INTERNATIONAL STANDARD

ISO 13209-4

First edition
2021-05

Road vehicles — Open Test sequence eXchange format (OTX) —

Part 4: Expanded extensions interface definition

Véhicules routiers — Format public d'échange de séquence-tests (OTX) —

Partie 4: Définition de l'interface des extensions étendues

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Reference number
ISO 13209-4:2021(E)

© ISO 2021

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021



COPYRIGHT PROTECTED DOCUMENT

© ISO 2021

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

© ISO 2021 – All rights reserved

Contents

Page

Foreword ..... **xii**
Introduction ..... **xiii**
1 Scope **1**
2 Normative references **1**
3 Terms, definitions and abbreviated terms **1**
3.1 Terms and definitions 1
3.2 Abbreviated terms 2
4 Requirements **2**
4.1 Basic principles for requirements definition 2
4.2 Requirement listing 2
4.2.1 Assertion requirements 2
4.2.2 BlackBox requirements 2
4.2.3 BusMonitoring requirements 3
4.2.4 ComInterface requirements 3
4.2.5 CommonDialogs requirements 3
4.2.6 DataType requirements 4
4.2.7 DiagComPlus requirements 4
4.2.8 DiagConfiguration requirements 5
4.2.9 DiagDataBrowsingPlus requirements 5
4.2.10 ECUConfiguration requirements 7
4.2.11 EventPlus requirements 7
4.2.12 ExternalServiceProvider requirements 7
4.2.13 File requirements 8
4.2.14 FlashPlus requirements 8
4.2.15 Persistence requirements 8
4.2.16 SQL requirements 9
4.2.17 StateMachineProcedure requirements 9
4.2.18 StateVariable requirements 9
4.2.19 TestResultHandling requirements 10
4.2.20 Util requirements 10
4.2.21 VehicleInfo requirements 11
4.2.22 XML requirements 12
4.2.23 ZipHandling requirements 13
5 Extension overview **13**
5.1 General 13
5.2 Dependencies 14
5.3 OTX packaging (PTX) 18
6 OTX Assertion extension **18**
6.1 General 18
6.2 Exceptions 18
6.2.1 Exceptions overview 18
6.2.2 Exceptions syntax 18
6.2.3 Exceptions semantics 19
6.3 Actions 19
6.3.1 Actions overview 19
6.3.2 Actions syntax 19
6.3.3 Actions general semantics 20
7 OTX BlackBox extension **21**
7.1 General 21
7.2 Datatypes 21
7.2.1 Datatypes overview 21
7.2.2 Datatypes syntax 21

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

7.2.3 Datatypes semantics 22
7.3 Variable access 22
7.3.1 Variable-access overview 22
7.3.2 Variable-access syntax 22
7.3.3 Variable-access semantics 23
7.4 Terms 23
7.4.1 Terms overview 23
7.4.2 Terms syntax 23
7.4.3 Terms general semantics 24

8 OTX BusMonitoring extension 25
8.1 General 25
8.2 Datatypes 25
8.2.1 Datatypes overview 25
8.2.2 Datatypes syntax 25
8.2.3 Datatypes semantics 26
8.3 Enumerations 27
8.3.1 Enumerations syntax 27
8.3.2 Enumerations semantics 27
8.4 Exceptions 28
8.4.1 Exceptions overview 28
8.4.2 Exceptions syntax 28
8.4.3 Exceptions semantics 28
8.5 Variable access 29
8.5.1 Variable-access overview 29
8.5.2 Variable-access syntax 29
8.5.3 Variable-access semantics 29
8.6 Actions 29
8.6.1 Actions overview 29
8.6.2 Actions syntax 29
8.6.3 Actions general semantics 30
8.7 Terms 32
8.7.1 Terms overview 32
8.7.2 Terms syntax 32
8.7.3 Terms general semantics 32

9 OTX ComInterface extension 38
9.1 General 38
9.2 Datatypes 38
9.2.1 Datatypes overview 38
9.2.2 Datatypes syntax 38
9.2.3 Datatypes semantics 39
9.3 Enumerations 39
9.3.1 Enumerations syntax 39
9.3.2 Enumerations semantics 40
9.4 Exceptions 40
9.4.1 Exceptions overview 40
9.4.2 Exceptions syntax 41
9.4.3 Exceptions semantics 41
9.5 Variable access 42
9.5.1 Variable-access overview 42
9.5.2 Variable-access syntax 42
9.5.3 Variable-access semantics 43
9.6 Actions 43
9.6.1 Actions overview 43
9.6.2 Actions syntax 43
9.6.3 Actions general semantics 44
9.7 Terms 47
9.7.1 Terms overview 47

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

9.7.2 Terms syntax 47
9.7.3 Semantics for ComInterface terms 48
9.7.4 Semantics for Enumeration terms 58
9.7.5 Terms general semantics 59

10 OTX CommonDialogs extension 59
10.1 General 59
10.2 Exceptions 60
10.2.1 Exceptions overview 60
10.2.2 Exceptions syntax 60
10.2.3 Exceptions semantics 60
10.3 Actions 60
10.3.1 Actions overview 60
10.3.2 Actions syntax 60
10.3.3 Actions general semantics 61

11 OTX DataType extension 64
11.1 General 64
11.2 Datatypes 64
11.2.1 Datatypes overview 64
11.2.2 Datatypes syntax 64
11.2.3 Datatypes semantics 65
11.3 Exceptions 67
11.3.1 Exceptions overview 67
11.3.2 Exceptions syntax 67
11.3.3 Exceptions semantics 68
11.4 Variable access 68
11.4.1 Variable-access overview 68
11.4.2 Variable-access syntax 68
11.4.3 Variable-access semantics 69
11.5 Declaration and arguments 69
11.5.1 Declaration and arguments syntax 69
11.5.2 Declaration and arguments semantics 70
11.6 Signatures 70
11.6.1 Signatures overview 70
11.6.2 Signatures syntax 70
11.6.3 Semantics for Enumerations 71
11.6.4 Semantics for Structures 72
11.7 Terms 73
11.7.1 Terms overview 73
11.7.2 Terms syntax 73
11.7.3 Semantics for EnumerationTerms 75
11.7.4 Semantics for ResourceLocationTerms 78
11.7.5 Semantics for StructureTerms 80

12 OTX DiagComPlus extension 81
12.1 General 81
12.2 Datatypes 82
12.2.1 Datatypes overview 82
12.2.2 Datatypes syntax 82
12.2.3 Datatypes semantics 82
12.3 Enumerations 83
12.3.1 Enumerations syntax 83
12.3.2 Enumerations semantics 83
12.4 Exceptions 84
12.4.1 Exceptions overview 84
12.4.2 Exceptions syntax 84
12.4.3 Exceptions semantics 84
12.5 Variable access 84
12.5.1 Variable-access overview 84

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

12.5.2 Variable-access syntax 85
12.5.3 Variable-access semantics 85
12.6 Actions 85
12.6.1 Actions overview 85
12.6.2 Actions syntax 85
12.6.3 Actions general semantics 86
12.7 Terms 88
12.7.1 Terms overview 88
12.7.2 Terms syntax 88
12.7.3 Terms general semantics 88

13 OTX DiagConfiguration extension 92
13.1 General 92
13.2 Exceptions 92
13.2.1 Exceptions overview 92
13.2.2 Exceptions syntax 92
13.2.3 Exceptions semantics 93
13.3 Actions 93
13.3.1 Actions overview 93
13.3.2 Actions syntax 93
13.3.3 Actions general semantics 94
13.4 Terms 95
13.4.1 Terms overview 95
13.4.2 Terms syntax 95
13.4.3 Terms general semantics 96

14 OTX DiagDataBrowsingPlus extension 98
14.1 General 98
14.2 Datatypes 98
14.2.1 Datatypes overview 98
14.2.2 Datatypes syntax 98
14.2.3 Datatypes semantics 100
14.3 Enumerations 103
14.3.1 Enumerations syntax 103
14.3.2 Enumerations semantics 104
14.4 Exceptions 107
14.4.1 Exceptions overview 107
14.4.2 Exceptions syntax 107
14.4.3 Exceptions semantics 107
14.5 Variable access 108
14.5.1 Variable-access overview 108
14.5.2 Variable-access syntax 108
14.5.3 Variable-access semantics 109
14.6 Terms 109
14.6.1 Terms overview 109
14.6.2 Semantics for DbComChannel terms 109
14.6.3 Semantics for DbDiagService terms 112
14.6.4 Semantics for DbDiagTroubleCode terms 118
14.6.5 Semantics for DbEcuVariant terms 123
14.6.6 Semantics for DbEnvDataDesc terms 123
14.6.7 Semantics for DbFaultMemory terms 126
14.6.8 Semantics for DbMatchingParameter terms 128
14.6.9 Semantics for DbObject terms 129
14.6.10 Semantics for DbParameter terms 133
14.6.11 Semantics for DbRequest terms 147
14.6.12 Semantics for DbResponse terms 151
14.6.13 Semantics for DbSpecialDataElement terms 155
14.6.14 Semantics for DbSpecialDataGroup terms 157
14.6.15 Semantics for DbSubComponent terms 161

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

14.6.16 Semantics for DbTable terms 167
14.6.17 Semantics for interval terms 172
14.6.18 Semantics for McdDataType terms 175
14.6.19 Semantics for McdParameter terms 176
14.6.20 Semantics for McdResponseType terms 178
14.6.21 Semantics for parameter terms 180
14.6.22 Semantics for TextTableElement terms 184

15 OTX EcuConfiguration extension 186
15.1 General 186
15.2 Datatypes 186
15.2.1 Datatypes overview 186
15.2.2 Datatypes syntax 187
15.2.3 Datatypes semantics 187
15.3 Exceptions 188
15.3.1 Exceptions overview 188
15.3.2 Exceptions syntax 188
15.3.3 Exceptions semantics 188
15.4 Variable access 188
15.4.1 Variable-access overview 188
15.4.2 Variable-access syntax 189
15.4.3 Variable-access semantics 189
15.5 Terms 189
15.5.1 Terms overview 189
15.5.2 Semantics for DbConfigurationData terms 190
15.5.3 Semantics for DbConfigurationRecord terms 191
15.5.4 Semantics for DbDataRecord terms 193
15.5.5 Semantics for DbItemValue terms 199
15.5.6 Semantics for DbOptionItem terms 203

16 OTX EventPlus extension 210
16.1 General 210
16.2 Exceptions 210
16.2.1 Exceptions overview 210
16.2.2 Exceptions syntax 210
16.2.3 Exceptions semantics 210
16.3 Terms 211
16.3.1 Terms overview 211
16.3.2 Terms syntax 211
16.3.3 Terms general semantics 211

17 OTX ExternalServiceProvider extension 212
17.1 General 212
17.2 Datatypes 213
17.2.1 Datatypes overview 213
17.2.2 Datatypes syntax 213
17.2.3 Datatypes semantics 214
17.3 Enumerations 215
17.3.1 Enumerations syntax 215
17.3.2 Enumerations semantics 215
17.4 Exceptions 215
17.4.1 Exceptions overview 215
17.4.2 Exceptions syntax 215
17.4.3 Exceptions semantics 216
17.5 Variable access 217
17.5.1 Variable-access overview 217
17.5.2 Variable-access syntax 217
17.5.3 Variable-access semantics 218
17.6 Declaration and arguments 218
17.6.1 Declaration and arguments syntax 218

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

17.6.2 Declaration and arguments semantics 219
17.7 Signatures 220
17.7.1 Signatures overview 220
17.7.2 Signatures syntax 220
17.7.3 Signatures general semantics 220
17.8 Actions 227
17.8.1 Actions overview 227
17.8.2 Actions syntax 227
17.8.3 Actions general semantics 229
17.9 Terms 234
17.9.1 Terms overview 234
17.9.2 Terms syntax 235
17.9.3 Semantics for Enumeration terms 236
17.9.4 Semantics for External Service provider event query terms 237
17.9.5 Semantics for External service provider event source terms 238
17.9.6 General semantics 239

18 OTX File extension 242
18.1 General 242
18.2 Datatypes 243
18.2.1 Datatypes overview 243
18.2.2 Datatypes syntax 243
18.2.3 Datatypes semantics 243
18.3 Exceptions 244
18.3.1 Exceptions overview 244
18.3.2 Exceptions syntax 244
18.3.3 Exceptions semantics 245
18.4 Variable access 246
18.4.1 Variable-access overview 246
18.4.2 Variable-access syntax 246
18.4.3 Variable-access semantics 247
18.5 Actions 247
18.5.1 Actions overview 247
18.5.2 Actions syntax 247
18.5.3 Actions general semantics 249
18.6 Terms 252
18.6.1 Terms overview 252
18.6.2 Terms syntax 252
18.6.3 Terms general semantics 254

19 OTX FlashPlus extension 263
19.1 General 263
19.2 Exceptions 264
19.2.1 Exceptions overview 264
19.2.2 Exceptions syntax 264
19.2.3 Exceptions semantics 264
19.3 Actions 264
19.3.1 Actions overview 264
19.3.2 Actions syntax 264
19.3.3 Actions general semantics 265
19.4 Terms 266
19.4.1 Terms overview 266
19.4.2 Terms syntax 266
19.4.3 Terms general semantics 266

20 OTX Persistence extension 268
20.1 General 268
20.2 Exceptions 269
20.2.1 Exceptions overview 269
20.2.2 Exceptions syntax 269

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

20.2.3 Exceptions semantics ... 269
20.3 Actions ... 270
20.3.1 Actions overview ... 270
20.3.2 Actions syntax ... 270
20.3.3 Actions general semantics ... 270

21 OTX SQL extension ... 271
21.1 General ... 271
21.2 Datatypes ... 272
21.2.1 Datatypes overview ... 272
21.2.2 Datatypes syntax ... 272
21.2.3 Datatypes semantics ... 272
21.3 Exceptions ... 273
21.3.1 Exceptions overview ... 273
21.3.2 Exceptions syntax ... 273
21.3.3 Exceptions semantics ... 273
21.4 Variable access ... 274
21.4.1 Variable-access overview ... 274
21.4.2 Variable-access syntax ... 274
21.4.3 Variable-access semantics ... 274
21.5 Actions ... 275
21.5.1 Actions overview ... 275
21.5.2 Actions syntax ... 275
21.5.3 Actions general semantics ... 275
21.6 Terms ... 277
21.6.1 Terms overview ... 277
21.6.2 Terms syntax ... 277
21.6.3 Terms general semantics ... 278

22 OTX StateMachineProcedure extension ... 284
22.1 General ... 284
22.2 Datatypes ... 286
22.2.1 Datatypes overview ... 286
22.2.2 Datatypes syntax ... 286
22.2.3 Datatypes semantics ... 287
22.3 Procedures ... 289
22.3.1 Procedures overview ... 289
22.3.2 Procedures syntax ... 289
22.3.3 Procedures general semantics ... 290

23 OTX StateVariable extension ... 293
23.1 General ... 293
23.2 Declaration and arguments ... 294
23.2.1 Declaration and arguments syntax ... 294
23.2.2 Declaration and arguments semantics ... 294

24 OTX TestResultHandling extension ... 294
24.1 General ... 294
24.2 Datatypes ... 295
24.2.1 Datatypes overview ... 295
24.2.2 Datatypes syntax ... 295
24.2.3 Datatypes semantics ... 296
24.3 Enumerations ... 298
24.3.1 Enumerations syntax ... 298
24.3.2 Enumerations semantics ... 299
24.4 Exceptions ... 300
24.4.1 Exceptions overview ... 300
24.4.2 Exceptions syntax ... 300
24.4.3 Exceptions semantics ... 301
24.5 Variable access ... 301

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

24.5.1 Variable-access overview ... 301
24.5.2 Variable-access syntax ... 301
24.5.3 Variable-access semantics ... 302
24.6 Declaration and arguments ... 302
24.6.1 Declaration and arguments syntax ... 302
24.6.2 Declaration and arguments semantics ... 303
24.7 Actions ... 303
24.7.1 Actions overview ... 303
24.7.2 Actions syntax ... 303
24.7.3 Actions general semantics ... 305
24.8 Terms ... 313
24.8.1 Terms overview ... 313
24.8.2 Terms syntax ... 314
24.8.3 Semantics for Enumeration terms ... 315
24.8.4 Terms general semantics ... 318

25 OTX Util extension ... 320
25.1 General ... 320
25.2 Exceptions ... 320
25.2.1 Exceptions overview ... 320
25.2.2 Exceptions syntax ... 321
25.2.3 Exceptions semantics ... 321
25.3 Terms ... 322
25.3.1 Terms overview ... 322
25.3.2 Terms syntax ... 322
25.3.3 Semantics for Util terms ... 324

26 OTX VehicleInfo extension ... 330
26.1 General ... 330
26.2 Datatypes ... 331
26.2.1 Datatypes overview ... 331
26.2.2 Datatypes syntax ... 331
26.2.3 Datatypes semantics ... 331
26.3 Enumerations ... 331
26.3.1 Enumerations syntax ... 331
26.3.2 Enumerations semantics ... 332
26.4 Exceptions ... 332
26.4.1 Exceptions overview ... 332
26.4.2 Exceptions syntax ... 332
26.4.3 Exceptions semantics ... 333
26.5 Variable access ... 333
26.5.1 Variable-access overview ... 333
26.5.2 Variable-access syntax ... 334
26.5.3 Variable-access semantics ... 334
26.6 Terms ... 334
26.6.1 Terms overview ... 334
26.6.2 Terms syntax ... 334
26.6.3 Terms general semantics ... 335

27 OTX XML extension ... 337
27.1 General ... 337
27.2 Datatypes ... 338
27.2.1 Datatypes overview ... 338
27.2.2 Datatypes syntax ... 338
27.2.3 Datatypes semantics ... 338
27.3 Exceptions ... 338
27.3.1 Exceptions overview ... 338
27.3.2 Exceptions syntax ... 339
27.3.3 Exceptions semantics ... 339
27.4 Variable access ... 340

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

27.4.1 Variable-access overview 340
27.4.2 Variable-access syntax 340
27.4.3 Variable-access semantics 340
27.5 Actions 340
27.5.1 Actions overview 340
27.5.2 Actions syntax 340
27.5.3 Actions general semantics 342
27.6 Terms 346
27.6.1 Terms overview 346
27.6.2 Terms syntax 346
27.6.3 Terms general semantics 348

28 OTX ZipHandling extension 354
28.1 General 354
28.2 Exceptions 354
28.2.1 Exceptions overview 354
28.2.2 Exceptions syntax 354
28.2.3 Exceptions semantics 355
28.3 Actions 356
28.3.1 Actions overview 356
28.3.2 Actions syntax 356
28.3.3 Actions general semantics 357

Annex A (normative) **Comprehensive checker rule listing 361**

Annex B (normative) **PTX file 369**

Bibliography 379

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

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

A list of all parts in the ISO 13209 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Introduction

Diagnostic test sequences are utilized whenever automotive components or functions with diagnostic abilities are being diagnosed, tested, reprogrammed or initialised by off-board test equipment. Test sequences define the succession of interactions between the user (i.e. workshop or assembly line staff), the diagnostic application (the test equipment) and the vehicle communication interface as well as any calculations and decisions that have to be carried out. Test sequences provide a means to define interactive, guided diagnostics or similar test logic.

Today, the automotive industry mainly relies on paper documentation and/or proprietary authoring environmments to document and to implement such test sequences for a specific test application. An author who is setting up engineering, assembly line or service diagnostic test applications implements the required test sequences manually, supported by non-uniform test sequence documentation, most likely using different authoring applications and formats for each specific test application. This redundant effort can be greatly reduced if processes and tools support the OTX concept.

The ISO 13209 series proposes an open and standardized format for the human- and machine-readable description of diagnostic test sequences. The format supports the requirements of transferring diagnostic test sequence logic uniformly between electronic system suppliers, vehicle manufacturers and service dealerships/repair shops.

This document extends the core of the ISO 13209 series with a set of additional, generally usable OTX extensions which are not part of ISO 13209-3, using the extension mechanism rules described in ISO 13209-2.

Figure 1 gives an overview of the additional extensions related to the existing extensions in ISO 13209-3.

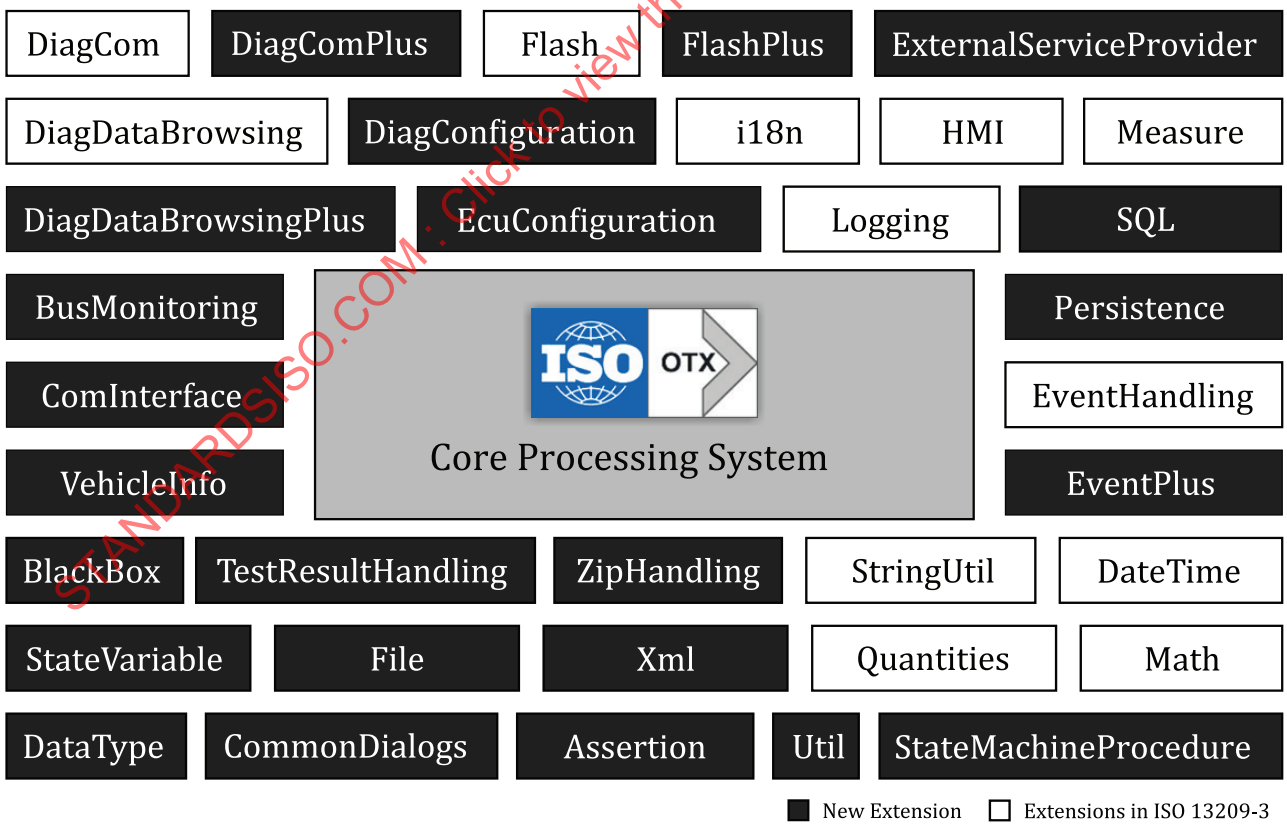


Figure 1 — OTX extensions overview

Besides these extensions a new container format, called PTX, is introduced (see Annex B).

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

The main goals are:

— enlargement of the field of application,

— ensuring exchangeability between different tools and suppliers,

— improvement of OTX code quality,

— improvement of standard acceptance.

The following common requirements are covered:

— general read-write access to files,

— XML processing,

— declare data, whose internal properties are unknown,

— deep change monitoring in lists or maps,

— store arbitrary information on the current runtime,

— handling of user-defined data types: structure and enumeration,

— ensure correct implementation of OTX test sequences,

— support of DoIP,

— configuration tasks,

— browsing of ODX database,

— extend OTX to functionality that is encapsulated in external services, part of a system, a device, a database or a simple library,

— functional access to ECUs,

— capturing, evaluating and persisting results of test sequences,

— transport status information from inside a sequence to the environment,

— advanced convenience functionality,

— collection of communication data at runtime,

— select and modify project and vehicle information's foe MVCI systems,

— support of variant coding,

— enable late bound flashing of ECUs,

— access to SQL databases,

— evaluating test sequence results in a structured way,

— access to information about vehicle network architecture,

— state machine,

— compressed data exchange.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Road vehicles — Open Test sequence eXchange format (OTX) —

Part 4:
Expanded extensions interface definition

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

1 Scope

This document defines the Open Test sequence eXchange (OTX) additional extension requirements and data model specifications.

The requirements are derived from the use cases described in ISO 13209-1. They are listed in Clause 4.

The data model specification aims at an exhaustive definition of all features of the OTX extensions which have been implemented to satisfy the requirements. This document establishes rules for the syntactical entities of each extension. Each of these syntactical entities is accompanied by semantic rules which determine how OTX documents containing extension features are to be interpreted. The syntax rules are provided by UML class diagrams and XML schemas, whereas the semantics are given by UML activity diagrams and prose definitions.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 13209-1:2011, *Road vehicles — Open Test sequence eXchange format (OTX) — Part 1: General information and use cases*

ISO 13209-2:—[1)], *Road vehicles — Open Test sequence eXchange format (OTX) — Part 2: Core data model specification and requirements*

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 13209-1, ISO 13209-2 apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

1) Second edition under preparation. Stage at the time of publication: ISO/DIS 13209-2:2021.

© ISO 2021 – All rights reserved

3.2 Abbreviated terms

API	Application Programmers Interface
ECU	Electronic Control Unit
MVCI	Modular Vehicle Communication Interface
OTX	Open Test sequence eXchange
PDU	Protocol Data Unit
URI	Uniform Resource Identifier
XSD	XML-Schema Definition
XML	Extensible Markup Language

4 Requirements

4.1 Basic principles for requirements definition

Some basic principles have been established as a guideline to define the OTX requirements.

a) OTX requirements specify the conditions that the OTX data model and format shall satisfy.

b) All stakeholders (system suppliers, OEMs, tool suppliers), which offer test procedures are expected to implement and follow the requirements of this document.

The content of OTX documents and the quality of the information is the responsibility of the originator.

4.2 Requirement listing

4.2.1 Assertion requirements

4.2.1.1 A01 – Add assertions to OTX procedures to verify correctness

Rationale: It shall be possible to include actions which verify that preconditions and/or postconditions are true.

Description: During development time, a failing assertion indicates that a programming mistake has been made.

4.2.2 BlackBox requirements

4.2.2.1 BB01 – Transport unknown data types

Rationale: It shall be possible to include actions which verify that preconditions and/or postconditions are true.

Description: It can be necessary to transport values of unknown data types in OTX. For example a user-specific extension needs to handle such values or a method of an external service provider that has properties of unknown complex data types.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

4.2.3 BusMonitoring requirements

4.2.3.1 BM01 – Bus monitoring

Rationale: It shall be possible to monitor a bus system like CAN.

Description: Frames on a connected bus shall be accessible for using signal data on the monitored bus.

4.2.3.2 BM02 – Bus tracing

Rationale: It shall be possible to trace a bus system like CAN.

Description: Frames on a connected bus shall be traceable for saving traces in test context and analyse them after a test.

4.2.4 ComInterface requirements

4.2.4.1 CI01 – Get communication interfaces and their resources at runtime and use them

Rationale: It shall be possible to get communication interfaces at runtime and use them for example to get a new **diag:ComChannel** on this interface.

Description: Functionality is required to get available communication interfaces and their resources at runtime and use them for diagnostics.

4.2.4.2 CI02 – Using of communication interfaces methods

Rationale: It shall be possible to use communication interfaces and their methods (e.g. get the current battery voltage or ignition state).

Description: Functionality is required to use communication interface methods.

4.2.4.3 CI03 – Activate and deactivate Ethernet

Rationale: It shall be possible to activate and deactivate the Ethernet on a VCI.

Description: Functionality is required to activate and deactivate Ethernet on a VCI.

4.2.5 CommonDialogs requirements

4.2.5.1 CD01 – It shall be possible to select a file system resource for the purposes of loading or saving file(s)

Rationale: There are occasions during the execution of an OTX sequence when a file system resource shall be selected, either to read or write information. The runtime system should provide a standard mechanism to request such resource(s).

Description: Dialogs are required to satisfy certain use cases (such as file system resource selection). Access to these resources and how information is returned is standardized to ensure exchangeability is preserved.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

4.2.6 DataType requirements

4.2.6.1 DT01 – Efficient access to named collections (enumeration)

Rationale: It shall be possible to support well known and often used enumerations.

Description: An enumeration provides an efficient way to define a set of named integral constants that may be assigned to a variable.

4.2.6.2 DT02 – Support of structures

Rationale: It shall be possible to support well known and often used structures.

Description: A structure is a value type that is typically used to encapsulate small groups of related variables, such as the coordinates of a rectangle or the characteristics of an item in an inventory.

4.2.7 DiagComPlus requirements

4.2.7.1 DCP01 – Creating diag:ComChannel without starting communication

Rationale: It shall be possible to create a **diag:ComChannel** without starting communication.

Description: A functionality that creates a **diag:ComChannel** without starting communication is required.

4.2.7.2 DCP02 – Getting the state of an initialised diag:ComChannel

Rationale: It shall be possible to get the current state of a **diag:ComChannel**.

Description: A functionality that gets the current state of a **diag:ComChannel** is required.

4.2.7.3 DCP03 – Getting all ComParameters as otx:List

Rationale: It shall be possible to get all ComParameters of a **diag:ComChannel** in an **otx:List** of **otx:String**.

Description: A functionality that gets all ComParameters of a **diag:ComChannel** in an **otx:List** of **otx:String** is required.

4.2.7.4 DCP04 – Executing a functional hex service and getting responses of each ECU that responded

Rationale: It shall be possible to execute a functional hexadecimal diagnostic service and get all available responses.

Description: A functionality that executes a functional hexadecimal diagnostic service and ensures that all responses of this service can be retrieved is required.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

4.2.7.5 DCP05 – Start and stop communication

Rationale: It shall be possible to explicitly execute a start communication and stop communication on a **diag:ComChannel**.

Description: Functionality is required with which an execution of a start communication and a stop communication on a **diag:ComChannel** is enabled.

4.2.8 DiagConfiguration requirements

4.2.8.1 DC01 – Selecting the diagnostic project that contains the diagnostic database

Rationale: It shall be possible to select and change the active diagnostic project that contains the diagnostic database while executing an OTX test sequence.

Description: In some cases there is a requirement to change the active database project.

4.2.8.2 DC02 – Selecting the vehicle information that contains the settings of the diagnostic database

Rationale: It shall be possible to select and change the active vehicle information that contains the diagnostic database while executing an OTX test sequence.

Description: If a change of the active database project is needed, the vehicle information shall be changeable as well.

4.2.9 DiagDataBrowsingPlus requirements

4.2.9.1 DDBP01 – Reading static information for MCDDbParameter

Rationale: It shall be possible to read static information for MCDDbParameter from the underlying ODX database.

Description: Detailed information of MCDDbParameters specified in the ODX database shall be read.

4.2.9.2 DDBP02 – Reading static information for MCDDbTable

Rationale: It shall be possible to read static information for MCDDbTable from the underlying ODX database.

Description: Detailed information of MCDDbTables specified in the ODX database shall be read.

4.2.9.3 DDBP03 – Reading static information for MCDDbEnvDataDesc

Rationale: It shall be possible to read static information for MCDDbEnvDataDesc from the underlying ODX database.

Description: Detailed information of MCDDbEnvDataDescs specified in the ODX database shall be read.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

4.2.9.4 DDBP04 – Reading static information for MCDDbDiagTroubleCode

Rationale: It shall be possible to read static information for MCDDbDiagTroubleCode from the underlying ODX database.

Description: Detailed information of MCDDbDiagTroubleCodes specified in the ODX database shall be read.

4.2.9.5 DDBP05 – Reading static information for MCDDbFaultMemory

Rationale: It shall be possible to read static information for MCDDbFaultMemory from the underlying ODX database.

Description: Detailed information of MCDDbFaultMemories specified in the ODX database shall be read.

4.2.9.6 DDBP06 – Reading static information for MCDDbMatchingParameter

Rationale: It shall be possible to read static information for MCDDbMatchingParameter from the underlying ODX database.

Description: Detailed information of MCDDbMatchingParameters specified in the ODX database shall be read.

4.2.9.7 DDBP07 – Reading static information for MCDDbSpecialDataElement

Rationale: It shall be possible to read static information for MCDDbSpecialDataElement from the underlying ODX database.

Description: Detailed information of MCDDbSpecialDataElements specified in the ODX database shall be read.

4.2.9.8 DDBP08 – Reading static information for MCDDbSpecialDataGroup

Rationale: It shall be possible to read static information for MCDDbSpecialDataGroup from the underlying ODX database.

Description: Detailed information of MCDDbSpecialDataGroups specified in the ODX database shall be read.

4.2.9.9 DDBP09 – Reading static information for MCDDbTextTableElement

Rationale: It shall be possible to read static information for MCDDbTextTableElement from the underlying ODX database.

Description: Detailed information of MCDDbTextTableElements specified in the ODX database shall be read.

4.2.9.10 DDBP10 – Reading static information for MCDInterval

Rationale: It shall be possible to read static information for MCDInterval from the underlying ODX database.

Description: Detailed information of MCDIntervals specified in the ODX database shall be read.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

4.2.9.11 DDBP11 – Reading static information for MCDDbSubComponent

Rationale: It shall be possible to read static information for MCDDbSubComponent from the underlying ODX database.

Description: Detailed information of MCDDbSubComponents specified in the ODX database shall be read.

4.2.10 ECUConfiguration requirements

4.2.10.1 EC01 – Configuration of an ECU by using a database (ODX-E file)

Rationale: In some cases ODX-E files are used to configure an ECU. Therefore configuration by using such files shall be enabled.

Description: OTX actions and terms are required to enable ECU configuration via a database.

4.2.11 EventPlus requirements

4.2.11.1 EP01 – Monitor changes on variables of complex types

Rationale: The OTX Core **EventHandling** extension shall be extended by a mechanism to monitor complex types.

Description: It shall be possible to react to changes deep inside the value of a variable which has a complex type, e.g. a List or Map variable.

4.2.12 ExternalServiceProvider requirements

4.2.12.1 ESP01 – Add method calls of encapsulated libraries into OTX test sequences

Rationale: Authors are able to add supplied functionality into OTX.

Description: Existing functionality of a third party company, that is encapsulated in a library, shall be added into an OTX test sequence (e.g. seed/key).

4.2.12.2 ESP02 – Add a proprietary test sequence into an OTX test sequence

Rationale: The author has the possibility to add small and simple pieces of OTX code into an already running proprietary test sequence. A step by step migration of existing proprietary test logic into OTX code is also possible.

Description: With the possibility to call a proprietary test sequence out of OTX, the author is able to exchange existing parts of proprietary test sequence into OTX (step by step migration). The author is also able to add new functionality written in OTX into proprietary test sequences. Accordingly the introduction of OTX test sequences is not complex and scalable.

4.2.12.3 ESP03 – Add any external functionality to an OTX test sequence

Rationale: Add any external functionality to an OTX test sequence.

Description: With the ExternalServiceProvider the author is able to connect custom specific systems to OTX test sequences.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

4.2.13 File requirements

4.2.13.1 F01 – Read and write textual files

Rationale: It shall be possible to read and write textual files in the local file system.

Description: An OTX extension is required which allows access to textual files.

4.2.13.2 F02 – Read and write binary files

Rationale: It shall be possible to read and write binary files in the local file system.

Description: An OTX extension is required which allows access to binary files.

4.2.14 FlashPlus requirements

4.2.14.1 FP01 – Enable latebound flashing (select binary file while execution of a test)

Rationale: It shall be possible to get, set and change binary flash files for **flash:FlashSession** at runtime.

Description: In some cases there is a requirement to set a binary flash file while executing a script.

4.2.15 Persistence requirements

4.2.15.1 P01 – It shall be possible to save and load the value of all SimpleTypes and ByteField using persistent storage

Rationale: Authors should not be burdened with details of the conversion to file format – the storage format and location should be the responsibility of the runtime system.

Description: Simple types and ByteFields are literal data that can successfully be saved between sessions. These types are the building blocks of more complex structured data.

4.2.15.2 P02 – It shall be possible to save and load collection types (list and map) using persistent storage if the contents of the collections can also be persisted

Rationale: If the contents of lists and maps can be persisted, then lists and maps of these types can also be persisted.

Description: Extending persistence to collection types gives authors a powerful tool for saving and loading structured data with single actions.

4.2.15.3 P03 – It shall be possible for OTX Extensions to define additional data types that can persist

Rationale: There may be other data types defined by OTX extensions that need to persist.

Description: See the relevant extensions for details.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

4.2.16 SQL requirements

4.2.16.1 SQL01 – Establishing database connections

Rationale: It shall be possible to create a database connection.

Description: A certain database connection can be established by the runtime system.

4.2.16.2 SQL02 – Executing of database commands

Rationale: It shall be possible to execute commands at a database connection.

Description: Depending of the database system, a command can be, e.g. an OTX SQL extension command.

4.2.16.3 SQL03 – Retrieving results of a database command

Rationale: It shall be possible to retrieve results from an executed command.

Description: The values of the result can be in an arbitrary structure.

4.2.16.4 SQL04 – Retrieving results shall be possible synchronously and asynchronously

Rationale: It shall be possible to retrieve results synchronously and asynchronously.

Description: If the data records read asynchronously the sequence shall not have to wait to retrieve a large amount of data.

4.2.17 StateMachineProcedure requirements

4.2.17.1 SMP01 – Description of state based workflows

Rationale: It shall be possible to describe state-based workflows.

Description: A state machine consists of 1 to *n* states and state transitions for the state change which will be realised by the occurrence of conditions supervised by triggers. The extension shall not contain layout information for the runtime system.

4.2.18 StateVariable requirements

4.2.18.1 SV01 – Support for state variables, which are the counterpart of context variables

Rationale: It shall be possible to easily transport status information from inside a sequence to the environment.

Description: State variables can only be evaluated by setting outcomes of the test application, e.g. to display the current execution progress.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

4.2.19 TestResultHandling requirements

4.2.19.1 RH01 – Capturing of test result data

Rationale: Functionality shall be provided to capture test result data.

Description: Test result data (e.g. measured values) shall be captured in a structured way in relation to the environment (e.g. test object, test identifier, date/time). For the test result structure a data model shall be used.

4.2.19.2 RH02 – Evaluation of test result data

Rationale: It shall be possible to evaluate captured test result data.

Description: For each test result, it shall be possible to define individual conditions to evaluate if the test result is OK or NOT OK.

4.2.19.3 RH03 – Summarising of test results in groups

Rationale: It shall be possible to summarise test results in groups.

Description: According to the test result data model, test results can be summarised in groups. It shall be possible to evaluate if the test result group is OK or NOT OK.

4.2.19.4 RH04 – Defining different test result levels

Rationale: It shall be possible to define different test result levels.

Description: It is necessary to have different test result levels according to the development process of OTX sequences, e.g. test results exclusively for debugging tasks.

4.2.19.5 RH05 – Multiple test instances

Rationale: Simultaneous operation with multiple test instances shall be possible.

Description: It shall be possible to capture and evaluate test result data of different test instances (e.g. several vehicles) at the same time.

4.2.20 Util requirements

4.2.20.1 U01 – Copying otx:ByteFields

Rationale: It shall be possible to copy an **otx:ByteField** to simplify assigning a value and not a reference to another **otx:ByteField**.

Description: **otx:ByteField** shall be copiable because it is needed to assign **otx:Bytefield** value of **otx:Bytefield** variable to another **otx:Bytefield** variable.

4.2.20.2 U02 – Comparing values of complex types

Rationale: It shall be possible to compare the values of complex types (not their references like **otx:IsEqual** does) in a simple way.

Description: An easy way to compare complex type values is needed (e.g. comparing of two hexadecimal responses as **otx:ByteField** Variable of an ECU).

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

4.2.20.3 U03 – Getting a random number

Rationale: It shall be possible to get a random number for special test cases.

Description: Some test sequences need to use a random numbers. Therefore, an easy way to generate a random number is needed.

4.2.20.4 U04 – Checking a variable's status

Rationale: It shall be possible to check whether a variable of a complex type is already initialised or not.

Description: In some cases it is important to check whether a complex variable is initialised or not. Therefore, an easy way to check the status of a complex variable is needed.

4.2.20.5 U05 – Simplify analysing the content of a list

Rationale: It shall be possible to simply analyse the content of **otx:List** instances.

Description: Analysis of an **otx:List** shall be simplified, because in some test sequences its analysis is necessary.

4.2.20.6 U06 – More possibilities to format a string

Rationale: It shall be possible to simply format an **otx:String**.

Description: In some cases strings have to be formatted in a special way.

4.2.21 VehicleInfo requirements

4.2.21.1 VI01 – Getting the gateway mode of an ECU

Rationale: It shall be possible to get the gateway mode of an ECU.

Description: An OTX term that gets the gateway mode of an ECU is required.

4.2.21.2 VI02 – Getting names of gateways that accesses an ECU

Rationale: It shall be possible to get the names of gateways that access an ECU.

Description: An OTX term that gets the names of gateways that access an ECU is required.

4.2.21.3 VI03 – Checking if ECU is accessed via gateways

Rationale: It shall be possible to check if an ECU is accessed by a gateway.

Description: An OTX term that checks if an ECU is accessed via one or more gateways is required.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

4.2.22 XML requirements

4.2.22.1 X01 – The user shall be able to set XML processing Instructions

Rationale: The functionality shall be provided to store processing instructions.

Description: Processing instructions at document level define additional information outside the XML root tag. Therefore, an extra functionality is needed to add this information to the XML document.

4.2.22.2 X02 – The user shall be able to edit attributes, nodes and node text values

Rationale: The functionality shall be provided to modify tag attributes, nodes and containing text values.

Description: To create a complete XML structure, the functionality to edit attributes, nodes and the text value between the start tag and end tag is mandatory.

4.2.22.3 X03 – The user shall be able to add comments to an XML document

Rationale: The functionality shall be provided to add comments to an XML document.

Description: There shall be an OTX extension, which allows the appending of comment tags to a document root node.

4.2.22.4 X04 – The user shall be able to remove attributes and nodes

Rationale: The functionality shall be provided to remove attribute entries as well as complete nodes.

Description: There shall be an OTX extension, which allows deleting an attribute of an XML node. It should be possible to delete a node containing all children as well.

4.2.22.5 X05 – Validation of an XML document

Rationale: The functionality shall be provided to validate an XML document.

Description: There shall be an OTX extension, which allows validation of an XML document against a referenced schema.

4.2.22.6 X06 – The user shall be able to define the document encoding and version

Rationale: The functionality shall be provided to define the document encoding and version.

Description: There shall be an OTX extension, which allows the addition of the encoding and version as process instructions at the head of the XML document.

4.2.22.7 X07 – The user shall be able to parse XML content from file or ByteField

Rationale: The functionality shall be provided to parse an XML document tree from an addressed file or in-memory ByteField.

Description: An extension to parse an XML document from a file which is addressed by a URI should be provided. An in-memory ByteField as input shall be supported as well.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

4.2.22.8 X08 – The user shall be able to query XML elements by XPath

Rationale: The functionality shall be provided to address an XML element using an XPath query.

Description: An extension to collect all matching XML elements starting from the document node shall be provided.

4.2.23 ZipHandling requirements

4.2.23.1 Z01 – Functionality shall be provided to extract a zip file

Rationale: The user shall be able to extract a zip file.

Description: An extension for zip file handling shall provide the possibility to extract a zip file to a target directory.

4.2.23.2 Z02 – Filter functionality for the file compress and decompress process shall be provided

Rationale: To reduce the amount of files, a wildcard filter for the file consistent extension shall be used.

Description: A wildcard filter reduces the amount of extracted or compressed files by applying a filter on the file consistent extension.

4.2.23.3 Z03 – The user shall decide whether existing files should be overwritten by the decompress process.

Rationale: The decompress functionality shall control whether existing files should be overwritten.

Description: The target directory of the extraction process can contain files already. The overwrite action of conflicting files shall be configured by the user.

4.2.23.4 Z04 – The user shall decide whether the target zip file should be overwritten by the compress process

Rationale: The compress functionality shall control how to handle an existing target zip file.

Description: If the addressed target zip file already exists, files can be added to the existing container or it will be overwritten. The functionality has to be configured by the user.

4.2.23.5 Z05 – The zip file shall be a URI

Rationale: The addressing of the zip file shall be performed using the URI definition.

Description: The URI path defines the address of the zip file resource.

5 Extension overview

5.1 General

OTX (Open Test sequence Exchange) is a platform and tester-independent exchange format for formal description of executable test sequences published in the ISO 13209 series. OTX is a standard for the reliable storage of test knowledge within vehicle diagnostics and vehicle test automation. The XML-based flow language provides the opportunity to share test sequences across departmental, tool and

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

process boundaries. The expertise stored in these flows is not lost, but can be used even after many years. OTX is targeted primarily at the automotive industry, although the core features of the standard are applicable in any industry. Such test sequences are used in development, production and after sales and can be exchanged between vehicle manufacturers, tier 1 suppliers and tool providers.

OTX is the logical evolution of standardization in vehicle diagnostics. Figure 2 shows the relation of OTX to other diagnostic standards.

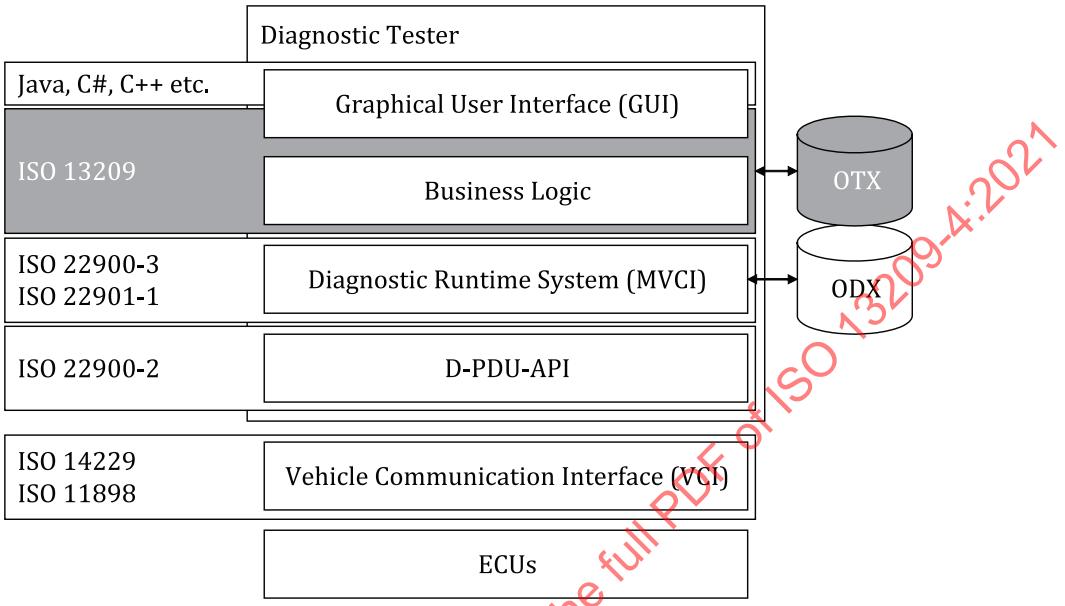


Figure 2 — Relation of OTX to standards in vehicle diagnostics

The ISO 13209 series consists of a core and an extension concept, which allows the integration of arbitrary interfaces. For example, extensions for diagnostic communication, flashing, event handling and logging are standardized in ISO standards. Based on these extensions the usage of an MVCI server (ISO 22900-3 also known as ASAM MCD-3 D Server API) is simplified, because many implementation steps are hidden.

OTX is not restricted to diagnostic sequences. It can be also seen as a strategic migration platform for different, currently separated standards, e.g. ASAM GDI, ASAM XIL, ASAM AE MCD-3 MC.

This document represents the specification of the OTX standard extensions for data model version **"1.0.0"**.

For the convenience of the user the schema definition files (xsd) are published together with the associated examples (otx) alongside this document, available at https://standards.iso.org/iso/13209/-4/ed-1/en/.

For the convenience of the user the schema definition files (xsd) for a possible, not normative TestResult data model are published together with the associated examples (xml) alongside this document, available at https://standards.iso.org/iso/13209/-4/ed-1/en/.

Annex A contains a comprehensive listing of all checker rules and shall be followed. The rules are needed because some constraints existing on OTX documents cannot be ensured by XSD validation alone. These constraints need to be checked by additional checker applications.

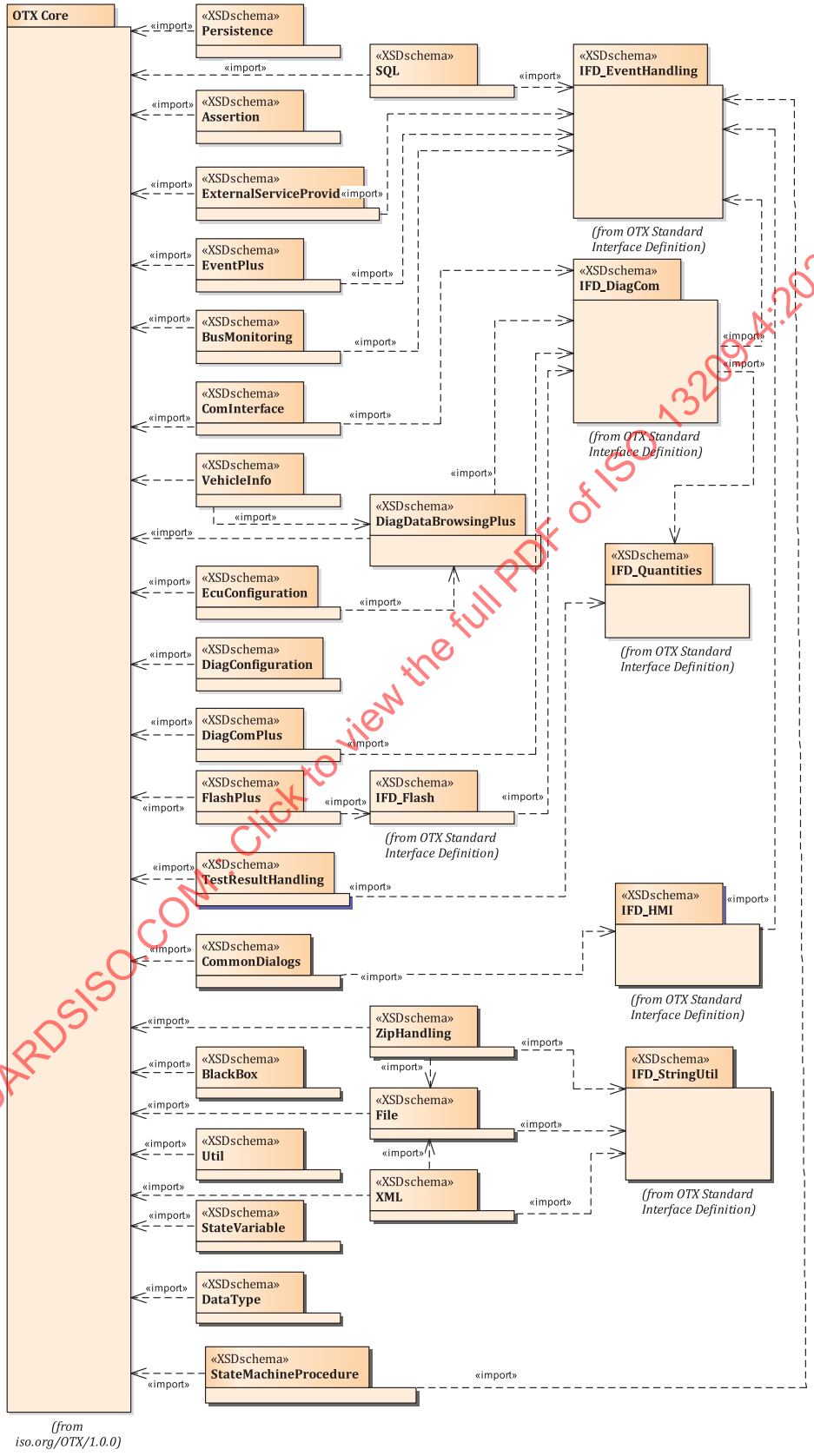
5.2 Dependencies

Figure 3 shows a UML package diagram describing the full set of OTX extensions (together with the OTX Core) and the import dependencies in between them. OTX extensions use or extend types defined

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

in the OTX Core. Therefore, all of the extensions are (directly or indirectly) based on the OTX Core data model, as specified by ISO 13209-2.



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 3 — Overview OTX schema dependencies

© ISO 2021 – All rights reserved

Table 1 provides an overview of additional OTX extensions and their basic characteristics.

The OTX Core is a prerequisite for any OTX application and shall always be fully supported. By contrast, OTX applications are not required to support all of the extensions specified in this document. The set of supported extensions may vary depending on the field of application. However, an OTX application supporting an extension which imports other extensions shall support these extensions also. This guarantees that the set of supported extensions is consistent with regard to the dependencies.

Table 1 — OTX extension characteristics

Extension (schema file)	Summary	Dependencies
Assertion **(otxIFD_Assertion.xsd)**	Ensuring OTX sequence quality	Core
BlackBox **(otxIFD_BlackBox.xsd)**	Transport of customer specific data types	Core
BusMonitoring **(otxIFD_BusMonitoring.xsd)**	Collection of communication data at runtime	Core ComInterface EventHandling
ComInterface **(otxIFD_ComInterface.xsd)**	Handling of communication interfaces and diagnostic over IP	Core DiagCom
CommonDialogs **(otxIFD_CommonDialogs.xsd)**	Dialogs for common use cases	Core HMI
DataType **(otxIFD_DataType.xsd)**	Definition of user defined structure and enumeration data types	Core
DiagComPlus **(otxIFD_DiagComPlus.xsd)**	Support of different ECU responses for hex service and com channel controlling	Core DiagCom
DiagConfiguration **(otxIFD_DiagConfiguration.xsd)**	Configuration of project and vehicle information	Core DiagCom
DiagDataBrowsingPlus **(otxIFD_DiagDataBrowsingPlus.xsd)**	Browsing of ODX database properties	Core DiagCom
ECUConfiguration **(otxIFD_ECUConfiguration.xsd)**	ECU variant coding	Core DiagDataBrowsingPlus
EventPlus **(otxIFD_EventPlus.xsd)**	Support for deep change monitoring in a List or Map	Core EventHandling
ExternalServiceProvider **(otxIFD_ExternalServiceProvider.xsd)**	Provides actions, terms, events and data types for an access to external services	Core EventHandling
File **(otxIFD_File.xsd)**	General read write access to files	Core StringUtil
FlashPlus **(otxIFD_FashPlus.xsd)**	Enable late binding of flash files	Core Flash DiagCom
Persistence **(otxIFD_Persistence.xsd)**	Store runtime information of an equal run-time system	Core
SQL **(otxIFD_SQL.xsd)**	Read and write access to a SQL database	Core EventHandling

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Table 1 *(continued)*

Extension (schema file)	Summary	Dependencies
StateMachineProcedure (otxIFD_StateMachineProcedure.xsd)	Description of a behaviour model with states, transactions and actions	Core EventHandling
StateVariable (otxIFD_StateVariable.xsd)	Mechanism to transport status information from inside a sequence to the environment	Core
TestResultHandling (otxIFD_TestResultHandling.xsd)	Evaluated and structured storage of result Information (e.g. for statistics) from test sequences	Core Quantities
Util (otxIFD_Util.xsd)	Advanced convenience functionality	Core
VehicleInfo (otxIFD_VehicleInfo.xsd)	Access to information about the vehicle network architecture	Core DiagDataBrowsingPlus
Xml (otxIFD_Xml.xsd)	Read and interpret XML documents, modification of existing and creation of new documents	Core File StringUtil
ZipHandling (otxIFD_ZipHandling.xsd)	Compressed data exchange	Core

Table 2 shows the XSD namespace associations of additional OTX extensions. Each namespace has a prefix assigned to it. In the remainder of this document, the prefixes defined here are used to mark types which belong to extensions other than the one which is currently described. In contrast, the types defined by the currently described extension are not prefixed.

Table 2 — OTX extension namespace associations

Extension	Namespace	Prefix
Assertion	http://iso.org/OTX/1.0.0/Assertion	assert:
BlackBox	http://iso.org/OTX/1.0.0/BlackBox	blackBox:
BusMonitoring	http://iso.org/OTX/1.0.0/BusMonitoring	busMon:
ComInterface	http://iso.org/OTX/1.0.0/ComInterface	comIf:
CommonDialogs	http://iso.org/OTX/1.0.0/CommonDialogs	commDlg:
DataType	http://iso.org/OTX/1.0.0/DataType	dataType:
DiagComPlus	http://iso.org/OTX/1.0.0/DiagComPlus	diagComPlus:
DiagConfiguration	http://iso.org/OTX/1.0.0/DiagConfiguration	diagConf:
DiagDataBrowsingPlus	http://iso.org/OTX/1.0.0/DiagDataBrowsingPlus	diagDataPlus:
ECUConfiguration	http://iso.org/OTX/1.0.0/ECUConfiguration	ecuConf:
EventPlus	http://iso.org/OTX/1.0.0/EventPlus	eventPlus:
ExternalServiceProvider	http://iso.org/OTX/1.0.0/ExternalServiceProvider	esp:
File	http://iso.org/OTX/1.0.0/File	file:
FlashPlus	http://iso.org/OTX/1.0.0/FlashPlus	flashPlus:
Persistence	http://iso.org/OTX/1.0.0/Persistence	persist:
SQL	http://iso.org/OTX/1.0.0/SQL	sql:
StateMachineProcedure	http://iso.org/OTX/1.0.0/StateMachineProcedure	smp:
StateVariable	http://iso.org/OTX/1.0.0/StateVariable	stateVar:
TestResultHandling	http://iso.org/OTX/1.0.0/TestResultHandling	testResult:
Util	http://iso.org/OTX/1.0.0/Util	util:

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Table 2 *(continued)*

Extension	Namespace	Prefix
VehicleInfo	http://iso.org/OTX/1.0.0/VehicleInfo	vehInf:
Xml	http://iso.org/OTX/1.0.0/XML	otxXml:
ZipHandling	http://iso.org/OTX/1.0.0/ZipHandling	zip:

5.3 OTX packaging (PTX)

The exchange of a whole OTX project can be realized with a file container called "PTX package". All files of an OTX project are packaged in one PTX container. A PTX container can handle OTX-files with the file extension *.otx as well as arbitrary files of any format that are necessary for the project. The file container format is described in Annex B and shall be followed.

6 OTX Assertion extension

6.1 General

Assertions are used to ensure the correct implementation of OTX test sequences. An assertion claims that a certain condition holds during OTX execution. It does not contribute to the test logic that is implemented in the OTX.

Assertions are usually evaluated during testing of the OTX test sequence itself. A failing assertion always indicates that the OTX test sequence contains a programming error and shall be changed.

During development of an OTX test sequence, assertions shall not be evaluated because it can be assumed that the sequence does not contain any programming errors. Runtime systems should, therefore, offer a configuration to enable or disable the evaluation of assertions. By default, the evaluation of assertions should be disabled. If the runtime system is configured not to evaluate assertion statements, it shall suppress any impact of assertions: it shall not evaluate their boolean conditions or error messages.

6.2 Exceptions

6.2.1 Exceptions overview

All exceptions introduced by this extension are derived from the **otx:Exception** type. They represent the full set of exceptions added by the **OTX Assertion extension**.

6.2.2 Exceptions syntax

The syntax of all **OTX Assertion extension** exception-type declarations is shown in Figure 4.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

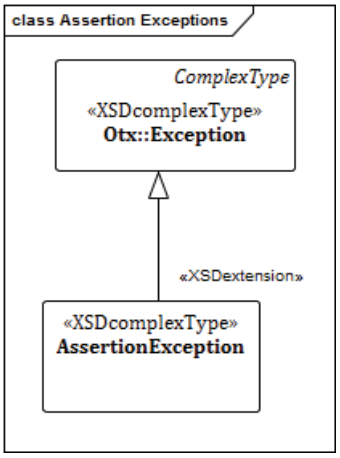


Figure 4 — Assertion exceptions

6.2.3 Exceptions semantics

6.2.3.1 General

Since all **OTX Assertion extension** exception types are implicit exceptions without initialisation parts, they cannot be declared constant.

6.2.3.2 AssertionException

IMPORTANT — All terms and action realisations in this extension may potentially throw this exception.

6.3 Actions

6.3.1 Actions overview

All actions introduced by this extension extend the **otx:ActionRealisation** extension interface.

6.3.2 Actions syntax

Figure 5 shows the syntax of all actions in the **OTX Assertion extension**.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

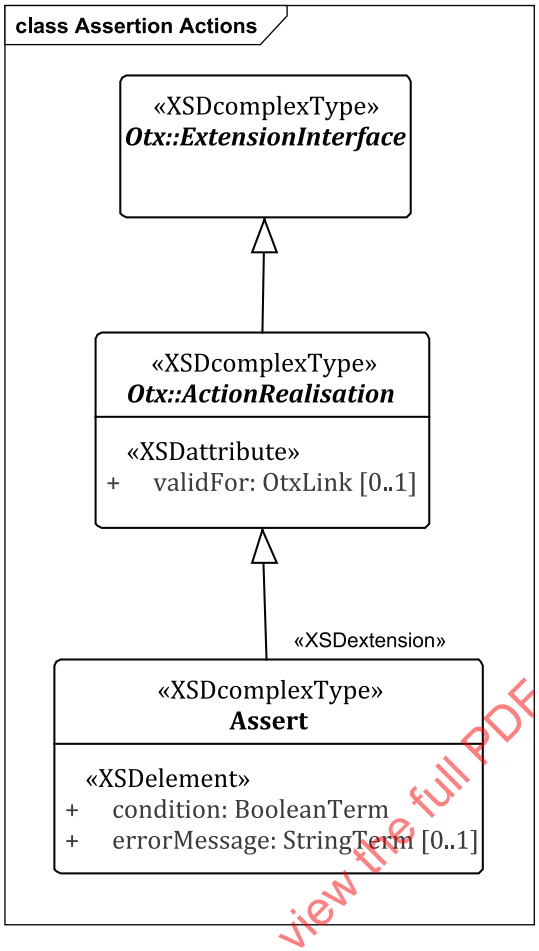


Figure 5 — Assertion actions

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

6.3.3 Actions general semantics

6.3.3.1 Assert

Semantic:

If the runtime is configured to evaluate assertions, it evaluates the given boolean expression.

If the expression evaluates to true, nothing happens.

If the expression evaluates to false, the execution of OTX procedure should be stopped (the exact runtime behaviour is unspecified), and the failed assertion is reported to the runtime system itself (out of OTX scope).

The <errorMessage> term, if present, shall only be evaluated when the assertion failed. In particular, the <errorMessage> term cannot throw any exceptions unless the assertion failed.

If the runtime is configured not to evaluate assertions, the assert:Assert action does nothing. In particular, the <condition> and <errorMessage> terms will not be evaluated, and thus cannot throw any exceptions.

Parameters:

— <condition> : otx:BooleanTerm [1..1]

The expression to evaluate.

© ISO 2021 – All rights reserved

— <errorMessage> : otx:StringTerm [0..1]

An error message which may be included in the report to the runtime system, if the assertion fails.

Exceptions:

— None

7 OTX BlackBox extension

7.1 General

The **OTX BlackBox extension** data type can be used as a transport or container data type. An **OTX BlackBox extension** can contain a value of a user-specific data type which is unknown to OTX.

For example, the main procedure may have an in parameter of type **OTX BlackBox extension**. When the main procedure is called, the runtime system uses this parameter to pass in a value of a proprietary (non-OTX) data type. The OTX procedures do not interact with the contents of this **OTX BlackBox extension**. They pass it on to an **otx:ActionRealisation** of a proprietary OTX extension. This proprietary extension is able to handle the value contained in the **OTX BlackBox extension**. In other words, the user can transport data through code layers (code is embedded in between test system specific runtime code) even if its type is unknown to OTX (not defined in core or standardized extensions). This extension provides no way of initializing **OTX BlackBox extension** values.

Terms and actions can be added in new, proprietary extensions to interact with the **OTX BlackBox extension** data type.

7.2 Datatypes

7.2.1 Datatypes overview

All data types introduced by this extension are derived from **otx:ComplexType**.

7.2.2 Datatypes syntax

The syntax of the data-type declarations of the **OTX BlackBox extension** is shown in Figure 6.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

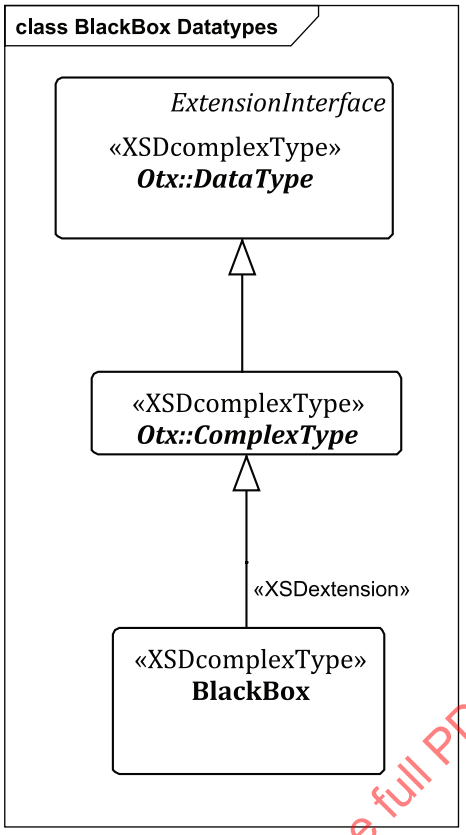


Figure 6 — BlackBox datatypes

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

7.2.3 Datatypes semantics

7.2.3.1 BlackBox

BlackBox is an **otx:ComplexType**.

