAN ID	P (Prio.)	R (Res.)	DP (Data Page)	PF (PDU Format)	PS (PDU Specific)	SA (Source Addr.)	PGN	PGN Label	In DBC?	
CF004FE	3	0	0	F0	4	FE	F004	Electronic Engine Controller 1	J1939 DBC	
98FEF4FE	6	0	0	FE	F4	FE	FEF4	Tire Condition Message 1	J1939 DBC	
18F111FE	6	0	0	F1	11	FE	F111	Gaseous Fuel Pressure 3	J1939 DBC	
CFE6CEE	3	0	0	FE	6C	EE	FE6C	Tachograph	J1939 DBC	
18FFA227	6	0	0	FF	A2	27	FFA2	Proprietary B	No	
18EBF900	6	0	0	EB	F9	0	EB00	Transport Protocol - Data Trans...	NMEA DBC	

J1939 PGN PDU1 vs. PDU2

+



Suspect Parameter Number (SPN)

The **J1939 SPN** serves as the identifier for the CAN signals (parameters) contained in the data payload. SPNs are grouped by PGNs and can be described in terms of their bit start position, bit length, scale, offset and unit - information required to extract and scale the SPN data to physical values.

Examples of SPNs include Engine Speed, Wheel Based Vehicle Speed, Fuel Level 1 and Engine Oil Temperature 2.

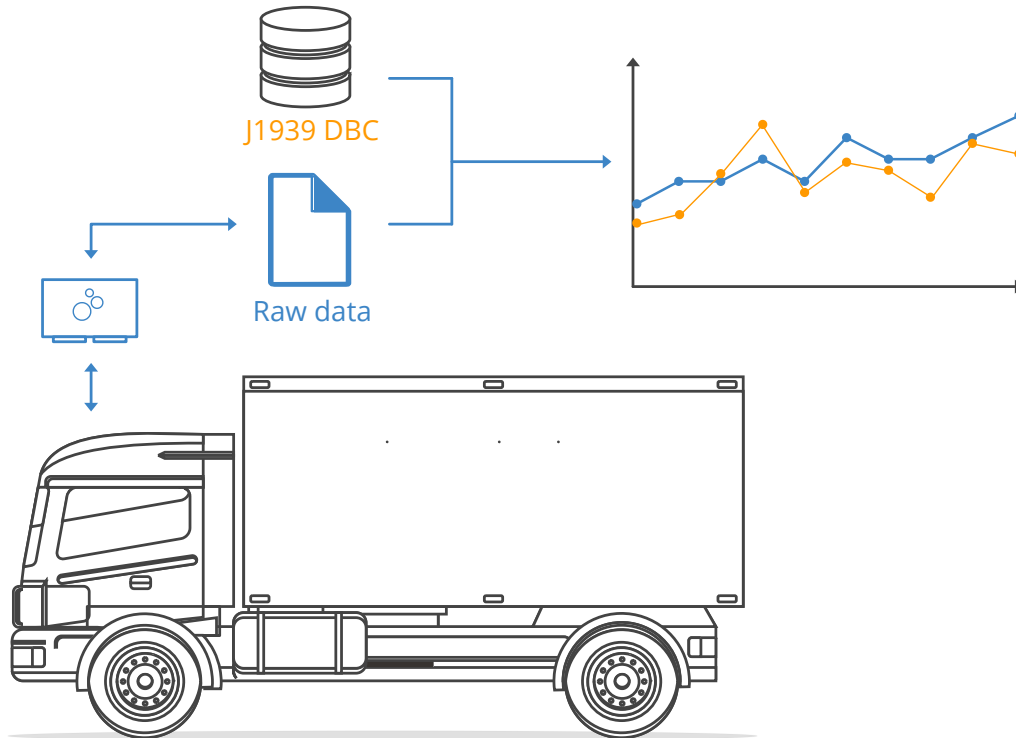
J1939 Digital Annex & J1939 DBC file

In 2013, the J1939 Digital Annex (DA) was introduced by SAE. This is an Excel file that contains all technical details on standard J1939 PGNs and SPNs, previously published in J1939-71.

Importantly, the J1939 DA only includes standardized PGNs/SPNs - not OEM specific proprietary PGNs/SPNs.

At CSS Electronics, we collaborate with SAE to transform the J1939 DA into a J1939 DBC file with 1,800+ PGNs and 10,000+ SPNs. This conversion is done quarterly with each new J1939 DA revision. The J1939 DBC file can be used in CAN software/API tools to enable easy decoding of raw J1939 data.

[j1939 dbc intro →](#)



How to decode raw J1939 data

As hinted above, practical decoding of J1939 data is done via CAN software/API tools and a J1939 DBC file. However, it is useful to understand what is happening under-the-hood.

Assume you have recorded a raw J1939 frame as below:

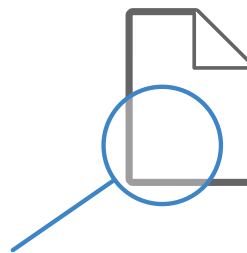
CAN ID	Data bytes
0x0CF00401	0xFF FF FF 68 13 FF FF FF

First, you need to determine the J1939 PGN, e.g. via our PGN converter. The 29-bit CAN ID 0x0CF00401 translates to J1939 PGN 0xF004 (61444) aka **EEC1** (Electronic Engine Controller 1).

