

---

---

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

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

*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)*



This document is a preview generated by EBS



**COPYRIGHT PROTECTED DOCUMENT**

© ISO 2017, Published in Switzerland

All rights reserved. Unless otherwise specified, 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  
Ch. de Blandonnet 8 • CP 401  
CH-1214 Vernier, Geneva, Switzerland  
Tel. +41 22 749 01 11  
Fax +41 22 749 09 47  
[copyright@iso.org](mailto:copyright@iso.org)  
[www.iso.org](http://www.iso.org)



## Contents

<b>1</b>	<b>Scope.....</b>	<b>1</b>
<b>2</b>	<b>Normative references.....</b>	<b>1</b>
<b>3</b>	<b>Terms, definitions and abbreviated terms.....</b>	<b>1</b>
<b>3.1</b>	<b>Terms and definitions.....</b>	<b>1</b>
<b>3.2</b>	<b>Abbreviated terms.....</b>	<b>2</b>
<b>4</b>	<b>Specification release version information .....</b>	<b>4</b>
<b>4.1</b>	<b>Specification release version location .....</b>	<b>4</b>
<b>4.2</b>	<b>Specification release version .....</b>	<b>5</b>
<b>5</b>	<b>Modular VCI use cases.....</b>	<b>5</b>
<b>5.1</b>	<b>OEM merger.....</b>	<b>5</b>
<b>5.2</b>	<b>OEM cross vehicle platform ECU(s).....</b>	<b>5</b>
<b>5.3</b>	<b>Central source diagnostic data and exchange during ECU development.....</b>	<b>5</b>
<b>5.4</b>	<b>OEM franchised dealer and aftermarket service outlet diagnostic tool support.....</b>	<b>6</b>
<b>6</b>	<b>Modular VCI software architecture.....</b>	<b>6</b>
<b>6.1</b>	<b>Overview.....</b>	<b>6</b>
<b>6.2</b>	<b>Modular VCI D-Server software .....</b>	<b>7</b>
<b>6.3</b>	<b>Runtime format based on ODX.....</b>	<b>7</b>
<b>6.4</b>	<b>MVCI protocol module software .....</b>	<b>8</b>
<b>6.5</b>	<b>MVCI protocol module configurations.....</b>	<b>8</b>
<b>7</b>	<b>D-PDU API use cases .....</b>	<b>9</b>
<b>7.1</b>	<b>Overview.....</b>	<b>9</b>
<b>7.2</b>	<b>Use case 1: Single MVCI protocol module.....</b>	<b>9</b>
<b>7.3</b>	<b>Use case 2: Multiple MVCI protocol modules supported by same D-PDU API implementation .....</b>	<b>10</b>
<b>7.4</b>	<b>Use case 3: Multiple MVCI protocol modules supported by different D-PDU API implementations.....</b>	<b>11</b>
<b>8</b>	<b>Diagnostic protocol data unit (D-PDU) API.....</b>	<b>12</b>
<b>8.1</b>	<b>Software requirements.....</b>	<b>12</b>
<b>8.1.1</b>	<b>General requirements.....</b>	<b>12</b>
<b>8.1.2</b>	<b>Vehicle protocol requirements .....</b>	<b>13</b>
<b>8.1.3</b>	<b>Timing requirements for protocol handler messages.....</b>	<b>13</b>
<b>8.1.4</b>	<b>Serialization requirements for protocol handler messages.....</b>	<b>14</b>
<b>8.1.5</b>	<b>Compatibility requirements.....</b>	<b>16</b>
<b>8.1.6</b>	<b>Timestamp requirements .....</b>	<b>16</b>
<b>8.2</b>	<b>API function overview and communication principles .....</b>	<b>16</b>
<b>8.2.1</b>	<b>Terms used within the D-PDU API.....</b>	<b>16</b>
<b>8.2.2</b>	<b>Function overview.....</b>	<b>17</b>
<b>8.2.3</b>	<b>General usage.....</b>	<b>18</b>
<b>8.2.4</b>	<b>Asynchronous and synchronous communication .....</b>	<b>21</b>
<b>8.2.5</b>	<b>Usage of resource locking and resource unlocking.....</b>	<b>21</b>
<b>8.2.6</b>	<b>Usage of ComPrimitives.....</b>	<b>21</b>
<b>8.3</b>	<b>Tool integration .....</b>	<b>36</b>
<b>8.3.1</b>	<b>Requirement for generic configuration .....</b>	<b>36</b>
<b>8.3.2</b>	<b>Tool integrator — Use case.....</b>	<b>36</b>
<b>8.4</b>	<b>API functions — Interface description.....</b>	<b>38</b>
<b>8.4.1</b>	<b>Overview.....</b>	<b>38</b>
<b>8.4.2</b>	<b>PDUConstruct.....</b>	<b>38</b>
<b>8.4.3</b>	<b>PDUDeconstruct.....</b>	<b>39</b>
<b>8.4.4</b>	<b>PDUIoCtl.....</b>	<b>40</b>