This type can be used as a transport or container datatype. It can contain a value of a user-specific data type which is unknown to OTX. If OTX code is embedded in between test system specific runtime code, OTX unknown types should be able to be passed through the code layers. This extension provides no way of initializing **BlackBox** values.

7.3 Variable access

7.3.1 Variable-access overview

All variable-access types introduced by this extension are derived from the OTX core variable extension interface.

7.3.2 Variable-access syntax

The syntax of the **OTX BlackBox extension's** variable-access types is shown in Figure 7.

© ISO 2021 – All rights reserved

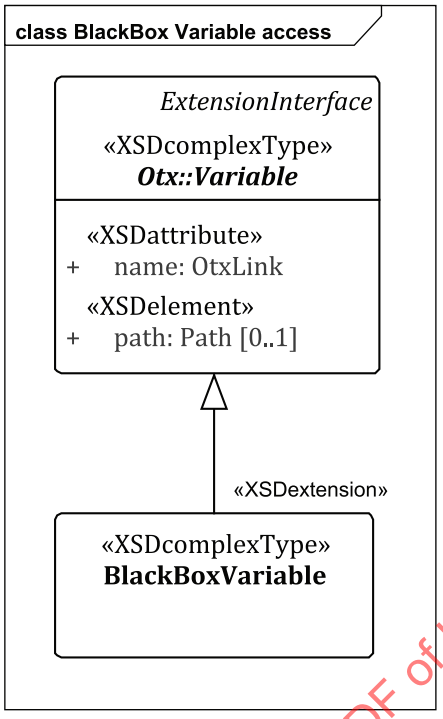


Figure 7 — BlackBox variable access

7.3.3 Variable-access semantics

The general semantics for all variable-access types shall apply. Please refer to ISO 13209-2 for details.

7.4 Terms

7.4.1 Terms overview

All terms introduced by this extension extend the **otx:Term** extension interface. Information about the specific super class of a term is provided in the individual term description clauses.

7.4.2 Terms syntax

Figure 8 shows the syntax of the **OTX BlackBox extension** related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

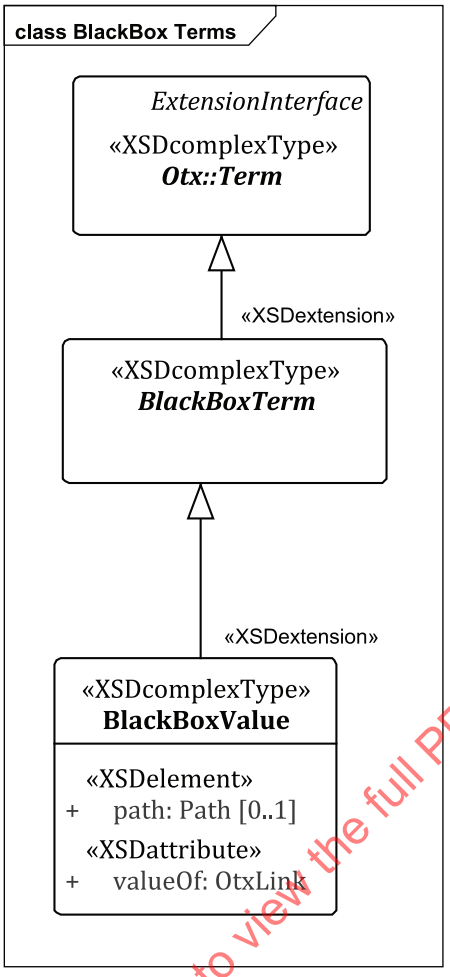


Figure 8 — BlackBox terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

7.4.3 Terms general semantics

7.4.3.1 BlackBoxTerm

Semantic:

The abstract type BlackBoxTerm is an otx:SimpleTerm. It serves as a base for all concrete terms, which return a BlackBox value. It has no special members.

7.4.3.2 BlackBoxValue

Semantic:

Returns the BlackBoxValue stored in a Blackbox variable.

Return type:

blackBox:BlackBoxTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

© ISO 2021 – All rights reserved

Exceptions:

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location, which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

8 OTX BusMonitoring extension

8.1 General

The **OTX BusMonitoring extension** is introduced to enable asynchronous collection of communication data (monitoring) at runtime and analysis of monitored data (frames).

IMPORTANT — The bus monitoring of the underlying system (e.g MVCI server) is not specified. The start of bus monitoring can throw an exception because the system does not support it. Also it is not guaranteed that the terms of the extension return any data because the underlying runtime system cannot get it. In such cases an exception is thrown as well. The extension is created to facilitate using bus monitoring.

NOTE The extension is closely related to an ODX/MVCI based system, as defined in the ISO 22901 and ISO 22900 series.

In case of an ODX/MVCI based system it is recommended that the result of MCDMonitorLink::fetchMonitoringFrames is formatted as given in ISO 22900-3:2012, Annex F. This is related to the terms **GetBusFrameAsString**, **GetBusFrameTimestamp**, **GetBusFrameAddress**, **GetBusFrameData** and **GetBusFrameType**.

8.2 Datatypes

8.2.1 Datatypes overview

All data types introduced by this extension are derived from **otx:ComplexType** or **otx:SimpleType**.

8.2.2 Datatypes syntax

The syntax of the data-type declarations of the **OTX BusMonitoring extension** is shown in Figure 9.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

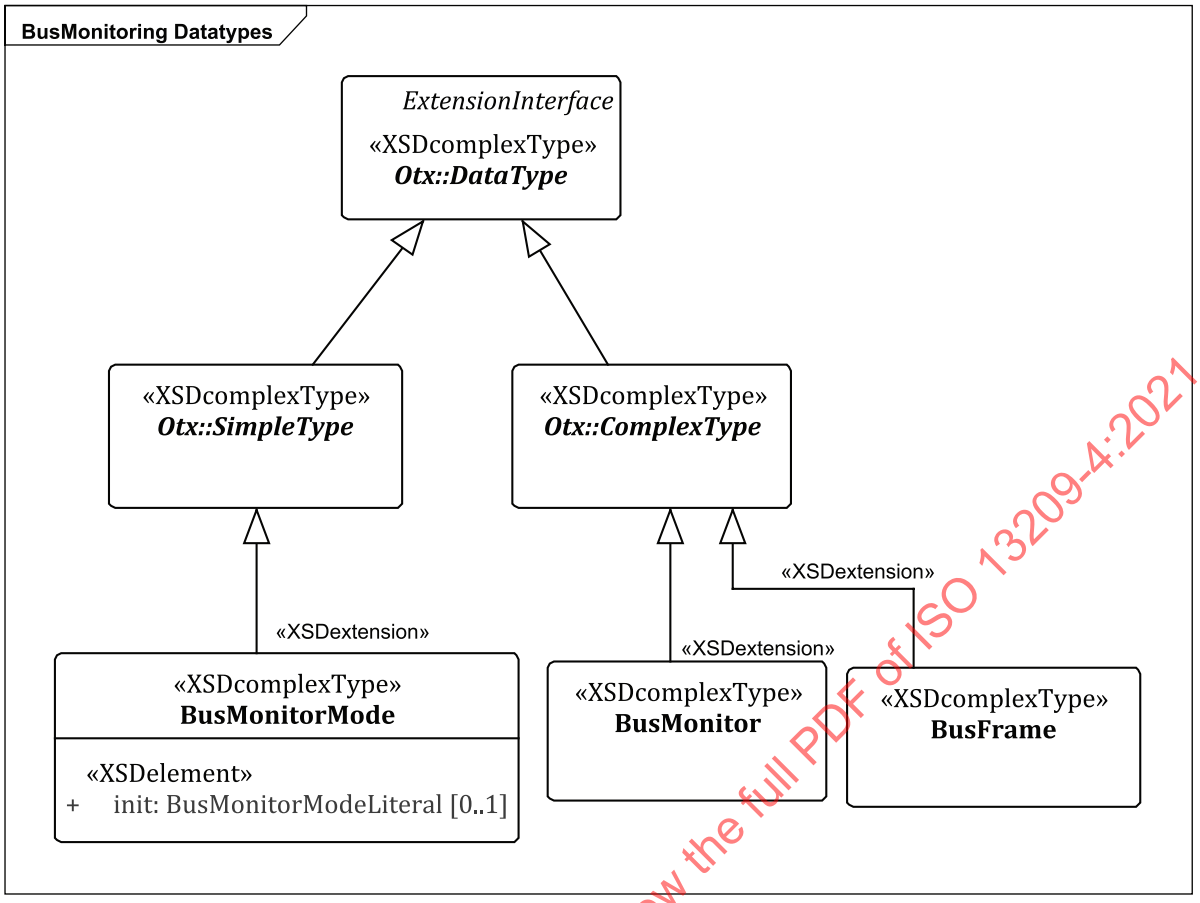


Figure 9 — BusMonitoring datatypes

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

8.2.3 Datatypes semantics

8.2.3.1 BusFrame

BusFrame is an otx:ComplexType.

Represents a handle of a bus frame that contains information about one frame, e.g. a message and a timestamp.

IMPORTANT — The otx:ToString conversion term shall return an otx:String that represents the whole frame delivered by the runtime system.

8.2.3.2 BusMonitor

BusMonitor is an **otx:ComplexType**.

A **BusMonitor** can be used to monitor a communications bus resource available at an interface (VCI module).

NOTE The related MVCI system class is MCDMonitoringLink.

IMPORTANT — In case of an ODX/MVCI based system the otx:ToString conversion shall return the SHORT-NAME of the related MCDMonitoringLink.

8.2.3.3 BusMonitorMode

BusMonitorMode is an **otx:SimpleType.**

© ISO 2021 – All rights reserved

BusMonitorMode is an enumeration type describing the modes of the monitor.

IMPORTANT — The otx:ToString conversion shall return the name of the enumeration value and the otx:ToInteger conversion shall return the index of the value in the enumeration (smallest index is 0).

Its members have the following semantics:

— <init> : busMon:BusMonitorModeLiteral[0..1]

This optional element stands for the hard-coded initialisation value of the identifier at declaration time.

IMPORTANT — If BusMonitorMode declaration is not explicitly initialised, the default value shall be MONITORING.

8.3 Enumerations

8.3.1 Enumerations syntax

The syntax of the enumeration declarations of the **OTX BusMonitoring extension** is shown in Figure 10.

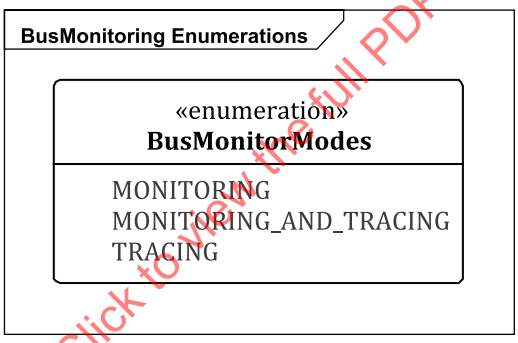


Figure 10 — BusMonitoring enumerations

8.3.2 Enumerations semantics

8.3.2.1 BusMonitorModes

BusMonitorModes is an enumeration type describing the mode of monitoring.

BusMonitorModes has the following enumeration values.

— MONITORING

Received frames will be stored into an internal buffer and NOT written into a trace resource.

— MONITORING_AND_TRACING

Received frames will be stored into an internal buffer and written into a trace resource.

— TRACING

Received frames will be written into a trace resource and NOT stored into an internal buffer.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

8.4 Exceptions

8.4.1 Exceptions overview

All exceptions introduced by this extension are derived from the **otx:Exception** type. They represent the full set of exceptions added by the **OTX BusMonitoring extension**.

8.4.2 Exceptions syntax

The syntax of all **OTX BusMonitoring extension** exception-type declarations is shown in Figure 11.

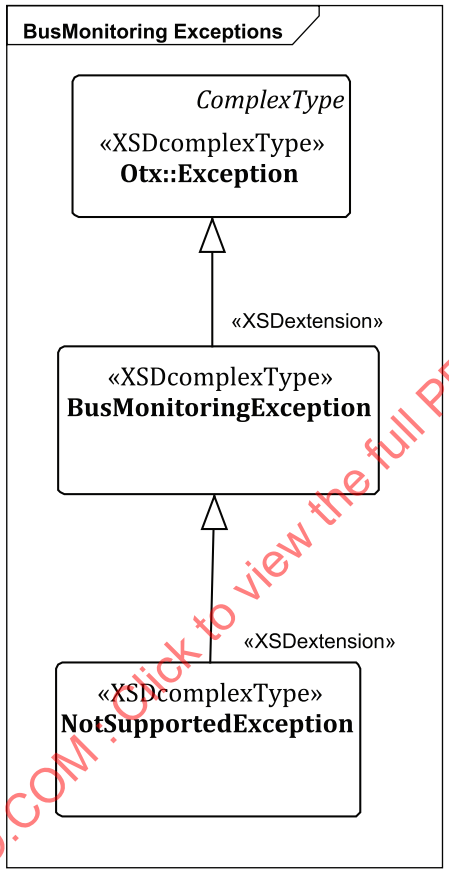


Figure 11 — BusMonitoring exceptions

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

8.4.3 Exceptions semantics

8.4.3.1 General

Since all **OTX BusMonitoring extension** exception types are implicit exceptions without initialisation parts, they cannot be declared constant.

8.4.3.2 BusMonitoringException

The **BusMonitoringException** is the super class for all exceptions in the **OTX BusMonitoring extension**. A **BusMonitoringException** shall be used in case the more specific exception types described in the remainder of this section do not apply to the problem at hand.

IMPORTANT — All terms and action realisations in this extension may potentially throw this exception.

© ISO 2021 – All rights reserved

8.4.3.3 NotSupportedException

This is thrown if the underlying runtime system (MVCI-Server) does not support this operation.

8.5 Variable access

8.5.1 Variable-access overview

All variable-access types introduced by this extension are derived from the OTX core variable extension interface.

8.5.2 Variable-access syntax

The syntax of the **OTX BusMonitoring extension** variable-access types is shown in Figure 12.

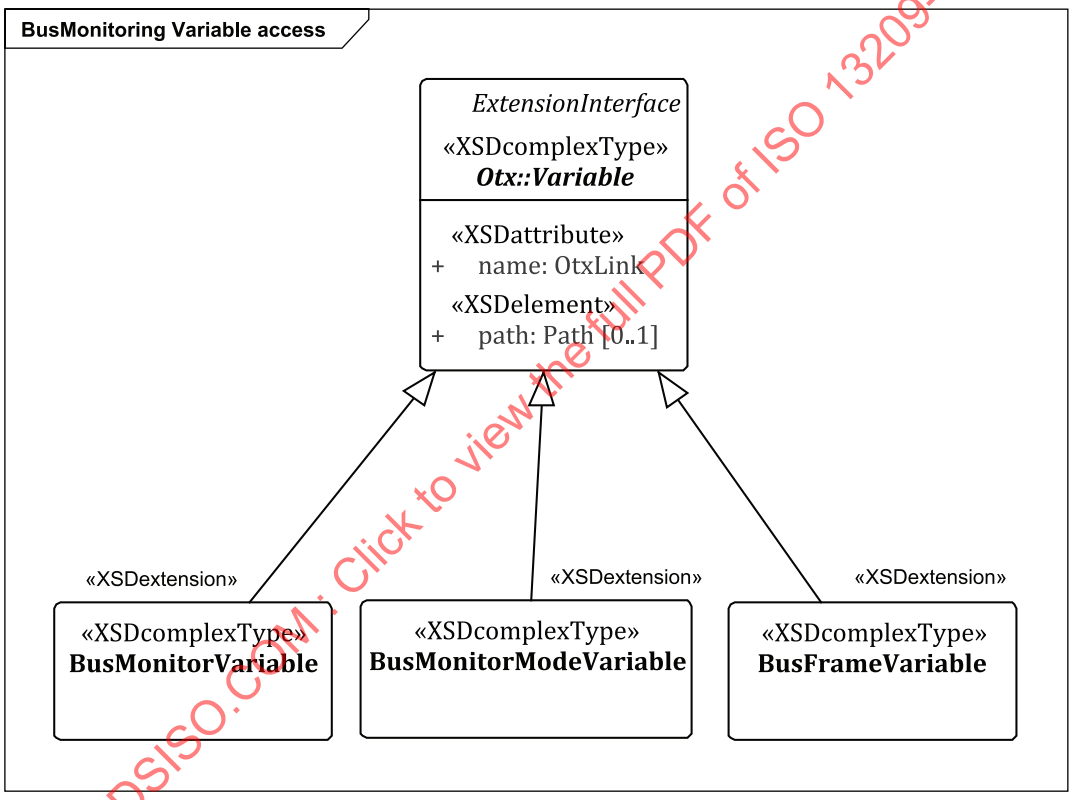


Figure 12 — BusMonitoring variable access

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

8.5.3 Variable-access semantics

The general semantics for all variable-access types shall apply. Please refer to ISO 13209-2 for details.

8.6 Actions

8.6.1 Actions overview

All actions introduced by this extension extend the **otx:ActionRealisation** extension interface.

8.6.2 Actions syntax

Figure 13 shows the syntax of all actions in the **OTX BusMonitoring extension**.

© ISO 2021 – All rights reserved

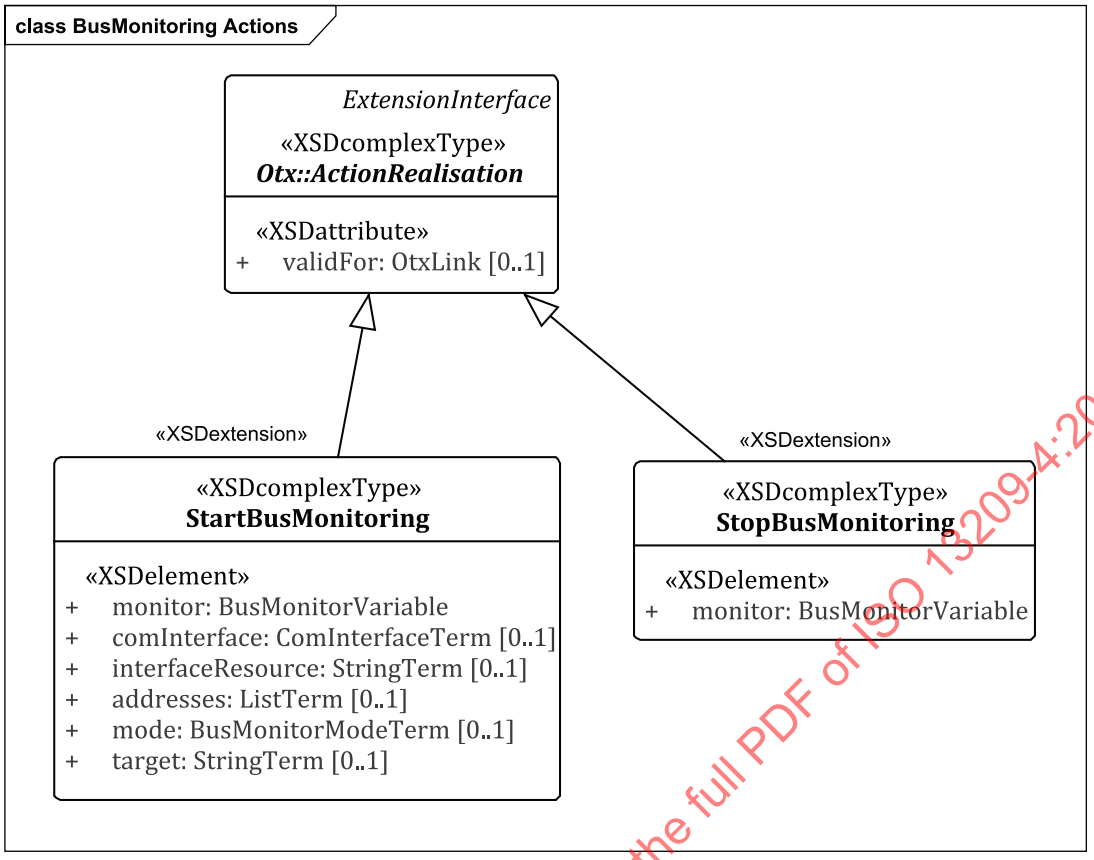


Figure 13 — BusMonitoring actions

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

8.6.3 Actions general semantics

8.6.3.1 StartBusMonitoring

Semantic:

Starts the monitoring and optional tracing on a physical link. Use FetchBusMonitorBusFrames to retrieve frames on a BusMonitor.

NOTE The related MVCI system methods are MCDProject::createMonitoringLink(MCDInterfaceResource mCDInterfaceResource); MCDMonitorLink::getMessageFilters(); MCDMessageFilters::addByFilterType(int messageFilterType); MCDMessageFilter::enableMessageFilter(true); MCDMonitoringLink::start().

Associated checker rules:

— BusMonitoring_Chk001 – correct element types for adresses list in StartBusMonitoring.

Parameters:

— <monitor> : busMon:BusMonitorVariable [1..1]

This element represents the variable that refers to the handle for the monitor started by this action. This can be used later to fetch the frames of the monitor (FetchBusMonitorBusFrames) or stop the monitor explicitly (StopBusMonitoring).

— <comInterface> : comIf:ComInterfaceTerm [0..1]

Represents a handle for the communication interface on which the monitoring should be started. If no **comIf:ComInterface** is given, the default communication interface shall be used (the interface that is needed for **diag:GetComChannel,** see ComInterface extension).

© ISO 2021 – All rights reserved

— <interfaceResource> : otx:StringTerm [0..1]

Represents the name of the physical link, e.g. "CAN1". If no interface resource is given, the default interface resource shall be used (interface that is needed for **diag:GetComChannel,** see ComInterface extension).

— <addresses> : otx:ListTerm [0..1]

Represents an optional list of integer numbers, defining the addresses to be monitored (e.g. on a CAN bus the CAN IDs of signals or ECUs). If no list or an empty list is given, all data shall be monitored.

NOTE In case an MVCI system is used pass filters are applied to CAN ID filtering.

— <mode> : busMon:BusMonitorModeTerm [0..1]

Represents an optional mode to specify how the monitored frames shall be processed, see **BusMonitorModes**.

— <target> : otx:StringTerm [0..1]

This optional element shall be used for locating the resource to which the trace shall be written. The target should be defined by a URI (see ISO 13209-2:—[2], Annex G). Other resource-location mechanisms may also be used.

Exceptions:

NotSupportedException

— In case a system without any addresses (e.g. DoIP or K-Line) is used and addresses are supplied by the addresses property, a **NotSupportedException** shall be thrown.

8.6.3.2 StopBusMonitoring

Semantic:

Stops the monitoring and the optional tracing on the BusMonitor that was started before by the StartBusMonitoring action.

Closing an uninitialised or already closed **BusMonitor** shall perform no operation and report no errors. It shall be for all effects a NOP.

NOTE The related MVCI system method is MCDMonitoringLink::stop().

Parameters:

— <monitor> : busMon:BusMonitorVariable [1..1]

This element represents the **BusMonitor** that shall be stopped. After stopping a bus monitor the variable is no longer initialised.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

2) Second edition under preparation. Stage at the time of publication: ISO/DIS 13209-2:2021.

© ISO 2021 – All rights reserved

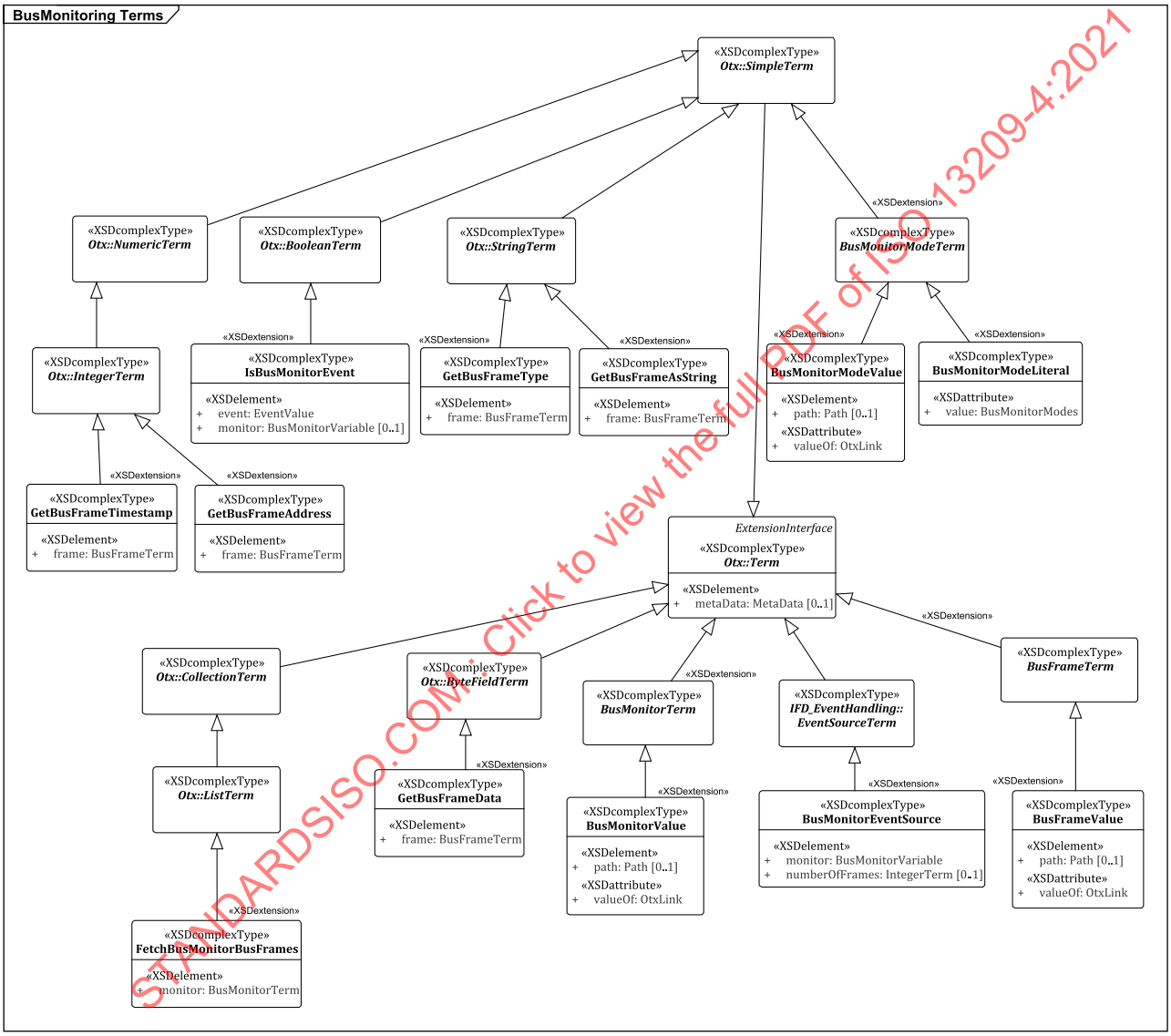
8.7 Terms

8.7.1 Terms overview

All terms introduced by this extension extend the **otx:Term** extension interface. Information about the specific super class of a term is provided in the individual term description clauses.

8.7.2 Terms syntax

Figure 14 shows the syntax of the **OTX BusMonitoring extension** related terms.



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 14 — BusMonitoring terms

8.7.3 Terms general semantics

8.7.3.1 BusFrameTerm

Semantic:

The abstract type BusFrameTerm serves as a base for all concrete terms that return a BusFrame. It has no special members.

© ISO 2021 – All rights reserved

8.7.3.2 BusFrameValue

Semantic:

Represents a frame value.

Return type:

busMon:BusFrameTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the **BusFrame** variable is not valid (no value was assigned to the variable before).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location that does not exist (like a list index exceeding list length, or a map key which is not part of the map).

8.7.3.3 BusMonitorEventSource

Semantic:

The **BusMonitorEventSource** term accepts a **BusMonitor** object that is to be made an event source. This term enables an OTX sequence to use a **BusMonitor** as a source for events in the context of the OTX EventHandling extension. A **BusMonitor** shall trigger an event every time the number of frames in the buffer reaches the number in the numberOfFrames property.

NOTE 1 The related MVCI system operations are MCDMonitoringLink::setNoOfSampleToFireEvent and MCDEventHandler::onMonitoringFramesReady.

NOTE 2 In case of BusMonitoringMode == TRACING, BusMonitorEventSource will never fire an event.

Return type:

event:EventSourceTerm

Parameters:

— <monitor> : busMon:BusMonitorVariable [1..1]

Represents the **BusMonitorVariable** which can cause the event.

— <numberOfFrames> : otx:IntegerTerm [0..1]

Represents a number of frames which have to be collected in the buffer to fire an event. If the internal buffer is less than the number of frames, the event is fired, when the buffer is filled. Default value is 1.

Exceptions:

otx:TypeMismatchException

— This exception is thrown if the specified event has not been raised by a BusMonitorEventSource.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

otx:InvalidReferenceException

— This exception is thrown if the variable is uninitialised.

8.7.3.4 BusMonitorModeLiteral

Semantic:

This term shall return a **BusMonitorMode** value from a hard-coded literal.

Return type:

busMon:BusMonitorModeTerm

Parameters:

— value : busMon:BusMonitorModes [1..1]

This attribute shall contain one of the values defined in the **BusMonitorModes** enumeration.

Exceptions:

— None

8.7.3.5 BusMonitorModeTerm

Semantic:

The abstract type **BusMonitorModeTerm** is an **otx:Term.** It serves as a base for all concrete terms that return a **BusMonitorMode** value. It has no special members.

8.7.3.6 BusMonitorModeValue

Semantic:

Returns the **BusMonitorMode** stored in a **BusMonitorMode** variable.

Return type:

busMon:BusMonitorModeTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location, which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

8.7.3.7 BusMonitorTerm

Semantic:

The abstract type **BusMonitorTerm** serves as a base for all concrete terms which returns a **BusMonitor** (e.g. CAN for monitoring). It has no special members.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

8.7.3.8 BusMonitorValue

Semantic:

Represents a **BusMonitor** value.

Return type:

busMon:BusMonitorTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the **BusFrame** variable is not valid (no value was assigned to the variable before).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

8.7.3.9 FetchBusMonitorBusFrames

Semantic:

The **FetchBusMonitorBusFrames** term accepts a **BusMonitor** and returns a list of **BusFrame**. All available frames will be returned and removed from buffer.

NOTE 1 The related MVCI system method is MCDMonitorLink::fetchMonitoringFrames(0).

NOTE 2 In case of BusMonitoringMode == TRACING, FetchBusMonitorBusFrames returns an empty list.

Return type:

otx:ListTerm

Parameters:

— <monitor> : busMon:BusMonitorTerm [1..1]

Represents a **BusMonitor** to get the frames from.

Exceptions:

— None

8.7.3.10 GetBusFrameAddress

Semantic:

The **GetBusFrameAddress** term accepts an **BusFrame** and returns an **otx:Integer** that represents the frame address [e.g. in case of CAN bus the 8 address bytes are converted (two's complement and little endian)].

Return type:

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

otx:IntegerTerm

Parameters:

— <frame> : busMon:BusFrameTerm [1..1]

Represents the **BusFrame** to get the address from.

Exceptions:

NotSupportedException

— If the frame address cannot be obtained a NotSupportedException shall be thrown.

8.7.3.11 GetBusFrameAsString

Semantic:

The **GetBusFrameAsString** term accepts a **BusFrame** and returns an **otx:String** that represents the whole frame delivered by the runtime system.

IMPORTANT — The Format of the returned string depends on the underlying runtime System.

NOTE The returned string can be formatted as the message string given in ISO 22900-3:2012, Annex F. Please note that this annex is informative and not normative, e.g. the separators between the single data components are not specified.

Return type:

otx:StringTerm

Parameters:

— <frame> : busMon:BusFrameTerm [1..1]

Represents the **BusFrame** to get the string from.

Exceptions:

NotSupportedException

— If frame string value cannot be obtained a NotSupportedException shall be thrown.

8.7.3.12 GetBusFrameData

Semantic:

The **GetBusFrameData** term accepts an **BusFrame** and returns an **otx:ByteField** that represents the frame communication data.

IMPORTANT — The returned ByteField shall be formatted as the data string of a message given in ISO 22900-3:2012, Annex F.

Return type:

otx:ByteFieldTerm

Parameters:

— <frame> : busMon:BusFrameTerm [1..1]

Represents the **BusFrame** to get the the data from, e.g. because the frame data are not formatted as it is given in ISO 22900-3:2012, Annex F.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Exceptions:

NotSupportedException

— If frame data cannot be obtained a NotSupportedException shall be thrown.

8.7.3.13 GetBusFrameTimestamp

Semantic:

The **GetBusFrameTimestamp** term accepts an **BusFrame** and returns an **otx:Integer** that represents the frame time stamp in nano seconds.

NOTE The timestamp is only useful for putting messages into chronological relation to each other.

Return type:

otx:IntegerTerm

Parameters:

— <frame> : busMon:BusFrameTerm [1..1]

Represents the **BusFrame** to get the timestamp from.

Exceptions:

NotSupportedException

— If frame time stamp cannot be obtained a **NotSupportedException** shall be thrown.

8.7.3.14 GetBusFrameType

Semantic:

The **GetBusFrameType** term accepts an **BusFrame** and returns an **otx:String** that represents the frame type (examples for CAN-specific message types are "RX", "TX", "ES", "EF", "EA" and "EC"). If the type cannot be obtained an empty string shall be returned.

IMPORTANT — The returned string shall be formatted as the type string of a message given in ISO 22900-3:2012, Annex F.

Return type:

otx:StringTerm

Parameters:

— <frame> : busMon:BusFrameTerm [1..1]

Represents the **BusFrame** to get the type from.

Exceptions:

— None

8.7.3.15 IsBusMonitorEvent

Semantic:

The **IsBusMonitorEvent** term accepts an EventValue term yielding an Event object that has been raised by the OTX runtime The term shall return true if and only if the Event originates from a **BusMonitorEventSource** term. In case an optional **BusMonitorVariable** is specified, the term

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

shall return true if and only if the Event was fired because that particular **BusMonitor** fired the event.

Return type:

otx:BooleanTerm

Parameters:

— <event> : event:EventValue [1..1]

Represents the Event whose type shall be tested.

— <monitor> : busMon:BusMonitorVariable [0..1]

Optionally specifies the particular **BusMonitor** which fired the event.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable is uninitialised.

9 OTX ComInterface extension

9.1 General

The **OTX ComInterface extension** enables handling vehicle interfaces at runtime and for example with which interface a **ComChannel** should be created. It is possible to get available interfaces through which communication to electrical control units (ECUs) in a vehicle can be started.

There are two major use cases for the need of handling interfaces at runtime:

— communication to ECUs over IP address (DoIP) as specified in the associated ISO specification,

— communication to ECUs in a vehicle that is temporarily connected to the test system because, for example a wireless LAN or bluetooth connection is used.

Additionally an action is introduced that enables Ethernet activation on the PDU-API for DoIP test sequences.

9.2 Datatypes

9.2.1 Datatypes overview

All data types introduced by this extension are derived from **otx:ComplexType** or **otx:SimpleType**.

9.2.2 Datatypes syntax

The syntax of the data-type declarations of the **OTX ComInterface extension** is shown in Figure 15.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

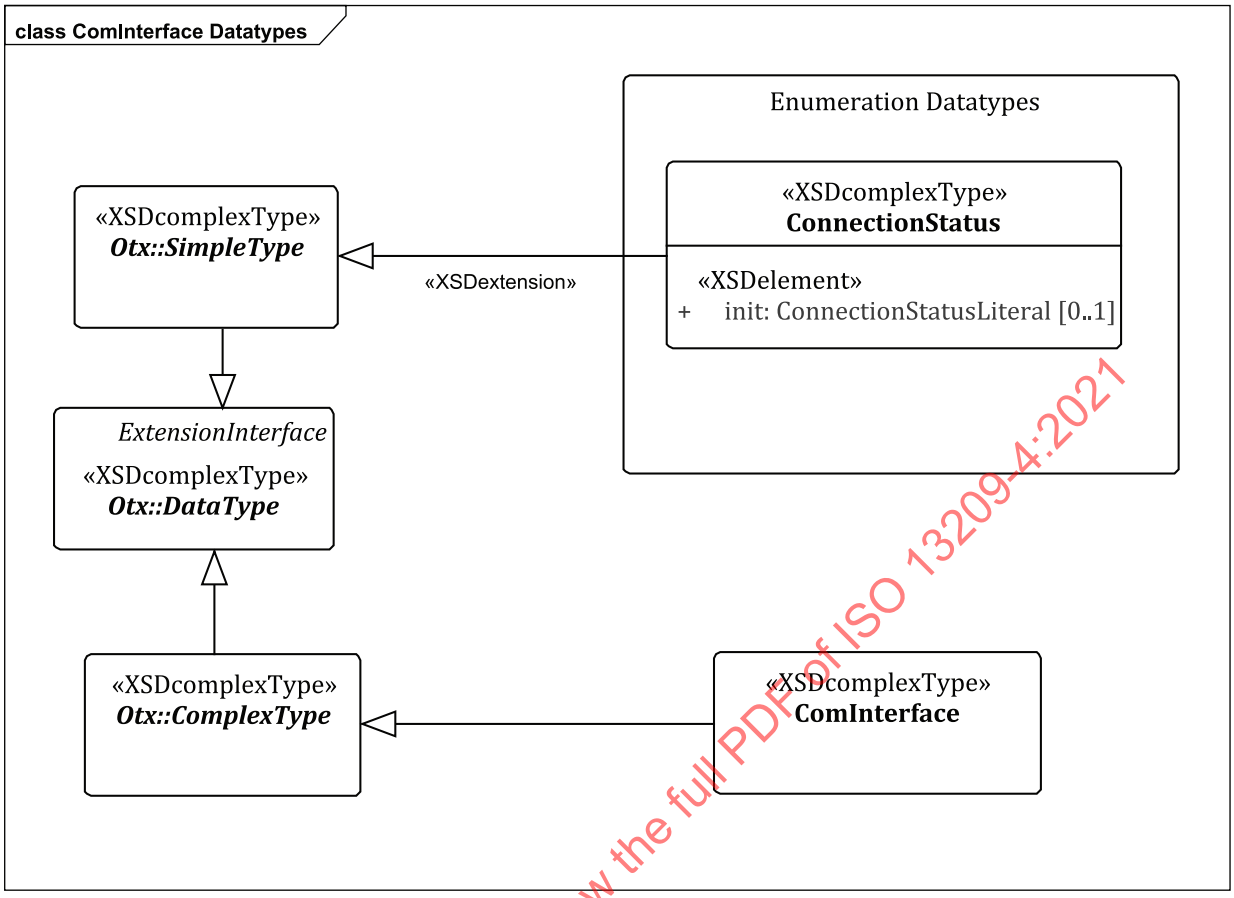


Figure 15 — ComInterface datatypes

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

9.2.3 Datatypes semantics

9.2.3.1 ComInterface

ComInterface is an **otx:ComplexType**.

Represents a handle of a communication interface [e.g. vehicle communication interface (VCI)].

9.2.3.2 ConnectionStatus

ConnectionStatus is an **otx:SimpleType.**

DataType for **ConnectionStates** enumeration.

Its members have the following semantics:

— <init> : comIf:ConnectionStatusLiteral[0..1]

This optional element sets the initial, hard-coded initialisation value of the identifier at declaration time.

9.3 Enumerations

9.3.1 Enumerations syntax

The syntax of the enumeration declarations of the **OTX ComInterface extension** is shown in Figure 16.

© ISO 2021 – All rights reserved

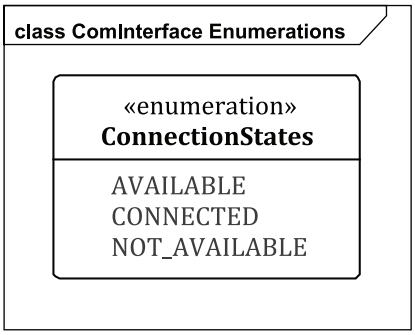


Figure 16 — ComInterface enumerations

9.3.2 Enumerations semantics

9.3.2.1 ConnectionStates

ConnectionStates represents the MCDInterfaceStatus enumerations that are mapped to the PDU module states.

IMPORTANT — If the ConnectionStatus declaration is not explicitly initialised (omitted <init> element), the default value shall be Current.

IMPORTANT — When applying otx:ToString on an ConnectionStatus, the resulting string shall be the name of the enumeration value, e.g. otx:ToString(CONNECTED)=" CONNECTED". Furthermore, applying otx:ToInteger shall return the index of the value in the ConnectionStates enumeration (smallest index is 0). The behaviour is undefined for other conversion terms (see ISO 13209-2).

IMPORTANT — ConnectionStatus values may occur as operands of comparisons (see ISO 13209-2:—, 7.15.8). For this case, the following order relation shall apply: NOT_AVAILABLE < AVAILABLE < CONNECTED .

ConnectionStates has the following enumeration values:

— AVAILABLE

The interfaces which are available in the underlying system (case of MVCI system: MCDInterfaceStatus = NOT_READY or AVAILABLE).

— CONNECTED

The interfaces which are connected to the underlying system (case of MVCI system: MCDInterfaceStatus = READY).

— NOT_AVAILABLE

The interfaces which are not available in the underlying system (case of MVCI system: MCDInterfaceStatus = NOT_AVAILABLE).

9.4 Exceptions

9.4.1 Exceptions overview

All exceptions introduced by this extension are derived from the **otx:Exception** type. They represent the full set of exceptions added by the **OTX ComInterface extension**.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

9.4.2 Exceptions syntax

The syntax of all **OTX ComInterface extension** exception-type declarations is shown in Figure 17.

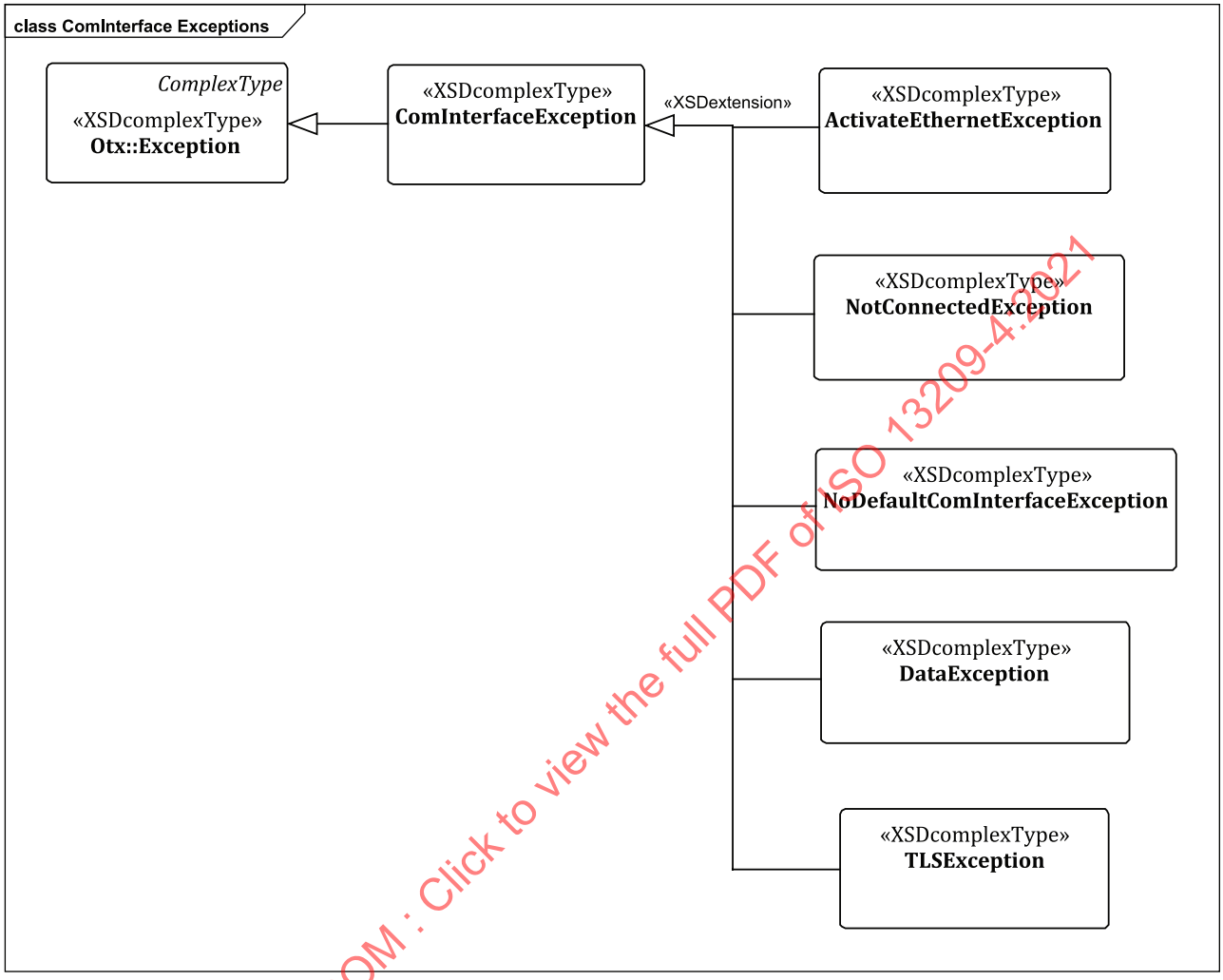


Figure 17 — ComInterface exceptions

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

9.4.3 Exceptions semantics

9.4.3.1 General

Since all **ComInterface** exception types are implicit exceptions without initialisation parts, they cannot be declared constant.

9.4.3.2 ActivateEthernetException

ActivateEthernetException (based on ComInterfaceException) is thrown if EthernetActivation is not supported by the selected **ComInterface**.

9.4.3.3 ComInterfaceException

A **ComInterfaceException** shall be used in case the more specific exception types described in the remainder of this section do not apply to the problem at hand.

IMPORTANT — All terms and action realisations in this extension may potentially throw this exception.

© ISO 2021 – All rights reserved

9.4.3.4 DataException

DataException (based on ComInterfaceException) is thrown if the ignition state or the battery voltage cannot be determined.

9.4.3.5 NoDefaultComInterfaceException

NoDefaultComInterfaceException (based on ComInterfaceException) is thrown if no default **ComInterface** is available.

9.4.3.6 NotConnectedException

NotConnectedException (based on ComInterfaceException) is thrown if the **ComInterface** is not connected.

9.4.3.7 TLSException

TLSException (based on ComInterfaceException) is thrown if setting the TLS certificate failed or the DoIP session mode cannot be determined.

9.5 Variable access

9.5.1 Variable-access overview

All variable-access types introduced by this extension are derived from the OTX core variable extension interface.

9.5.2 Variable-access syntax

The syntax of the **OTX ComInterface extension's** variable-access types is shown in Figure 18.

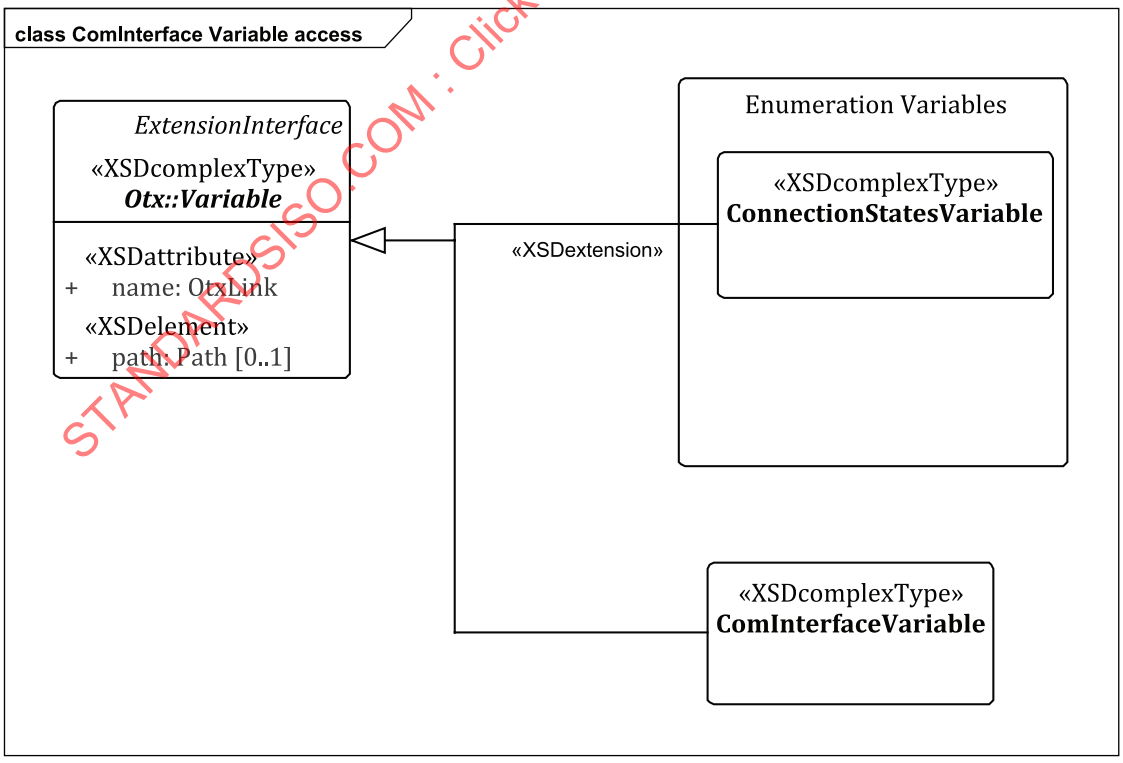


Figure 18 — ComInterface variable access

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

9.5.3 Variable-access semantics

The general semantics for all variable-access types shall apply. Please refer to ISO 13209-2 for details.

9.6 Actions

9.6.1 Actions overview

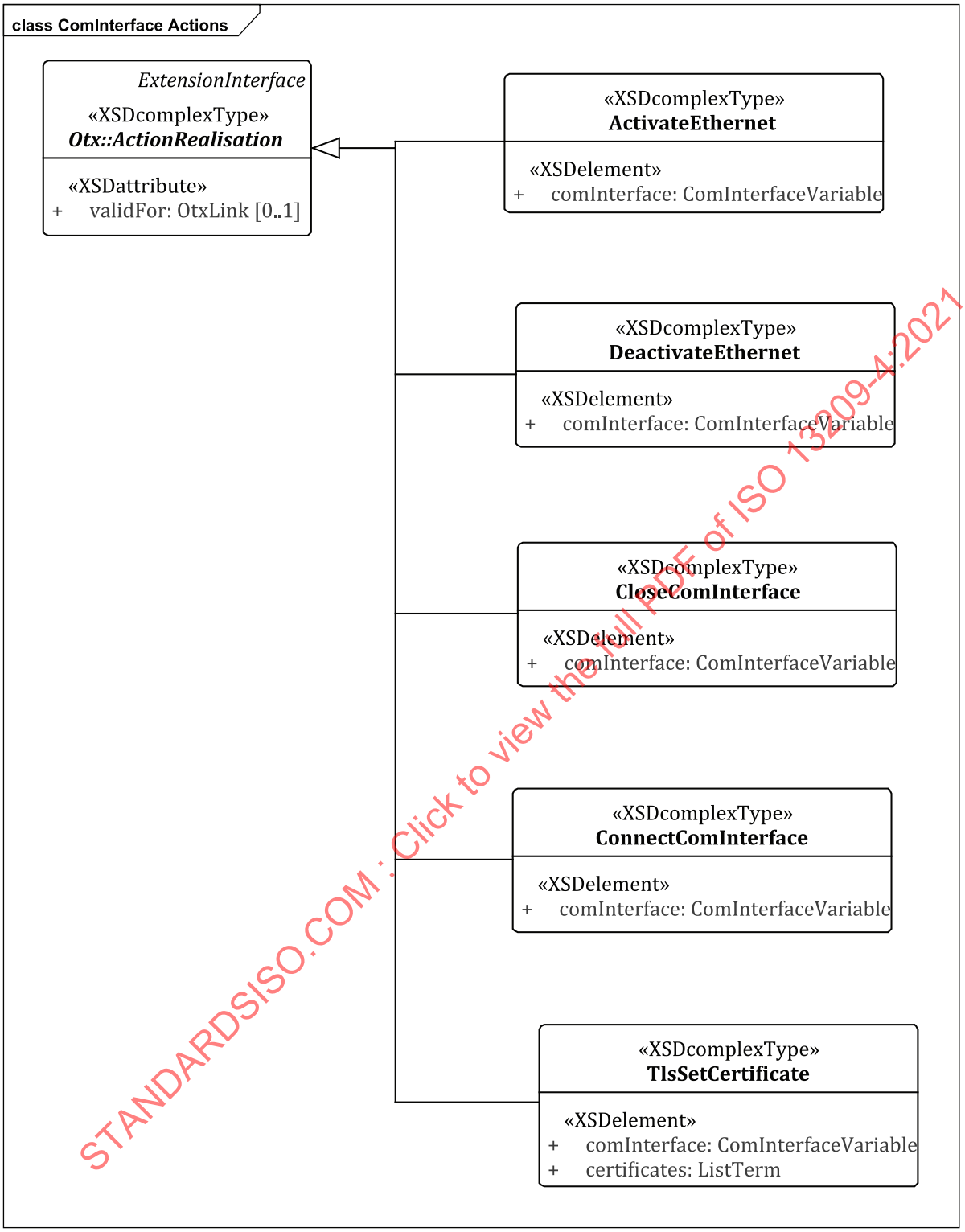
All actions introduced by this extension extend the **otx:ActionRealisation** extension interface.

9.6.2 Actions syntax

Figure 19 shows the syntax of all actions in the **OTX ComInterface extension**.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 19 — ComInterface actions

9.6.3 Actions general semantics

9.6.3.1 ActivateEthernet

Semantic:

This action activates the Ethernet communication on the selected interface.

© ISO 2021 – All rights reserved

NOTE The appropriate MVCI system operation is MCDInterface::execIOCtrl(PDU_IOCTL_SET_ETH_SWITCH_STATE).

Parameters:

— <comInterface> : comIf:ComInterfaceVariable [1..1]

Represents the handle of the communication interface on which the Ethernet should be activated.

Exceptions:

ActivateEthernetException

— This exception is thrown if, for example, EthernetActivation is not supported by the selected **ComInterface**.

InvalidReferenceException

— This exception is thrown if the **ComInterface** variable is uninitialised, or if the interface has been freed by a **CloseComInterface** action.

9.6.3.2 CloseComInterface

Semantic:

The **CloseComInterface** action is called when the diagnostic application has finished its diagnostic communication via a selected interface. This action closes the given interface regardless of whether it is the default interface or not.

CloseComInterface has no influence on the Ethernet activation state!

Please note that the use of the **CloseComInterface** action by an OTX sequence only indicates that the interface is not needed any more – it is up to the implementation of a specific runtime system whether it actually frees all resources and closes the interface at this point or not.

Closing an uninitialised or already closed **ComInterface** shall perform no operation and report no errors. It shall be for all effects a NOP.

NOTE The appropriate MVCI system operation is MCDInterface::disconnect().

Parameters:

— <comInterface> : comIf:ComInterfaceVariable [1..1]

Represents the handle of the communication interface which should be closed.

Exceptions:

— None

9.6.3.3 ConnectComInterface

Semantic:

The **ConnectComInterface** action establishes a connection to an interface for diagnostic communication. It can be used, for example to connect to an available **ComInterface** that is not already connected or to reconnect to an **ComInterface** if the connection to it is lost. If the **ComInterface** is already connected, nothing should happen and the **ComInterface** shall stay connected.

NOTE The appropriate MVCI system operation is MCDInterface::connect().

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <comInterface> : comIf:ComInterfaceVariable [1..1]

Represents the handle of the communication interface which should be connected.

Exceptions:

NotConnectedException

— This exception is thrown if **ComInterface** cannot be connected for any reasons.

InvalidReferenceException

— This exception is thrown if the **ComInterface** variable is uninitialised, or if the interface has been freed by a **CloseComInterface** action.

9.6.3.4 DeactivateEthernet

Semantic:

This action deactivates the Ethernet communication on the selected interface.

NOTE The appropriate MVCI system operation is MCDInterface::execIOCtrl(PDU_IOCTL_SET_ETH_SWITCH_STATE).

Parameters:

— <comInterface> : comIf:ComInterfaceVariable [1..1]

Represents the handle of the communication interface on which the Ethernet should be deactivated.

Exceptions:

ActivateEthernetException

— This exception is thrown if EthernetActivation is not supported by the selected **ComInterface**.

InvalidReferenceException

— This exception is thrown if the **ComInterface** variable is uninitialised, or if the interface has been freed by a **CloseComInterface** action.

9.6.3.5 TlsSetCertificate

Semantic:

For the server (ECU) based TLS authentication, at least the availability of the root certificate of the CA which issued the DoIP entity certificate shall be required to check the authenticity of the DoIP entity. This action can be used to set one or more certificate chains to the D-PDU API.

NOTE 1 The appropriate MVCI system operation is MCDInterface::execIOCtrl(PDU_IOCTL_TLS_SET_CERTIFICATE).

NOTE 2 In case of secured DoIP communication, it is necessary to invoke the action before the PDUConnect() for the ComLogicalLink on the related MVCI module.

NOTE 3 For connected ComLogicalLinks, calling this action or changing any other TLS related ComParam will have no effect on the ComLogicalLink state. Any changed security information will be applied only when the ComLogicalLink is connected with PDUConnect().

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <comInterface> : comIf:ComInterfaceVariable [1..1]

Represents the handle of the communication interface on which the certificate should be set.

— <certificates> : otx:ListTerm [1..1]

A list of list of ByteField represents a Buffer holding the data formatted with DER certificate encoding of all certificate chains.

Exceptions:

TLSException

— If setting the TLS certificate failed. In the case of an MVCI system, this happens if the return value of the related IOCtrl is one of the following values:

— PDU_ERR_PDUAPI_NOT_CONSTRUCTED

D-PDU API has not been constructed before

— PDU_ERR_COMM_PC_TO_VCI_FAILED

Communication between host and MVCI protocol module failed

— PDU_ERR_INVALID_HANDLE

Invalid MVCI protocol module handle or ComLogicalLink handle

— PDU_ERR_MODULE_NOT_CONNECTED

MVCI protocol module has not been connected

— PDU_ERR_FCT_FAILED

Function call failed

9.7 Terms

9.7.1 Terms overview

All terms introduced by this extension extend the **otx:Term** extension interface. Information about the specific super class of a term is provided in the individual term description clauses.

9.7.2 Terms syntax

Figure 20 shows the syntax of the **OTX ComInterface extension** related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

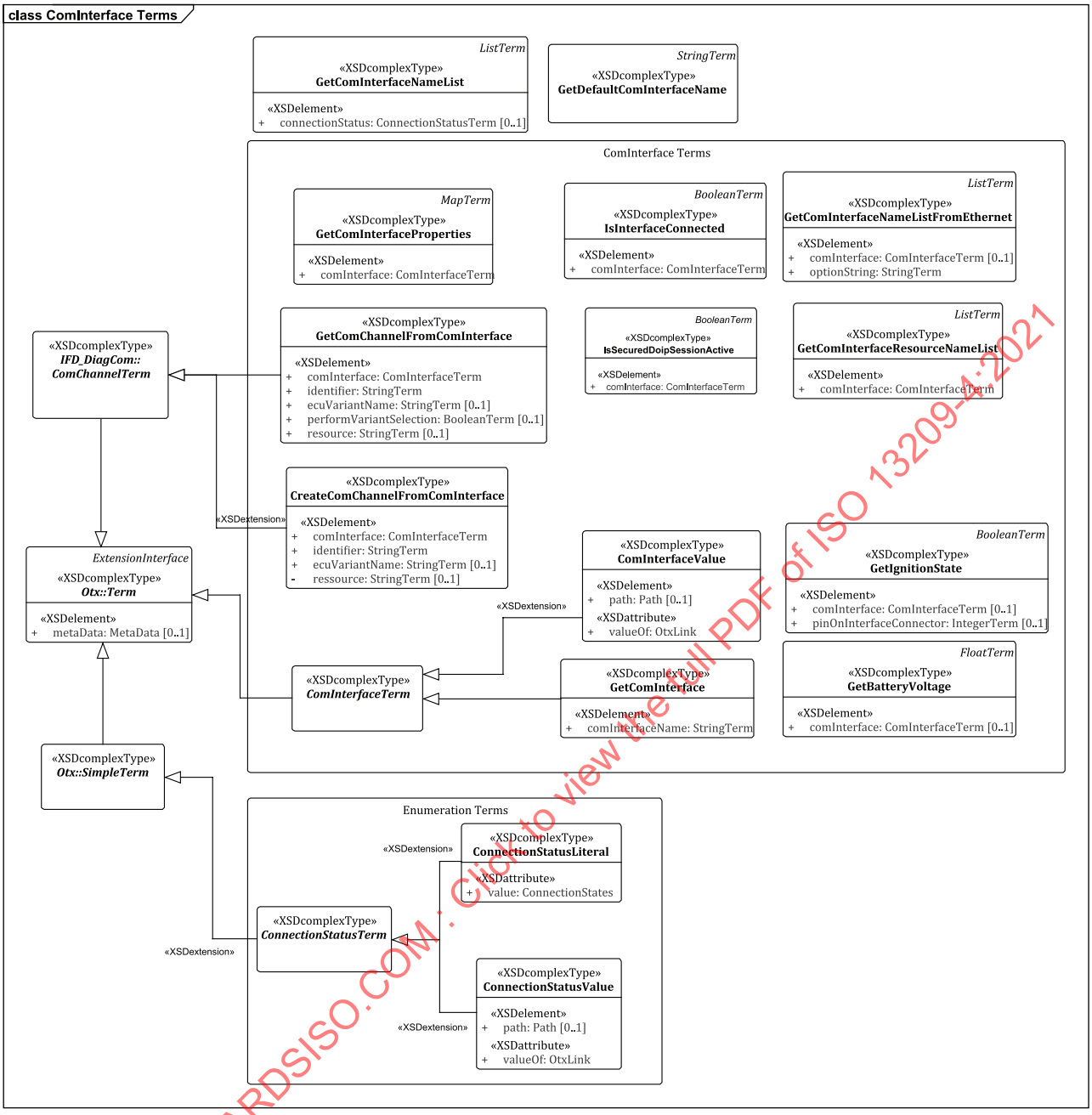


Figure 20 — ComInterface terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

9.7.3 Semantics for ComInterface terms

9.7.3.1 ComInterfaceTerm

Semantic:

The handle for the **ComInterface** that encapsulates the related **ComInterface** object.

9.7.3.2 ComInterfaceValue

Semantic:

Returns the **ComInterfaceValue** stored in a **ComInterface** variable.

© ISO 2021 – All rights reserved

Return type:

comIf:ComInterfaceTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the **ComInterface** variable is not valid (no value was assigned to the variable before).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location that does not exist (like a list index exceeding list length, or a map key which is not part of the map).

9.7.3.3 CreateComChannelFromComInterface

Semantic:

This term shall create a communication channel to an ECU on the specified communication interface. The term is needed, because the **comIf:GetComChannelFromInterface** term does not guarantee that the communication is not started. For exchangeability the usage of this term is recommended instead of **comIf:GetComChannelFromInterface,** if the user wants to set communication parameters before starting communication.

IMPORTANT — The channel is created at the time this term is executed.

IMPORTANT — If the ComChannel already exists, the state of the diag:ComChannel shall not be changed by this term.

IMPORTANT — In case an ODX/MVCI system is used, a StartCommunication on the logical link is not executed. In this case the logical link state is ONLINE.

The term CreateComChannelFromInterface shall always return a handle to the same ComChannel for a given ECU. It is possible to manually control the lifecycle of a ComChannel object by closing a ComChannel handle using the **diag:CloseComChannel** action. This is up to the author of a diagnostic sequence, an OTX runtime system is expected to clean up open ComChannel handles at the end of a diagnostic session.

NOTE The appropriate MVCI system operation is MCDProject::createLogicalLinkByNameAndInterface or MCDProject::createLogicalLinkByVariantAndInterface.

Return type:

diag:ComChannelTerm

Parameters:

— <comInterface> : comIf:ComInterfaceTerm [1..1]

This element identifies the communication interface.

— <identifier> : otx:StringTerm [1..1]

This element represents a string identifying the communication channel, which shall be created.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

NOTE In case an MVCI/ODX system is used, the identifier specifies the SHORT-NAME of the MCDLogicalLink to be used for communication.

— <ecuVariantName> : otx:StringTerm [0..1]

This optional element allows an OTX sequence to explicitly specify a particular ECU variant that the ComChannel shall be associated with.

It is provided in addition to the identifier attribute based on the assumption that the ComChannel identifier specifies a connection to a base variant of an ECU.

The <ecuVariantName> element can be used to directly create a ComChannel to a specific ECU variant without needing to perform the ECU variant identification step.

NOTE In case an MVCI/ODX system is used, the <ecuVariantName> element specifies the SHORT-NAME of the MCDDbEcuVariant to be associated with the logical link.

— <ressource> : otx:StringTerm [0..1]

The InterfaceResource as String that shall be used for physical communication.

NOTE The related MVCI system method is MCDProject::createLogicalLinkByNameAndInterfaceResource() if the resource parameter is defined.

MCDProject::createLogicalLinkByVariantAndInterfaceResource() has to be called if the EcuVariantName parameter and the Resource parameter are defined.

Default: the standard resource selected by runtime system.

Exceptions:

diag:LossOfComException

— This exception is thrown if communication to the ECU was interrupted during the variant identification process.

diag:UnknownTargetException

— This exception is thrown if the **diag:ComChannel** identifier provided by the <identifer> element does not exist or is invalid.

9.7.3.4 GetBatteryVoltage

Semantic:

It shall return the battery voltage of an interface (as measured in Volts) as a Float value.

NOTE The appropriate MVCI system operation is MCDInterface::getBatteryVoltage() if a **ComInterface** was given. If no **ComInterface** was given, the appropriate MVCI system operation is MCDProject::execIOCtrl ("PDU_IOCTL_READ_VBATT").