From the J1939-71 DA, we find that the PGN EEC1 contains 8 SPNs - including **Engine Speed**.

J1939 PGN 61444 | EEC1

Name: **Electronic Engine Controller 1**
 Length: **8 bytes**
 Data page: **0**
 PDU format: **240**
 PDU specific: **4**
 Default priority: **3**

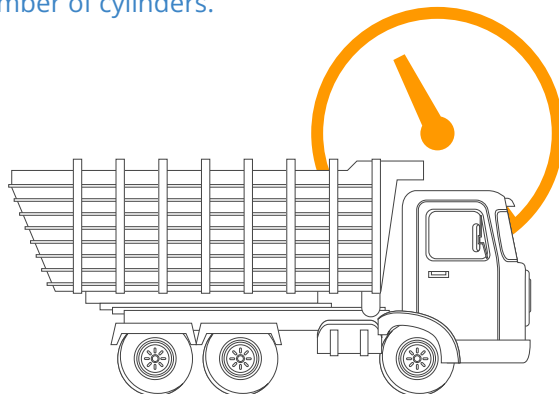


Bit Start	Bit Length	SPN ID	SPN name
0	4	899	Engine Torque Mode
4	4	4154	Actual Engine - Percent Torque (Fractional)
8	8	512	Driver's Demand Engine - Percent Torque
16	8	513	Actual Engine - Percent Torque
24	16	190	Engine Speed
40	8	1483	SA of Controlling Device for Engine Control
48	4	1675	Engine Starter Mode
56	8	2432	Engine Demand - Percent Torque

J1939 SPN 190 | Engine Speed

Actual engine speed which is calculated over a minimum crankshaft angle of 720 degrees divided by the number of cylinders.

Bit start: **24**
 Length: **2 bytes**
 Scale: **0.125**
 Offset: **0**
 Unit: **rpm**
 min-max: **0-8031.875**
 PGN reference: **61444**



Next, we will decode the value of Engine Speed. In the J1939 DA we can look up the decoding rules and follow the below steps:

1. **Extract raw bits:** The raw payload is in bytes 4 to 5, i.e. **0x6813**
2. **Use Intel:** Next, reverse the byte order to get **0x1368**
3. **Convert to decimal:** The raw payload in decimal form is **4968**
4. **Scale/offset:** Multiply by 0.125 and offset by 0 to get **621 RPM**

J1939 signal ranges

As per J1939-71, some signal values have special interpretations. If an ECU has a sensor error or lacks certain functionality entirely, the 'error range' or 'not available range' can be used to communicate this. You can see the practical use of 'not available' values in the below raw J1939 data example.

Valid vs. operational range	+
Regarding 2-bit signals	+

J1939 signal ranges (Valid, Error, Not Available)

Bit length	Valid range	Error range	Not available range
4	0x0 - 0xA	0xE	0xF
8	0x0 - 0xFA	0xFE	0xFF
10	0x0 - 0x3FA	0x3FE	0x3FF
12	0x0 - 0xFAF	0xFE0 - 0xFEF	0xFF0 - 0xFFF
16	0x0 - 0FAFF	0xFE00 - 0FEFF	0xFF00 - 0FFFF
20	0x0 - 0FAFFF	0xFE000 - 0FEFFF	0xFF000 - 0FFFFFF
24	0x0 - 0FAFFFF	0xFE0000 - 0FEFFFF	0xFF0000 - 0FFFFFFF
28	0x0 - 0FAFFFFF	0xFE00000 - 0FEFFFFFF	0xFF00000 - 0FFFFFFF
32	0x0 - 0FAFFFFFF	0xFE000000 - 0FEFFFFFFF	0xFF000000 - 0FFFFFFF

The ranges show extracted data byte values after reversing the byte order

Example: J1939 truck data - raw vs. decoded

Below we illustrate raw vs. decoded J1939 data through an example.

