

---

---

**Road vehicles — Modular vehicle  
communication interface (MVCI) —**

**Part 2:**

**Diagnostic protocol data unit application  
programming interface (D-PDU API)**

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

ISO 22900-2:2009

<https://standards.iteh.ai/catalog/standards/sist/58e37f3a-a36b-41da-a46f-e7c7aeb93f67/iso-22900-2-2009>



Reference number  
ISO 22900-2:2009(E)

**PDF disclaimer**

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

**iTeh STANDARD PREVIEW**  
**(standards.iteh.ai)**

ISO 22900-2:2009

<https://standards.iteh.ai/catalog/standards/sist/58e37f3a-a36b-41da-a46f-e7c7aeb93f67/iso-22900-2-2009>



**COPYRIGHT PROTECTED DOCUMENT**

© ISO 2009

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office  
Case postale 56 • CH-1211 Geneva 20  
Tel. + 41 22 749 01 11  
Fax + 41 22 749 09 47  
E-mail [copyright@iso.org](mailto:copyright@iso.org)  
Web [www.iso.org](http://www.iso.org)

Published in Switzerland

# Contents

Page

Foreword .....	vi
Introduction.....	vii
1 Scope .....	1
2 Normative references .....	1
3 Terms and definitions .....	2
4 Symbols and abbreviated terms .....	2
5 Specification release version information .....	4
5.1 Specification release version location .....	4
5.2 Specification release version .....	4
6 Modular VCI use cases .....	4
6.1 OEM merger .....	4
6.2 OEM cross vehicle platform ECU(s) .....	4
6.3 Central source diagnostic data and exchange during ECU development .....	5
6.4 OEM franchised dealer and aftermarket service outlet diagnostic tool support.....	5
7 Modular VCI software architecture .....	5
7.1 Overview.....	5
7.2 Modular VCI D-Server software.....	6
7.3 Runtime format based on ODX .....	7
7.4 MVCI protocol module software .....	7
7.5 MVCI protocol module configurations .....	7
8 D-PDU API use cases .....	8
8.1 Overview.....	8
8.2 Use case 1: Single MVCI protocol module.....	8
8.3 Use case 2: Multiple MVCI protocol modules supported by same D-PDU API implementation .....	9
8.4 Use case 3: Multiple MVCI protocol modules supported by different D-PDU API implementations .....	10
9 Diagnostic protocol data unit (D-PDU) API.....	11
9.1 Software requirements.....	11
9.1.1 General requirements .....	11
9.1.2 Vehicle protocol requirements.....	12
9.1.3 Timing requirements for protocol handler messages .....	12
9.1.4 Serialization requirements for protocol handler messages.....	14
9.1.5 Compatibility requirements .....	15
9.1.6 Timestamp requirements.....	16
9.2 API function overview and communication principles.....	17
9.2.1 Terms used within the D-PDU API .....	17
9.2.2 Function overview .....	17
9.2.3 General usage.....	19
9.2.4 Asynchronous and synchronous communication .....	21
9.2.5 Usage of resource locking and resource unlocking.....	22
9.2.6 Usage of ComPrimitives .....	22
9.3 Tool integration .....	38
9.3.1 Requirement for generic configuration.....	38
9.3.2 Tool integrator – use case .....	38
9.4 API functions – interface description .....	40
9.4.1 Overview.....	40