Return type:

otx:FloatTerm

Parameters:

— <comInterface> : comIf:ComInterfaceTerm [0..1]

This element identifies the communication interface.

Exceptions:

DataException

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

— This exception is thrown if the battery voltage cannot be determined.

NotConnectedException

— This exception is thrown if a **ComInterface** was given and the interface is not connected.

9.7.3.5 GetComChannelFromComInterface

Semantic:

This term shall return a communication channel from the **ComInterface**. It depends on the implementation of the OTX runtime system when the channel is actually created by the communication layer. There are three possible scenarios: the channel is created at the time this term is executed.The channel already exists; no additional action is carried out by the execution of this term. The channel is created when it is first needed for actual diagnostic communication.

No matter which approach is chosen, the term **GetComChannelFromInterface** shall always return a handle to the same **ComChannel** for a given **ComInterface**. It is possible to manually control the lifecycle of a **ComChannel** object by closing a **ComChannel** handle using the **CloseComChannel** action. This is up to the author of a diagnostic sequence, an OTX runtime system is expected to clean up open **ComChannel** handles at the end of a diagnostic session. The OTX runtime shall perform an ECU variant selection after opening of the channel if the term given by the optional element performVariantSelection yields true. This implies that when the next action on a communication channel is performed, the runtime system has identified the variant of the ECU actually present at runtime and configured the **ComChannel** accordingly. In case both an ecuVariantName is provided and performVariantSelection yields true, the channel is created to point at the desired ECU variant and variant selection is performed on the link afterwards. The variant identification functionality also exists as a separate action.

Return type:

diag:ComChannelTerm

Parameters:

— <comInterface> : comIf:ComInterfaceTerm [1..1]

This element identifies the communication interface.

— <identifier> : otx:StringTerm [1..1]

This element represents a string identifying the communication channel, which shall be created.

NOTE 1 In case an MVCI/ODX system is used, the identifier specifies the SHORT-NAME of the MCDLogicalLink to be used for communication.

The appropriate MVCI system operation MCDProject::createLogicalLinkByNameAndInterface() has to be called if all optional parameters of this term are not set.

— <ecuVariantName> : otx:StringTerm [0..1]

This optional element allows an OTX sequence to explicitly specify a particular ECU variant that the **ComChannel** shall be associated with.

It is provided in addition to the identifier attribute based on the assumption that the **ComChannel** identifier specifies a connection to a base variant of an ECU, the precise variant of which then can be implicitly or explicitly identified by the diagnostic application (compare the performVariantSelection element in this clause and the IdentifyAndSelectVariant action).

The ecuVariantName element can be used to directly create a **ComChannel** to a specific ECU variant without needing to perform the ECU variant identification step.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

NOTE 2 In case an MVCI/ODX system is used, the ecuVariantName element specifies the SHORT-NAME of the MCDDbEcuVariant to be associated with the logical link.

If the ecuVariantName element is defined, the PerformVariantSelection element shall be omitted or set to false.

The related MVCI system methods MCDProject::createLogicalLinkByVariantAndInterface() has to be called if only the ecuVariantName parameter is defined.

MCDProject::createLogicalLinkByVariantAndInterfaceResource() has to be called if the ecuVariantName parameter and the Resource parameter are defined.

Default: If this parameter is not transmitted, no variant selection takes place.

— <performVariantSelection> : otx:BooleanTerm [0..1]

This optional element can be used by the OTX author to control whether an implicit variant selection shall be performed. If performVariantSelection yields true at runtime, the variant selection is done automatically after the **ComChannel** is created. This document cannot make assumptions about whether the vehicle communication component used by an OTX runtime supports the concept of ECU variant identification or about the behaviour of the communication component in case it does. The relevant parts of the OTX DiagCom standard are based on the following assumptions.

— A communication channel to an **ComInterface** is associated with a data set describing diagnostic behaviour of a specific variant of that ECU.

— The vehicle communication component is able to explicitly perform an ECU variant identification operation on a communication channel to an ECU.

— The required logic and data for performing the variant identification is intrinsic to the vehicle communication component, i.e. there is no additional external information required for the communication component to perform the ECU variant identification.

— After an ECU variant has been identified, the vehicle communication component is able to explicitly associate the communication channel to that ECU with the specific data set for that ECU variant, effectively switching the communication channel from the old variant data set to a new one.

— The IdentifyAndSelectVariant action tells the runtime system to perform the variant identification operation on the provided communication channel and then switch the data set associated with that channel to the one fitting the newly identified variant (if any).

— The **diag:GetComChannel** term tells the runtime system to create a new communication channel, immediately perform the variant identification operation on the new communication channel and then switch the data set associated with that channel to the one fitting the newly identified variant (if any).

NOTE 3 In case an ODX/MVCI system is used, the exact semantics of Variant Identification and Selection are specified by the ISO ODX and MVCI standards.

Default: false

If the PerformVariantSelection element is set to true, the EcuVariantName element shall be omitted.

The related MVCI system methods MCDProject::createLogicalLinkByNameAndInterface() has to be called if all optional parameters of this term are not set.

MCDProject::createLogicalLinkByNameAndInterfaceResource() has to be called if only the Resource parameter is defined.

MCDLogicalLink::gotoOnline()

MCDLogicalLink::open()

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Control Primitive MCDStartCommunication

Control Primitive MCDVariantIdentificationAndSelection

— <resource> : otx:StringTerm [0..1]

The MVCI communication channel.

The related MVCI system methods MCDProject::createLogicalLinkByNameAndInterfaceResource() has to be called if only the Resource parameter is defined.

MCDProject::createLogicalLinkByVariantAndInterfaceResource() has to be called if the EcuVariantName parameter and the Resource parameter are defined.

Default: the standard resource selected by runtime system.

Exceptions:

LossOfComException

— This exception is thrown if communication to the ECU was interrupted during the variant identification process.

UnknownTargetException

— This exception is thrown if the **ComChannel** identifier provided by the identifer element does not exist or is invalid.

9.7.3.6 GetComInterface

Semantic:

Returns the **ComInterface** for the given name. **ComInterface** can only delivered for a **ComInterface**, which is included in the current interface list.

NOTE The appropriate MVCI system operation is MCDsystem::getCurrentInterfaces::getItemByName().

Return type:

comIf:ComInterfaceTerm

Parameters:

— <comInterfaceName> : otx:StringTerm [1..1]

The unique name (ID) of the interface which defines it.

Exceptions:

UnknownTargetException

— This exception is thrown if the **ComInterface** name provided by the ComInterfaceName element does not exist or is invalid.

9.7.3.7 GetComInterfaceNameListFromEthernet

Semantic:

This term performs a DoIP vehicle identification request and returns the unique names of the interfaces as a ListTerm of String. In case a **ComInterface** is given, the detection of the interfaces is done at the given module. Otherwise the detection is executed at the underlying system.

NOTE The appropriate MVCI system operation is:

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

In case of a given ComInterface: MCDInterface::detectInterfaces()

Otherwise: MCDSystem:detectInterfaces()

In both cases: MCDSystem:getCurrentInterfaces()

Return type:

otx:ListTerm

Parameters:

— <comInterface> : comIf:ComInterfaceTerm [0..1]

This optional element allows as input a **ComInterface** with an Ethernet activation pin on.

— <optionString> : otx:StringTerm [1..1]

The OptionString element sets the configuration of the base system that is needed to get interfaces. It is specified in the MCD specification. If no option is entered the configuration of the base system is used.

The MVCI server supports DoIP interface detection. The VCI access layer has been prepared by calling prepareVciAccessLayer. The input parameter "optionString" has the following format:

OptStringFmt ::= PDU_IOCTL='PDU_IOCTL_VEHICLE_ID_REQUEST' []

PSM ::= PreselectionMode=

PreselectionMode ::= 'None' | 'VIN' | 'EID' PSV ::= PreselectionValue=

PreselectionValue ::= 'string' // containing an optional VIN or an EID to preselect // detection of DoIP entities, depending on parameter // "PreselectionMode". This string is left empty if no // preselection is desired (PreselectionMode='None').

CM ::= CombinationMode= CombinationMode ::= 'DoIP-Entity' | 'DoIP-Vehicle' | 'DoIP-Group' | 'DoIP-Collection'

VDT::= VehicleDiscoveryTime= VehicleDiscoveryTime ::= 'unsigned short' // Time-out to wait for vehicle identification responses. // 0=return immediately, or time in milliseconds.

DAC ::= DestinationAddressCount= DestinationAddressCount ::= 'unsigned short' // No. of broadcast/multicast addresses in the destination // address list DAList. In case of an empty list, the // sequence is omitted.

DAList ::= (DA='' '')+ n ::= // current number starting with 1, maximum value // according to DestinationAddressCount IPv4 ::= // dottet decimal notation, e.g. 192.168.1.255 IPv6 ::= // hexadecimal, colon separated according to // RFC4291, section 2.2 sep ::= blank

Exceptions:

— None

9.7.3.8 GetComInterfaceProperties

Semantic:

Returns a map of interface properties as Map[String, String].

The Map contains the following information:

— In case of an MVCI server, the application determines the properties of a DoIP module by evaluating the LONG-NAME (the information's are contained in concatenation of pVendorModuleName and pVendorAdditionalInfo, see ISO 22900-3:2012, 8.27) of the corresponding MCDInterface object.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

The application adds each key value pair to the map. Only available values shall be added to the map. This is because battery voltage and clamp state are only available if the connection of the **ComInterface** is established or this functionality is supported by the VCI.

— In case of another **ComInterface** the key value pairs are supplier specific.

Additionally keys can be returned in either case.

Key	**Value**
Type	MVCI_ISO_13400_DoIP_Entity
EID	0A:1B:2C:3D:4E:5F
LA	4096
IP	192.168.1.255
VIN	ABCDEFG1234567XYZ
GroupID	12
ClampState	true
BatteryVoltage	12.4

GetComInterfaceResourceNameList or **GetComChannelFromComInterface** realises the connection to the COMInterface. After **GetComInterface** the connection can be realised, but is not required. The **GetComInterfaceProperties** are dependent on an available connection, this means in not connected mode less keys are contained.

Return type:

otx:MapTerm

Parameters:

— <comInterface> : comIf:ComInterfaceTerm [1..1]

Represents the handle of the communication interface of which the properties should be retrieved.

Exceptions:

— None

9.7.3.9 GetComInterfaceResourceNameList

Semantic:

Returns a list of ComInterfaceResource names, which are provided for this **ComInterface**. For example {“CAN1”,”CAN2”} (represents String[]).

NOTE The appropriate MVCI system operation is MCDInterface::getInterfaceResources().

Return type:

otx:ListTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Parameters:

— <comInterface> : comIf:ComInterfaceTerm [1..1]

Represents the handle of the communication interface for which the resource name list should be extracted.

Exceptions:

NotConnectedException

— This exception is thrown if the referenced **ComInterface** is not connected.

9.7.3.10 GetIgnitionState

Semantic:

It shall return the current state of the ignition clamp of an interface. It returns TRUE, if ignition state is ON otherwise FALSE.

NOTE The appropriate MVCI system operation is MCDInterface::getClampState(0, "IgnitionClamp") if a **ComInterface** was given. If no ComInterface was given, the appropriate MVCI system operation is MCDProject::getClampState("IgnitionClamp").

Return type:

otx:BooleanTerm

Parameters:

— <comInterface> : comIf:ComInterfaceTerm [0..1]

This element identifies the communication interface.

— <pinOnInterfaceConnector> : otx:IntegerTerm [0..1]

This element identifies the pin number on the ComInterface. For runtime systems which automatically detect the pin number (e.g. MVCI system), the parameter value should be ignored.

If the element is omitted, a default mechanism to determine the ignition state is used.

Exceptions:

DataException

— This exception is thrown if the ignition state cannot be determined.

NotConnectedException

— This exception is thrown if a **ComInterface** was given and the interface is not connected.

9.7.3.11 IsInterfaceConnected

Semantic:

Returns true if the **ConnectionStatus** is CONNECTED, otherwise false.

NOTE The related MVCI system method is MCDInterface::getStatus().

Return type:

otx:BooleanTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Parameters:

— <comInterface> : comIf:ComInterfaceTerm [1..1]

Represents the handle of the communication interface of which the connection state should be checked.

Exceptions:

— None

9.7.3.12 IsSecuredDoipSessionActive

Semantic:

Returns TRUE, if the secured DoIP session mode of the related **ComInterface** is active and FALSE, if no DoIP session or unsecured DoIP session is active.

An application may require to determine the mode of the currently established DoIP connection on the related **ComInterface**, e.g. insecure DoIP session or secured DoIP session (with TLS).

NOTE The appropriate MVCI system operation is MCDInterface::execIOCtrl(DOIP_GET_CURRENT_SESSION_MODE).

Return type:

otx:BooleanTerm

Parameters:

— <comInterface> : comIf:ComInterfaceTerm [1..1]

Represents the handle of the communication interface on which the DoIP session mode should be retrieved.

Exceptions:

TLSException

— If getting the DoIP session mode failed. In the case of an MVCI system, this happens if the return value of the related IOCtrl is one of the following values:

— PDU_ERR_PDUAPI_NOT_CONSTRUCTED

D-PDUAPI has not been constructed before

— PDU_ERR_COMM_PC_TO_VCI_FAILED

Communication between host and MVCI protocol module failed

— PDU_ERR_INVALID_HANDLE

Invalid MVCI protocol module Handle or ComLogicalLink handle

— PDU_ERR_MODULE_NOT_CONNECTED

MVCI protocol module has not been connected

— PDU_ERR_FCT_FAILED

Function call failed

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

9.7.4 Semantics for Enumeration terms

9.7.4.1 ConnectionStatusLiteral

Semantic:

Literal of **ConnectionStates** enumeration.

Return type:

comIf:ConnectionStatusTerm

Parameters:

— <value> : comIf:ConnectionStates [1..1]

This attribute shall contain one of the values defined in the **ConnectionStates** enumeration.

Exceptions:

— None

9.7.4.2 ConnectionStatusTerm

Semantic:

Contains one element of the **ConnectionStates** enumeration.

9.7.4.3 ConnectionStatusValue

Semantic:

Value of **ConnectionStates** enumeration.

Return type:

comIf:ConnectionStatusTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures like ListTerm or MapTerm. It is built out of a series of index- and name-steps which allow navigate into the structure:

<stepByIndex> : NumericTerm [1]

This step of a <path> shall be used for locations addressed by index. Items in a List and items in a Map with Integer keys shall be addressed in this way.

<stepByName> : StringTerm [1]

This step of a <path> shall be used for addressing locations by name. Items in a Map with String keys shall be addressed in this way.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

9.7.5 Terms general semantics

9.7.5.1 GetComInterfaceNameList

Semantic:

Returns the names (MCDInterface unique SHORT-NAME) of the **ComInterface** with the given state as a ListTerm of String. This call has no dependencies on the SelectProject and SelectVehicleInformation method. This call is independent of the selected project and selected vehicle information.

NOTE The related MVCI system methods are MCDSystem::prepareVciAccessLayer(); MCDSystem::getCurrentInterfaces().

Return type:

otx:ListTerm

Parameters:

— <connectionStatus> : comIf:ConnectionStatusTerm [0..1]

Represents the state value that filters the returned interfaces .

Exceptions:

— None

9.7.5.2 GetDefaultComInterfaceName

Semantic:

Returns the name (MCDInterface unique LONG-NAME) of the default **ComInterface** with the MCDInterfaceStatus READY state as a String. This call has no dependencies on the SelectProject and SelectVehicleInformation method. This call is independent of the selected project and selected vehicle information.

Return type:

otx:StringTerm

Parameters:

— None

Exceptions:

NoDefaultComInterfaceException

— This exception is thrown if no default **ComInterface** is available.

10 OTX CommonDialogs extension

10.1 General

The CommonDialogs extensions provide support for dialogs that have common use cases in OTX but were not included in the original specification.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

10.2 Exceptions

10.2.1 Exceptions overview

All exceptions introduced by this extension are derived from the **otx:Exception** type. They represent the full set of exceptions added by the **OTX CommonDialogs extension**.

10.2.2 Exceptions syntax

The syntax of all **OTX CommonDialogs extension** exception-type declarations is shown in Figure 21.

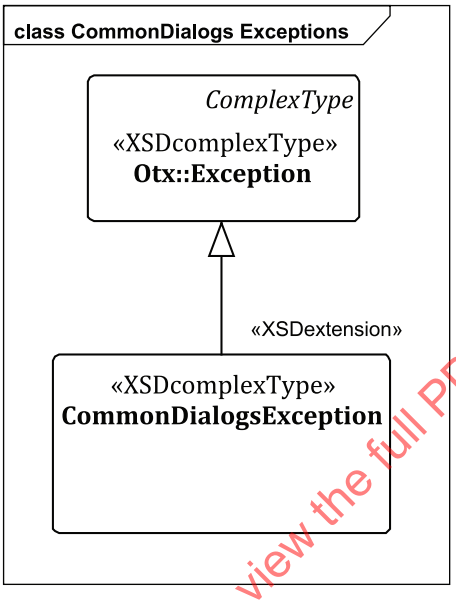


Figure 21 — CommonDialogs exceptions

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

10.2.3 Exceptions semantics

10.2.3.1 General

Since all **OTX CommonDialogs extension** exception types are implicit exceptions without initialisation parts, they cannot be declared constant.

10.2.3.2 CommonDialogsException

IMPORTANT — All terms and action realisations in this extension may potentially throw this exception.

10.3 Actions

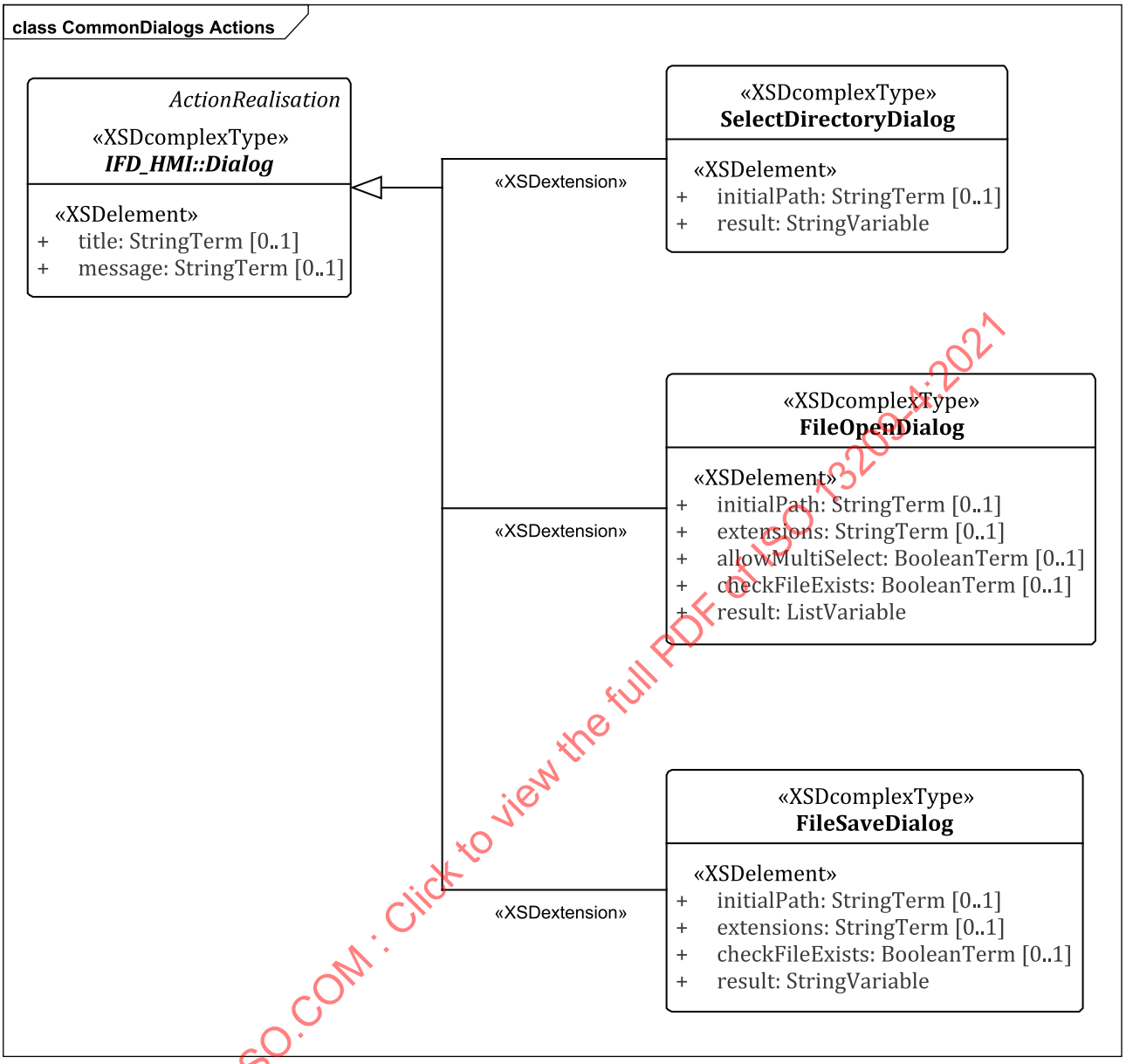
10.3.1 Actions overview

All actions introduced by this extension extend the **otx:ActionRealisation** extension interface.

10.3.2 Actions syntax

Figure 22 shows the syntax of all actions in the **OTX CommonDialogs extension**.

© ISO 2021 – All rights reserved



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 22 — CommonDialogs actions

10.3.3 Actions general semantics

10.3.3.1 FileOpenDialog

Semantic:

> Opens a (runtime-specific) file selection dialog. The user can select one or more files and click to accept the selection, or can close the dialog. The dialog can be configured using the optional elements to add a title filter message and modify behaviour (such as selection of single or multiple files).

Associated checker rules:

— CommonDialogs_Chk001 – type-safe FileOpenDialog.

© ISO 2021 – All rights reserved

Parameters:

— <initialPath> : otx:StringTerm [0..1]

The initial path to select.

If a directory is specified, and it does not exist, or the user does not have sufficient access rights, the behaviour shall be dictated by the runtime system.

— <extensions> : otx:StringTerm [0..1]

The optional (comma separated) list of file extensions to filter in this dialog.

— <allowMultiSelect> : otx:BooleanTerm [0..1]

The optional flag to determine if the dialog will allow the user to select more than one files.

— <checkFileExists> : otx:BooleanTerm [0..1]

The optional flag to determine if the dialog shall check for the file before allowing the user to complete.

— <result> : otx:ListVariable [1..1]

The selected file(s), returned as a list of strings.

If no file is selected, the result will be returned as the empty list.

— <message> : otx:StringTerm [0..1] (derived from Dialog)

The message to be displayed in the dialog to aid the user in their selection.

Some runtime systems may not be able to display a message in the dialog. If this is the case, the runtime system can choose to drop the message, or append it to the title.

— <title> : otx:StringTerm [0..1] (derived from Dialog)

The title to be displayed in the dialog.

Some runtime systems may not be able to display a title in the dialog. If this is the case, the runtime system can choose to drop the title, or append it to the message.

Exceptions:

— None

10.3.3.2 FileSaveDialog

Semantic:

Opens a (runtime-specific) file selection dialog. The user can select a file and click to accept the selection, or can close the dialog. The dialog can be configured using the optional elements to add a title filter message and modify behaviour.

Parameters:

— <initialPath> : otx:StringTerm [0..1]

The initial path to select.

If a directory is specified, and it does not exist, or the user does not have sufficient access rights, the behaviour shall be dictated by the runtime system.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— <extensions> : otx:StringTerm [0..1]

The optional (comma separated) list of file extensions to filter in this dialog.

— <checkFileExists> : otx:BooleanTerm [0..1]

The optional flag to determine if the dialog shall check for the file before allowing the user to complete.

— <result> : otx:StringVariable [1..1]

The selected file as a string.

If no file is selected, the result will be the empty string "".

— <message> : otx:StringTerm [0..1] (derived from Dialog)

The message to be displayed in the dialog to aid the user in their selection.

Some runtime systems may not be able to display a message in the dialog. If this is the case, the runtime system can choose to drop the message, or append it to the title.

— <title> : otx:StringTerm [0..1] (derived from Dialog)

The title to be displayed in the dialog.

Some runtime systems may not be able to display a title in the dialog. If this is the case, the runtime system can choose to drop the title, or append it to the message.

Exceptions:

— None

10.3.3.3 SelectDirectoryDialog

Semantic:

Opens a (runtime-specific) directory selection dialog. The user can select a directory and click to accept the selection, or can close the dialog. The user can configure parts of the dialog behaviour, such as a title and message.

Parameters:

— <initialPath> : otx:StringTerm [0..1]

The initial path to select.

If a directory is specified, and it does not exist, or the user does not have sufficient access rights, the behaviour shall be dictated by the runtime system.

— <result> : otx:StringVariable [1..1]

The selected directory.

If no directory is selected, the result will be the empty string "".

This dialog will not allow the user to select (or enter) a directory that does not exist.

— <message> : otx:StringTerm [0..1] (derived from Dialog)

The message to be displayed in the dialog to aid the user in their selection.

Some runtime systems may not be able to display a message in the dialog. If this is the case, the runtime system can choose to drop the message, or append it to the title.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— <title> : otx:StringTerm [0..1] (derived from Dialog)

The title to be displayed in the dialog.

Some runtime systems may not be able to display a title in the dialog. If this is the case, the runtime system can choose to drop the title, or append it to the message.

Exceptions:

— None

11 OTX DataType extension

11.1 General

The purpose of the OTX DataTypes extension is to support often required data types **Enumeration** and **Structure** as well as providing a location for resource information.

The inner structure of data types **Enumeration** and **Structure** have to be defined during authoring time via a signature. This approach has the advantage that a validation is possible at design time.

Identical enumerations and structures, which refer to the same signature are handled as identical data types.

The **ResourceLocation** data type defines a way to handle resources inside OTX which differ between different runtime environments.

11.2 Datatypes

11.2.1 Datatypes overview

All data types introduced by this extension are derived from **otx:ComplexType** or **otx:SimpleType**.

11.2.2 Datatypes syntax

The syntax of the data-type declarations of the **OTX DataType extension** is shown in Figure 23.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

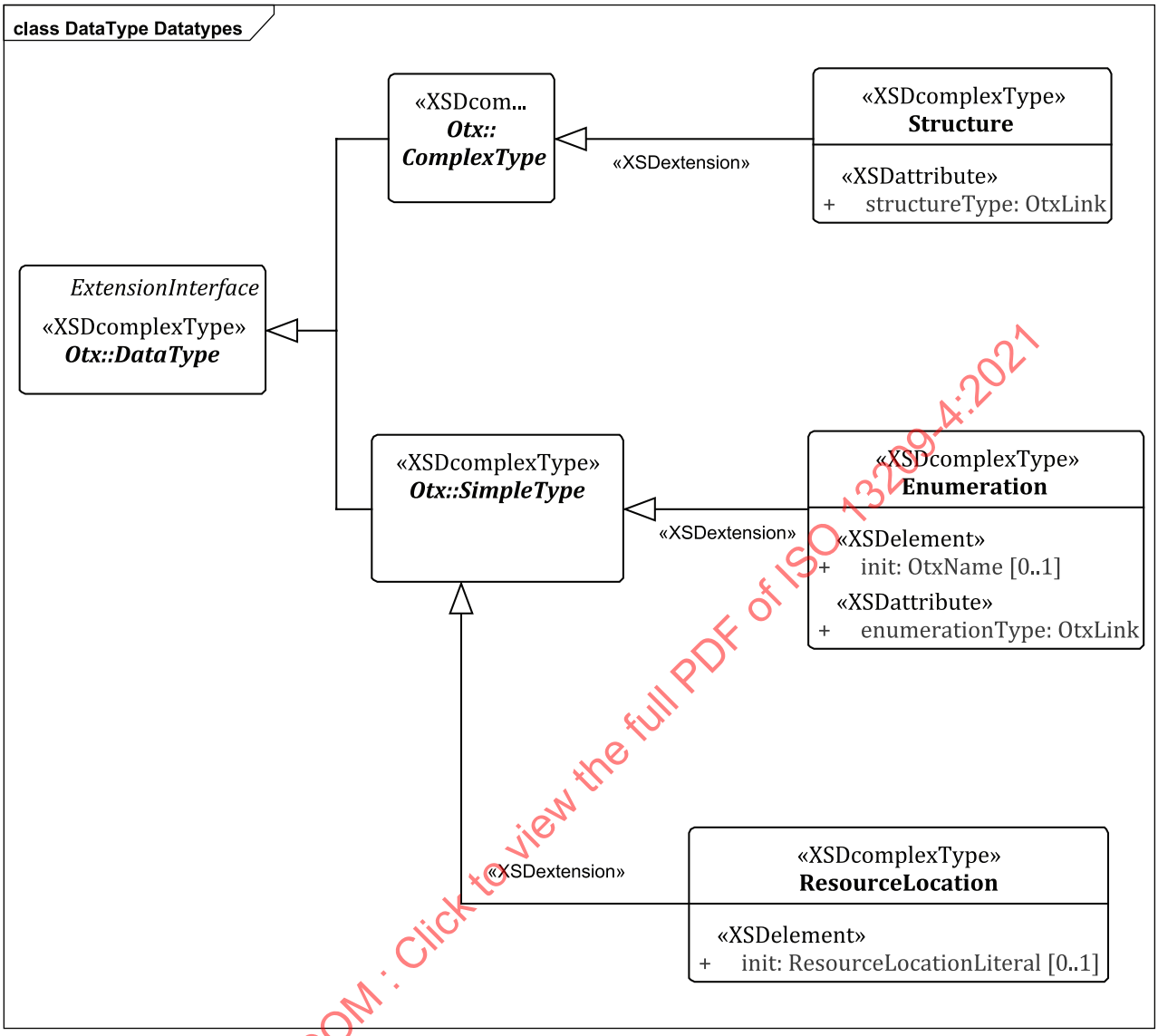


Figure 23 — DataType datatypes

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

11.2.3 Datatypes semantics

11.2.3.1 Enumeration

Enumeration is an **otx:SimpleType**.

An **Enumeration** consists of a set of named constants called enumeration elements. Each enumeration element assigns a unique name to a unique **otx:IntegerLiteral**, so that the name can be used as an **EnumerationLiteral** inside OTX instead of the integer value. To define the structure of an **Enumeration** the E**numerationSignature** is used. The simple term of the enumerator value is its integer value. Thus, for comparison terms, the integer value of the specific enumerator value is used.

IMPORTANT — If the OTX Core conversion term otx:ToString is applied to an Enumeration term, the element name shall be returned.

IMPORTANT — If the OTX Core conversion term otx:ToInteger is applied to an Enumeration term, the Integer value shall be returned.

IMPORTANT — An Enumeration value can be persisted, see OTX Persistence extension.

© ISO 2021 – All rights reserved

Its members have the following semantics:

— <init> : otx:OtxName[0..1]

Element name of the related enumeration. If not specified, the first enumeration element shall be used as initial value.

— <enumerationType> : otx:OtxLink[1..1]

Refers to a certain **EnumerationSignature**.

Associated checker rules:

— DataType_Chk002 – correct target for Enumeration,

— Core_Chk053 – no dangling OtxLink associations (see ISO 13209-2).

11.2.3.2 ResourceLocation

ResourceLocation is an **otx:SimpleType**.

In some use cases, an OTX sequence needs to address certain resources, e.g. XML files, text documents, pictures. If this is the case, each resource has to be addressable to a location. The location itself should not be part of the test logic, because it can be different in different runtime environments.

A **ResourceLocation** represents a string value which addresses a location of a resource associated with an ID. At runtime it can be mapped via an appropriate mapping mechanism to a real address. If the related resource ID cannot be mapped, because a possible mapping table does not contain the ID or the runtime does not support a mapping mechanism, the location is the default location (see ResourceLocationLiteral). If no default location is specified, the location is an empty string.

IMPORTANT — The mapping mechanism is not part of the standard.

IMPORTANT — When applied to a ResourceLocation, the otx:ToString term shall return the location of the ResourceLocation.

Its members have the following semantics:

— <init> : ResourceLocationLiteral[0..1]

This optional element represents the hard-coded value from which the declared resource shall be created. It includes an ID which uniquely identifies the resource and an optional default path, which is used if no path was mapped and an optional description of the resource type.

IMPORTANT — If the ResourceLocation declaration is not explicitly initialised (omitted <init> element), the default path shall be an empty string.

11.2.3.3 Structure

Structure is an **otx:ComplexType**.

A structure is a user-defined data type composed of elements that can have different data types. It is typically used to encapsulate small groups of related elements, such as the coordinates of a rectangle. Structures can contain elements of any OTX data type and also **Structure** itself in any depth.

To define a **Structure** the **StructureSignature** is used. A **Structure** itself has no literal and also has no init value.

To access structure elements only StepByName and not StepByIndex shall be used. Additionally, the StepByName element shall only contain StringLiteral terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

"Deep change-monitoring" shall detect changing of items in a structure of any depth. Any changes of structure elements that support the eventplus:IsDeepMonitorChangeEvent shall trigger an event as defined for the data type of the element.

For example, if a structure contains a structure, which contains a list and an element of this list was added, removed or changed an eventplus:DeepMonitorChangeEventSource shall fire an event.

Since Structure has no initialisation parts, a Structure cannot be declared constant.

IMPORTANT — A Structure value can be persisted if all contained elements can be persisted, see Persistence extension.

Its members have the following semantics:

— <structureType> : otx:OtxLink[1..1]

  Refers to a certain **StructureSignature**.

Associated checker rules:

— DataType_Chk001 – Accessing **structure** elements.

11.3 Exceptions

11.3.1 Exceptions overview

All exceptions introduced by this extension are derived from the **otx:Exception** type. They represent the full set of exceptions added by the **OTX DataType extension**.

11.3.2 Exceptions syntax

The syntax of all **OTX DataType extension** exception-type declarations is shown in Figure 24.

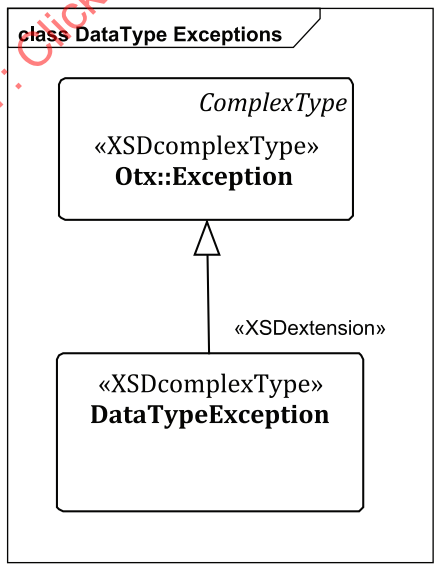


Figure 24 — DataType exceptions

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

11.3.3 Exceptions semantics

11.3.3.1 General

Since all **OTX DataType extension** exception types are implicit exceptions without initialisation parts, they cannot be declared constant.

11.3.3.2 DataTypeException

IMPORTANT — All terms and action realisations in this extension may potentially throw this exception.

11.4 Variable access

11.4.1 Variable-access overview

All variable-access types introduced by this extension are derived from the OTX core variable extension interface.

11.4.2 Variable-access syntax

The syntax of the **OTX DataType extension** variable-access types is shown in Figure 25.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

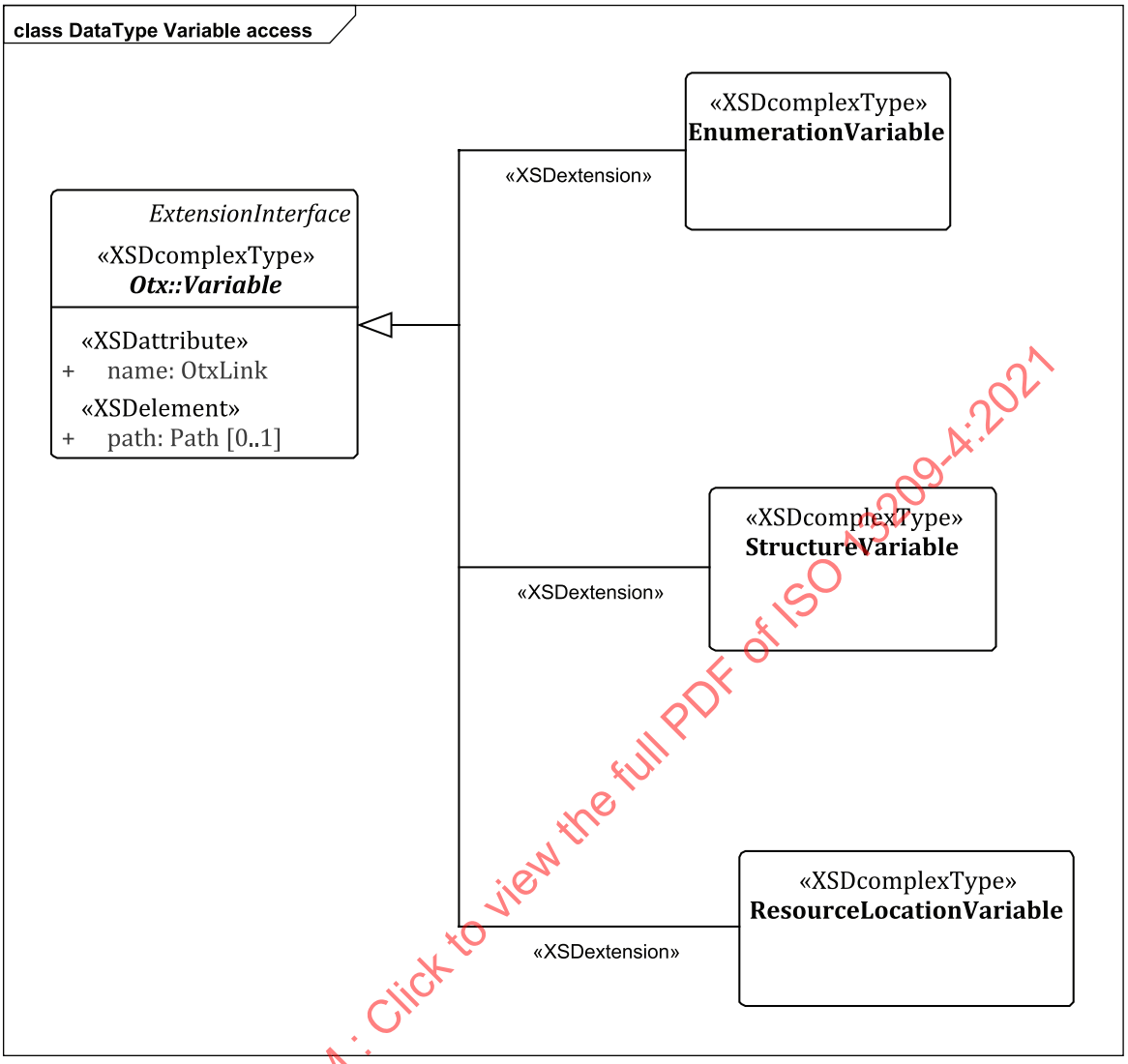


Figure 25 — DataType variable access

11.4.3 Variable-access semantics

The general semantics for all variable-access types shall apply. Please refer to ISO 13209-2 for details.

11.5 Declaration and arguments

11.5.1 Declaration and arguments syntax

Figure 26 shows the syntax of the **OTX DataType extension** declarations and arguments.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

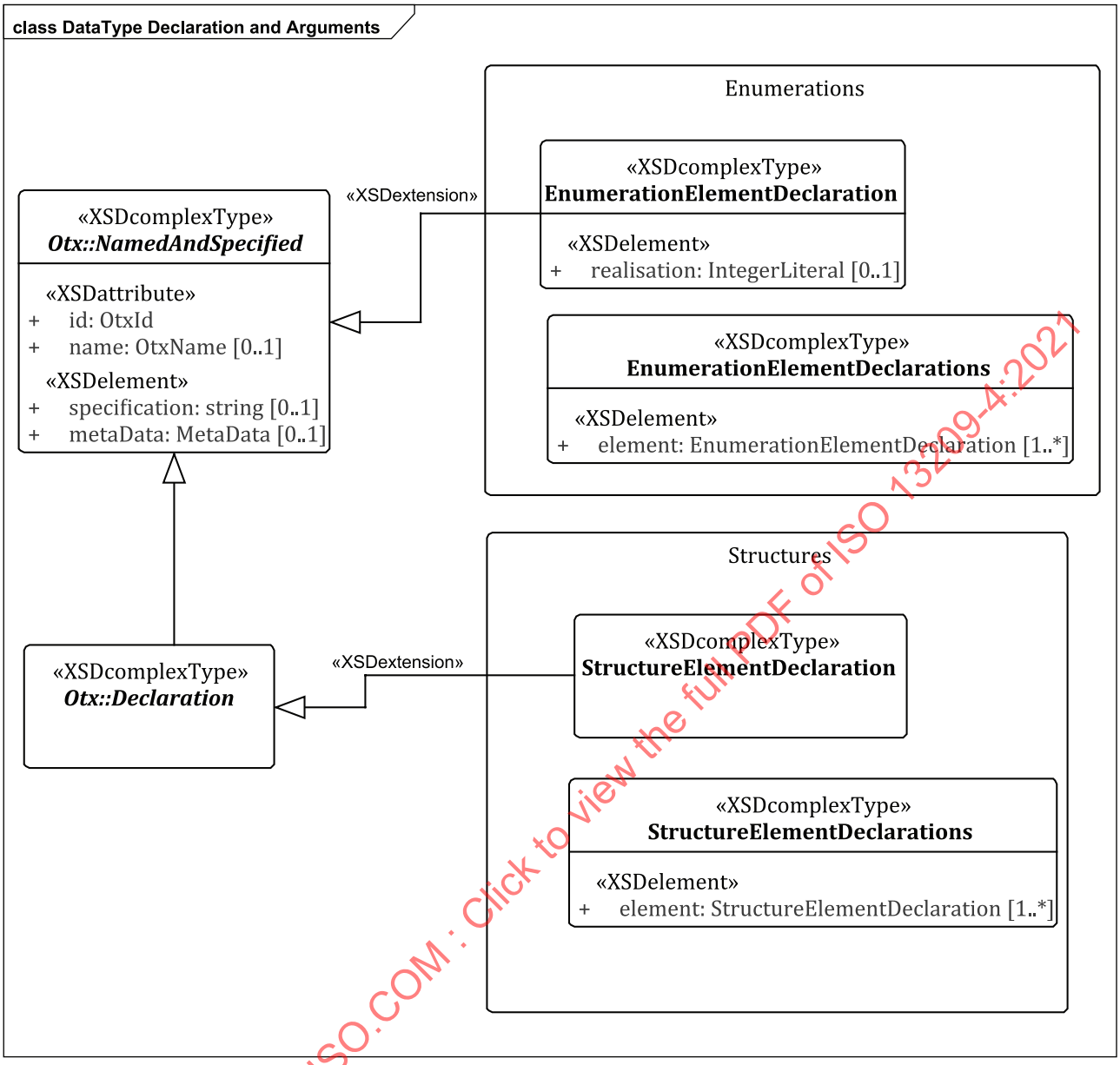


Figure 26 — DataType declaration and arguments

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

11.5.2 Declaration and arguments semantics

The semantic of the declarations and arguments is shown at the actions and terms, where these are used.

11.6 Signatures

11.6.1 Signatures overview

All signatures introduced by this extension extend the **otx:signature** or replace the **otx:signature** realisation.

11.6.2 Signatures syntax

Figure 27 shows the syntax of the **OTX DataType extension** related signatures.

© ISO 2021 – All rights reserved

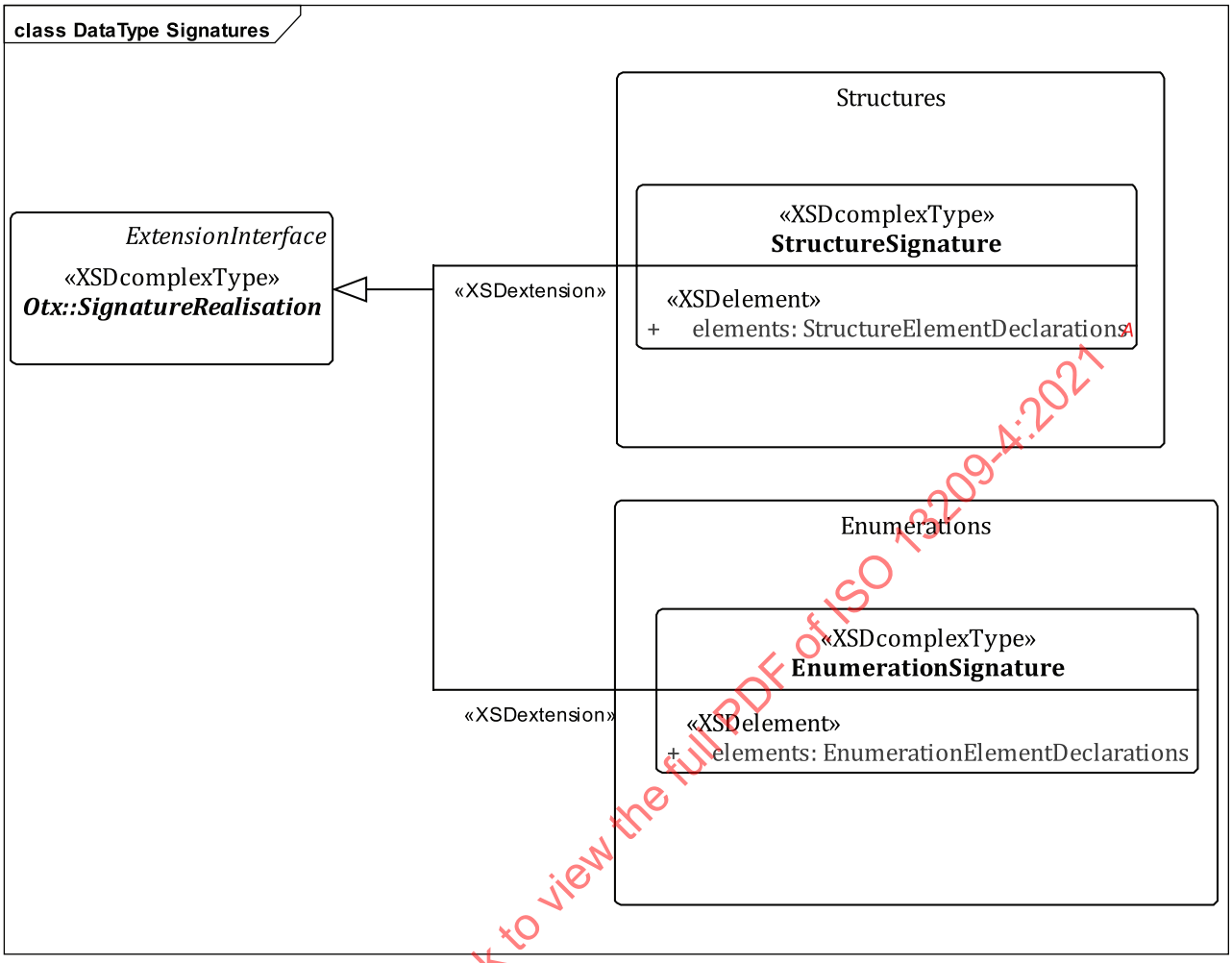


Figure 27 — DataType signatures

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

11.6.3 Semantics for Enumerations

11.6.3.1 EnumerationSignature

Semantic:

The **EnumerationSignature** is a specialisation of the OTX Core type **otx:SignatureRealisation** that adds additional interface description functionality along with the OTX Core type **otx: ProcedureSignature.** An **EnumerationSignature** represents the elements of an enumeration. Each element consists of a unique name and a unique Integer value of that element.

Parameters:

— <elements> : dataType:EnumerationElementDeclarations [1..1]

Elements of an **EnumerationSignature**.

<xsd:key name="UniqueEnumerationElementNames">

<xsd:selector xpath="dataType:element" />

<xsd:field xpath="@name" />

<xsd:key name="UniqueEnumerationElementValues">

© ISO 2021 – All rights reserved

<xsd:selector xpath="dataType:element/dataType:realisation" />

<xsd:field xpath="@value" /></xsd:key>

— <element> : dataType:EnumerationElementDeclaration [1..*]

Each **Enumeration** element consists of the mandatory name (see NamedAndSpecified) and a unique Integer value.

IMPORTANT — Since the element name is necessary, the name attribute is mandatory for EnumerationElementDeclaration (unlike the base type NamedAndSpecified where it is optional).

Associated checker rules:

— DataType_Chk004 – no mixture of implicit and explicit enumeration values,

— id : otx:OtxId [1..1] (derived from NamedAndSpecified).

This value represents the OtxId of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— name : otx:OtxName [0..1] (derived from NamedAndSpecified)

This value represents the OtxName of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— <realisation> : otx:IntegerLiteral [0..1]

Unique value of the **Enumeration** element.

— <specification> : xs:string [0..1] (derived from NamedAndSpecified)

This value represents the string of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

11.6.4 Semantics for Structures

11.6.4.1 StructureSignature

Semantic:

The **StructureSignature** is a specialisation of the OTX Core type **otx:SignatureRealisation** that adds additional interface description functionality along with the OTX Core type **otx: ProcedureSignature.** A **StructureSignature** represents the elements of a structure. Each element consists of a unique name and an arbitrary data type. The data type can be also a **Structure**. A structure cannot contain itself. In other words, circular dependencies of structures are not allowed.

Parameters:

— <elements> : dataType:StructureElementDeclarations [1..1]

Elements of a **StructureSignature**.

<xsd:key name="UniqueStructureElementNames">

<xsd:selector xpath="dataType:element" />

<xsd:field xpath="@name" />

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— <element> : dataType:StructureElementDeclaration [1..*]

Each **Structure** element consists of the mandatory name (see NamedAndSpecified) and a data type.

IMPORTANT — Since the element name is necessary, the name attribute is mandatory for StructureElementDeclaration (unlike the base type NamedAndSpecified where it is optional).

Associated checker rules:

— DataType_Chk003 – no Circular type definitions of Structures,

— id : otx:OtxId [1..1] (derived from NamedAndSpecified).

This value represents the OtxId of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— name : otx:OtxName [0..1] (derived from NamedAndSpecified)

This value represents the OtxName of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— <specification> : xs:string [0..1] (derived from NamedAndSpecified)

This value represents the string of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

11.7 Terms

11.7.1 Terms overview

All terms introduced by this extension extend the **otx:Term** extension interface. Information about the specific super class of a term is provided in the individual term description clauses.

11.7.2 Terms syntax

Figure 28 shows the syntax of the **OTX DataType extension** related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

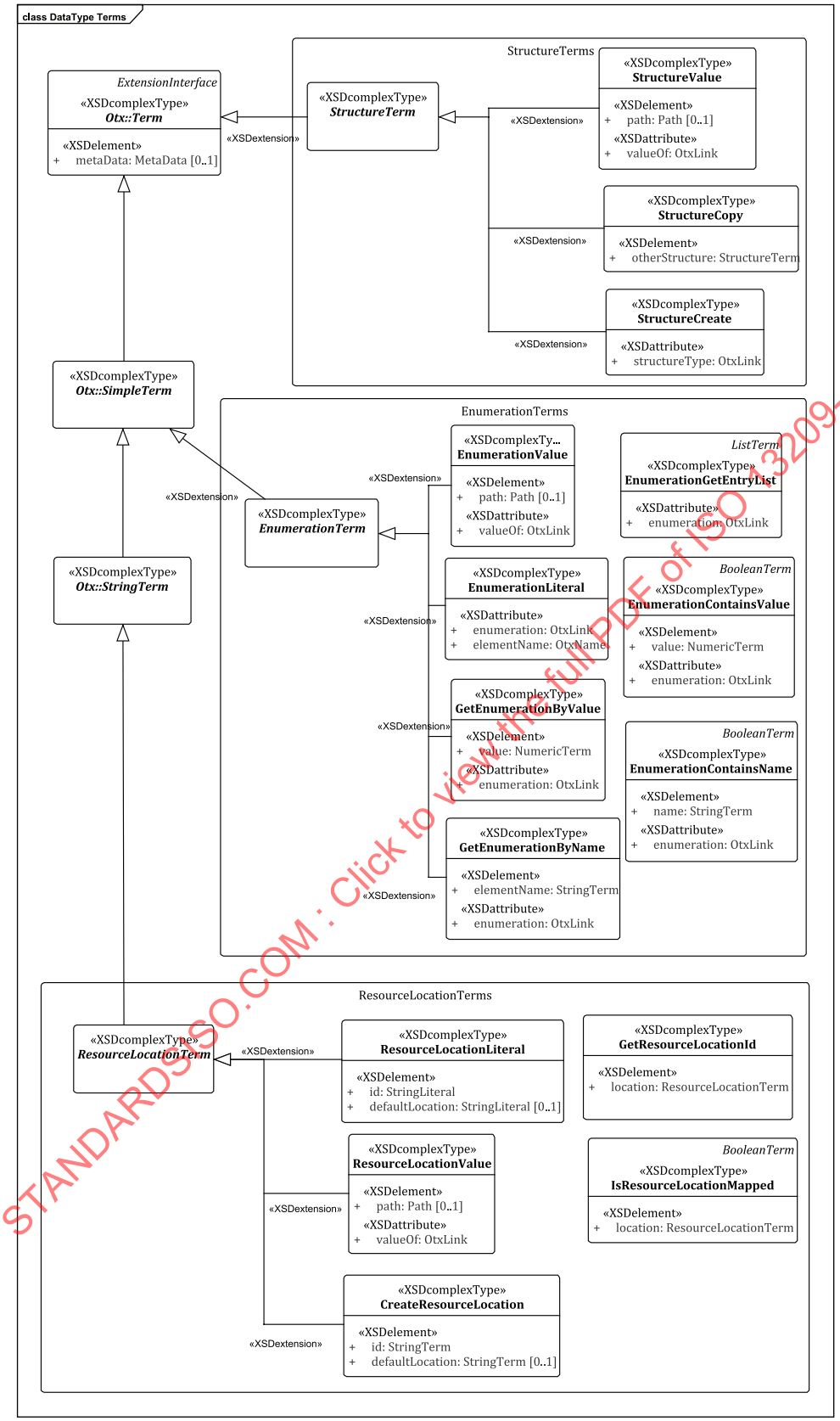


Figure 28 — DataType terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

11.7.3 Semantics for EnumerationTerms

11.7.3.1 EnumerationContainsName

Semantic:

Checks if the specified enumeration contains the given name. Returns true if the name exists otherwise false.

Return type:

otx:BooleanTerm

Parameters:

— enumeration : otx:OtxLink [1..1]

Refers to a certain **EnumerationSignature**.

— <name> : otx:StringTerm [1..1]

The name of the element to be checked for inside the specified enumeration.

Associated checker rules:

— Core_Chk053 – no dangling OtxLink associations (see ISO 13209-2).

Exceptions:

— None

11.7.3.2 EnumerationContainsValue

Semantic:

Checks if the specified enumeration contains the value.

Return type:

otx:BooleanTerm

Parameters:

— enumeration : otx:OtxLink [1..1]

Refers to a certain **EnumerationSignature**.

— <value> : otx:NumericTerm [1..1]

The value of the element to be checked for inside the specified enumeration. Float values shall be truncated.

Associated checker rules:

— Core_Chk053 – no dangling OtxLink associations (see ISO 13209-2).

Exceptions:

— None

11.7.3.3 EnumerationGetEntryList

Semantic:

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Returns the names in a specified enumeration as a List of String.

Return type:

otx:ListTerm

Parameters:

— enumeration : otx:OtxLink [1..1]

Refers to a certain **EnumerationSignature**.

Associated checker rules:

— Core_Chk053 – no dangling OtxLink associations (see ISO 13209-2).

Exceptions:

— None

11.7.3.4 EnumerationLiteral

Semantic:

Enumeration values are named constants. Therefore the **EnumerationLiteral** term shall return a reference to the value of the constant that has the name provided in the EnumerationLiteral's elementName field.

Return type:

dataType:EnumerationTerm

Parameters:

— enumeration : otx:OtxLink [1..1]

Refers to a certain **EnumerationSignature**.

— elementName : otx:OtxName [1..1]

The elementName shall correspond to a name of an element inside the linked **EnumerationSignature**.

Associated checker rules:

— Core_Chk053 – no dangling OtxLink associations (see ISO 13209-2),

— DataType_Chk002 – correct target for Enumeration.

Exceptions:

— None

11.7.3.5 EnumerationTerm

Semantic:

The Term for the **Enumeration**. For user-defined enumerations the different relational terms are using the integer value of the specific enumerator value.

11.7.3.6 EnumerationValue

Semantic:

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Returns the **EnumerationValue** stored in an **Enumeration** variable. No implicit conversion from enumeration to integer shall be used.

Return type:

dataType:EnumerationTerm

Parameters:

— <path> : otx:Path [0..1]

The element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:OutOfBoundsException

— Occurs only if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

11.7.3.7 GetEnumerationByName

Semantic:

Returns the **Enumeration** related to the element name in the specified enumeration.

Return type:

dataType:EnumerationTerm

Parameters:

— enumeration : otx:OtxLink [1..1]

Refers to a certain **EnumerationSignature**.

— <elementName> : otx:StringTerm [1..1]

Element name inside the specified **Enumeration**.

Associated checker rules:

— Core_Chk053 – no dangling OtxLink associations (see ISO 13209-2).

Exceptions:

otx:OutOfBoundsException

— This exception is thrown if the requested enumeration value does not exist.

11.7.3.8 GetEnumerationByValue

Semantic:

Returns the **Enumeration** related to the element value in the specified enumeration.

Return type:

dataType:EnumerationTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— enumeration : otx:OtxLink [1..1]

Refers to a certain **EnumerationSignature**.

— <value> : otx:NumericTerm [1..1]

Element value inside the specified **Enumeration**. Float values shall be truncated.

Associated checker rules:

— Core_Chk053 – no dangling OtxLink associations (see ISO 13209-2).

Exceptions:

otx:OutOfBoundsException

— This exception is thrown if the requested enumeration value does not exist.

11.7.4 Semantics for ResourceLocationTerms

11.7.4.1 CreateResourceLocation

Semantic:

This term creates creates a **ResourceLocation** out of a given ID and default location.

Return type:

dataType:ResourceLocationTerm

Parameters:

— <id> : otx:StringTerm [1..1]

Represents the resource ID value. At runtime the ID should be used to map the **ResourceLocation** to a certain path.

— <defaultLocation> : otx:StringTerm [0..1]

Represents the optional default location. This value should be used, if the related resource ID cannot be mapped. The value should be defined by a URI (see ISO 13209-2:—, Annex G).

Exceptions:

— None

11.7.4.2 GetResourceLocationId

Semantic:

The **GetResourceLocationId** term shall return the ID of the given **ResourceLocation**.

Return type:

otx:StringTerm

Parameters:

— <location> : ResourceLocationTerm [1..1]

Represents the **ResourceLocation** value on which ID shall be returned.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Exceptions:

— None

11.7.4.3 IsResourceLocationMapped

Semantic:

The **IsResourceLocationMapped** term shall return true, if the given **ResourceLocation** ID was found inside the mapping table, otherwise false.

IMPORTANT — The mapping table is not part of the standard.

Return type:

otx:BooleanTerm

Parameters:

— <location> : ResourceLocationTerm [1..1]

Represents the **ResourceLocation** value which shall be checked to determine if the ID is inside the mapping table or not.

Exceptions:

— None

11.7.4.4 ResourceLocationLiteral

Semantic:

This term shall be used to create a **ResourceLocation** object based on a hard-coded ID and a default location.

Return type:

dataType:ResourceLocationTerm

Parameters:

— <id> : otx:StringLiteral [1..1]

Represents the hard-coded resource ID value. At runtime the ID should be used to map the **ResourceLocation** to a certain location.

— <defaultLocation> : otx:StringLiteral [0..1]

Represents the optional hard-coded default location of the **ResourceLocationLiteral**. This value should be used, if the related resource ID cannot be mapped. The value should be defined by a URI (see ISO 13209-2:—, Annex G).

Exceptions:

— None

11.7.4.5 ResourceLocationTerm

Semantic:

The abstract type **ResourceLocationTerm** is an **otx:StringTerm.** It serves as a base for all concrete terms which return a **ResourceLocation** value. It returns the location of the related resource. If the related resource ID cannot be mapped, because a possible mapping table does not

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

contain the ID or the runtime does not support a mapping mechanism, the location is the default location (see ResourceLocationLiteral). If no default location is specified, the location is an empty string. It has no special members.

11.7.4.6 ResourceLocationValue

Semantic:

Returns the **ResourceLocation** stored in a **ResourceLocation** variable. For more information on value-terms and the syntax and semantics of the valueOf attribute and <path> element, please refer to ISO 13209-2.

Associated checker rules:

— Core_Chk053 – no dangling OtxLink associations (see ISO 13209-2).

Return type:

dataType:ResourceLocationTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

— None

11.7.5 Semantics for StructureTerms

11.7.5.1 StructureCopy

Semantic:

Create a deep copy of a **Structure** and return a new **Structure** instance. StructureCopy shall deeply copy Lists, Maps, ByteFields and Structures. Other OTX extensions may specify that certain complex data types can be deep-copied within StructureCopy.

Return type:

dataType:StructureTerm

Parameters:

— <otherStructure> : dataType:StructureTerm [1..1]

Stucture, which should be copied.

Exceptions:

— None

11.7.5.2 StructureCreate

Semantic:

Creates a new, initialised instance of the specified **Structure**.

Return type:

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

dataType:StructureTerm

Parameters:

— structureType : otx:OtxLink [1..1]

Refers to a certain **StructureSignature**.

Associated checker rules:

— Core_Chk053 – no dangling OtxLink associations (see ISO 13209-2).

Exceptions:

— None

11.7.5.3 StructureTerm

Semantic:

Term of the **Structure**.

11.7.5.4 StructureValue

Semantic:

Returns the StructureValue stored in a **Structure** variable. No implicit conversion from enumeration to integer shall be used.

Return type:

dataType:StructureTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:OutOfBoundsException

— Occurs only if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

12 OTX DiagComPlus extension

12.1 General

In ISO 13209-3 the DiagCom extension is specified, but the functionality of that extension is limited. It is not possible to send a functional hexadecimal diagnostic service request because with the **diag:ExecuteHexDiagService** action of ISO 13209-3 only one response can be received.

The execution of hexadecimal diagnostic services is extended to a functional addressing level, this means that more than one ECU is involved.

Additionally, a term is introduced, that enables creating a **diag:ComChannel** to a base variant without starting communication, in case of an underlying MVCI system.

© ISO 2021 – All rights reserved

12.2 Datatypes

12.2.1 Datatypes overview

All data types introduced by this extension are derived from **otx:SimpleType**.

12.2.2 Datatypes syntax

The syntax of the data-type declarations of the **OTX DiagComPlus extension** is shown in Figure 29.

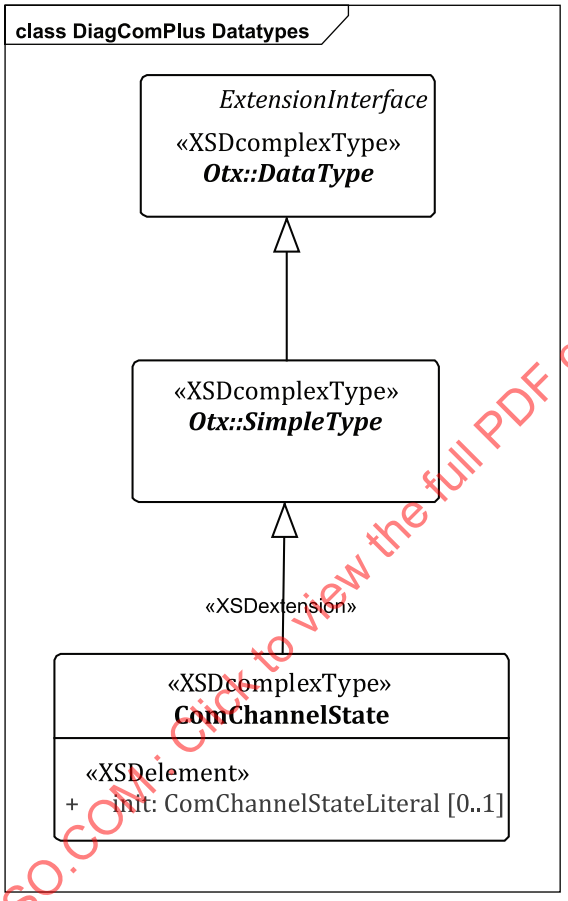


Figure 29 — DiagComPlus datatypes

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

12.2.3 Datatypes semantics

12.2.3.1 ComChannelState

ComChannelState is an **otx:SimpleType**.

ComChannelState is an enumeration type describing the state of a **diag:ComChannel.**

IMPORTANT — ComChannelState values may occur as operands of comparisons (see ISO 13209-2:—, 7.15.8). For this case, the following order relation shall apply: CREATED > OFFLINE > ONLINE > COMMUNICATION.

IMPORTANT — When applying otx:ToString on a ComChannelState value, the resulting string shall be the name of the enumeration value, e.g. otx:ToString(OFFLINE)="OFFLINE". Furthermore, applying otx:ToInteger shall return the index of the value in the ComChannelStates enumeration (smallest index is 0). The behaviour is undefined for other conversion terms (see ISO 13209-2).

© ISO 2021 – All rights reserved

Its members have the following semantics:

— <init> : diagComPlus:ComChannelStateLiteral[0..1]

This optional element sets the hard-coded initialisation value of the identifier at declaration time.

value : ComChannelStates={ CREATED | OFFLINE | ONLINE | COMMUNICATION } [1]

This attribute shall contain one of the values defined in the **ComChannelStates** enumeration.

IMPORTANT — If the ComChannelState declaration is not explicitly initialised (omitted <init> element), the default value shall be OFFLINE.

12.3 Enumerations

12.3.1 Enumerations syntax

The syntax of the enumeration declarations of the **OTX DiagComPlus extension** is shown in Figure 30.