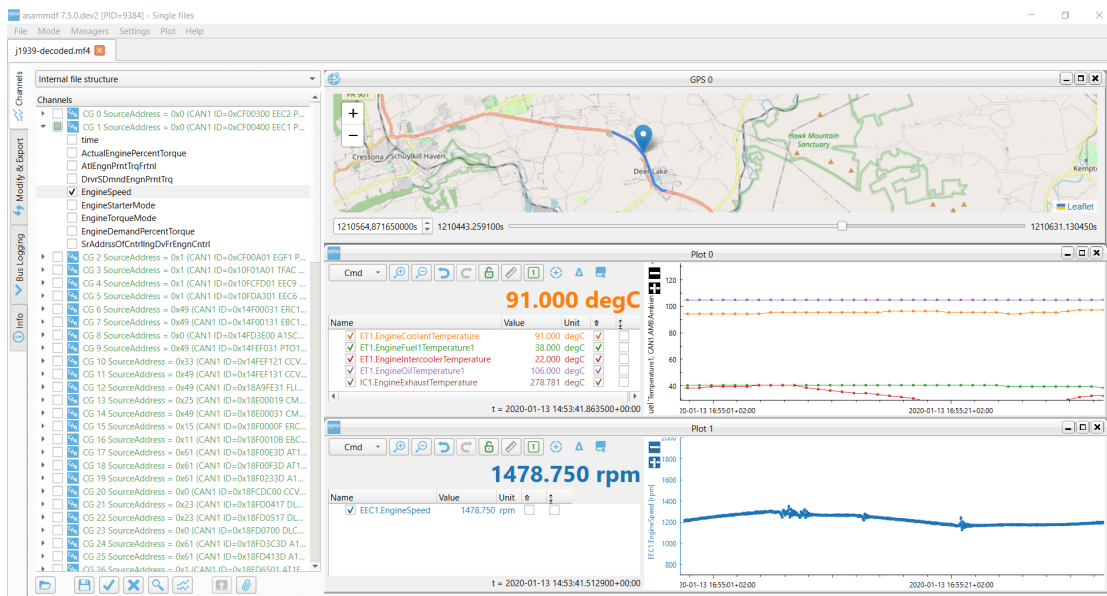
The two pictures show a log file recorded using a CANedge2 from a truck, visualized via the free asammdf GUI:

Raw J1939 data

Index	timestamps	Bus	ID	IDE	Direction	Name	Event Type	Details	ESI	EDL	BRS	DLC	Data Length	Data Bytes
259	2020-01-13 15:53:41.872900+01:00	CAN 1	0x0CF00400	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	00 70 70 51 2E 00 F4 7D
260	2020-01-13 15:53:41.873450+01:00	CAN 1	0x10FEF000	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	FF FF FF 00 00 F0 00 FF
261	2020-01-13 15:53:41.874100+01:00	CAN 1	0x10FEA400	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	FF FF FF FF 52 7F 20 22
262	2020-01-13 15:53:41.877050+01:00	CAN 1	0x0CF00421	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	FF FF FF FF FF FF FF FF
263	2020-01-13 15:53:41.879400+01:00	CAN 1	0x0CF00A01	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	FS 10 8F 80 FF FF FF FF
264	2020-01-13 15:53:41.879950+01:00	CAN 1	0x10F01A01	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	00 00 00 FF FF FF FF FF FF
265	2020-01-13 15:53:41.880600+01:00	CAN 1	0x10F00E3D	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	A0 0F FF FF FF FF FF FF
266	2020-01-13 15:53:41.881150+01:00	CAN 1	0x18F00F3D	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	18 10 FF FF FF FF FF FF
267	2020-01-13 15:53:41.881750+01:00	CAN 1	0x18F08C3D	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	78 00 48 00 FF FF FF FF
268	2020-01-13 15:53:41.882300+01:00	CAN 1	0x10FF2121	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	72 55 1F C0 C0 FF FF E1
269	2020-01-13 15:53:41.882850+01:00	CAN 1	0x18F0823D	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	FF FF D4 4B 34 00 FF FF
270	2020-01-13 15:53:41.883400+01:00	CAN 1	0x0CF00400	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	00 70 70 5A 2E 00 F4 7D
271	2020-01-13 15:53:41.884000+01:00	CAN 1	0x18F08300	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	FF FF CC 4F FF FF FF FF
272	2020-01-13 15:53:41.884600+01:00	CAN 1	0x14FD3E00	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	A0 4E FF A6 51 FF FF FF
273	2020-01-13 15:53:41.885150+01:00	CAN 1	0x18F08400	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	FB 44 FB 44 FF FF FF FF
274	2020-01-13 15:53:41.885700+01:00	CAN 1	0x18FEE000	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	1A B9 38 00 1A B9 38 00
275	2020-01-13 15:53:41.887150+01:00	CAN 1	0x14FD4421	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	FF FF FF FF FF FF FF FF
276	2020-01-13 15:53:41.893050+01:00	CAN 1	0x0CF00400	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	00 70 70 56 2E 00 F4 7D
277	2020-01-13 15:53:41.893650+01:00	CAN 1	0x0CF00300	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	00 00 00 00 FF FF FF FF
278	2020-01-13 15:53:41.894250+01:00	CAN 1	0x0C000000	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	FC FF FA FA FF FF FF FF
279	2020-01-13 15:53:41.894800+01:00	CAN 1	0x18FEF100	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	C3 14 54 10 00 00 00 30
280	2020-01-13 15:53:41.895400+01:00	CAN 1	0x18FED000	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	00 00 F0 00 00 FF FF FF FF
281	2020-01-13 15:53:41.897250+01:00	CAN 1	0x14FEF031	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	FF FF FF FF FF FC 00 FF
282	2020-01-13 15:53:41.902950+01:00	CAN 1	0x0CF00400	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	00 70 70 4F 2E 00 F4 7D
283	2020-01-13 15:53:41.903550+01:00	CAN 1	0x18FEF000	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	86 00 13 FF 7D AA 19 FF
284	2020-01-13 15:53:41.904150+01:00	CAN 1	0x18FF7400	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	06 00 00 FF FF FF FC FF FF
285	2020-01-13 15:53:41.904950+01:00	CAN 1	0x0CF08281	0x01	RX		CAN Frame			Standard CAN	0	0x08	0x08	44 03 7E 01 D2 00 61 00

