

# Road vehicles — Modular vehicle communication interface (MVCI)

## Part 3: Diagnostic server application programming interface (D-Server API)

ICS 43.040.15

# National foreword

This British Standard is the UK implementation of ISO 22900-3:2009.

The UK participation in its preparation was entrusted to Technical Committee AUE/16, Electrical and electronic equipment.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

**Compliance with a British Standard cannot confer immunity from legal obligations.**

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 July 2009

© BSI 2009

ISBN 978 0 580 55418 6

## Amendments/corrigenda issued since publication

| Date | Comments |
|------|----------|
|      |          |
|      |          |
|      |          |
|      |          |

# INTERNATIONAL STANDARD

**ISO**  
**22900-3**

First edition  
2009-06-15

---

---

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

### Part 3: Diagnostic server application programming interface (D-Server API)

*Véhicules routiers — Interface de communication modulaire du véhicule  
(MVCI) —*

*Partie 3: Interface pour la programmation des applications du serveur  
de diagnostic (D-Server API)*



Reference number  
ISO 22900-3:2009(E)

**AVA** Documents  
Document Sharing Hub  
© ISO 2009

**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.



**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 .....                                                                   | viii |
| Introduction.....                                                                | ix   |
| 1 Scope .....                                                                    | 1    |
| 2 Normative references .....                                                     | 1    |
| 3 Terms, definitions, symbols and abbreviated terms .....                        | 2    |
| 3.1 Terms and definitions .....                                                  | 2    |
| 3.2 Abbreviated terms .....                                                      | 3    |
| 3.3 Typographical conventions and mnemonics used in this part of ISO 22900 ..... | 4    |
| 3.4 Legends for used graphics.....                                               | 5    |
| 3.4.1 Hierarchical diagrams .....                                                | 5    |
| 3.4.2 Sequence diagrams.....                                                     | 5    |
| 3.5 Stereotypes .....                                                            | 6    |
| 4 General considerations.....                                                    | 6    |
| 5 Specification release version information .....                                | 6    |
| 6 Structure of MCD systems.....                                                  | 7    |
| 7 Function block common MCD.....                                                 | 10   |
| 7.1 MCD system object .....                                                      | 10   |
| 7.2 Version information retrieval .....                                          | 11   |
| 7.3 Description of Terms.....                                                    | 11   |
| 7.3.1 General .....                                                              | 11   |
| 7.3.2 Client-controlled object .....                                             | 11   |
| 7.3.3 Location.....                                                              | 11   |
| 7.3.4 Logical Link (LOGICAL-LINK) .....                                          | 12   |
| 7.3.5 Project.....                                                               | 12   |
| 7.3.6 Server-controlled mutable object .....                                     | 13   |
| 7.3.7 Server-controlled object (shared object) .....                             | 13   |
| 7.4 States of the MCD system object.....                                         | 13   |
| 7.5 State changes .....                                                          | 15   |
| 7.6 Project configuration .....                                                  | 16   |
| 7.7 Interface structure of MCD-server API .....                                  | 18   |
| 7.7.1 Separation in database and runtime side .....                              | 18   |
| 7.7.2 Hierarchical model .....                                                   | 19   |
| 7.8 Structure of the database .....                                              | 20   |
| 7.8.1 Overview.....                                                              | 20   |
| 7.8.2 Associations of DbLocation for MCD .....                                   | 20   |
| 7.8.3 Database within the field Measurement and Calibration.....                 | 22   |
| 7.8.4 Database within the field Diagnostics.....                                 | 22   |
| 7.9 Collections .....                                                            | 22   |
| 7.9.1 Types and methods.....                                                     | 22   |
| 7.9.2 RunTime collections .....                                                  | 24   |
| 7.9.3 Database collections.....                                                  | 26   |
| 7.9.4 Handling of collection of ASCIISTRING .....                                | 28   |
| 7.10 EventHandler .....                                                          | 28   |
| 7.10.1 Registering/deregistering of the EventHandlers.....                       | 28   |
| 7.10.2 Methods of the EventHandlers.....                                         | 30   |
| 7.10.3 Eventfilter .....                                                         | 35   |
| 7.11 Multi-Client capability .....                                               | 43   |
| 7.11.1 Requirements.....                                                         | 43   |

|        |                                                                                       |     |
|--------|---------------------------------------------------------------------------------------|-----|
| 7.11.2 | Design .....                                                                          | 46  |
| 7.11.3 | Proxy in Multi Client Architecture .....                                              | 47  |
| 7.11.4 | Cooperation Level.....                                                                | 50  |
| 7.11.5 | Symbolic Names of Clients.....                                                        | 52  |
| 7.11.6 | Selection and de-selection of Project and VehicleInfo in a multi-client setting ..... | 52  |
| 7.11.7 | Notification .....                                                                    | 53  |
| 7.11.8 | Remove shared objects .....                                                           | 54  |
| 7.11.9 | Locking .....                                                                         | 55  |
| 7.12   | Client Controlled Objects.....                                                        | 57  |
| 7.13   | Resource Release .....                                                                | 58  |
| 7.13.1 | Use cases .....                                                                       | 58  |
| 7.13.2 | Requirements .....                                                                    | 59  |
| 7.13.3 | Solution.....                                                                         | 59  |
| 7.14   | Critical Section, Critical Groups of Methods .....                                    | 59  |
| 7.15   | Result access .....                                                                   | 61  |
| 7.16   | MCD value.....                                                                        | 62  |
| 7.16.1 | Value types .....                                                                     | 62  |
| 7.16.2 | Method getValue .....                                                                 | 63  |
| 7.16.3 | Method setValue .....                                                                 | 63  |
| 7.16.4 | Method createValue .....                                                              | 64  |
| 7.17   | Use cases .....                                                                       | 68  |
| 7.17.1 | View .....                                                                            | 68  |
| 7.17.2 | Instantiation of projects .....                                                       | 68  |
| 7.17.3 | Database access .....                                                                 | 71  |
| 7.17.4 | Destruction .....                                                                     | 73  |
| 8      | Function block Common MD .....                                                        | 74  |
| 8.1    | Collector .....                                                                       | 74  |
| 8.1.1  | ERD .....                                                                             | 74  |
| 8.1.2  | Concept.....                                                                          | 75  |
| 8.1.3  | Result access .....                                                                   | 79  |
| 8.1.4  | Collector usage in diagnostics.....                                                   | 84  |
| 8.2    | Use cases .....                                                                       | 85  |
| 8.2.1  | Measurement with a collector - activation .....                                       | 85  |
| 8.2.2  | Measurement with a collector - result access .....                                    | 87  |
| 8.2.3  | Measurement with a collector - polling for results.....                               | 90  |
| 9      | Function block Diagnostics .....                                                      | 91  |
| 9.1    | Description of Terms .....                                                            | 91  |
| 9.1.1  | General.....                                                                          | 91  |
| 9.1.2  | Access Key.....                                                                       | 91  |
| 9.1.3  | Functional Class (FUNCTIONAL-CLASS).....                                              | 91  |
| 9.1.4  | Job (SINGLE-ECU-JOB, MULTIPLE-ECU-JOB).....                                           | 91  |
| 9.1.5  | Physical Interface Link.....                                                          | 91  |
| 9.1.6  | Physical Link .....                                                                   | 92  |
| 9.1.7  | Physical Vehicle Link (PHYSICAL-VEHICLE-LINK) .....                                   | 92  |
| 9.2    | Structuring of the function block Diagnostics .....                                   | 92  |
| 9.2.1  | Separation in database and runtime side .....                                         | 92  |
| 9.2.2  | Relation between Vehicle Connector Information Table and Logical Link Table .....     | 94  |
| 9.2.3  | Hierarchical model.....                                                               | 95  |
| 9.2.4  | Entity Relationship Diagrams.....                                                     | 97  |
| 9.3    | System Properties .....                                                               | 109 |
| 9.4    | Diagnostic DiagComPrimitives and Services .....                                       | 109 |
| 9.4.1  | Diagnostic DiagComPrimitives and States .....                                         | 109 |
| 9.4.2  | Service overview .....                                                                | 113 |
| 9.4.3  | Non cyclic single shot diag service .....                                             | 120 |
| 9.4.4  | Cyclic diag service.....                                                              | 122 |
| 9.4.5  | Repeated diag service .....                                                           | 123 |
| 9.4.6  | Repeated send only diag service .....                                                 | 124 |
| 9.4.7  | Repeated receive only diag service .....                                              | 125 |
| 9.4.8  | Updating repetition parameters .....                                                  | 125 |

|        |                                                                                      |     |
|--------|--------------------------------------------------------------------------------------|-----|
| 9.4.9  | Summary .....                                                                        | 126 |
| 9.4.10 | Protocol parameters.....                                                             | 127 |
| 9.4.11 | Suppress Positive Response .....                                                     | 140 |
| 9.5    | Diagnostic variables.....                                                            | 143 |
| 9.6    | eEND_OF_PDU as RequestParameter .....                                                | 145 |
| 9.6.1  | Database side .....                                                                  | 145 |
| 9.6.2  | Runtime side .....                                                                   | 147 |
| 9.6.3  | COMPUCODE .....                                                                      | 148 |
| 9.7    | Variable length parameters .....                                                     | 149 |
| 9.8    | Layer inheritance of services .....                                                  | 151 |
| 9.8.1  | Goal.....                                                                            | 151 |
| 9.8.2  | Layer inheritance of services .....                                                  | 151 |
| 9.8.3  | Service handling on functional and physical locations .....                          | 156 |
| 9.9    | Variant Identification and Selection (VI / VIS).....                                 | 158 |
| 9.9.1  | Goal.....                                                                            | 158 |
| 9.9.2  | Variant Identification Algorithm.....                                                | 158 |
| 9.9.3  | General VI/VIS handling considerations .....                                         | 166 |
| 9.9.4  | Deselecting of selected variants.....                                                | 168 |
| 9.9.5  | Request and Response parameters of VI and VIS .....                                  | 168 |
| 9.9.6  | Example Scenarios for VI and VIS .....                                               | 173 |
| 9.10   | Base Variant Identification and Selection.....                                       | 183 |
| 9.11   | Use Cases .....                                                                      | 190 |
| 9.11.1 | Creation of LogicalLink and usage of DiagComPrimitives .....                         | 190 |
| 9.11.2 | Removal of communication objects .....                                               | 192 |
| 9.11.3 | Service Handling .....                                                               | 194 |
| 9.11.4 | Result access.....                                                                   | 196 |
| 9.12   | Filtering of results .....                                                           | 225 |
| 9.12.1 | Principle.....                                                                       | 225 |
| 9.12.2 | Handling rules.....                                                                  | 229 |
| 9.13   | Read DTC.....                                                                        | 230 |
| 9.14   | Logical Link.....                                                                    | 234 |
| 9.14.1 | General .....                                                                        | 234 |
| 9.14.2 | Connection overview .....                                                            | 234 |
| 9.14.3 | State diagram of Logical Link .....                                                  | 235 |
| 9.14.4 | Logical Link examples .....                                                          | 243 |
| 9.14.5 | Gateway handling.....                                                                | 248 |
| 9.14.6 | Examples and Relations between Logical Links, Locations and relevant AccessKeys..... | 250 |
| 9.15   | Functional Addressing.....                                                           | 256 |
| 9.16   | Tables .....                                                                         | 258 |
| 9.16.1 | General .....                                                                        | 258 |
| 9.16.2 | Usage of tables within DiagComPrimitives .....                                       | 260 |
| 9.17   | Dynamically Defined Local Id / Table Parameters (DDLID) .....                        | 263 |
| 9.17.1 | General .....                                                                        | 263 |
| 9.17.2 | DDLID principle and requirements .....                                               | 263 |
| 9.17.3 | Lifecycle .....                                                                      | 264 |
| 9.18   | Internationalisation .....                                                           | 272 |
| 9.18.1 | Multi language support.....                                                          | 272 |
| 9.18.2 | Units.....                                                                           | 272 |
| 9.19   | Special Data Groups .....                                                            | 272 |
| 9.20   | ECU Flash programming .....                                                          | 273 |
| 9.20.1 | Goal.....                                                                            | 273 |
| 9.20.2 | Description of Terms for ECU-Reprogramming .....                                     | 274 |
| 9.20.3 | Structure of the function block flash programming .....                              | 275 |
| 9.20.4 | Management of ECU-MEMs .....                                                         | 288 |
| 9.20.5 | Physical Memories .....                                                              | 289 |
| 9.20.6 | Executing flash sessions.....                                                        | 291 |
| 9.21   | Library.....                                                                         | 299 |
| 9.22   | Java Jobs .....                                                                      | 300 |
| 9.22.1 | General .....                                                                        | 300 |
| 9.22.2 | General information Java Jobs.....                                                   | 300 |

|         |                                                                                                           |     |
|---------|-----------------------------------------------------------------------------------------------------------|-----|
| 9.22.3  | Types of Java Jobs .....                                                                                  | 301 |
| 9.22.4  | Handling of Java Jobs .....                                                                               | 304 |
| 9.22.5  | Job execution .....                                                                                       | 314 |
| 9.22.6  | Job example .....                                                                                         | 322 |
| 9.23    | ECU configuration .....                                                                                   | 332 |
| 9.23.1  | General .....                                                                                             | 332 |
| 9.23.2  | ECU Configuration Database Part .....                                                                     | 332 |
| 9.23.3  | ECU Configuration Runtime Part .....                                                                      | 336 |
| 9.23.4  | Error Handling .....                                                                                      | 339 |
| 9.23.5  | Initialising an MCDConfigurationRecord .....                                                              | 339 |
| 9.23.6  | Offline versus Online Configuration .....                                                                 | 340 |
| 9.23.7  | Uploading and Downloading Configuration Strings .....                                                     | 341 |
| 9.24    | Audiences and Additional Audiences .....                                                                  | 346 |
| 9.24.1  | General .....                                                                                             | 346 |
| 9.24.2  | Audiences .....                                                                                           | 348 |
| 9.24.3  | Additional Audiences .....                                                                                | 348 |
| 9.25    | Function Dictionary and Sub-Components .....                                                              | 349 |
| 9.25.1  | Terms and requirements .....                                                                              | 349 |
| 9.25.2  | Functions and function group in ODX .....                                                                 | 349 |
| 9.25.3  | Function dictionary data model description .....                                                          | 351 |
| 9.25.4  | Function dictionary usage scenario .....                                                                  | 353 |
| 9.25.5  | Sub-Component data model description .....                                                                | 355 |
| 9.25.6  | Sub-Component usage scenario .....                                                                        | 356 |
| 9.26    | ECU States .....                                                                                          | 357 |
| 9.27    | Monitoring vehicle bus traffic .....                                                                      | 360 |
| 9.28    | Support of VCI module selection and other VCI module features in accordance with D-PDU API Standard ..... | 362 |
| 9.28.1  | General .....                                                                                             | 362 |
| 9.28.2  | Definitions .....                                                                                         | 362 |
| 9.28.3  | General behaviour of D-PDU API related D-server methods .....                                             | 362 |
| 9.28.4  | Overview of VCI module related classes .....                                                              | 363 |
| 9.28.5  | VCI module selection .....                                                                                | 364 |
| 9.28.6  | MCDInterface .....                                                                                        | 364 |
| 9.28.7  | VCI module selection sequence .....                                                                       | 365 |
| 9.28.8  | Interface status events .....                                                                             | 366 |
| 9.28.9  | MCDInterfaceResource .....                                                                                | 367 |
| 9.28.10 | Selection of an interface resource .....                                                                  | 367 |
| 9.28.11 | Send Break Signal .....                                                                                   | 368 |
| 9.28.12 | MCDDBInterfaceCable .....                                                                                 | 369 |
| 9.28.13 | Accessing VCI module features .....                                                                       | 369 |
| 9.28.14 | Adding Logical Links which are not found in the Vehicle Information .....                                 | 370 |
| 9.28.15 | Behaviour of a MCD-server not supporting VCI Modules in accordance with D-PDU API Standard .....          | 371 |
| 9.29    | Mapping of D-PDU API methods .....                                                                        | 372 |
| 9.29.1  | General .....                                                                                             | 372 |
| 9.29.2  | Initialization and Selection of VCI Modules .....                                                         | 372 |
| 9.29.3  | Communication on a Logical Link .....                                                                     | 372 |
| 9.29.4  | Handling of Communication Parameters .....                                                                | 374 |
| 9.29.5  | MCDStartCommunication and MCDStopCommunication .....                                                      | 376 |
| 10      | Error Codes .....                                                                                         | 377 |
| 10.1    | Principle .....                                                                                           | 377 |
| 10.2    | Description of the errors .....                                                                           | 379 |
| 10.2.1  | Error free behaviour .....                                                                                | 379 |
| 10.2.2  | Parameterisation errors .....                                                                             | 379 |
| 10.2.3  | Runtime / ProgramViolation errors .....                                                                   | 379 |
| 10.2.4  | Database errors .....                                                                                     | 379 |
| 10.2.5  | System errors .....                                                                                       | 379 |
| 10.2.6  | Communication errors .....                                                                                | 379 |
| 10.2.7  | Job error .....                                                                                           | 379 |