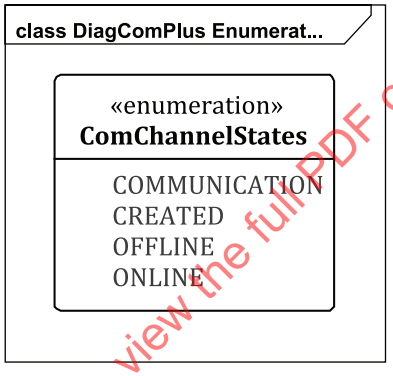


Figure 30 — DiagComPlus enumerations

12.3.2 Enumerations semantics

12.3.2.1 ComChannelStates

The **ComChannelStates** enumeration consists of values that specify the status of a **diag:ComChannel.**

ComChannelStates has the following enumeration values:

— COMMUNICATION

diag:ComChannel status: communication already started.

In case of an MVCI based system, this means that the logical link is online, communication is already established and that a DiagComPrimitive is executing.

— CREATED

diag:ComChannel status: created.

In case of an MVCI based system, this means that the logical link is created.

— OFFLINE

diag:ComChannel status: offline.

In case of an MVCI based system, this means that the logical link is offline. No communication has been established.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— ONLINE

diag:ComChannel status: online

In case of an MVCI based system, this means that the logical link is online, communication is already established and no DiagComPrimitive is executing.

12.4 Exceptions

12.4.1 Exceptions overview

All exceptions introduced by this extension are derived from the **otx:Exception** type. They represent the full set of exceptions added by the **OTX DiagComPlus extension**.

12.4.2 Exceptions syntax

The syntax of all **OTX DiagComPlus extension** exception-type declarations is shown in Figure 31.

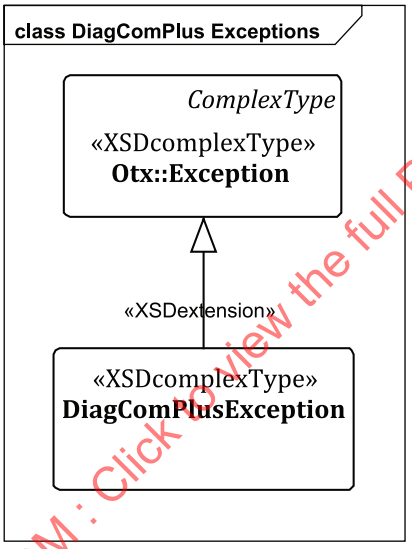


Figure 31 — DiagComPlus exceptions

12.4.3 Exceptions semantics

12.4.3.1 General

Since all **OTX DiagComPlus extension** exception types are implicit exceptions without initialisation parts, they cannot be declared constant.

12.4.3.2 DiagComPlusException

IMPORTANT — All terms and action realisations in this extension may potentially throw this exception.

12.5 Variable access

12.5.1 Variable-access overview

All variable-access types introduced by this extension are derived from the OTX core variable extension interface.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

12.5.2 Variable-access syntax

The syntax of the **OTX DiagComPlus extension** variable-access types is shown in Figure 32.

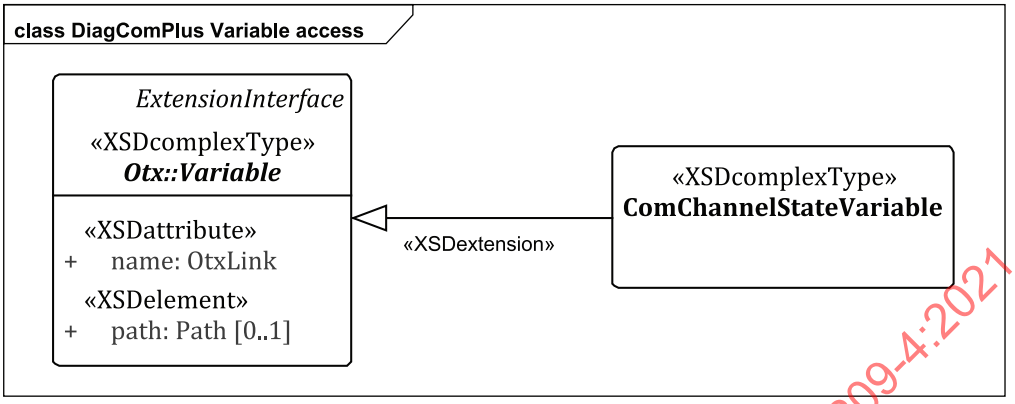


Figure 32 — DiagComPlus variable access

12.5.3 Variable-access semantics

The general semantics for all variable-access types shall apply. Please refer to ISO 13209-2 for details.

12.6 Actions

12.6.1 Actions overview

All actions introduced by this extension extend the **otx:ActionRealisation** extension interface.

12.6.2 Actions syntax

Figure 33 shows the syntax of all actions in the **OTX DiagComPlus extension**.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

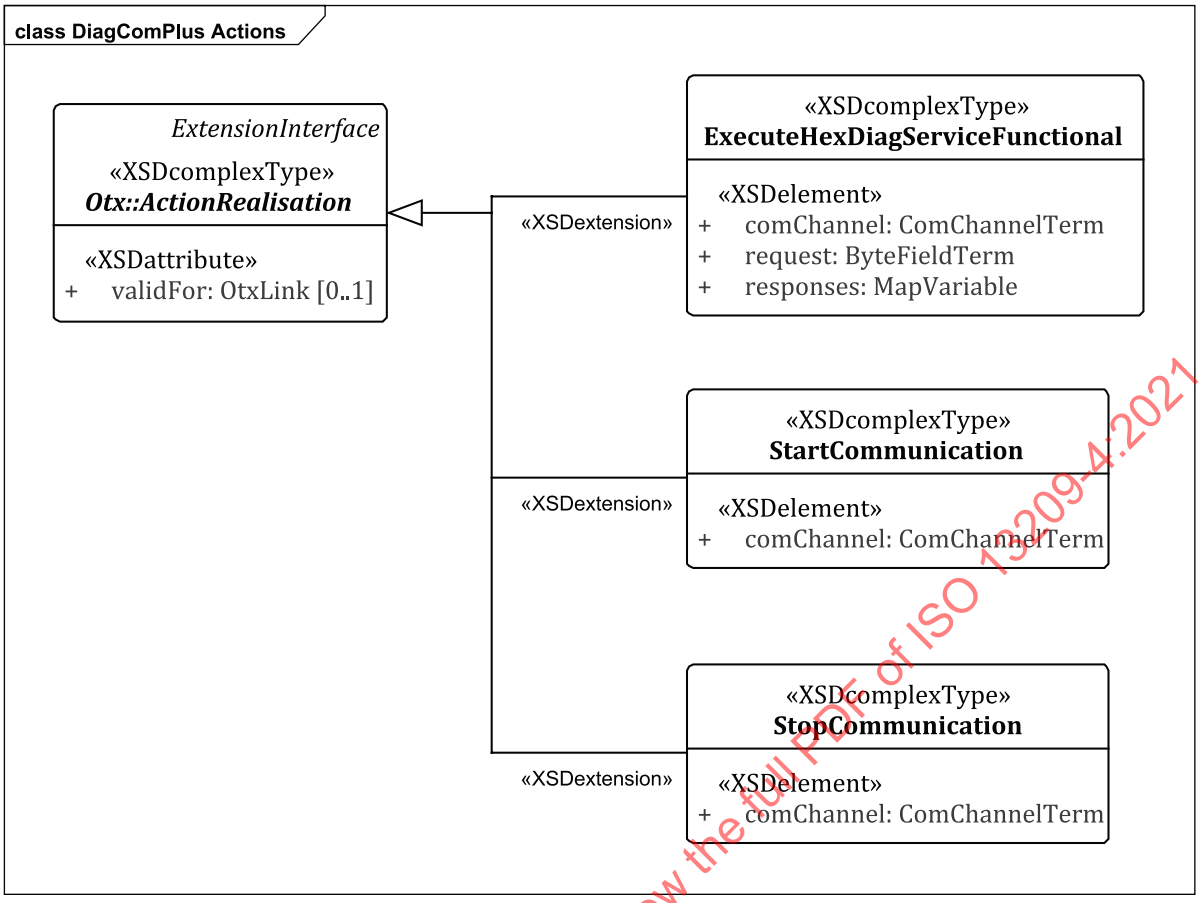


Figure 33 — DiagComPlus actions

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

12.6.3 Actions general semantics

12.6.3.1 ExecuteHexDiagServiceFunctional

Semantic:

The ExecuteHexDiagServiceFunctional action allows the sending of diagnostic services by directly entering the request byte stream. Unlike the **diag:ExecuteHexDiagService** action, it can receive responses from different ECUs (functional addressing, refer to the example in ISO 22900-3:2012, Figure One-shot service, functional addressing, single-part responses).

The time that the action shall wait for responses is not specified in this action. It depends on the communication parameters of the given **diag:ComChannel** or the underlying system.

Associated checker rules:

— DiagComPlus_Chk001 – correct element types for responses map.

Parameters:

— <comChannel> : diag:ComChannelTerm [1..1]

This element shall reference the handle of the communication channel, which shall be used for communication with the functional (or physical) link.

— <request> : otx:ByteFieldTerm [1..1]

This element shall contain the service request as a set of raw bytes.

© ISO 2021 – All rights reserved

— <responses> : otx:MapVariable [1..1]

This element specifies the OTX Map[String,Bytefield] variable to which the combination of ComChannelIdentifier and raw response bytes of the service shall be assigned. The key of the map is the ComChannelIdentifier. The associated value is the hexadecimal response of the ECU.

IMPORTANT — If there is more than one ComChannelIdentifier with the same ECU SHORT-NAME, but a different baud rate value: the one that uses the baud rate that is currently being used shall be returned.

IMPORTANT — If there is more than one ComChannelIdentifier with the same communication ID (e.g. CAN ID) and same baud rate, but a different ECU SHORT-NAME only one (not specified which one) of them shall be returned.

NOTE The map can be empty. For example, if the hexadecimal service is configured to suppress positive responses and all connected ECUs respond positively.

Exceptions:

diag:LossOfComException

— This exception is thrown if communication was interrupted during diagnostic service execution.

12.6.3.2 StartCommunication

Semantic:

This action starts the communication on a **diag:ComChannel.**

IMPORTANT — In case an ODX/MVCI system is used, the ComPrimitive "StartCommunication" is executed on the logical link. In this case the logical link state is COMMUNICATION (see ComChannelState).

Parameters:

— <comChannel> : diag:ComChannelTerm [1..1]

This element comprises the communication channel, which shall be used for starting the communication with the ECU the communication channel is connected to.

Exceptions:

diag:LossOfComException

— This exception is thrown if communication to the ECU was interrupted during the start communication process.

12.6.3.3 StopCommunication

Semantic:

This action stops the communication on a **diag:ComChannel.**

IMPORTANT — In case an ODX/MVCI system is used, the ComPrimitive "StopCommunication" is executed on the logical link. In this case the logical link state is OFFLINE (see ComChannelState).

Parameters:

— <comChannel> : diag:ComChannelTerm [1..1]

This element comprises the communication channel, which shall be used for stopping the communication with the ECU the communication channel is connected to.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Exceptions:

diag:LossOfComException

— If communication to the ECU was interrupted during the stop communication process.

12.7 Terms

12.7.1 Terms overview

All terms introduced by this extension extend the **otx:Term** extension interface. Information about the specific super class of a term is provided in the individual term description clauses.

12.7.2 Terms syntax

Figure 34 shows the syntax of the **OTX DiagComPlus extension** related terms.

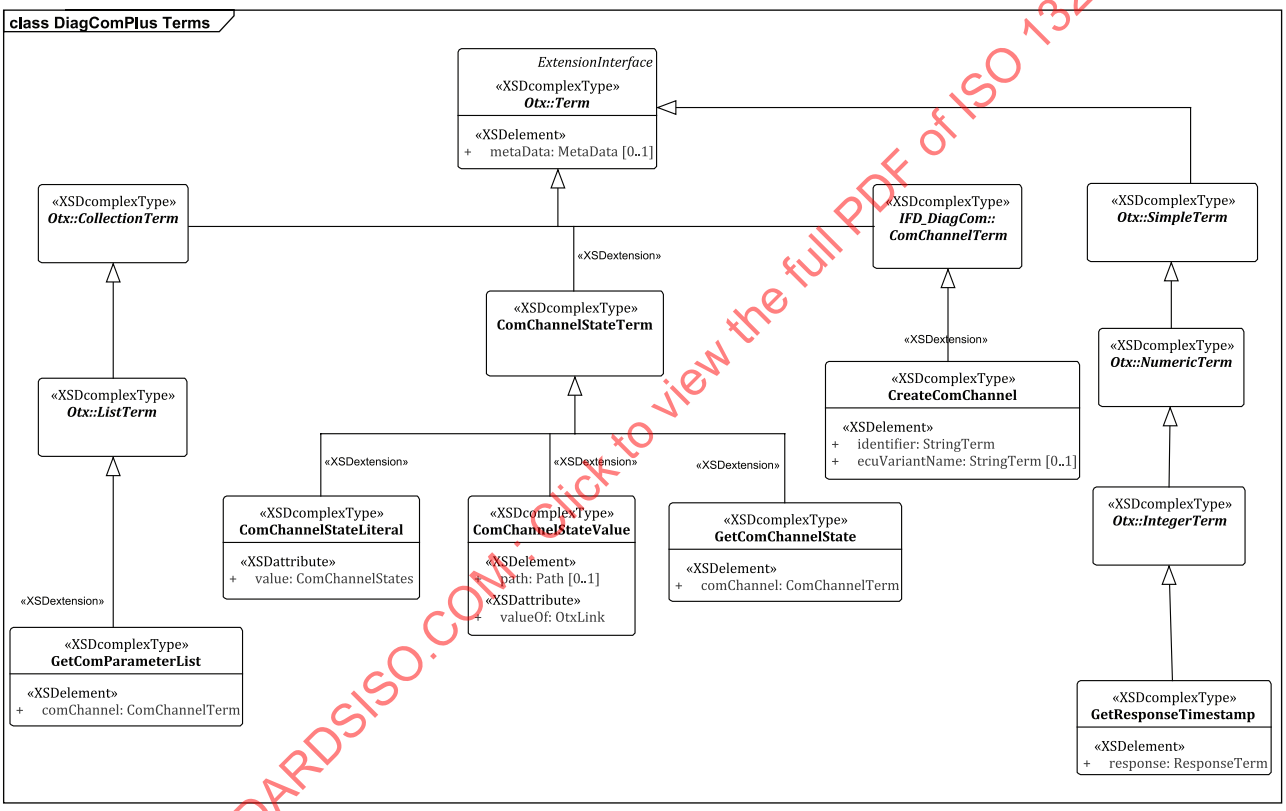


Figure 34 — DiagComPlus terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

12.7.3 Terms general semantics

12.7.3.1 ComChannelStateLiteral

Semantic:

This term shall return a **ComChannelState** value from a hard-coded literal.

Return type:

diagComPlus:ComChannelStateTerm

© ISO 2021 – All rights reserved

Parameters:

— value : diagComPlus:ComChannelStates [0..1]

This attribute shall contain one of the values defined in the **ComChannelStates** enumeration.

Exceptions:

— None

12.7.3.2 ComChannelStateTerm

Semantic:

The abstract type **ComChannelStateTerm** is an **otx:Term.** It serves as a base for all concrete terms, which return a **ComChannelState** value. It has no special members.

Return type:

diagComPlus:ComChannelStateTerm

Parameters:

— None

Exceptions:

— None

12.7.3.3 ComChannelStateValue

Semantic:

Returns the **ComChannelState** stored in a **ComChannelState** variable. For more information on value-terms and the syntax and semantics of the valueOf attribute and <path> element, please refer to ISO 13209-2.

Return type:

diagComPlus:ComChannelStateTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location, which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

12.7.3.4 CreateComChannel

Semantic:

This term shall create a communication channel to an ECU. The term is needed, because the diag:GetComChannel term does not guarantee that the communication is not started. For exchangeability the usage of this term is recommended instead of diag:GetComChannel, if the user wants to set communication parameters before starting communication.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

IMPORTANT — The channel is created at the time this term is executed.

IMPORTANT — If a diag:GetComChannel was executed before CreateComChannel the state of the diag:ComChannel shall not be changed by this term.

IMPORTANT — In case an ODX/MVCI system is used, a StartCommunication on the logical link is not executed. In this case the logical link state is ONLINE.

The term **CreateComChannel** shall always return a handle to the same **ComChannel** for a given ECU. It is possible to manually control the lifecycle of a **ComChannel** object by closing a **ComChannel** handle using the **diag:CloseComChannel** action. This is up to the author of a diagnostic sequence, an OTX runtime system is expected to clean up open **ComChannel** handles at the end of a diagnostic session.

NOTE 1 The appropriate MVCI system operation is MCDProject::createLogicalLink.

Return type:

diag:ComChannelTerm

Parameters:

— <identifier> : otx:StringTerm [1..1]

This element represents a string identifying the communication channel, which shall be created.

NOTE 2 In case an MVCI/ODX system is used the identifier specifies the SHORT-NAME of the MCDLogicalLink to be used for communication.

— <ecuVariantName> : otx:StringTerm [0..1]

This optional element allows an OTX sequence to explicitly specify a particular ECU variant that the **ComChannel** shall be associated with. It is provided in addition to the identifier attribute based on the assumption that the **ComChannel** identifier specifies a connection to a base variant of an ECU. The <ecuVariantName> element can be used to directly create a **ComChannel** to a specific ECU variant without needing to perform the ECU variant identification step.

NOTE 3 In case an MVCI/ODX system is used the <ecuVariantName> element specifies the SHORT-NAME of the MCDDbEcuVariant to be associated with the logical link.

Exceptions:

diag:LossOfComException

— This exception is thrown if communication to the ECU was interrupted during the variant identification process.

diag:UnknownTargetException

— This exception is thrown if the **diag:ComChannel** identifier provided by the <identifer> element does not exist or is invalid.

12.7.3.5 GetComChannelState

Semantic:

This term shall return the current state of the diag:ComChannel.

NOTE The appropriate MVCI system operation is MCDLogicalLink::getState.

Return type:

diagComPlus:ComChannelStateTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Parameters:

— <comChannel> : diag:ComChannelTerm [1..1]

This element shall reference the handle of the communication channel whose status should be returned.

In case of an MVCI base system, the four statuses are available for a logical link. For other base systems it should be possible to logically map the statuses to the existing ones.

Exceptions:

diag:UnknownTargetException

— This exception is thrown if the **diag:ComChannel** identifier provided by the identifier element does not exist or is invalid.

12.7.3.6 GetComParameterList

Semantic:

The **GetComParameterList** term accepts a **diag:ComChannel** and returns an **otx:List** of **otx:Strings** with the available communication parameter names. In case no communication parameters can be determined an empty list shall be returned.

NOTE 1 The appropriate MVCI system operation is MCDDbLocation::getDbDiagComPrimitives(eMCDDBPROTOCOLPARAMETERSET).

Return type:

otx:ListTerm

Parameters:

— <comChannel> : diag:ComChannelTerm [1..1]

This element shall be the handle of the communication channel whose communication parameter names should be returned.

Exceptions:

— None

12.7.3.7 GetResponseTimestamp

Semantic:

GetResponseTimestamp shall return a timestamp to identify the time at which a Response from a vehicle was received. Systems should provide as much precision as possible - ideally by recording the time in receiving hardware. If this is not possible the timestamp can be recorded in the receiving software stack but in all cases a timestamp shall be returned.

GetResponseTimestamp gives greater precision than could be obtained by using DateTime: GetTimestamp in an OTX sequence since the timestamp may exclude delays in communication processing and OTX execution. Consequently, the term is useful for time-critical applications.

GetResponseTimestamp is an **otx:IntegerTerm** that returns a timestamp expressed in milliseconds elapsed since 1970-01-01 00:00:00 UTC. The timestamp uses the same definition as DateTime:GetTimestamp.

NOTE 2 The related MVCI system method is MCDResponse::getEndTime().

Return type:

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

otx:IntegerTerm

Parameters:

— <response> : diag:ResponseTerm [1..1]

The Response whose timestamp shall be returned.

Exceptions:

— None

13 OTX DiagConfiguration extension

13.1 General

In ISO 13209-3 extensions are specified to use, amongst other systems, MCD projects and the associated VehicleInformation implicitly. In case of an underlying MVCI-Server the functionality of this extension is limited. There is no possibility to select or change the current project and one associated VehicleInformation. The OTX DiagConfiguration extension has therefore been introduced to overcome this limitation. It enables selecting and changing the currently used project and one associated VehicleInformation. Furthermore it enables getting the current project SHORT-NAME, version and the current VehicleInformation SHORT-NAME.

The extension can be used to set an MCD project at the beginning of an OTX sequence or to change the currently used MCD project. It does not prohibit the run-time system from implicitly selecting an MCD project.

IMPORTANT — If an OTX runtime system is running in parallel to another diagnostic system that uses the same base system, this extension can cause problems because in case of an MVCI-Server only one project can be selected at same time.

13.2 Exceptions

13.2.1 Exceptions overview

All exceptions introduced by this extension are derived from the **otx:Exception** type. They represent the full set of exceptions added by the OTX DiagConfiguration extension.

13.2.2 Exceptions syntax

The syntax of all OTX DiagConfiguration extension exception-type declarations is shown in Figure 35.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

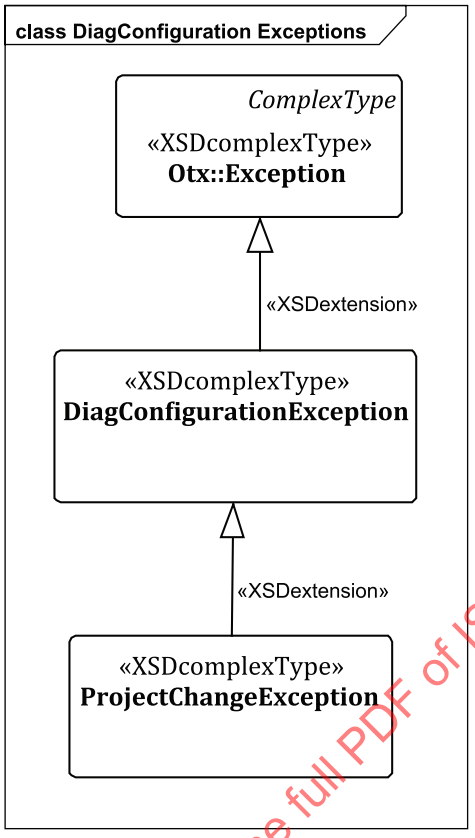


Figure 35 — DiagConfiguration exceptions

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

13.2.3 Exceptions semantics

13.2.3.1 General

Since all OTX DiagConfiguration extension exception types are implicit exceptions without initialisation parts, they cannot be declared constant.

13.2.3.2 ProjectChangeException

If the Project could not be changed or set this exception is thrown.

13.2.3.3 DiagConfigurationException

A **DiagConfigurationException** shall be used in case the more specific exception types described in the remainder of this section do not apply to the problem at hand.

IMPORTANT — All terms and action realisations in this extension may potentially throw this exception.

13.3 Actions

13.3.1 Actions overview

All actions introduced by this extension extend the **otx:ActionRealisation** extension interface.

13.3.2 Actions syntax

Figure 36 shows the syntax of all actions in the OTX DiagConfiguration extension.

© ISO 2021 – All rights reserved

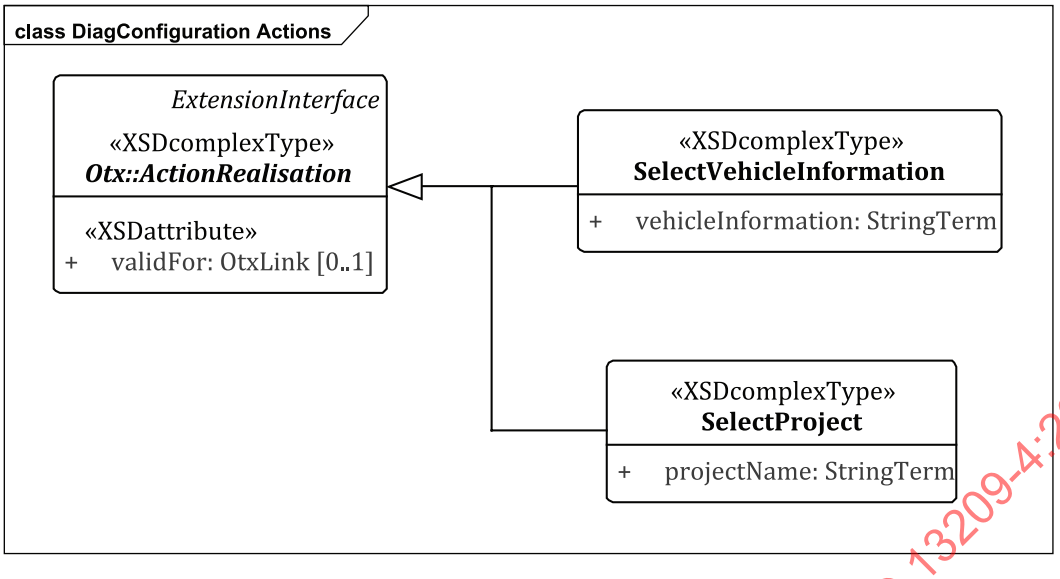


Figure 36 — DiagConfiguration actions

13.3.3 Actions general semantics

13.3.3.1 SelectProject

Semantic:

Selects the MCD project using the given SHORT-NAME.

NOTE The appropriate MVCI system operation is MCDSystem::selectProjectByName.

If selecting or changing the active project is not possible a **DiagConfigurationException** is thrown.

IMPORTANT — SelectProject action closes all Diag:ComChannels. ComChannelVariables are no longer initialised.

IMPORTANT — If an OTX runtime system is running in parallel to another diagnostic system that uses the same base system, this action can cause problems because in case of an MVCI-Server only one project can be selected at same time.

Parameters:

— projectName : otx:StringTerm [1..1]

SHORT-NAME of the MCD project.

Exceptions:

diag:UnknownTargetException

— This exception is thrown if the project identifier provided by the ProjectName element does not exist or is invalid.

ProjectChangeException

— This exception is thrown if the project cannot be set or changed for any reason.

13.3.3.2 SelectVehicleInformation

Semantic:

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Selects the vehicle information by the given SHORT-NAME. The vehicle information contains the used protocols (CAN/DoIP) and the used **ComChannel**s.

NOTE The appropriate MVCI system operation is MCDProject::selectDbVehicleInformationByName.

If selecting or changing the vehicle information is not possible a **DiagConfigurationException** is thrown.

IMPORTANT — SelectVehicleInformation action closes all Diag:ComChannels. ComChannelVariables are no longer initialised.

IMPORTANT — If an OTX runtime system is running in parallel to another diagnostic system that uses the same base system, this action can cause problems because in case of an MVCI-Server only one project can be selected at same time.

Parameters:

— vehicleInformation : otx:StringTerm [1..1]

The SHORT-NAME of the VehicleInformation.

Exceptions:

diag:UnknownTargetException

— This exception is thrown if the vehicle information identifier provided by the VehicleInformation element does not exist or is invalid.

13.4 Terms

13.4.1 Terms overview

All terms introduced by this extension extend the **otx:Term** extension interface. Information about the specific super class of a term is provided in the individual term description clauses.

13.4.2 Terms syntax

Figure 37 shows the syntax of the OTX DiagConfiguration extension-related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

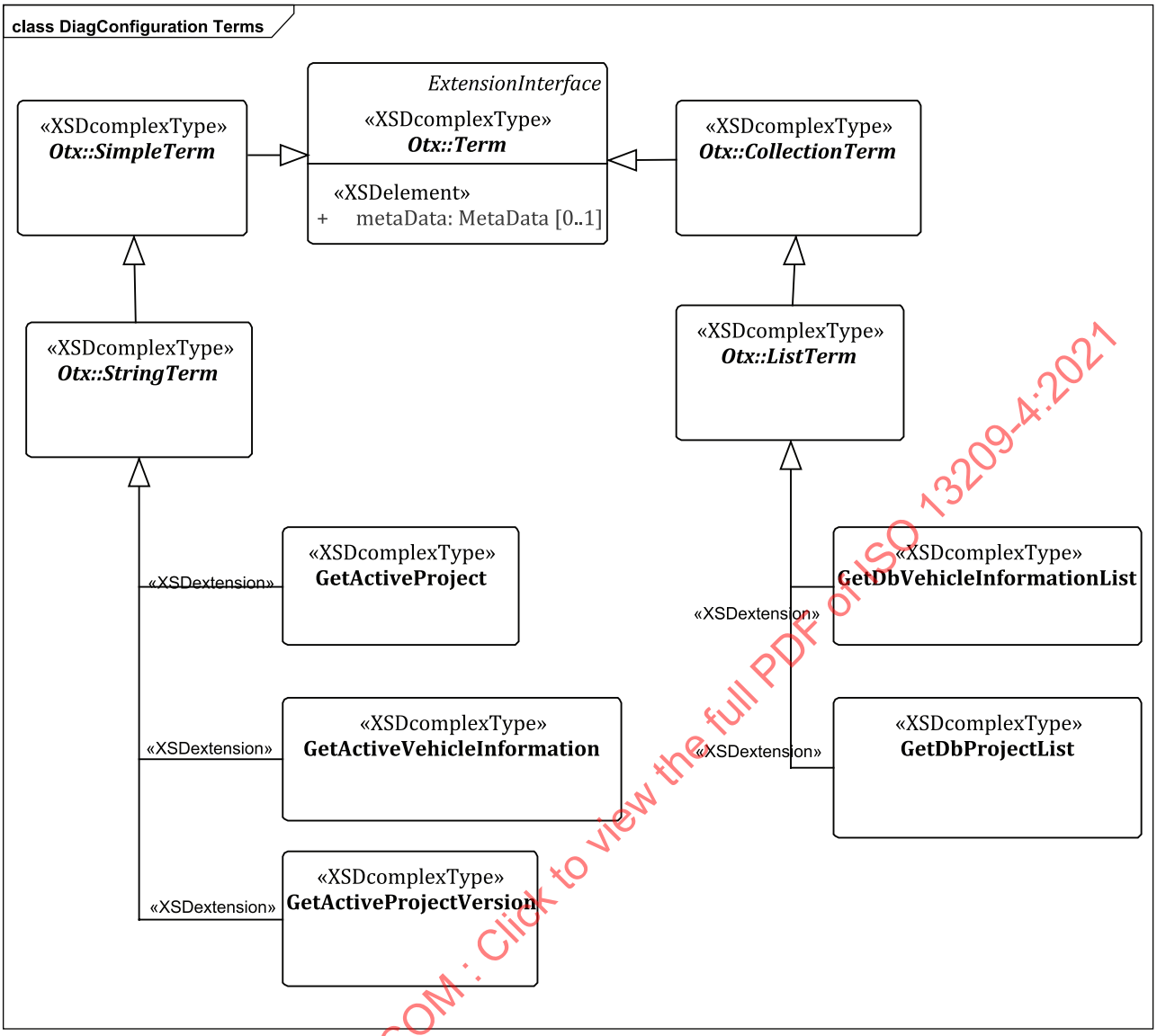


Figure 37 — DiagConfiguration terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

13.4.3 Terms general semantics

13.4.3.1 GetActiveProject

Semantic:

Returns the SHORT-NAME of the active project. If no project is active the returned String is empty.

Return type:

otx:StringTerm

Parameters:

— None

Exceptions:

— None

© ISO 2021 – All rights reserved

13.4.3.2 GetActiveProjectVersion

Semantic:

Returns the version of the active project as an **otx:String.** If no project is active the returned String is empty.

NOTE The appropriate MVCI system operation is MCDDbProject::getVersion().

Return type:

otx:StringTerm

Parameters:

— None

Exceptions:

— None

13.4.3.3 GetActiveVehicleInformation

Semantic:

Returns the SHORT-NAME of the active vehicle information. If no vehicle information is active, the returned String is empty.

NOTE The appropriate MVCI system operation is MCDProject::getActiveVehicleInformation.

Return type:

otx:StringTerm

Parameters:

— None

Exceptions:

— None

13.4.3.4 GetDbProjectList

Semantic:

Returns the Project SHORT-NAMEs as a list of strings.

NOTE The appropriate MVCI system operation is MCDSytem::getDbProjectDescriptions::getItemByIndex::getShortName.

Return type:

otx:ListTerm

Parameters:

— None

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

13.4.3.5 GetDbVehicleInformationList

Semantic:

Returns a list of strings with all vehicle information SHORT-NAMEs of the project. If no project is selected, the returned String is empty. A selected project can have an empty vehicle information list.

NOTE The appropriate MVCI system operation is MCDDbProject::getDbVehicleInformations::getItemByIndex::GetShortName.

Return type:

otx:ListTerm

Parameters:

— None

Exceptions:

— None

14 OTX DiagDataBrowsingPlus extension

14.1 General

The **OTX DiagDataBrowsingPlus extension** extends the DiagDataBrowsing extension specified in ISO 13209-3. It provides a set of datatypes and terms for reading static information associated with communication channels, diagnostic services, request- or response-parameters and other data stored in an ODX database, which are not covered by the DiagDataBrowsing extension.

NOTE 1 In contradiction to the design goal of the DiagCom extension in ISO 13209-3, this extension only makes sense together with an ODX/MVCI [ISO 22901] [ISO 22900] based system.

NOTE 2 The informations provided by the **OTX DiagDataBrowsingPlus extension** terms are dependent on the pre-loaded ODX data [ISO 22901] and can be queried via the MVCI-Server-API [ISO 22900-3]. At each term the appropriate MVCI server method is listed.

NOTE 3 The OTX DiagDataBrowsing extension is based on the OTX DiagCom extension. It uses the diag:DiagService and diag:Parameter objects from which diverse static information can be queried.

14.2 Datatypes

14.2.1 Datatypes overview

All data types introduced by this extension are derived from **otx:ComplexType** or **otx:SimpleType**.

14.2.2 Datatypes syntax

The syntax of the data-type declarations of the **OTX DiagDataBrowsingPlus extension** is shown in Figure 38.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

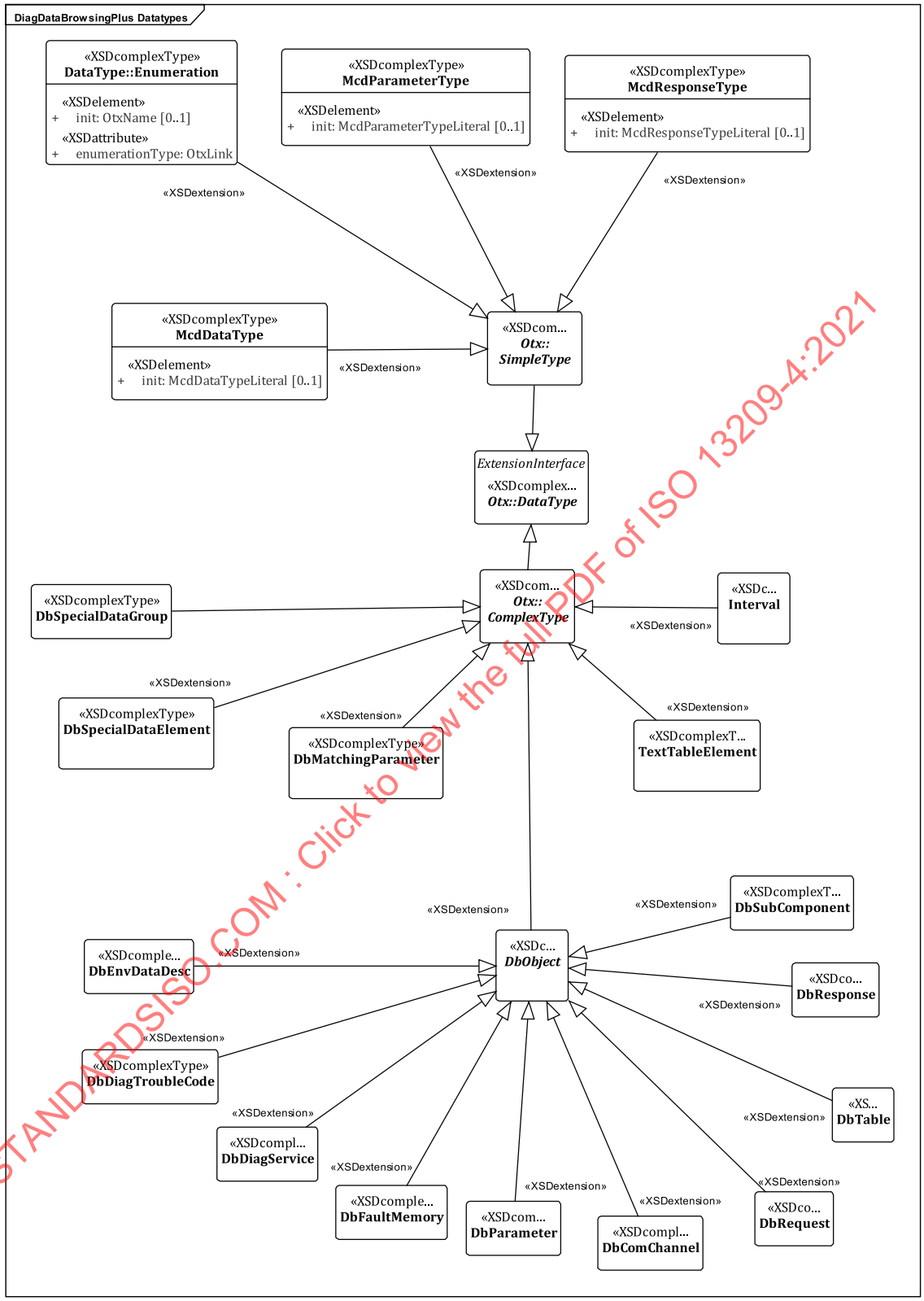


Figure 38 — DiagDataBrowsingPlus datatypes

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

14.2.3 Datatypes semantics

14.2.3.1 DbComChannel

DbComChannel is a **DbObject**.

A **DbComChannel** represents the MCDDbLocation object of an MVCI system. A **DbComChannel** is a handle to the database part of a **ComChannel** object in DiagCom, see ISO 13209-3:2012, 7.3.3.2.

14.2.3.2 DbDiagService

DbDiagService is a **DbObject**.

A **DbDiagService** represents the MCDDbDiagComPrimitive object of an MVCI system. A **DbDiagService** is a handle to the database part of a DiagService object in DiagCom, see ISO 13209-3:2012, 7.3.3.3.

14.2.3.3 DbDiagTroubleCode

DbDiagTroubleCode is a **DbObject**.

A **DbDiagTroubleCode** represents the MCDDbDiagTroubleCode object of an MVCI system which contains an interface for a DTC from ODX of the project's databases.

IMPORTANT — It shall be possible to map a variable of datatype DbDiagTroubleCode to a ResponseParameter of type eDTC via Inline Mapping, see diagCom:ExecuteDiagService action.

14.2.3.4 DbEnvDataDesc

DbEnvDataDesc is a **DbObject**.

A **DbEnvDataDesc** represents the MCDDbEnvDataDesc object of an MVCI system, which contains a collection of MCDDbResponseParameter objects of type ENV-DATA-DESC in ODX.

14.2.3.5 DbFaultMemory

DbFaultMemory is a **DbObject**.

A **DbFaultMemory** represents the MCDDbFaultMemory object of an MVCI system which contains a collection of MCDDbDiagTroubleCode objects.

14.2.3.6 DbMatchingParameter

DbMatchingParameter is an **otx:ComplexType**.

A **DbMatchingParameter** represents the MCDDbMatchingParameter object of an MVCI system.

14.2.3.7 DbObject

DbObject is an **otx:ComplexType**.

A **DbObject** represents the MCDDbObject object of an MVCI system. It is the common interface for all database objects. It defines the basic functionality available for all database objects.

IMPORTANT — It is recommended that the otx:ToString conversion should return the SHORT-NAME of the related MCDDbObject.

14.2.3.8 DbParameter

DbParameter is a **DbObject**.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

A **DbParameter** represents the MCDDbParameter object of an MVCI system. A **DbParameter** is a handle to the database part of a parameter object in DiagCom, see ISO 13209-3:2012, 7.3.3.6.

14.2.3.9 DbRequest

DbRequest is a **DbObject**.

A **DbRequest** represents the MCDDbRequest object of an MVCI system. A **DbRequest** is a handle to the database part of a Request object in DiagCom, see ISO 13209-3:2012, 7.3.3.9.

14.2.3.10 DbResponse

DbResponse is a **DbObject**.

A **DbResponse** represents the MCDDbResponse object of an MVCI system. A D**bResponse** is a handle to the database part of a Response object in DiagCom, see ISO 13209-3:2012, 7.3.3.8.

14.2.3.11 DbSpecialDataElement

DbSpecialDataElement is an **otx:ComplexType**.

A **DbSpecialDataElement** represents the MCDDbSpecialDataElement object of an MVCI system which provides access to the elementary text of the additional information.

IMPORTANT — It is recommended that the otx:ToString conversion should return the content of the related MCDDbSpecialDataElement, see MCDDbSpecialDataElement::getContent.

14.2.3.12 DbSpecialDataGroup

DbSpecialDataGroup is an **otx:ComplexType**.

A **DbSpecialDataGroup** represents the MCDDbSpecialDataGroup object of an MVCI system which provides additional information for specific ODX elements, e.g. for DIAG-COMM (in MCD3 realised as MCDDataPrimitive), for DTCs or FlashSessions. SDGs have been introduced to be able to capture information that is not covered by general ODX mechanisms. For example, SDGs can be used to attach error set conditions to DTCs or to provide error trees.

The SHORT-NAME, LONG-NAME, etc. come from the caption because MCDDbSpecialDataGroup does not inherit from MCDDbObject. If no caption exists, SHORT-NAME, LONG-NAME, etc. are adjusted to empty strings.

14.2.3.13 DbSubComponent

DbSubComponent is a **DbObject**.

A **DbSubComponent** represents the MCDDbSubComponent object of an MVCI system, whose is considered to be a functional unit in- or outside of an ECU that covers certain additional diagnostics relevant functionality either physically (e.g. a LIN slave) or logically.

14.2.3.14 DbTable

DbTable is a **DbObject**.

A **DbTable** represents the MCDDbTable object of an MVCI system. Such a table is composed of a non-empty set of TABLE-ROWs. Each TABLE-ROW of a TABLE is mapped to a **DbParameter** contained in a **DbTable** object. If a containing **DbParameter** with datatype eTABLE_ROW references a simple DOP in ODX, the resulting DbParameters collection delivered by term **GetDbParameters** contains exactly one **DbParameter**. The D**bParameter** has the parameter type eGENERATED, the datatype of the referenced simple DOP, and the SHORT-NAME #RtGen_.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

14.2.3.15 Interval

Interval is an **otx:ComplexType**.

An **Interval** represents the MCDInterval object of an MVCI system which provides the range of an interval.

IMPORTANT — It is recommended that the otx:ToString conversion should return the following string "[LowerLimitValueAsString, UpperLimitValueAsString]", e.g. "[1, 6]" or "[A, Z]".

14.2.3.16 McdDataType

McdDataType is an **otx:SimpleType**.

McdDataType is an enumeration type describing the ASAM data types used by the MVCI/ODX system.

IMPORTANT — The otx:ToString conversion shall return the name of the enumeration value and the otx:ToInteger conversion shall return the index of the value in the enumeration (smallest index is 0).

Its members have the following semantics:

— <init> : diagDataPlus:McdDataTypeLiteral[0..1]

This optional element stands for the hard-coded initialisation value of the identifier at declaration time.

14.2.3.17 McdParameterType

McdParameterType is an **otx:SimpleType**.

McdParameterType is an enumeration type describing the ASAM types of parameters used by the MVCI/ODX system.

IMPORTANT — The otx:ToString conversion shall return the name of the enumeration value and the otx:ToInteger conversion shall return the index of the value in the enumeration (smallest index is 0).

Its members have the following semantics:

— <init> : diagDataPlus:McdParameterTypeLiteral[0..1]

This optional element stands for the hard-coded initialisation value of the identifier at declaration time.

14.2.3.18 McdResponseType

McdResponseType is an **otx:SimpleType**.

McdResponseType is an enumeration type describing the ASAM types of responses used by the MVCI/ODX system.

IMPORTANT — The otx:ToString conversion shall return the name of the enumeration value and the otx:ToInteger conversion shall return the index of the value in the enumeration (smallest index is 0).

Its members have the following semantics:

— <init> : diagDataPlus:McdResponseTypeLiteral[0..1]

This optional element stands for the hard-coded initialisation value of the identifier at declaration time.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

14.2.3.19 TextTableElement

TextTableElement is an **otx:ComplexType**.

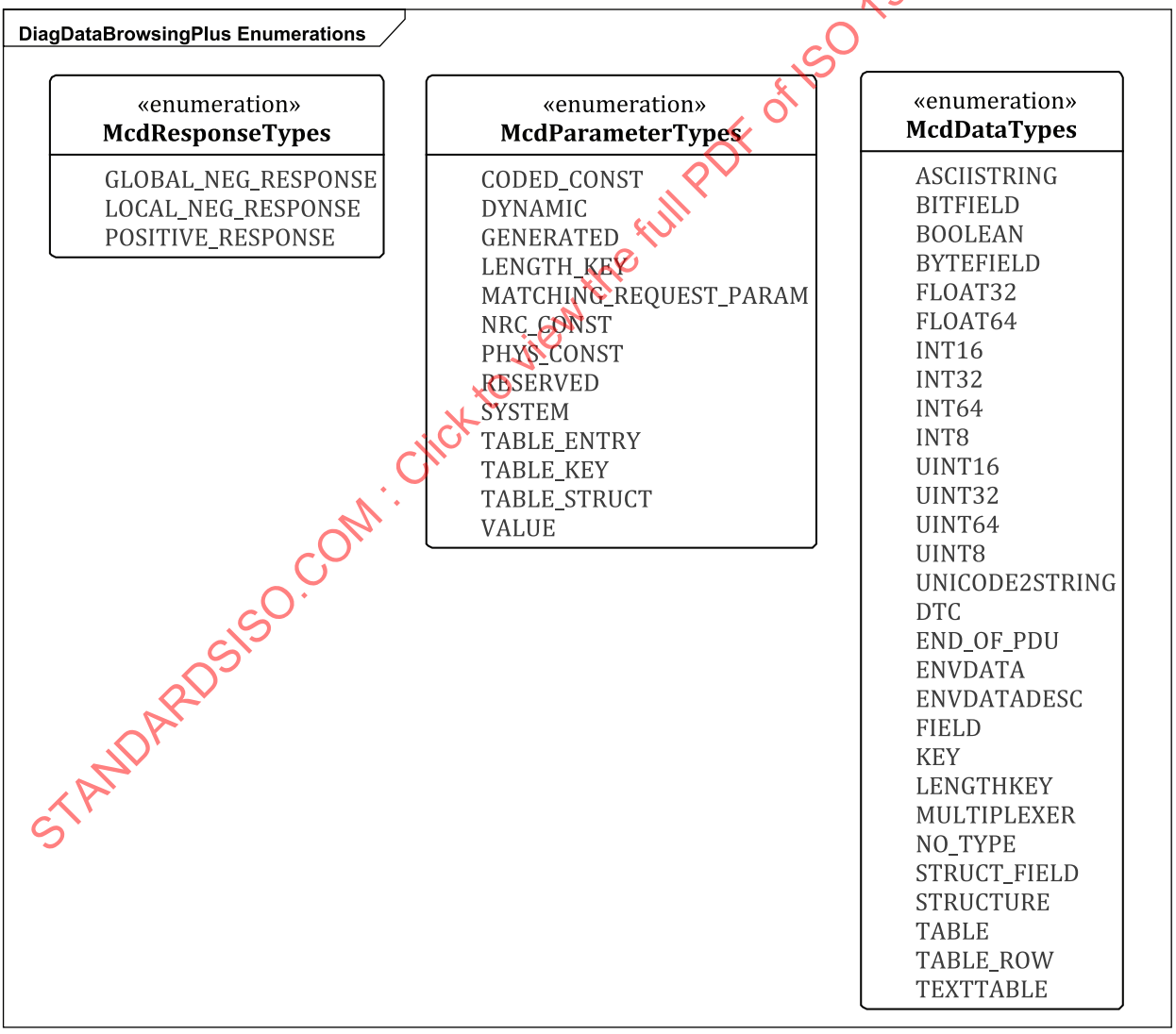
A **TextTableElement** is an instance of the MCDTextTableElement interface. It provides the meta information of one element of a TextTable. Any non-existent properties like SHORT-NAME always return an empty string.

IMPORTANT — It is recommended that the otx:ToString conversion should return the LONG-NAME of the MCDTextTableElement (LONG-NAME is mandatory).

14.3 Enumerations

14.3.1 Enumerations syntax

The syntax of the enumeration declarations of the **OTX DiagDataBrowsingPlus extension** is shown in Figure 39.



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 39 — DiagDataBrowsingPlus enumerations

© ISO 2021 – All rights reserved

14.3.2 Enumerations semantics

14.3.2.1 McdDataTypes

Defines the constant ASAM data types used by the MVCI/ODX system (McdDataType).

McdDataTypes has the following enumeration values:

— ASCIISTRING

Data type is MCDDataType::eA_ASCIISTRING (simple data type).

— BITFIELD

Data type is MCDDataType::eA_BITFIELD (simple data type).

— BOOLEAN

Data type is MCDDataType::eA_BOOLEAN (simple data type).

— BYTEFIELD

Data type is MCDDataType::eA_BYTEFIELD (simple data type).

— FLOAT32

Data type is MCDDataType::eA_FLOAT32 (simple data type).

— FLOAT64

Data type is MCDDataType::eFLOAT16 (simple data type).

— INT16

Data type is MCDDataType::eA_INT16 (simple data type).

— INT32

Data type is MCDDataType::eA_INT32 (simple data type).

— INT64

Data type is MCDDataType::eA_INT64 (simple data type).

— INT8

Data type is MCDDataType::eA_INT8 (simple data type).

— UINT16

Data type is MCDDataType::eA_UINT16 (simple data type).

— UINT32

Data type is MCDDataType::eA_UINT32 (simple data type).

— UINT64

Data type is MCDDataType::eA_UINT64 (simple data type).

— UINT8

Data type is MCDDataType::eA_UINT8 (simple data type).

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— UNICODE2STRING

Data type is MCDDataType::eA_UNICODE2STRING (simple data type).

— DTC

Data type is MCDDataType::eDTC (simple data type).

— END_OF_PDU

Data type is MCDDataType::eeEND_OF_PDU (complex data type).

— ENVDATA

Data type is MCDDataType::eENVDATA (complex data type).

— ENVDATADESC

Data type is MCDDataType::eEnvDataDesc (complex data type).

— FIELD

Data type is MCDDataType::eFIELD (complex data type).

— KEY

Data type is MCDDataType::eKey (simple data type).

— LENGTHKEY

Data type is MCDDataType::eLengthKey (simple data type) which is only allowed in request.

— MULTIPLEXER

Data type is MCDDataType::eMULTIPLEXER (complex data type).

— NO_TYPE

No explicit data type.

— STRUCT_FIELD

Data type is MCDDataType::eSTRUCT_FIELD (complex data type)can only used inside complex Com Params.

— STRUCTURE

Data type is MCDDataType::eSTRUCTURE (complex data type).

— TABLE

Data type is MCDDataType::eTable (complex data type).

— TABLE_ROW

Data type is MCDDataType::eTABLE-ROW (complex data type).

— TEXTTABLE

Data type is MCDDataType::eTEXTTABLE (simple data type).

14.3.2.2 McdParameterTypes

Defines the constant ASAM types of parameters used by the MVCI/ODX system (MCDParameterType).

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

McdParameterTypes has the following enumeration values:

— CODED_CONST

Corresponds to the ODX parameter type MCDParameterType::eCODED-CONST.

— DYNAMIC

Corresponds to the ODX parameter type MCDParameterType::eDYNAMIC.

— GENERATED

Parameters of parameter type MCDParameterType::eGENERATED have been generated by the MCD-3 server. That is, these parameters do not have a correspondance in ODX.

— LENGTH_KEY

Corresponds to the ODX parameter type MCDParameterType::eLENGTH-KEY.

— MATCHING_REQUEST_PARAM

Corresponds to the ODX parameter type MCDParameterType::eMATCHING-REQUEST-PARAM.

— NRC_CONST

Corresponds to the ODX parameter type MCDParameterType::eNRC-CONST.

— PHYS_CONST

Corresponds to the ODX parameter type MCDParameterType::ePHYS-CONST.

— RESERVED

Corresponds to the ODX parameter type MCDParameterType::eRESERVED.

— SYSTEM

Corresponds to the ODX parameter type MCDParameterType::eSYSTEM.

— TABLE_ENTRY

Corresponds to the ODX parameter type MCDParameterType::eTABLE-ENTRY.

— TABLE_KEY

Corresponds to the ODX parameter type MCDParameterType::eTABLE-KEY.

— TABLE_STRUCT

Corresponds to the ODX parameter type MCDParameterType::eTABLE-STRUCT.

— VALUE

Corresponds to the ODX parameter type MCDParameterType::eVALUE.

14.3.2.3 McdResponseTypes

Defines the constant ASAM types of responses used by the MVCI/ODX system (MCDResponseType).

McdResponseTypes has the following enumeration values:

— GLOBAL_NEG_RESPONSE

The response should match the negative response template at the Location (MCDResponseType::eGLOBAL_NEG_RESPONSE).

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

— LOCAL_NEG_RESPONSE

The response should match the negative response template at the DiagComPrimitive (MCDResponseType::eLOCAL_NEG_RESPONSE).

— POSITIVE_RESPONSE

The response should match the positive response template (MCDResponseType::ePOSITIVE_RESPONSE).

14.4 Exceptions

14.4.1 Exceptions overview

All exceptions introduced by this extension are derived from the **otx:Exception** type. They represent the full set of exceptions added by the **OTX DiagDataBrowsingPlus extension**.

14.4.2 Exceptions syntax

The syntax of all **OTX DiagDataBrowsingPlus extension** exception-type declarations is shown in Figure 40.

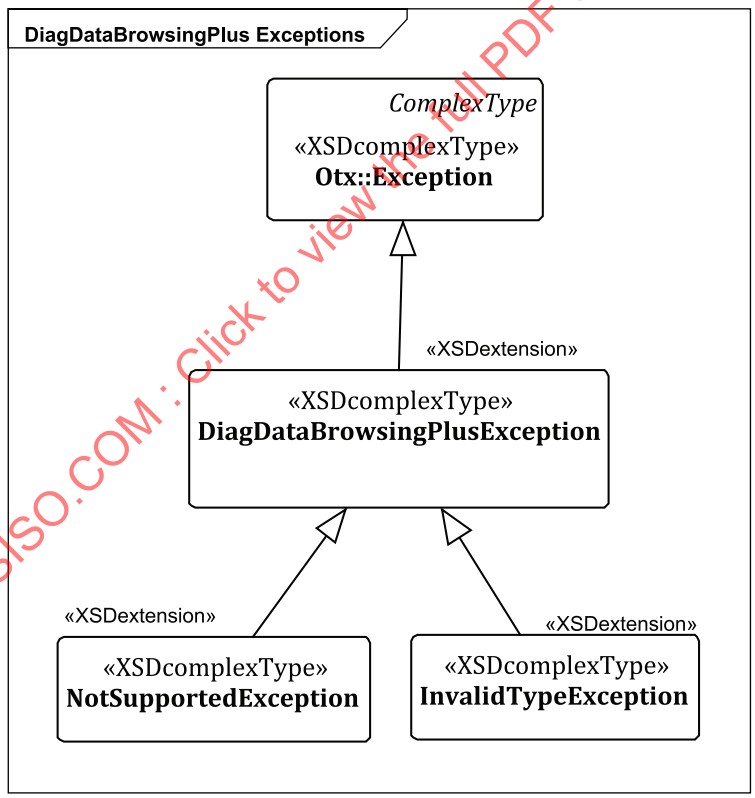


Figure 40 — DiagDataBrowsingPlus exceptions

14.4.3 Exceptions semantics

14.4.3.1 General

Since all **OTX DiagDataBrowsingPlus extension** exception types are implicit exceptions without initialisation parts, they cannot be declared constant.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

14.4.3.2 DiagDataBrowsingPlusException

The DiagDataBrowsingException is the super class for all exceptions in the **OTX DiagDataBrowsingPlus extension**. A DiagDataBrowsingException shall be used in case the more specific exception types described in the remainder of this section do not apply to the problem at hand (e.g. inconsistent database).

IMPORTANT — All terms and action realisations in this extension may potentially throw this exception.

14.4.3.3 InvalidTypeException

This is thrown if an element does not have the expected type, e.g. the parameter is not of type MCDParameterType::eTABLE_STRUCT.

14.4.3.4 NotSupportedException

This is thrown if the underlying runtime system (MVCI-Server) does not support this operation.

14.5 Variable access

14.5.1 Variable-access overview

All variable-access types introduced by this extension are derived from the OTX core variable extension interface.

14.5.2 Variable-access syntax

The syntax of the **OTX DiagDataBrowsingPlus extension** variable-access types is shown in Figure 41.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

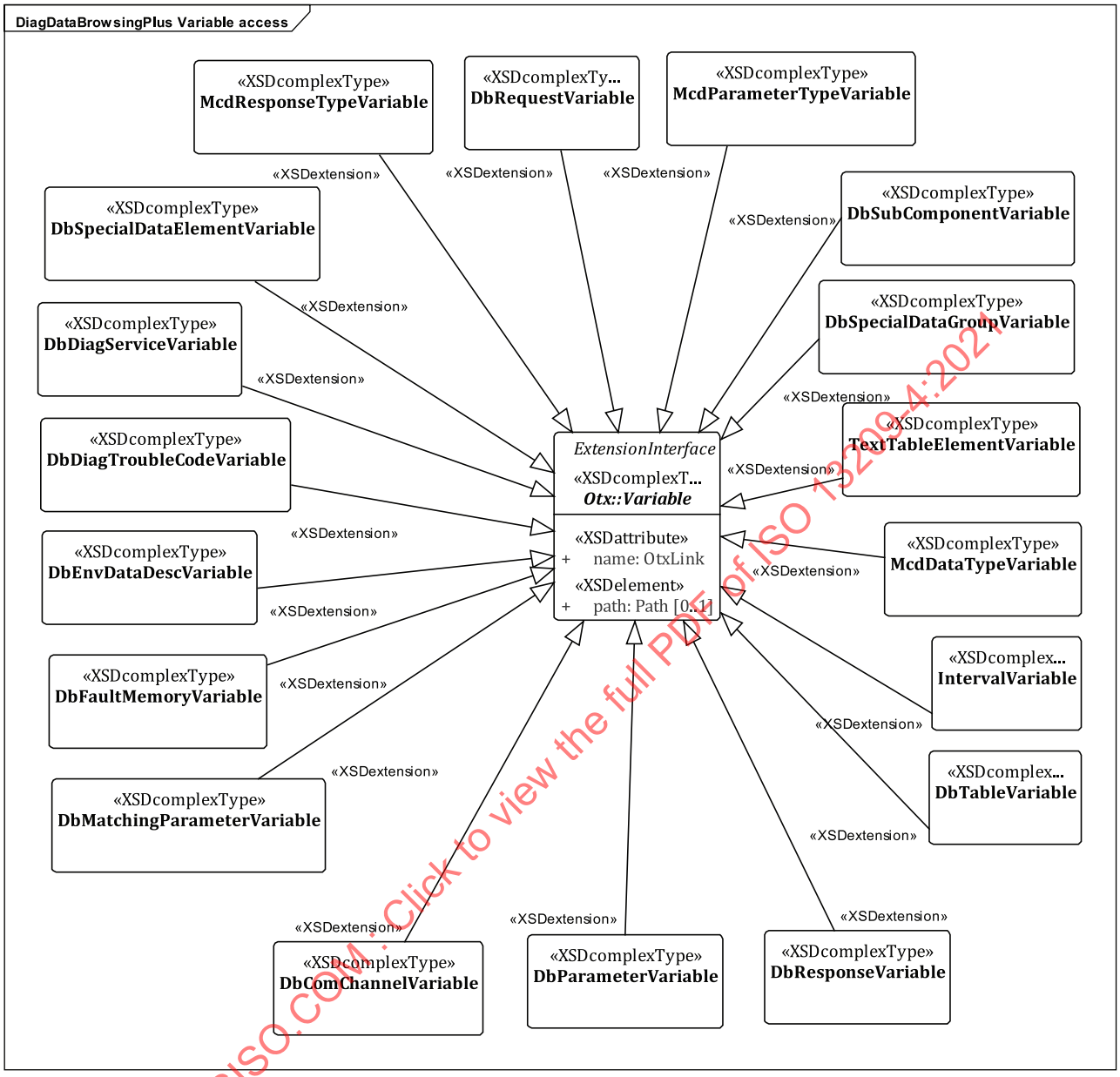


Figure 41 — DiagDataBrowsingPlus variable access

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

14.5.3 Variable-access semantics

The general semantics for all variable-access types shall apply. Please refer to ISO 13209-2 for details.

14.6 Terms

14.6.1 Terms overview

All terms introduced by this extension extend the **otx:Term** extension interface. Information about the specific super class of a term is provided in the individual term description clauses.

14.6.2 Semantics for DbComChannel terms

Figure 42 shows the syntax of the **OTX DiagDataBrowsingPlus extension DbComChannel** related terms.

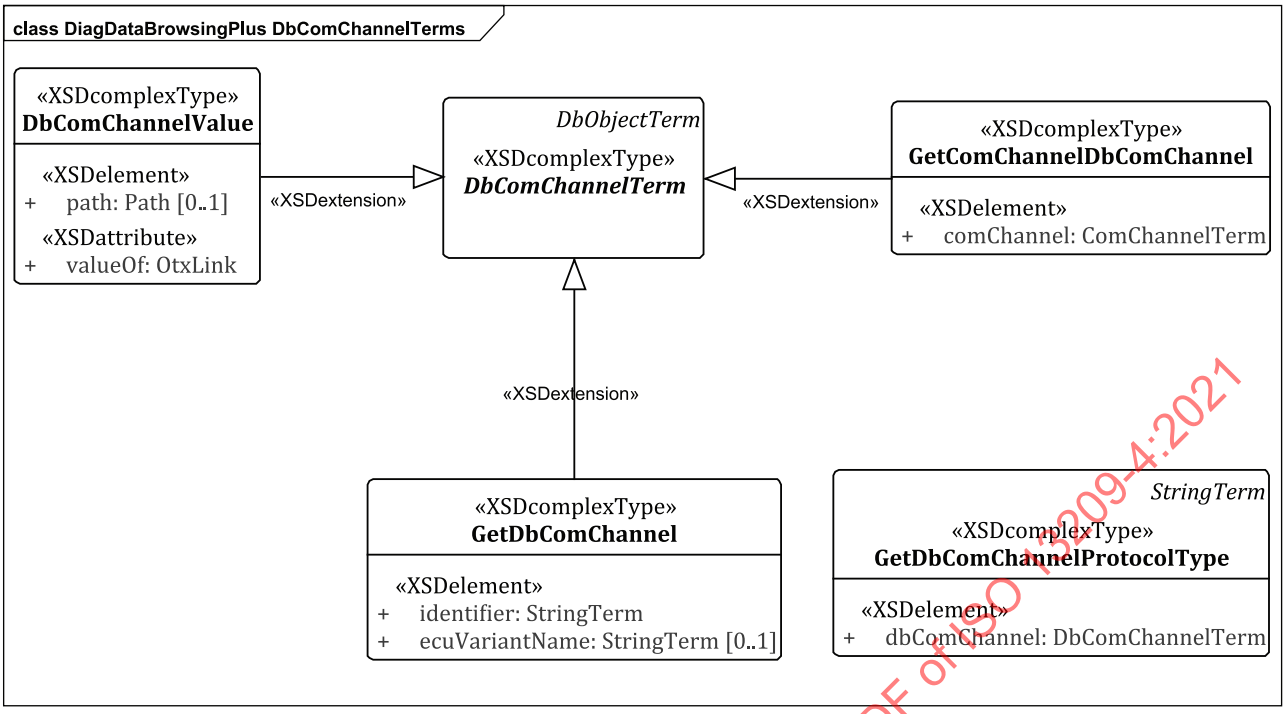


Figure 42 — DiagDataBrowsingPlus – DbComChannel terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

14.6.2.1 DbComChannelTerm

Semantic:

A **DbComChannelTerm** is a term that returns a **DbComChannel**.

14.6.2.2 DbComChannelValue

Semantic:

Returns the **DbComChannel** stored in a **DbComChannel** variable.

Return type:

diagDataPlus:DbComChannelTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

© ISO 2021 – All rights reserved

14.6.2.3 GetComChannelDbComChannel

Semantic:

Returns the the database part (MCDDbLocation) of a **diag:ComChannel** (MCDLogicalLink).

NOTE The appropriate MVCI system operation is MCDLogicalLink::getDbObject.getDbLocation.

Return type:

diagDataPlus:DbComChannelTerm

Parameters:

— <comChannel> : diag:ComChannelTerm [1..1]

This element represents the **diag:ComChannel** whose database part shall be returned.

Exceptions:

— None

14.6.2.4 GetDbComChannel

Semantic:

Returns a **DbComChannel** objects which represents the MCDDbLocation of an MCDDbLogicalLink which is identified by the SHORT-NAME of the related base variant and ECU variant.

NOTE The appropriate MVCI system operation is

MCDDbLocation baseVariant =

MCDSystem::getDbProjectConfiguration().MCDDbProjectConfiguration::getActiveDbProject().
MCDDbProject::getDbVehicleInformations().MCDDbVehicleInformations::getItemByNa
me(selectedVehicle).MCDDbVehicleInformation::getDbLogicalLinks().MCDDbLogicalL-
inks::getItemByName(identifier).MCDDbLogicalLink::getDbLocation(); if (ecuVariantName !=
empty) { return baseVariant.getDbECU().getDbEcuVariants().getItemByName(ecuVariantName); }
else { return baseVariant; }

Return type:

diagDataPlus:DbComChannelTerm

Parameters:

— <identifier> : otx:StringTerm [1..1]

This element represents a string identifying the base variant.

