

INTERNATIONAL
STANDARD

ISO
22900-2

Third edition
2022-06

Road vehicles — Modular vehicle communication interface (MVCI) —

**Part 2:
Diagnostic protocol data unit (D-PDU API)**

iTeh STANDARD PREVIEW
Véhicules routiers — Interface de communication modulaire du véhicule (MVCI) —
(standards.iteh.ai)
Partie 2: Interface de programmation d'application d'unité de données du protocole de diagnostic (D-PDU API)

[ISO 22900-2:2022](#)

<https://standards.iteh.ai/catalog/standards/sist/5e549e42-182c-407f-a0cc-504a0f96866b/iso-22900-2-2022>



Reference number
ISO 22900-2:2022(E)

© ISO 2022

iTeh STANDARD PREVIEW (standards.iteh.ai)

[ISO 22900-2:2022](#)

<https://standards.iteh.ai/catalog/standards/sist/5e549e42-182c-407f-a0cc-504a0f96866b/iso-22900-2-2022>



COPYRIGHT PROTECTED DOCUMENT

© ISO 2022

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

	Page
Foreword.....	vi
Introduction.....	vii
1 Scope	1
2 Normative references	1
3 Terms, definitions and abbreviated terms.....	1
3.1 Terms and definitions.....	2
3.2 Abbreviated terms	3
4 Specification release version information	6
4.1 Specification release version location	6
4.2 Specification release version.....	6
5 Modular VCI use cases.....	6
5.1 OEM merger	6
5.2 OEM cross vehicle platform ECU(s)	6
5.3 Central source diagnostic data and exchange during ECU development	7
5.4 OEM franchised dealer and aftermarket service outlet diagnostic tool support	7
6 Modular VCI software architecture.....	7
6.1 Overview.....	7
6.2 Modular VCI D-Server software	8
6.3 Runtime format based on ODX	9
6.4 MVCI protocol module software	9
6.5 MVCI protocol module configurations	9
7 D-PDU API use cases	10
7.1 Overview	10
7.2 Use case 1: Single MVCI protocol module.....	11
7.3 Use case 2: Multiple MVCI protocol modules supported by same D-PDU API implementation.....	12
7.4 Use case 3: Multiple MVCI protocol modules supported by different D-PDU API implementations.....	13
8 Diagnostic protocol data unit (D-PDU) API	14
8.1 Software requirements.....	14
8.1.1 General requirements.....	14
8.1.2 Vehicle protocol requirements.....	15
8.1.3 Timing requirements for protocol handler messages	16
8.1.4 Serialization requirements for protocol handler messages	17
8.1.5 Compatibility requirements	19
8.1.6 Timestamp requirements.....	19
8.2 API function overview and communication principles	20
8.2.1 Terms used within the D-PDU API.....	20
8.2.2 Function overview.....	20
8.2.3 General usage.....	21
8.2.4 Asynchronous and synchronous communication.....	24
8.2.5 Usage of resource locking and resource unlocking.....	25
8.2.6 Usage of ComPrimitives.....	25
8.3 Tool integration	42
8.3.1 Requirement for generic configuration.....	42
8.3.2 Tool integrator — Use case	42
8.4 API functions — Interface description	44