9.4.2	PDUConstruct .....	40
9.4.3	PDUDeconstruct .....	41
9.4.4	PDUIoCtl .....	42
9.4.5	PDUGetVersion .....	43
9.4.6	PDUGetStatus .....	44
9.4.7	PDUGetLastError .....	45
9.4.8	PDUGetResourceStatus .....	47
9.4.9	PDUCreateComLogicalLink .....	48
9.4.10	PUDestroyComLogicalLink .....	50
9.4.11	PDUConnect .....	51
9.4.12	PDUDisconnect .....	53
9.4.13	PDULockResource .....	54
9.4.14	PDUUnlockResource .....	55
9.4.15	PDUGetComParam .....	56
9.4.16	PDUSetComParam .....	63
9.4.17	PDUStartComPrimitive .....	65
9.4.18	PDUCancelComPrimitive .....	69
9.4.19	PDUGetEventItem .....	70
9.4.20	PUDestroyItem .....	71
9.4.21	PDURegisterEventCallback .....	72
9.4.22	EventCallback prototype .....	74
9.4.23	PDUGetObjectid .....	75
9.4.24	PDUGetModuleids .....	76
9.4.25	PDUGetResourceids .....	78
9.4.26	PDUGetConflictingResources .....	79
9.4.27	PDUGetUniqueRespldTable .....	80
9.4.28	PDUSetUniqueRespldTable .....	82
9.4.29	PDUModuleConnect .....	87
9.4.30	PDUModuleDisconnect .....	88
9.4.31	PDUGetTimestamp .....	89
9.5	I/O control section .....	90
9.5.1	IOCTL API command overview .....	90
9.5.2	PDU_IOCTL_RESET .....	92
9.5.3	PDU_IOCTL_CLEAR_TX_QUEUE .....	93
9.5.4	PDU_IOCTL_SUSPEND_TX_QUEUE .....	93
9.5.5	PDU_IOCTL_RESUME_TX_QUEUE .....	94
9.5.6	PDU_IOCTL_CLEAR_RX_QUEUE .....	94
9.5.7	PDU_IOCTL_READ_VBATT .....	95
9.5.8	PDU_IOCTL_SET_PROG_VOLTAGE .....	95
9.5.9	PDU_IOCTL_READ_PROG_VOLTAGE .....	96
9.5.10	PDU_IOCTL_GENERIC .....	97
9.5.11	PDU_IOCTL_SET_BUFFER_SIZE .....	97
9.5.12	PDU_IOCTL_GET_CABLE_ID .....	98
9.5.13	PDU_IOCTL_START_MSG_FILTER .....	98
9.5.14	PDU_IOCTL_STOP_MSG_FILTER .....	100
9.5.15	PDU_IOCTL_CLEAR_MSG_FILTER .....	101
9.5.16	PDU_IOCTL_SET_EVENT_QUEUE_PROPERTIES .....	101
9.5.17	PDU_IOCTL_SEND_BREAK .....	102
9.5.18	PDU_IOCTL_READ_IGNITION_SENSE_STATE .....	103
9.6	API functions — error handling .....	104
9.6.1	Synchronous error handling .....	104
9.6.2	Asynchronous error handling .....	104
9.7	Installation .....	104
9.7.1	Generic description .....	104
9.7.2	Windows installation process .....	105
9.7.3	Linux installation process .....	106
9.7.4	Selecting MSCI protocol modules .....	106
9.8	Application notes .....	106
9.8.1	Interaction with the MDF .....	106
9.8.2	Accessing additional hardware features for MSCI protocol modules .....	106

STANDARD PREVIEW  
(standards.itech.ai)

ISO 22900-2:2009

<https://standards.itech.ai/catalog/standards/sist/58e37f3a-a36b-41da-a46f-e7c7aeb93f67/iso-22900-2-2009>

9.8.3	Documentation and information provided by MVCI protocol module vendors .....	107
9.8.4	Performance Testing .....	107
10	Using the D-PDU API with existing applications.....	108
10.1	SAE J2534-1 and RP1210a existing standards .....	108
11	Data structures .....	108
11.1	API functions — data structure definitions .....	108
11.1.1	Abstract basic data types .....	108
11.1.2	Definitions .....	109
11.1.3	Bit encoding for UNUM32 .....	109
11.1.4	API data structures.....	110
Annex A	(normative) D-PDU API compatibility mappings.....	123
Annex B	(normative) D-PDU API standard ComParams and protocols.....	141
Annex C	(informative) D-PDU API manufacturer specific ComParams and protocols.....	209
Annex D	(normative) D-PDU API constants .....	211
Annex E	(normative) Application defined tags .....	225
Annex F	(normative) Description files .....	226
Annex G	(informative) Resource handling scenarios .....	269
Annex H	(informative) D-PDU API partitioning.....	274
Annex I	(informative) Use case scenarios .....	278
Annex J	(normative) OBD protocol initialization .....	310
Bibliography	.....	325

<https://standards.iteh.ai/catalog/standards/sist/58e37f3a-a36b-41da-a46f-e7c7aeb93f67/iso-22900-2-2009>  
 ISO 22900-2:2009

## 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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

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.

ISO 22900-2 was prepared by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 3, *Electrical and electronic equipment*.