— <ecuVariantName> : otx:StringTerm [0..1]

This optional element represents a string identifying the ECU variant.

Exceptions:

diag:UnknownTargetException

— This exception is thrown if the identifier and the ecuVariantName does not exist or is invalid.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

14.6.2.5 GetDbComChannelProtocolType

Semantic:

Returns the protocol type as a string with the format as defined in the D-PDU API specification in version ISO 22900-2:—[3], Tables B.3 and B.4. These tables, however, do not only contain standardized names, vendor specific names can also be returned.

NOTE The related MVCI system method is

MCDSystem::getDbProjectConfiguration().MCDDbProjectConfiguration::getActiveDbProject(). MCDDbProject::getDbVehicleInformations().MCDDbVehicleInformations::getItemByNa me(selectedVehicle).MCDDbVehicleInformation::getDbLogicalLinks().MCDDbLogicalL- inks::getItemByName(identifier).getProtocolType().

Return type:

otx:StringTerm

Parameters:

— <dbComChannel> : diagDataPlus:DbComChannelTerm [1..1]

This element addresses the **DbComChannel** whose protocol type shall be returned.

Exceptions:

— None

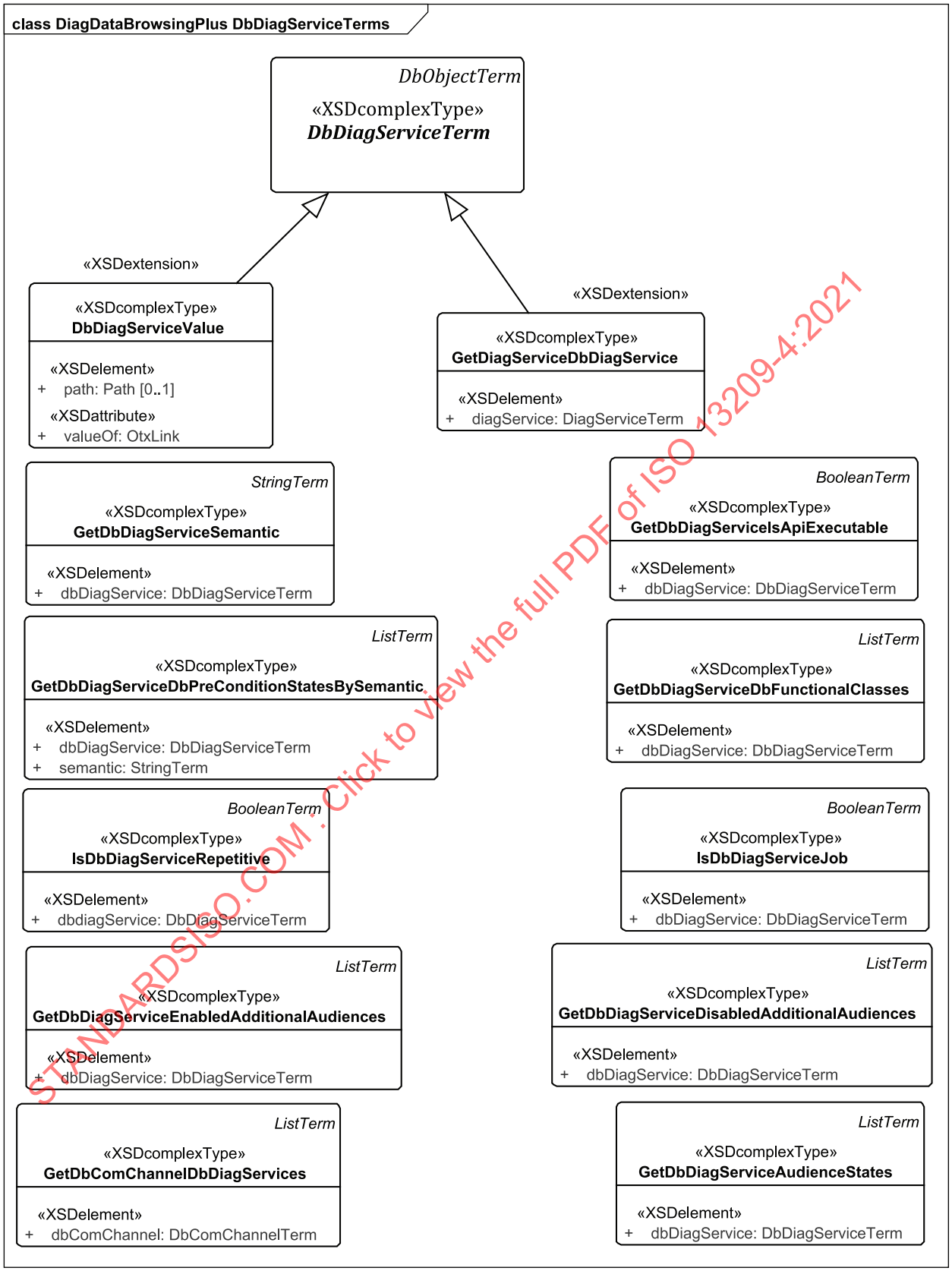
14.6.3 Semantics for DbDiagService terms

Figure 43 shows the syntax of the **OTX DiagDataBrowsingPlus extension DbDiagService** related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

3) Third edition under preparation. Stage at the time of publication: ISO/DIS 22900-2:2021.

© ISO 2021 – All rights reserved



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 43 — DiagDataBrowsingPlus – DbDiagService terms

© ISO 2021 – All rights reserved

14.6.3.1 DbDiagServiceTerm

Semantic:

A **DbDiagServiceTerm** is a term that returns a **DbDiagService**.

14.6.3.2 DbDiagServiceValue

Semantic:

Returns the **DbDiagService** stored in a **DbDiagService** variable.

Return type:

diagDataPlus:DbDiagServiceTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

14.6.3.3 GetDbComChannelDbDiagServices

Semantic:

Returns a list of **DbDiagService** objects which represents all MCDDbDiagComPrimitive for the given **DbComChannel**. If the **DbComChannel** has no associated MCDDbDiagComPrimitive elements, an empty list shall be returned.

NOTE The appropriate MVCI system operation is MCDDbLocation::getDbDiagComPrimitives.

Return type:

otx:ListTerm

Parameters:

— <dbComChannel> : diagDataPlus:DbComChannelTerm [1..1]

This element addresses the **DbComChannel** whose **DbDiagService** elements shall be returned.

Exceptions:

— None

14.6.3.4 GetDbDiagServiceAudienceStates

Semantic:

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Returns a list of strings of all audience attribute names, which are set to true for this DiagService in ODX. Valid names are "isAfterMarket", "isAfterSales", "isDevelopment", "isManufacturing", "isSupplier".

NOTE The appropriate MVCI system operation is MCDDbDataPrimitive::getAudienceState.

Return type:

otx:ListTerm

Parameters:

— <dbDiagService> : diagDataPlus:DbDiagServiceTerm [1..1]

This element addresses the **DbDiagService** whose audience states shall be returned.

Exceptions:

— None

14.6.3.5 GetDbDiagServiceDbFunctionalClasses

Semantic:

Returns a list of String of names of DbFunctionalClasses that the **DbDiagService** is assigned to within the project's databases. A Functional Class is a user-defined collection of Diagnostic Services. It can, for example be used to separate Services with different tasks. The length of the list is zero if the **DbDiagService** is not assigned with a Functional Class in the project's databases.

NOTE The appropriate MVCI system operation is MCDDbDiagComPrimitive::getDbFunctionalClasses.

Return type:

otx:ListTerm

Parameters:

— <dbDiagService> : diagDataPlus:DbDiagServiceTerm [1..1]

This element addresses the **DbDiagService** whose functional classes shall be returned.

Exceptions:

— None

14.6.3.6 GetDbDiagServiceDbPreConditionStatesBySemantic

Semantic:

Returns a list of String of ECU state (MCDDbEcuState) SHORT-NAMEs of the given semantic, one of which is required to be able to execute the given **DbDiagService** successfully. The result is reduced to states that have a semantic value equal to the given semantic. An empty list will be delivered if no ECU state of the given semantic is associated with the current **DbDiagService** as a pre-condition.

NOTE The appropriate MVCI system operation is MCDDbDiagComPrimitive::getDbPreConditionStatesBySemantic.

Return type:

otx:ListTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <dbDiagService> : diagDataPlus:DbDiagServiceTerm [1..1]

This element addresses the **DbDiagService** whose pre condition states shall be returned.

— <semantic> : otx:StringTerm [1..1]

This element addresses the semantic of an ECU state one of which is required to be able to execute an instance of the current DiagService. The semantic of an ECU state is given by the semantic definition of the MCDDbEcuStateChart object.

Exceptions:

— None

14.6.3.7 GetDbDiagServiceDisabledAdditionalAudiences

Semantic:

Returns a list of strings of the SHORT-NAMEs of all disabled "Additional Audiences" (MCDDbAdditionalAudiences) for this **DbDiagService** in ODX. An empty list will be delivered if no disabled audiences are defined.

NOTE The appropriate MVCI system operation is MCDDbDataPrimitive::getDbDisabledAdditionalAudiences.

Return type:

otx:ListTerm

Parameters:

— <dbDiagService> : diagDataPlus:DbDiagServiceTerm [1..1]

This element addresses the **DbDiagService** whose disabled additional audiences shall be returned.

Exceptions:

— None

14.6.3.8 GetDbDiagServiceEnabledAdditionalAudiences

Semantic:

Returns a list of strings of the SHORT-NAMEs of all enabled "Additional Audiences" (MCDDbAdditionalAudiences) for this **DbDiagService** in ODX. An empty list will be delivered if no enabled audiences are defined.

NOTE The appropriate MVCI system operation is MCDDbDataPrimitive::getDbEnabledAdditionalAudiences.

Return type:

otx:ListTerm

Parameters:

— <dbDiagService> : diagDataPlus:DbDiagServiceTerm [1..1]

This element addresses the **DbDiagService** whose enabled additional audiences shall be returned.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

14.6.3.9 GetDbDiagServiceIsApiExecutable

Semantic:

Returns the information from the database, if this **DbDiagService** is executable at the interface (true) or only within the D-System (false).

NOTE The appropriate MVCI system operation is MCDDbDiagComPrimitive::isApiExecutable.

Return type:

otx:BooleanTerm

Parameters:

— <dbDiagService> : diagDataPlus:DbDiagServiceTerm [1..1]

This element addresses the **DbDiagService** whose executable state shall be returned.

Exceptions:

— None

14.6.3.10 GetDbDiagServiceSemantic

Semantic:

Returns the value of the semantic attribute for that **DbDiagService** out of the ODX file as a string. The semantic desribes the diagnosis class the service belongs to, e.g. IO_Control, Read, Write or VariantIdentification.

NOTE The appropriate MVCI system operation is MCDDbDiagComPrimitive::getSemantic.

Return type:

otx:StringTerm

Parameters:

— <dbDiagService> : diagDataPlus:DbDiagServiceTerm [1..1]

This element addresses the **DbDiagService** whose semantic attribute shall be returned.

Exceptions:

— None

14.6.3.11 GetDiagServiceDbDiagService

Semantic:

Returns the the database part (MCDDbDiagComPrimitive) of a **diag:DiagService** (MCDDiagComPrimitive).

NOTE The appropriate MVCI system operation is MCDDiagComPrimitive::getDbObject.

Return type:

diagDataPlus:DbDiagServiceTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <diagService> : diag:DiagServiceTerm [1..1]

This element represents the **diag:DiagService** whose database part shall be returned.

Exceptions:

— None

14.6.3.12 IsDbDiagServiceJob

Semantic:

Returns true if the **DbDiagService** is a job otherwise false. It returns only true if the method getOjectType returns eMCDDBMULTIPLEECUJOB, eMCDDBFLASHJOB or eMCDDBSINGLEECUJOB.

NOTE The appropriate MVCI system operation is MCDDbDiagComPrimitive::getObjectType.

Return type:

otx:BooleanTerm

Parameters:

— <dbDiagService> : diagDataPlus:DbDiagServiceTerm [1..1]

This element addresses the **DbDiagService** which is checked to determine whether it is a job or not.

Exceptions:

— None

14.6.3.13 IsDbDiagServiceRepetitive

Semantic:

Returns true if MVCI-Server returns eREPEATED in getRepetitionMode, otherwise false. If getRepetitionMode is not supported and **NotSupportedException** shall be thrown.

NOTE The appropriate MVCI system operation is MCDDbDataPrimitive::getRepetitionMode.

Return type:

otx:BooleanTerm

Parameters:

— <dbdiagService> : diagDataPlus:DbDiagServiceTerm [1..1]

This element addresses the **DbDiagService** which is checked whether it is repetitive or not.

Exceptions:

NotSupportedException

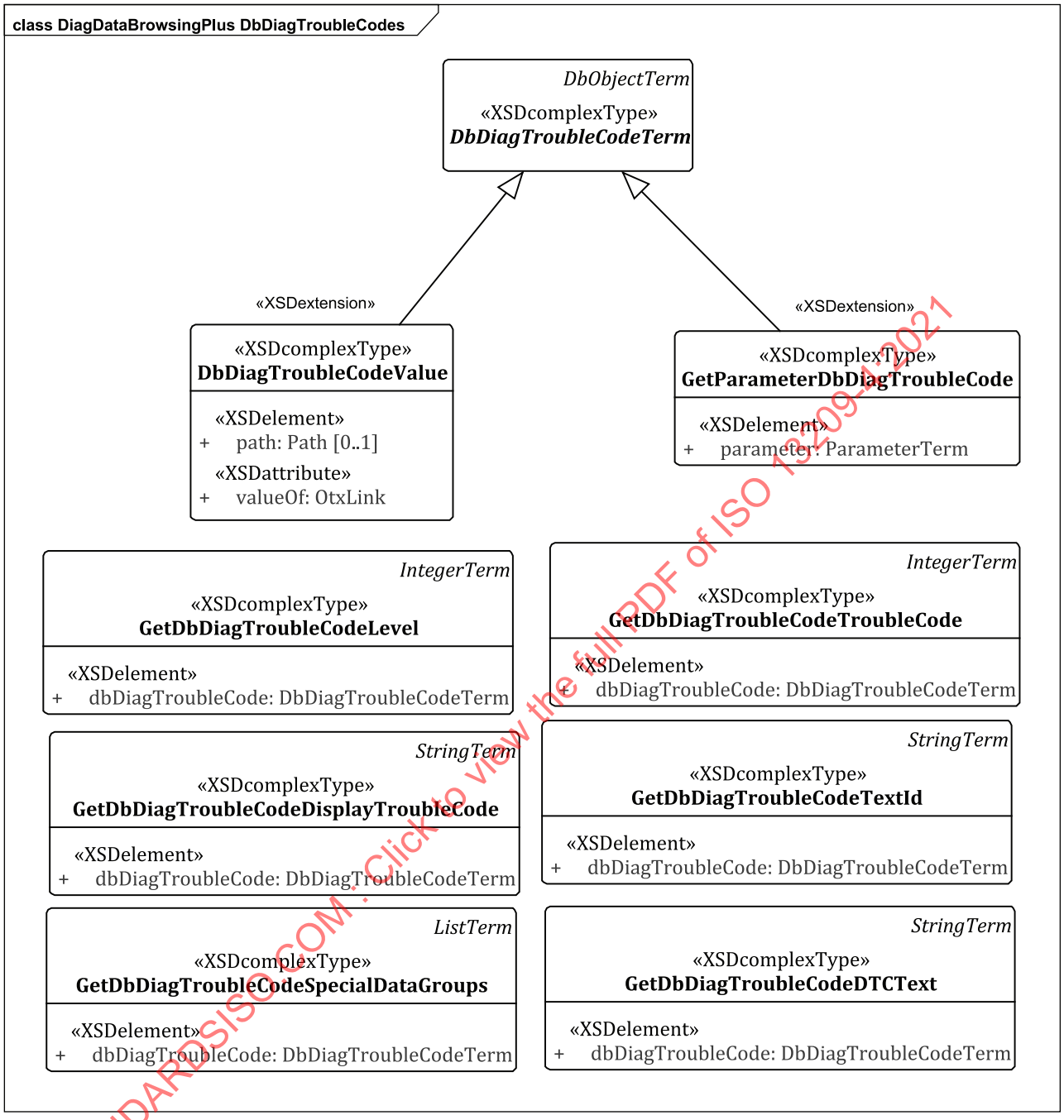
— This exception is thrown if getRepetitionMode is not supported by MVCI-Server.

14.6.4 Semantics for DbDiagTroubleCode terms

Figure 44 shows the syntax of the **OTX DiagDataBrowsingPlus extension DbDiagTroubleCode** related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 44 — DiagDataBrowsingPlus – DbDiagTroubleCode terms

14.6.4.1 DbDiagTroubleCodeTerm

Semantic:

A **DbDiagTroubleCodeTerm** is a term that returns a **DbDiagTroubleCode**.

14.6.4.2 DbDiagTroubleCodeValue

Semantic:

Returns the **DbDiagTroubleCode** stored in a **DbDiagTroubleCode** variable.

Return type:

© ISO 2021 – All rights reserved

diagDataPlus:DbDiagTroubleCodeTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

14.6.4.3 GetDbDiagTroubleCodeDisplayTroubleCode

Semantic:

Returns the display trouble code of **DbDiagTroubleCode**.

NOTE The appropriate MVCI system operation is MCDDbDiagTroubleCode::getDisplayTroubleCode.

Return type:

otx:StringTerm

Parameters:

— <dbDiagTroubleCode> : diagDataPlus:DbDiagTroubleCodeTerm [1..1]

This element addresses the **DbDiagTroubleCode** (MCDDbDiagTroubleCode) whose display trouble code shall be returned.

Exceptions:

— None

14.6.4.4 GetDbDiagTroubleCodeDTCText

Semantic:

Returns the DTC text of **DbDiagTroubleCode**.

NOTE The appropriate MVCI system operation is MCDDbDiagTroubleCode::getUnicodeDTCText.

Return type:

otx:StringTerm

Parameters:

— <dbDiagTroubleCode> : diagDataPlus:DbDiagTroubleCodeTerm [1..1]

This element addresses the **DbDiagTroubleCode** (MCDDbDiagTroubleCode) whose DTC text shall be returned.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Exceptions:

— None

14.6.4.5 GetDbDiagTroubleCodeLevel

Semantic:

Returns the level of **DbDiagTroubleCode**.

NOTE The appropriate MVCI system operation is MCDDbDiagTroubleCode::getLevel.

Return type:

otx:IntegerTerm

Parameters:

— <dbDiagTroubleCode> : diagDataPlus:DbDiagTroubleCodeTerm [1..1]

This element addresses the **DbDiagTroubleCode** (MCDDbDiagTroubleCode) whose level shall be returned.

Exceptions:

— None

14.6.4.6 GetDbDiagTroubleCodeSpecialDataGroups

Semantic:

Returns the special data groups defined for the corresponding DTC in the project's databases as a list of **DbSpecialDataGroup** objects. An empty list will be delivered if no DbSDGs available in the ODX data.

NOTE The appropriate MVCI system operation is MCDDbDiagTroubleCode::getDbSDGs.

Return type:

otx:ListTerm

Parameters:

— <dbDiagTroubleCode> : diagDataPlus:DbDiagTroubleCodeTerm [1..1]

This element addresses the **DbDiagTroubleCode** (MCDDbDiagTroubleCode) whose special data groups shall be returned.

Exceptions:

— None

14.6.4.7 GetDbDiagTroubleCodeTextId

Semantic:

Returns the text ID of the **DbDiagTroubleCode** from ODX as a string. If the TextID cannot be obtained from ODX, an empty string is returned. There are two different text IDs in ODX, one in the SHORTLABEL and one in the VT. Only the TI of the VT will be considered. The returned value represents the TI of the physical value as selected by the server. For a request parameter, the TI may be empty in the case where no valid TI can be determined.

NOTE The appropriate MVCI system operation is MCDDbDiagTroubleCode::getDiagTroubleCodeTextID.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Return type:

otx:StringTerm

Parameters:

— <dbDiagTroubleCode> : diagDataPlus:DbDiagTroubleCodeTerm [1..1]

This element addresses the **DbDiagTroubleCode** (MCDDbDiagTroubleCode) whose TI shall be returned.

Exceptions:

— None

14.6.4.8 GetDbDiagTroubleCodeTroubleCode

Semantic:

Returns the trouble code of **DbDiagTroubleCode** as an Integer.

NOTE The appropriate MVCI system operation is MCDDbDiagTroubleCode::getTroubleCode.

Return type:

otx:IntegerTerm

Parameters:

— <dbDiagTroubleCode> : diagDataPlus:DbDiagTroubleCodeTerm [1..1]

This element addresses the **DbDiagTroubleCode** (MCDDbDiagTroubleCode) whose trouble code shall be returned.

Exceptions:

— None

14.6.4.9 GetParameterDbDiagTroubleCode

Semantic:

Returns the **DbDiagTroubleCode** of the response **diag:Parameter.**

NOTE The appropriate MVCI system operation is MCDResponseParameter::getDbDTC.

Return type:

diagDataPlus:DbDiagTroubleCodeTerm

Parameters:

— <parameter> : diag:ParameterTerm [1..1]

This element addresses the response **diag:Parameter** (MCDResponseParameter) whose DTC shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the **diag:Parameter** is not a response parameter or the response parameter does not have a DTC.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

14.6.5 Semantics for DbEcuVariant terms

Figure 45 shows the syntax of the **OTX DiagDataBrowsingPlus extension DbEcuVariant** related terms.

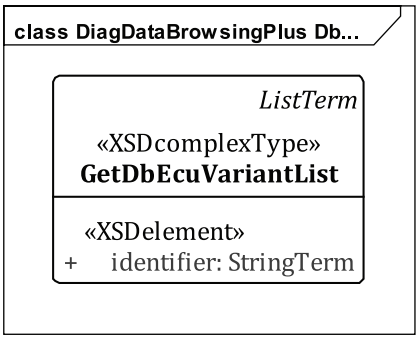


Figure 45 — DiagDataBrowsingPlus – DbEcuVariant terms

14.6.5.1 GetDbEcuVariantList

Semantic:

Returns a list of strings which represents the names of all ECU variants for a given logical link. The identifier shall point to a base variant. If a base variant has no associated ECU variants, an empty list shall be returned. Furthermore, in case the communication channel points to a protocol or functional group, an exception shall be thrown.

NOTE The appropriate MVCI system operation is MCDDbLogicalLink::getDbLocation::getDbECU::getDbEcuVariants.

Return type:

otx:ListTerm

Parameters:

— <identifier> : otx:StringTerm [1..1]

This element represents a string identifying the communication channel which provides the data. The identifier can be retrieved by the term **data:GetComChannelList,** see ISO 13209-3:2012, 8.4.3.1.

Exceptions:

diag:UnknownTargetException

— This exception is thrown if the identifier does not exist or is invalid.

14.6.6 Semantics for DbEnvDataDesc terms

Figure 46 shows the syntax of the **OTX DiagDataBrowsingPlus extension DbEnvDataDesc** related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

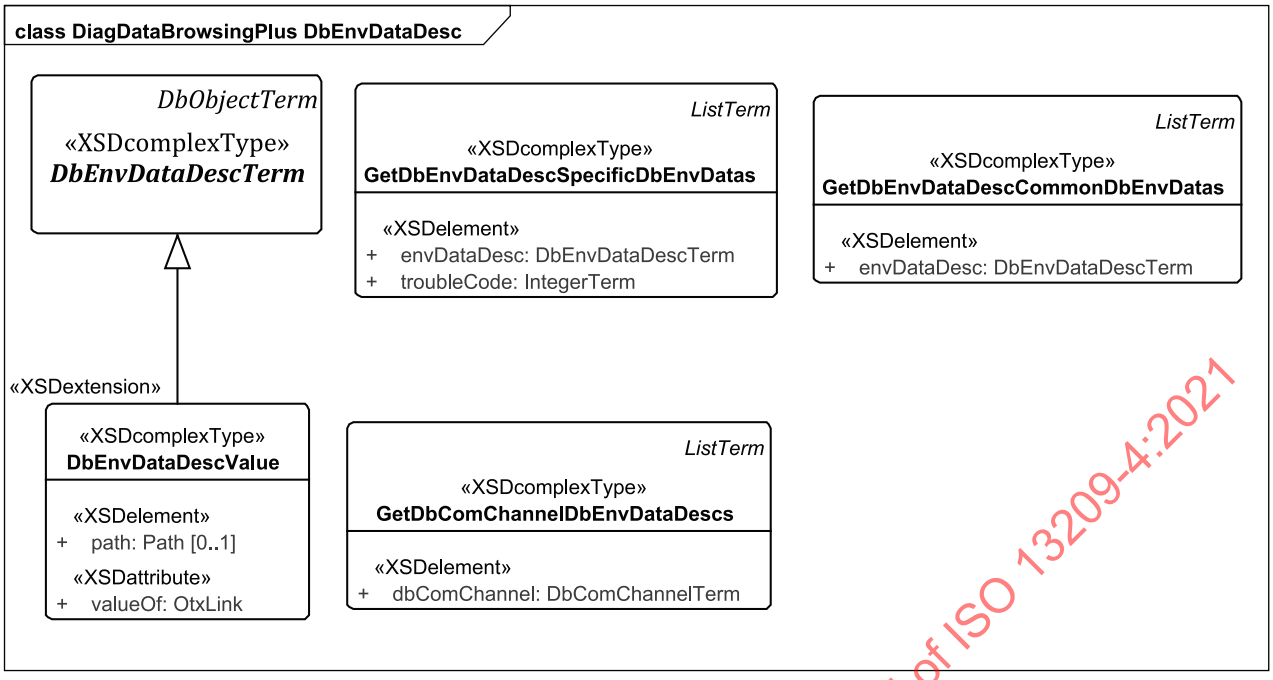


Figure 46 — DiagDataBrowsingPlus – DbEnvDataDesc terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

14.6.6.1 DbEnvDataDescTerm

Semantic:

A **DbEnvDataDescTerm** is a term that returns a **DbEnvDataDesc**.

14.6.6.2 DbEnvDataDescValue

Semantic:

Returns the **DbEnvDataDesc** stored in a **DbEnvDataDesc** variable.

Return type:

diagDataPlus:DbEnvDataDescTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

© ISO 2021 – All rights reserved

14.6.6.3 GetDbComChannelDbEnvDataDescs

Semantic:

Returns a list of all **DbEnvDataDesc** (MCDDbEnvDataDesc) referenced by the **DbComChannel**. The list can be empty.

NOTE The appropriate MVCI system operation is MCDDbLocation::getDbEnvDataDescs.

Return type:

otx:ListTerm

Parameters:

— <dbComChannel> : diagDataPlus:DbComChannelTerm [1..1]

This element addresses the **ComChannel** whose environment data descriptions shall be returned.

Exceptions:

— None

14.6.6.4 GetDbEnvDataDescCommonDbEnvDatas

Semantic:

Returns a list of **DbParameter** (MCDDbResponseParameters) which has ALL-VALUE flag. If no elements are available in database, then an empty list will be delivered.

NOTE The appropriate MVCI system operation is MCDDbEnvDataDesc::getCommonDbEnvDatas.

Return type:

otx:ListTerm

Parameters:

— <envDataDesc> : diagDataPlus:DbEnvDataDescTerm [1..1]

This element addresses the **DbEnvDataDesc** whose common environment data shall be returned.

Exceptions:

— None

14.6.6.5 GetDbEnvDataDescSpecificDbEnvDatas

Semantic:

Returns a list of **DbParameter** (MCDDbResponseParameters) that match the given trouble code. If no elements are available in the database, an empty list will be delivered.

NOTE The appropriate MVCI system operation is MCDDbEnvDataDesc::getSpecificDbEnvDatasForDiagTroubleCode.

Return type:

otx:ListTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <envDataDesc> : diagDataPlus:DbEnvDataDescTerm [1..1]

This element addresses the **DbEnvDataDesc** whose specific environment data shall be returned.

— <troubleCode> : otx:IntegerTerm [1..1]

This element addresses the trouble code from which the parameter shall be returned.

Exceptions:

— None

14.6.7 Semantics for DbFaultMemory terms

Figure 47 shows the syntax of the **OTX DiagDataBrowsingPlus extension DbFaultMemory** related terms.

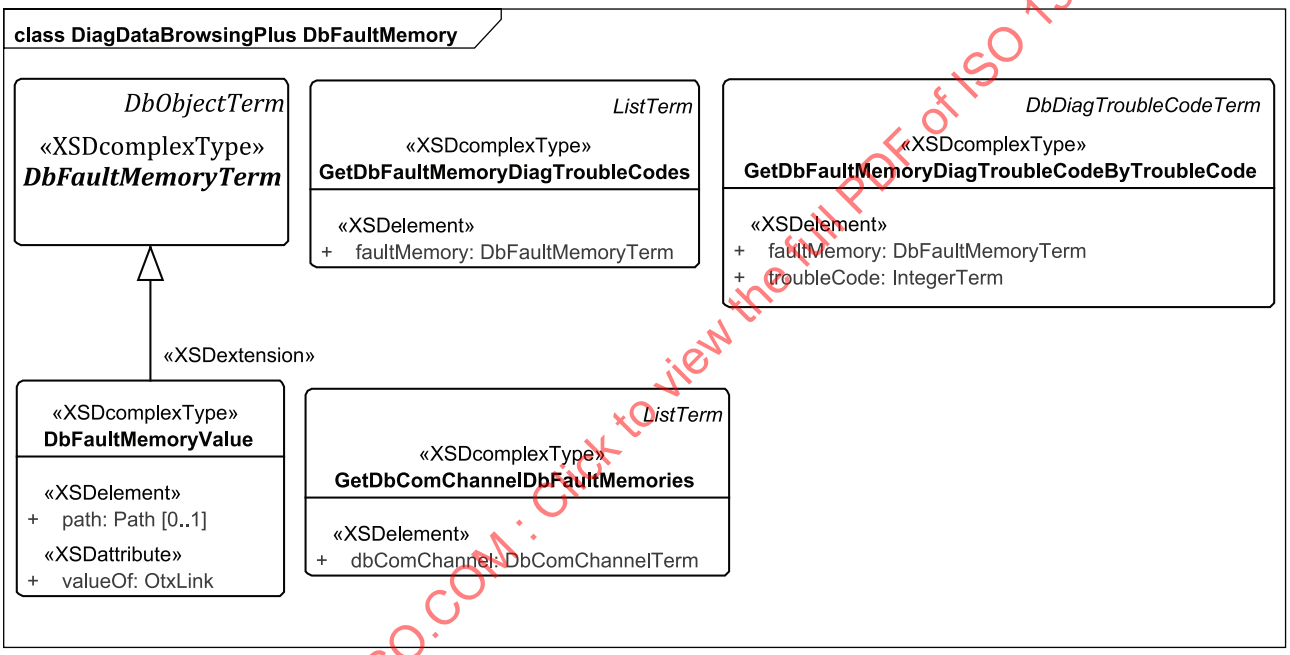


Figure 47 — DiagDataBrowsingPlus – DbFaultMemory terms

14.6.7.1 DbFaultMemoryTerm

Semantic:

A **DbFaultMemoryTerm** is a term that returns a **DbFaultMemory**.

14.6.7.2 DbFaultMemoryValue

Semantic:

Returns the **DbFaultMemory** stored in a **DbFaultMemory** variable.

Return type:

diagDataPlus:DbFaultMemoryTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

14.6.7.3 GetDbComChannelDbFaultMemories

Semantic:

Returns a list of all **DbFaultMemory** (McdDbFaultMemory) referenced by the **DbComChannel**. The list can be empty.

NOTE The appropriate MVCI system operation is MCDDbLocation::getDbFaultMemories.

Return type:

otx:ListTerm

Parameters:

— <dbComChannel> : diagDataPlus:DbComChannelTerm [1..1]

This element addresses the **ComChannel** whose fault memories shall be returned.

Exceptions:

— None

14.6.7.4 GetDbFaultMemoryDiagTroubleCodeByTroubleCode

Semantic:

Returns the **DbDiagTroubleCode** (MCDDbDiagTroubleCode) object which matches the given trouble code.

NOTE The appropriate MVCI system operation is MCDDbFaultMemory::getDbDiagTroubleCodeByTroubleCode.

Return type:

diagDataPlus:DbDiagTroubleCodeTerm

Parameters:

— <faultMemory> : diagDataPlus:DbFaultMemoryTerm [1..1]

This element addresses the **DbFaultMemory** whose **DbDiagTroubleCode** shall be returned.

— <troubleCode> : otx:IntegerTerm [1..1]

This element addresses the trouble code from which the **DbDiagTroubleCode** object shall be returned.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Exceptions:

InvalidTypeException

— This exception is thrown if the trouble code (Integer) cannot be determined.

14.6.7.5 GetDbFaultMemoryDiagTroubleCodes

Semantic:

Returns a list of **DbDiagTroubleCode** (MCDDbDiagTroubleCode) for the **DbFaultMemory** which is available in ODX. If no elements are available in the database, then an empty list will be returned.

NOTE The appropriate MVCI system operation is MCDDbFaultMemory::getDbDiagTroubleCodes.

Return type:

otx:ListTerm

Parameters:

— <faultMemory> : diagDataPlus:DbFaultMemoryTerm [1..1]

This element addresses the **DbFaultMemory** whose DbDiagTroubleCodes shall be returned.

Exceptions:

— None

14.6.8 Semantics for DbMatchingParameter terms

Figure 48 shows the syntax of the **OTX DiagDataBrowsingPlus extension DbMatchingParameter** related terms.

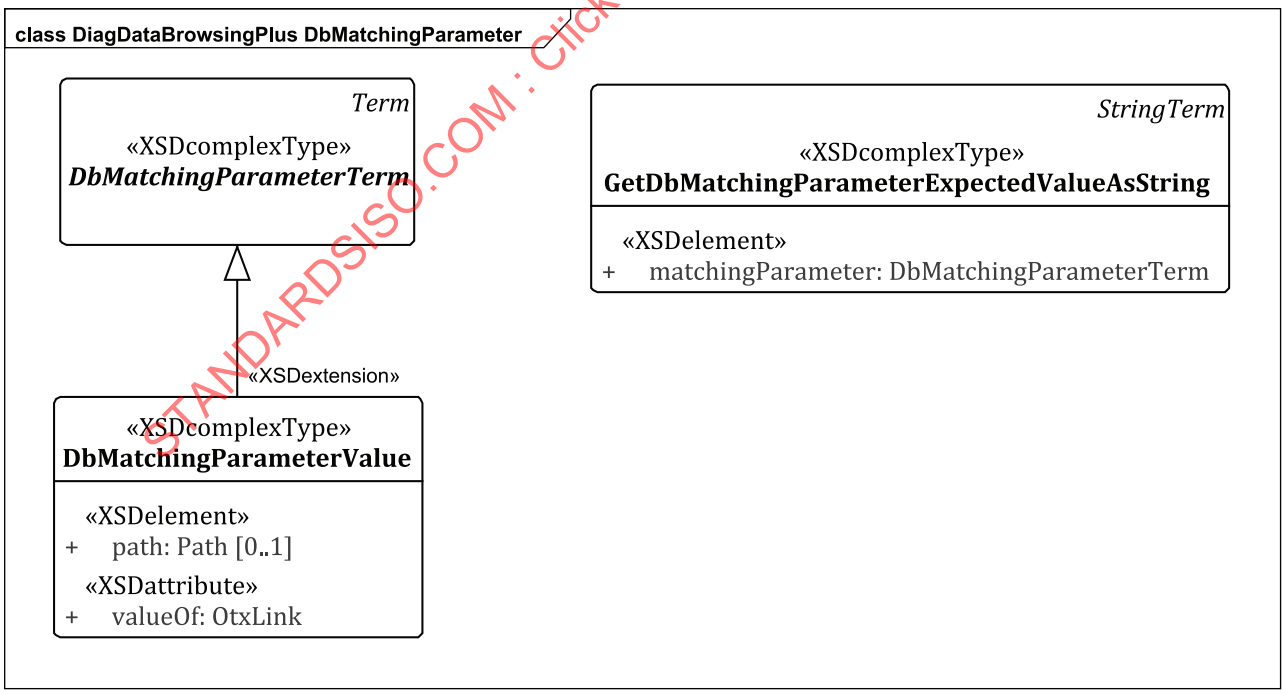


Figure 48 — DiagDataBrowsingPlus – DbMatchingParameter terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

14.6.8.1 DbMatchingParameterTerm

Semantic:

A **DbMatchingParameterTerm** is a term that returns a **DbMatchingParameter**.

14.6.8.2 DbMatchingParameterValue

Semantic:

Returns the **DbMatchingParameter** stored in a **DbMatchingParameter** variable.

Return type:

diagDataPlus:DbMatchingParameterTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

14.6.8.3 GetDbMatchingParameterExpectedValueAsString

Semantic:

Returns the expected value of the matching parameter as a string.

NOTE The appropriate MVCI system operation is MCDDbMatchingParameter::getExpectedValue::getValueAsString.

Return type:

otx:StringTerm

Parameters:

— <matchingParameter> : diagDataPlus:DbMatchingParameterTerm [1..1]

This element addresses the **DbMatchingParameter** whose expected value shall be returned.

Exceptions:

— None

14.6.9 Semantics for DbObject terms

Figure 49 shows the syntax of the **OTX DiagDataBrowsingPlus extension DbObject** related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

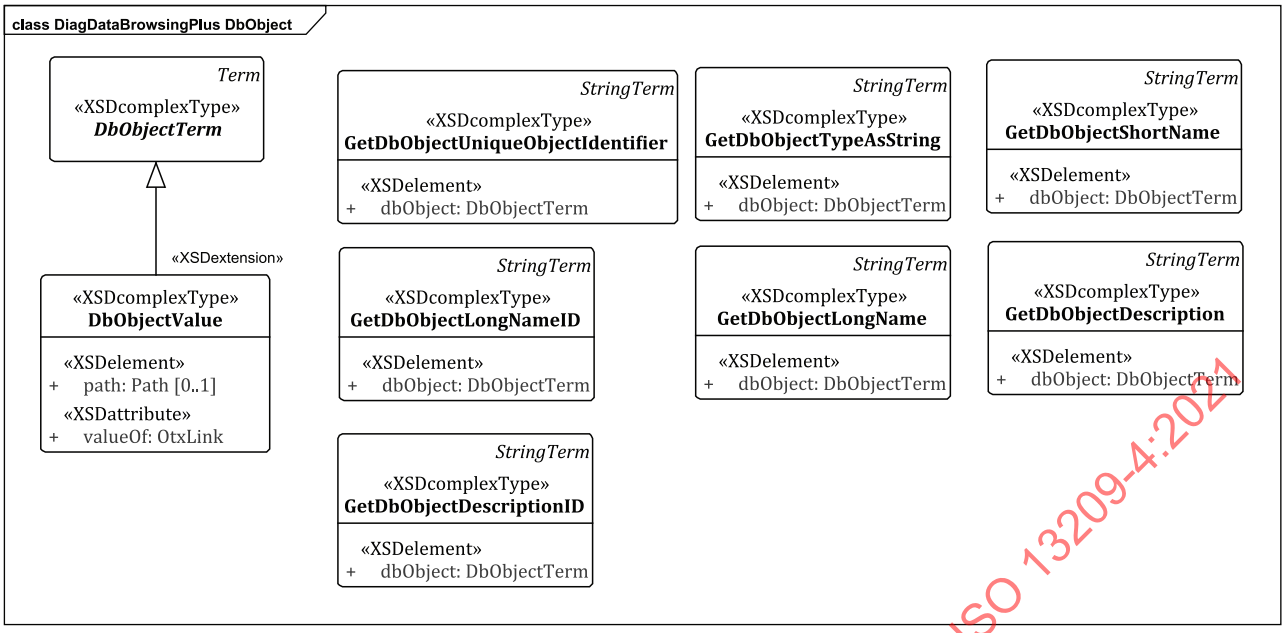


Figure 49 — DiagDataBrowsingPlus – DbObject terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

14.6.9.1 DbObjectTerm

Semantic:

A **DbObjectTerm** is a term that returns a **DbObject**.

14.6.9.2 DbObjectValue

Semantic:

Returns the **DbObject** stored in a **DbObject** variable.

Return type:

diagDataPlus:DbObjectTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

14.6.9.3 GetDbObjectDescription

Semantic:

© ISO 2021 – All rights reserved

Returns the Description of the object which can also be empty (Description is not mandatory).

NOTE The appropriate MVCI system operation is MCDNamedObject::getDescription.

Return type:

otx:StringTerm

Parameters:

— <dbObject> : diagDataPlus:DbObjectTerm [1..1]

This element addresses the dbObject whose description shall be returned.

Exceptions:

— None

14.6.9.4 GetDbObjectDescriptionID

Semantic:

Returns the ID for the description of the object as a string, which is made for multi language purposes. It returns the ID for a text in a texttable, in each language texttable the ID points to the same content but different language content. The return value can be empty if the ID cannot be determined.

NOTE The appropriate MVCI system operation is MCDDbObject::getDescriptionID.

Return type:

otx:StringTerm

Parameters:

— <dbObject> : diagDataPlus:DbObjectTerm [1..1]

This element addresses the dbObject whose description ID shall be returned.

Exceptions:

— None

14.6.9.5 GetDbObjectLongName

Semantic:

Returns the LONG-NAME of the object as a string.

NOTE The appropriate MVCI system operation is MCDNamedObject::getLongName.

Return type:

otx:StringTerm

Parameters:

— <dbObject> : diagDataPlus:DbObjectTerm [1..1]

This element addresses the dbObject whose LONG-NAME shall be returned.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

14.6.9.6 GetDbObjectLongNameID

Semantic:

Returns the ID for the LONG-NAME of the object as a string, which is made for multi language purposes. It returns the ID for a text in a texttable, in each language texttable the ID points to the same content but different language content. The return value can be empty if the ID cannot be determined.

NOTE The appropriate MVCI system operation is MCDDbObject::getLongNameID.

Return type:

otx:StringTerm

Parameters:

— <dbObject> : diagDataPlus:DbObjectTerm [1..1]

This element addresses the dbObject whose LONG-NAME ID shall be returned.

Exceptions:

— None

14.6.9.7 GetDbObjectShortName

Semantic:

Returns the SHORT-NAME of the object. If no SHORT-NAME exists (e.g. in Caption of DbSpecialDataGroup) an empty string shall be returned.

NOTE The appropriate MVCI system operation is MCDNamedObject::getShortName.

Return type:

otx:StringTerm

Parameters:

— <dbObject> : diagDataPlus:DbObjectTerm [1..1]

This element addresses the dbObject whose SHORT-NAME shall be returned.

Exceptions:

— None

14.6.9.8 GetDbObjectTypeAsString

Semantic:

Returns the MCDObjectType of the related MCDObject as string, e.g. "eMCDDBTABLE".

NOTE The appropriate MVCI system operation is MCDObject::getObjectType.

Return type:

otx:StringTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Parameters:

— <dbObject> : diagDataPlus:DbObjectTerm [1..1]

This element addresses the dbObject whose type shall be returned.

Exceptions:

— None

14.6.9.9 GetDbObjectUniqueObjectIdentifier

Semantic:

Returns the unique object ID of the object as a string. If the identifier is not defined an empty string will be returned.

NOTE The appropriate MVCI system operation is MCDDbObject::getUniqueObjectIdentifier.

Return type:

otx:StringTerm

Parameters:

— <dbObject> : diagDataPlus:DbObjectTerm [1..1]

This element addresses the dbObject whose unique object identifier shall be returned.

Exceptions:

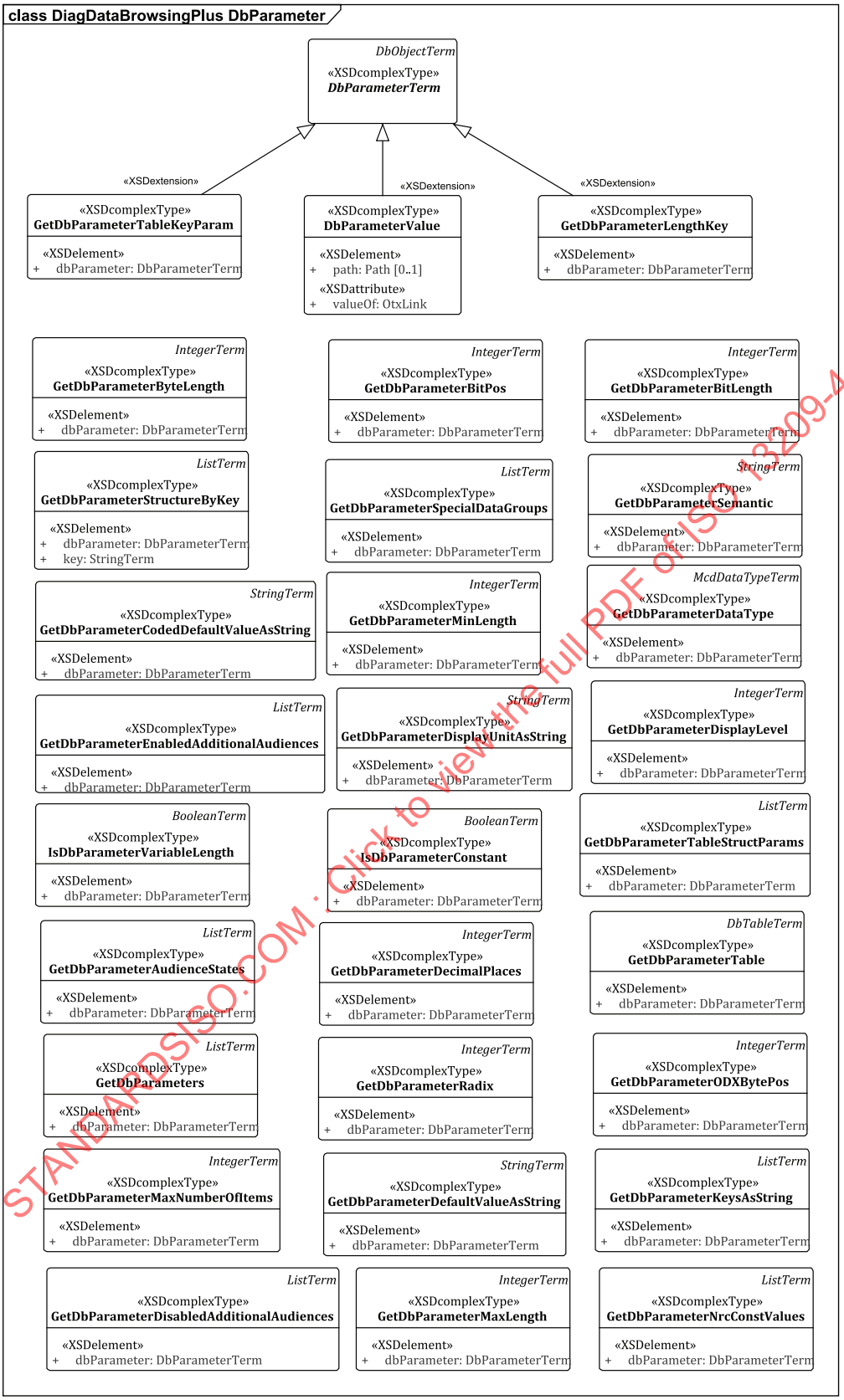
— None

14.6.10 Semantics for DbParameter terms

Figure 50 shows the syntax of the **OTX DiagDataBrowsingPlus extension DbParameter** related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 50 — DiagDataBrowsingPlus – DbParameter terms

© ISO 2021 – All rights reserved

14.6.10.1 DbParameterTerm

Semantic:

A **DbParameterTerm** is a term that returns a **DbParameter**.

14.6.10.2 DbParameterValue

Semantic:

Returns the **DbParameter** stored in a **DbParameter** variable.

Return type:

diagDataPlus:DbParameterTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

14.6.10.3 GetDbParameterAudienceStates

Semantic:

Returns a list of strings of all Audiences (Valid names are "isAfterMarket", "isAfterSales", "isDevelopment", "isManufacturing", "isSupplier") which are marked as applicable (true) for this parameter.

NOTE The appropriate MVCI system operation is MCDDbTableParameter::getAudienceState.

Return type:

otx:ListTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbTableParameter) whose Audiences shall be returned.

Exceptions:

— None

14.6.10.4 GetDbParameterBitLength

Semantic:

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Returns the bit-length of the parameter in the PDU as defined in ODX.

NOTE The appropriate MVCI system operation is MCDDbParameter::getBitLength.

Return type:

otx:IntegerTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose bit length shall be returned.

Exceptions:

— None

14.6.10.5 GetDbParameterBitPos

Semantic:

Returns the bit position of the parameter. The counting starts with 0. It returns the Bit where the parameter in the byte starts. The byte should be determined by **GetDbParameterODXBytePos**.

NOTE The appropriate MVCI system operation is MCDDbParameter::getBitPos.

Return type:

otx:IntegerTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose bit position shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter is not a float value (MCDDataType::FLOAT32 or MCDDataType::FLOAT64).

14.6.10.6 GetDbParameterByteLength

Semantic:

Returns the byte length of the parameter in the PDU.

NOTE The appropriate MVCI system operation is MCDDbParameter::getByteLength.

Return type:

otx:IntegerTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose byte length shall be returned.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Exceptions:

— None

14.6.10.7 GetDbParameterCodedDefaultValueAsString

Semantic:

Determines the coded default value of the parameter as a string. In case of a CODED-CONST parameter, the corresponding CODED-VALUE is returned. In case of a PHYS-CONST parameter, the coded default value is computed from the the physical value.

NOTE The appropriate MVCI system operation is MCDDbParameter::getCodedDefaultValue::getValueAsString.

Return type:

otx:StringTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose coded default value shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the coded default value of the parameter cannot be determined (e.g. the parameter is of no simple data type).

14.6.10.8 GetDbParameterDataType

Semantic:

Returns the data type of the parameter.

NOTE The appropriate MVCI system operation is MCDDbParameter::getDataType.

Return type:

diagDataPlus:McdDataTypeTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose data type shall be returned.

Exceptions:

— None

14.6.10.9 GetDbParameterDecimalPlaces

Semantic:

Returns the number of valid decimal places for the parameter. The number of decimal places shall be used by the client to display only the valid decimal places of a float parameter as defined in the database. According to ODX, the number of decimal places is defined as the number of valid digits after the decimal point and not as precision of the floating point value.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

NOTE The appropriate MVCI system operation is MCDDbParameter::getDecimalPlaces.

Return type:

otx:IntegerTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose decimal places shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter is not a float value (MCDDataType::FLOAT32 or MCDDataType::FLOAT64).

14.6.10.10 GetDbParameterDefaultValueAsString

Semantic:

Returns the compu default value of the parameter as string.

NOTE The appropriate MVCI system operation is MCDDbParameter::getDefaultValue::getValueAsString.

Return type:

otx:StringTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose default value shall be returned.

Exceptions:

InvalidTypeException

— If the parameter is a complex data type, the default value of the parameter cannot be determined and the parameter type is NRC-CONST.

14.6.10.11 GetDbParameterDisabledAdditionalAudiences

Semantic:

Returns a list of strings (SHORT-NAMEs) of all disabled Additional Audiences. An empty list is returned if there are no MCDDbAdditionalAudiences available.

NOTE The appropriate MVCI system operation is MCDDbTableParameter::getDbDisabledAdditionalAudiences[]::getShortname.

Return type:

otx:ListTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbTableParameter) for whom disabled Additional Audiences shall be returned.

Exceptions:

— None

14.6.10.12 GetDbParameterDisplayLevel

Semantic:

Returns the display level of the parameter read from ODX. Returns zero if no display level is available.

NOTE The appropriate MVCI system operation is MCDDbParameter::getDisplayLevel.

Return type:

otx:IntegerTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose display level shall be returned.

Exceptions:

— None

14.6.10.13 GetDbParameterDisplayUnitAsString

Semantic:

Returns the display unit information of the **DbParameter** as a string. If the parameter is not of simple data type or does not contain any unit information an empty string is returned. The unit is only available for parameters with simple data types.

NOTE The appropriate MVCI system operation is MCDDbParameter::getUnit.

Return type:

otx:StringTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose display unit shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter is not of simple data type A_UINT (8-64), A_INT (8-64), FLOAT (32,64), TEXTTABLE, BITFIELD, ASCIISTRING, UNICODE2STRING or BYTEFIELD.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

14.6.10.14 GetDbParameterEnabledAdditionalAudiences

Semantic:

Returns a list of strings (SHORT-NAMEs) of all enabled Additional Audiences. An empty list is returned if there are no MCDDbAdditionalAudiences available.

NOTE The appropriate MVCI system operation is MCDDbTableParameter::getDbEnabledAdditionalAudiences[]::getShortname.

Return type:

otx:ListTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbTableParameter) whose enabled Additional Audiences shall be returned.

Exceptions:

— None

14.6.10.15 GetDbParameterKeysAsString

Semantic:

Returns a list of strings. If the corresponding TABLE-KEY references a TABLE (dynamic case), this method delivers all keys of this TABLE (keys of all TABLE-ROWs). If the corresponding TABLE-KEY references a TABLE-ROW directly (static case), this method delivers exactly one key – the key of that TABLE-ROW. The length of the list is zero, if no keys are available.

NOTE The appropriate MVCI system operation is MCDDbParameter::getKeys[]::getValueAsString.

Return type:

otx:ListTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose table key shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter is not of parameter type eTABLE_KEY.

14.6.10.16 GetDbParameterLengthKey

Semantic:

Returns the corresponding request parameter with type eLENGTHKEY for this parameter with variable length.

NOTE The appropriate MVCI system operation is MCDDbParameter::getLengthKey.

Return type:

diagDataPlus:DbParameterTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose corresponding length key parameter shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter is not of variable length or no lengthkey can be determined.

14.6.10.17 GetDbParameterMaxLength

Semantic:

Returns the maximum length in bytes of the parameter or the maximum count of items. For the parameter types BYTEFIELD, ASCIISTRING and UNICODE2STRING the return value is the maximum length in bytes. For the parameter types eFIELD and eEND_OF_PDU the return value is the maximum count of items.

NOTE The appropriate MVCI system operation is MCDDbParameter::getMaxLength.

Return type:

otx:IntegerTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose maximum length in bytes shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter type is one of BYTEFIELD, ASCIISTRING and UNICODE2STRING and the parameter has no length information.

14.6.10.18 GetDbParameterMaxNumberOfItems

Semantic:

Returns 1 in case of a simple parameter type or eSTRUCTURE or the maximum number of items of a parameter of type eEND_OF_PDU.

NOTE The appropriate MVCI system operation is MCDDbRequestParameter::getMaxNumberOfItems.

Return type:

otx:IntegerTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbRequestParameter) whose maximum number of items shall be returned.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Exceptions:

— None

14.6.10.19 GetDbParameterMinLength

Semantic:

Returns the minimum length in bytes of the parameter or the minimum count of items. For the parameter types BYTEFIELD, ASCIISTRING and UNICODE2STRING the return value is the minimum length in bytes. For the parameter types eFIELD and eEND_OF_PDU the return value is the minimum count of items.

NOTE The appropriate MVCI system operation is MCDDbParameter::getMinLength.

Return type:

otx:IntegerTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose minimum length shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter type is one of BYTEFIELD, ASCIISTRING and UNICODE2STRING and the parameter has no length information.

14.6.10.20 GetDbParameterNrcConstValues

Semantic:

Returns a list of Integer, containing the coded values of the NRC-CONST element of the parameter (MCDDbResponseParameter). In case of a RequestParameter the list is empty.

NOTE The appropriate MVCI system operation is MCDDbResponseParameter::getNrcConstValues.

Return type:

otx:ListTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbRequestParameter) whose negative response code values shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter is not of parameter type eNRC_CONST.

14.6.10.21 GetDbParameterODXBytePos

Semantic:

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Returns the Byte position, which is available in ODX. The counting starts with 0.

NOTE The appropriate MVCI system operation is MCDDbParameter::getODXBytePos.

Return type:

otx:IntegerTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose ODX byte position shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the the byte position is not available.

14.6.10.22 GetDbParameterRadix

Semantic:

Returns the display radix of the parameter. The display radix shall be used by the client to display a numeric parameter value in the format as defined in the database (2 - ODX value BIN, 10 - ODX value DEC, 16 - ODX value HEX).

NOTE The appropriate MVCI system operation is Operation MCDDbParameter::getRadix.

Return type:

otx:IntegerTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose radix shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter is not an A_UINT or A_INT value.

14.6.10.23 GetDbParameters

Semantic:

Returns a list of DbParameters of the next level (child parameters). This list can be empty if there is no next level.

NOTE The appropriate MVCI system operation is MCDDbParameter::getDbParameters.

Return type:

otx:ListTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose child parameters shall be returned.

Exceptions:

— None

14.6.10.24 GetDbParameterSemantic

Semantic:

Returns the semantic attribute defined in ODX as a string. In case of MUX the method returns an empty string.

NOTE The appropriate MVCI system operation is MCDDbParameter::getSemantic.

Return type:

otx:StringTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose semantic shall be returned.

Exceptions:

— None

14.6.10.25 GetDbParameterSpecialDataGroups

Semantic:

Returns the special data groups defined for the corresponding **DbParameter** in the project's database as a list. An empty list will be delivered if no SDGs are available in the ODX data.

NOTE The appropriate MVCI system operation is MCDDbParameter::getDbSDGs.

Return type:

otx:ListTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) for which SDGs shall be returned.

Exceptions:

— None

14.6.10.26 GetDbParameterStructureByKey

Semantic:

Returns a list of **DbParameter** that represent the contents of the STRUCTURE referenced by the table row of the given key. The length of the list is zero, if no MCDDbParameters is available.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

NOTE The appropriate MVCI system operation is MCDDbParameter::getStructureByKey.

Return type:

otx:ListTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) for which content of the related structure shall be returned.

— <key> : otx:StringTerm [1..1]

This key addresses the table row of the **DbParameter** of type eTABLE_STRUCT.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter is not of parameter type eTABLE_STRUCT.

14.6.10.27 GetDbParameterTable

Semantic:

Returns the **DbTable** parameter structure of the **DbParameter**, if available. That is, the decomposition of this parameter into further parameters represents exactly one table row of the table which is returned by this term. In case of a static reference, the decomposition can already be obtained at this **DbParameter** by calling the term **GetDbParameters**. In case of a dynamic reference, the collection returned by the term **GetDbParameters** is empty.

NOTE The appropriate MVCI system operation is MCDDbParameter::getDbTable.

Return type:

diagDataPlus:DbTableTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose table parameter shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter is not of parameter type eTABLE_STRUCT.

14.6.10.28 GetDbParameterTableKeyParam

Semantic:

Returns the parameter of type eTABLE_KEY referenced by this parameter of type eTABLE_STRUCT.

NOTE The appropriate MVCI system operation is MCDDbParameter::getDbTableKeyParam.

Return type:

diagDataPlus:DbParameterTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose table key parameter shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter is not of parameter type eTABLE_STRUCT.

14.6.10.29 GetDbParameterTableStructParams

Semantic:

Returns a list of parameters of type eTABLE_STRUCT which reference the parameter of type eTABLE_KEY. In ODX several parameters of type eTABLE_STRUCT are allowed to reference the same parameter of type eTABLE_KEY. The only restriction is that these parameters need to be located on the same level in the parameter hierarchy. Therefore, this method returns a list of parameters of type eTABLE_STRUCT which reference the parameter of type eTABLE_KEY. More precisely, if the parameter is an MCDDbTableParameter of type eTABLE_KEY, a list of MCDDbTableParameters of type eTABLE_STRUCT which reference this MCDDbTableParameter is returned. The list can be empty if there are no matching parameters.

NOTE The appropriate MVCI system operation is MCDDbParameter::getDbTableStructParams.

Return type:

otx:ListTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose table struct parameter(s) shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter is not of parameter type eTABLE_KEY.

14.6.10.30 GetParameterDbParameter

Semantic:

Returns the corresponding database object (MCDDbParameter) which was used to create this **diag:Parameter** (MCDParameter).

NOTE The appropriate MVCI system operation is MCDParameter::getDbObject.

Return type:

diagDataPlus:DbParameterTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Parameters:

— <parameter> : diag:ParameterTerm [1..1]

This element addresses the **diag:Parameter** (MCDParameter) whose corresponding database object (MCDDbParameter) shall be returned.

Exceptions:

— None

14.6.10.31 IsDbParameterConstant

Semantic:

Returns true if the parameter is of ODX type CODED-CONST or PHYS-CONST otherwise false.

NOTE The appropriate MVCI system operation is MCDDbParameter::isConstant.

Return type:

otx:BooleanTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) which shall be checked whether if it is constant or not.

Exceptions:

— None

14.6.10.32 IsDbParameterVariableLength

Semantic:

Returns true if the parameter is of variable length otherwise false.

NOTE The appropriate MVCI system operation is MCDDbParameter::isVariableLength.

Return type:

otx:BooleanTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) which shall be checked whether if it is of variable length or not.

Exceptions:

— None

14.6.11 Semantics for DbRequest terms

Figure 51 shows the syntax of the **OTX DiagDataBrowsingPlus extension DbRequest** related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

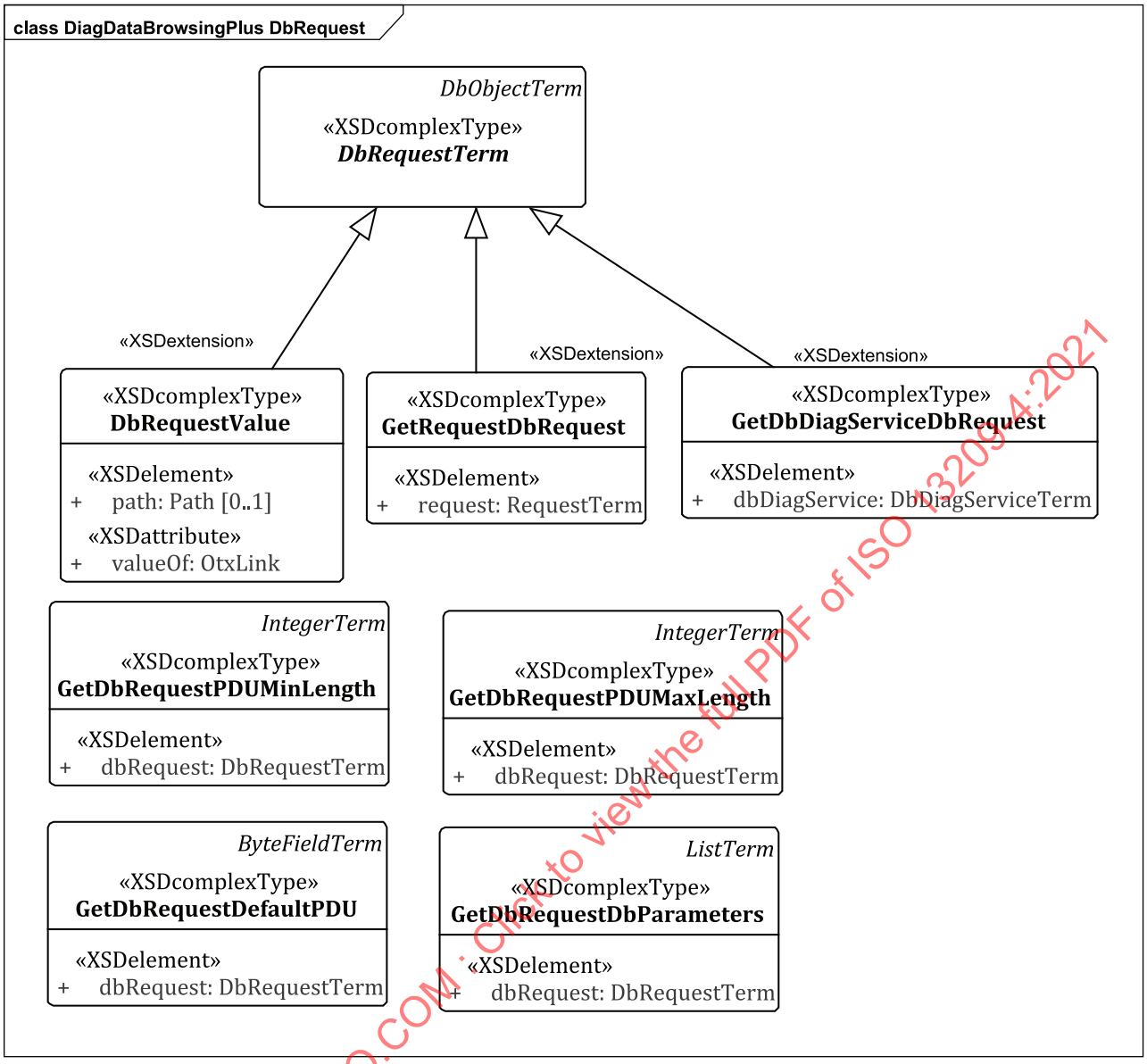


Figure 51 — DiagDataBrowsingPlus – DbRequest terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

14.6.11.1 DbRequestTerm

Semantic:

A **DbRequestTerm** is a term that returns a **DbRequest**.

14.6.11.2 DbRequestValue

Semantic:

Returns the **DbRequest** stored in a **DbRequest** variable.

Return type:

diagDataPlus:DbRequestTerm

© ISO 2021 – All rights reserved

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

14.6.11.3 GetDbDiagServiceDbRequest

Semantic:

Returns a **DbRequest** object which represents the MCDDbRequest for the given **DbDiagService**.

NOTE The appropriate MVCI system operation is MCDDbDiagComPrimitive::getDbRequest.

Return type:

diagDataPlus:DbRequestTerm

Parameters:

— <dbDiagService> : diagDataPlus:DbDiagServiceTerm [1..1]

This element addresses the dbDiagService whose associated request shall be returned.

Exceptions:

— None

14.6.11.4 GetDbRequestDbParameters

Semantic:

Returns a list of **DbParameter** objects which represents the MCDDbRequestParameters for the given **DbRequest**. An empty list can be returned, if no request parameters are defined for the request in the project's database.