8.4.5	PDUGetVersion .....	42
8.4.6	PDUGetStatus.....	42
8.4.7	PDUGetLastError.....	43
8.4.8	PDUGetResourceStatus .....	44
8.4.9	PDUCreateComLogicalLink.....	45
8.4.10	PDUDestroyComLogicalLink .....	48
8.4.11	PDUConnect .....	49
8.4.12	PDUDisconnect.....	51
8.4.13	PDUlockResource.....	52
8.4.14	PDUUnlockResource .....	53
8.4.15	PDUGetComParam.....	54
8.4.16	PDUSetComParam.....	61
8.4.17	PDUStartComPrimitive.....	63
8.4.18	PDUCancelComPrimitive .....	67
8.4.19	PDUGetEventItem .....	68
8.4.20	PDUDestroyItem.....	69
8.4.21	PDURegisterEventCallback.....	70
8.4.22	EventCallback prototype .....	72
8.4.23	PDUGetObjectID.....	73
8.4.24	PDUGetModuleIDs .....	74
8.4.25	PDUGetResourceIDs .....	76
8.4.26	PDUGetConflictingResources .....	77
8.4.27	PDUGetUniqueRespIDTable.....	78
8.4.28	PDUSetUniqueRespIDTable .....	79
8.4.29	PDUModuleConnect.....	84
8.4.30	PDUModuleDisconnect.....	86
8.4.31	PDUGetTimestamp .....	87
8.5	I/O control section.....	88
8.5.1	IOCTL API command overview .....	88
8.5.2	PDU_IOCTL_RESET .....	90
8.5.3	PDU_IOCTL_CLEAR_TX_QUEUE .....	91
8.5.4	PDU_IOCTL_SUSPEND_TX_QUEUE.....	91
8.5.5	PDU_IOCTL_RESUME_TX_QUEUE .....	92
8.5.6	PDU_IOCTL_CLEAR_RX_QUEUE.....	92
8.5.7	PDU_IOCTL_CLEAR_TX_QUEUE_PENDING .....	93
8.5.8	PDU_IOCTL_READ_VBATT.....	93
8.5.9	PDU_IOCTL_SET_PROG_VOLTAGE.....	94
8.5.10	PDU_IOCTL_READ_PROG_VOLTAGE.....	95
8.5.11	PDU_IOCTL_GENERIC.....	95
8.5.12	PDU_IOCTL_SET_BUFFER_SIZE .....	96
8.5.13	PDU_IOCTL_GET_CABLE_ID.....	96
8.5.14	PDU_IOCTL_START_MSG_FILTER.....	97
8.5.15	PDU_IOCTL_STOP_MSG_FILTER.....	98
8.5.16	PDU_IOCTL_CLEAR_MSG_FILTER .....	99
8.5.17	PDU_IOCTL_SET_EVENT_QUEUE_PROPERTIES.....	99
8.5.18	PDU_IOCTL_SEND_BREAK.....	100
8.5.19	PDU_IOCTL_READ_IGNITION_SENSE_STATE .....	101
8.5.20	PDU_IOCTL_VEHICLE_ID_REQUEST .....	102
8.5.21	PDU_IOCTL_SET_ETH_SWITCH_STATE.....	103
8.5.22	PDU_IOCTL_GET_ENTITY_STATUS.....	104
8.5.23	PDU_IOCTL_GET_DIAGNOSTIC_POWER_MODE.....	105
8.5.24	PDU_IOCTL_GET_ETH_PIN_OPTION .....	105
8.6	API functions — Error handling .....	106
8.6.1	Synchronous error handling .....	106
8.6.2	Asynchronous error handling.....	106
8.7	Installation .....	106
8.7.1	Generic description .....	106
8.7.2	Windows installation process.....	107