8.4.1	Overview	44
8.4.2	PDUConstruct	44
8.4.3	PDUDestruct.....	45
8.4.4	PDUIoCtl	46
8.4.5	PDUGetVersion.....	48
8.4.6	PDUGetStatus.....	49
8.4.7	PDUGetLastError	50
8.4.8	PDUGetResourceStatus	51
8.4.9	PDUCreateComLogicalLink.....	52
8.4.10	PDUDestroyComLogicalLink.....	55
8.4.11	PDUConnect.....	57
8.4.12	PDUDisconnect.....	59
8.4.13	PDULockResource.....	60
8.4.14	PDUUnlockResource	61
8.4.15	PDUGetComParam	62
8.4.16	PDUSetComParam.....	69
8.4.17	PDUSTartComPrimitive	72
8.4.18	PDUCancelComPrimitive	76
8.4.19	PDUGetEventItem	77
8.4.20	PDUDestroyItem.....	78
8.4.21	PDURegisterEventCallback	79
8.4.22	EventCallback prototype	81
8.4.23	PDUGetObjectId	83
8.4.24	PDUGetModuleIds	84
8.4.25	PDUGetResourceIds	86
8.4.26	PDUGetConflictingResources	87
8.4.27	PDUGetUniqueRespIdTable	88
8.4.28	PDUSetUniqueRespIdTable.....	90
8.4.29	PDUModuleConnect.....	95
8.4.30	PDUModuleDisconnect.....	98
8.4.31	PDUGetTimestamp	99
8.5	I/O control section	99
8.5.1	IOCTL API command overview	99
8.5.2	PDU_IOCTL_RESET	102
8.5.3	PDU_IOCTL_CLEAR_TX_QUEUE.....	102
8.5.4	PDU_IOCTL_SUSPEND_TX_QUEUE	103
8.5.5	PDU_IOCTL_RESUME_TX_QUEUE.....	103
8.5.6	PDU_IOCTL_CLEAR_RX_QUEUE	104
8.5.7	PDU_IOCTL_CLEAR_TX_QUEUE_PENDING	104
8.5.8	PDU_IOCTL_READ_VBATT	105
8.5.9	PDU_IOCTL_SET_PROG_VOLTAGE	105
8.5.10	PDU_IOCTL_READ_PROG_VOLTAGE	106
8.5.11	PDU_IOCTL_GENERIC	107
8.5.12	PDU_IOCTL_SET_BUFFER_SIZE	108
8.5.13	PDU_IOCTL_GET_CABLE_ID	108
8.5.14	PDU_IOCTL_START_MSG_FILTER	109
8.5.15	PDU_IOCTL_STOP_MSG_FILTER	111
8.5.16	PDU_IOCTL_CLEAR_MSG_FILTER	111
8.5.17	PDU_IOCTL_SET_EVENT_QUEUE_PROPERTIES	112
8.5.18	PDU_IOCTL_SEND_BREAK	113
8.5.19	PDU_IOCTL_READ_IGNITION_SENSE_STATE	113
8.5.20	PDU_IOCTL_VEHICLE_ID_REQUEST	114
8.5.21	PDU_IOCTL_SET_ETH_SWITCH_STATE	117
8.5.22	PDU_IOCTL_GET_ENTITY_STATUS	118

8.5.23	PDU_IOCTL_GET_DIAGNOSTIC_POWER_MODE	119
8.5.24	PDU_IOCTL_GET_ETH_PIN_OPTION	120
8.5.25	PDU_IOCTL_TLS_SET_CERTIFICATE	121
8.5.26	PDU_IOCTL_DOIP_GET_CURRENT_SESSION_MODE	122
8.5.27	PDU_IOCTL_ISOBUS_GET_DETECTED_CFS	123
8.6	API functions — Error handling	123
8.6.1	Synchronous error handling	123
8.6.2	Asynchronous error handling	123
8.7	Installation	124
8.7.1	Generic description	124
8.7.2	Windows installation process	124
8.7.3	Linux installation process	125
8.7.4	Selecting MVCI protocol modules	126
8.8	Application notes	126
8.8.1	Interaction with the MDF	126
8.8.2	Accessing additional hardware features for MVCI protocol modules	126
8.8.3	Documentation and information provided by MVCI protocol module vendors	126
9	Using the D-PDU API with existing applications	127
9.1	SAE J2534-1 and RP1210a existing standards	127
10	Data structures	128
10.1	API functions — Data structure definitions	128
10.1.1	Abstract basic data types	128
10.1.2	Definitions	129
10.1.3	Bit encoding for UNUM32	129
10.1.4	API data structures	129
Annex A (normative)	D-PDU API compatibility mappings	145
Annex B (normative)	D-PDU API standard ComParams and protocols	146
Annex C (informative)	D-PDU API manufacturer-specific ComParams and protocols	235
Annex D (informative)	D-PDU API constants	237
Annex E (informative)	Application defined tags	253
Annex F (informative)	RDF and MDF description files	254
Annex G (informative)	Resource handling scenarios	325
Annex H (informative)	D-PDU API partitioning	331
Annex I (informative)	Use case scenarios	335
Annex J (informative)	OBD protocol initialization	376
Annex K (normative)	DoIP implementation	392
Annex L (normative)	ISOBUS	425
Bibliography	433	

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 31, *Data communication*.

ISO 22900-2:2022
22900-2-2022

This third edition cancels and replaces the second edition (ISO 22900-2:2017), which has been technically revised.

The main changes are as follows:

- introduction of reference to ISO 15765-5 for CAN-FD;
- introduction of secured communication for the DoIP protocol based on TLS (as defined in ISO 13400-2:2019);
- introduction of automatic DoIP reconnect handling;
- introduction of ISOBUS protocol.

A list of all parts in the ISO 22900 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

The ISO 22900 series is applicable to vehicle electronic control module diagnostics and programming.

