



Association for Standardisation of  
Automation and Measuring Systems

---

## **ASAM MCD-3 D**

Application Programming Interface for MVCI  
Diagnostic Server

Part 1 of 4

Version 3.0.0

Date: 2011-10-31

**Base Standard**

---

© by ASAM e.V., 2011

## **Disclaimer**

This document is the copyrighted property of ASAM e.V.  
Any use is limited to the scope described in the license terms. The license terms can be viewed at [www.asam.net/license](http://www.asam.net/license)

---

**Table of contents**

<b>1</b>	<b>Introduction</b>	<b>9</b>
<b>2</b>	<b>Scope</b>	<b>10</b>
2.1	Objective of the object model	10
2.2	Typographical conventions and mnemonics	10
2.3	Abbreviations	10
2.4	Legends for used graphics	11
2.4.1	Hierarchical diagrams	11
2.4.2	Sequence diagrams	11
2.5	Stereotypes	12
<b>3</b>	<b>Terms and Definition</b>	<b>13</b>
<b>4</b>	<b>Structure of a MVCI diagnostic server</b>	<b>16</b>
<b>5</b>	<b>Diagnostic Server</b>	<b>20</b>
5.1	MCD system object	20
5.2	Description of terms	21
5.2.1	General	21
5.3	Version information retrieval	25
5.4	States of the MCD system	25
5.5	State changes	28
5.6	Project configuration	29
5.7	Interface structure of server API	31
5.7.1	Hierarchical model overview	31
5.7.2	Separation in database and runtime side	34
5.7.3	Parent functionality	37
5.7.4	Entity Relationship Diagrams	37
5.7.5	ODX Data Type mapping for data base and runtime side	46
5.8	Collections	56
5.8.1	Types and methods	56
5.8.2	RunTime collections	57
5.8.3	Data base collections	57
5.9	EventHandler	59
5.9.1	Registering/deregistering of the EventHandlers	59
5.9.2	Methods of the EventHandlers	60
5.10	MCD value	63
5.11	Use cases	65
5.11.1	View	65
5.11.2	Instantiation of projects	66
5.11.3	Data base access	69

5.11.4	Destruction	71
<b>6</b>	<b>Function block Diagnostic in detail</b>	<b>72</b>
<b>6.1</b>	<b>Constraints</b>	<b>72</b>
<b>6.2</b>	<b>System Properties</b>	<b>80</b>
<b>6.3</b>	<b>Diagnostic DiagComPrimitives and Services</b>	<b>81</b>
6.3.1	Diagnostic DiagComPrimitives	81
6.3.2	Service overview	84
6.3.3	Non cyclic single diag service	91
6.3.4	Cyclic diag service	93
6.3.5	Repeated diag service	94
6.3.6	Repeated send only diag service	95
6.3.7	Repeated receive only diag service	96
6.3.8	Summary	96
6.3.9	Protocol parameters	97
<b>6.4</b>	<b>Suppress Positive Response</b>	<b>108</b>
<b>6.5</b>	<b>eEND_OF_PDU as RequestParameter</b>	<b>111</b>
6.5.1	Data base side	111
6.5.2	Runtime side	112
<b>6.6</b>	<b>Variable length parameters</b>	<b>113</b>
<b>6.7</b>	<b>Variant Identification</b>	<b>115</b>
6.7.1	Interpretation algorithm	115
6.7.2	Identification Algorithm	116
6.7.3	Request and ResponseParameter of VI and VIS	120
6.7.4	Sample for VI and VIS	124
6.7.5	Service handling in case of different locations	137
6.7.6	Variant Patterns and Matching Parameters	138
<b>6.8</b>	<b>Use Cases</b>	<b>140</b>
6.8.1	Create Logical Link and use DiagComPrimitives	140
6.8.2	Remove of communication objects	142
6.8.3	Service Handling	143
6.8.4	Result access	147
6.8.5	Error handling in results	148
<b>6.9</b>	<b>Read DTC</b>	<b>159</b>
6.9.1	ODX Data for Example Read DTC	159
6.9.2	Reading without FaultMemories	162
6.9.3	Reading with FaultMemories	165
6.9.4	DTC Read Service	167
<b>6.10</b>	<b>Logical Link</b>	<b>168</b>
6.10.1	Connection overview	168
6.10.2	State diagram of Logical Link	169
6.10.3	VCI Communication Lost handling	175
6.10.4	Logical Link examples	176
<b>6.11</b>	<b>Functional Addressing</b>	<b>180</b>
<b>6.12</b>	<b>Tables</b>	<b>182</b>
6.12.1	General	182
6.12.2	Usage of tables within DiagComPrimitives	190
<b>6.13</b>	<b>Dynamically Defined Identifiers (DynId)</b>	<b>195</b>