8.7.3	Linux installation process.....	108
8.7.4	Selecting MVCI protocol modules.....	108
8.8	Application notes.....	108
8.8.1	Interaction with the MDF .....	108
8.8.2	Accessing additional hardware features for MVCI protocol modules .....	108
8.8.3	Documentation and information provided by MVCI protocol module vendors.....	109
9	Using the D-PDU API with existing applications.....	109
9.1	SAE J2534-1 and RP1210a existing standards .....	109
10	Data structures.....	110
10.1	API functions — Data structure definitions .....	110
10.1.1	Abstract basic data types .....	110
10.1.2	Definitions.....	111
10.1.3	Bit encoding for UNUM32.....	111
10.1.4	API data structures.....	111
Annex A	(normative) D-PDU API compatibility mappings .....	125
A.1	Mapping of D-PDU API and D-Server API .....	125
A.2	Mapping of D-PDU API and ODX.....	125
Annex B	(normative) D-PDU API standard ComParams and protocols.....	126
B.1	Standardized protocols — Support and naming conventions.....	126
B.1.1	General.....	126
B.1.2	SAE J2534 and RP1210a standard protocol names.....	126
B.1.3	Protocol names — Combination list.....	126
B.1.4	Standard protocol naming guidelines.....	128
B.1.5	Standard protocol short names.....	128
B.1.6	D-PDU API optional OBD protocol short names.....	129
B.2	Standard protocol pin types and short names.....	129
B.3	Standard protocol communication parameters (ComParams) .....	130
B.3.1	Protocol ComParam description method .....	130
B.3.2	ComParam class .....	131
B.3.3	ComParam data type.....	132
B.3.4	ComParam support .....	135
B.3.5	ComParam usage .....	135
B.3.6	ComParam OSI layer reference .....	136
B.4	ComParam summary tables .....	136
B.4.1	Application layer.....	136
B.4.2	Transport layer .....	140
B.4.3	Physical layer .....	148
B.4.4	CAN identifier format for ISO 15765 and ISO 11898 protocols .....	150
B.4.5	Non-CAN ComParam examples.....	159
B.4.6	29-bit CAN identifier data page bits.....	160
B.5	ComParam detailed descriptions .....	160
B.5.1	ComParam definitions for application layer .....	160
B.5.2	ComParam definitions for transport layer.....	176
B.5.3	ComParam definitions for physical layer .....	199
Annex C	(informative) D-PDU API manufacturer-specific ComParams and protocols.....	203
C.1	Manufacturer-specific protocols — Support and naming conventions .....	203
C.1.1	General.....	203
C.1.2	Manufacturer protocol naming guidelines.....	203
C.1.3	Manufacturer protocol communication parameters (ComParams) .....	203
Annex D	(normative) D-PDU API constants.....	205
D.1	Constants.....	205
D.1.1	D-PDU API item type values .....	205
D.1.2	ComPrimitive type values .....	205
D.1.3	Object type values.....	206
D.1.4	Status code values .....	206

D.1.5	Information event values .....	207
D.1.6	Resource status values .....	208
D.1.7	Resource lock values .....	208
D.1.8	Event callback data values .....	208
D.1.9	Reserved ID and handle values .....	209
D.1.10	IOCTL filter types values .....	209
D.1.11	IOCTL event queue mode type values .....	209
D.2	Flag definitions .....	210
D.2.1	TxFlag definition .....	210
D.2.2	RxFlag definition .....	211
D.2.3	ClICreateFlag definition .....	212
D.2.4	TimestampFlag definition .....	213
D.3	Function return values .....	214
D.4	Event error codes .....	216
D.4.1	Error event code returned in PDU_IT_ERROR .....	216
D.4.2	Additional error code returned in PDU_IT_ERROR .....	216
Annex E (normative)	Application defined tags .....	219
Annex F (normative)	RDF and MDF description files .....	220
F.1	D-PDU API root description file (RDF) .....	220
F.1.1	General .....	220
F.1.2	UML diagram of RDF .....	220
F.2	MVCI module description file (MDF) .....	221
F.2.1	General .....	221
F.2.2	ComParam string format .....	221
F.2.3	ComParam resolution tag .....	224
F.2.4	UML diagram of MDF .....	224
F.2.5	UML diagram of MDF elements COMPARAM .....	224
F.2.6	UML diagram of MDF element RESOURCE .....	225
F.2.7	UML diagram of MDF element MODULETYPE .....	226
F.3	Cable description file (CDF) .....	226
F.3.1	General .....	226
F.3.2	UML diagram of CDF .....	226
F.4	XML schema .....	227
F.5	Description file examples .....	232
F.5.1	Example MVCI protocol module .....	232
F.5.2	Example root description file .....	234
F.5.3	Example module description file .....	235
F.5.4	Example cable description file .....	276
Annex G (informative)	Resource handling scenarios .....	278
G.1	Resource handling at the API level .....	278
G.1.1	Obtaining resource and object ids .....	278
G.1.2	Example MVCI protocol module resource selection .....	281
Annex H (informative)	D-PDU API partitioning .....	283
H.1	Functional partitioning of a D-PDU API .....	283
H.1.1	ODX data base .....	283
H.1.2	MVCI D-Server .....	283
H.1.3	VCI protocol module .....	283
H.1.4	Vehicle bus network .....	285
Annex I (informative)	Use case scenarios .....	286
I.1	Negative response handling scenarios .....	286
I.1.1	General .....	286
I.1.2	Physical addressing .....	286
I.1.3	Functional addressing .....	291
I.2	ISO 14229-1 UDS .....	298
I.2.1	Suppress positive response scenarios .....	298