This document was established in order to more easily exchange software and hardware of vehicle communication interfaces (VCIs) among diagnostic applications. It defines a generic and protocol independent software interface towards the modular vehicle communication interface (MVCI) protocol module, such that a diagnostic application based on this software interface can exchange the MVCI protocol module or add a new MVCI protocol module with minimal effort. Today, the automotive aftermarket requires flexible usage of different protocol modules for vehicles of different brands. Many of today's protocol modules are incompatible with regard to their hardware and software interface, such that, depending on the brand, a different protocol module is required.

The objective of this document is to specify the diagnostic protocol data unit application programming interface (D-PDU API) as a generic software interface and to provide a "plug and play" concept for access onto different MVCI protocol modules from different tool manufacturers. The D-PDU API will address the generic software interface, the protocol abstraction, its exchangeability, as well as the compatibility towards existing standards such as SAE J2534-1 and RP1210a.

The implementation of the modular VCI concept facilitates co-existence and re-use of MVCI protocol modules, especially in the aftermarket. As a result, diagnostic or programming applications can be adapted for different vehicle communication interfaces and different vehicles with minimal effort, thus helping to reduce overall costs for the tool manufacturer and end user.

Vehicle communication interfaces compliant with ISO 22900 series support a protocol-independent D-PDU API as specified in this document.

[ISO 22900-2:2022](#)

<https://standards.iteh.ai/catalog/standards/sist/5e549e42-182c-407f-a0cc-504a0f96866b/iso-22900-2-2022>

Road vehicles — Modular vehicle communication interface (MVCI) —

Part 2: Diagnostic protocol data unit (D-PDU API)

1 Scope

This document specifies the diagnostic protocol data unit application programming interface (D-PDU API) as a modular vehicle communication interface (MVCI) protocol module software interface and common basis for diagnostic and reprogramming software applications.

This document covers the descriptions of the application programming interface (API) functions and the abstraction of diagnostic protocols, as well as the handling and description of MVCI protocol module features. Sample MVCI module description files accompany this document.

The purpose of this document is to ensure that diagnostic and reprogramming applications from any vehicle or tool manufacturer can operate on a common software interface and can easily exchange MVCI protocol module implementations.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

SAE J2411, *Single Wire CAN Network for Vehicle Applications*

3 Terms, definitions and abbreviated terms

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

- ISO Online browsing platform: available at <https://www.iso.org/obp>
- IEC Electropedia: available at <https://www.electropedia.org/>

3.1 Terms and definitions

3.1.1

application

way of accessing the diagnostic protocol data unit application programming interface (D-PDU API)

Note 1 to entry: From the perspective of the D-PDU API, it does not make any difference whether an application accesses the software interface directly or through an MVCI D-Server. Consequently, in this document, the term "application" represents both ways of accessing the D-PDU API.

3.1.2

ComLogicalLink

logical communication channel towards a single electronic control unit (ECU) or towards multiple electronic control units

3.1.3

COMPARAM-SPEC

protocol-specific set of predefined communication parameters (ComParams), the value of which can be changed in the context of a layer or specific diagnostic service

Note 1 to entry: This part of the model can also contain OEM-specific ComParams.

3.1.4

ComPrimitive

smallest aggregation of a communication service or function

EXAMPLE A request message to be sent to an ECU.

3.1.5

Ethernet

physical network media type

[ISO 22900-2:2022](#)

<https://standards.iteh.ai/catalog/standards/sist/5e549e42-182c-407f-a0cc-504a0f96866b/iso-22900-2-2022>

3.1.6

local network

part of the network connected directly to the tester

Note 1 to entry: Also called "primary network" in some subclauses.

3.1.7

remote network

part of the network located behind a gateway connected to the tester

Note 1 to entry: Also called "secondary network" in some subclauses.

3.1.8

DoIP MVCI module

diagnostic communication over Internet Protocol modular vehicle communication interface module

MVCI module able to handle connections to one or multiple *DoIP entities* (3.1.11)

3.1.9

DoIP gateway

diagnostic communication over Internet Protocol gateway

gateway connected to the tester via DoIP protocol

3.1.10**DoIP node****diagnostic communication over Internet Protocol node**

ECU connected to the tester directly via DoIP protocol

Note 1 to entry: ECU has no gateway capabilities.

3.1.11**DoIP entity****diagnostic communication over Internet Protocol entity**

general term for either a *DoIP gateway* (3.1.9) or a *DoIP node* (3.1.10)

3.1.12**DoIP entity certificate**

certificate issued by an *intermediate CA* (3.1.13) to *DoIP entity* (3.1.11)

Note 1 to entry: It is presented during the TLS handshake to the external test equipment to verify the authenticity of this DoIP entity.