|                                                                                                  |            |
|--------------------------------------------------------------------------------------------------|------------|
| <b>Annex A</b> (informative) <b>Code examples</b> .....                                          | <b>380</b> |
| <b>Annex B</b> (normative) <b>COMPUCODE template</b> .....                                       | <b>387</b> |
| <b>Annex C</b> (normative) <b>Job Templates</b> .....                                            | <b>388</b> |
| <b>Annex D</b> (informative) <b>Gateway Handling</b> .....                                       | <b>392</b> |
| <b>Annex E</b> (normative) <b>Value reading and setting by string</b> .....                      | <b>393</b> |
| <b>Annex F</b> (normative) <b>Bus types</b> .....                                                | <b>396</b> |
| <b>Annex G</b> (normative) <b>System parameter</b> .....                                         | <b>398</b> |
| <b>Annex H</b> (normative) <b>Cooperation level</b> .....                                        | <b>401</b> |
| <b>Annex I</b> (normative) <b>System properties</b> .....                                        | <b>428</b> |
| <b>Annex J</b> (informative) <b>VCI Module selection sequence</b> .....                          | <b>431</b> |
| <b>Annex K</b> (informative) <b>Service example illustrations</b> .....                          | <b>432</b> |
| <b>Annex L</b> (informative) <b>Exemplary monitoring message format</b> .....                    | <b>437</b> |
| <b>Annex M</b> (normative) <b>Overview of the methods for isModifiedByOtherClient flag</b> ..... | <b>439</b> |
| <b>Bibliography</b> .....                                                                        | <b>447</b> |

## 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 22900-3 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

The purpose of this part of ISO 22900 is to define a universal application programmer interface of a vehicle communication server application. At present, the automotive market requires different vehicle communication interfaces for different vehicle original equipment manufacturers (OEMs) supporting multiple communication protocols. However, up until now, many vehicle communication interfaces have been incompatible with regard to interoperability with multiple communication applications and vehicle communication protocols.

Implementation of the measurement calibration diagnostic (MCD) server concept supports overall cost reduction to the end user, e.g. because 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 part of ISO 22900 supports a protocol-independent protocol data unit application programming interface (D-PDU API) as specified in ISO 22900-2. The server application needs to be configured with vehicle and electronic control unit (ECU) specific information. This is accomplished by supporting the open diagnostic exchange (ODX) data format, as specified in ISO 22901-1<sup>1)</sup>.

---

1) Equivalent to ASAM MCD 2 D ODX<sup>[11]</sup>.



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

## Part 3: Diagnostic server application programming interface (D-Server API)

### 1 Scope

This part of ISO 22900 defines an object-oriented programming interface, the objective of which is to be able to implement server applications used during the design, production and maintenance phase of a vehicle communication system which are compatible with each other and therefore exchangeable.

A server compliant with this part of ISO 22900 has three function blocks:

- M: measurement,
- C: calibration, and
- D: diagnostics.

A more detailed description of the server API is given in the Programmers Reference Guide<sup>[14]</sup>.

### 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 11898-2, *Road vehicles — Controller area network (CAN) — Part 2: High-speed medium access unit*

ISO 11898-3, *Road vehicles — Controller area network (CAN) — Part 3: Low-speed, fault-tolerant, medium-dependent interface*

ISO 11992-1, *Road vehicles — Interchange of digital information on electrical connections between towing and towed vehicles — Part 1: Physical and data-link layers*

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

ISO 14230-1, *Road vehicles — Diagnostic systems — Keyword Protocol 2000 — Part 1: Physical layer*

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

ISO 22900-2:2009, *Road vehicles — Modular vehicle communication interface (MVCI) — Part 2: Diagnostic protocol data unit application programming interface (D-PDU API)*

### 3 Terms, definitions, symbols and abbreviated terms

#### 3.1 Terms and definitions

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

##### 3.1.1

###### **AccessKey**

path identifier through the inheritance hierarchy as defined in open diagnostic data exchange (ODX) to a diagnostic data element

##### 3.1.2

###### **ancestor object**

###### **parent object**

object located above in the object hierarchy with respect to a given object

##### 3.1.3

###### **descendant object**

###### **child object**

object located below in the object hierarchy with respect to a given object

##### 3.1.4

###### **functional class**

set of diagnostic services

##### 3.1.5

###### **interface connector**

connector at the vehicle end of the interface cable between the vehicle and the communication device

##### 3.1.6

###### **job**

sequence of diagnostic services and other jobs with a control flow

##### 3.1.7

###### **location**

representation of a set of diagnostic data valid on a given hierarchical level of inheritance as defined in open diagnostic data exchange (ODX)

EXAMPLE Multiple ECU Job; Protocol; Functional Group; ECU Base Variant; ECU Variant; Module.

##### 3.1.8

###### **logical link**

set of data, identifying the physical line, the interface and protocol used for an electronic control unit (ECU)

##### 3.1.9

###### **physical interface link**

physical connection between the vehicle communication interface (VCI) connector of a VCI and the interface connector

##### 3.1.10

###### **physical link**

connection from the interface of the D-server to the electronic control unit (ECU) in the vehicle

NOTE A physical link is a **physical vehicle link** (3.1.11) connected to a **physical interface link** (3.1.9).

##### 3.1.11

###### **physical vehicle link**

connection between the vehicle connector and the electronic control unit (ECU), which describes the unique bus system in a vehicle

### 3.1.12 project

pool of diagnostic, measurement and/or calibration data

NOTE References between such data are all resolvable within the same project.

### 3.1.13 vehicle connector

connector on a vehicle providing access to the bus systems in the vehicle

## 3.2 Abbreviated terms

|           |                                                                     |
|-----------|---------------------------------------------------------------------|
| A2L       | ASAM 2 MC language (see ASAM MCD 2 MC <sup>[12]</sup> )             |
| API       | Application Programming Interface                                   |
| ASAM      | Association for Standardisation of Automation and Measuring Systems |
| ASCII     | American Standard for Character Information Interchange             |
| AUSY      | AUTomation SYstem                                                   |
| BVI       | Base Variant Identification                                         |
| BVIS      | Base Variant Identification and Selection                           |
| C         | Calibration                                                         |
| CAN       | Controller Area Network                                             |
| COM/DCOM  | Distributed Component Object Model                                  |
| CORBA     | Common Object Request Broker Architecture                           |
| CRC       | Cyclic Redundancy Check                                             |
| D         | Diagnostics                                                         |
| DDLID     | Dynamically Defined Local ID                                        |
| Diag      | Diagnostic                                                          |
| DLL       | Dynamic Link Library                                                |
| DOP       | diagnostic Data Object Property                                     |
| D-server  | Function block of MCD-server for Diagnostic                         |
| DTC       | Diagnostic Trouble Code                                             |
| DTD       | Document Type Definition                                            |
| DynId     | Dynamic Identifiers                                                 |
| ECU       | Electronic Control Unit                                             |
| ECU MEM   | Electronic Control Unit MEMory                                      |
| ERD       | Entity Relationship Diagram                                         |
| IDL       | Interface Definition Language                                       |
| JAVA RMI  | JAVA Remote Method Invocation                                       |
| KWP       | KeyWord Protocol                                                    |
| M         | Measurement                                                         |
| MCD       | Measurement Calibration Diagnostic                                  |
| MC-server | Function block of MCD-server for Measurement and Calibration        |
| MVCI      | Modular Vehicle Communication Interface                             |

|     |                                                                         |
|-----|-------------------------------------------------------------------------|
| ODX | Open Diagnostic data eXchange                                           |
| OEM | Original Equipment Manufacturer                                         |
| PC  | Personal Computer                                                       |
| PDU | Protocol Data Unit (referred to as D-PDU API in this part of ISO 22900) |
| SDG | Special Data Groups                                                     |
| SI  | Système International d'unités                                          |
| UDS | Unified Diagnostic Services                                             |
| UML | Unified Modeling Language                                               |
| UTC | Coordinated Universal Time                                              |
| VCI | Vehicle Communication Interface                                         |
| VI  | Variant Identification                                                  |
| VIS | Variant Identification and Selection                                    |

### 3.3 Typographical conventions and mnemonics used in this part of ISO 22900

Source code and technical artefacts within the text are presented in `Courier new` type.

Diagrams that denote interaction sequences, relationships or dependencies between interfaces are presented using the UML convention.

The name of each interface and each class defined by this part of ISO 22900 shall use the prefix of the stereotype, e.g. "MCD".

The leading letter of each method and each parameter is small.

The leading word of each method shall be a verb.

The letter "\_" is not permitted in interface names, method names and parameter names, but it is permitted for constants.

The leading letter of each constant is "e" followed by the name in block capitals.

ODX element names are written in upper case, e.g. SHORT-NAME.

Names of classes, objects and interfaces in ISO 22900-3 (MCD-3D) are written in mixed fixed, e.g. MCDDbProject.

### 3.4 Legends for used graphics

#### 3.4.1 Hierarchical diagrams

Figure 1 illustrates the legends for hierarchical models.

|                                         | color print | black/white print |
|-----------------------------------------|-------------|-------------------|
| ● Interface not derived from MLD Object | blue        | black             |
| ○ Interface not directly used           | white       | white             |
| ⊗ Client Interface                      | yellow      | grey              |
| ⊗ Client Interface                      | green       | dark grey         |
| ⊗ Database Interface                    | yellow      | grey              |
| ⊗ Database Interface                    | green       | dark grey         |
| ⊗ Database Interface                    | yellow      | grey              |
| ⊗ Database Interface                    | green       | dark grey         |
| ⊗ Database Interface                    | yellow      | grey              |
| ⊗ Database Interface                    | green       | dark grey         |
| ⊗ Database Interface                    | yellow      | grey              |
| ⊗ Database Interface                    | green       | dark grey         |
| ⊗ Database Interface                    | yellow      | grey              |
| ⊗ Database Interface                    | green       | dark grey         |
| ⊗ Database Interface                    | yellow      | grey              |
| ⊗ Database Interface                    | green       | dark grey         |

Figure 1 — Legend for hierarchical models

#### 3.4.2 Sequence diagrams

With the help of sequence diagrams, the interactive use of the API and the sequences for certain general cases are presented in chronological order.

The sequence diagrams are oriented in accordance with the presentation in UML and are structured as follows. The chronological sequence arises while reading from the top downwards. The commentary column, in which single activities are commented on, is placed in the left margin. Within the sequence diagram, the Client application is shown left; if necessary for the respective case, the EventHandler is shown there as well. Right of the Client (with or without EventHandler) are located the API objects necessary for the respective case. On the far right the D-server is presented if necessary.

Not all API objects possible for the respective instant of time are shown, but only those of relevance for the respective case. The thin line leading down vertically from the objects represents the life line, the wider sections on it represent activities of the object.

The black horizontal arrows between the single objects, Client and D-server, represent the actions necessary for the respective case. The object at which the arrow points will execute the action. The grey horizontal arrows represent the return of objects.

### 3.5 Stereotypes

Stereotypes are abbreviation characters used in this part of ISO 22900 to mark the affiliation of statements, interfaces and methods to one of the possible function blocks. The following stereotypes are used:

| <u>Stereotype</u> | <u>Usage of method and class is permitted in the following Function Blocks</u>                                           |
|-------------------|--------------------------------------------------------------------------------------------------------------------------|
| << M,C,D >>       | Measurement, Calibration and Diagnostic                                                                                  |
| << M,D >>         | Measurement and Diagnostic                                                                                               |
| << M,C >>         | Measurement and Calibration                                                                                              |
| << M >>           | Measurement                                                                                                              |
| << C >>           | Calibration                                                                                                              |
| << D >>           | Diagnostics                                                                                                              |
| << JD >>          | Methods with this stereotype can only be used inside of Diagnostic Jobs. These methods are not available for use at API. |

## 4 General considerations

From the perspective of the user, access and integration of on-board control units is provided by a corresponding application, the communication server and a VCI module for diagnostics, as well as measurement and calibration modules.

The user is granted access to the measurement and calibration objects and to the diagnostic services. The server provides access to the shared use of several function blocks, via identical methods using only one server (the MCD-server) for all of these tasks.

The function blocks are used for the handling of ECUs in vehicles

- for taking measurements (M),
- for making adjustments of calibration parameters (C), and
- for diagnostics (D).

Not all of the three function blocks need to be supported in an implementation of a server. A vendor can choose only to implement the D function block in a D-server; implementation of the M and C function blocks would be termed an MC-server.

A compliant D-server also supports Job-Language (Java) and may support optional features such as ECU (re)programming. The defined object-oriented API provides for a simple, time-saving and efficient interchangeability of different servers.

The client application and the communication server do not need to run on the same computer. Remote use via an interface is also possible, supported by the design of the server API. This interface is provided for COM/DCOM<sup>[17]</sup>, for Java<sup>[19]</sup>, and for C++<sup>[18]</sup>.

## 5 Specification release version information

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

## 6 Structure of MCD systems

Each MCD-server is divided into the functional blocks of M, C, D and the database.