ISO 22900 consists of the following parts, under the general title *Road vehicles — Modular vehicle communication interface (MVCI)*:

- *Part 1: Hardware design requirements*
- *Part 2: Diagnostic protocol data unit application programming interface (D-PDU API)*
- *Part 3: Diagnostic server application programming interface (D-Server API)*

## Introduction

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

This part of ISO 22900 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 after market 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 part of ISO 22900 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 after market. 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 support a protocol-independent D-PDU API as specified in this part of ISO 22900.

<https://standards.iteh.ai/catalog/standards/sist/58e37f3a-a36b-41da-a46f-e7c7aeb93f67/iso-22900-2-2009>

## **iTeh STANDARD PREVIEW** **(standards.iteh.ai)**

ISO 22900-2:2009

<https://standards.iteh.ai/catalog/standards/sist/58e37f3a-a36b-41da-a46f-e7c7aeb93f67/iso-22900-2-2009>



# Road vehicles — Modular vehicle communication interface (MVCI) —

## Part 2: Diagnostic protocol data unit application programming interface (D-PDU API)

### 1 Scope

This part of ISO 22900 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 part of ISO 22900 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 part of ISO 22900.

Migration from and to the existing standards SAE J2534-1 and RP1210a is addressed. This part of ISO 22900 contains a description of how to convert between the APIs. Corresponding wrapper APIs accompany this part of ISO 22900.

<https://standards.iteh.ai/catalog/standards/sist/58e37f3a-a36b-41da-a46f-iso-22900-2-2009>

The purpose of this part of ISO 22900 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 referenced documents are indispensable for the application 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.

ISO 9141-2, *Road vehicles — Diagnostic systems — Part 2: CARB requirements for interchange of digital information*

ISO 14229-1, *Road vehicles — Unified diagnostic services (UDS) — Part 1: Specification and requirements*

ISO 14230 (all parts), *Road vehicles — Diagnostic systems — Keyword Protocol 2000*

ISO 15031-5, *Road vehicles — Communication between vehicle and external equipment for emissions-related diagnostics — Part 5: Emissions-related diagnostic services*

ISO 15765 (all parts), *Road vehicles — Diagnostics on Controller Area Networks (CAN)*

ISO 22901-1, *Road vehicles — Open diagnostic data exchange (ODX) — Part 1: Data model specification*

ISO/IEC 8859-1, *Information technology — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1*

### 3 Terms and definitions

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

#### 3.1 application

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

NOTE 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 part of ISO 22900, the term “application” represents both ways of accessing the D-PDU API.

#### 3.2 ComLogicalLink

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

#### 3.3 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 This part of the model can also contain OEM-specific ComParams.

#### 3.4 ComPrimitive

smallest aggregation of a communication service or function

EXAMPLE A request message to be sent to an ECU.

#### 3.5 Ethernet

physical network media type

ISO 22900-2:2009  
<https://standards.iteh.ai/catalog/standards/sist/58e37f3a-a36b-41da-a46f-e7c7aeb93f67/iso-22900-2-2009>

### 4 Symbols and abbreviated terms

API	Application Programming Interface
ASCII	American Standard for Character Information Interchange
CAN	Controller Area Network
CDF	Cable Description File
CLL	ComLogicalLink
ComParam	Communication Parameter
COP	Communication Primitive
CRC	Cyclic Redundancy Check
DLC	Data Link Connector
DLL	Dynamic Link Library
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
K	UART K-Line
KWP	Keyword Protocol
L	UART L-Line
LOW	Differential Line — Low
MDF	Module Description File
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
TX	UART uni-directional transmit
USB	Universal Serial Bus
USDT	Unacknowledged segmented data transfer <sup>1)</sup>
UUDT	Unacknowledged un-segmented data transfer <sup>2)</sup>

1) ISO 15765-2 network layer includes protocol control information for segmented data transmission, which results in a maximum of 7 data bytes for normal addressing and 6 data bytes for extended addressing.

2) Single CAN frames do not include protocol control information, which results in a maximum of 8 data bytes for normal addressing and 7 data bytes for extended addressing.

VCI	Vehicle Communication Interface
VPW	Variable Pulse Width
XML	Extensible Markup Language

## 5 Specification release version information

### 5.1 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.

**iTeh STANDARD PREVIEW**

### 5.2 Specification release version (standards.iteh.ai)

The specification release version of this part of ISO 22900 is: 2.2.0.

<https://standards.iteh.ai/catalog/standards/sist/58e37f3a-a36b-41da-a46f-e7c7aeb93f67/iso-22900-2-2009>

## 6 Modular VCI use cases

### 6.1 OEM merger

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).