NOTE The appropriate MVCI system operation is MCDDbRequest::getDbRequestParameters.

Return type:

otx:ListTerm

Parameters:

— <dbRequest> : diagDataPlus:DbRequestTerm [1..1]

This element represents the **DbRequest** for which associated **DbParameters** shall be returned.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

14.6.11.5 GetDbRequestDefaultPDU

Semantic:

Returns the default PDU of the **DbRequest** as a ByteField.

NOTE The appropriate MVCI system operation is MCDDbRequest::getDefaultPDU.

Return type:

otx:ByteFieldTerm

Parameters:

— <dbRequest> : diagDataPlus:DbRequestTerm [1..1]

This element represents the **DbRequest** whose default PDU shall be returned.

Exceptions:

NotSupportedException

— This exception is thrown if the communication primitive does not support the PDU information.

14.6.11.6 GetDbRequestPDUMaxLength

Semantic:

Returns the maximum PDU length of the **DbRequest**.

NOTE The appropriate MVCI system operation is MCDDbRequest::getPDUMaxLength.

Return type:

otx:IntegerTerm

Parameters:

— <dbRequest> : diagDataPlus:DbRequestTerm [1..1]

This element represents the **DbRequest** whose maximum PDU length shall be returned.

Exceptions:

NotSupportedException

— This exception is thrown if the communication primitive does not support the PDU information.

14.6.11.7 GetDbRequestPDUMinLength

Semantic:

Returns the minimum PDU length of the **DbRequest**.

NOTE The appropriate MVCI system operation is MCDDbRequest::getPDUMinLength.

Return type:

otx:IntegerTerm

Parameters:

— <dbRequest> : diagDataPlus:DbRequestTerm [1..1]

This element represents the **DbRequest** whose minimum PDU length shall be returned.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Exceptions:

NotSupportedException

— This exception is thrown if the communication primitive does not support the PDU information.

14.6.11.8 GetRequestDbRequest

Semantic:

Returns the database part (MCDDbRequest) of a **diag:Request** (MCDRequest).

NOTE The appropriate MVCI system operation is MCDRequest::getDbObject.

Return type:

diagDataPlus:DbRequestTerm

Parameters:

— <request> : diag:RequestTerm [1..1]

This element represents the **diag:Request** whose database part shall be returned.

Exceptions:

— None

14.6.12 Semantics for DbResponse terms

Figure 52 shows the syntax of the **OTX DiagDataBrowsingPlus extension DbResponse** related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

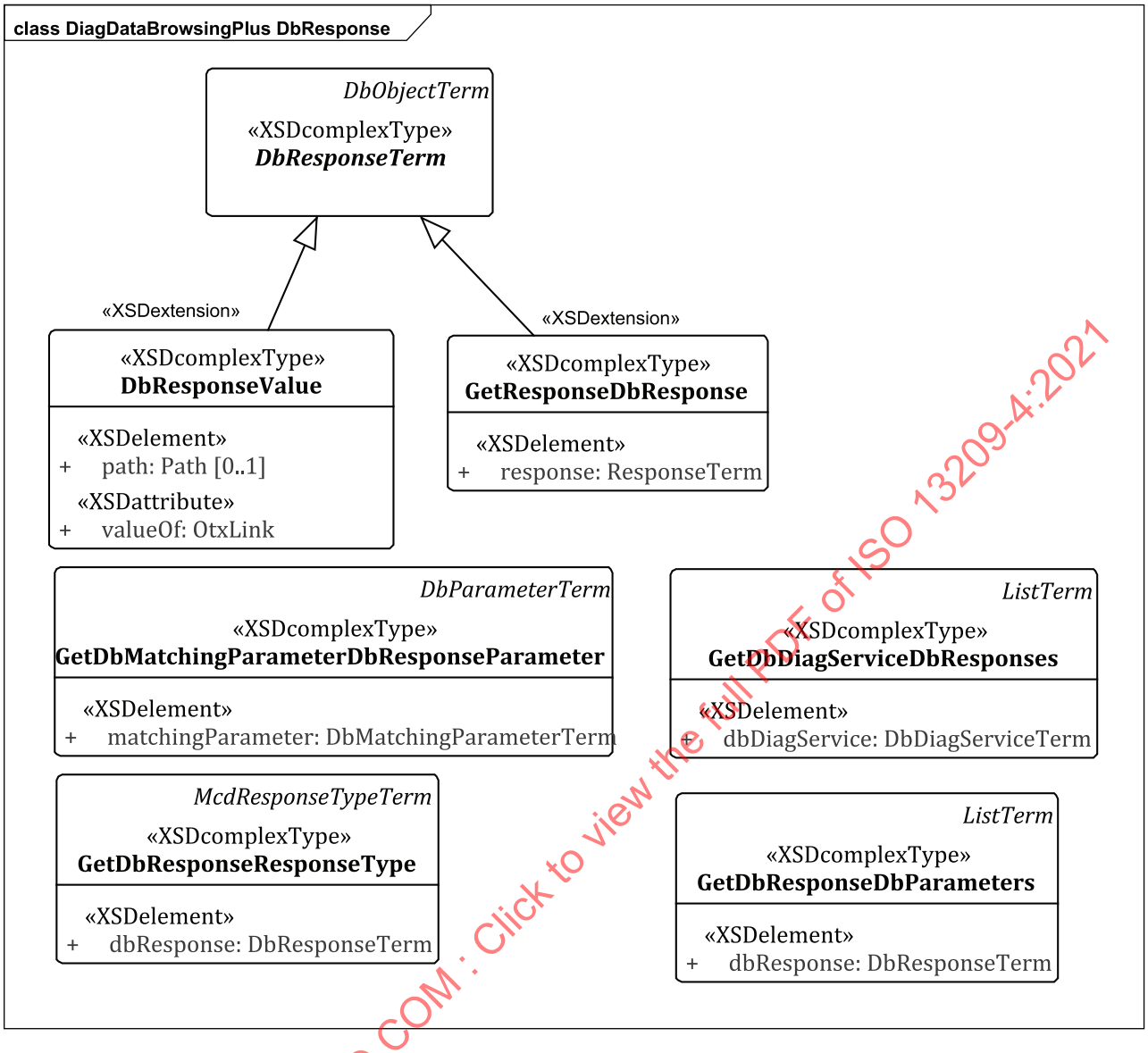


Figure 52 — DiagDataBrowsingPlus – DbResponse terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

14.6.12.1 DbResponseTerm

Semantic:

A **DbResponseTerm** is a term that returns a **DbResponse**.

14.6.12.2 DbResponseValue

Semantic:

Returns the **DbResponse** stored in a **DbResponse** variable.

Return type:

diagDataPlus:DbResponseTerm

© ISO 2021 – All rights reserved

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

14.6.12.3 GetDbDiagServiceDbResponses

Semantic:

Returns a list of **DbResponse** objects which represents the MCDDbResponses for the given **DbDiagService**. An empty list can be returned, if no response is defined for the diag service in the project's database.

NOTE The appropriate MVCI system operation is MCDDbDiagComPrimitive::getDbResponses.

Return type:

otx:ListTerm

Parameters:

— <dbDiagService> : diagDataPlus:DbDiagServiceTerm [1..1]

This element addresses the dbDiagService whose associated responses shall be returned.

Exceptions:

— None

14.6.12.4 GetDbMatchingParameterDbResponseParameter

Semantic:

Returns the corresponding **DbParameter** (MCDDbResponseParameter) responsible for the matching parameter.

NOTE The appropriate MVCI system operation is MCDDbMatchingParameter::getDbResponseParameter.

Return type:

diagDataPlus:DbParameterTerm

Parameters:

— <matchingParameter> : diagDataPlus:DbMatchingParameterTerm [1..1]

This element addresses the **DbMatchingParameter** whose response parameter shall be returned.

Exceptions:

InvalidTypeException

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— This exception is thrown if the corresponding response parameter is not available.

14.6.12.5 GetDbResponseDbParameters

Semantic:

Returns a list of **DbParameter** objects which represents the MCDDbResponseParameters for the given **DbResponse**. An empty list can be returned, if no response parameters are defined for the response in the project's database.

NOTE The appropriate MVCI system operation is MCDDbResponse::getDbResponseParameters.

Return type:

otx:ListTerm

Parameters:

— <dbResponse> : diagDataPlus:DbResponseTerm [1..1]

This element represents the **DbResponse** whose associated **DbParameters** shall be returned.

Exceptions:

— None

14.6.12.6 GetDbResponseResponseType

Semantic:

Returns the type (MCDResponseType) of this response.

NOTE The appropriate MVCI system operation is MCDDbResponse::getResponseType.

Return type:

diagDataPlus:McdResponseTypeTerm

Parameters:

— <dbResponse> : diagDataPlus:DbResponseTerm [1..1]

This element represents the **DbResponse** whose type shall be returned.

Exceptions:

— None

14.6.12.7 GetResponseDbResponse

Semantic:

Returns the the database part (MCDDbResponse) of a **diag:Response** (MCDResponse).

NOTE The appropriate MVCI system operation is MCDResponse::getDbObject.

Return type:

diagDataPlus:DbResponseTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Parameters:

— <response> : diag:ResponseTerm [1..1]

This element represents the **diag:Response** whose database part shall be returned.

Exceptions:

— None

14.6.13 Semantics for DbSpecialDataElement terms

Figure 53 shows the syntax of the **OTX DiagDataBrowsingPlus extension DbSpecialDataElement** related terms.

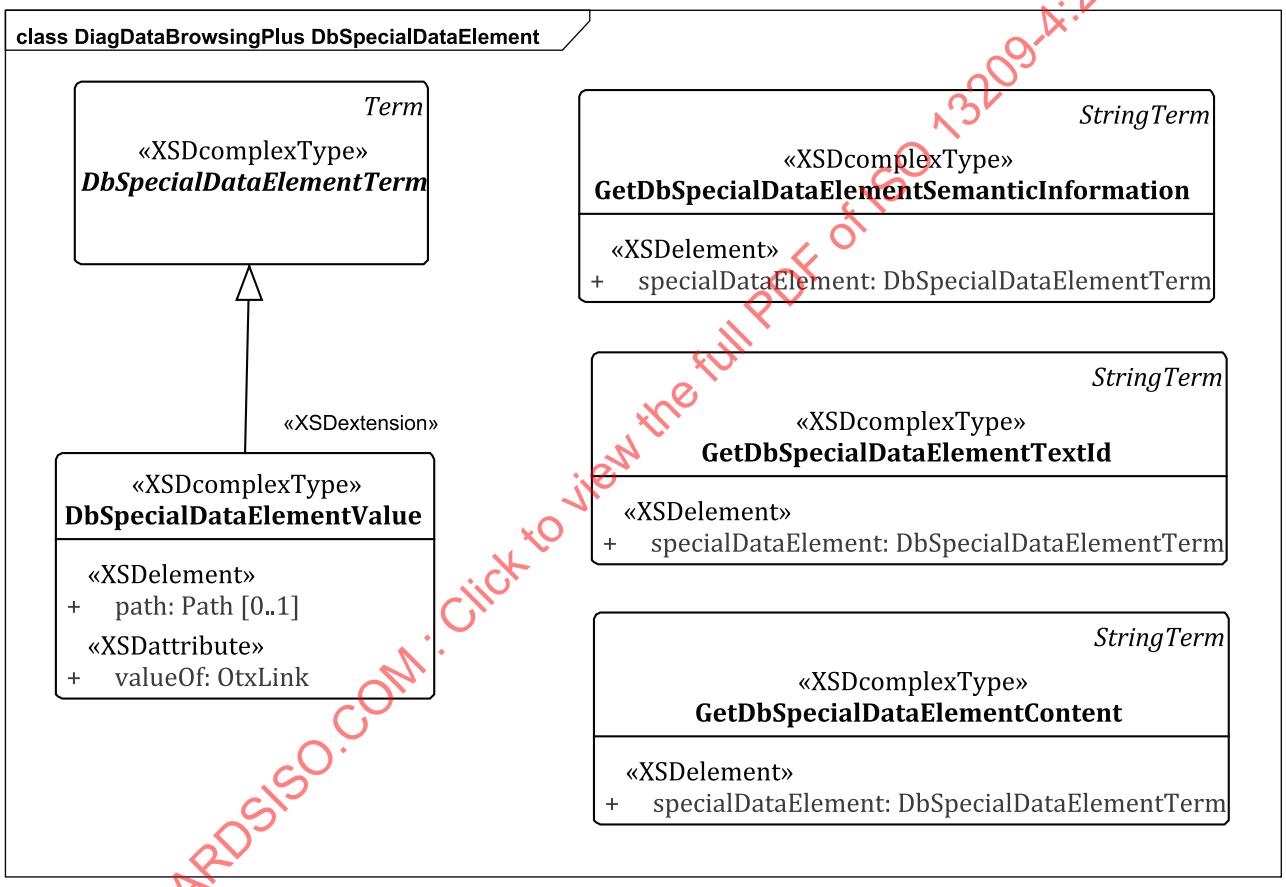


Figure 53 — DiagDataBrowsingPlus – DbSpecialDataElement terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

14.6.13.1 DbSpecialDataElementTerm

Semantic:

A **DbSpecialDataElementTerm** is a term that returns a **DbSpecialDataElement**.

14.6.13.2 DbSpecialDataElementValue

Semantic:

Returns the **DbSpecialDataElement** stored in a **DbSpecialDataElement** variable.

Return type:

© ISO 2021 – All rights reserved

diagDataPlus:DbSpecialDataElementTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

14.6.13.3 GetDbSpecialDataElementContent

Semantic:

Returns the content of the **DbSpecialDataElement**.

NOTE The appropriate MVCI system operation is MCDDbSpecialDataElement::getContent.

Return type:

otx:StringTerm

Parameters:

— <specialDataElement> : diagDataPlus:DbSpecialDataElementTerm [1..1]

This element addresses the **DbSpecialDataElement** (MCDDbSpecialDataElement) whose content shall be returned.

Exceptions:

— None

14.6.13.4 GetDbSpecialDataElementSemanticInformation

Semantic:

Returns the Semantic Information (SI) of the **DbSpecialDataElement**.

NOTE The appropriate MVCI system operation is MCDDbSpecialData::getSemanticInformation.

Return type:

otx:StringTerm

Parameters:

— <specialDataElement> : diagDataPlus:DbSpecialDataElementTerm [1..1]

This element addresses the **DbSpecialDataElement** (MCDDbSpecialDataElement) whose semantic information shall be returned.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Exceptions:

— None

14.6.13.5 GetDbSpecialDataElementTextId

Semantic:

Returns the text ID (TextIdentifier) of the **DbSpecialDataElement**.

NOTE The appropriate MVCI system operation is MCDDbSpecialDataElement::getTextID.

Return type:

otx:StringTerm

Parameters:

— <specialDataElement> : diagDataPlus:DbSpecialDataElementTerm [1..1]

This element addresses the **DbSpecialDataElement** (MCDDbSpecialDataElement) whose text ID shall be returned.

Exceptions:

— None

14.6.14 Semantics for DbSpecialDataGroup terms

Figure 54 shows the syntax of the **OTX DiagDataBrowsingPlus extension DbSpecialDataGroup** related terms.

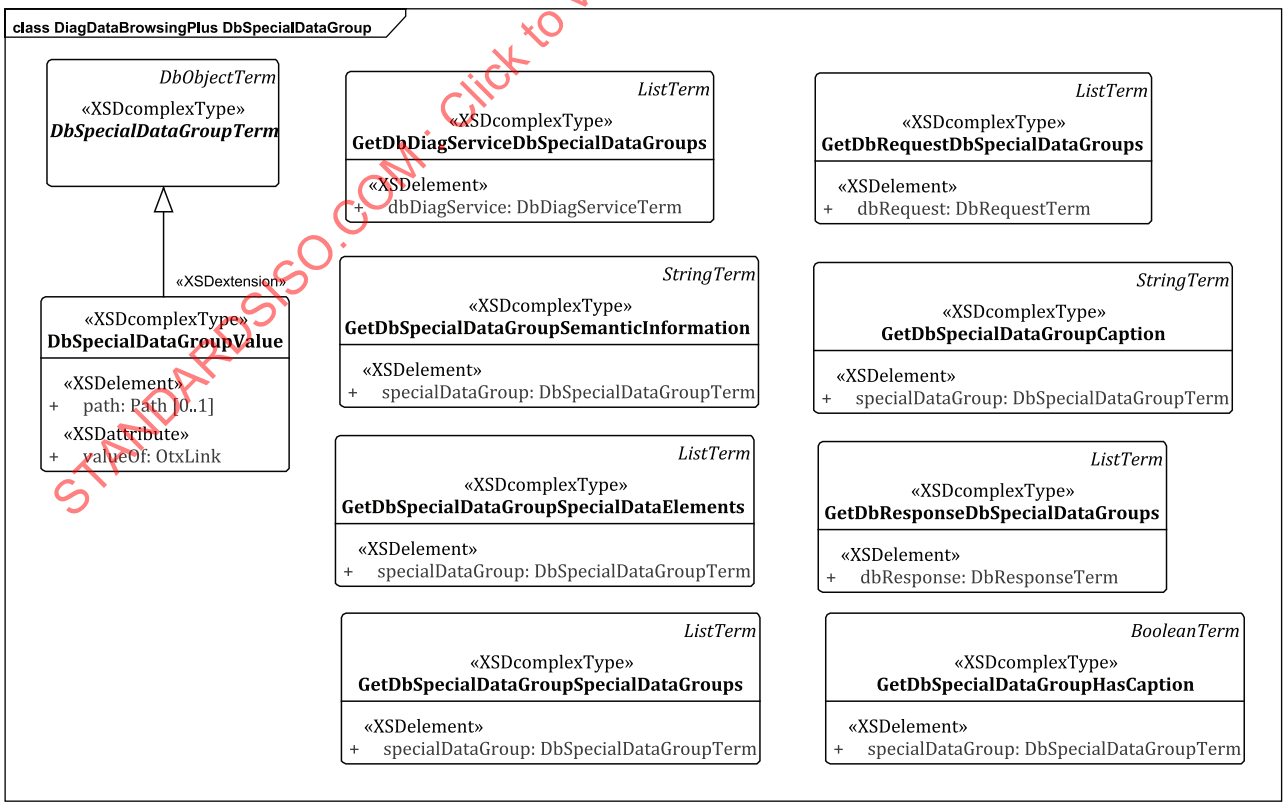


Figure 54 — DiagDataBrowsingPlus – DbSpecialDataGroup terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

14.6.14.1 DbSpecialDataGroupTerm

Semantic:

A **DbSpecialDataGroupTerm** is a term that returns a **DbSpecialDataGroup**.

14.6.14.2 DbSpecialDataGroupValue

Semantic:

Returns the **DbSpecialDataGroup** stored in a **DbSpecialDataGroup** variable.

Return type:

diagDataPlus:DbSpecialDataGroupTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

14.6.14.3 GetDbDiagServiceDbSpecialDataGroups

Semantic:

Returns a list of **DbSpecialDataGroup** (SDG) for the corresponding **DbDiagService** in the project's database. An empty list will be delivered if no SDGs are available in the ODX data.

NOTE The appropriate MVCI system operation is MCDDbDataPrimitive::getDbSDGs.

Return type:

otx:ListTerm

Parameters:

— <dbDiagService> : diagDataPlus:DbDiagServiceTerm [1..1]

This element addresses the **DbDiagService** whose special data groups shall be returned.

Exceptions:

— None

14.6.14.4 GetDbRequestDbSpecialDataGroups

Semantic:

Returns a list of **DbSpecialDataGroup** (SDG) of the **DbRequest**. An empty list can be returned if no SDG are available.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

NOTE The appropriate MVCI system operation is MCDDbRequest::getDbSDGs.

Return type:

otx:ListTerm

Parameters:

— <dbRequest> : diagDataPlus:DbRequestTerm [1..1]

This element represents the **DbRequest** whose SDGs shall be returned.

Exceptions:

— None

14.6.14.5 GetDbResponseDbSpecialDataGroups

Semantic:

Returns a list of **DbSpecialDataGroup** (SDG) of the **DbResponse**. An empty list can be returned if no SDGs are available.

NOTE The appropriate MVCI system operation is MCDDbResponse::getDbSDGs.

Return type:

otx:ListTerm

Parameters:

— <dbResponse> : diagDataPlus:DbResponseTerm [1..1]

This element represents the **DbResponse** whose SDGs shall be returned.

Exceptions:

— None

14.6.14.6 GetDbSpecialDataGroupCaption

Semantic:

Returns the name of the MCDDbSpecialDataGroupCaption as string.

NOTE The appropriate MVCI system operation is MCDDbSpecialDataGroup::getCaption.

Return type:

otx:StringTerm

Parameters:

— <specialDataGroup> : diagDataPlus:DbSpecialDataGroupTerm [1..1]

This element addresses the **DbSpecialDataGroup** (MCDDbSpecialDataGroup) whose caption shall be returned.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

14.6.14.7 GetDbSpecialDataGroupHasCaption

Semantic:

Check the availability of a Caption element.

NOTE The appropriate MVCI system operation is MCDDbSpecialDataGroup :: hasCaption.

Return type:

otx:BooleanTerm

Parameters:

— <specialDataGroup> : diagDataPlus:DbSpecialDataGroupTerm [1..1]

This element addresses the **DbSpecialDataGroup** (MCDDbSpecialDataGroup) whose caption shall be checked.

Exceptions:

— None

14.6.14.8 GetDbSpecialDataGroupSemanticInformation

Semantic:

Returns the Semantic Information (SI) of the **DbSpecialDataGroup** Caption.

NOTE The appropriate MVCI system operation is MCDDbSpecialData::getSemanticInformation.

Return type:

otx:StringTerm

Parameters:

— <specialDataGroup> : diagDataPlus:DbSpecialDataGroupTerm [1..1]

This element addresses the **DbSpecialDataGroup** (MCDDbSpecialDataGroup) whose semantic information shall be returned.

Exceptions:

— None

14.6.14.9 GetDbSpecialDataGroupSpecialDataElements

Semantic:

Returns a list of **DbSpecialDataElement** (SD) of the **DbSpecialDataGroup**. An empty list can be returned if no SDs are available.

NOTE The appropriate MVCI system operation is MCDDbSpecialDataGroup::getItemByIndex.

Return type:

otx:ListTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <specialDataGroup> : diagDataPlus:DbSpecialDataGroupTerm [1..1]

This element addresses the **DbSpecialDataGroup** (MCDDbSpecialDataGroup) whose SDs shall be returned.

Exceptions:

— None

14.6.14.10 GetDbSpecialDataGroupSpecialDataGroups

Semantic:

Returns a list of **DbSpecialDataGroup** (SDG) of the **DbSpecialDataGroup**. An empty list can be returned if no SDGs are available.

NOTE The appropriate MVCI system operation is MCDDbSpecialDataGroup::getItemByIndex.

Return type:

otx:ListTerm

Parameters:

— <specialDataGroup> : diagDataPlus:DbSpecialDataGroupTerm [1..1]

This element addresses the **DbSpecialDataGroup** (MCDDbSpecialDataGroup) whose SDGs shall be returned.

Exceptions:

— None

14.6.15 Semantics for DbSubComponent terms

Figure 55 shows the syntax of the **OTX DiagDataBrowsingPlus extension DbSubComponent** related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

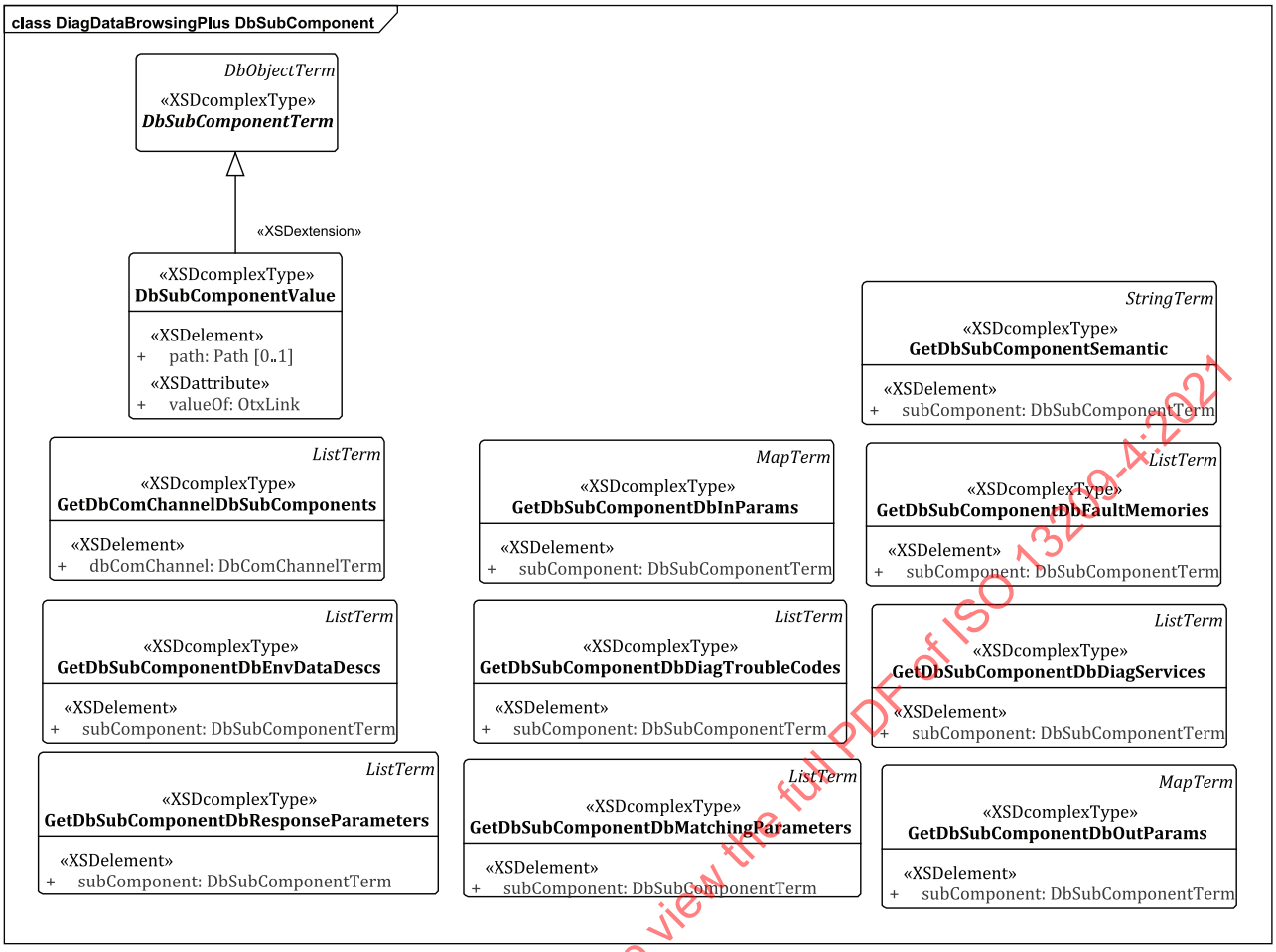


Figure 55 — DiagDataBrowsingPlus – DbSubComponent terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

14.6.15.1 DbSubComponentTerm

Semantic:

A **DbSubComponentTerm** is a term that returns a **DbSubComponent**.

14.6.15.2 DbSubComponentValue

Semantic:

Returns the **DbSubComponent** stored in a **DbSubComponent** variable.

Return type:

diagDataPlus:DbSubComponentTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

© ISO 2021 – All rights reserved

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

14.6.15.3 GetDbComChannelDbSubComponents

Semantic:

Returns a list of all **DbSubComponent** (MCDDbSubComponent) referenced by the **DbComChannel**. The list can be empty.

NOTE The appropriate MVCI system operation is MCDDbLocation::getDbSubComponents.

Return type:

otx:ListTerm

Parameters:

— <dbComChannel> : diagDataPlus:DbComChannelTerm [1..1]

This element addresses the **ComChannel** whose subcomponents shall be returned.

Exceptions:

— None

14.6.15.4 GetDbSubComponentDbDiagServices

Semantic:

Returns a list of strings containing the SHORT-NAMEs of all diagnostic services for the **DbSubComponent** that are available in ODX. If no elements are available in the database, then an empty list is returned.

NOTE The appropriate MVCI system operation is MCDDbSubComponent::getDbSubComponentParamConnectors[n]::getDbDiagComPrimitive.

Return type:

otx:ListTerm

Parameters:

— <subComponent> : diagDataPlus:DbSubComponentTerm [1..1]

This element addresses the **DbSubComponent** whose diag services shall be returned.

Exceptions:

— None

14.6.15.5 GetDbSubComponentDbDiagTroubleCodes

Semantic:

Returns a list of **DbDiagTroubleCode** for the **DbSubComponents** that are available in ODX. If no elements are available in the database, then an empty list will be returned.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

NOTE The appropriate MVCI system operation is MCDDbSubComponent::getDbDiagTroubleCodeConnectors[n]::getDbDiagTroubleCode.

Return type:

otx:ListTerm

Parameters:

— <subComponent> : diagDataPlus:DbSubComponentTerm [1..1]

This element addresses the **DbSubComponent** whose DTCs shall be returned.

Exceptions:

— None

14.6.15.6 GetDbSubComponentDbEnvDataDescs

Semantic:

Returns a list of **DbEnvDataDesc** for the **DbSubComponents** that are is available in ODX. If no elements are available in the database, then an empty list will be returned.

NOTE The appropriate MVCI system operation is MCDDbSubComponent::getDbEnvDataConnectors[n]::getDbEnvDataDesc.

Return type:

otx:ListTerm

Parameters:

— <subComponent> : diagDataPlus:DbSubComponentTerm [1..1]

This element addresses the **DbSubComponent** whose environment data descriptions shall be returned.

Exceptions:

— None

14.6.15.7 GetDbSubComponentDbFaultMemories

Semantic:

Returns a list of **DbFaultMemory** for the **DbSubComponents** that are is available in ODX. If no elements are available in the database, then an empty list will be returned.

NOTE The appropriate MVCI system operation is MCDDbSubComponent::getDbDiagTroubleCodeConnectors[n]::getDbFaultMemory.

Return type:

otx:ListTerm

Parameters:

— <subComponent> : diagDataPlus:DbSubComponentTerm [1..1]

This element addresses the **DbSubComponent** which shall be returned.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Exceptions:

— None

14.6.15.8 GetDbSubComponentDbInParams

Semantic:

Returns a Map<String, List<DbParameters>> which contains all available MCDDbRequestParameter of the **DbSubComponent**. The map key is the related DiagService SHORT-NAME, see **GetDbSubComponentDbDiagServices**. If no elements are available in database, then an empty list will be returned.

NOTE The appropriate MVCI system operation is MCDDbSubComponent::getDbSubComponentParamConnectors[n]::getDbSubComponentInParams.

Return type:

otx:MapTerm

Parameters:

— <subComponent> : diagDataPlus:DbSubComponentTerm [1..1]

This element addresses the **DbSubComponent** whose MCDDbRequestParameters shall be returned.

Exceptions:

— None

14.6.15.9 GetDbSubComponentDbMatchingParameters

Semantic:

Returns a List<List<DbMatchingParameter>> for the **DbSubComponent** which is available in ODX. If no elements are available in database, then an empty list will be returned.

NOTE The appropriate MVCI system operation is MCDDbSubComponent::getDbSubComponentPatterns[n]::getDbMatchingParameters.

Return type:

otx:ListTerm

Parameters:

— <subComponent> : diagDataPlus:DbSubComponentTerm [1..1]

This element addresses the **DbSubComponent** whose matching parameters are returned.

Exceptions:

— None

14.6.15.10 GetDbSubComponentDbOutParams

Semantic:

Returns a Map<String, List<DbParameters>> which contains all available MCDDbResponseParameter of the **DbSubComponent**. The map key is the related DiagService SHORT-NAME, see **GetDbSubComponentDbDiagServices**. If no elements are available in database, then an empty list will be returned.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

NOTE The appropriate MVCI system operation is MCDDbSubComponent::getDbSubComponentParamConnectors[n]::getDbSubComponentOutParams.

Return type:

otx:MapTerm

Parameters:

— <subComponent> : diagDataPlus:DbSubComponentTerm [1..1]

This element addresses the **DbSubComponent** whose MCDDbResponseParameters shall be returned.

Exceptions:

— None

14.6.15.11 GetDbSubComponentDbResponseParameters

Semantic:

Returns a list of **DbParameters** for the **DbSubComponent** which is available in ODX. If no elements are available in database, then an empty list will be returned.

NOTE The appropriate MVCI system operation is MCDDbSubComponent::getDbEnvDataConnectors[n]::getDbEnvData.

Return type:

otx:ListTerm

Parameters:

— <subComponent> : diagDataPlus:DbSubComponentTerm [1..1]

This element addresses the **DbSubComponent** whose response parameters shall be returned.

Exceptions:

— None

14.6.15.12 GetDbSubComponentSemantic

Semantic:

Returns the value of the semantic attribute for a **DbSubComponent** from ODX as a string. If no semantic can be obtained from ODX an empty string will be returned.

NOTE The appropriate MVCI system operation is MCDDbSubComponent::getSemantic.

Return type:

otx:StringTerm

Parameters:

— <subComponent> : diagDataPlus:DbSubComponentTerm [1..1]

This element addresses the **DbSubComponent** whose semantic shall be returned.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

14.6.16 Semantics for DbTable terms

Figure 56 shows the syntax of the **OTX DiagDataBrowsingPlus extension DbTable** related terms.

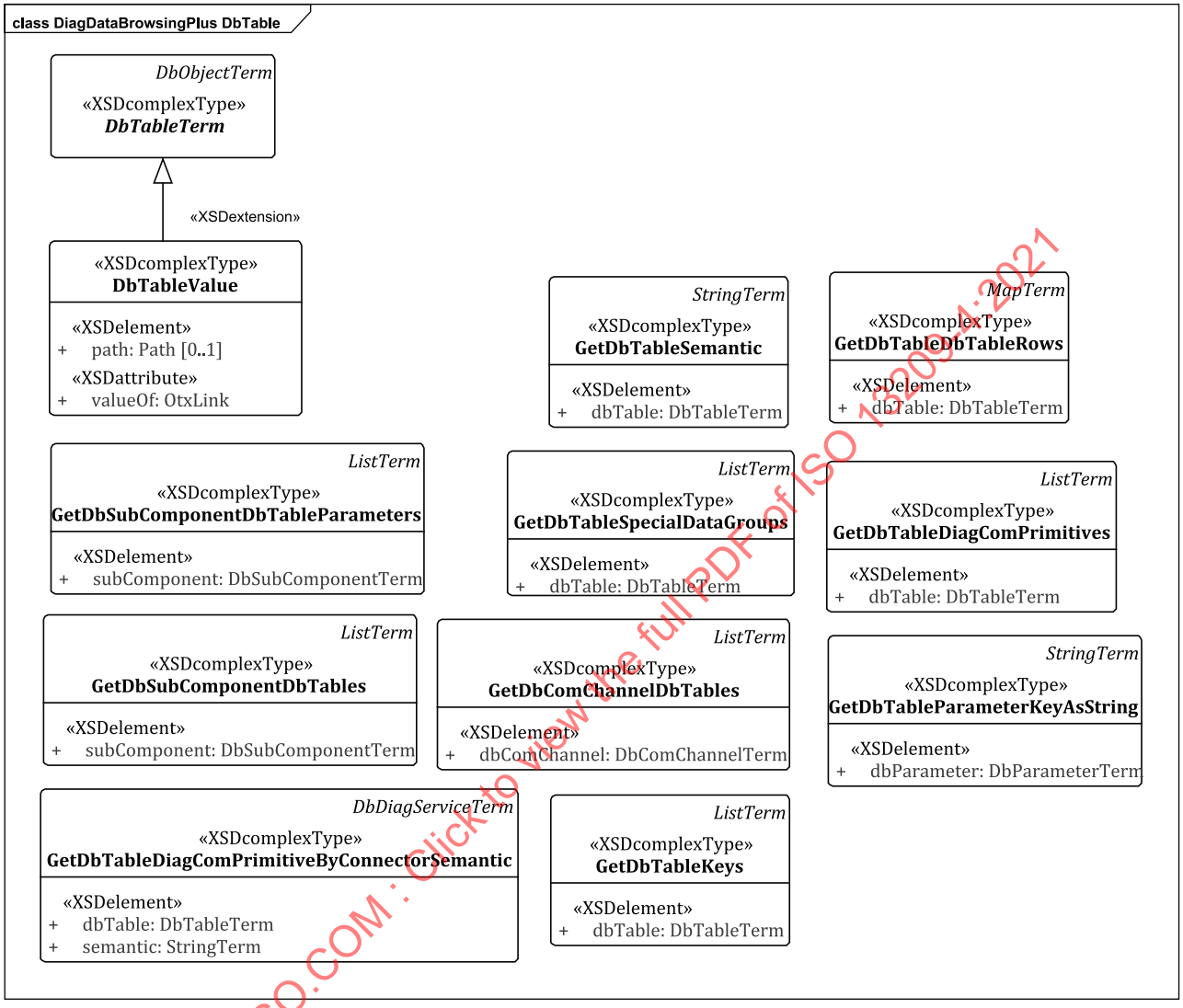


Figure 56 — DiagDataBrowsingPlus – DbTable terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

14.6.16.1 DbTableTerm

Semantic:

A **DbTableTerm** is a term that returns a **DbTable**.

14.6.16.2 DbTableValue

Semantic:

Returns the **DbTable** stored in a **DbTable** variable.

Return type:

diagDataPlus:DbTableTerm

© ISO 2021 – All rights reserved

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

14.6.16.3 GetDbComChannelDbTables

Semantic:

Returns a list of all **DbTable** (McdDbTable) referenced by the **ComChannel**. The list can be empty.

NOTE The appropriate MVCI system operation is MCDDbLocation::getDbTables.

Return type:

otx:ListTerm

Parameters:

— <dbComChannel> : diagDataPlus:DbComChannelTerm [1..1]

This element addresses the **ComChannel** whose tables shall be returned.

Exceptions:

— None

14.6.16.4 GetDbSubComponentDbTableParameters

Semantic:

Returns a list of DbParamters (MCDDbTableParameter) for the **DbSubComponent** that are available in ODX. If no elements are available in database, then an empty list will be returned.

NOTE The appropriate MVCI system operation is MCDDbSubComponent::getDbTableRowConnectors[n]::getDbTableRow.

Return type:

otx:ListTerm

Parameters:

— <subComponent> : diagDataPlus:DbSubComponentTerm [1..1]

This element addresses the **DbSubComponent** whose table parameters shall be returned.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

14.6.16.5 GetDbSubComponentDbTables

Semantic:

Returns a list of **DbTables** for the **DbSubComponent** that are available in ODX. If no elements are available in database, then an empty list will be returned.

NOTE The appropriate MVCI system operation is MCDDbSubComponent::getDbTableRowConnectors[n]::getDbTable.

Return type:

otx:ListTerm

Parameters:

— <subComponent> : diagDataPlus:DbSubComponentTerm [1..1]

This element addresses the **DbSubComponent** whose tables shall be returned.

Exceptions:

— None

14.6.16.6 GetDbTableDbTableRows

Semantic:

Returns all table rows contained in a given table as a Map<String, DbParameter>. The key of the map contains the value of the key associated with the MCDDbTableParameter in the corresponding MCDDbTable. The value contains the related MCDDbTableParameter. An empty map is returned if no MCDDbTableParameters are available for this table.

NOTE The appropriate MVCI system operation is MCDDbTable::getDbTableRows[]::getKey::getValueAsString; MCDDbTable::getDbTableRows.

Return type:

otx:MapTerm

Parameters:

— <dbTable> : diagDataPlus:DbTableTerm [1..1]

This element addresses the **DbTable** (MCDDbTable) whose table rows shall be returned.

Exceptions:

— None

14.6.16.7 GetDbTableDiagComPrimitiveByConnectorSemantic

Semantic:

Returns the **DbDiagService** which is referenced from the given table by means of a TABLE-DIAG-COM-CONNECTOR of the given semantic. Note that the semantic is unique for all TABLE-DIAG-COM-CONNECTORs referenced by a TABLE in ODX.

NOTE The appropriate MVCI system operation is MCDDbTable::getDbDiagComPrimitiveByConnectorSemantic.

Return type:

diagDataPlus:DbDiagServiceTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <dbTable> : diagDataPlus:DbTableTerm [1..1]

This element addresses the **DbTable** (MCDDbTable) whose **DbDiagService** shall be returned.

— <semantic> : otx:StringTerm [1..1]

This element addresses the unique semantic attribute.

Exceptions:

InvalidTypeException

— This exception is thrown if the current table has no TABLE-DIAG-COM-CONNECTOR with the given semantic.

14.6.16.8 GetDbTableDiagComPrimitives

Semantic:

Returns a List of diagDataPlus:DbDiagServices referenced by the given **DbTable**. The list is empty if no DbDiagService is available for this table.

NOTE The appropriate MVCI system operation is MCDDbTable::getDbDiagComPrimitives.

Return type:

otx:ListTerm

Parameters:

— <dbTable> : diagDataPlus:DbTableTerm [1..1]

This element addresses the **DbTable** (MCDDbTable) whose DiagServices shall be returned.

Exceptions:

— None

14.6.16.9 GetDbTableKeys

Semantic:

Returns the key values of all table rows in this table as a List of strings. The returned list can be empty.

NOTE The appropriate MVCI system operation is MCDDbTable::getKeys.

Return type:

otx:ListTerm

Parameters:

— <dbTable> : diagDataPlus:DbTableTerm [1..1]

This element addresses the **DbTable** (MCDDbTable) whose keys shall be returned.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

14.6.16.10 GetDbTableParameterKeyAsString

Semantic:

Returns the value of the key associated with the **DbParameter** in the corresponding **DbTable** (MCDDbTableParameter). If the **DbParameter** is not associated with a **DbTable** an empty string is returned.

NOTE The appropriate MVCI system operation is MCDDbTableParameter::getKey::getValueAsString.

Return type:

otx:StringTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbTableParameter) whose key value shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter is not of data type eTABLE_ROW.

14.6.16.11 GetDbTableSemantic

Semantic:

Returns the value of the semantic attribute attached to the corresponding **DbTable**. The semantic gives additional information on the tables nature, e.g. SECURITY or IDENTIFICATION. If no semantic can be obtained from ODX an empty string will be returned.

NOTE The appropriate MVCI system operation is MCDDbTable::getSemantic.

Return type:

otx:StringTerm

Parameters:

— <dbTable> : diagDataPlus:DbTableTerm [1..1]

This element addresses the **DbTable** (MCDDbTable) whose semantic shall be returned.

Exceptions:

— None

14.6.16.12 GetDbTableSpecialDataGroups

Semantic:

Returns the special data groups defined for the corresponding TABLE in the project's databases as a list of **DbSpecialDataGroup**.

NOTE The appropriate MVCI system operation is MCDDbTable::getDbSDGs.

Return type:

otx:ListTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <dbTable> : diagDataPlus:DbTableTerm [1..1]

This element addresses the **DbTable** (MCDDbTable) whose SDGs shall be returned.

Exceptions:

— None

14.6.17 Semantics for interval terms

Figure 57 shows the syntax of the O**TX DiagDataBrowsingPlus extension Interval** related terms.

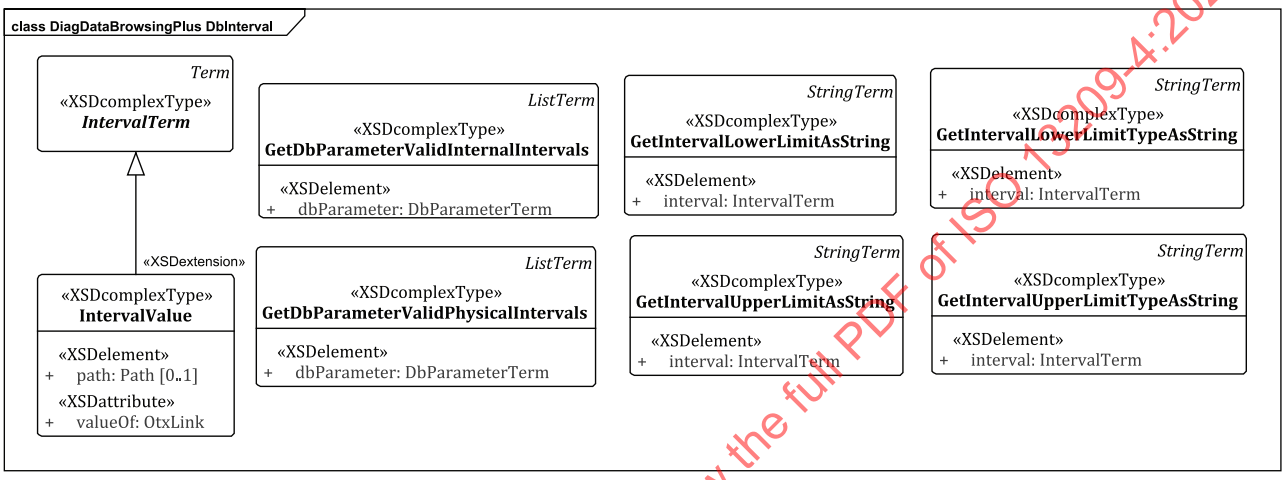


Figure 57 — DiagDataBrowsingPlus – Interval terms

14.6.17.1 GetDbParameterValidInternalIntervals

Semantic:

Returns the list of valid internal **Interval** objects defined by the intersection of COMPUSCALE intervals and valid ranges in ODX. The list might be empty if no internal interval exists.

NOTE The appropriate MVCI system operation is MCDDbParameter::getValidInternalIntervals.

Return type:

otx:ListTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose valid internal intervals shall be returned.

Exceptions:

— None

14.6.17.2 GetDbParameterValidPhysicalIntervals

Semantic:

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Returns a list of valid physical interval objects defined by the intersection of COMPUSCALE intervals and valid ranges in ODX. The list might be empty.

NOTE The appropriate MVCI system operation is MCDDbParameter::getValidPhysicalIntervals.

Return type:

otx:ListTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose valid physical intervals shall be returned.

Exceptions:

— None

14.6.17.3 GetIntervalLowerLimitAsString

Semantic:

Returns the lower limit of the text table element of the parameter as a string.

NOTE The appropriate MVCI system operation is MCDInterval::getLowerLimit::getValueAsString.

Return type:

otx:StringTerm

Parameters:

— <interval> : diagDataPlus:IntervalTerm [1..1]

This element addresses the I**nterval** (MCDInterval) whose lower limit shall be returned.

Exceptions:

— None

14.6.17.4 GetIntervalLowerLimitTypeAsString

Semantic:

Returns the lower limit type of the text table element of the parameter as a string. Valid values are "eLIMIT_CLOSED", "eLIMIT_INFINITE", "eLIMIT_OPEN".

NOTE The appropriate MVCI system operation is MCDInterval::getLowerLimitIntervalType.

Return type:

otx:StringTerm

Parameters:

— <interval> : diagDataPlus:IntervalTerm [1..1]

This element addresses the I**nterval** (MCDInterval) whose lower limit type shall be returned.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

14.6.17.5 GetIntervalUpperLimitAsString

Semantic:

Returns the upper limit of the text table element of the parameter as a string.

NOTE The appropriate MVCI system operation is MCDInterval::getUpperLimit::getValueAsString.

Return type:

otx:StringTerm

Parameters:

— <interval> : diagDataPlus:IntervalTerm [1..1]

This element addresses the **Interval** (MCDInterval) whose upper limit shall be returned.

Exceptions:

— None

14.6.17.6 GetIntervalUpperLimitTypeAsString

Semantic:

Returns the upper limit type of the text table element of the parameter as a string. Valid values are "eLIMIT_CLOSED", "eLIMIT_INFINITE", "eLIMIT_OPEN".

NOTE The appropriate MVCI system operation is MCDInterval::getUpperLimitIntervalType.

Return type:

otx:StringTerm

Parameters:

— <interval> : diagDataPlus:IntervalTerm [1..1]

This element addresses the **Interval** (MCDInterval) whose upper limit type shall be returned.

Exceptions:

— None

14.6.17.7 IntervalTerm

Semantic:

An **IntervalTerm** is a term that returns an **Interval**.

14.6.17.8 IntervalValue

Semantic:

Returns the **Interval** stored in an **Interval** variable.

Return type:

diagDataPlus:IntervalTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

14.6.18 Semantics for McdDataType terms

Figure 58 shows the syntax of the OTX **DiagDataBrowsingPlus extension McdDataType** related terms.

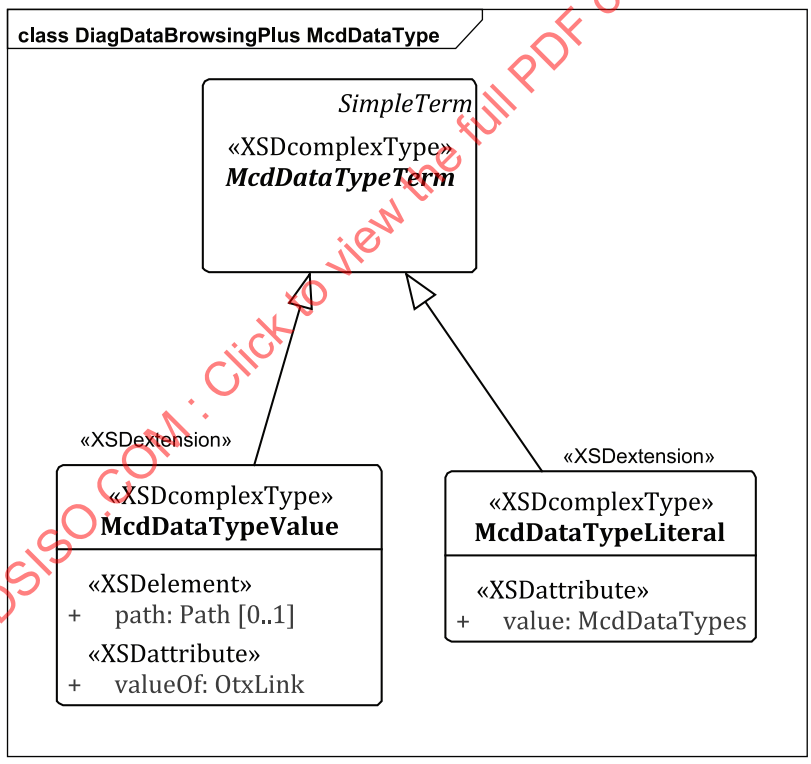


Figure 58 — DiagDataBrowsingPlus – McdDataType terms

14.6.18.1 McdDataTypeLiteral

Semantic:

Returns an **McdDataType** value from a hard-coded literal.

Return type:

diagDataPlus:McdDataTypeTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— value : diagDataPlus:McdDataTypes [1..1]

This value contains one of the values defined in the **McdDataTypes** enumeration.

Exceptions:

— None

14.6.18.2 McdDataTypeTerm

Semantic:

An **McdDataTypeTerm** is a term that returns an MCDDataType.

14.6.18.3 McdDataTypeValue

Semantic:

Returns the **McdDataType** stored in an **McdDataType** variable.

Return type:

diagDataPlus:McdDataTypeTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

14.6.19 Semantics for McdParameter terms

Figure 59 shows the syntax of the O**TX DiagDataBrowsingPlus extension McdParameter** related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

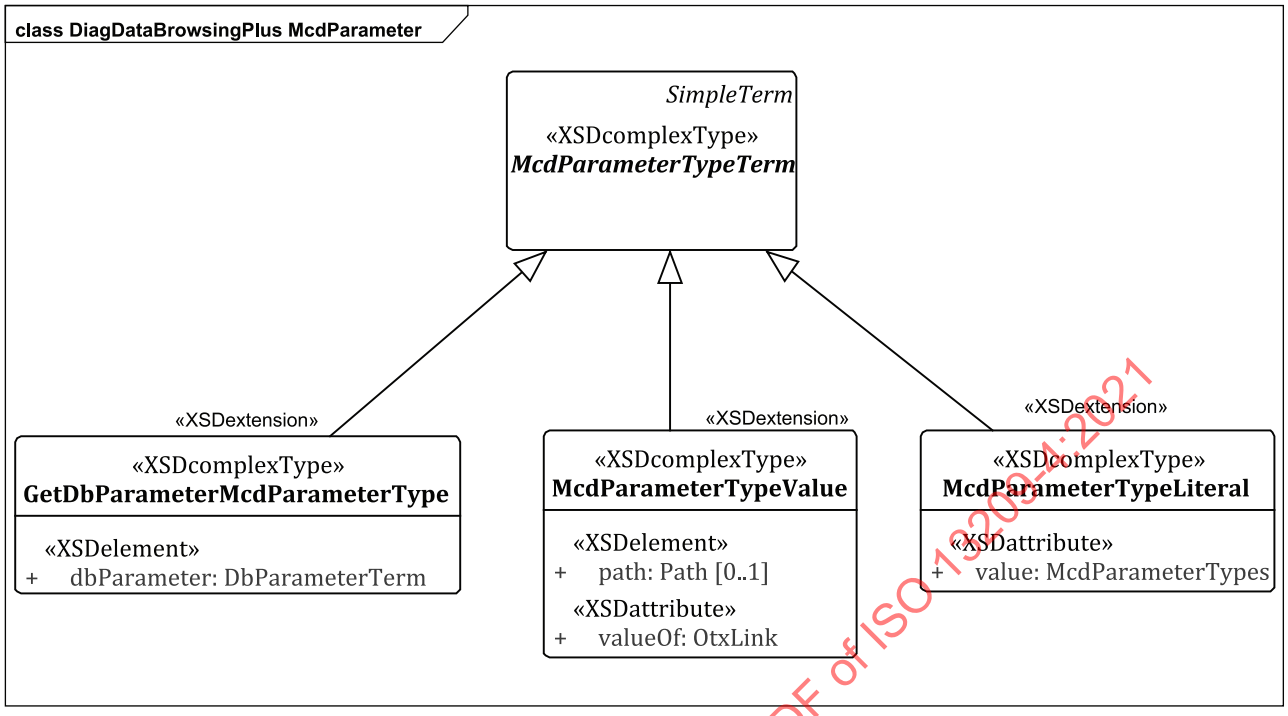


Figure 59 — DiagDataBrowsingPlus – McdParameterType terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

14.6.19.1 GetDbParameterMcdParameterType

Semantic:

Returns the type of the parameter (MCDParameterType).

NOTE The appropriate MVCI system operation is MCDDbParameter::getMCDParameterType.

Return type:

diagDataPlus:McdParameterTypeTerm

Parameters:

— <dbParameter>: diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose parameter type shall be returned.

Exceptions:

— None

14.6.19.2 McdParameterTypeLiteral

Semantic:

Returns an **McdParameterType** value from a hard-coded literal.

Return type:

diagDataPlus:McdParameterTypeTerm

© ISO 2021 – All rights reserved

Parameters:

— value : diagDataPlus:McdParameterTypes [1..1]

This value contains one of the values defined in the **McdParameterTypes** enumeration.

Exceptions:

— None

14.6.19.3 McdParameterTypeTerm

Semantic:

An **McdParameterTypeTerm** is a term that returns an **McdParameterType**.

14.6.19.4 McdParameterTypeValue

Semantic:

Returns the **McdParameterType** stored in an **McdParameterType** variable.

Return type:

diagDataPlus:McdParameterTypeTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

14.6.20 Semantics for McdResponseType terms

Figure 60 shows the syntax of the O**TX DiagDataBrowsingPlus extension McdResponseType** related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

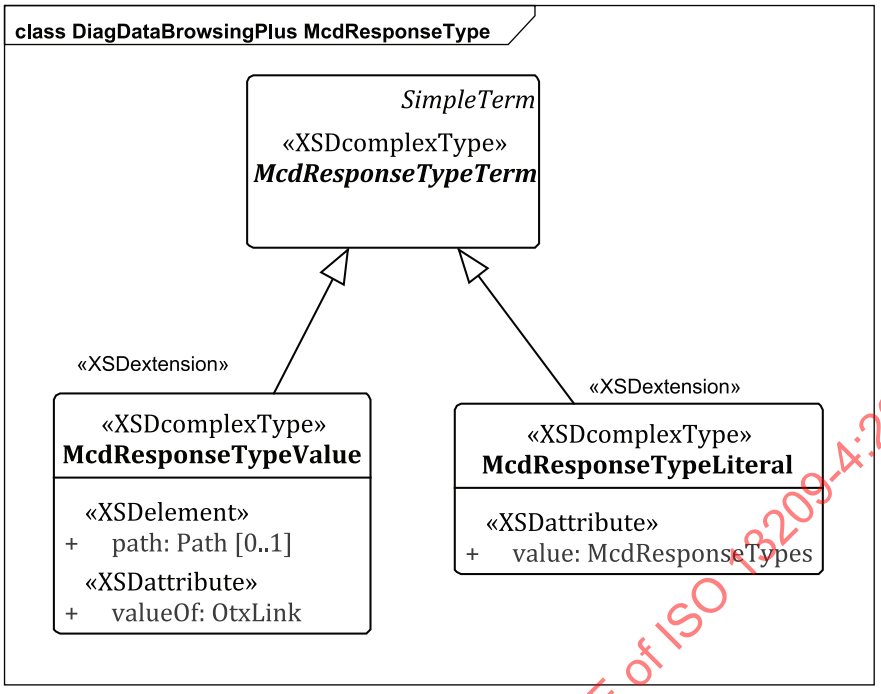


Figure 60 — DiagDataBrowsingPlus – McdResponseType terms

14.6.20.1 McdResponseTypeLiteral

Semantic:

Returns an **McdResponseType** value from a hard-coded literal.

Return type:

diagDataPlus:McdResponseTypeTerm

Parameters:

— value : diagDataPlus:McdResponseTypes [1..1]

This value contains one of the values defined in the **McdResponseTypes** enumeration.

Exceptions:

— None

14.6.20.2 McdResponseTypeTerm

Semantic:

An **McdResponseTypeTerm** is a term that returns an **McdResponseType**.

14.6.20.3 McdResponseTypeValue

Semantic:

Returns the **McdResponseType** stored in an **McdResponseType** variable.

Return type:

diagDataPlus:McdResponseTypeTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

14.6.21 Semantics for parameter terms

Figure 61 shows the syntax of the O**TX DiagDataBrowsingPlus extension Parameter** related terms.

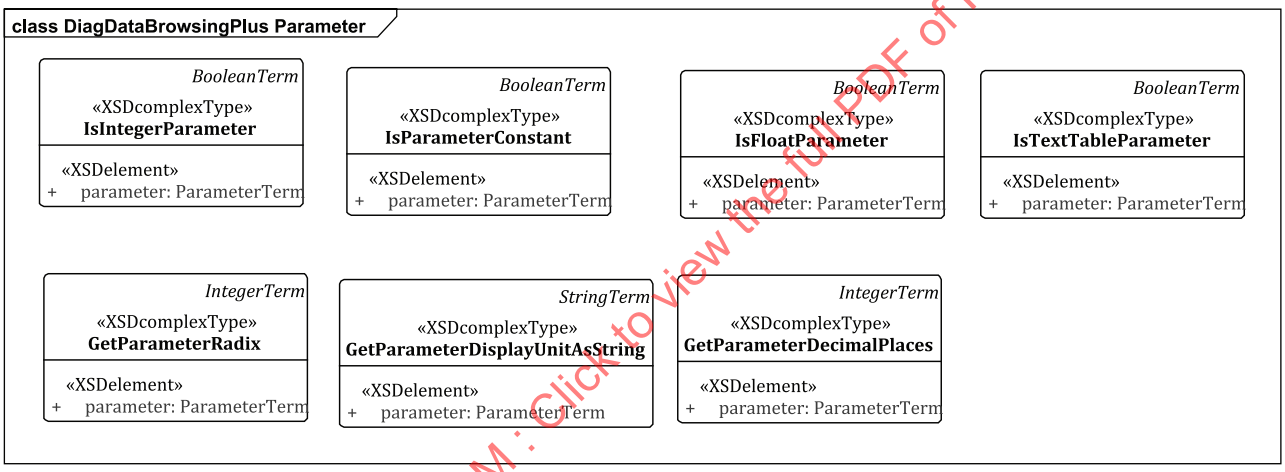


Figure 61 — DiagDataBrowsingPlus – Parameter terms

14.6.21.1 GetParameterDecimalPlaces

Semantic:

Returns the number of valid decimal places of the parameter. The number of decimal places shall be used by the Client to display only the valid decimal places of a float parameter as defined in the database. According to ODX, the number of decimal places is defined as the number of valid digits after the decimal point and not as precision of the floating point value.

NOTE 1 The appropriate MVCI system operation is MCDParameter::getDecimalPlaces.

NOTE 2 This term is introduced to complete the existing terms in the DiagDataBrowsing extension (ISO 13209-3) that uses ParameterTerms. This completion primarily enables a generically interpretation of responses by using runtime parameters without using DbParameters.

Return type:

otx:IntegerTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <parameter> : diag:ParameterTerm [1..1]

This element addresses the parameter (MCDParameter) whose decimal places shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter is not a float value (MCDDataType::FLOAT32 or MCDDataType::FLOAT64).

14.6.21.2 GetParameterDisplayUnitAsString

Semantic:

Returns the display unit information of the **diag:Parameter** as a string. If the parameter is not of simple data type or does not contain any unit information an empty string is returned. The unit is only available for parameters with simple data types.

NOTE 1 The appropriate MVCI system operation is MCDParameter::getUnit.

NOTE 2 This term is introduced to complete the existing terms in the DiagDataBrowsing extension (ISO 13209-3) that uses ParameterTerms. This completion primarily enables a generically interpretation of responses by using runtime parameters without using DbParameters.

Return type:

otx:StringTerm

Parameters:

— <parameter> : diag:ParameterTerm [1..1]

This element addresses the parameter (MCDParameter) whose display unit shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter is not a simple data type from A_UINT (8-64), A_INT (8-64), FLOAT (32,64), TEXTTABLE, BITFIELD, ASCIISTRING, UNICODE2STRING or BYTEFIELD.

14.6.21.3 GetParameterRadix

Semantic:

Returns the display radix of the parameter. The display radix shall be used by the client to display a numeric parameter value in the format as defined in the database (2 - ODX value BIN, 10 - ODX value DEC, 16 - ODX value HEX).

NOTE 1 The appropriate MVCI system operation is Operation MCDParameter::getRadix.

NOTE 2 This term is introduced to complete the existing terms in the DiagDataBrowsing extension (ISO 13209-3) that uses ParameterTerms. This completion primarily enables a generically interpretation of responses by using runtime parameters without using DbParameters.

Return type:

otx:IntegerTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <parameter> : diag:ParameterTerm [1..1]

This element addresses the parameter (MCDParameter) whose radix shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter is not an A_UINT or A_INT value.

14.6.21.4 IsFloatParameter

Semantic:

Returns true if the value of this **diag:Parameter** is of data type Float (FLOAT32, FLOAT64). In all other cases it returns false.

NOTE 1 The appropriate MVCI system operation is MCDParameter::getDataType.

NOTE 2 This term is introduced to complete the existing terms in the DiagDataBrowsing extension (ISO 13209-3) that uses ParameterTerms. This completion primarily enables a generically interpretation of responses by using runtime parameters without using DbParameters.

Return type:

otx:BooleanTerm

Parameters:

— <parameter> : diag:ParameterTerm [1..1]

This element addresses the **diag:Parameter** (MCDParameter) which is checked to determine if it is of data type Float or not.

Exceptions:

— None

14.6.21.5 IsIntegerParameter

Semantic:

Returns true if the value of this **diag:Parameter** is of data type Integer (INT16, INT32, INT64, INT8, UINT16, UINT32, UINT64, UINT8, DTC, END_OF_PDU, ENVDATA, ENVDATADESC, FIELD, MULTIPLEXER, STRUCTURE, LENGTHKEY, TABLE_ROW). In all other cases it returns false.

NOTE 1 The appropriate MVCI system operation is MCDParameter::getDataType.

NOTE 2 This term is introduced to complete the existing terms in the DiagDataBrowsing extension (ISO 13209-3) that uses ParameterTerms. This completion primarily enables a generically interpretation of responses by using runtime parameters without using DbParameters.

Return type:

otx:BooleanTerm

Parameters:

— <parameter> : diag:ParameterTerm [1..1]

This element addresses the **diag:Parameter** (MCDParameter) which is checked to determine if it is of data type Integer or not.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Exceptions:

— None

14.6.21.6 IsParameterConstant

Semantic:

Returns true if this parameter is of type CODED-CONST, NRC_CONST or PHYS-CONST. In all other cases it returns false.

NOTE 1 The appropriate MVCI system operation is MCDParameter::getMCDParameterType.

NOTE 2 This term is introduced to complete the existing terms in the DiagDataBrowsing extension (ISO 13209-3) that uses ParameterTerms. This completion primarily enables a generically interpretation of responses by using runtime parameters without using DbParameters.

Return type:

otx:BooleanTerm

Parameters:

— <parameter> : diag:ParameterTerm [1..1]

This element addresses the **diag:Parameter** (MCDParameter) which is checked to determine if it is constant or not.

Exceptions:

— None

14.6.21.7 IsTextTableParameter

Semantic:

Returns true whether the value of this **diag:Parameter** is of data type TextTable (eTEXTTABLE). In all other cases it returns false.

NOTE 1 The appropriate MVCI system operation is MCDParameter::getDataType.

NOTE 2 This term is introduced to complete the existing terms in the DiagDataBrowsing extension (ISO 13209-3) that uses ParameterTerms. This completion primarily enables a generically interpretation of responses by using runtime parameters without using DbParameters.

Return type:

otx:BooleanTerm

Parameters:

— <parameter> : diag:ParameterTerm [1..1]

This element addresses the **diag:Parameter** (MCDParameter) which is checked to determine if it is of data type TextTable or not.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

14.6.22 Semantics for TextTableElement terms

Figure 62 shows the syntax of the **OTX DiagDataBrowsingPlus extension TextTableElement** related terms.