I.2.2	Service 0x2A use case scenario .....	303
I.3	Service shop use case scenario.....	306
I.4	Vehicle bus baud rate changing scenario.....	307
I.4.1	General.....	307
I.4.2	Device use .....	307
I.4.3	Example scenarios.....	308
I.5	SAE J1939 use cases .....	310
I.5.1	SAE J1939 CAN ID Formation .....	310
I.5.2	Setting up ComParams for a SAE J1939 ComLogicalLink.....	311
I.5.3	Case 1: Receiving active DTC from DM1 PGN 65226 (0xFECA) .....	312
I.5.4	Case 2: Receive PGN 65264 (0xfef0) — ECU data .....	312
I.5.5	Case 3: Request previously active DTC PGN 65227 (0xFECA) .....	313
I.5.6	Case 4: Read VIN PGN 65260 (0xFEED).....	314
I.5.7	Case 5: Clear specific DTC (DM22) PGN 49920 (0xC300) .....	314
I.5.8	Case 6: Read VIN in raw mode PGN 65260 (0xFEED).....	315
I.5.9	Case 7: Data security in raw mode (DM18) PGN 54272 (0xD400).....	316
I.6	Multiple clients use cases.....	317
I.6.1	Definition.....	317
I.6.2	Multiple clients configurations .....	317
I.6.3	Example scenarios.....	321
I.7	P3 sequencing.....	322
Annex J	(normative) OBD protocol initialization .....	324
J.1	OBD application .....	324
J.1.1	OBD concept .....	324
J.1.2	Automatic OBD protocol determination .....	324
J.1.3	Simultaneous protocol scan sequence using the D-PDU API .....	325
Annex K	(normative) DoIP implementation .....	339
K.1	DoIP vehicle architecture.....	339
K.2	DoIP use case definition .....	340
K.2.1	Diagram with use cases and actors .....	340
K.2.2	Use cases.....	340
K.2.3	D-PDU API and D-Server .....	342
K.2.4	MVCI and D-PDU API .....	343
K.2.5	ISO 13400-3 Ethernet pin option determination and activation .....	351
K.3	Handling of the DoIP protocol in D-PDU API.....	352
K.3.1	General.....	352
K.3.2	Overview: Usage of D-PDU API for DoIP protocol .....	352
K.3.3	Socket handling.....	357
K.3.4	D-PDU API DoIP NACK behaviour.....	358



## 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](http://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](http://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 on 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 the following URL: [www.iso.org/iso/foreword.html](http://www.iso.org/iso/foreword.html).

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

This second edition cancels and replaces the first edition (ISO 22900-2:2009), which has been technically revised and includes the following changes:

- former corrigendum concerning Kline handling;
- former DoIP amendment;
- how to detect the 2 possible DoIP pin assignment; and
- introduction of CAN-FD.

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

## 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.

# 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.

ISO 15765-2, *Road vehicles — Diagnostic communication over Controller Area Network (DoCAN) — Part 2: Transport protocol and network layer services*

ISO 22900-1, *Road vehicles — Modular vehicle communication interface (MVCI) — Part 1: Hardware design requirements*

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 terminological databases for use in standardization at the following addresses:

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

### 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**