---

6.13.1	General	195
6.13.2	DYNID principle and requirements	196
6.13.3	Lifecycle	197
<b>6.14</b>	<b>Internationalization</b>	<b>205</b>
6.14.1	Multi language support	205
6.14.2	Units	205
<b>6.15</b>	<b>Special Data Groups</b>	<b>205</b>
<b>6.16</b>	<b>ECU (re-) programming</b>	<b>207</b>
6.16.1	Goal	207
6.16.2	Structuring of the function block flash	207
6.16.3	ECU-MEM	212
<b>6.17</b>	<b>Handling binary flash data</b>	<b>213</b>
6.17.1	Late-bound data files	213
6.17.2	Wildcards in data file names	214
6.17.3	Flash Segment Iterator	214
<b>6.18</b>	<b>Library</b>	<b>215</b>
<b>6.19</b>	<b>Jobs</b>	<b>216</b>
6.19.1	General	216
6.19.2	Input and output parameters	218
6.19.3	Job result	219
6.19.4	Single ECU jobs	220
6.19.5	FlashJobs	221
6.19.6	Multiple ECU jobs	221
6.19.7	Job execution	222
6.19.8	Allowed java libraries	230
6.19.9	Naming conventions	232
6.19.10	Job Communication Parameter handling	232
6.19.11	Job Result Generation	232
6.19.12	Job example	235
6.19.13	Job template SingleEcuJob	247
6.19.14	Job template MultipleEcuJob	247
6.19.15	Job template FlashJob	248
<b>6.20</b>	<b>ECU configuration</b>	<b>248</b>
6.20.1	Introduction	248
6.20.2	ECU Configuration Database Part	249
6.20.3	ECU Configuration Runtime Part	253
6.20.4	Error Handling	256
6.20.5	Initialising an MCDConfigurationRecord	256
6.20.6	Offline versus Online Configuration	257
6.20.7	Uploading and Downloading Configuration Strings	258
<b>6.21</b>	<b>Audiences and Additional Audiences</b>	<b>264</b>
6.21.1	General	264
6.21.2	Audiences	265
6.21.3	Additional Audiences	265
<b>6.22</b>	<b>ECU States</b>	<b>266</b>
<b>6.23</b>	<b>Function Dictionary</b>	<b>269</b>
6.23.1	General	269
6.23.2	Functions and funtion groups in ODX	269
6.23.3	Function dictionary data model description	271