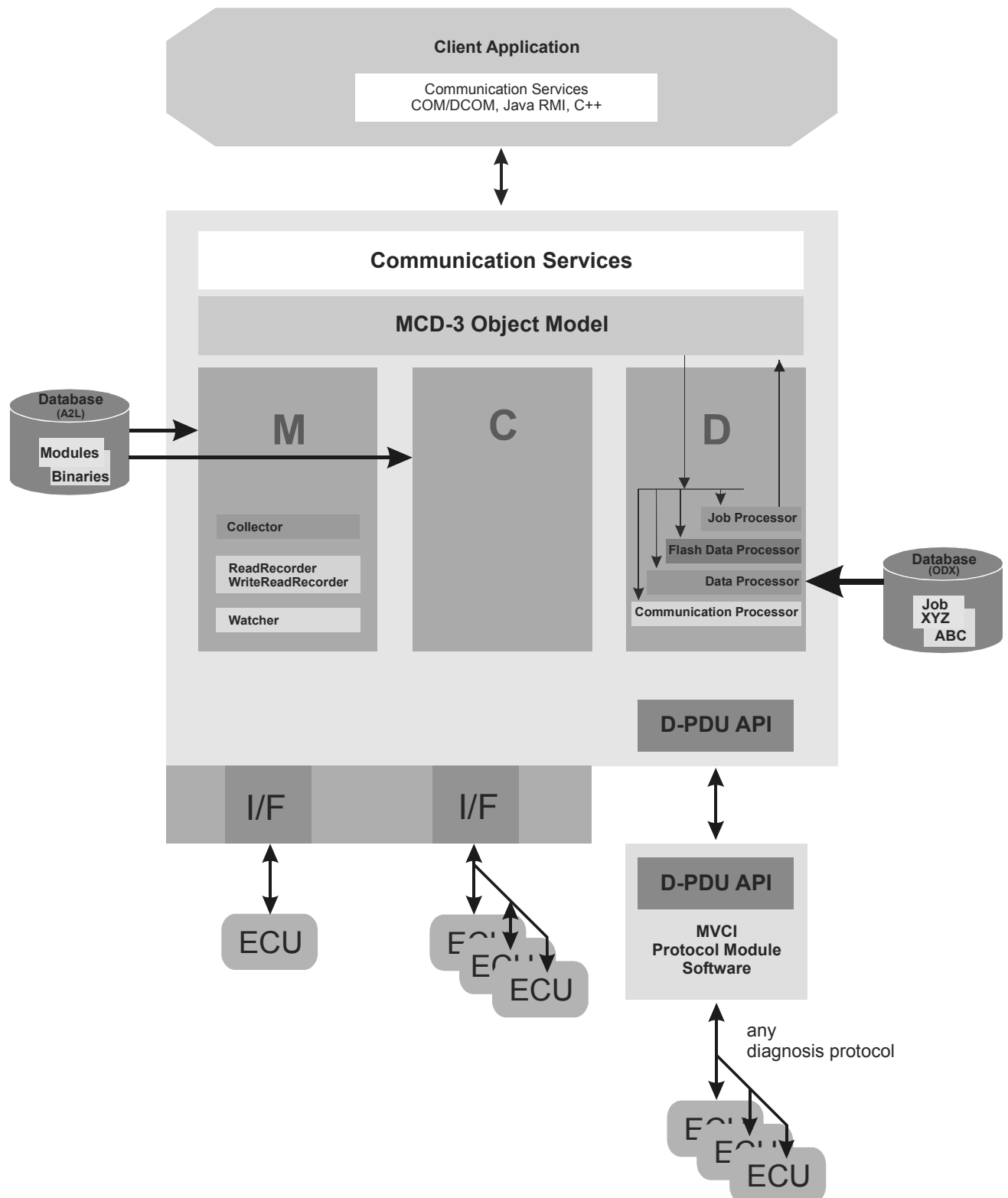


Figure 2 — Architecture of a MCD System

MCD-servers have three function blocks:

- Measurement (measurement sequences),
- Calibration (adjustment tasks) and
- Diagnostics (diagnostic services).

To realize the separate block functionalities, they shall always be accessed via the respective function block.

Within a real server, not all function blocks have to be supported. The function blocks are used for the handling of control units (ECU s ) within vehicles for the realization of measurements, adjustments and for diagnostics. With the help of MCD-server the control units are optimally adapted to the relevant requirements for their use in vehicles. This procedure is often referred to as “Applying”.

The MCD-3 Object Model shall enable the implementation of M-, C-, D-, MC-, CD-, MD- and MCD-server:

- A future MC-server shall implement M-functionality following the collector principle defined in the MCD-3 MC Object Model.
- If there exists a Diagnostic service which allows reading and writing variables on ECU this is not seen as the standard MC-server functionality but as a special form of a service.
- If there exists a Diagnostic service which allows free-running data exchange between ECU and D-server this is not seen as the standard M- functionality but as a special form of a service.
- A future D-server shall implement D-server functionality following the communication primitive principle defined in the D object model.

MC-server, which meet the requirements of this specification do not necessarily need a user interface (GUI), but still may have one.

All interfaces and methods inside a function block are mandatory! Exception: MCDDbProjectconfiguration, ECU Configuration, ECU Flash Programming, DDLID, support of Diagnostic Variables and monitoring in diagnostic area is optional. Optional means that the runtime- as well as the database part of the model do not have to be implemented by a D-server that is omitting the feature in question. In case a client application calls a method that is part of an optional feature, the D-server should return an empty collection in case the method's return type is a inheriting from MCDCollection. Otherwise, such a method call should throw a MCDSysException of type eSYSTEM\_METHOD\_NOT\_SUPPORTED.

The number of control units applied in vehicles is continuously increasing. The capabilities of the single control unit concerning measuring, adjusting and diagnostics become available for the MCD-server by means of control unit description files (ASAM MCD 2 database<sup>[11],[12]</sup>). The control unit description files represent a manufacturer independent data exchange format, that means any MCD-server may handle the data out of a control unit description file. All configuration data of the MCD-server, the internal data of ECU s or ECU nets and the communication methods for the ECU access are stored in the ASAM MCD 2 database. This database is MCD-server and operating system independent and therefore allows data to be exchanged between vehicle manufacturers and ECU suppliers.

An application can read out all data from the database that is necessary to drive the MCD-server, that means only the MCD-server can access the information of the separate control unit description files comprised within one database. With this, at the same time the consistency of the information between AUSY and MCD-server is guaranteed.

Also, a decoupling from the used data description exchange format (a2l or XML) takes place.

The MCD-server shall manage the database and to provide for the required and necessary information for the single Objects. The database does not belong to one specific Object, but is available within the whole MCD-server. The organization of the object or reference allocation is solved implementation specifically by the MCD-server.

Standardized drivers are used to access to the most different control units, to enable the physical and logical linkage of control units to MCD-server.

The object model supports Multi Client Systems. Although, the design has been oriented in accordance with a Single Client System, to provide for a simple use for this most typical and most frequently occurring application case. That means, that no client references are included within the single objects. The administration of the client references is done by the MCD-server and shall be solved implementation specifically.

The object model has been designed in a technology and programming language independent way. It may be used remotely as well as locally.

The object model enables the linking of MCD-server to automation systems (MCD-server API). Objective of this linking is the remote control of the MCD-server in test stands. By means of the object model the functionality is standardized, that means the interfaces for the function blocks with accompanying methods are standardized. The communication shall be realized via the particular implementation of the object model for the used platform, programming language and linking mechanisms.

Among others realized are:

- JAVA,
- COM / DCOM and
- C++.

The necessary specifications for this will be described and published in separate documents. For this process design patterns and mapping rules are defined and published.

All other specifications will be set up and realized implementation specifically by the respective system provider.

Within the function block diagnostic a breaking down into characteristic sub tasks will take place, which are shown in the following:

#### The Communication Processor

Is responsible for generating and analysing of Request and Response telegrams to ECU s. This processor handles all protocol specific tasks like timings, creation of protocol-headers and -checksums, etc. Diagnostic protocol specification is (at the moment) not a task of ASAM, because this is covered by ISO activities like KWP2000 and "Diagnostics on CAN". Nevertheless, the communication processor shall be parameterised via ODX using Communication Parameters. The Communication Processor is an interface (the only one) to the ECU.

#### The Data Processor

Is responsible for the supply of parameters and results on physical level. By means of the Data Processor all necessary information is fetched from the database. Additionally, the Data Processor converts ECU answers from hexadecimal representation into a physical or any text representation and vice versa. The Data Processor is an interface (the only one) to the ODX data and offers an library to the Job Processor. The Data processor also handles Jobs, as they are stored in ASAM MCD 2 D ODX.

#### The Flash Data Processor

Is responsible for the loading of programs and data in ECU s. The flash data is part of the database. The Flash Data Processor provides access to the ECU-MEM which contains all information about physical / logical data- / code-layout and possible combinations of data and code segments and more. The Flash Data Processor is an interface (the only one) to the flash data and offers a flash library (flash object) to the Job Processor.

#### The Job Processor

Is responsible for the execution of service sequences and only uses objects of the MCD-server API. Via the Job Processor all processors may be accessed. The Jobs are part of the database. The Job Processor is

based on the ODX format. The Job Processor provides several libraries for standardized access to the Communication Processor, Data Processor, Flash Data Processor and to the Job Processor itself. The Job Processor uses the same objects to interact with the different Processors like the D-server API to insure consistency between D-server API and Job Processor. The Job Processor gets its code to be executed from the Data Processor. The Data Processor itself reads the ODX files.

## 7 Function block common MCD

### 7.1 MCD system object

The server is only accessible by a certain number of clients at the same time. The number of allowed clients can be requested via the method `getMaxNoOfClients`.

Every Client gets its own `MCDSystem` object (implements the `MCDSystem` interface) from the MCD-server. But all Clients work on the same project and database. The runtime `MCDProject` has the connection to a certain `MCDDbProject` which along with its sub items will be available after selecting the project. The selection of another project at the same time is not allowed and will throw an exception.

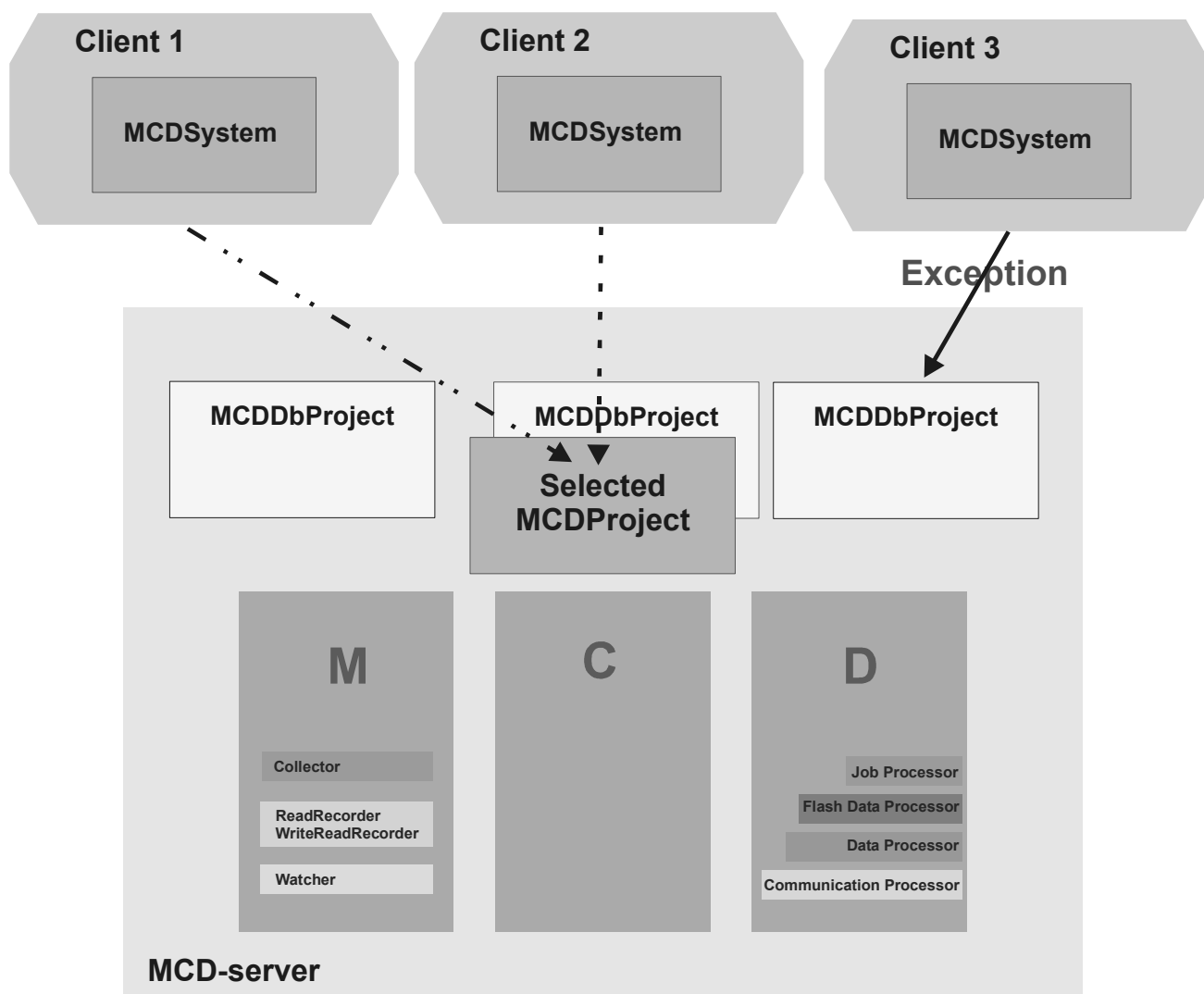


Figure 3 — System scheduling

At the moment, the other diagrams within the specification always refer to the representation within the client and are not designed for MultiClient scenarios, except where noted otherwise.

## 7.2 Version information retrieval

The method `MCDSystem::getASAMMCDVersion():MCDVersion` returns the ASAM release number of the interface. For this specification it is the major value 2 and the minor value 1 defined.

The method `MCDSystem::getVersion():MCDVersion` returns the tool version.

All function blocks implemented in a tool use the same ASAM version.

The technology version number is available via the interface

`MCDSystem::getInterfaceNumber():A_UINT32`.

The version number of the model (specification) and the version number of the interface definition are kept synchronous. The so-called “interface number (IFN)” is used as a build number for the interface definition. Each interface definition maintains an own interface number. This number will be incremented continuously within a minor version. Increments are applied whenever a new interface definition is generated and shipped. The “interface number” is reset to zero for each new minor or major version. Please, note that this number is completely independent from the version numbers. The “interface number” is incremented separately for each technology.

## 7.3 Description of Terms

### 7.3.1 General

The key terms and definitions which apply in this part of ISO 22900 are given in Clause 3. This subclause describes the most important terms in more detail.

### 7.3.2 Client-controlled object

A client-controlled object is an object that is passed to the client and modification of these object does not result in a direct modification of objects within the server. All values in a MCD-server, which have an ASAM data type, are considered to be client-controlled. The client is responsible for the deletion of client-controlled objects.

### 7.3.3 Location

A `MCDDbLocation` is an entry point for available Services, DiagComPrimitives, Measurements, Characteristics and other data of an ECU. Please note, that for the location no `AccessKey` is used.

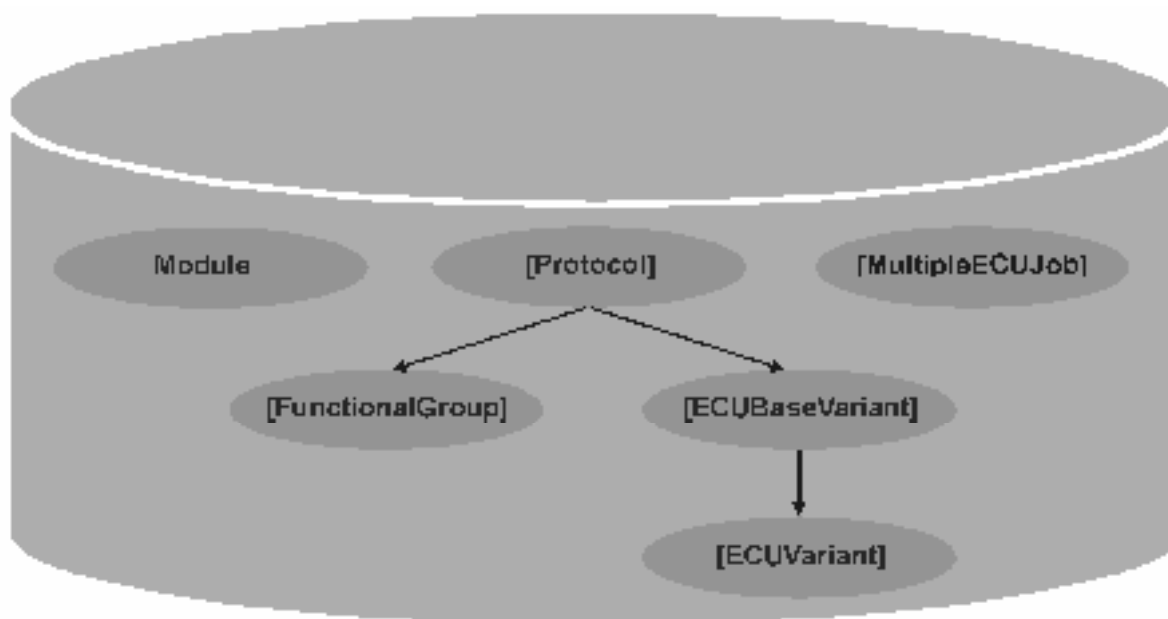


**Figure 4 — Location hierarchy of ASAM MCD database**

The MCDDbLocation represents the contents of the corresponding description file, e.g. ODX or A2L file.