3.1.13**intermediate CA**

certificate authority which issues subordinal certificates

Note 1 to entry: A reason to issue subordinal certificates can be, for example to further to another intermediate CA or *DoIP entities* (3.1.11).

3.1.14**intermediate certificate**

certificate used by an *intermediate CA* (3.1.13)
900-2:2022

Note 1 to entry: The intermediate certificate(s) is/are either stored in the external test equipment or are presented during authentication together with the end node certificate to complete the chain of trust.

3.1.15**root CA**

certificate authority (CA) acting as the root of trust

Note 1 to entry: Root CA typically issues *intermediate certificates* (3.1.14) to allow an *intermediate CA* (3.1.13) to further submit certificates.

3.1.16**root certificate**

certificate of the *root CA* (3.1.15) used as the trust anchor

Note 1 to entry: It is securely stored and used by all entities that wants to validate end node certificates [e.g. from the *DoIP entity* (3.1.11)] together with all necessary *intermediate certificates* (3.1.14) in the chain of trust.

3.2 Abbreviated terms

API application programming interface

ASCII American Standard for Character Information Interchange

CA certificate authority

CAN	controller area network
CAN IDs	controller area network identifiers
CDF	cable description file
CLL	ComLogicalLink
ComLogicalLinks	communication logical links
ComParam	communication parameter
ComParamSpec	communication parameter specification
ComPrimitive	communication primitive
CRC	cyclic redundancy check
DLC	data link connector
DLL	dynamic link library
DoIP	diagnostic communication over Internet Protocol
D-PDU	diagnostic protocol data unit
D-Server	diagnostic server
ECU	electronic control unit
HDD	hard disk drive
HI	differential line — high
HW	Hardware
IEEE 1394	firewire serial bus
IFR	in-frame response
IGN	Ignition
IOCTL	input/output control
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
K	UART K-Line
KWP	keyword protocol
L	UART L-Line
LL	Logical Link

LOW	differential line — low
MDF	module description file
MOST	media oriented systems transport
MVCI	modular vehicle communication interface
ODX	open diagnostic data exchange
OEM	original equipment manufacturer
OSI	open systems interconnection
PC	personal computer
PCI	protocol control information
PGN	parameter group number
PROGV	programmable voltage
PWM	pulse width modulation
RDF	root description file
RX	UART uni-directional receive
SCI	serial communications interface
SCP	standard corporate protocol https://standards.iteh.ai/catalog/standards/sis/Se549e42-182c-407f-a0cc-504a0f96866b/iso-22900-2-2022
TCP	transmission control protocol
TLS	transport layer security
TX	UART uni-directional transmit
UDS	unified diagnostic services
UDSonCAN	unified diagnostic services on CAN
UDSonIP	unified diagnostic services on IP
USB	universal serial bus
USDT	unacknowledged segmented data transfer
UUDT	unacknowledged un-segmented data transfer
VCI	vehicle communication interface
VPW	variable pulse width
XML	extensible markup language

ITEH STANDARD PREVIEW

(standards.iteh.ai)

4 Specification release version information

4.1 Specification release version location

Specification release version information is contained in each modular VCI release document specification under the same clause title “Specification release version information”. It is important to check for feature support between modular VCI release specifications if the most recent API features shall be implemented. The D-PDU API supports the reading of version information by the API function call PDUGetVersion.

Release version information is also contained in the following files:

- root description file (RDF);
- module description file (MDF);
- cable description file (CDF);
- D-PDU API library file.

4.2 Specification release version

The specification release version of this document is 2.2.2.

PREVIEW
(standards.iteh.ai)

5 Modular VCI use cases

5.1 OEM merger

[ISO 22900-2:2022](https://standards.iteh.ai/catalog/standards/sist/5e549e42-182c-407f-a0cc-504a0f96866b/iso-22900-2:2022)

In the past, several OEMs in the automotive industry have merged into one company.

All companies try to leverage existing (legacy) components and jointly develop new products, which are common across different vehicle types and badges.

If OEMs already had modular VCI compliant test equipment, it would be simple to connect MVCI protocol modules from merged OEMs into one chassis or device. All protocols would be supported by a single MVCI protocol module configuration without any replacement of MVCI protocol module hardware at the dealer site. The same applies for engineering and some of this concept might also work for production plants (end of line).

5.2 OEM cross vehicle platform ECU(s)

OEMs specify requirements and design electronic systems to be implemented in multiple vehicle platforms in order to avoid re-inventing a system for different vehicles. The majority of design, normal operation and diagnostic data of an electronic system are re-used if installed in various vehicles. The engineering development centres are located worldwide. A great amount of re-authoring of diagnostic data is performed to support different engineering test tools.