### 6.2 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.

### 6.3 Central source diagnostic data and exchange during ECU development

Single source origin of diagnostic data (as depicted in Figure 1 — Example of central source engineering diagnostic data process), 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.

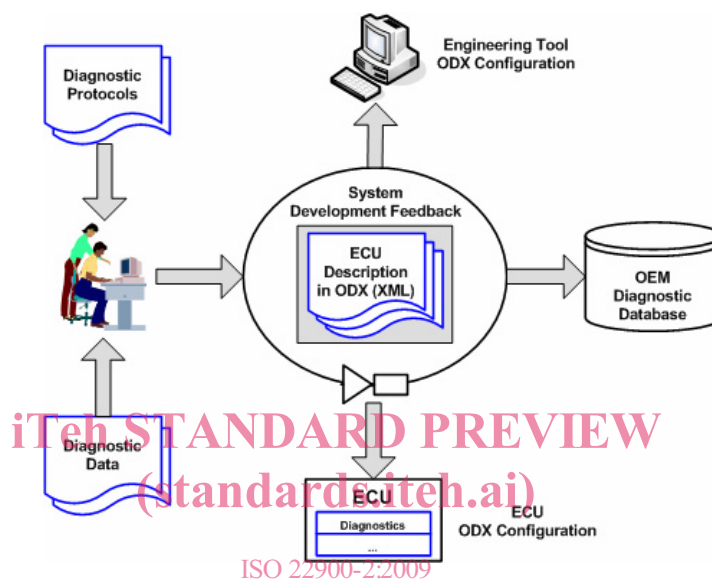


Figure 1 — Example of central source engineering diagnostic data process

### 6.4 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 utilised as an embedded device.

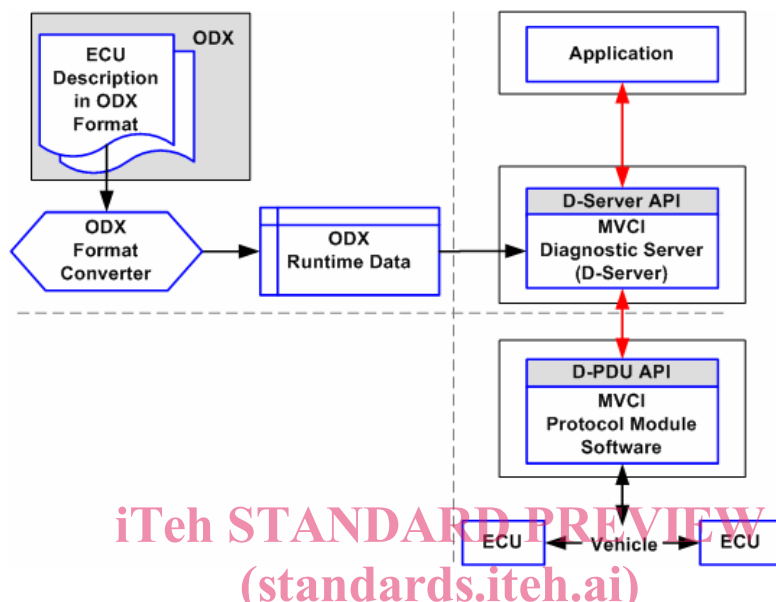
## 7 Modular VCI software architecture

### 7.1 Overview

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

- MVCI D-Server software;
- runtime data based on ODX;
- MVCI protocol module software.

The application accesses the MVCI D-Server through the MVCI D-Server API. The D-Server obtains all required information about the ECU(s) out of the ODX runtime data. Using the ODX runtime data information, the D-Server converts the application's request into a byte stream, which is called a diagnostic protocol data unit (D-PDU). The D-PDU is handed over to the MVCI protocol module through the D-PDU API. The MVCI protocol module transmits the D-PDU to the vehicle's ECU(s). The other way around, the MVCI protocol module receives the vehicle's response(s) and reports the response data to the D-Server. Again using the ODX runtime data, the D-Server interprets the D-PDU and provides the interpreted symbolic information to the application.



NOTE The grey shading of symbols indicates reference to the following International Standards:

- ODX: ISO 22901-1; <https://standards.iteh.ai/catalog/standards/sist/58e37f3a-a36b-41da-a46f-e7c7aeb93f67/iso-22900-2-2009>
- D-Server API: ISO 22900-3;
- D-PDU API: ISO 22900-2;
- MVCI protocol module: ISO 22900-1.