#### 7.3.4 Logical Link (LOGICAL-LINK)

A Logical Link represents the physical line, the used interface and protocol of an ECU as one object in an MCD-server. Typically, the Logical Link represents the access to only one ECU, but in case of a Functional Group it may also describe more than one ECU. The Logical Link is identified by a short name. Information about a Logical Link is contained in the Logical Link Table. Elements of this table are the AccessKey and the Physical Vehicle Link or Physical Interface. Logical Links may also be used to access the same ECU on different ways, or access more than one ECU on different links. For more details see the chapters Function block Diagnostics. Logical Links can be locked for exclusive use by one client (for more information see Locking).

#### 7.3.5 Project

A project is a logical grouping for defined specific installations selected by the user. Within a project, all information necessary for a test installation shall be contained. It is only permitted to work within one project, this shall be considered for the logical grouping (e.g. two model series within one project). That is why the project tuple is not part of the AccessKey.

At project level, the forming of manufacturer specific hierarchies is possible, as for this level no standardization takes place.

Typically, a project contains

- all static database information (Measurements, Characteristics, Diagnostic Services, ...),
- jobs,
- flash containers and
- configuration files.

Within the MCD-server, first the selection of a project out of the list of the existing projects takes place, before any database information can be accessed.

### 7.3.6 Server-controlled mutable object

A server-controlled mutable object is a server-controlled object that provides methods for clients for locking, access restriction, change notification and change of the object state, properties, or the content of owned sub-objects. The server-controlled mutable objects can be divided in primary and secondary server-controlled mutable objects. A primary server-controlled mutable object is a root node of a set of objects which belong together because they build a complex semantic object. The server-controlled mutable objects that are contained in a primary server-controlled mutable object are called secondary server-controlled mutable objects. The set of objects are only accessible through the object hierarchy under the root node and contain all objects until no child object is available or the child object is a primary server-controlled mutable object.

### 7.3.7 Server-controlled object (shared object)

A server-controlled object is an object that is not a client-controlled object in the object model. The MCD-server is responsible for the deletion of server-controlled objects.

## 7.4 States of the MCD system object

Within the state `eINITIALIZED` the `MCDSys` object is transmitted to the client by the MCD-server. Within this state, the database project descriptions can be listed and general system initializations can be done.

By selecting the project to be worked on, the system takes the state `ePROJECT_SELECTED`. Within this state the database can be polled for information, but no communication to the hardware is possible.

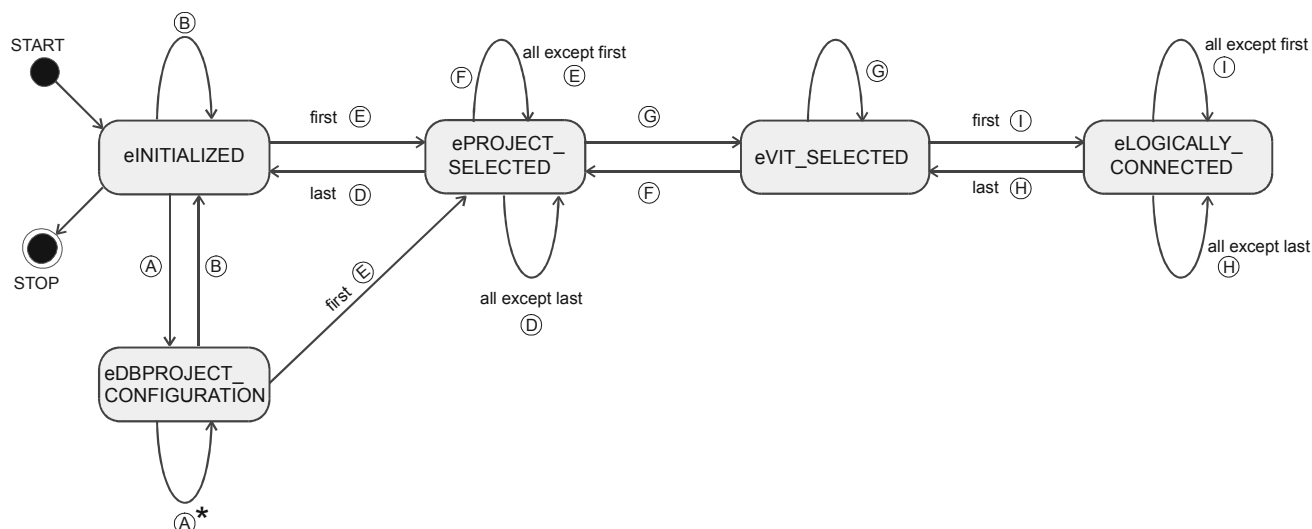
With the selection of a `VehicleInformationTable` the state is changed to `eVIT_SELECTED`.

As soon as the first Logical Link has been created in an MCD-server, the `MCDSys` object's state changes to `eLOGICALLY_CONNECTED`. In state `eLOGICALLY_CONNECTED`, communication to an ECU can be established by using the available Logical Links. If the last Logical Link has been removed from the runtime project, the system automatically takes the state `eVIT_SELECTED` again. With deselection of the `VehicleInformationTable` the state `ePROJECT_SELECTED` is taken. By means of `deselectProject()` the `MCDSys` Object is set to the initial state `eINITIALIZED` again.

The MCD-server changes the state to `eDBPROJECT_CONFIGURATION` by loading a database project at object `MCDDbProjectConfiguration`. In this state, searching in the database is possible. In MC-servers, `MCBinaries` and `Modules` can be added. In D-servers, `ECU-MEMs` or `CONFIGURATION-DATAs` can be added (removed only in state `eDBPROJECT_CONFIGURATION`) at any system-state except `eINITIALIZED`. The corresponding `MCDDLogicalLink` could be in any state. If additional data is added, a system event will be thrown (`onSystemDbEcuMemsModified/onSystemDbEcuConfigurationDatAsModified`), so the clients are informed about these changes. The collection `MCDDbEcuMems/MCDDbConfigurationDatAs` shall be reloaded by the client at the `MCDDbLocation` or at the `MCDDbProject`. `ECU-MEMs` or `CONFIGURATION-DATAs` should be loaded temporarily to a `MCDDbProject` or permanently added to a project configuration.

`ECU-MEMs` are necessary for (re-) programming of an ECU. `CONFIGURATION-DATAs` are necessary for configure an ECU.

In case of MC-server it is also possible to add new `MCDDbProject` objects at the interface `MCDDbProjectConfigurations` as long as the system is in state `eDBPROJECT_CONFIGURATION` or `eINITIALIZED`.



(A) MCD MCDDbProjectConfiguration: load

\* load or add of DbProject includes an implicit close of actual opened DbProject

(B) MCD MCDDbProjectConfiguration: close

(D) MCD MCDSysyem : deselectProject

(E) MCD MCDSysyem : selectProject

(F) MCD MCDProject : deselectVehicleInformation

(G) MCD MCDProject : selectDbVehicleInformation  
MCD MCDProject : selectDbVehicleInformationByName

(H) D MCDLogicalLinks : remove  
MCD MCDLogicalLinks : removeByName  
MCD MCDLogicalLinks : removeByIndex  
MCD MCDLogicalLinks : removeAll

(I) D MCDLogicalLinks : addByAccessKeyAndInterfaceResourceLink  
D MCDLogicalLinks : addByAccessKeyAndPhysicalInterfaceLink  
D MCDLogicalLinks : addByAccessKeyAndVehicleLink  
D MCDLogicalLinks : addByDbObject  
D MCDLogicalLinks : addByName  
D MCDLogicalLinks : addByVariant

**Figure 5 — System state transitions**

The states of clients and the central state of the server (internal MCDSysyem object) shall be distinguished.

At the server side there may only be one state (eINITIALIZED, eDBPROJECT\_CONFIGURATION, ePROJECT\_SELECTED, eVIT\_SELECTED, eLOGICALLY\_CONNECTED). As soon as a client has successfully called selectProject, the internal MCDSysyem object transits to the state ePROJECT\_SELECTED.

MCDProject/MCDSysyem LongName and Description are server specific values and can be empty. An exception will not be used.

Table 1 — System states

|              | MCDSystem       |                     | MCDDbProjectConfiguration |      |       |                          | MCDProject                          |               | MCDLogicalLinks |                            |                                  |                            | MCDDbProject                 |               |           |            |              |              |                                    |        |                |               |           |                          |                                    |               |                          |                                |                    |
|--------------|-----------------|---------------------|---------------------------|------|-------|--------------------------|-------------------------------------|---------------|-----------------|----------------------------|----------------------------------|----------------------------|------------------------------|---------------|-----------|------------|--------------|--------------|------------------------------------|--------|----------------|---------------|-----------|--------------------------|------------------------------------|---------------|--------------------------|--------------------------------|--------------------|
| System State | selectProject   | selectProjectByName |                           |      |       |                          |                                     |               |                 |                            |                                  |                            |                              |               |           |            |              |              |                                    |        |                |               |           |                          |                                    |               |                          |                                |                    |
|              | deselectProject |                     | add                       | load | close | getAdditionalEcuMemNames | getAdditionalConfigurationDataNames | removeByIndex | removeByName    | selectDbVehicleInformation | selectDbVehicleInformationByName | deselectVehicleInformation | addByAccessKeyAndVehicleLink | addByDbObject | addByName | addByNames | addByObjects | addByVariant | addByAccessKeyAndInterfaceResource | remove | removeByName   | removeByIndex | removeAll | loadNewConfigurationData | loadNewConfigurationDataByFileName | loadNewEcuMem | loadNewEcuMemsByFileName | removeConfigurationDataByIndex | removeEcuMemByName |
|              | X               | X                   | X                         | X    |       |                          | X                                   |               |                 |                            |                                  |                            |                              |               |           |            |              |              |                                    |        |                |               |           |                          |                                    |               |                          |                                |                    |
|              | X <sup>1</sup>  | X                   |                           |      |       |                          |                                     |               |                 | X                          | X                                |                            |                              |               |           |            |              |              |                                    |        |                |               |           |                          | X                                  |               |                          |                                |                    |
|              |                 |                     |                           |      |       |                          |                                     |               |                 | X <sup>2</sup>             | X                                |                            | X                            |               |           |            |              |              |                                    |        | X <sup>3</sup> |               |           |                          | X                                  |               |                          |                                |                    |
|              |                 |                     |                           |      |       |                          |                                     |               |                 |                            |                                  |                            |                              | X             |           |            |              |              |                                    |        | X <sup>3</sup> |               |           |                          | X                                  |               |                          |                                |                    |
|              | X               |                     |                           | X    | X     | X                        |                                     |               |                 |                            |                                  |                            |                              |               |           |            |              |              |                                    |        |                |               |           |                          | X                                  |               |                          | X                              |                    |

**Key**

|                |                                                                              |
|----------------|------------------------------------------------------------------------------|
| X <sup>1</sup> | only identical project                                                       |
| X <sup>2</sup> | only identical VIT                                                           |
| X <sup>3</sup> | It is only allowed to remove a LogicalLink if it is in the state D: eCREATED |

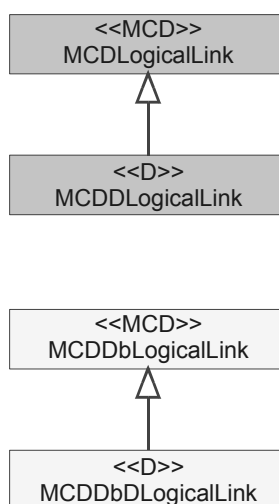


Figure 6 — Class diagram of MCDDLogicalLink

**7.5 State changes**

State changes in an object are triggered directly via method calls or indirectly due to internal state changes of the MCD-server. A successful state change is transported via events to the clients. These events transport the object, which is changed, and the previous state of the object. A method, which triggers a state change, will throw an exception if the state change could not be processed successfully and the state is not changed. Feedback transitions, state changes to the previous state, do not trigger any events and do not throw an exception.

## 7.6 Project configuration

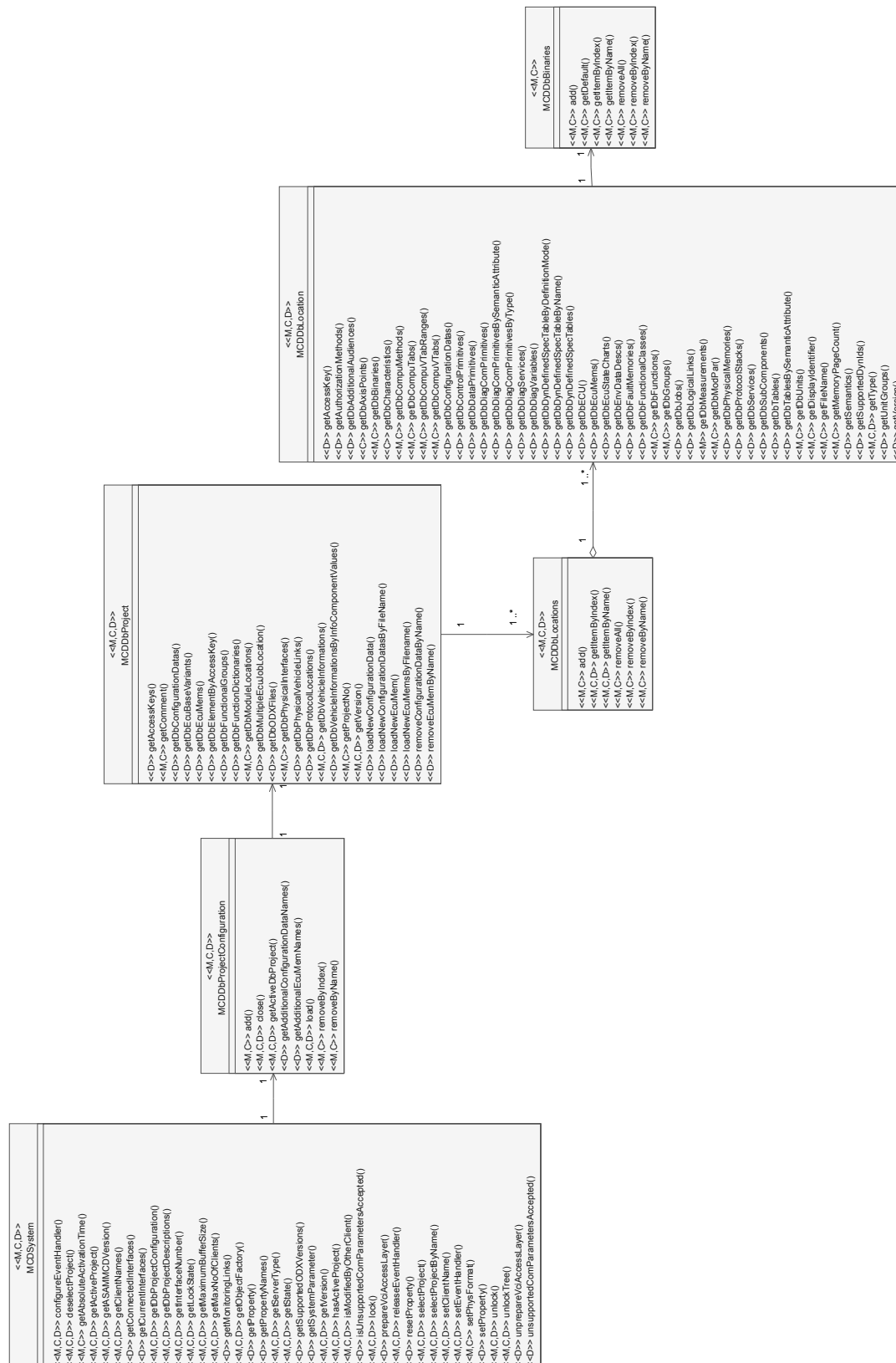


Figure 7 — Project configuration

Within the state `ePROJECT_CONFIGURATION` of the MCD-server a database project may be loaded and browsed. Runtime objects may not be created within this state. Once the transition to this state has taken place, this state can only be left by closing down the database project or selecting the corresponding runtime project. This selection implicates the saving of all changes, which were made. In this state the used database project can be changed by loading or creating another database project, the so far used project will be implicitly saved if changes were made.