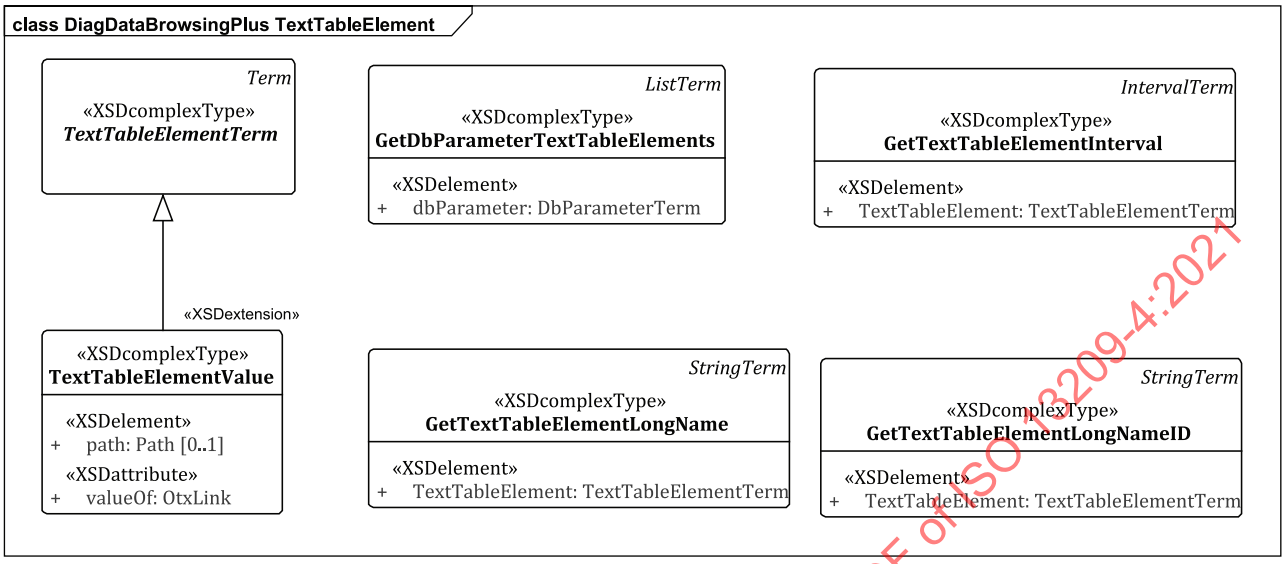


Figure 62 — DiagDataBrowsingPlus – TextTableElement terms

14.6.22.1 GetDbParameterTextTableElements

Semantic:

Returns all table entries of the parameter, as a list of DbTextTableElements.

NOTE The appropriate MVCI system operation is MCDDbParameter::getTextTableElements.

Return type:

otx:ListTerm

Parameters:

— <dbParameter> : diagDataPlus:DbParameterTerm [1..1]

This element addresses the **DbParameter** (MCDDbParameter) whose text table elements shall be returned.

Exceptions:

InvalidTypeException

— This exception is thrown if the parameter is not a text table.

14.6.22.2 GetTextTableElementInterval

Semantic:

Returns the **Interval** of the text table element.

NOTE The appropriate MVCI system operation is MCDTextTableElement::getInterval.

Return type:

diagDataPlus:IntervalTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <TextTableElement> : diagDataPlus:TextTableElementTerm [1..1]

This element addresses the **TextTableElement** (MCDTextTableElement) whose interval shall be returned.

Exceptions:

— None

14.6.22.3 GetTextTableElementLongName

Semantic:

Returns the LONG-NAME of the text table element.

NOTE The appropriate MVCI system operation is MCDTextTableElement::getLongName.

Return type:

otx:StringTerm

Parameters:

— <TextTableElement> : diagDataPlus:TextTableElementTerm [1..1]

This element addresses the **TextTableElement** (MCDTextTableElement) whose LONG-NAME shall be returned.

Exceptions:

— None

14.6.22.4 GetTextTableElementLongNameID

Semantic:

Returns the ID for the LONG-NAME of the text table element.

NOTE The appropriate MVCI system operation is MCDTextTableElement::getLongNameID.

Return type:

otx:StringTerm

Parameters:

— <TextTableElement> : diagDataPlus:TextTableElementTerm [1..1]

This element addresses the **TextTableElement** (MCDTextTableElement) whose LONG-NAME ID shall be returned.

Exceptions:

— None

14.6.22.5 TextTableElementTerm

Semantic:

A **TextTableElementTerm** is a term that returns a **TextTableElement**.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

14.6.22.6 TextTableElementValue

Semantic:

Returns the **TextTableElement** stored in a **TextTableElement** variable.

Return type:

diagDataPlus:TextTableElementTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

15 OTX EcuConfiguration extension

15.1 General

ECU configuration, also known as variant coding describes the data elements and the process of configuring an ECU – either in the vehicle or in a test bench.

The ECU configuration extension is introduced to enable a simple way to completely configure an ECU. The configuration of an ECU often consists of more than one data block in the ECU. These data blocks for their parts often consist of more than one configurable value. In general each of these values can be changed to configure an ECU so that its behaviour matches its environment.

The method that is enabled by the ECUConfiguration extension simplifies the configuration. That means the complete data, that should be written to an ECU to configure it, can be retrieved very easily from the database (ODX data). For example, an ID like a part number can be used to decide which data should be written to an ECU to configure it completely. The diagnostic services that are needed to write and read the configuration data can automatically be created.

IMPORTANT — The ECUConfiguration extension can only be implemented in a runtime system that is based on an MVCI base system that supports the optional ECU configuration feature because the terms and actions are mapped to specific MVCI API calls of this feature.

Some new complex types of the extension derive from the DbObject type (which is defined in the DiagDataBrowsingPlus extension). To get SHORT-NAMEs and additional information of these objects items of the DiagDataBrowsing extension have to be used.

15.2 Datatypes

15.2.1 Datatypes overview

All data types introduced by this extension are derived from **otx:ComplexType**.

© ISO 2021 – All rights reserved

15.2.2 Datatypes syntax

The syntax of the data-type declarations of the **OTX EcuConfiguration extension** is shown in Figure 63.

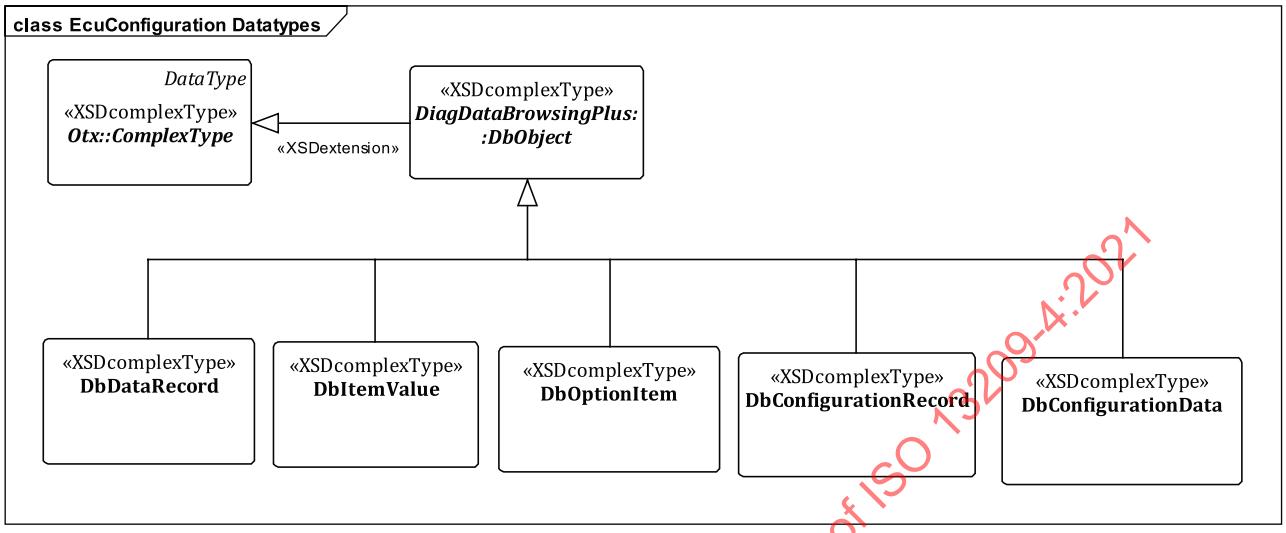


Figure 63 — EcuConfiguration datatypes

15.2.3 Datatypes semantics

15.2.3.1 DbConfigurationData

DbConfigurationData is a **diagDataPlus:DbObject**.

This element represents configuration data of an ECU in the database.

15.2.3.2 DbConfigurationRecord

DbConfigurationRecord is a **diagDataPlus:DbObject**.

This element represents one set of possible configuration segments and the possible values of the segments in the diagnostic database of an ECU. It also contains DbDataRecords that contains the predefined value sets for a configuration.

15.2.3.3 DbDataRecord

DbDataRecord is a **diagDataPlus:DbObject**.

This element represents a set of predefined value sets for a configuration.

15.2.3.4 DbItemValue

DbItemValue is a **diagDataPlus:DbObject**.

This element represents a variable parameter of an ITEM-VALUE from ODX, which has a number of possible values.

15.2.3.5 DbOptionItem

DbOptionItem is a **diagDataPlus:DbObject**.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

This element represents a variable parameter of a CONFIG-RECORD from ODX, which has a number of possible values.

15.3 Exceptions

15.3.1 Exceptions overview

All exceptions introduced by this extension are derived from the **otx:Exception** type. They represent the full set of exceptions added by the **OTX EcuConfiguration extension**.

15.3.2 Exceptions syntax

The syntax of all **OTX EcuConfiguration extension** exception-type declarations is shown in Figure 64.

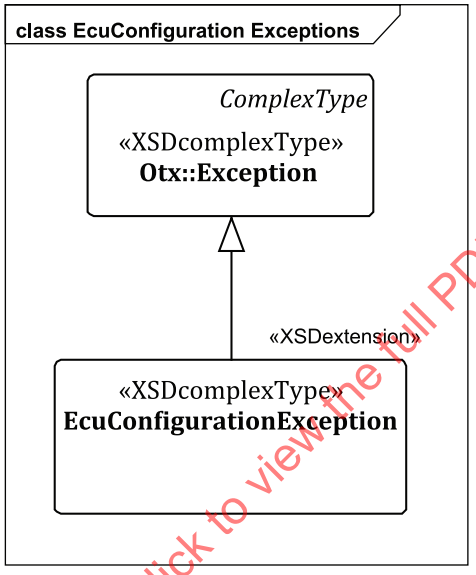


Figure 64 — EcuConfiguration exceptions

15.3.3 Exceptions semantics

15.3.3.1 General

Since all **OTX EcuConfiguration extension** exception types are implicit exceptions without initialisation parts, they cannot be declared constant.

15.3.3.2 EcuConfigurationException

IMPORTANT — All terms and action realisations in this extension may potentially throw this exception.

15.4 Variable access

15.4.1 Variable-access overview

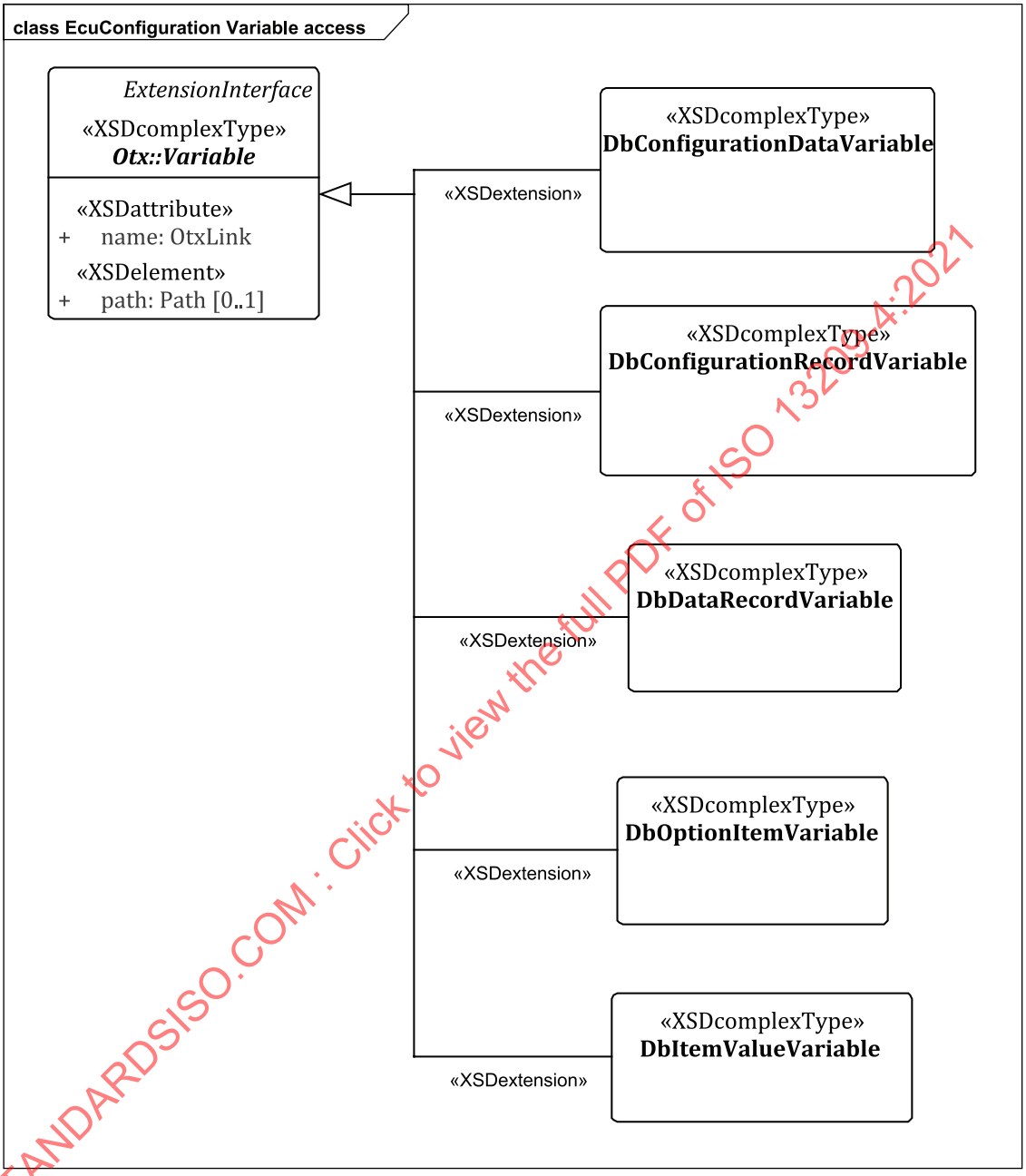
All variable-access types introduced by this extension are derived from the OTX core variable extension interface.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

15.4.2 Variable-access syntax

The syntax of the **OTX EcuConfiguration extension** variable-access types is shown in Figure 65.



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 65 — EcuConfiguration variable access

15.4.3 Variable-access semantics

The general semantics for all variable-access types shall apply. Please refer to ISO 13209-2 for details.

15.5 Terms

15.5.1 Terms overview

All terms introduced by this extension extend the **otx:Term** extension interface. Information about the specific super class of a term is provided in the individual term description clauses.

© ISO 2021 – All rights reserved

15.5.2 Semantics for DbConfigurationData terms

Figure 66 shows the syntax of the **OTX EcuConfiguration extension DbConfigurationData** related terms.

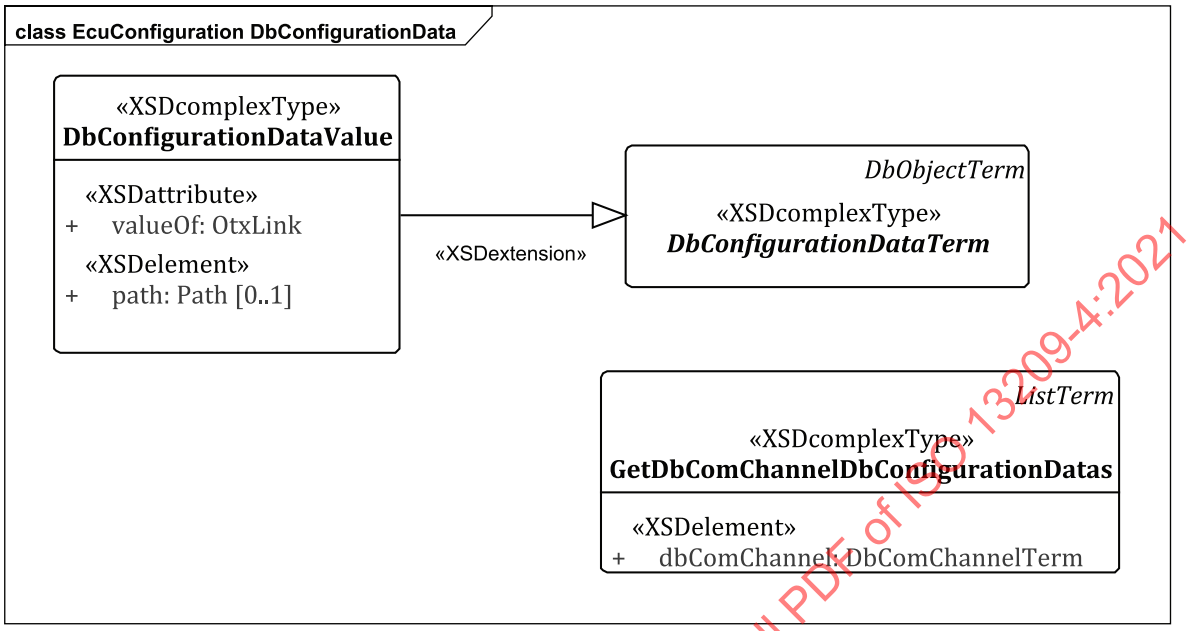


Figure 66 — ECUConfiguration – DbConfigurationData terms

15.5.2.1 DbConfigurationDataTerm

Semantic:

The abstract type **DbConfigurationDataTerm** is a **dataPlus:DbObjectTerm.** It serves as a base for all concrete terms, which return a **DbConfigurationData** value. It has no special members.

15.5.2.2 DbConfigurationDataValue

Semantic:

Returns the **DbConfigurationData** value stored in a **DbConfigurationData** variable. For more information on value-terms and the syntax and semantics of the valueOf attribute and <path> element, please refer to ISO 13209-2.

Associated checker rules:

— Core_Chk053 – no dangling OtxLink associations (see ISO 13209-2).

Return type:

ecuConf:DbConfigurationDataTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— If the variable value is not valid (no value was assigned to the variable before).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location, which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

15.5.2.3 GetDbComChannelDbConfigurationDatas

Semantic:

Returns the list of **DbConfigurationData** objects of the ECU database represented by the **diagDataPlus:dbComChannel.**

NOTE The appropriate MVCI system operation is DbLocation.getDbConfigurationDatas.

Return type:

otx:ListTerm

Parameters:

— <dbComChannel> : diagDataPlus:DbComChannelTerm [1..1]

This element represents the communication channel (as database object) whose DbConfigurationDatas should be listed.

Exceptions:

otx:TypeMismatchException

— This exception is thrown if the communication channel belongs to the category PROTOCOL or FUNCTIONAL_GROUP.

15.5.3 Semantics for DbConfigurationRecord terms

Figure 67 shows the syntax of the **OTX EcuConfiguration extension DbConfigurationRecord** related terms.

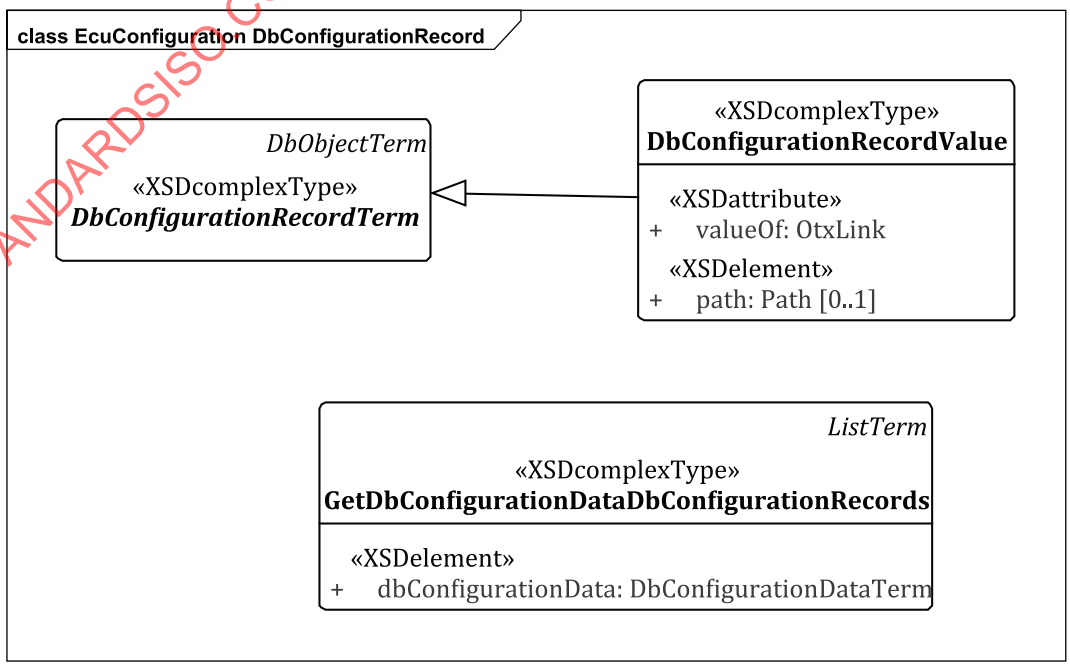


Figure 67 — ECUConfiguration – DbConfigurationRecord terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

15.5.3.1 DbConfigurationRecordTerm

Semantic:

The abstract type **DbConfigurationRecordTerm** is a **dataPlus:DbObjectTerm.** It serves as a base for all concrete terms, which return a **DbConfigurationRecord** value. It has no special members.

15.5.3.2 DbConfigurationRecordValue

Semantic:

Returns the **DbConfigurationRecord** value stored in a **DbConfigurationRecord** variable. For more information on value-terms and the syntax and semantics of the valueOf attribute and <path> element, please refer to ISO 13209-2.

Associated checker rules:

— Core_Chk053 – no dangling OtxLink associations (see ISO 13209-2).

Return type:

ecuConf:DbConfigurationRecordTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value was assigned to the variable before).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location, which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

15.5.3.3 GetDbConfigurationDataDbConfigurationRecords

Semantic:

Returns the list of **DbConfigurationRecord** objects of the **DbConfigurationData**.

NOTE The appropriate MVCI system operation is ConfigurationData.getDbConfigurationRecords.

Return type:

otx:ListTerm

Parameters:

— <dbConfigurationData> : ecuConf:DbConfigurationDataTerm [1..1]

This element represents the **DbConfigurationData** whose DbConfigurationRecords should be listed.

Exceptions:

diag:UnknownTargetException

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

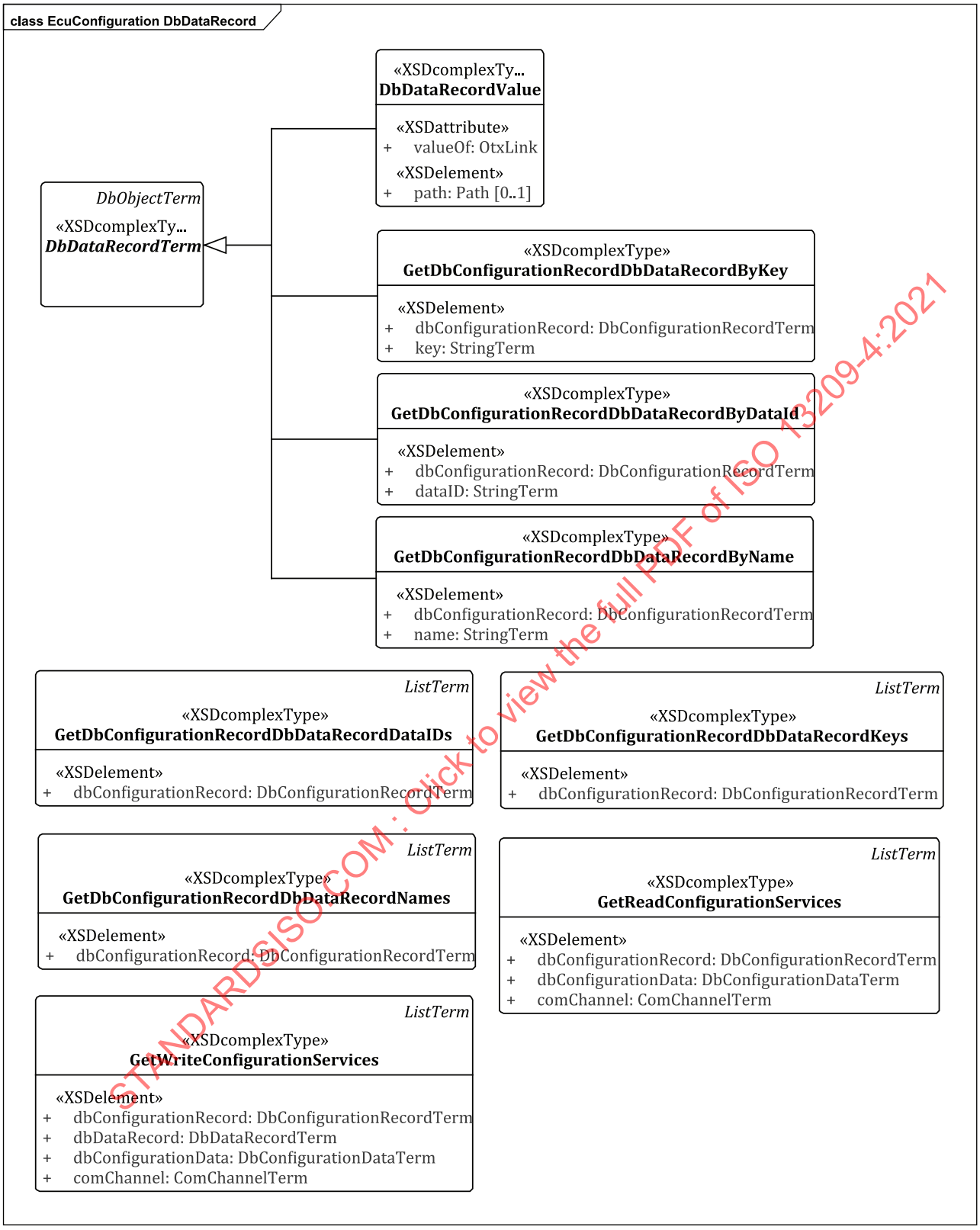
— This exception is thrown if the term references an object in the underlying system that is not available or not defined.

15.5.4 Semantics for DbDataRecord terms

Figure 68 shows the syntax of the **OTX EcuConfiguration extension DbDataRecord** related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 68 — ECUConfiguration – DbDataRecord terms

15.5.4.1 DbDataRecordTerm

Semantic:

© ISO 2021 – All rights reserved

The abstract type **DbDataRecordTerm** is a **dataPlus:DbObjectTerm.** It serves as a base for all concrete terms, which return a **DbDataRecord** value. It has no special members.

15.5.4.2 DbDataRecordValue

Semantic:

Returns the **DbDataRecord** value stored in a **DbDataRecord** variable. For more information on value-terms and the syntax and semantics of the valueOf attribute and <path> element, please refer to ISO 13209-2.

Associated checker rules:

— Core_Chk053 – no dangling OtxLink associations (see ISO 13209-2).

Return type:

ecuConf:DbDataRecordTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value was assigned to the variable before).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location, which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

15.5.4.3 GetDbConfigurationRecordDbDataRecordByDataId

Semantic:

Returns the **DbDataRecord** object of the **DbConfigurationRecord** by data id.

NOTE The appropriate MVCI system operation is DbDataRecords::getItemByDataID.

Return type:

ecuConf:DbDataRecordTerm

Parameters:

— <dbConfigurationRecord> : ecuConf:DbConfigurationRecordTerm [1..1]

This element represents the **DbConfigurationRecord** whose **DbDataRecord** should be returned by the given data id.

— <dataID> : otx:StringTerm [1..1]

This element represents the data ID of the **DbDataRecord** that shall be returned.

Exceptions:

diag:UnknownTargetException

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— This exception is thrown if the term references an object in the underlying system that is not available or not defined.

15.5.4.4 GetDbConfigurationRecordDbDataRecordByKey

Semantic:

Returns the **DbDataRecord** object of the **DbConfigurationRecord** by key.

NOTE The appropriate MVCI system operation is DbDataRecords.getItemByKey.

Return type:

ecuConf:DbDataRecordTerm

Parameters:

— <dbConfigurationRecord> : ecuConf:DbConfigurationRecordTerm [1..1]

This element represents the **DbConfigurationRecord** whose **DbDataRecord** should be returned by the given key.

— <key> : otx:StringTerm [1..1]

This element represents the key of the **DbDataRecord** that shall be returned.

Exceptions:

diag:UnknownTargetException

— This exception is thrown if the term references an object in the underlying system that is not available or not defined.

15.5.4.5 GetDbConfigurationRecordDbDataRecordByName

Semantic:

Returns the **DbDataRecord** object of the **DbConfigurationRecord** by name.

NOTE The appropriate MVCI system operation is DbDataRecords.getItemByName.

Return type:

ecuConf:DbDataRecordTerm

Parameters:

— <dbConfigurationRecord> : ecuConf:DbConfigurationRecordTerm [1..1]

This element represents the **DbConfigurationRecord** whose **DbDataRecord** should be returned by the given name.

— <name> : otx:StringTerm [1..1]

This element represents the name of the **DbDataRecord** that shall be returned.

Exceptions:

diag:UnknownTargetException

— This exception is thrown if the term references an object in the underlying system that is not available or not defined.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

15.5.4.6 GetDbConfigurationRecordDbDataRecordDataIDs

Semantic:

Returns a list of the available **DbDataRecord** data ids as Strings of the **DbConfigurationRecord**.

NOTE The appropriate MVCI system operation is DbConfigurationRecord::getDbDataRecords::getItemByIndex::getDataId.

Return type:

otx:ListTerm

Parameters:

— <dbConfigurationRecord> : ecuConf:DbConfigurationRecordTerm [1..1]

This element represents the **DbConfigurationRecord** whose **DbDataRecord** data ids should be listed.

Exceptions:

diag:UnknownTargetException

— This exception is thrown if the term references an object in the underlying system that is not available or not defined.

15.5.4.7 GetDbConfigurationRecordDbDataRecordKeys

Semantic:

Returns a list of the available **DbDataRecord** keys as strings of the **DbConfigurationRecord**.

NOTE The appropriate MVCI system operation is DbConfigurationRecord. getDbDataRecords::getItemByIndex::getKey.

Return type:

otx:ListTerm

Parameters:

— <dbConfigurationRecord> : ecuConf:DbConfigurationRecordTerm [1..1]

This element represents the **DbConfigurationRecord** whose DbDataRecordKeys should be listed.

Exceptions:

diag:UnknownTargetException

— This exception is thrown if the term references an object in the underlying system that is not available or not defined.

15.5.4.8 GetDbConfigurationRecordDbDataRecordNames

Semantic:

Returns a list of the available **DbDataRecord** names as Strings of the **DbConfigurationRecord**.

NOTE The appropriate MVCI system operation is DbConfigurationRecord. getDbDataRecords::getItemByIndex::getShortName.

Return type:

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

otx:ListTerm

Parameters:

— <dbConfigurationRecord> : ecuConf:DbConfigurationRecordTerm [1..1]

This element represents the **DbConfigurationRecord** whose DbDataRecordNames should be listed.

Exceptions:

diag:UnknownTargetException

— This exception is thrown if the term references an object in the underlying system that is not available or not defined.

15.5.4.9 GetReadConfigurationServices

Semantic:

Returns the list of configured **diag:DiagService** objects for reading the configuration of an ECU by **DbConfigurationRecord**, **DbConfigurationData** and **diag:comChannel.**

NOTE The appropriate MVCI system operation is LogicalLink.getConfigurationRecords(). addByDbObject(DbConfigurationRecord)::getReadDiagComPrimitives.

Return type:

otx:ListTerm

Parameters:

— <dbConfigurationData> : ecuConf:DbConfigurationDataTerm [1..1]

This element represents the **DbConfigurationData** of the ECU in the database.

— <dbConfigurationRecord> : ecuConf:DbConfigurationRecordTerm [1..1]

This element represents the **DbConfigurationRecord** whose read configuration services should be listed.

— <comChannel> : diag:ComChannelTerm [1..1]

This element represents the communication channel to an ECU whose DiagServices should be listed.

Exceptions:

diag:UnknownTargetException

— This exception is thrown if the term references an object in the underlying communication system that is not available or not defined.

15.5.4.10 GetWriteConfigurationServices

Semantic:

Returns the list of configured **diagDataPlus:DiagServices** for writing the configuration of an ECU through **DbDataRecord**, **DbConfigurationRecord**, **DbConfigurationData** and **diag:comChannel.**

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

NOTE The related MVCI system methods are

ConfigurationRecord.LogicalLink::getConfigurationRecords()::addByDbObject(DbConfiguration-Record):: setConfigurationRecordByDbObject(DbDataRecord); ConfigurationRecord::getWriteDiagComPrimitives.

Return type:

otx:ListTerm

Parameters:

— <dbConfigurationRecord> : ecuConf:DbConfigurationRecordTerm [1..1]

This element represents the **DbConfigurationRecord** whose write configuration services should be listed.

— <dbDataRecord> : ecuConf:DbDataRecordTerm [1..1]

This element represents the **DbDataRecord** that parameterise the write configuration services that should be listed.

— <dbConfigurationData> : ecuConf:DbConfigurationDataTerm [1..1]

This element represents the **DbConfigurationData** of the ECU in the database.

— <comChannel> : diag:ComChannelTerm [1..1]

This element represents the communication channel to an ECU whose DiagServices should be listed.

Exceptions:

diag:UnknownTargetException

— This exception is thrown if the term references an object in the underlying system that is not available or not defined.

15.5.5 Semantics for DbItemValue terms

Figure 69 shows the syntax of the **OTX EcuConfiguration extension DbItemValue** related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

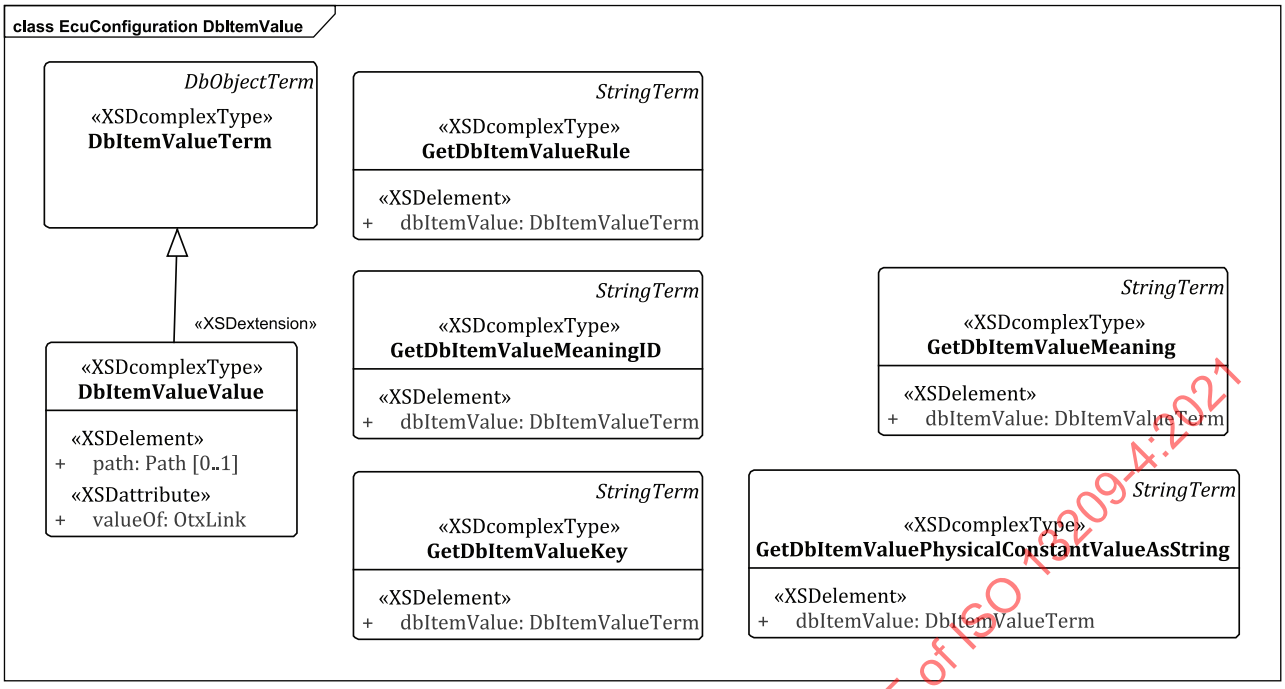


Figure 69 — ECUConfiguration – DbItemValue terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

15.5.5.1 DbItemValueTerm

Semantic:

The abstract type **DbItemValueTerm** is a **dataPlus:DbObjectTerm.** It serves as a base for all concrete terms, which return a **DbItemValue** value. It has no special members.

Return type:

ecuConf:DbItemValueTerm

Parameters:

— None

Exceptions:

— None

15.5.5.2 DbItemValueValue

Semantic:

Returns the **DbItemValue** value stored in a **DbItemValue** variable. For more information on value-terms and the syntax and semantics of the valueOf attribute and <path> element, please refer to ISO 13209-2.

Associated checker rules:

— Core_Chk053 – no dangling OtxLink associations (see ISO 13209-2).

Return type:

ecuConf:DbItemValueTerm

© ISO 2021 – All rights reserved

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value was assigned to the variable before).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location, which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

15.5.5.3 GetDbItemValueKey

Semantic:

Returns the key as an **otx:StringTerm.**

NOTE The appropriate MVCI system operation is MCDDbItemValue.getKey.

Return type:

otx:StringTerm

Parameters:

— <dbItemValue> : ecuConf:DbItemValueTerm [1..1]

This element represents the **DbItemValue** whose key should be returned.

Exceptions:

— None

15.5.5.4 GetDbItemValueMeaning

Semantic:

Returns the meaning as a string.

NOTE The appropriate MVCI system operation is MCDDbItemValue.getMeaning.

Return type:

otx:StringTerm

Parameters:

— <dbItemValue> : ecuConf:DbItemValueTerm [1..1]

This element represents the DbItemValue whose meaning should be returned.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

15.5.5.5 GetDbItemValueMeaningID

Semantic:

Returns the meaning ID as a string.

NOTE The appropriate MVCI system operation is MCDDbItemValue.getMeaningID.

Return type:

otx:StringTerm

Parameters:

— <dbItemValue> : ecuConf:DbItemValueTerm [1..1]

This element represents the **DbItemValue** whose meaning ID should be returned.

Exceptions:

— None

15.5.5.6 GetDbItemValuePhysicalConstantValueAsString

Semantic:

Returns the physical constant value of the item value as a string.

NOTE The appropriate MVCI system operation is MCDDbItemValue.getPhysicalConstantValue.

Return type:

otx:StringTerm

Parameters:

— <dbItemValue> : ecuConf:DbItemValueTerm [1..1]

This element represents the **DbItemValue** whose physical constant value should be returned.

Exceptions:

— None

15.5.5.7 GetDbItemValueRule

Semantic:

Returns the rule as a string.

NOTE The appropriate MVCI system operation is MCDDbItemValue.getRule.

Return type:

otx:StringTerm

Parameters:

— <dbItemValue> : ecuConf:DbItemValueTerm [1..1]

This element represents the **DbItemValue** whose rule should be returned.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

15.5.6 Semantics for DbOptionItem terms

Figure 70 shows the syntax of the **OTX EcuConfiguration extension DbOptionItem** related terms.

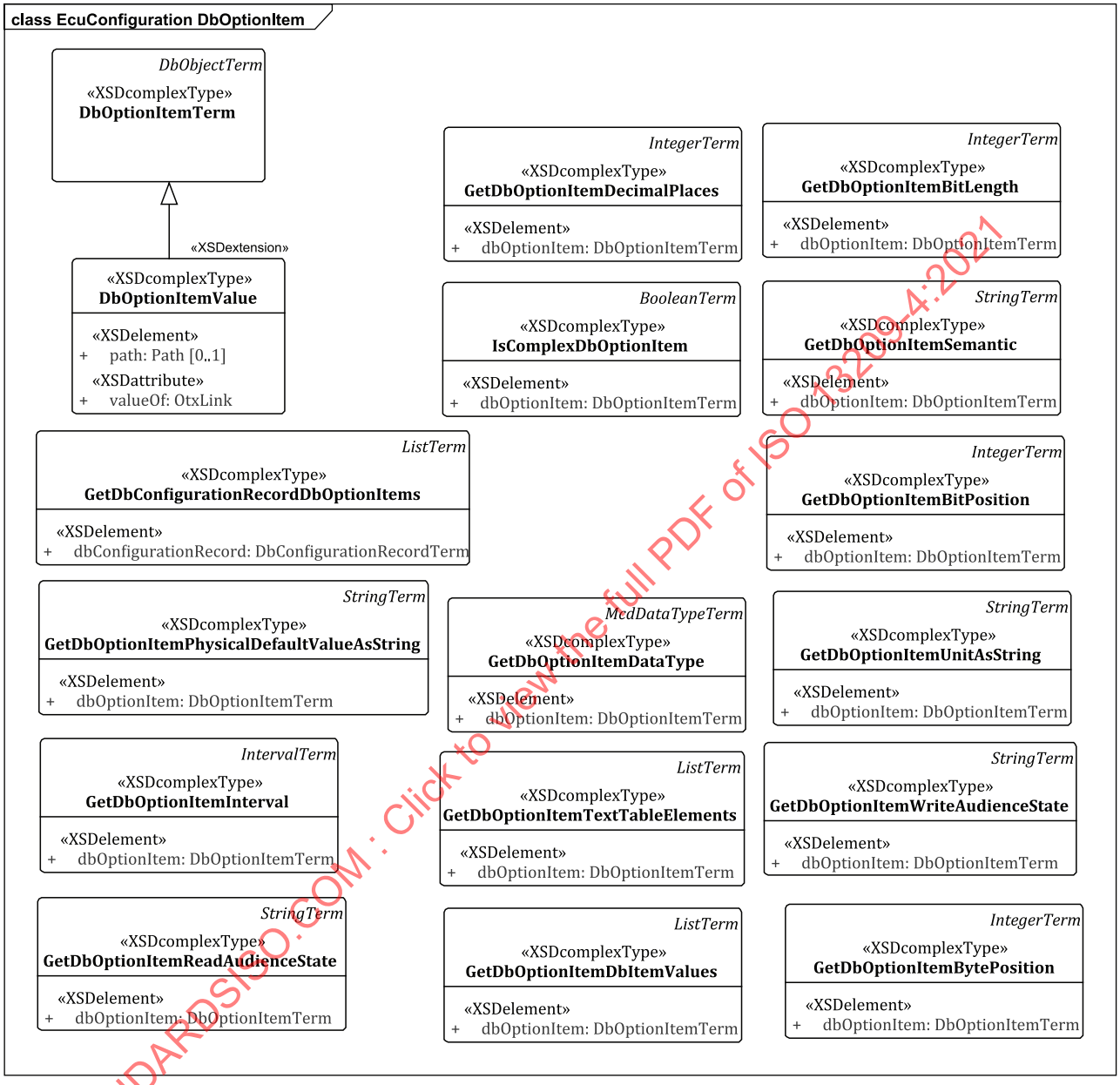


Figure 70 — ECUConfiguration – DbOptionItem terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

15.5.6.1 DbOptionItemTerm

Semantic:

The abstract type **DbOptionItemTerm** is a **dataPlus:DbObjectTerm.** It serves as a base for all concrete terms, which return a **DbConfigurationData** value. It has no special members.

Return type:

ecuConf:DbOptionItemTerm

© ISO 2021 – All rights reserved

Parameters:

— None

Exceptions:

— None

15.5.6.2 DbOptionItemValue

Semantic:

Returns the **DbOptionItem** value stored in a **DbOptionItem** variable. For more information on value-terms and the syntax and semantics of the valueOf attribute and <path> element, please refer to ISO 13209-2.

Associated checker rules:

— Core_Chk053 – no dangling OtxLink associations (see ISO 13209-2).

Return type:

ecuConf:DbOptionItemTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value was assigned to the variable before).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location, which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

15.5.6.3 GetDbConfigurationRecordDbOptionItems

Semantic:

Returns the list of **DbOptionItem** objects of the **DbConfigurationRecord**.

NOTE The appropriate MVCI system operation is MCDDbConfigurationRecord.getDbOptionItems.

Return type:

otx:ListTerm

Parameters:

— <dbConfigurationRecord> : ecuConf:DbConfigurationRecordTerm [1..1]

This element represents the **DbConfigurationRecord** whose DbOptionItems should be listed.

Exceptions:

diag:UnknownTargetException

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

— This exception is thrown if the term references an object in the underlying system that is not available or not defined.

15.5.6.4 GetDbOptionItemBitLength

Semantic:

Returns the bit length of the option item in the PDU as defined in ODX.

NOTE The appropriate MVCI system operation is MCDDbOptionItem.getBitLength.

Return type:

otx:IntegerTerm

Parameters:

— <dbOptionItem> : ecuConf:DbOptionItemTerm [1..1]

This element represents the **DbOptionItem** whose Bit-length should be returned.

Exceptions:

— None

15.5.6.5 GetDbOptionItemBitPosition

Semantic:

Returns the bit position.

NOTE The appropriate MVCI system operation is MCDDbOptionItem.getBitPos.

Return type:

otx:IntegerTerm

Parameters:

— <dbOptionItem> : ecuConf:DbOptionItemTerm [1..1]

This element represents the **DbOptionItem** whose Bit position should be returned.

Exceptions:

— None

15.5.6.6 GetDbOptionItemBytePosition

Semantic:

Returns the byte position.

NOTE The appropriate MVCI system operation is MCDDbOptionItem.getBytePos.

Return type:

otx:IntegerTerm

Parameters:

— <dbOptionItem> : ecuConf:DbOptionItemTerm [1..1]

This element represents the **DbOptionItem** whose Byte position should be returned.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Exceptions:

— None

15.5.6.7 GetDbOptionItemDataType

Semantic:

Returns the data type of the option item as a **diagDataPlus:McdDataTypeTerm.**

NOTE The appropriate MVCI system operation is MCDDbOptionItem.getDataType.

Return type:

diagDataPlus:McdDataTypeTerm

Parameters:

— <dbOptionItem> : ecuConf:DbOptionItemTerm [1..1]

This element represents the **DbOptionItem** whose data type should be returned.

Exceptions:

— None

15.5.6.8 GetDbOptionItemDbItemValues

Semantic:

Returns all item value objects of the option item, as a List of **DbItemValue**.

NOTE The appropriate MVCI system operation is MCDDbOptionItem.getDbItemValues.

Return type:

otx:ListTerm

Parameters:

— <dbOptionItem> : ecuConf:DbOptionItemTerm [1..1]

This element represents the **DbOptionItem** whose item values should be returned.

Exceptions:

diag: UnknownTargetException

— This exception is thrown if the term references an object in the underlying system that is not available or not defined

15.5.6.9 GetDbOptionItemDecimalPlaces

Semantic:

Returns the number of valid decimal places of the option item.

NOTE The appropriate MVCI system operation is MCDDbOptionItem.getDecimalPlaces.

Return type:

otx:IntegerTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Parameters:

— <dbOptionItem> : ecuConf:DbOptionItemTerm [1..1]

This element represents the **DbOptionItem** whose decimal places should be returned.

Exceptions:

— None

15.5.6.10 GetDbOptionItemInterval

Semantic:

Returns the interval of the option item.

NOTE The appropriate MVCI system operation is MCDDbOptionItem.getInterval.

Return type:

diagDataPlus:IntervalTerm

Parameters:

— <dbOptionItem> : ecuConf:DbOptionItemTerm [1..1]

This element represents the **DbOptionItem** whose interval should be returned.

Exceptions:

— None

15.5.6.11 GetDbOptionItemPhysicalDefaultValueAsString

Semantic:

Returns the physical default value of the option item as a string.

NOTE The appropriate MVCI system operation is MCDDbOptionItem.getPhysicalDefaultValue.

Return type:

otx:StringTerm

Parameters:

— <dbOptionItem> : ecuConf:DbOptionItemTerm [1..1]

This element represents the **DbOptionItem** whose physical default value should be returned.

Exceptions:

— None

15.5.6.12 GetDbOptionItemReadAudienceState

Semantic:

Returns the read audience state of this option item (as defined in the ODX data) as a string.

NOTE The appropriate MVCI system operation is MCDDbOptionItem.getReadAudienceState.

Return type:

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

otx:StringTerm

Parameters:

— <dbOptionItem> : ecuConf:DbOptionItemTerm [1..1]

This element represents the **DbOptionItem** whose read audience state should be returned.

Exceptions:

— None

15.5.6.13 GetDbOptionItemSemantic

Semantic:

Returns the semantic attribute defined in ODX.

NOTE The appropriate MVCI system operation is MCDDbOptionItem.getSemantic.

Return type:

otx:StringTerm

Parameters:

— <dbOptionItem> : ecuConf:DbOptionItemTerm [1..1]

This element represents the **DbOptionItem** whose semantic should be returned.

Exceptions:

— None

15.5.6.14 GetDbOptionItemTextTableElements

Semantic:

Returns all table entries of the option item, as a List of **diagDataPlus:TextTableElement.**

NOTE The appropriate MVCI system operation is MCDDbOptionItem.getTextTableElements.

Return type:

otx:ListTerm

Parameters:

— <dbOptionItem> : ecuConf:DbOptionItemTerm [1..1]

This element represents the **DbOptionItem** whose text table elements should be returned.

Exceptions:

— None

15.5.6.15 GetDbOptionItemUnitAsString

Semantic:

Returns the unit information of the option item as a string.

NOTE The appropriate MVCI system operation is MCDDbOptionItem.getDbUnit.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Return type:

otx:StringTerm

Parameters:

— <dbOptionItem> : ecuConf:DbOptionItemTerm [1..1]

This element represents the **DbOptionItem** whose unit should be returned.

Exceptions:

— None

15.5.6.16 GetDbOptionItemWriteAudienceState

Semantic:

Returns the write audience state of this option item as defined in the ODX data as a string.

NOTE The appropriate MVCI system operation is MCDDbOptionItem.getWriteAudienceState.

Return type:

otx:StringTerm

Parameters:

— <dbOptionItem> : ecuConf:DbOptionItemTerm [1..1]

This element represents the **DbOptionItem** whose write audience state should be returned.

Exceptions:

— None

15.5.6.17 IsComplexDbOptionItem

Semantic:

Returns true if the datatype of the option item is complex.

NOTE The appropriate MVCI system operation is MCDDbOptionItem.isComplex.

Return type:

otx:BooleanTerm

Parameters:

— <dbOptionItem> : ecuConf:DbOptionItemTerm [1..1]

This element represents the **DbOptionItem** whose data type should be checked.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

16 OTX EventPlus extension

16.1 General

The **OTX EventPlus extension** extends the Event extension specified in ISO 13209-3 to add support for deep change monitoring of complex types such as List, Map, or ByteField.

This solves the limitation in the MonitorChangeEventSource term where this behaviour is explicitly excluded.

In addition to changes of variable values, the deep change monitoring shall detect any change of items in structures in any depth. Any changes to structures' elements that support the **IsDeepMonitorChangeEvent** shall trigger as defined for this data type.

For example: if a **DeepMonitorChangeEventSource** is registered on a map which contains lists as values, and an element of one list is added, removed, or changed, the event source shall fire an event.

Other complex data types can support the **DeepMonitorChangeEventSource**, if the behaviour is described in the specification.

16.2 Exceptions

16.2.1 Exceptions overview

All exceptions introduced by this extension are derived from the **otx:Exception** type. They represent the full set of exceptions added by the **OTX EventPlus extension**.

16.2.2 Exceptions syntax

The syntax of all **OTX EventPlus extension** exception-type declarations is shown in Figure 71.

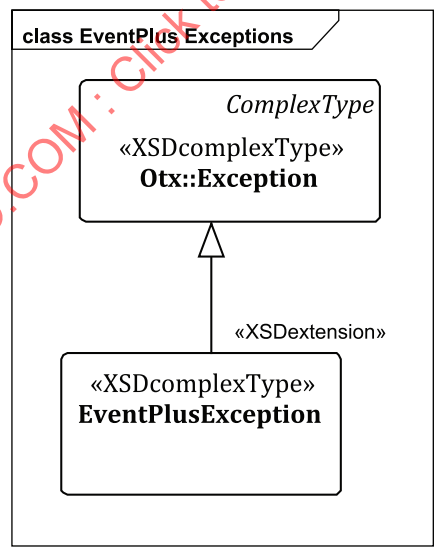


Figure 71 — EventPlus exceptions

16.2.3 Exceptions semantics

16.2.3.1 General

Since all **OTX EventPlus extension** exception types are implicit exceptions without initialisation parts, they cannot be declared constant.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

16.2.3.2 EventPlusException

IMPORTANT — All terms and action realisations in this extension may potentially throw this exception.

16.3 Terms

16.3.1 Terms overview

All terms introduced by this extension extend the **otx:Term** extension interface. Information about the specific super class of a term is provided in the individual term description clauses.

16.3.2 Terms syntax

Figure 72 shows the syntax of the **OTX EventPlus extension** related terms.

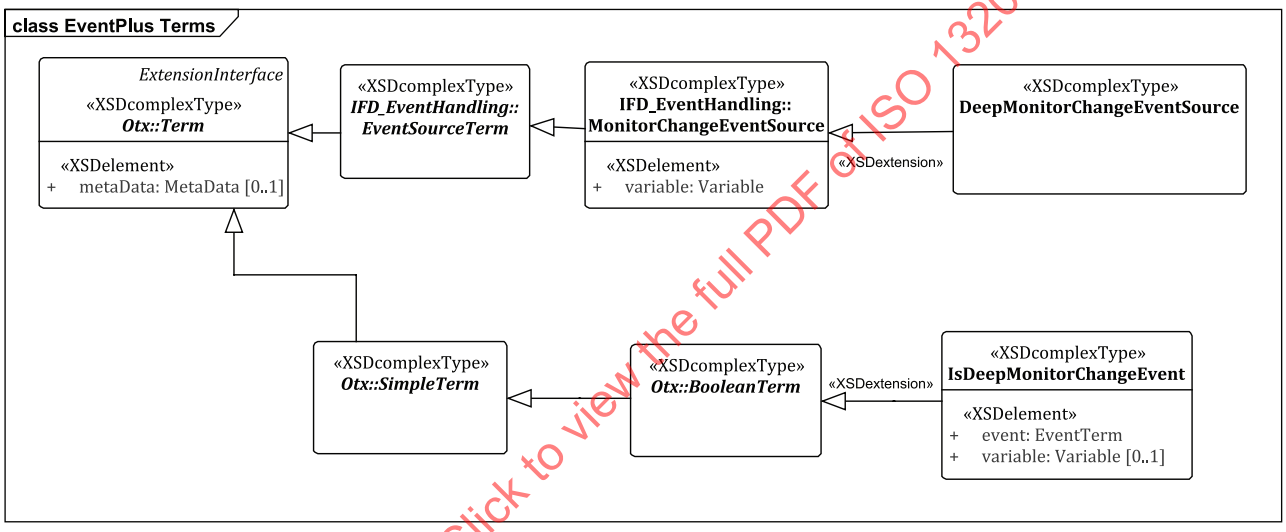


Figure 72 — EventPlus terms

16.3.3 Terms general semantics

16.3.3.1 DeepMonitorChangeEventSource

Semantic:

The behaviour is similar to **event:MonitorChangeEventSource,** but with additional deep change-monitoring of variables of complex datatypes like List, Map and ByteField. The deep change-monitoring shall detect changing, adding and removing of items in Lists, Maps and ByteFields in any depth. For variables of simple datatypes, this term shall behave like **event:MonitorChangeEventSource.**

Because **DeepMonitorChangeEventSource** is derived from **event:MonitorChangeEventSource,** the term **event:IsMonitorChangeEvent** shall return true when the Event originates from a **DeepMonitorChangeEventSource**.

Associated checker rules:

— event:Event_Chk002 – no Path in MonitorChange related terms.

Return type:

event:MonitorChangeEventSource

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <variable> : otx:Variable [1..1] (derived from MonitorChangeEventSource)

Represents the variable that shall be monitored. If the variable value changes or (for complex values) the contents change, the event shall be fired, causing a blocking **WaitForEvent** action to exit.

Exceptions:

— None

16.3.3.2 IsDeepMonitorChangeEvent

Semantic:

The **IsDeepMonitorChangeEvent** term accepts an event:EventTerm yielding an event:Event object that has been raised by the OTX runtime system, as a result of using a **DeepMonitorChangeEventSource** in a WaitForEvent action. The term shall return true if and only if the Event originates from such a kind of event source. In case an optional otx:Variable is specified, the term shall return true if and only if the event:Event was fired because that particular otx:Variable changed. If the given otx:Variable was not the reason for the event, false shall be returned.

Return type:

otx:BooleanTerm

Parameters:

— <event> : event:EventTerm [1..1]

Represents the Event whose type shall be tested.

— <variable> : otx:Variable [0..1]

Optionally specifies the variable which shall be tested for being the cause of the event.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

17 OTX ExternalServiceProvider extension

17.1 General

The **OTX ExternalServiceProvider extension** provides actions, terms, events and data types for accessing external services. These services can be a part of a system, a device, a database or a simple library. The purpose of the **OTX ExternalServiceProvider extension** is to extend OTX to functionality that is encapsulated in these services to help fulfill the requirements of vehicle manufacturing and diagnosis. The existing OTX signature concept gives the author a very flexible construction kit to add existing functionality into OTX test sequences without any knowledge of the complex schema structure of OTX. To extend OTX for general purpose functionality a new OTX extension will be introduced. The criteria for new extensions are:

— as general as possible, as specific as necessary,

— each new extension has to be able to be potentially standardized.

© ISO 2021 – All rights reserved

Use cases for the **OTX ExternalServiceProvider extension** are the following.

— Add the seed/key functionality of a third party company that is encapsulated in a library and has to be added into an OTX test sequence.

— Add an existing proprietary test sequence of an existing test system into an OTX test sequence. In this use case new OTX test sequences can be easily added to an existing test sequence. A step by step migration of the test logic to OTX is possible as well as the possibility to add new functionality written in OTX into proprietary test sequences. So the introduction of OTX test sequences is not complex, the author has the possibility to add small and simple pieces of OTX code into already running code. With this knowledge the author is able to write larger and more complex OTX programs.

— Add any external functionality to an OTX test sequence.

Keep in mind that usage of the ExternalServiceProvider Extension guarantees the exchangeability of the OTX source code itself, but not the exchangeability of the functionality. If you want to add, for example, a proprietary library into OTX, with the help of the ExternalServiceProvider, the binding between the OTX runtime system and the library has to be separately implemented and is not part of the standard.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

17.2 Datatypes

17.2.1 Datatypes overview

All data types introduced by this extension are derived from **otx:ComplexType** or **otx:SimpleType**.

17.2.2 Datatypes syntax

The syntax of the data-type declarations of the **OTX ExternalServiceProvider extension** is shown in Figure 73.

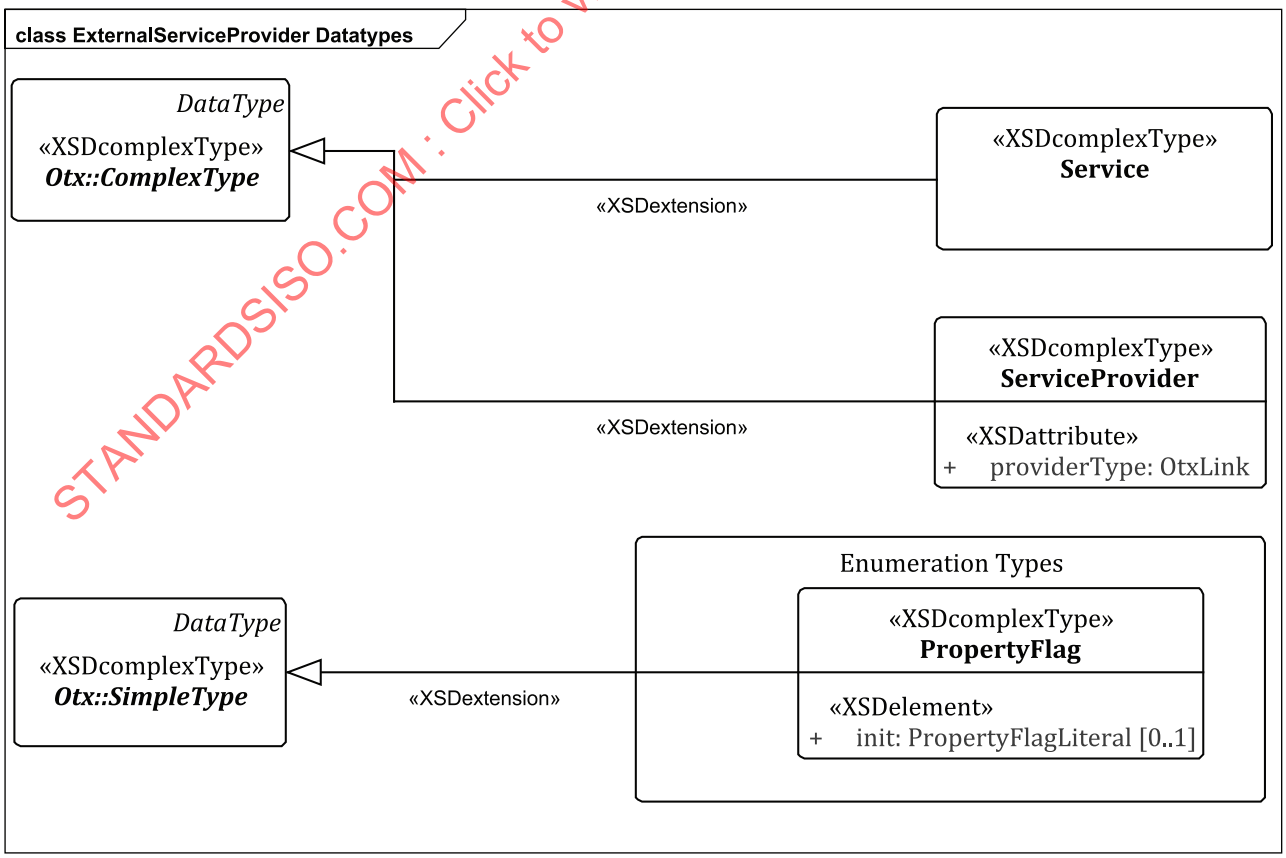


Figure 73 — ExternalServiceProvider datatypes

© ISO 2021 – All rights reserved

17.2.3 Datatypes semantics

17.2.3.1 PropertyFlag

PropertyFlag is an **otx:SimpleType**.

PropertyFlag is an enumeration type describing access rights used in the ServiceProviderSignatures **Properties**.

IMPORTANT — PropertyFlag values may occur as operands of comparisons (see ISO 13209-2:— , 7.15.8). For this case, the following order relation shall apply: READ-ONLY < WRITE-ONLY < READ-WRITE.

IMPORTANT — When applying otx:ToString on a PropertyFlag value, the resulting string shall be the name of the enumeration value, e.g. otx:ToString(READ-WRITE)="READ-WRITE". Furthermore, applying otx:ToInteger shall return the index of the value in the PropertyFlags enumeration (smallest index is 0). The behaviour is undefined for other conversion terms (see ISO 13209-2).

IMPORTANT — If the PropertyFlag declaration is not explicitly initialised (omitted <init> element), the default value shall be READ-ONLY.

Its members have the following semantics:

— <init> : esp:PropertyFlagLiteral[0..1]

This optional element stands for the hard-coded initialisation value of the identifier at declaration time.

17.2.3.2 Service

Service is an **otx:ComplexType**.

The **OTX ExternalServiceProvider extension** introduces a data type named **Service**. The new **Service** type is a handle to a running service. If the **ExecuteService** action is called, the service handle is given back in a variable. When the **ExecuteService** is used with executeAsync attribute set to true, the only way to be notified that the service has finished is to use the **ServiceExecutionFinishedEventSource** term.

17.2.3.3 ServiceProvider

ServiceProvider is an **otx:ComplexType**.

The **OTX ExternalServiceProvider extension** introduces a data type named **ServiceProvider**. The **ServiceProvider** type is a container for all available constructors, services, properties and events that are defined in the corresponding **ServiceProviderSignature**.

Associated checker rules:

— ExternalServiceProvider_Chk001 – correct target for service provider.

Its members have the following semantics:

— <providerType> : otx:OtxLink[1..1]

The link shall point to the corresponding **ServiceProviderSignature**.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

17.3 Enumerations

17.3.1 Enumerations syntax

The syntax of the enumeration declarations of the **OTX ExternalServiceProvider extension** is shown in Figure 74.

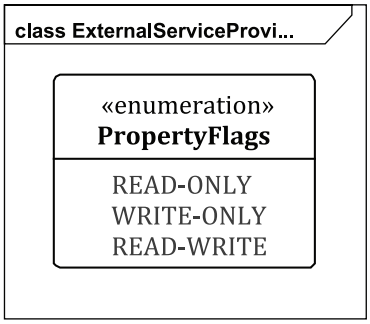


Figure 74 — ExternalServiceProvider enumerations

17.3.2 Enumerations semantics

17.3.2.1 PropertyFlags

The property value, defined in **PropertyDeclaration**, is read-only. This is the default access right if the <init> of the **PropertyFlag** variable is not defined.

PropertyFlags has the following enumeration values:

— READ-ONLY

The property value, defined in **PropertyDeclaration**, is read only. This is the default access right if the <init> of the **PropertyFlag** variable is not defined.

— WRITE-ONLY

The property value, defined in **PropertyDeclaration**, is only writable.

— READ-WRITE

The property value, defined in **PropertyDeclaration**, is readable as well as writable.