---

6.23.4	Uniqueness of D-server FD data resolution	273
6.23.5	Function dictionary usage scenario	275
<b>6.24</b>	<b>Sub-Component data model description</b>	<b>277</b>
6.24.1	Sub-Component usage scenario	278
<b>6.25</b>	<b>Monitoring vehicle bus traffic</b>	<b>279</b>
<b>6.26</b>	<b>Support of VCI module selection and other VCI module features according to D-PDU API Standard</b>	<b>280</b>
6.26.1	Introduction	280
6.26.2	Definitions	281
6.26.3	General behaviour of D-PDU API related D-server methods	282
6.26.4	Overview of VCI module related classes	282
6.26.5	VCI module selection	283
6.26.6	MCDInterface	283
6.26.7	VCI module selection sequence	284
6.26.8	Interface status events	285
6.26.9	MCDInterfaceResource	285
6.26.10	Selection of an interface resource	286
6.26.11	Send Break Signal	287
6.26.12	MCDDbInterfaceCable	287
6.26.13	Accessing VCI module features	288
6.26.14	Behaviour of a MCD-server not using the VCI Module API	289
<b>6.27</b>	<b>Handling DoIP Entities</b>	<b>289</b>
6.27.1	Detection of DoIP Entities	289
6.27.2	Selection of DoIP Entities	291
<b>6.28</b>	<b>Mapping of D-PDU API methods</b>	<b>292</b>
6.28.1	Introduction	292
6.28.2	Initialization and Selection of VCI Modules	292
6.28.3	Communication on a Logical Link	292
6.28.4	Handling of Communication Parameters	295
6.28.5	MCDStartCommunication and MCDStopCommunication	297
6.28.6	D-PDU API IO-Control support	297
<b><u>7</u></b>	<b><u>Error Codes</u></b>	<b><u>298</u></b>
<b>7.1</b>	<b>Principle</b>	<b>298</b>
<b>7.2</b>	<b>Description of the errors</b>	<b>299</b>
7.2.1	Error free behaviour	299
7.2.2	Parameterisation errors	300
7.2.3	RunTime / ProgramViolation errors	300
7.2.4	Data base errors	300
7.2.5	System errors	300
7.2.6	Communication errors	300
7.2.7	Share error	301
<b><u>8</u></b>	<b><u>Appendix</u></b>	<b><u>302</u></b>
<b><u>A</u></b>	<b><u>Code examples</u></b>	<b><u>302</u></b>
<b>A.1</b>	<b>Sample 1: pseudocode of sequences</b>	<b>302</b>
<b>A.2</b>	<b>Example job source code for a single ECU job</b>	<b>324</b>

---

<b><u>B</u></b>	<b><u>Gateway handling</u></b>	<b><u>328</u></b>
<b><u>C</u></b>	<b><u>Flashing for the protocol KWP2000</u></b>	<b><u>329</u></b>
C.1	Content	329
C.2	Flash microsequence inside the MVCI diagnostic server for KWP2000	329
C.2.1	RequestDownload	329
C.2.2	TransferData	330
C.2.3	RequestTransferExit	330
<b><u>D</u></b>	<b><u>Value reading and setting by string</u></b>	<b><u>332</u></b>
D.1	Datatype conversion into Unicode2 string	332
D.2	Representation floating numbers	332
D.3	Normalized floating-point numbers	332
<b><u>E</u></b>	<b><u>BUS TYPES</u></b>	<b><u>334</u></b>
<b><u>F</u></b>	<b><u>system parameter</u></b>	<b><u>336</u></b>
F.1	Overview	336
F.2	Description of the system parameters	336
F.2.1	TIMEZONE	336
F.2.2	YEAR	337
F.2.3	MONTH	337
F.2.4	DAY	337
F.2.5	HOUR	337
F.2.6	MINUTE	337
F.2.7	SECOND	337
F.2.8	TESTERID	337
F.2.9	USERID	337
F.2.10	CENTURY	337
F.2.11	WEEK	338
<b><u>G</u></b>	<b><u>Deprecated Methods and Classes</u></b>	<b><u>339</u></b>
<b><u>H</u></b>	<b><u>Overview optional functionalities</u></b>	<b><u>340</u></b>
<b><u>I</u></b>	<b><u>MONITORING MESSAGE FORMAT</u></b>	<b><u>345</u></b>
I.1	CAN Format	345
I.2	K-Line Format	346
I.3	DoIP Format	346

## 1 Introduction

This International Standard has been established in order to define a universal application programmer interface of a vehicle communication server application. Today's situation in the automotive market requires different vehicle communication interfaces for different vehicle OEMs supporting multiple communication protocols. However, until today, many vehicle communication interfaces are incompatible with regard to interoperability with multiple communication applications and vehicle communication protocols.

Implementation of the MVCI diagnostic server concept supports overall cost reduction to the end user because e.g. a single diagnostic or programming application will support many vehicle communication interfaces supporting different communication protocols and different vehicle communication modules of different vendors at one time.

A vehicle communication application, compliant with this document supports a protocol independent D-PDU API (Protocol Data Unit Application Programming Interface) as specified in [ISO 22900-2]. The server application will need to be configured with vehicle and ECU specific information. This is accomplished by supporting the ODX data format (Open Diagnostic Exchange format) as specified in [ISO 22901-1].