The Method `MCDSysyem:getDbProjectConfiguration` returns a `MCDProjectConfiguration` Object, which can be used for

- browsing (without existing `RuntimeObjects`, `MCD: MCDProjectConfiguration:load`) and
- modification (`MC: MCDDbProjectConfiguration:add`, `MC: MCDDbBinaries:add`, `MC: MCDDbLocations:add`, `D: MCDDbProject:loadNew...`) of `DbProjects`.

The state transition from `eINITIALIZED` to `ePROJECT_CONFIGURATION` takes place only after a Client opens a `DbProject` by means of `add/ load`.

MC-server may also delete Modules or Binaries. The deleting of database projects takes place within the state `eINITIALIZED`.

That means, that the MC-server may at first create a database, then use this database for measuring and/or calibration and then delete this database again. A modification of the database during measurement/calibration is not possible, as the MC-server is not within the state `eDBPROJECT_CONFIGURATION`.

The Methods:

- `MCDSysyem::selectProject()`,
- `MCDDbProject::loadNewEcuMem()`,
- `MCDDbProjectConfiguration::getAdditionalEcuMemNames` and
- check the consistency of data read from the database.

An error of type `eDB_INCONSISTENT_DATABASE` should be thrown in case inconsistent data has been identified.

A client cannot change the server state to `eDBPROJECT_CONFIGURATION` if more than one client is connected to the server. In this case, the methods `MCDDbProjectConfiguration::add()` and `MCDDbProjectConfiguration::load()` throws an exception of type `MCDSysyemException` with error code `eSYSTEM_NOT_ALLOWED_IN_MULTI_CLIENT_MODE`. If a client has successfully changed to the state `eDBPROJECT_CONFIGURATION`, no further clients can connect to the server. How the error is transported to the client is defined in the interface definition.

## 7.7 Interface structure of MCD-server API

### 7.7.1 Separation in database and runtime side

Within the whole model there is a separation into database and runtime objects.

These interfaces are colour coded in all diagrams:

- in colour prints, yellow (database) and green (runtime);
- in black/white prints, light (database) and dark (runtime).

Apart from the configuration within the MC function block the database part is invariable and is used as the basis for the creation of most of the runtime objects. Database objects / classes contain the sub string "Db" in their names. This sub string signals that the corresponding class is associated with information originating from a data file (ODX, Flash Data, a2I, etc.). Db objects cannot be modified by the client application. A Db object is static, has no status and exists only once.

All access is done by methods of the interfaces. There are no attributes except in the classes which should behave like an enumeration. These classes have the Stereotype <<enum>>.

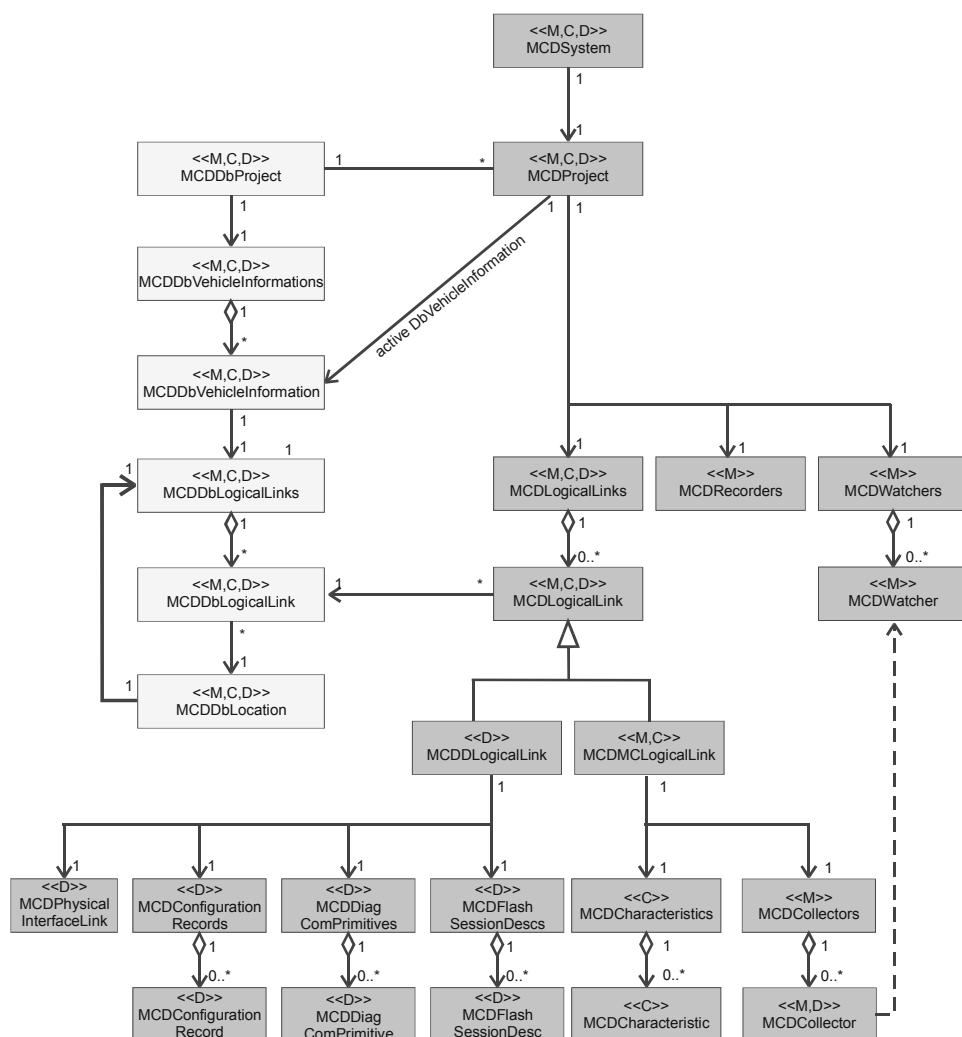


Figure 8 — Project associations

The entry object is the MCDSysSystem, a runtime object. From this a runtime project will be created on base of its database project. Selecting the Vehicle Information Table makes the access to all database Logical Links possible. This access is necessary to create the runtime Logical Links from which all runtime activities can be done.

### 7.7.2 Hierarchical model

The figure shows only the interfaces available for MCD. In addition to these, there exist also those for MD, MC, M, C and D. The interfaces marked with the Stereotype MCD shall be implemented by each MCD-server, no matter what type it is.

As can be seen, all database interfaces are derived from `MCDDBObject`, and all database Collections from `MCDNamedCollection`. The Runtime interfaces usually are derived directly from `MCDObject`. Collections, Exceptions and NamedObjects each form basic classes for a further detailed structuring. Apart from `MCDSystem` and `MCDProject` also all database interfaces belong to the NamedObjects.

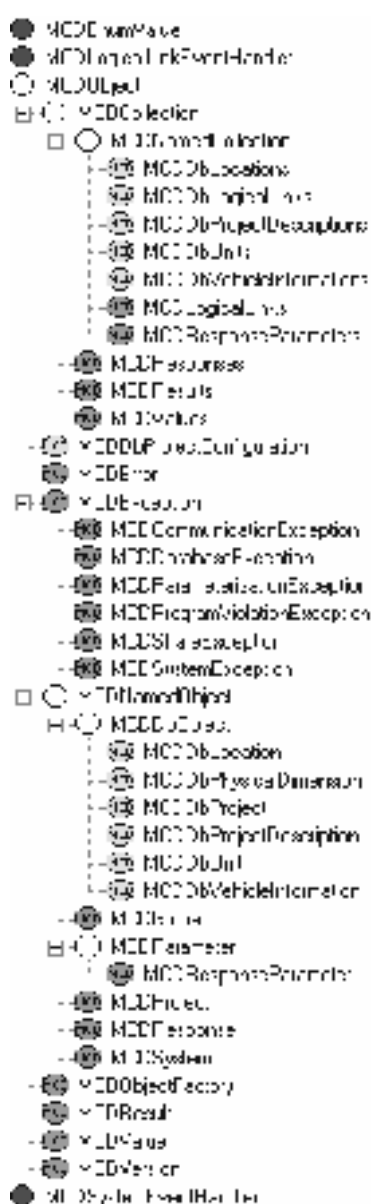


**Figure 9 — Hierarchical model of a MCD-server**

The `MCD[ObjectType]EventHandler` classes are an exception concerning this classification and are not derived from `MCDObject`. These are used as means of communication of the MCD-server with the client and provide callback methods for the event handling.

The `MCDEnumValue` (realized as a singleton or single instance class) is a global converter class for all classes representing enumerations in the API. The class `MCDEnumValue` holds static information only and is considered to be stateless. It provides two methods to translate between the object and the string representation of an enumeration value and vice versa. The implementation of the class `MCDEnumValue` is specific to the interface definition. Therefore, its realization with respect to one of the interface definitions is described in the corresponding rules and regulations document.

Apart from the different `EventHandler` interfaces (the `MCDEnumValue` is a class not an interface) all interfaces of the ASAM MCD3 model are derived from `MCDObject`.

In the MCD-3 object model only classes without inheriting child classes can be instantiated. For example, the method `MCDDbLocation::getDbEcu` returns an object of type `MCDDbEcu`. A call to this method only may return an object of type `MCDDbEcuBaseVariant`, `MCDDbEcuVariant` or `MCDDbFunctionalGroup`. The `MCDDbEcu` class that these classes are inheriting from can be considered as abstract and no object of this class shall exist. This is valid for all applicable cases except where explicitly noted.

## 7.8 Structure of the database

### 7.8.1 Overview

The database, which can be polled via the Object Model, is structured in such a way, that each database element is available only once. Because of this, redundancies are avoided. Connections between the single database elements can be queried via methods. These methods return instances or references.

All database interfaces have methods for the polling of `ShortName`, `LongName` and `Description`.

A more detailed description can be found with the description of the hierarchic model and the Entity Relationship diagrams.

Database information may be accessed immediately in state `eDBPROJECT_CONFIGURATION` or after the instantiation of the runtime project.

### 7.8.2 Associations of `DbLocation` for MCD

Within this ERD the associations between a `DbLocation`, and the related interfaces are shown. At the `DbLocation` a number of collections can be accessed, which depends on the association of the `DbLocation` to M, C, D or MC and MD may be empty or filled with items. The methods are marked with the respective stereotypes, so that the implementation depends on the type of the server. It shall be noted, that in case of MC also the collections of M and C are accessible. The same applies for MD, where also the collections of M and D are accessible.



Figure 10 — DbLocation associations

### 7.8.3 Database within the field Measurement and Calibration

Represents the information from the underlying database description, e.g. parts of an A2L file<sup>[12]</sup>.

### 7.8.4 Database within the field Diagnostics

The database of an D-server is based on the supported parts and elements of ODX<sup>[11]</sup> and related information as external flash data files and Java Job libraries. That is, the database part comprises, e.g. information on projects, vehicle information tables, logical links, and diagnostic services.

## 7.9 Collections

### 7.9.1 Types and methods

A collection is an object, which manages a list of objects of similar kind. Single objects within the collection may be accessed by means of an index. The index may also be used to iterate over all elements of the collection. The ordering criteria are not defined by the MCD-3 specification. Hence the index of an object could change during the lifetime of the collection. The start index in collections is always zero and the maximum usable index could be queried with the `getCount` method located at a collection and subtracting 1.

Access to collection items via index is considered harmful. Every time a server-controlled collection is retrieved, the index stays constant, as long as

- the same server instance is used,
- no internal change in the collection has occurred in the server, e.g. the server itself calls `add/remove` on the collection, and
- no `add` or `removed` method is called in the collection by any client or the project has been deselected.

The interface `NamedCollection` has been derived from the general collection. Named collections additionally offer the possibility to query a list with the names of all elements currently located within the collection (`getNames`) and to access an element via its name (`getItemByName`). The name inside a named collection is always unique, therefore the name could be stored during the lifetime of the collection.

If a collection provides an “add”-method, this method might throw an exception of type `MCDProgramViolationException` with Error Code `eRT_ELEMENT_ALREADY_EXISTS`, if the element to be added is already contained in the collection. Whether a collection throws this exception or not depends on the semantics defined for this specific collection.

**NOTE** Collections in MCD-3 have the semantics of a set, that is, every element can only be contained once. Here, it is sufficient that the elements in the collection can be distinguished uniquely by their shortname. Corresponding name generation rules for elements in runtime collections are defined in 7.9.2.2 and 7.9.2.3.

This semantic allows to uniquely identify if a client is the “master” of a resource. Only if the `add`-method returns without throwing an exception, the client that has called the `add`-method is considered the “master” of this particular resource.

The semantic of a `<<M,C,D>>removeAll`-method at a collection is “all or nothing”. That is, if at least one element cannot be removed because of insufficient rights, no element is removed but an exception of type `MCDProgramViolationException` with error code `eRT_INSUFFICIENT_RIGHTS` is thrown.

Some (database and runtime) collections have “`removeByXXX`” methods by which specific single items can be removed. After successful call of such a method the remaining items move one position up in the collection. As an consequence the index of the remaining items may change after calling such an method!

All objects in the collection `MCDLogicalLinks` can always be obtained by any client – independent of the cooperation level. That is, the methods `getItemByName()` and `getItemByIndex()` always return an (proxy)

object. However, the cooperation level of the returned object might restrict the access to some methods at the proxy object.

A client being able to access a Logical Link (LL has cooperation level of at least eREAD\_ONLY), is also able to obtain all elements in the respective MCDDiagComPrimitives collection, the respective MCDCollectors collection, and the MCDCharacteristics collection. However, the cooperation level of the returned object might restrict the access to some methods at the proxy object.

A client being able to access a Collector (Collector has cooperation level of at least eREAD\_ONLY), is also able to obtain all elements in the respective MCDCollectedObjects collection. However, the cooperation level of the returned object might restrict the access to some methods at the proxy object.

This solution allows iterating through elements of a collection without causing exceptions to be thrown due to this kind of access.

**NOTE** The owner (master) of an object in a collection can decide to remove this object from the collection, which causes other clients proxy objects to be marked invalid.

A method returning a collection shall not throw an exception because an empty collection will be returned. All collections will be handled in exactly the same way.

As a general design rule it has been defined that get methods returning a collection shall never throw an exception, e.g. because the collection is empty. However, when accessing the database, it is always possible that inconsistencies in the database are detected by the MCD-server. This also applies to get methods for collections. As a result, the cases 'empty collection' because no data available in the database and 'inconsistent database' need to be distinguished.

To solve this, it has been proposed that all methods accessing database objects (getDbXXXX()) can throw a MCDDatabaseException with error code eDB\_INCONSISTENT\_DATABASE.

The method `getItemByIndex(A_UINT32 index)` throws an exception of type `MCDParameterizationException` with error code `ePAR_INDEX_OUT_OF_RANGE` in case the value supplied for the parameter `index` is greater or equal to the number of elements in the collection. This does also apply in case of an empty collection.

The numbering in collections starts at zero.

The method `getItemByName(MCDDatatypeShortname name)` throws an exception of type `MCDParameterizationException` with error code `ePAR_ITEM_NOT_FOUND`. This does also apply in case of an empty collection.