The raw J1939 data is comprised of timestamped 29-bit CAN IDs and 8-byte data payloads. This data can be filtered and searched through, but cannot e.g. be plotted as time series

Decoded J1939 data



By decoding the raw J1939 data with a J1939 DBC file, we match 85 of 142 CAN IDs (the rest is proprietary). The decoded file contains J1939 SPNs grouped by PGNs - ready for analysis

Want to try this yourself? Download asammdf, our J1939 sample data and J1939 DBC demo:

[try it now →](#)



CANedge: J1939 data logger

The CANedge lets you easily record J1939 data to an 8-32 GB SD card. Simply connect it to e.g. a truck to start logging - and decode the data via free software/APIs and our J1939 DBC.

[j1939 logger intro](#) [canedge →](#)

Request message [J1939-21]

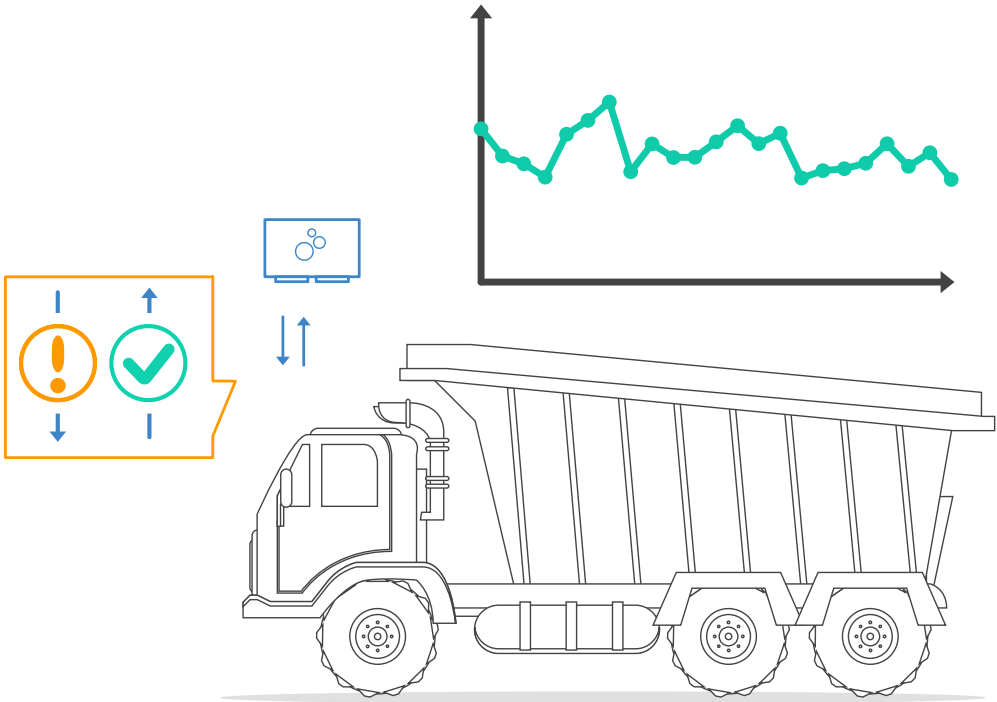
Most J1939 messages are transmitted on the CAN bus at a fixed periodic rate, but some are only transmitted 'on-request' (e.g. when polled by a diagnostic tool or J1939 data logger).

A common example includes J1939 diagnostic messages like the DM2, which contains diagnostic trouble codes (DTCs). See also our J1939-73 DBC file.

To send a J1939 request via the CAN bus, a special 'request message' is used (**PGN 59904**), which is the only J1939 message with just 3 data bytes. The data bytes contain the requested PGN in Intel byte order.

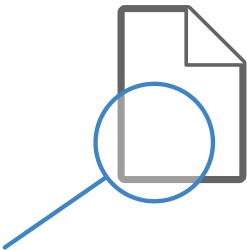
To show how this works, let us consider a real-life example. Here, an engineer is using the CANedge to request data on the PGN HOURS (0xFEE5 i.e. 65253), which includes the SPN 'Engine Total Hours of Operation' (often used in e.g. telematics).

To send the request, the engineer configures the CANedge to transmit a CAN frame with a payload of 0xE5FE00 (the HOURS PGN in Intel byte order).



J1939 PGN 59904 | RQST

Name: Request
Length: 3 bytes
Data page: 0
PDU format: 234
PDU specific: Destination address
Default priority: 6



Bit Start	Bit Length	SPN ID	SPN name
0	24	899	PGN to be requested

For the 29-bit CAN ID we illustrate two cases:

- **Example 1:** Here the destination address is 0xFF (a 'global request'), forcing all ECUs to respond if they have data
- **Example 2:** Here the destination address is 0x00 (a 'specific request'), ensuring only the ECU with SA 0x00 responds

J1939 PGN 59904 (Request): Requesting data for PGN 65253 (Engine Hours)

Example 1: Global request (PS = 0xFF)

Time	CAN ID	PGN (HEX)	PGN (DEC)	DataBytes
1.0135	18EAFFFA	EA00	59904	E5 FE 00
1.0228	18FEE500	FEE5	65253	BA 00 00 00 4F 02 00 00