Figure 2 — MVCI software architecture

## 7.2 Modular VCI D-Server software

The MVCI D-Server is accessible through the MVCI D-Server API. By accessing this API, the application may browse the available features for each ECU and initiate a request towards an ECU using simple symbolic expressions. If the request requires input parameters, they can be specified using symbolic expressions as well. The MVCI D-server takes the symbolic request, including input parameters, and converts them into a diagnostic request message as defined at the protocol level. The diagnostic request message represents the diagnostic protocol data unit (D-PDU) as passed to the MVCI protocol module through the D-PDU API. Vice versa, the D-Server converts diagnostic response messages as retrieved from the MVCI protocol module back to symbolic information and provides it to the application.

For a detailed description and the complete MVCI D-Server API definition, see ISO 22900-3.

### 7.3 Runtime format based on ODX

For every conversion from symbolic requests to diagnostic request messages, and vice versa for responses, the MVCI D-Server obtains the required information from the runtime database. The database defines the structure of every diagnostic request and response as supported by an ECU. The database defines byte and bit positions, width, and type of every input and output parameter.

Even though the MVCI D-Server obtains its information from a runtime database, the runtime database and format are not specified by the MVCI standard. Instead, the MVCI standard defines an exchange format to import and export the ECU description across OEMs and tool suppliers. The runtime format is left up to the system designers.

The exchange format is called open diagnostic data exchange format (ODX format). For a detailed description, see ISO 22901-1.

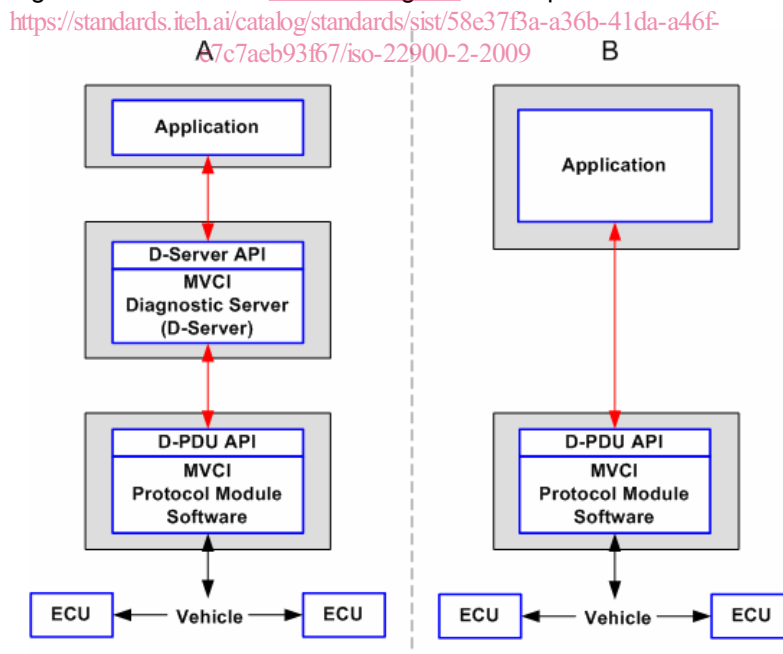
### 7.4 MVCI protocol module software

The MVCI protocol module is accessible through the D-PDU API. The application issues diagnostic requests through the D-PDU API. The MVCI protocol module takes the request D-PDU and transmits it to the vehicle's ECU(s) according to the vehicle communication protocol. Header type, checksum information, and D-PDU segmentation depend on the vehicle communication protocol, and shall be handled transparently by the MVCI protocol module. Also, the MVCI protocol module observes the timing between message frames and requests and responses on the physical interfaces. After completion, the MVCI protocol module simply has to deliver the response back to the application or report an error condition.

### 7.5 MVCI protocol module configurations

The D-PDU API and MVCI protocol modules work in many configurations. A MVCI D-Server is not required as the application interface to the D-PDU API.

Figure 3 — MVCI configurations shows two such configurations to point out the differences.



#### Key

- A application with MVCI D-Server
- B application without MVCI D-Server

NOTE From the perspective of the D-PDU API, it does not make a difference whether an application accesses the software interface directly, or through an MVCI D-Server. Consequently, in this part of ISO 22900, the term “application” represents both ways of accessing the D-PDU API.

Figure 3 — MVCI configurations