## 7.9.2 RunTime collections

### 7.9.2.1 Overview

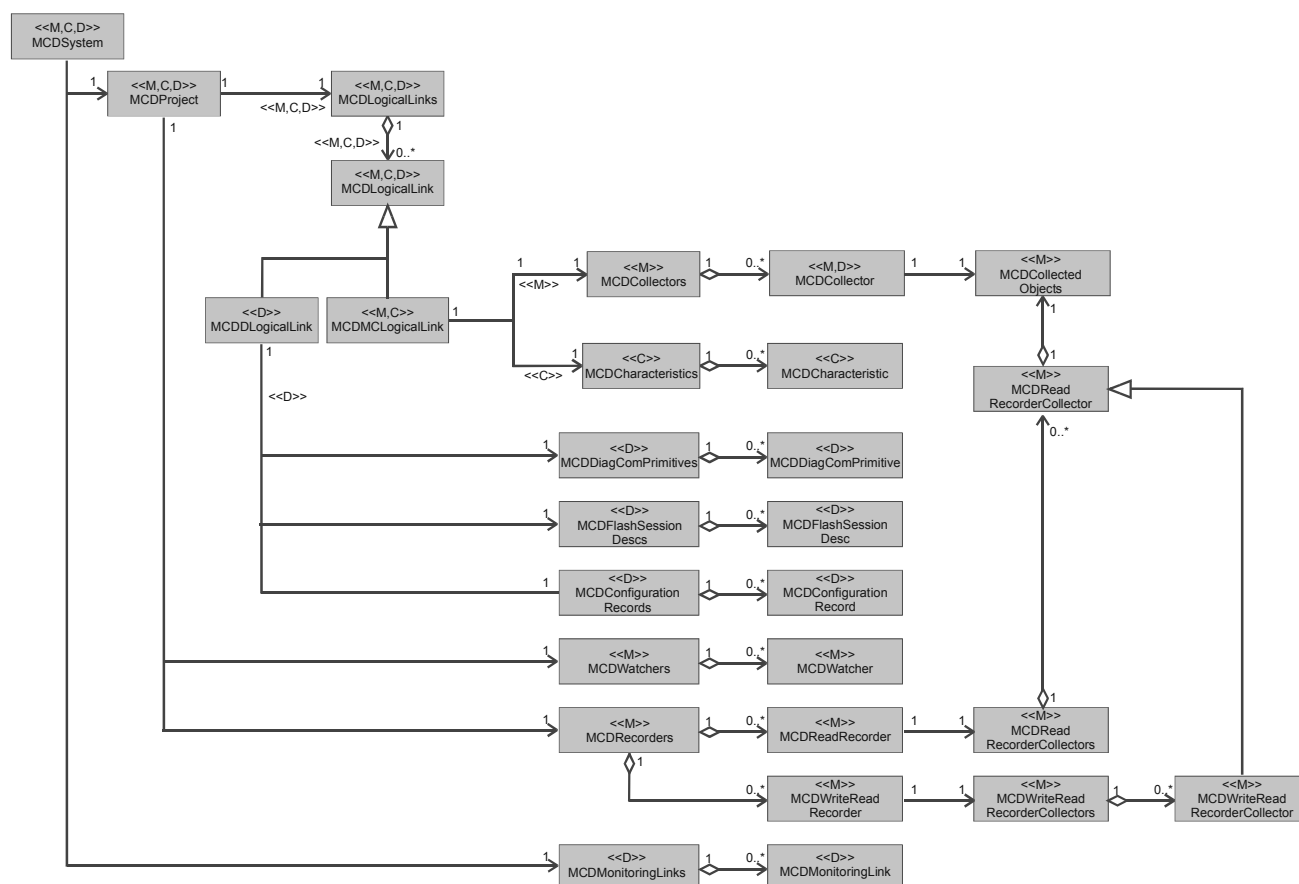


Figure 11 — RunTime collections MCD

Within this ERD the associations between `MCDProject` and its Logical Links are shown. On the one hand the relations of the runtime project to the `DbProject` and to the selected Vehicle Information Table are shown and on the other hand the relations of the Logical Link to the Collectors (M), Characteristics (C) and DiagComPrimitives and Services (D). Depending on the type the Logical Link is assigned to, it may create and use the respective runtime objects. The logical link type is determined by the Db object that is used for its creation.

In general, obtaining runtime collections does not cause any exception. If no corresponding information is available, an empty collection is returned.

If a database collection is obtained, an exception can be thrown in case the database is inconsistent. If the corresponding information is not available in the database, an empty collection is returned. In case an error has been detected in the database when trying to obtain the information, an exception of type `MCDDatabaseException` with error code `eDB_INCONSISTENT_DATABASE` is thrown.

If the access to a method to obtain a collection is restricted because of cooperation levels or because of locks, corresponding exceptions can be thrown:

- the called object or an object above in the object hierarchy is locked by another client (`eRT_OBJECT_IS_LOCKED`, `MCDProgramViolationException`);
- the method can't be called because of insufficient rights of the non-master client in a multi-client scenario. (`eRT_INSUFFICIENT_RIGHTS`, `MCDProgramViolationException`).

### 7.9.2.2 RunTime collections in MC

For all run time collections within the function block MC applies: Each Client may see and use all items of a Collection existing within the Server, independently from the fact which Client has created the Items or uses the Items. Because of this it is e.g. not possible to create several Runtime Characteristics for a database Characteristic and to parameterise them differently. That means, that existing Runtime Characteristics may be parameterised as new by each Client if no safety mechanisms are intended (locking).

There is the method `removeAll`, which deals with all objects in the collection. The MC-server will try to remove each of the objects. The objects which cannot be removed (and only these) will be left in the collection. After returning from this method the Client can call `getCount` to get the number of objects, which were not removed, and `getItemByIndex` to get the items itself. The other method depending on all objects of their collection is `MCDCollectors::deactivate():MCDCollectors`. This method will create and return a temporary collection of Collectors which could not be deactivated.

### 7.9.2.3 RunTime collections in D

At D-server runtime, the runtime counterpart of a DB-object can occur multiple times in the same collection, e.g. a `MCDDiagComPrimitive` in the collection `MCDDiagComPrimitives` or a `MCDResponseParameter` in the collection `MCDResponseParameters`. However, as the majority of these collections are derived from class `MCDNamedCollection`, a unique `ShortName` for the different objects is required.

While a collection can contain multiple runtime objects referring to the same DB-object, the following pattern is to be used to generate the `ShortName` of the runtime objects contained in this collection: `#RtGen_<ShortnameOfCorrespondingDbObject>_<Number>`.

The number in this generated `ShortName` is increased with every new instance of the same DB-object in the same collection. For the first instance, the number is zero. This pattern is also used if only a single instance of a DB-object is part of a collection. As this fact is nondeterministic, it cannot be predicted by the D-server whether more instances of the same DB-object will be added at a later point in time. The name of the corresponding database objects can still be obtained by using the `getDbObject():getShortName()` methods found on the runtime objects with generated short names.

The following collections shall apply this generation rule for their runtime elements:

- `MCDResponseParameters`, if the containing `MCDResponseParameter` is of type `eFIELD`;
- `MCDRequestParameters`, if the containing `MCDRequestParameter` is of type `eEND_OF_PDU` or `eFIELD`;
- `MCDDiagComPrimitives`;
- `MCDFlashSessionDescs`;
- `MCDConfigurationRecords`.

**NOTE** The collection `MCDLogicalLinks` is not affected, as there can only be a single instance of each logical link at runtime.

Service result structures and their parameter structures are also mapped to runtime collections. However, there is no danger that names are assigned twice in these collections, as they are always created by the D-server or a Java job.

Besides the naming rules for multiple runtime instances of Db-elements, another issue that needs consideration is the question of how to deal with indexes of elements in a runtime collection in case an element is removed. In this case, the index values of the remaining items will be recalculated to fill any gaps that are left by the removed items. For example, if an item at index/position 1 is removed from a collection

which contains 3 items [at indexes 0, 1, 2], after the removal of the item the collection will contain 2 elements with indexes 0 and 1.

### **7.9.3 Database collections**

Collections are used for the listing of identical database objects. The Collections reflect the content of the database of the used project. The database is static; that means nothing can be added (except for MC within the configuration phase and except for D for ConfigurationDatas and ECU-Mems) or modified. The collections are always derived from `MCDNamedCollection`, so that its items may be accessed via index and via name. It shall be guaranteed for the uniqueness of the names of the items within the database.

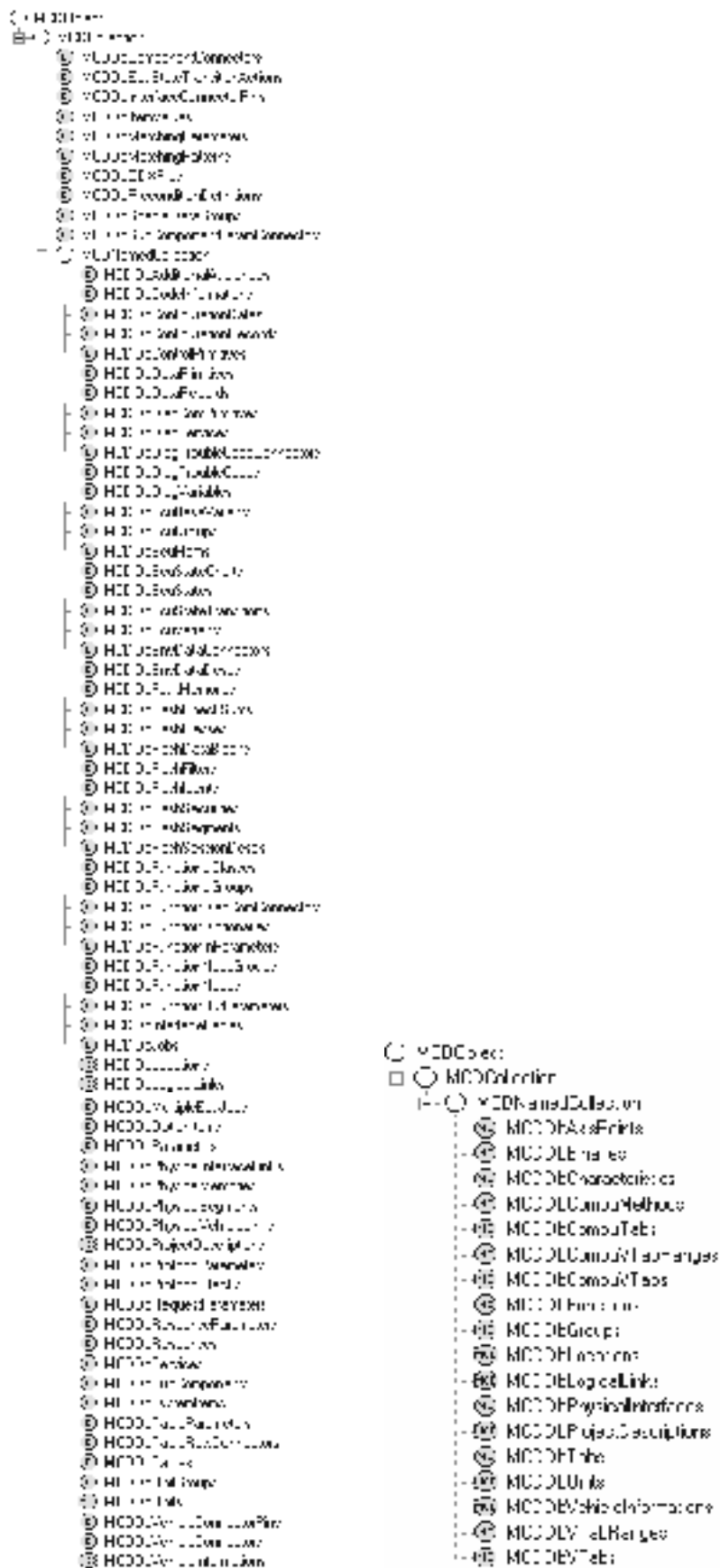


**Figure 12 — Database collections in D and MC**

## 7.9.4 Handling of collection of ASCIISTRING

For a better handling of the ASAM data type `A_ASCIISTRING` a class was introduced to the object model: `MCDDatatypeAsciiString`. It maps the `A_ASCIISTRING` to a more model compliant type in a one to one relation.

To allow collections of MCD data types a class `MCDDatatypeCollection` is defined. It provides the core functionality for data type collections: `getCount` and `removeAll`. The collection of objects of the type `MCDDatatypeAsciiString` is inherited from this.

## 7.10 EventHandler

### 7.10.1 Registering/deregistering of the EventHandlers

The MCD-server API provides a set of specialized event handlers. Each of these event handlers offers methods to send a set of closely related events. However, the general procedure of registering and deregistering an event handler is the same for all of these event handlers.

The current state of an object will NOT be delivered to a registered event handler by means of a synthetic event. Instead, the client registering the event handler is required to actively poll the current state by means of the corresponding methods.

The event which results from a state change or which results from an action, e.g. adding or removing elements from a collection of shared objects, is sent immediately after the new state has been reached successfully or after the corresponding action has been executed successfully.

The following methods shall be used with extreme care in the body of an event handler as their usage can result in deadlocks or cascades of events at runtime:

- `setEventHandler();`
- `releaseEventHandler();`
- execution of methods resulting in new events, especially if these events are sent to the same event handler;
- all methods, which change the data of, shared objects.

Besides above exception, it is always allowed to register, deregister, and configure event handlers at any time, provided that the cooperation level of the corresponding (shared) object is at least `READ_ONLY`. The result of these actions becomes active with the next event to be sent to the corresponding event handler.

NOTE 1 Above can result in an incomplete set of events being received at an event handler.

Because of this kind of registering several `EventHandler` may be connected per object at the same time. These are identified by means of the `Id` that is assigned at registration.

NOTE 2 These event handler IDs are server-wide unique at server runtime.

If the Server cannot register another `EventHandler` because of internal restrictions, e.g. resources or exceeding `MAX_A_UINT32` (232) `EventHandlers`, a `MCDProgramViolation` will be thrown. It contains the error `eSYSTEM_RESSOURCE_OVERLOAD`.

However, it is recommended to register at least a `MCDSystemEventHandler` at the client's `MCDSystem` object. Otherwise, the client will not be informed about system events or unchecked system errors. Registering an event handler at any other object is optional.

NOTE 3 No events will be forwarded to a higher level. The only exception to this rule is that DiagComPrimitive events are forwarded to the MCDDiagComPrimitives collection the corresponding DiagComPrimitive is contained in. Analogously events from FlashSessionDesc are forwarded to FlashSessionDescs collection.

For a client implementing an event handler is optional (scripted client). For a MCD-server, the implementation of event handling is mandatory.

State changes (attribute changes) of shared objects, e.g. LogicalLinks, DiagComPrimitives, and Collectors should only be delivered to those clients that have attached a EventListener at the corresponding object (e.g., System, LogicalLink, Collector, or DiagComPrimitive).

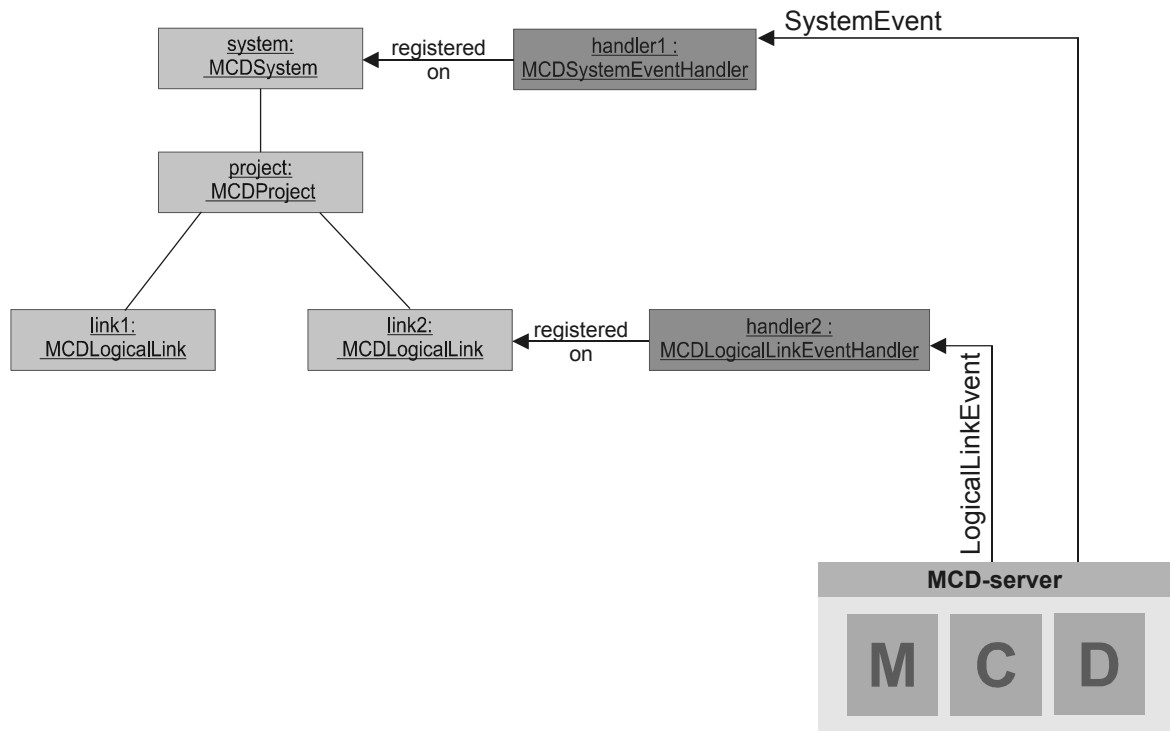


Figure 13 — Using of EventHandler

The `releaseEventHandler` method can be called in any state without throwing an exception with respect to the state – provided that the cooperation level of the corresponding (shared) object is at least `READ_ONLY` (Implementation hint to the client programmer: The `releaseEventHandler` method should be called in the same states where the corresponding `setEventHandler` had been called).