Example 2: Destination specific request (PS = 0x00)

Time	CAN ID	PGN (HEX)	PGN (DEC)	DataBytes
1.0135	18EA00FA	EA00	59904	E5 FE 00
1.0228	18FEE500	FEE5	65253	BA 00 00 00 4F 02 00 00

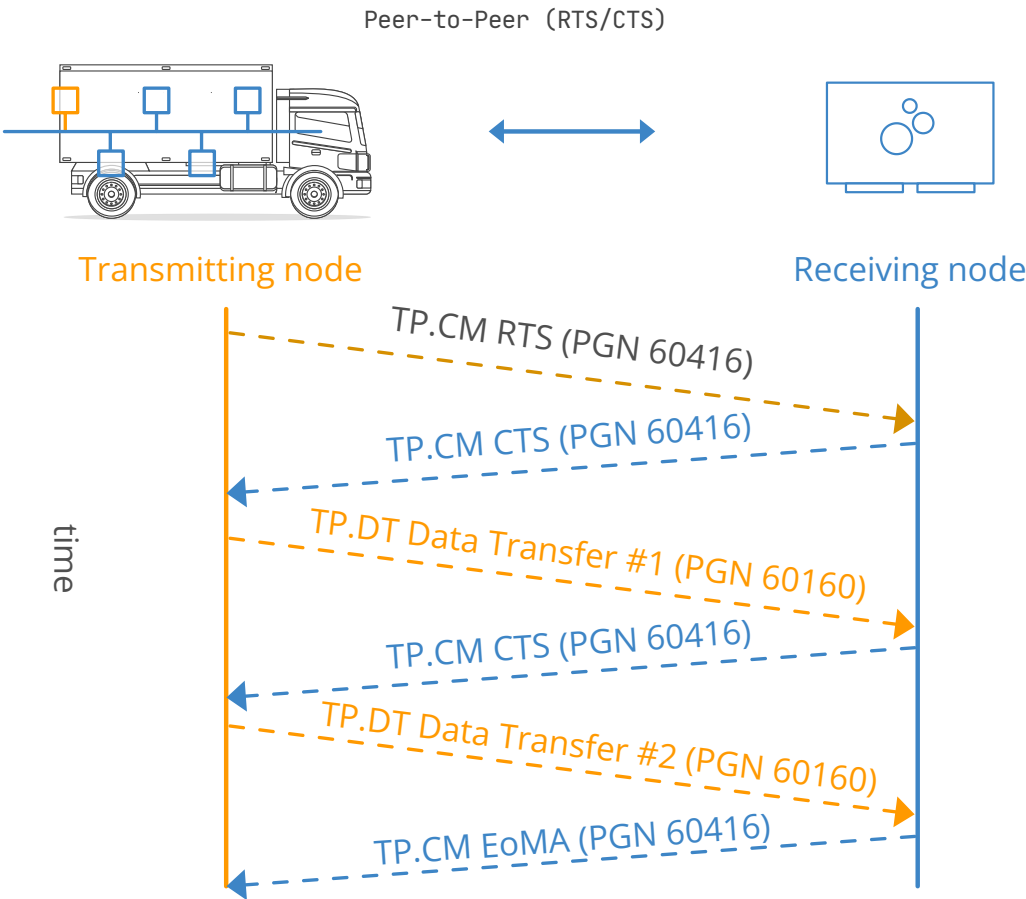
Legend
requested J1939 PGN
(0x00FEE5 = 65253)
payload data (8 bytes)
SA of requesting node
(0xFA = arbitrarily selected)
destination address
(0xFF = global, 0x00 = node SA)
SA of responding node

Details on the 29-bit request CAN ID	+
J1939 requests and warranty compliance	+

J1939 transport protocol (TP) [J1939-21]