Providing diagnostic data in an industry standard format like ODX and XML will avoid re-authoring into various test tool specific formats at different system engineering locations. The D-PDU API supports this re-use concept by fully abstracting vehicle protocols into the industry supported ComParam descriptions.

5.3 Central source diagnostic data and exchange during ECU development

Single source origin of diagnostic data (as depicted in Figure 1), combined with a verification and feedback mechanism and distribution to the end users, is highly desirable in order to lower engineering expenses. Engineering, manufacturing and service specify which protocol and data shall be implemented in the ECU. This information will be documented in a structured format like XML. Furthermore, the same structured data files can be used to setup the diagnostic engineering tools to verify proper communication with the ECU and to perform functional verification and compliance testing of the ECU. Once all quality goals are met, these structured data files shall be released to the OEM database. An Open Diagnostic data eXchange (ODX) schema has been developed for the purpose of supporting these structured formatted files used for ECU diagnosis and validation.

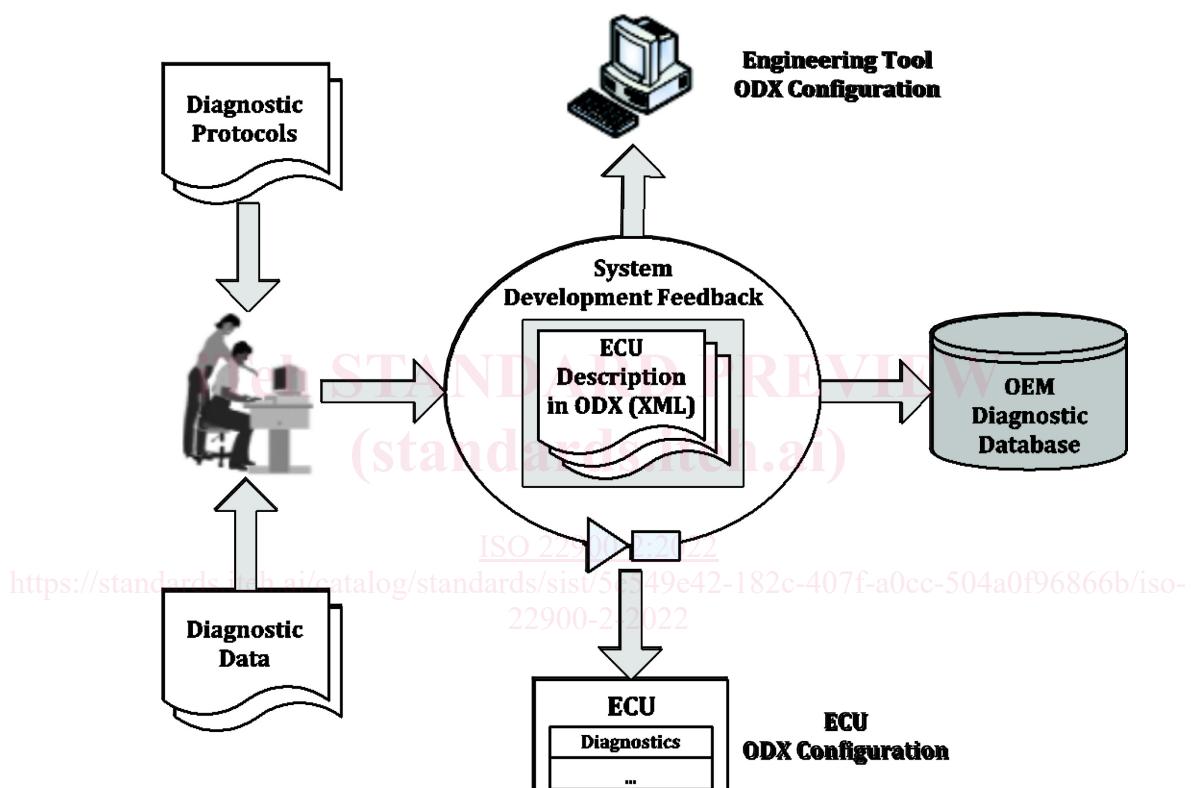


Figure 1 — Example of central source engineering diagnostic data process

5.4 OEM franchised dealer and aftermarket service outlet diagnostic tool support

The service shop uses the modular VCI hardware and software for vehicle diagnosis and enhanced procedural testing. By using the same engineering, manufacturing and service functions as those used for individual ECU testing, the reliability of the data is maintained. A modular VCI protocol module can be used with any PC (handheld or stationary) and can be utilized as an embedded device.

6 Modular VCI software architecture

6.1 Overview

The modular VCI concept is mainly based on three software components (see Figure 2):

- MVCI D-Server software;