To conveniently be able to listen to all events of any DiagComPrimitive in the collection of DiagComPrimitives at a LogicalLink, it is sufficient to set an event handler at the corresponding collection. The same applies to FlashSessionDescs and MCDConfigurationRecords. To provide additional convenience, EventHandlers registered at an object are automatically deregistered by the MCD-server in case this object is removed.

EventHandlers are removed automatically in case an observed object is removed.

## 7.10.2 Methods of the EventHandlers

The MCDGlobalEventHandler could not be set directly. Only the derived classes (MCDCollectorEventHandler and MCDWriteReadRecorderEventHandler) are set at MCDCollector and MCDWriteReadRecorder. DiagComPrimitives EventHandler can be set at the collection of DiagComPrimitives.

**Table 2 — Methods of the MCDSystemEventHandler**

| Stereotype | Event                                | Semantic                                                                                                                                                                                        |
|------------|--------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <<D>>      | onBaseVariantIdentified              | Invoked by the MCD-server when a Base Variant within the current project is identified.                                                                                                         |
| <<D>>      | onBaseVariantSelected                | Invoked by the MCD-server when a Base Variant within the current project is selected.                                                                                                           |
| <<D>>      | onInterfaceError                     | Invoked by the D-server whenever an interface error event is received by the D-server, i.e. the PDU-API sends one of the events PDU_ERR_EVT_VCI_HARDWARE_FAULT or PDU_ERR_EVT_LOST_COMM_TO_VCI. |
| <<D>>      | onMonitoringLinksModified            | Invoked by the MCD-server when a client modifies the collection MCDMonitoringLinks.                                                                                                             |
| <<D>>      | onSystemDbConfigurationDatasModified | Invoked by the MCD-server when a client modifies the collection MCDDbConfigurationDatas.                                                                                                        |
| <<D>>      | onSystemDbEcuMemsModified            | Invoked by the MCD-server when a client modifies the collection MCDDbEcuMems.                                                                                                                   |
| <<D>>      | onSystemInterfacesModified           | A new VCI module has become available, or an available VCI module is not available any more. The application shall call getCurrentInterfaces() to retrieve the currently available modules.     |
| <<D>>      | onSystemInterfaceStatusChanged       | The status of the given VCI module has changed.                                                                                                                                                 |
| <<M,C,D>>  | onSystemClientConnected              | Invoked by the MCD-server whenever a client has been connected to the MCDSystem.                                                                                                                |
| <<M,C,D>>  | onSystemClientDisconnected           | Invoked by the MCD-server whenever a client has been disconnected to the MCDSystem.                                                                                                             |
| <<M,C,D>>  | onSystemClientNameChanged            | Invoked by the MCD-server whenever a client named has been modified.                                                                                                                            |
| <<M,C,D>>  | onSystemDbProjectConfiguration       | Invoked by the MCD-server when the state of the MCDSystem is changed to eDBPROJECT_CONFIGURATION.                                                                                               |
| <<M,C,D>>  | onSystemError                        | Invoked by the MCD-server when an error in the System occurs. This is not a special error for a Collector, Logical Link or DiagComPrimitive.                                                    |
| <<M,C,D>>  | onSystemInitialized                  | Invoked by the MCD-server when the state of the MCDSystem is changed to eINITIALIZED.                                                                                                           |
| <<M,C,D>>  | onSystemLocked                       | Invoked by the MCD-server when the MCDSystem object is locked.                                                                                                                                  |
| <<M,C,D>>  | onSystemLogicalLinksModified         | Invoked by the MCD-server when a client modifies the collection MCDLogicalLinks.                                                                                                                |
| <<M,C,D>>  | onSystemLogicallyConnected           | Invoked by the MCD-server when the state of the MCDSystem is changed to eLOGICALLY_CONNECTED.                                                                                                   |
| <<M,C,D>>  | onSystemProjectSelected              | Invoked by the MCD-server when the state of the MCDSystem is changed to ePROJECT_SELECTED.                                                                                                      |

Table 2 (continued)

| Stereotype | Event                              | Semantic                                                                                                                                                         |
|------------|------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <<M,C,D>>  | onSystemUnlocked                   | Invoked by the MCD-server when the MCDSystem object is unlocked.                                                                                                 |
| <<M,C,D>>  | onSystemVehicleInformationSelected | Invoked by the MCD-server when the state of the MCDSystem is changed to eVIT_SELECTED.                                                                           |
| <<M,C>>    | onSystemRecordersModified          | Invoked by the MCD-server when a client modifies the collection MCDRecorders.                                                                                    |
| <<M>>      | onSystemWatchersModified           | Invoked by the MCD-server when a client modifies the collection MCDWatchers, e.g. adds a MCDWatcher to the collection or removes MCDWatcher from the collection. |

Table 3 — Methods of the MCDConfigurationRecordEventHandler

| Stereotype | Event                         | Semantic                                                                      |
|------------|-------------------------------|-------------------------------------------------------------------------------|
| <<D>>      | onConfigurationRecordLoaded   | Invoked by the MCD-server when a configuration record is loaded.              |
| <<D>>      | onConfigurationRecordLocked   | Invoked by the MCD-server when the MCDConfigurationRecord object is locked.   |
| <<D>>      | onConfigurationRecordUnlocked | Invoked by the MCD-server when the MCDConfigurationRecord object is unlocked. |

Table 4 — Methods of the MCDWriteReadRecorderEventHandler

| Stereotype | Event                                  | Semantic                                                                                                                                                                                                                            |
|------------|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <<M>>      | onGlobalObjectActivated                | Invoked by the MCD-server if the respective state was entered.                                                                                                                                                                      |
| <<M>>      | onGlobalObjectConfigured               | Invoked by the MCD-server if the respective state was entered.                                                                                                                                                                      |
| <<M,D>>    | onGlobalObjectCreated                  | Invoked by the MCD-server if the respective state was entered.                                                                                                                                                                      |
| <<M>>      | onGlobalObjectLocked                   | Invoked by the MCD-server if the respective state was entered.                                                                                                                                                                      |
| <<M,D>>    | onGlobalObjectStarted                  | Invoked by the MCD-server if the respective state was entered.                                                                                                                                                                      |
| <<M>>      | onGlobalObjectUnlocked                 | Invoked by the MCD-server if the respective state was entered.                                                                                                                                                                      |
| <<M>>      | onWriteReadRecorderError               | Invoked by the MCD-server if an error occurred.                                                                                                                                                                                     |
| <<M>>      | onWriteReadRecorderModified            | Invoked by the MCD-server when a client modifies the collection MCDWriteReadRecorderCollectors or any contained MCDCollectedObjects collection, e.g. adds a collector to the collection or removes a collector from the collection. |
| <<M>>      | onWriteReadRecorderPaused              | Invoked by the MCD-server if the respective state was entered.                                                                                                                                                                      |
| <<M>>      | onWriteReadRecorderStopped Acquisition | This event transports the information about a MCDWriteReadRecorder if a recorder collector has stopped acquiring data due to an error state in the associated LogicalLink.                                                          |

**Table 5 — Methods of the MCDFlashSessionDescEventHandler**

| Stereotype | Event                      | Semantic                                                                   |
|------------|----------------------------|----------------------------------------------------------------------------|
| <<D>>      | onFlashSegmentsCleared     | Invoked by the MCD-server when a flash segment is cleared.                 |
| <<D>>      | onFlashSegmentsLoaded      | Invoked by the MCD-server when a flash segment is loaded.                  |
| <<D>>      | onFlashSessionDescLocked   | Invoked by the MCD-server when the MCDFlashSessionDesc object is locked.   |
| <<D>>      | onFlashSessionDescUnlocked | Invoked by the MCD-server when the MCDFlashSessionDesc object is unlocked. |

**Table 6 — Methods of the MCDCollectorEventHandler**

| Stereotype | Event                               | Semantic                                                                             |
|------------|-------------------------------------|--------------------------------------------------------------------------------------|
| <<D>>      | onCollectorMonitoringFramesReady    | Invoked by the MCD-server when new monitoring frames are available at a collector.   |
| <<M>>      | onCollectorCollectedObjectsModified | Invoked by the MCD-server when a client modifies the collection MCDCollectedObjects. |
| <<M,D>>    | onCollectorError                    | Invoked by the MCD-server when an error occurs while using the Collector.            |
| <<M>>      | onCollectorResultReady              | Invoked by the MCD-server when a result of a Collector is ready.                     |
| <<M>>      | onGlobalObjectActivated             | Invoked by the MCD-server if the respective state was entered.                       |
| <<M>>      | onGlobalObjectConfigured            | Invoked by the MCD-server if the respective state was entered.                       |
| <<M,D>>    | onGlobalObjectCreated               | Invoked by the MCD-server if the respective state was entered.                       |
| <<M>>      | onGlobalObjectLocked                | Invoked by the MCD-server if the respective state was entered.                       |
| <<M,D>>    | onGlobalObjectStarted               | Invoked by the MCD-server if the respective state was entered.                       |
| <<M>>      | onGlobalObjectUnlocked              | Invoked by the MCD-server if the respective state was entered.                       |

**Table 7 — Methods of the MCDGlobalEventHandler**

| Stereotype | Event                    | Semantic                                                       |
|------------|--------------------------|----------------------------------------------------------------|
| <<M>>      | onGlobalObjectActivated  | Invoked by the MCD-server if the respective state was entered. |
| <<M>>      | onGlobalObjectConfigured | Invoked by the MCD-server if the respective state was entered. |
| <<M,D>>    | onGlobalObjectCreated    | Invoked by the MCD-server if the respective state was entered. |
| <<M>>      | onGlobalObjectLocked     | Invoked by the MCD-server if the respective state was entered. |
| <<M,D>>    | onGlobalObjectStarted    | Invoked by the MCD-server if the respective state was entered. |
| <<M>>      | onGlobalObjectUnlocked   | Invoked by the MCD-server if the respective state was entered. |

**Table 8 — Methods of the MCDDiagComPrimitiveEventHandler**

| Stereotype | Event                            | Semantic                                                                                                                                                                                                                             |
|------------|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <<D>>      | onPrimitiveBufferOverflow        | Invoked by the MCD-server when the result buffer of a DiagComPrimitive has been overflowed.                                                                                                                                          |
| <<D>>      | onPrimitiveHasIntermediateResult | Invoked by the MCD-server when the DiagComPrimitive (Job) has been delivered an intermediate result.                                                                                                                                 |
| <<D>>      | onPrimitiveHasResult             | Invoked by the MCD-server when the DiagComPrimitive has been delivered an result. This event is sent if a DiagComPrimitive or a Job have finished asynchronous execution completely or if a repetition cycle has finished.           |
| <<D>>      | onPrimitiveIdle                  | This event is sent if the state eIDLE is reached in the state diagram of an MCDDiagComPrimitive.                                                                                                                                     |
| <<D>>      | onPrimitiveJobInfo               | Invoked by the D-server when a Java job has sent a job info by calling MCDJobApi::setJobInfo(...).                                                                                                                                   |
| <<D>>      | onPrimitiveLocked                | Invoked by the MCD-server when the MCDDiagComPrimitive object is locked.                                                                                                                                                             |
| <<D>>      | onPrimitiveNotExecutable         | This event is sent if the state eNOT_EXECUTABLE is reached in the state diagram of an MCDDiagComPrimitive. This event is only send if the MCDDiagComPrimitive become invalid if a variant identification and selection is processed. |
| <<D>>      | onPrimitivePending               | This event is sent if the state ePENDING is reached in the state diagram of an MCDDiagComPrimitive.                                                                                                                                  |
| <<D>>      | onPrimitiveProgressInfo          | Invoked by the MCD-server when the DiagComPrimitive has been sent an progress info.                                                                                                                                                  |
| <<D>>      | onPrimitiveRepeating             | This event is sent if the state eREPEATING is reached in the state diagram of an MCDDiagComPrimitive.                                                                                                                                |
| <<D>>      | onPrimitiveUnlocked              | Invoked by the MCD-server when the MCDDiagComPrimitive object is unlocked.                                                                                                                                                           |
| <<D>>      | onStartMessageIndication         | Called only if PDU API fires the corresponding events (The corresponding ComParams shall be enabled)                                                                                                                                 |
| <<D>>      | onTransmitIndication             | Called only if PDU API fires the corresponding events (The corresponding ComParams shall be enabled)                                                                                                                                 |

**Table 9 — Methods of the MCDWatcherEventHandler**

| Stereotype | Event                | Semantic                                                                                                                                                                                                  |
|------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <<M>>      | onWatcherActivated   | This event transports the information about an MCDWatcher if the respective state was entered.                                                                                                            |
| <<M>>      | onWatcherCreated     | Invoked by the MCD-server if the respective state was entered.                                                                                                                                            |
| <<M>>      | onWatcherDeactivated | This event transports the information about an MCDWatcher if the respective state was entered.                                                                                                            |
| <<M>>      | onWatcherError       | This method is called if the watcher has an asynchronous error.<br>Following errors could occur:<br>the watcher could not be activated<br>eRT_WATCHER_ACTIVATION_FAILED<br>(MCDProgramViolationException) |

**Table 9** (continued)

| Stereotype | Event             | Semantic                                                                                       |
|------------|-------------------|------------------------------------------------------------------------------------------------|
| <<M>>      | onWatcherFired    | This event transports the information about an MCDWatcher if the respective state was entered. |
| <<M>>      | onWatcherLocked   | This method is called if the watcher reaches the locked state.                                 |
| <<M>>      | onWatcherUnlocked | This method is called if the watcher reaches the unlocked state.                               |