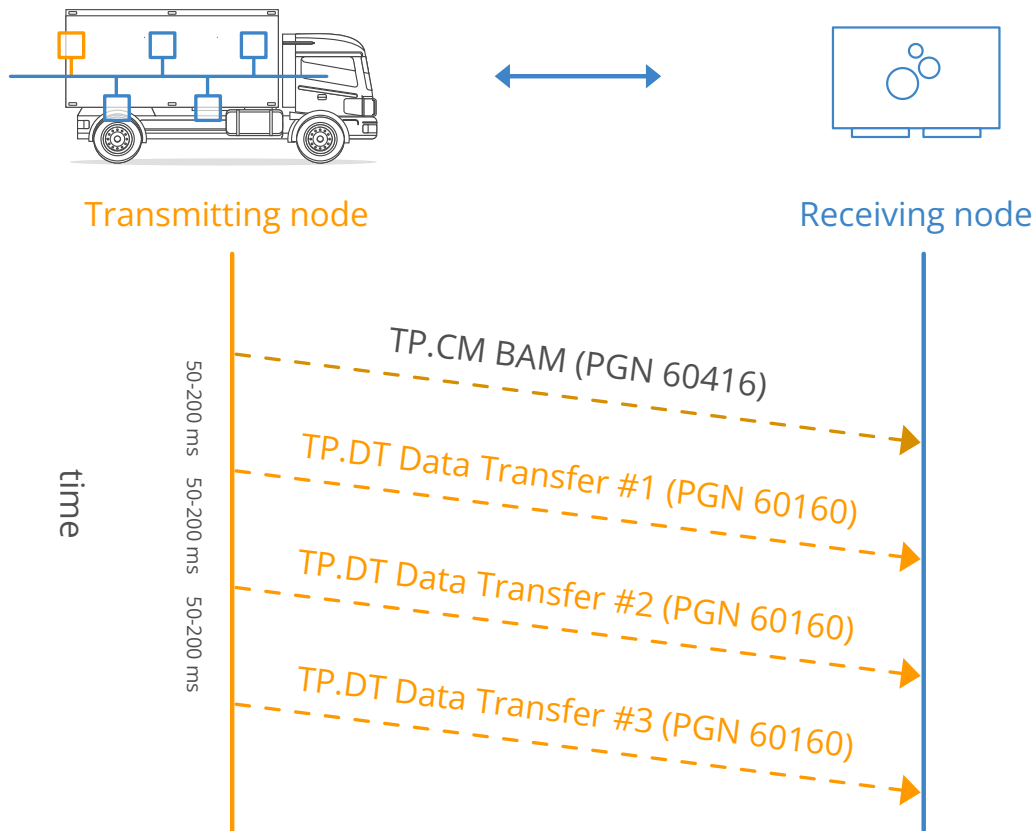
Some message payloads exceed 8 bytes - for example ECU software updates, vehicle configurations or diagnostic trouble codes (DM1). To communicate such payloads it is necessary to split the data across multiple CAN frames. The receiving node must then subsequently reassemble the individual data packets.

J1939 specifies how this can be achieved through two alternative transport protocols:



The transmitting node initiates a connection via a Request To Send (RTS) message. Subsequent communication is controlled by the receiver through Clear To Send (CTS) messages, ending with a End of Message Acknowledge (EoMA) message.

Broadcast (BAM)



The transmitting node initiates the communication with a Broadcast Announce Message (BAM, PGN 0xEC00 i.e. 60416) followed by a number of Data Transfer (DT, PGN 0xEB00 i.e. 60160) messages. The receiving node does not exert any control.

Example: J1939 BAM transport protocol incl. reassembly

The J1939 transport protocol uses two PGNs:

- **TP.CM:** TP - Connection Management (0xEC00 i.e. 60416)
- **TP.DT:** TP - Data Transfer (0xEB00 i.e. 60160)

The TP.CM payload contains a control byte that differs depending on the type of message (it is e.g. 0x20 for the BAM message). The payload also contains the number of data bytes and packets to be sent and the PGN of the multi-packet message (Intel byte order)

In the BAM protocol, the TP.CM message is followed by up to 255 packets of data in TP.DT messages. The TP.DT message uses the 1st byte as a sequence counter (1 to 255), while the remaining 7 bytes contain the segmented data payload. As a result, a single J1939 TP sequence can carry up to $7 \times 255 = 1785$ data bytes.

J1939 PGN 60416 | TP.CM

Name: Transport Protocol - Connection Management
Length: 8 bytes

Bit Start	Bit Length	SPN name
0	8	Control Byte
8	16	Message Size
24	8	Number of Packets
32	8	Reserved (0xFF)
40	24	PGN of transported message

Control Byte values
0x10 Request to Send (RTS)
0x11 Clear to Send (CTS)
0x13 End of Message Acknowledge
0x14 Connection Abort
0x20 Broadcast Announce Message (BAM)

J1939 PGN 60160 | TP.DT

Name: Transport Protocol - Data transfer
Length: 8 bytes

Bit Start	Bit Length	SPN name
0	8	Sequence Counter (1 to 255)
8	56	Data Payload (unused = 0xFF)

Below we show a real-life trace of a J1939 transport protocol (BAM) sequence with data from the PGN EC1:

Timestamp (Epoch)	CAN ID	PGN (HEX)	PGN (DEC)	DataBytes	legend
1605078459.438750	1CECF00	EC00	60416	20270006FFE3FE00	control byte (0x20=BAM)
1605078459.498750	1CEBFF00	EB00	60160	013011B2A041B240	#data bytes (0x0027=39)
1605078459.559750	1CEBFF00	EB00	60160	021FB2102CB2603B	#packets (0x06=6)
1605078459.618750	1CEBFF00	EB00	60160	03B230430000D309	reserved
1605078459.678750	1CEBFF00	EB00	60160	04C0441E37967DE1	J1939 PGN (0x00FEE3=65251)
1605078459.738750	1CEBFF00	EB00	60160	05E02E7B02FFFF80	sequence numbers
1605078459.799850	1CEBFF00	EB00	60160	06E0FFFFFFFFFFFF	payload data (39 bytes)
					unused (3 bytes)
1605078459.438750	18FEE300	FEE3	65251	3011B2A041B240 ... E0FFFFFF	

To interpret the EC1 data, the BAM protocol multi-packet data payload has to be reassembled:

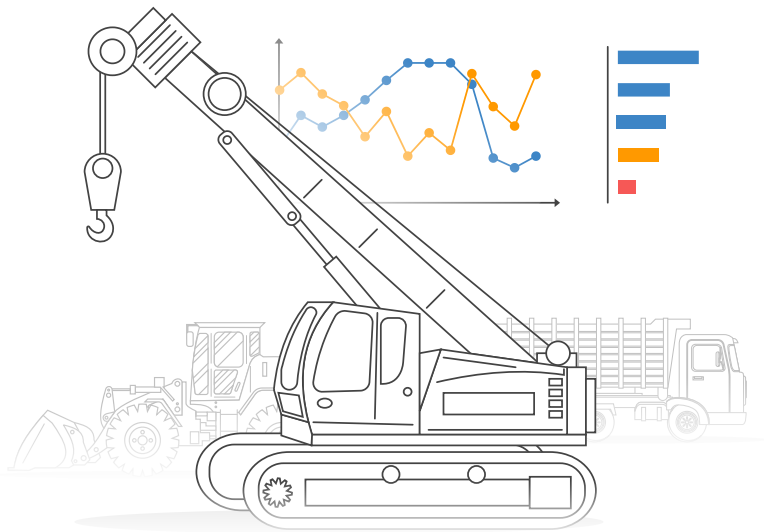
- Identify the BAM frame (PGN 60416, control byte 0x20), indicating a new sequence being initiated
- Extract the PGN from the last 3 bytes of the BAM payload
- Convert the J1939 PGN into a 29-bit ID and use it as identifier of the reassembled frame
- Construct the reassembled data payload by concatenating the last 7 bytes of the data transfer frames

The reassembly and subsequent DBC decoding can be done via e.g. the CANedge software/API tools.

Comments on the transport protocol trace	+
Comments on DBC decoding J939 transport protocol messages	+

J1939 data logging - example use cases

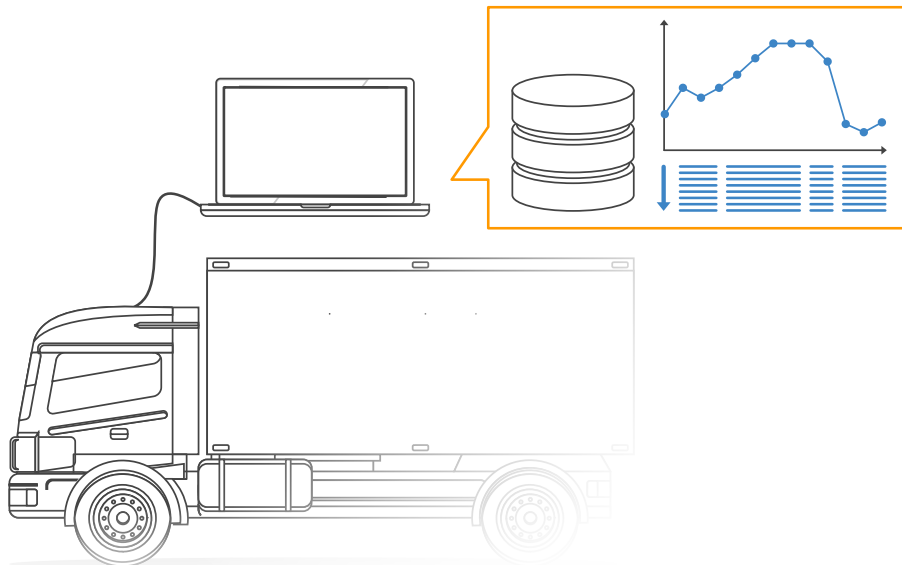
There are several common use cases for recording J1939 data:



Heavy duty fleet telematics

J1939 data from trucks, buses, tractors etc. can be used in fleet management to reduce costs or improve safety

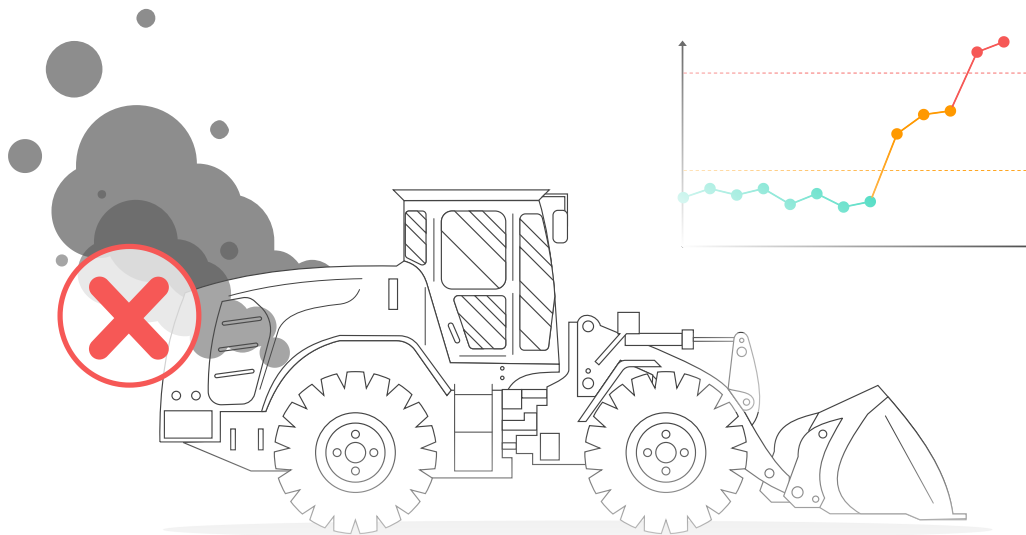
j1939 telematics →



Live stream diagnostics

By streaming decoded J1939 data to a PC, technicians can perform real-time J1939 diagnostics on vehicles

