

ISO 22900-2:2017-06 (E)

Road vehicles - Modular vehicle communication interface (MVCI) - Part 2: Diagnostic protocol data unit (D-P DU API)

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

8.4	API functions -- Interface description	38
8.4.1	Overview	38
8.4.2	PDUConstruct	38
8.4.3	PDUDeconstruct	39
8.4.4	PDUIoCtl	40
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	PUDestroyComLogicalLink	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	PUDestroyItem	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	PDUGetUniqueRespldTable	78
8.4.28	PDUSetUniqueRespldTable	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	CllCreateFlag 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 (0xFECB)	313
I.5.6	Case 4: Read VIN PGN 65260 (0xFEEC)	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 (0xFEEC)	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