**Table 10 — Methods of the MCDLogicalLinkEventHandler**

| Stereotype | Event                              | Semantic                                                                                                                                                                            |
|------------|------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <<C>>      | onLinkCharacteristicsModified      | Invoked by the MCD-server when a client modifies the collection MCDCharacteristics.                                                                                                 |
| <<D>>      | onLinkActivityIdle                 | Invoked by the MCD-server when the instruction queue of a LogicalLink has performed a state transition to eACTIVITY_IDLE either through creation or from eACTIVITY_RUNNING.         |
| <<D>>      | onLinkActivityRunning              | Invoked by the MCD-server when the instruction queue of a LogicalLink has performed a state transition to eACTIVITY_RUNNING either from eACTIVITY_IDLE or from eACTIVITY_SUSPENDED. |
| <<D>>      | onLinkActivitySuspended            | Invoked by the MCD-server when the instruction queue of a LogicalLink has performed a state transition to eACTIVITY_SUSPENDED from eACTIVITY_RUNNING.                               |
| <<D>>      | onLinkConfigurationRecordsModified | Invoked by the MCD-server client modifies the collection MCDConfigurationRecords.                                                                                                   |
| <<D>>      | onLinkCreated                      | Invoked by the MCD-server when the state of the MCDLogicalLink is changed to eCREATED.                                                                                              |
| <<D>>      | onLinkDefinableDynIdListChanged    | Invoked by the MCD-server when a DynID list has changed.                                                                                                                            |
| <<D>>      | onLinkDiagComPrimitivesModified    | Invoked by the MCD-server when a client modifies the collection MCDDiagComPrimitives.                                                                                               |
| <<D>>      | onLinkFlashSessionDescsModified    | Invoked by the MCD-server client modifies the collection MCDFlashSessionDescs.                                                                                                      |
| <<D>>      | onLinkQueueCleared                 | Invoked by the MCD-server when the instruction queue of a LogicalLink has been cleared by invoking the corresponding method.                                                        |
| <<D>>      | onLinkVariantIdentified            | Invoked by the D-server in case a variant has been identified for a logical link with location type eFUNCTIONAL_GROUP.                                                              |
| <<D>>      | onLinkVariantSelected              | Invoked by the MCD-server when the Logical Link has been identified and selected for a logical link with location type eFUNCTIONAL_GROUP.                                           |
| <<D>>      | onLinkVariantSet                   | Invoked by the MCD-server when the Variant of a Logical Link is set.                                                                                                                |
| <<M, C>>   | onLinkTargetOff                    | This event is invoked by the MCD-server when the state of the connected target changes to off.                                                                                      |
| <<M, C>>   | onLinkTargetOn                     | This event is invoked by the MCD-server when the state of the connected target changes to on.                                                                                       |

Table 10 (continued)

| Stereotype | Event                    | Semantic                                                                                                |
|------------|--------------------------|---------------------------------------------------------------------------------------------------------|
| <<M, C>>   | onLinkTargetUnknown      | This event is invoked by the MCD-server when the state of the connected target could not be determined. |
| <<M,C,D>>  | onLinkCommunication      | Invoked by the MCD-server when the state of the MCDLogicalLink is changed to eCOMMUNICATION.            |
| <<M,C,D>>  | onLinkError              | Invoked by the MCD-server when an error on a Logical Link occurs.                                       |
| <<M,C,D>>  | onLinkLocked             | Invoked by the MCD-server when a Logical link has been locked.                                          |
| <<M,C,D>>  | onLinkOffline            | Invoked by the MCD-server when the state of the MCDLogicalLink is changed to eOFFLINE.                  |
| <<M,C,D>>  | onLinkOnline             | Invoked by the MCD-server when the state of the MCDLogicalLink is changed to eONLINE.                   |
| <<M,C,D>>  | onLinkUnlocked           | Invoked by the MCD-server when a Logical Link has been unlocked.                                        |
| <<M>>      | onLinkCollectorsModified | Invoked by the MCD-server when a client modifies the collection MCDCollectors.                          |

### 7.10.3 Eventfilter

A Client has now the possibility to define which events should be reported, that means, if a client creates an event handler, the client is also able to select which events to be delivered by the server.

The method `configureEventHandler(eventHandlerId : A_UINT32, filterSettings : A_UINT32) : void` is used at all objects, where `setEventHandler` and `releaseEventHandler` is allowed. The method is only valid if the accompanying Event Handler exists, otherwise the error of class Run Time / Program Violation Errors `eRT_NO_ACTIVE_OBJECT` is used. If an Event Handler exists `configureEventHandler` can be called at any time.

The filter settings are defined by enumerations, which define the bit position inside the filter setting. These enumerations are eventhandler specific.

Table 11 — Enumeration MCDSystemEventTypes

| Type  | Name                                              | InitValue  | Documentation                                                                                                                                                                                                       |
|-------|---------------------------------------------------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <<D>> | eFSS_ONBASEVARIANT<br>IDENTIFIED                  | 0x00080000 | This bit can be used to temporarily deactivate the event <code>onBaseVariantIdentified</code> . For this purpose, the <code>eFSS_ONBASEVARIANTIDENTIFIED</code> bit shall be set to zero.                           |
| <<D>> | eFSS_ONBASEVARIANT<br>SELECTED                    | 0x00100000 | This bit can be used to temporarily deactivate the event <code>onBaseVariantSelected</code> . For this purpose, the <code>eFSS_ONBASEVARIANTSELECTED</code> bit shall be set to zero.                               |
| <<D>> | eFSS_ONMONITORINGLINKS<br>MODIFIED                | 0x00200000 | This bit can be used to temporarily deactivate the event <code>onBaseVariantSelected</code> . For this purpose, the <code>eFSS_ONMONITORINGLINKSMODIFIED</code> bit shall be set to zero.                           |
| <<D>> | eFSS_ONSYSTEMDB<br>CONFIGURATIONDATAS<br>MODIFIED | 0x00040000 | This bit can be used to temporarily deactivate the event <code>onSystemDbConfigurationDatasModified</code> . For this purpose, the <code>eFSS_ONSYSTEMDBCONFIGURATIONDATASMODIFIED</code> bit shall be set to zero. |

Table 11 (continued)

| Type      | Name                                    | InitValue  | Documentation                                                                                                                                                                |
|-----------|-----------------------------------------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <<D>>     | eFSS_ONSYSTEMDBECUMEMS<br>MODIFIED      | 0x00020000 | This bit can be used to temporarily deactivate the event onSystemDbEcuMemsModified. For this purpose, the eFSS_ONSYSTEMDBECUMEMSMODIFIED bit shall be set to zero.           |
| <<D>>     | eFSS_ONSYSTEMINTERFACES<br>MODIFIED     | 0x00008000 | This bit can be used to temporarily deactivate the event onSystemInterfacesChanged. For this purpose, the eFSS_ONSYSTEMINTERFACESMODIFIED bit shall be set to zero.          |
| <<D>>     | eFSS_ONSYSTEMINTERFACE<br>STATUSCHANGED | 0x00010000 | This bit can be used to temporarily deactivate the event onSystemInterfaceStatusChanged. For this purpose, the eFSS_ONSYSTEMINTERFACESTATUSCHANGED bit shall be set to zero. |
| <<M,C,D>> | eFSS_ONSYSTEMCLIENT<br>CONNECTED        | 0x00000002 | This bit can be used to temporarily deactivate the event onSystemClientConnected. For this purpose, the eFSS_ONSYSTEMCLIENTCONNECTED bit shall be set to zero.               |
| <<M,C,D>> | eFSS_ONSYSTEMCLIENT<br>DISCONNECTED     | 0x00000004 | This bit can be used to temporarily deactivate the event onSystemClientDisconnected. For this purpose, the eFSS_ONSYSTEMCLIENTDISCONNECTED bit shall be set to zero.         |
| <<M,C,D>> | eFSS_ONSYSTEMCLIENTNAME<br>CHANGED      | 0x00000008 | This bit can be used to temporarily deactivate the event onSystemClientNameChanged. For this purpose, the eFSS_ONSYSTEMCLIENTNAMECHANGED bit shall be set to zero.           |
| <<M,C,D>> | eFSS_ONSYSTEMDBPROJECT<br>CONFIGURATION | 0x00000010 | This bit can be used to temporarily deactivate the event onSystemDbProjectConfiguration. For this purpose, the eFSS_ONSYSTEMDBPROJECTCONFIGURATION bit shall be set to zero. |
| <<M,C,D>> | eFSS_ONSYSTEMERROR                      | 0x00000020 | This bit can be used to temporarily deactivate the event onSystemError. For this purpose, the eFSS_ONSYSTEMERROR bit shall be set to zero.                                   |
| <<M,C,D>> | eFSS_ONSYSTEMINITIALIZED                | 0x00000040 | This bit can be used to temporarily deactivate the event onSystemInitialized. For this purpose, the eFSS_ONSYSTEMINITIALIZED bit shall be set to zero.                       |
| <<M,C,D>> | eFSS_ONSYSTEMLOCKED                     | 0x00000800 | This bit can be used to temporarily deactivate the event onSystemLocked. For this purpose, the eFSS_ONSYSTEMLOCKED bit shall be set to zero.                                 |
| <<M,C,D>> | eFSS_ONSYSTEMLOGICAL<br>LINKSMODIFIED   | 0x00000080 | This bit can be used to temporarily deactivate the event onSystemLogicalLinksModified. For this purpose, the eFSS_ONSYSTEMLOGICALLINKSMODIFIED bit shall be set to zero.     |
| <<M,C,D>> | eFSS_ONSYSTEMLOGICALLY<br>CONNECTED     | 0x00000100 | This bit can be used to temporarily deactivate the event onSystemLogicallyConnected. For this purpose, the eFSS_ONSYSTEMLOGICALLYCONNECTED bit shall be set to zero.         |
| <<M,C,D>> | eFSS_ONSYSTEMPROJECT<br>SELECTED        | 0x00000200 | This bit can be used to temporarily deactivate the event onSystemProjectSelected. For this purpose, the eFSS_ONSYSTEMPROJECTSELECTED bit shall be set to zero.               |

Table 11 (continued)

| Type      | Name                                        | InitValue  | Documentation                                                                                                                                                                        |
|-----------|---------------------------------------------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <<M,C,D>> | eFSS_ONSYSTEMUNLOCKED                       | 0x00001000 | This bit can be used to temporarily deactivate the event onSystemUnlocked. For this purpose, the eFSS_ONSYSTEMUNLOCKED bit shall be set to zero.                                     |
| <<M,C,D>> | eFSS_ONSYSTEMVEHICLE<br>INFORMATIONSELECTED | 0x00000400 | This bit can be used to temporarily deactivate the event onSystemVehicleInformationSelected. For this purpose, the eFSS_ONSYSTEMVEHICLEINFORMATIONSELECTED bit shall be set to zero. |
| <<M,C>>   | eFSS_ONSYSTEMRECORDERS<br>MODIFIED          | 0x00002000 | This bit can be used to temporarily deactivate the event onSystemRecordersModified. For this purpose, the eFSS_ONSYSTEMRECORDERSMODIFIED bit shall be set to zero.                   |
| <<M>>     | eFSS_ONSYSTEMWATCHERS<br>MODIFIED           | 0x00004000 | This bit can be used to temporarily deactivate the event onSystemWatchersModified. For this purpose, the eFSS_ONSYSTEMWATCHERSMODIFIED bit shall be set to zero.                     |

Table 12 — Enumeration MCDConfigurationRecordEventTypes

| Type  | Name                                    | InitValue  | Documentation                                                                                                                                                               |
|-------|-----------------------------------------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <<D>> | eFSCR_ONCONFIGURATION<br>RECORDLOADED   | 0x00000004 | This bit can be used to temporarily deactivate the event onConfigurationRecordLoaded. For this purpose, the eFSCR_ONCONFIGURATIONRECORDLOADED bit shall be set to zero.     |
| <<D>> | eFSCR_ONCONFIGURATION<br>RECORDLOCKED   | 0x00000001 | This bit can be used to temporarily deactivate the event onConfigurationRecordLocked. For this purpose, the eFSCR_ONCONFIGURATIONRECORDLOCKED bit shall be set to zero.     |
| <<D>> | eFSCR_ONCONFIGURATION<br>RECORDUNLOCKED | 0x00000002 | This bit can be used to temporarily deactivate the event onConfigurationRecordUnlocked. For this purpose, the eFSCR_ONCONFIGURATIONRECORDUNLOCKED bit shall be set to zero. |

Table 13 — Enumeration MCDWriteReadRecorderEventTypes

| Type  | Name                                | InitValue  | Documentation                                                                                                                                                 |
|-------|-------------------------------------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <<M>> | eFSWRR_ONGLOBALOBJECT<br>ACTIVATED  | 0x00000001 | This bit can be used to temporarily deactivate the event onCollectorActivated. For this purpose, the eFSG_ONGLOBALOBJECTACTIVATED bit shall be set to zero.   |
| <<M>> | eFSWRR_ONGLOBALOBJECT<br>CONFIGURED | 0x00000002 | This bit can be used to temporarily deactivate the event onCollectorConfigured. For this purpose, the eFSG_ONGLOBALOBJECTCONFIGURED bit shall be set to zero. |
| <<M>> | eFSWRR_ONGLOBALOBJECT<br>CREATED    | 0x00000004 | This bit can be used to temporarily deactivate the event onCollectorCreated. For this purpose, the eFSG_ONGLOBALOBJECTCREATED bit shall be set to zero.       |

Table 13 (continued)

| Type  | Name                                                 | InitValue  | Documentation                                                                                                                                                                                |
|-------|------------------------------------------------------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <<M>> | eFSWRR_ONGLOBALOBJECT<br>LOCKED                      | 0x00000020 | This bit can be used to temporarily deactivate the event onCollectorLocked. For this purpose, the eFSG_ONGLOBALOBJECTLOCKED bit shall be set to zero.                                        |
| <<M>> | eFSWRR_ONGLOBALOBJECT<br>STARTED                     | 0x00000008 | This bit can be used to temporarily deactivate the event onCollectorStarted. For this purpose, the eFSG_ONGLOBALOBJECTSTARTED bit shall be set to zero.                                      |
| <<M>> | eFSWRR_ONGLOBALOBJECT<br>UNLOCKED                    | 0x00000010 | This bit can be used to temporarily deactivate the event onCollectorUnlocked. For this purpose, the eFSG_ONGLOBALOBJECTUNLOCKED bit shall be set to zero.                                    |
| <<M>> | eFSWRR_ONWRITEREAD<br>RECORDERERROR                  | 0x00000080 | This bit can be used to temporarily deactivate the event onWriteReadRecorderError. For this purpose, the eFSWRR_ONWRITEREADRECORDERERROR bit shall be set to zero.                           |
| <<M>> | eFSWRR_ONWRITEREAD<br>RECORDERMODIFIED               | 0x00000200 | This bit can be used to temporarily deactivate the event onWriteReadRecorderModified. For this purpose, the eFSWRR_ONWRITEREADRECORDERMODIFIED bit shall be set to zero.                     |
| <<M>> | eFSWRR_ONWRITEREAD<br>RECORDERPAUSED                 | 0x00000040 | This bit can be used to temporarily deactivate the event onWriteReadRecorderPaused. For this purpose, the eFSWRR_ONWRITEREADRECORDERPAUSED bit shall be set to zero.                         |
| <<M>> | eFSWRR_ONWRITEREAD<br>RECORDERSTOPPED<br>ACQUISITION | 0x00000100 | This bit can be used to temporarily deactivate the event onWriteReadRecorderStoppedAcquisition. For this purpose, the eFSWRR_ONWRITEREADRECORDERSTOPPEDACQUISITION bit shall be set to zero. |

Table 14 — Enumeration MCDFlashSessionDescEventTypes

| Type  | Name                                 | InitValue  | Documentation                                                                                                                                                      |
|-------|--------------------------------------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <<D>> | eFSSD_ONFLASHSEGMENTS<br>CLEARED     | 0X00000001 | This bit can be used to temporarily deactivate the event onFlashSegmentsCleared. For this purpose, the eFSSD_ONFLASHSEGMENTSCLEARED bit shall be set to zero.      |
| <<D>> | eFSSD_ONFLASHSEGMENTS<br>LOADED      | 0X00000002 | This bit can be used to temporarily deactivate the event onFlashSegmentsLoaded. For this purpose, the eFSSD_ONFLASHSEGMENTSLOADED bit shall be set to zero.        |
| <<D>> | eFSSD_ONFLASHSESSIONDESC<br>LOCKED   | 0X00000004 | This bit can be used to temporarily deactivate the event onFlashSegmentsLocked. For this purpose, the eFSSD_ONFLASHSESSIONDESCLOCKED bit shall be set to zero.     |
| <<D>> | eFSSD_ONFLASHSESSIONDESC<br>UNLOCKED | 0X00000008 | This bit can be used to temporarily deactivate the event onFlashSegmentsUnlocked. For this purpose, the eFSSD_ONFLASHSESSIONDESCUNLOCKED bit shall be set to zero. |