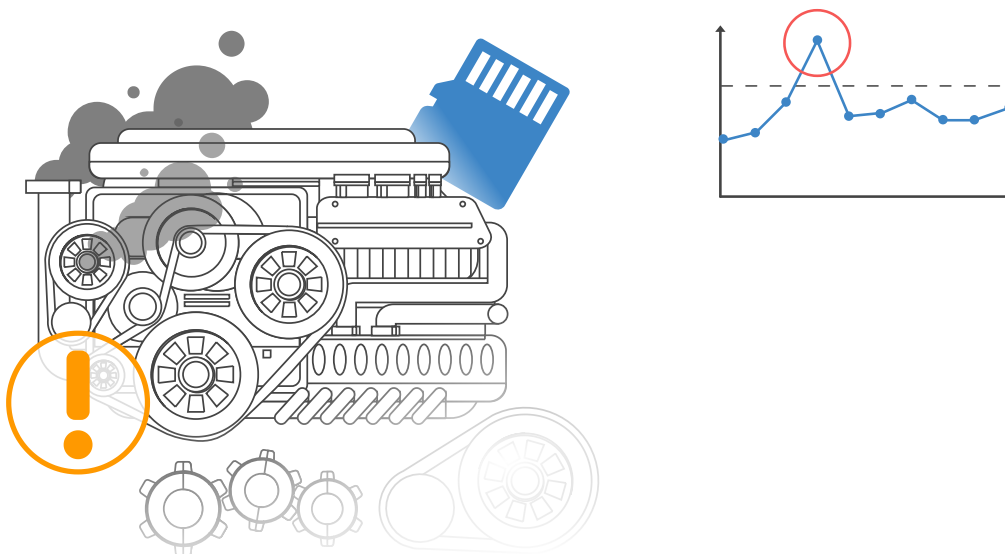
j1939 streaming →



Predictive maintenance

Vehicles can be monitored via WiFi CAN loggers in the cloud to predict breakdowns based on the J1939 data

[predictive maintenance →](#)



Heavy-duty vehicle blackbox

A CAN logger can serve as a 'blackbox' for heavy-duty vehicles, providing data for e.g. disputes or J1939 diagnostics

[can bus blackbox →](#)

Do you have a J1939 data logging use case? Reach out for free sparring!

[contact us →](#)

For more intros, see our guides section - or download the '**Ultimate Guide**' PDF.

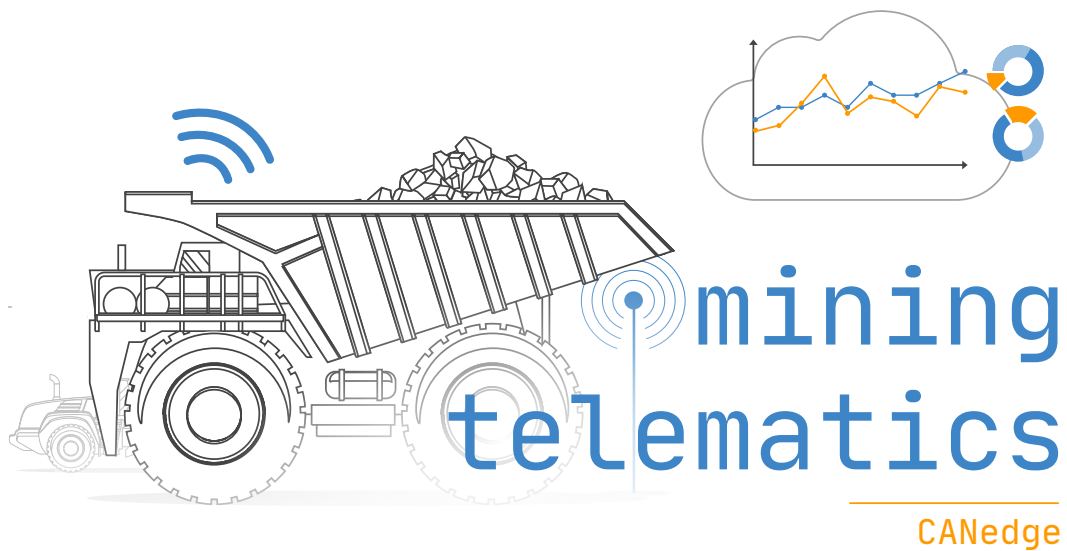
Need to log/stream J1939 data?

Get your CAN logger today!

Recommended for you



J1939 VEHICLE TELEMATICS



MINING TELEMATICS & DASHBOARDS



J1939-73 - A SIMPLE INTRO

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

[subscribe](#)



© 2025, CSS Electronics