17.4 Exceptions

17.4.1 Exceptions overview

All exceptions introduced by this extension are derived from the **otx:Exception** type. They represent the full set of exceptions added by the **OTX ExternalServiceProvider extension**.

17.4.2 Exceptions syntax

The syntax of all **OTX ExternalServiceProvider extension** exception-type declarations is shown in Figure 75.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

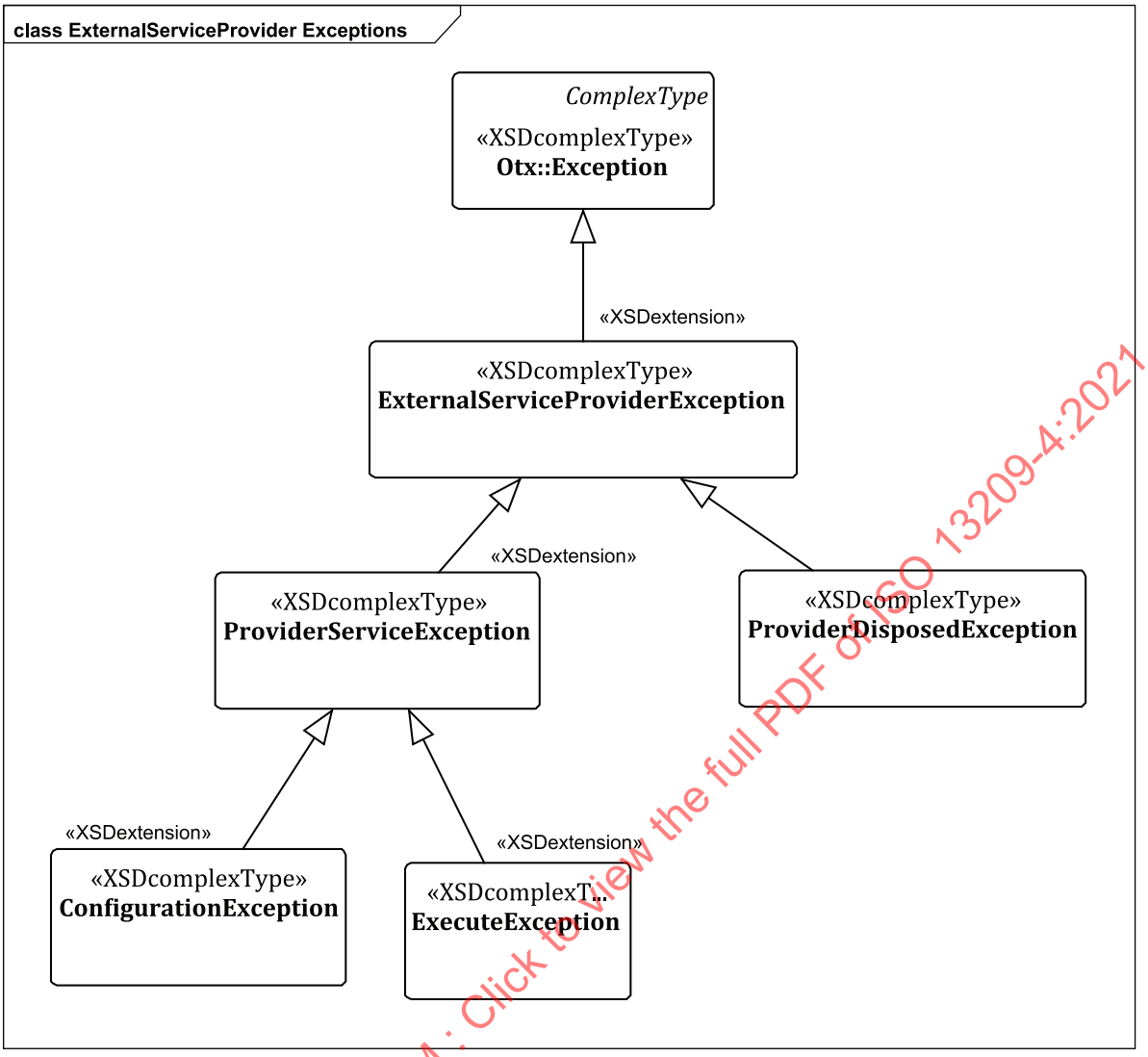


Figure 75 — ExternalServiceProvider exceptions

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

17.4.3 Exceptions semantics

17.4.3.1 General

Since all **OTX ExternalServiceProvider extension** exception types are implicit exceptions without initialisation parts, they cannot be declared constant.

17.4.3.2 ConfigurationException

This exception is thrown if the execution preconditions of a service are missing.

17.4.3.3 ExecuteException

This exception is thrown if a synchronous service call aborts abnormally.

17.4.3.4 ExternalServiceProviderException

The **ExternalServiceProviderException** is the base exception for all provider exceptions. This exception is also thrown if the **CreateProvider** action aborts with an error.

© ISO 2021 – All rights reserved

An **ExternalServiceProviderException** shall be used in case the more specific exception types described in the remainder of this section do not apply to the problem at hand.

IMPORTANT — All terms and action realisations in this extension may potentially throw this exception.

17.4.3.5 ProviderDisposedException

This exception is thrown if the provider is already disposed. The reason can be a previous **DisposeProvider** action call was executed, or the runtime system has already disposed the provider instance.

17.4.3.6 ProviderServiceException

The **ProviderServiceException** is the base exception for all service exceptions of the provider.

17.5 Variable access

17.5.1 Variable-access overview

All variable-access types introduced by this extension are derived from the OTX core variable extension interface.

17.5.2 Variable-access syntax

The syntax of the **OTX ExternalServiceProvider extension** variable-access types is shown in Figure 76.

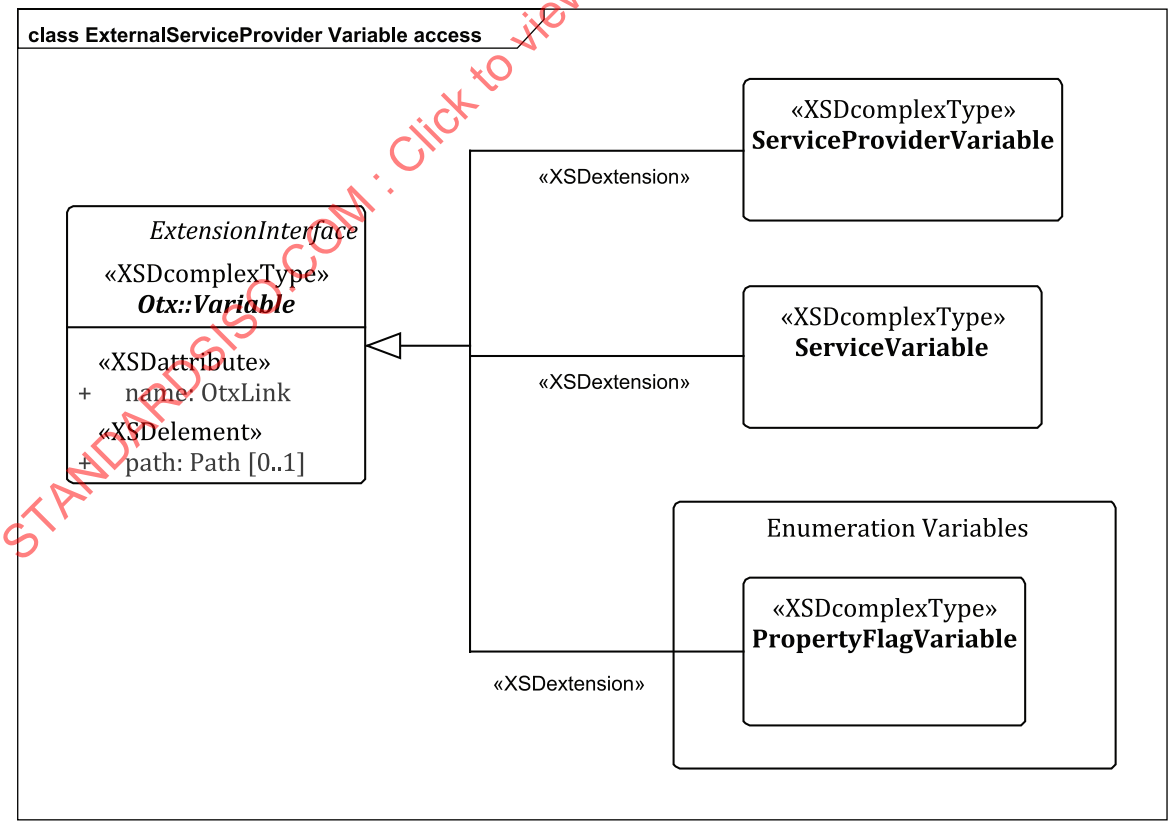


Figure 76 — ExternalServiceProvider variable access

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

17.5.3 Variable-access semantics

The general semantics for all variable-access types shall apply. Please refer to ISO 13209-2 for details.

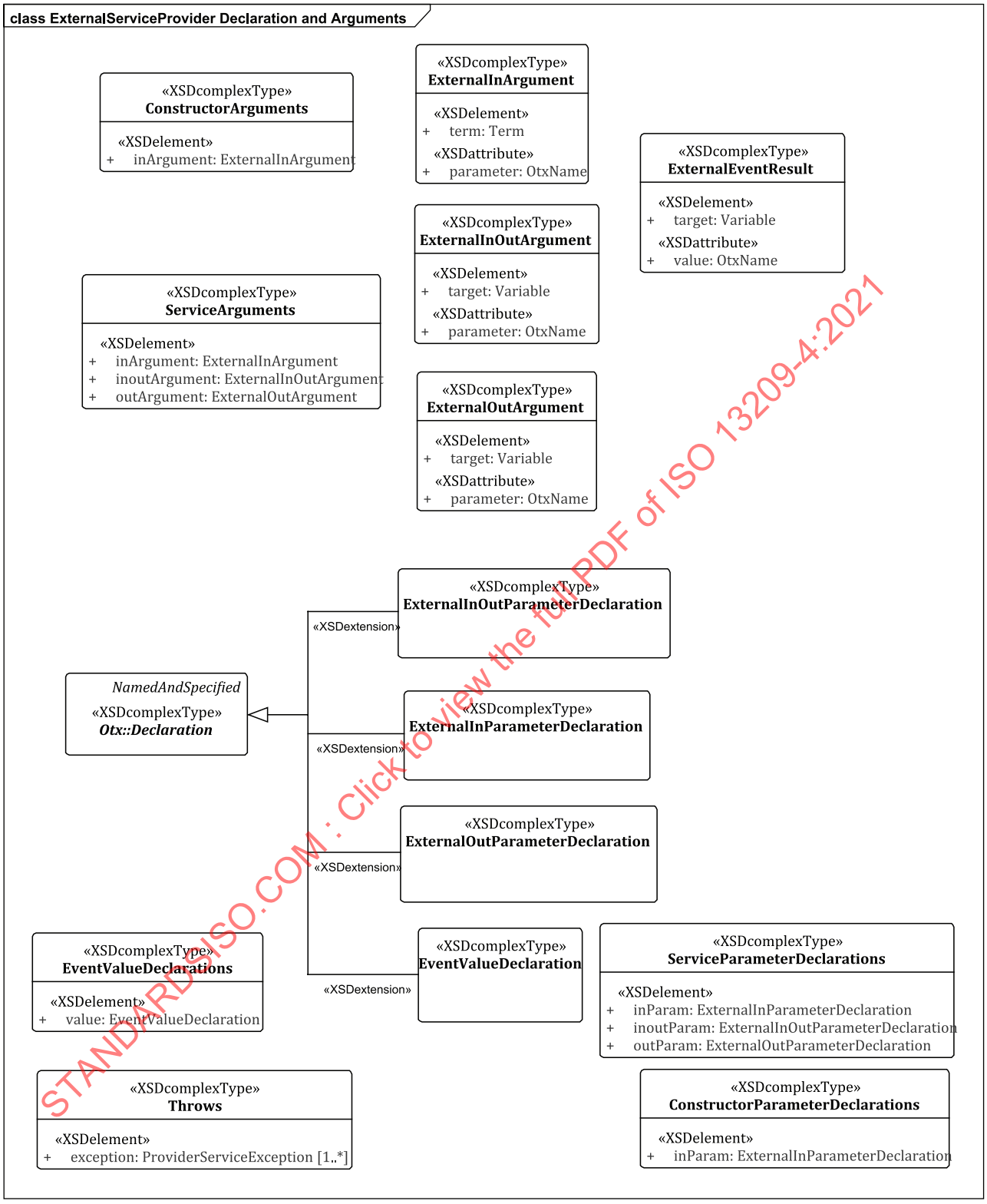
17.6 Declaration and arguments

17.6.1 Declaration and arguments syntax

Figure 77 shows the syntax of the **OTX ExternalServiceProvider extension** declarations and arguments.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 77 — ExternalServiceProvider declaration and arguments

17.6.2 Declaration and arguments semantics

The semantic of the declarations and arguments is shown at the actions and terms, where these are used.

© ISO 2021 – All rights reserved

17.7 Signatures

17.7.1 Signatures overview

All signatures introduced by this extension extend the **otx:signature** or replace the **otx:signature** realisation.

The **OTX ExternalServiceProvider extension** uses this extensibility by adding several signatures:

— **ServiceProviderSignature** – this signature contains all available constructors, properties, services and events of a specific provider,

— **ConstructorSignature** – this signature describes the constructors of a provider,

— **EventSignature** – this signature describes the events of a provider,

— **ServiceSignature** – this signature describes the services of a provider.

17.7.2 Signatures syntax

Figure 78 shows the syntax of the **OTX ExternalServiceProvider extension** related signatures.

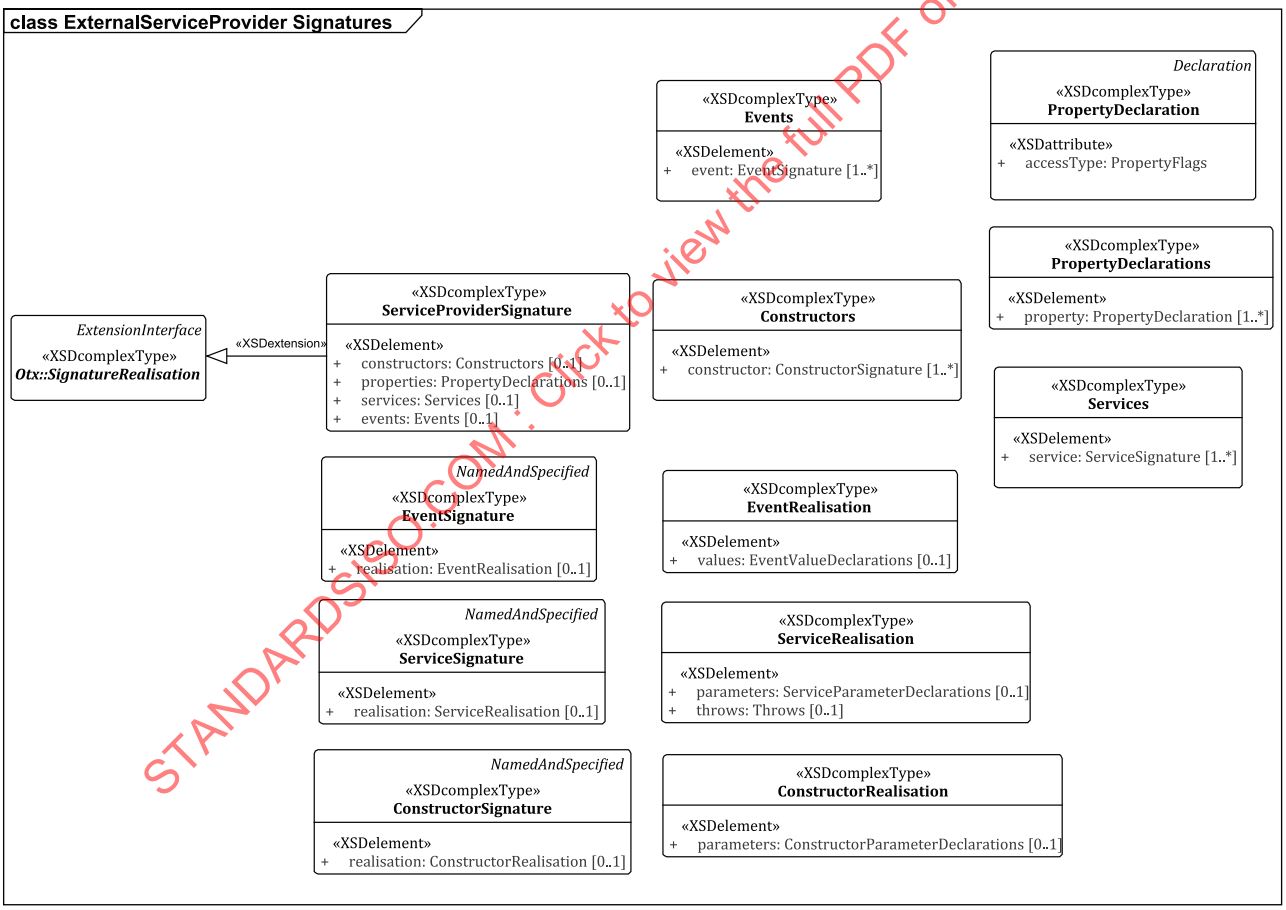


Figure 78 — ExternalServiceProvider signatures

17.7.3 Signatures general semantics

17.7.3.1 ConstructorSignature

Semantic:

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

For the creation of an **OTX ExternalServiceProvider extension** a constructor might be needed. All supported constructors can be defined here. For a constructor only IN parameters are supported. A constructor can have one or more parameters. If no constructor is defined a standard creation sequence of the **OTX ExternalServiceProvider extension** is used. The name attribute is mandatory (unlike the base type NamedAndSpecified where it is optional).

ConstructorSignature is an **otx:SignatureRealisation**.

Associated checker rules:

— ExternalServiceProvider_Chk003 – input argument omission.

Parameters:

— id : otx:OtxId [1..1] (derived from NamedAndSpecified)

This value represents the OtxId of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— name : otx:OtxName [0..1] (derived from NamedAndSpecified)

This value represents the OtxName of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— <realisation> : esp:ConstructorRealisation [0..1]

This is the formal counterpart of the <specification>. It contains a list of parameter descriptions adhering to a constructor.

— <parameters> : esp:ConstructorParameterDeclarations [0..1]

This simple container element represents the list of arguments for a constructor call. The content-type of **ConstructorParameterDeclarations** is <xsd:choice> [1..*] which allows an arbitrary-length list of input parameters of a constructor.

<xsd:key name="UniqueConstructorParameterNames">
<xsd:selector xpath="esp:*" />
<xsd:field xpath="@name" />
</xsd:key>

— <inParam> : esp:ExternalInParameterDeclaration [1..1]

Describes an input parameter for a constructor.

ExternalInParameterDeclaration is based on type **otx:Declaration.** Therefore, an <inParam> element has a name, a specification and a data-type declaration (please refer to ISO 13209-2 for details about declarations).

id : otx:OtxId [1..1] (derived from NamedAndSpecified)

This value represents the OtxId of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

name : otx:OtxName [0..1] (derived from NamedAndSpecified)

This value represents the OtxName of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

<specification> : xs:string [0..1] (derived from NamedAndSpecified)

This value represents the string of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— <specification> : xs:string [0..1] (derived from NamedAndSpecified)

This value represents the string of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

17.7.3.2 EventSignature

Semantic:

An **EventSignature** defines an event that can be fired by a **ServiceProvider**. The event is defined by its name and a set of values that are associated with the event. The name attribute is mandatory (unlike the base type NamedAndSpecified where it is optional).

EventSignature is an **otx:SignatureRealisation**.

Associated checker rules:

— ExternalServiceProvider_Chk003 – input argument omission.

Parameters:

— id : otx:OtxId [1..1] (derived from NamedAndSpecified)

This value represents the OtxId of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— name : otx:OtxName [0..1] (derived from NamedAndSpecified)

This value represents the OtxName of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— <realisation> : esp:EventRealisation [0..1]

This is the formal counterpart of the <specification>. It contains a list of values describing this event.

— <values> : esp:EventValueDeclarations [0..1]

This simple container element represents the list of values for an event. The content-type of **EventValueDeclarations** is <xsd:sequence> [1..*] which allows an arbitrary-length list of values of an event.

```
<xsd:key name="UniqueEventValueNames">
<xsd:selector xpath="esp:value" />
<xsd:field xpath="@name" />
</xsd:key>
```

— <value> : esp:EventValueDeclaration [1..1]

Here the event value is defined.

id : otx:OtxId [1..1] (derived from NamedAndSpecified)

This value represents the OtxId of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

name : otx:OtxName [0..1] (derived from NamedAndSpecified)

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

This value represents the OtxName of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

<specification> : xs:string [0..1] (derived from NamedAndSpecified)

This value represents the string of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— <specification> : xs:string [0..1] (derived from NamedAndSpecified)

This value represents the string of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

17.7.3.3 PropertyDeclaration

Semantic:

An **OTX ExternalServiceProvider extension** might have properties to read and write. The name attribute is mandatory (unlike the base type NamedAndSpecified where it is optional).

Parameters:

— accessType : esp:PropertyFlags [0..1]

This optional attribute specifies whether the property can be read or written. The default value is READ-ONLY.

— id : otx:OtxId [1..1] (derived from NamedAndSpecified)

This value represents the OtxId of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— name : otx:OtxName [0..1] (derived from NamedAndSpecified)

This value represents the OtxName of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— <specification> : xs:string [0..1] (derived from NamedAndSpecified)

This value represents the string of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

17.7.3.4 PropertyDeclarations

Semantic:

This container element holds a collection of properties available for a provider.

Parameters:

— <property> : esp:PropertyDeclaration [1..*]

Contains the property declaration with the appropriate name and data type.

17.7.3.5 ServiceProviderSignature

Semantic:

An **OTX ExternalServiceProvider extension** can be a part of a system, a device, a database or a simple library. The **ServiceProviderSignature** describes the interface to an external system in terms of its services, events, properties and constructors. It is also possible to start and stop services of the provider.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

ServiceProviderSignature is an **otx:SignatureRealisation**.

Parameters:

— <constructors> : esp:Constructors [0..1]

This container element holds a collection of constructor signatures describing the set of constructors available for a provider.

<xsd:key name="UniqueConstructorNames">
<xsd:selector xpath="esp:constructor" />
<xsd:field xpath="@name" />
</xsd:key>

— <constructor> : esp:ConstructorSignature [1..*]

This element describes constructor(s) that are needed to instantiate the specific **OTX External-ServiceProvider extension**.

— <properties> : esp: PropertyDeclarations [0..1]

This container element holds a collection of property signatures describing the set of properties available for a provider.

<xsd:key name="UniquePropertyNames">
<xsd:selector xpath="esp:property" />
<xsd:field xpath="@name" />
</xsd:key>

— <services> : esp:Services [0..1]

This container element holds a collection of service signatures describing the set of services available for a provider.

<xsd:key name="UniqueServiceNames">
<xsd:selector xpath="esp:service" />
<xsd:field xpath="@name" />

— <service> : esp:ServiceSignature [1..*]

This element describes the services or methods of the **OTX ExternalServiceProvider extension.**

— <events> : esp:Events [0..1]

This container element holds a collection of event signatures describing the set of events available from a provider.

<xsd:key name="UniqueEventNames">
<xsd:selector xpath="esp:event" />
<xsd:field xpath="@name" />
</xsd:key>

— <event> : esp:EventSignature [1..*]

This element describes event(s) that can be fired by the **OTX ExternalServiceProvider extension**.

17.7.3.6 ServiceSignature

Semantic:

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

The available services of the **ServiceProvider** are defined here. The name attribute is mandatory (unlike the base type NamedAndSpecified where it is optional).

ServiceSignature is an **otx:SignatureRealisation**.

Associated checker rules:

— ExternalServiceProvider_Chk003 – input argument omission.

Parameters:

— id : otx:OtxId [1..1] (derived from NamedAndSpecified)

This value represents the OtxId of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— name : otx:OtxName [0..1] (derived from NamedAndSpecified)

This value represents the OtxName of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— <realisation> : esp:ServiceRealisation [0..1]

This is the formal counterpart of the <specification>. It contains a list of parameter descriptions adhering to a service.

— <parameters> : esp:ServiceParameterDeclarations [0..1]

This container element represents the list of arguments for a service call. The content-type of **ServiceParameterDeclarations** is <xsd:choice> [1..*] which allows an arbitrary-length list of in-, out- and inout-put parameters of a service.

```
<xsd:key name="UniqueServiceParameterNames">
<xsd:selector xpath="esp:*" />
<xsd:field xpath="@name" />
</xsd:key>
```

— <inParam> : esp:ExternalInParameterDeclaration [1..1]

This parameter describes an input parameter for a service. InParameterDeclaration is based on type **otx:Declaration.** Therefore, an <inParam> element has a name; a specification and a data-type declaration (please refer to ISO 13209-2 for details about declarations). Input parameters shall be treated as constants – the value is not allowed to change throughout service execution.

id : otx:OtxId [1..1] (derived from NamedAndSpecified)

This value represents the OtxId of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

name : otx:OtxName [0..1] (derived from NamedAndSpecified)

This value represents the OtxName of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

<specification> : xs:string [0..1] (derived from NamedAndSpecified)

This value represents the string of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— <inoutParam> : esp:ExternalInOutParameterDeclaration [1..1]

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

This parameter describes an inout-put parameter for a service. InOutParameterDeclaration is based on type **otx:Declaration.** Therefore, an <inoutParam> element has a name; a specification and a data-type declaration (please refer to ISO 13209-2 for details about declarations). Arguments for inout-put parameters shall be passed by reference. This means that any change to the parameter value from the external service is visible to the caller also.

id : otx:OtxId [1..1] (derived from NamedAndSpecified)

This value represents the OtxId of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

name : otx:OtxName [0..1] (derived from NamedAndSpecified)

This value represents the OtxName of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

<specification> : xs:string [0..1] (derived from NamedAndSpecified)

This value represents the string of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— <outParam> : esp:ExternalOutParameterDeclaration [1..1]

This parameter describes an output parameter for a service. OutParameterDeclaration is based on type **otx:Declaration.** Therefore, an <outParam> element has a name; a specification and a data-type declaration (please refer to ISO 13209-2 for details about declarations). Output parameters shall be returned to the caller. The caller may omit output parameters freely (e.g. in the case when there is no interest in one of the returned values). If the service never writes a declared output parameter, the output argument variable stays unchanged. If the procedure throws an exception the output parameter stays unchanged.

id : otx:OtxId [1..1] (derived from NamedAndSpecified)

This value represents the OtxId of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

name : otx:OtxName [0..1] (derived from NamedAndSpecified)

This value represents the OtxName of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

<specification> : xs:string [0..1] (derived from NamedAndSpecified)

This value represents the string of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— <throws> : esp:Throws [0..1]

This shall declare an arbitrary-length list of ExternalServiceProviderException types which this service may potentially throw.

— <exception> : esp:ProviderServiceException [1..*]

Describes an exception type which may possibly be thrown by the enclosing service.

— <specification> : xs:string [0..1] (derived from NamedAndSpecified)

This value represents the string of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

17.8 Actions

17.8.1 Actions overview

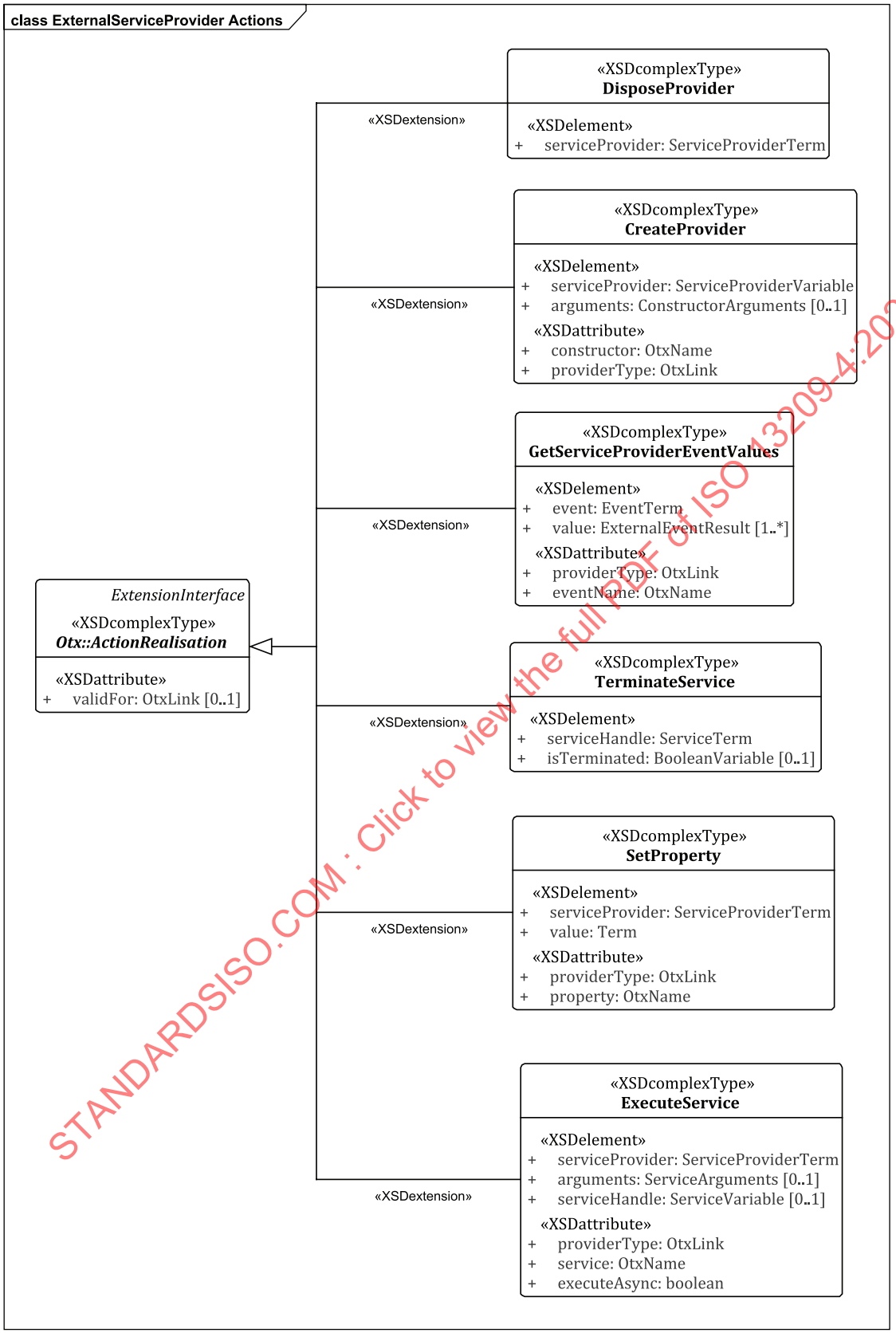
All actions introduced by this extension extend the **otx:ActionRealisation** extension interface.

17.8.2 Actions syntax

Figure 79 shows the syntax of all actions in the **OTX ExternalServiceProvider extension**.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 79 — ExternalServiceProvider actions

© ISO 2021 – All rights reserved

17.8.3 Actions general semantics

17.8.3.1 CreateProvider

Semantic:

CreateProvider is an **otx:ActionRealisation** which returns an instance of the related provider via calling the specified constructor. The actual creation of new instances or use of a singleton pattern depends on the underlying system and service.

In the case of a singleton or a static implementation of the provider **CreateProvider** returns an existing instance. For a static implementation of the provider a **CreateProvider** without a constructor attribute and without any parameters shall be called.

Associated checker rules:

— ExternalServiceProvider_Chk001 – correct target for service provider,

— ExternalServiceProvider_Chk002 – correct CreateProvider constructor arguments,

— ExternalServiceProvider_Chk004 – correct CreateProvider constructor reference,

— ExternalServiceProvider_Chk010 – correct CreateProvider constructor attribute.

Parameters:

— constructor : otx:OtxName [0..1]

This attribute identifies the constructor that shall be called. If this attribute is not given no constructor is called within the runtime environment. This feature is needed for already instantiated classes that need no construction. The link shall point to the corresponding **ConstructorSignature**.

— providerType : otx:OtxLink [1..1]

This attribute identifies the external provider, from which the constructor shall be called. The link shall point to the corresponding **ServiceProviderSignature**.

— <serviceProvider> : esp:ServiceProviderVariable [1..1]

Specifies a **ServiceProvider** declaration to hold the returned instance of **ServiceProvider**.

— <arguments> : esp:ConstructorArguments [0..1]

This simple container element represents the list of arguments for a constructor.

The content-type is a <xsd:choice> [0..1] which allows an arbitrary-length list of input arguments for a provider constructor.

```
<xsd:key name="UniqueConstructorParameterArgumentNames">
<xsd:selector xpath="esp:*" />
<xsd:field xpath="@parameter" />
</xsd:key>
```

— <inArgument> : esp:ExternalInArgument [1..1]

Describes an input parameter for a provider constructor.

— parameter : otx:OtxName [1..1]

This attribute represents the target parameter to which the argument shall be assigned.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— <term> : otx:Term [1..1]

This term represents the value to be used as input argument for the service parameter. The value data type shall match the parameter data type as declared in the corresponding service signature.

Exceptions:

otx:TypeMismatchException

— This exception is thrown if the mappings of the parameters are wrong this exception will be thrown.

17.8.3.2 DisposeProvider

Semantic:

DisposeProvider is an ActionRealisation that informs the runtime system, that the instance of the service provider is no longer needed within OTX. Depending on the underlying implementation of the external service provider, all used resources can be released. After calling this action the appropriate **IsDisposed** term specifying the **ServiceProvider** will return true.

Parameters:

— <serviceProvider> : esp:ServiceProviderTerm [1..1]

This **ServiceProvider** will be disposed.

Exceptions:

— None

17.8.3.3 ExecuteService

Semantic:

The **ExecuteService** action shall execute a service provided by an external service provider. The action executes externally implemented functionality of the provider. The required parameters of the service are defined in the **ServiceSignature**.

Associated checker rules:

— ExternalServiceProvider_Chk001 – correct target for service provider,

— ExternalServiceProvider_Chk005 – correct ExecuteService service reference,

— ExternalServiceProvider_Chk006 – correct ExecuteService arguments.

Parameters:

— providerType : otx:OtxLink [1..1]

This attribute identifies the external provider to execute the service on.

The link shall point to the corresponding **ServiceProviderSignature**.

— service : otx:OtxName [1..1]

This attribute identifies the service that shall be executed. The service name shall be defined within the corresponding service declaration within the **ServiceProviderSignature**.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— executeAsync : boolean [0..1]

This option tells the communication backend to make this service execution non-blocking. This means that if executeAsync is set to true, the OTX execution flow will immediately move on to the next action, without waiting for the result of the **ExecuteService** action. An OTX sequence can make use of the **ServiceExecutionFinishedEventSource** term to be notified when the service has finished.

— <serviceProvider> : esp:ServiceProviderTerm [1..1]

A service of this **ServiceProvider** will be executed.

— <arguments> : esp:ServiceArguments [0..1]

This simple container element represents the list of arguments for a service

The content-type is a <xsd:choice> [0..1] which allows an arbitrary-length list of arguments for a service.

<xsd:key name="UniqueServiceParameterArgumentNames">
<xsd:selector xpath="esp:*" />
<xsd:field xpath="@parameter" />
</xsd:key>

— <inArgument> : esp:ExternalInArgument [1..1]

Describes an input argument for a service call. An input argument may be omitted if and only if there is an explicit initial value defined for the corresponding parameter. This initial value applies in place of the missing argument. The parameter for the argument is identified by name.

— parameter : otx:OtxName [1..1]

This attribute represents the target parameter to which the argument shall be assigned.

— <term> : otx:Term [1..1]

This term represents the value to be used as input argument for the service parameter. The value data type shall match the parameter data type as declared in the corresponding service signature.

— <inoutArgument> : esp:ExternalInOutArgument [1..1]

Describes an inout-put argument for a service call. Arguments for inout-put parameters shall be passed by reference. This means that any change to the parameter value is immediately visible to the caller also. An inout-put argument may be omitted if and only if there is an initial value defined for the parameter. This initial value applies in place of the missing argument. The parameter for the argument is identified by name.

IMPORTANT — In parallel execution (Parallel node), any changes made to an inout-put argument or parameter will be visible in the caller and the callee at the same time; if the callee changes the value of the parameter, the new value will be available on the outside, in the parallel lanes using the argument variable. Vice versa, if the argument variable value is changed in one of the parallel lanes, the change will be visible in the corresponding parameter of the callee.

— parameter : otx:OtxName [1..1]

This attribute represents the output parameter whose value shall be assigned to the target OTX variable.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— <target> : otx:Variable [1..1]

This variable represents the OTX variable to hold the value of the parameter of the service. The variable's data type shall match the parameter data type as declared in the corresponding signature.

— <outArgument> : esp:ExternalOutArgument [1..1]

Describes an output argument for a service call. Output arguments may be omitted freely (e.g. in the case when there is no interest in the returned data). The parameter is identified by name, the argument is a variable.

IMPORTANT — The writing-back of the returned parameter value into the argument variable shall not happen until termination of the called service.

— parameter : otx:OtxName [1..1]

This attribute represents the output parameter whose value shall be assigned to the target OTX variable.

— <target> : otx:Variable [1..1]

This variable represents the OTX variable to hold the value of the parameter of the service. The variable's data type shall match the parameter data type as declared in the **ServiceSignature**.

— <serviceHandle> : esp:ServiceVariable [0..1]

For every started service a serviceHandle is returned. With this handle the running service can be stopped (if possible). It is also possible to check if the service is already running.

Exceptions:

esp:ProviderDisposedException

— This exception will be thrown if the provider is already disposed.

esp:ProviderServiceException

— If the service call aborts abnormally, this exception will be thrown. This exception shall be also thrown if the service is stopped via the **TerminateService** action.

otx:TypeMismatchException

— This exception will be thrown if the mappings of the parameters are wrong.

17.8.3.4 GetServiceProviderEventValues

Semantic:

GetServiceProviderEventValues is an ActionRealisation which retrieves the values of a provider event. To address the correct ServiceProviderEvent the EventName and the ProviderType should be adjusted. If the event inside the event attribute does not match the specified ServiceProviderEvent, a TypeMismatchException shall be thrown.

Associated checker rules:

— ExternalServiceProvider_Chk001 – correct target for service provider,

— ExternalServiceProvider_Chk009 – correct event attribute reference,

— ExternalServiceProvider_Chk011 – correct Event values.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— providerType : otx:OtxLink [1..1]

This attribute identifies the external provider to execute the service on.

The link shall point to the corresponding **ServiceProviderSignature**.

— eventName : otx:OtxName [1..1]

This attribute identifies the **EventSignature** in the given **ServiceProviderSignature** for which event the values shall be returned.

— <event> : event:EventTerm [1..1]

This term represents the value of a ServiceProviderEvent.

— <value> : esp:ExternalEventResult [1..*]

Here the event values of the event can be returned to variables. [1..*] results are possible.

— value : otx:OtxName [1..1]

This attribute identifies the event value. The link shall point to the corresponding **EventSignature** event value.

— <target> : otx:Variable [1..1]

This variable represents the OTX variable to hold the value of the event. The variable's data type shall match the value data type as declared in the corresponding event signature.

Exceptions:

esp:ProviderDisposedException

— This exception will be thrown if the provider is already disposed.

otx:TypeMismatchException

— This exception will be thrown:

— if the mappings of the parameters are wrong, or

— if the given event is not ServiceProviderEvent, or

— if the given event provider type and the event name are not the expected, or

— if the mappings of the event value types are wrong.

17.8.3.5 SetProperty

Semantic:

SetProperty is an ActionRealisation which sets the value of a property.

Associated checker rules:

— ExternalServiceProvider_Chk001 – correct target for service provider,

— ExternalServiceProvider_Chk007 – correct property attribute reference,

— ExternalServiceProvider_Chk008 – correct Property access.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— providerType : otx:OtxLink [1..1]

This attribute identifies the external provider to execute the service on.

The link shall point to the corresponding **ServiceProviderSignature**.

— property : otx:OtxName [1..1]

This attribute identifies the property to set.

— <serviceProvider> : esp:ServiceProviderTerm [1..1]

A property of this **ServiceProvider** will be set.

— <value> : otx:Term [1..1]

This term represents the value that will be stored in the property

Exceptions:

esp:ProviderDisposedException

— This exception will be thrown if the provider is already disposed.

otx:TypeMismatchException

— If the mappings or the access rights of the properties are wrong this exception will be thrown.

17.8.3.6 TerminateService

Semantic:

TerminateService is an ActionRealisation that informs the runtime system that the running service should be terminated if possible. If the service can be terminated, the action is blocking until the service is terminated. The isTerminated flag is set to true. If the service is not interruptible the action returns immediately with the isTerminated flag set to false.

Parameters:

— <serviceHandle> : esp:ServiceTerm [1..1]

This term represents the handle to the service.

— <isTerminated> : otx:BooleanVariable [0..1]

If the service can be terminated the isTerminated returns true. If the service is not interruptible the action returns immediately with the isTerminated flag set to false.

Exceptions:

esp:ProviderDisposedException

— This exception will be thrown if the provider is already disposed.

17.9 Terms

17.9.1 Terms overview

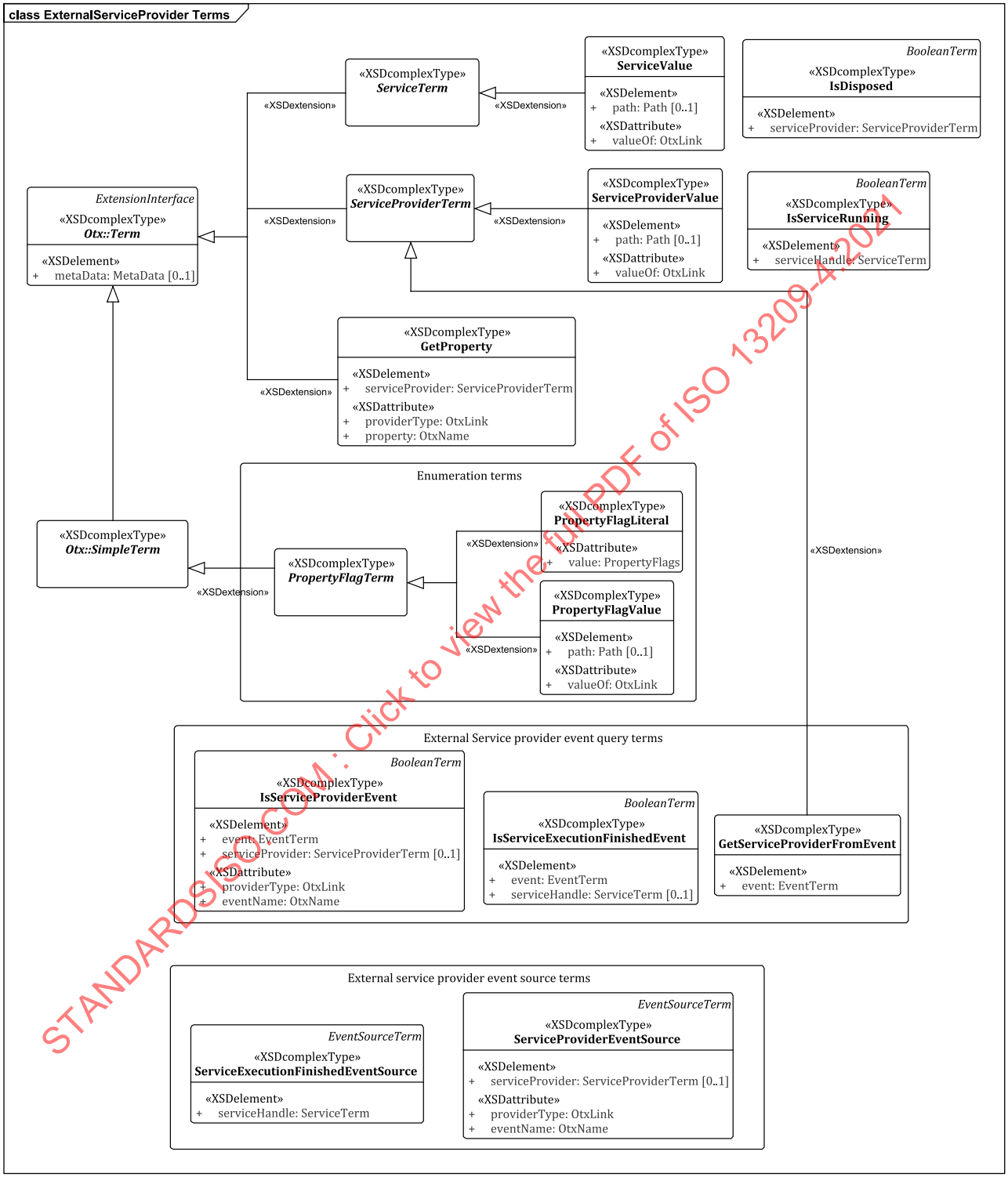
All terms introduced by this extension extend the **otx:Term** extension interface. Information about the specific super class of a term is provided in the individual term description clauses.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

17.9.2 Terms syntax

Figure 80 shows the syntax of the **OTX ExternalServiceProvider extension** related terms.



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 80 — ExternalServiceProvider terms

© ISO 2021 – All rights reserved

17.9.3 Semantics for Enumeration terms

17.9.3.1 PropertyFlagLiteral

Semantic:

Literal of **PropertyFlag** enumeration.

Return type:

esp:PropertyFlagTerm

Parameters:

— value : esp:PropertyFlags [1..1]

This attribute shall contain one of the values defined in the **PropertyFlags** enumeration.

Exceptions:

— None

17.9.3.2 PropertyFlagTerm

Semantic:

A term that returns one element of the **PropertyFlag** enumeration.

17.9.3.3 PropertyFlagValue

Semantic:

A **PropertyFlagTerm** that returns the Value of **PropertyFlag** declaration.

Return type:

esp:PropertyFlagTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures like List or Map. It is built out of a series of index- and name-steps which allow navigation into the structure:

<stepByIndex> : NumericTerm [1]

This step of a <path> shall be used for locations addressed by index. Items in a List and items in a Map with Integer keys shall be addressed in this way.

<stepByName> : StringTerm [1]

This step of a <path> shall be used for addressing locations by name. Items in a Map with String keys shall be addressed in this way.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

17.9.4 Semantics for External Service provider event query terms

17.9.4.1 GetServiceProviderFromEvent

Semantic:

The **GetServiceProviderFromEvent** term is a **ServiceProviderTerm** that returns the **ServiceProvider** that fired the event.

Return type:

esp:ServiceProviderTerm

Parameters:

— <event> : event:EventTerm [1..1]

Represents the Event that holds the **ServiceProvider**.

Exceptions:

otx:TypeMismatchException

— This exception will be thrown if the event is not a ServiceProviderEvent.

17.9.4.2 IsServiceExecutionFinishedEvent

Semantic:

IsServiceExecutionFinishedEvent is a BooleanTerm that tests if the given event indicates the termination of a **Service**. Events that indicate termination of a **Service** are fired by a **ServiceExecutionFinishedEventSource**. **IsServiceExecutionFinishedEvent**s shall be fired at normal termination of service and also abnormal termination by the **TerminateService** action.

Return type:

otx:BooleanTerm

Parameters:

— <event> : event:EventTerm [1..1]

Represents the Event whose type shall be tested.

— <serviceHandle> : esp:ServiceTerm [0..1]

The events of this **Service** instance will be tested.

Exceptions:

— None

17.9.4.3 IsServiceProviderEvent

Semantic:

The **IsServiceProviderEvent** term accepts an EventTerm yielding an event object that has been raised by the OTX runtime. The term shall return true if and only if the event originates from a **ServiceProviderEventSource**. If the eventName attribute is given the term checks if exactly this specified **ServiceProvider** event was fired. Otherwise it returns false.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Associated checker rules:

— ExternalServiceProvider_Chk001 – correct target for service provider,

— ExternalServiceProvider_Chk009 – correct event attribute reference.

Return type:

otx:BooleanTerm

Parameters:

— providerType : otx:OtxLink [1..1]

This attribute identifies the external provider to execute the service on.

The link shall point to the corresponding **ServiceProviderSignature**.

— eventName : otx:OtxName [0..1]

This attribute identifies for which event the values shall be returned.

— <event> : event:EventTerm [1..1]

Represents the Event whose type shall be tested.

— <serviceProvider> : esp:ServiceProviderTerm [0..1]

If specified, the term will return false if the event does not come from the given **ServiceProvider**.

Exceptions:

— None

17.9.5 Semantics for External service provider event source terms

17.9.5.1 ServiceExecutionFinishedEventSource

Semantic:

ServiceExecutionFinishedEventSource is an EventSourceTerm which fires an event if a service execution has finished. This event is also fired if an asynchronous running service is stopped via the **TerminateService** action.

Return type:

event:EventSourceTerm

Parameters:

— <serviceHandle> : esp:ServiceTerm [1..1]

Specifies the service to check for events.

Exceptions:

— None

17.9.5.2 ServiceProviderEventSource

Semantic:

The **ServiceProviderEventSource** is an EventSourceTerm which is an event source for all provider events.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Depending of the usage of the term three different selections of events can be realised.

— Only the mandatory providerType attribute is given. All ServiceProviderEvents of all service provider instances of a given providerType are returned.

— The mandatory providerType attribute and the eventName attribute are given. The ServiceProviderEvents defined by eventName of all service provider instances of a given providerType are returned.

— The mandatory providerType attribute the eventName attribute, and the serviceProvider term are given. The ServiceProviderEvents defined by eventName of a specific service provider instance of a given providerType are returned.

Associated checker rules:

— ExternalServiceProvider_Chk001 – correct target for service provider,

— ExternalServiceProvider_Chk009 – correct event attribute reference.

Return type:

event:EventSourceTerm

Parameters:

— providerType : otx:OtxLink [1..1]

This attribute identifies the external provider. The link shall point to the corresponding **ServiceProviderSignature**.

— eventName : otx:OtxName [0..1]

This attribute identifies the name of the event that is used to select a specific event. The link shall point to the corresponding event in the **EventSignature**.

— <serviceProvider> : esp:ServiceProviderTerm [0..1]

The serviceProvider is a specific Instance of a **ServiceProvider** as a result of a **ServiceProvider** term.

Exceptions:

— None

17.9.6 General semantics

17.9.6.1 GetProperty

Semantic:

GetProperty is a term which gets the value of a property from a service provider.

Associated checker rules:

— ExternalServiceProvider_Chk001 – correct target for service provider,

— ExternalServiceProvider_Chk007 – correct property attribute reference,

— ExternalServiceProvider_Chk008 – correct property access.

Return type:

otx:Term

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— providerType : otx:OtxLink [1..1]

This attribute identifies the external provider to execute the service on.

The link shall point to the corresponding **ServiceProviderSignature**.

— property : otx:OtxName [1..1]

This attribute identifies the property to get.

— <serviceProvider> : esp:ServiceProviderTerm [1..1]

A property of this **ServiceProvider** will be read.

Exceptions:

esp:ProviderDisposedException

— If the provider is already disposed this exception shall be thrown.

otx:TypeMismatchException

— If the mappings or the access rights of the properties are wrong this exception will be thrown.

17.9.6.2 IsDisposed

Semantic:

IsDisposed is a BooleanTerm that returns true if the given serviceProvider has been disposed, either by the **DisposeProvider** action or by the runtime system.

Return type:

otx:BooleanTerm

Parameters:

— <serviceProvider> : esp:ServiceProviderTerm [1..1]

This **ServiceProvider** will be tested

Exceptions:

— None

17.9.6.3 IsServiceRunning

Semantic:

The **IsServiceRunning** term is an **otx:BooleanTerm,** which checks if a service is still running. This check can be performed while a synchronous or asynchronous service is running. For a synchronous ExcecuteService call this term can be called from a parallel lane.

Return type:

otx:BooleanTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <serviceHandle> : esp:ServiceTerm [1..1]

With serviceHandle of a **CreateProvider** action, a check can be performed to determine if the service is still running. Specifies the service to test.

Exceptions:

— None

17.9.6.4 ServiceProviderTerm

Semantic:

The abstract type **ServiceProviderTerm** is an **otx:Term.** It serves as a base for all concrete terms, which return a **ServiceProvider**. It has no special members.

17.9.6.5 ServiceProviderValue

Semantic:

Returns the **ServiceProvider** stored in a **ServiceProvider** variable. For more information on value-terms and the syntax and semantics of the valueOf attribute and <path> element, please refer to ISO 13209-2.

Associated checker rules:

— Core_Chk053 – no dangling OtxLink associations (see ISO 13209-2).

Return type:

esp:ServiceProviderTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception will be thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

17.9.6.6 ServiceTerm

Semantic:

The abstract type **ServiceTerm** is an **otx:Term.** It serves as a base for all concrete terms which return a **Service**. It has no special members.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

17.9.6.7 ServiceValue

Semantic:

Returns the **Service** stored in a **Service** variable. For more information on value-terms and the syntax and semantics of the valueOf attribute and <path> element, please refer to ISO 13209-2.

Associated checker rules:

— Core_Chk053 – no dangling OtxLink associations (see ISO 13209-2).

Return type:

esp:ServiceTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception will be thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

18 OTX File extension

18.1 General

The file extension allows general read and write access to files. The extension allows files to be read or written in one of three modes:

— reading and writing the contents of a file as bytes,

— reading and writing the contents of a file as lines of text,

— reading and writing the entire file as a string.

The mode of access is determined by the first read or write operation performed on the file after it is opened. Once the mode is determined it cannot be changed and attempts to use other modes on the same open file will result in exceptions.

Actions for deleting files and directories are also included.

It is unspecified whether write actions will immediately influence the file system, or whether data will only be physically written when the **SaveFile** or **CloseFile** actions are called.

This extension is limited to those file system operations that are required for typical vehicle diagnostics use cases. Advanced file system functions like setting file rights, moving files, etc. have been intentionally omitted.

All paths shall be defined by a URI (see ISO 13209-2:—, Annex G).

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

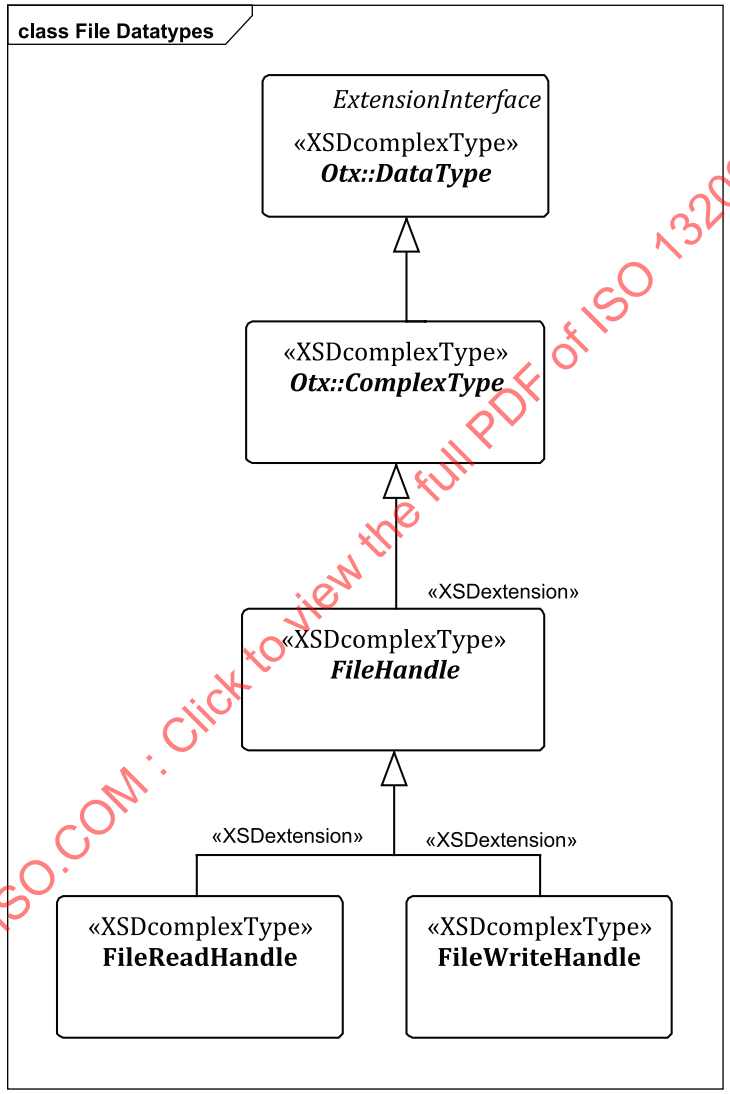
18.2 Datatypes

18.2.1 Datatypes overview

All data types introduced by this extension are derived from **otx:ComplexType**.

18.2.2 Datatypes syntax

The syntax of the data-type declarations of the **OTX File extension** is shown in Figure 81.



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 81 — File datatypes

18.2.3 Datatypes semantics

18.2.3.1 FileHandle

FileHandle is an otx:ComplexType.

It is an abstract handle for a file that has been opened. Please note that for new data types, the ToString term within Core should be expanded.

© ISO 2021 – All rights reserved

18.2.3.2 FileReadHandle

FileReadHandle is a **FileHandle**.

A **FileReadHandle** is a handle for a file that has been opened for read operations.

A ToString() operation on a **FileReadHandle** shall produce the String "FileReadHandle <path>" where <path> represents the file system path in URI (see ISO 13209-2:—, Annex G) format that the handle refers to.

18.2.3.3 FileWriteHandle

FileWriteHandle is a **FileHandle**.

A **FileWriteHandle** is a handle for a file that has been opened for write operations.

A ToString() operation on a **FileWriteHandle** shall produce the String "FileWriteHandle <path>" where <path> represents the file system path in URI (see ISO 13209-2:—, Annex G) format that the handle refers to.

18.3 Exceptions

18.3.1 Exceptions overview

All exceptions introduced by this extension are derived from the **otx:Exception** type. They represent the full set of exceptions added by the **OTX File extension**.

18.3.2 Exceptions syntax

The syntax of all **OTX File extension** exception-type declarations is shown in Figure 82.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

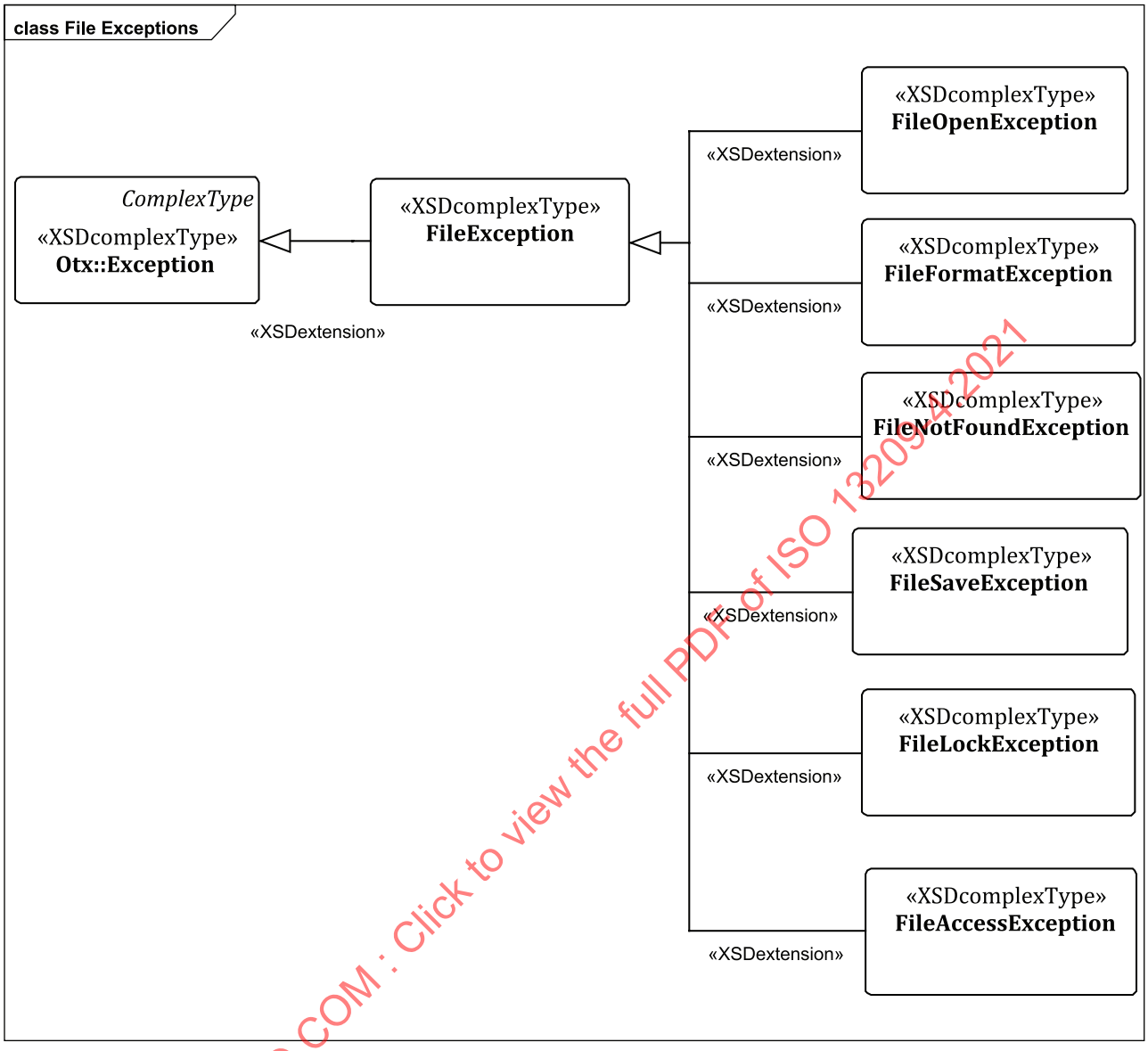


Figure 82 — File exceptions

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

18.3.3 Exceptions semantics

18.3.3.1 General

Since all **OTX File extension** exception types are implicit exceptions without initialisation parts, they cannot be declared constant.

18.3.3.2 FileAccessException

If the file is not accessible this exception is thrown.

18.3.3.3 FileException

The **FileException** is the super class for all exceptions in the **OTX File extension**. A **FileException** is used in case the more specific exception types described in the remainder of this section do not apply to the problem at hand.

© ISO 2021 – All rights reserved

IMPORTANT — All terms and action realisations in this extension may potentially throw this exception.

18.3.3.4 FileFormatException

This exception is thrown when the file format is not equal to the expected format, or in the case when the file was accessed using a different mode (modes are lines, bytes, files).

18.3.3.5 FileLockException

This is the exception that is thrown when the operating system denies access because a file is locked (e.g. for security reasons).

18.3.3.6 FileNotFoundException

This exception indicates that a denoted file has not been found. This might happen if the path to a source file in any of the actions of the **OTX File extension** does not exist.

18.3.3.7 FileOpenException

This exception is thrown if an error occurs, while opening a file, e.g. if the operating system denies access because of an I/O error.

18.3.3.8 FileSaveException

This exception is thrown if an error occurs during file saving, e.g. the file is read-only or the user's permissions for writing to the file are not sufficient.

18.4 Variable access

18.4.1 Variable-access overview

All variable-access types introduced by this extension are derived from the OTX core variable extension interface.

18.4.2 Variable-access syntax

The syntax of the **OTX File extension** variable-access types is shown in Figure 83.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

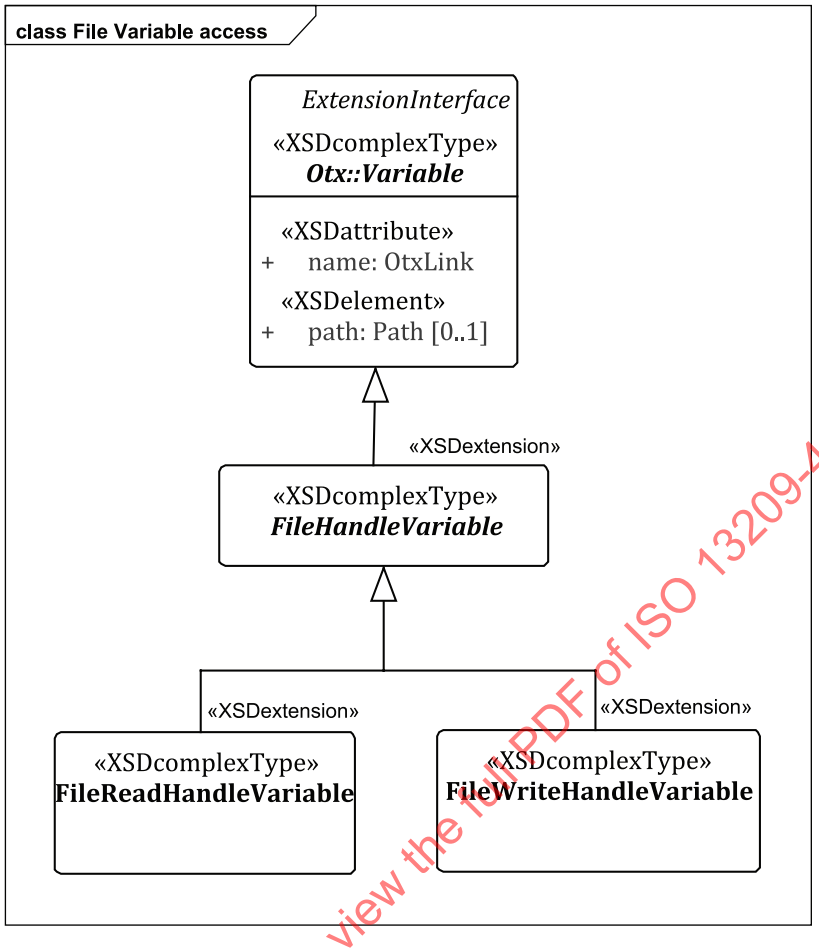


Figure 83 — File variable access

18.4.3 Variable-access semantics

The general semantics for all variable-access types shall apply. Please refer to ISO 13209-2 for details.

18.5 Actions

18.5.1 Actions overview

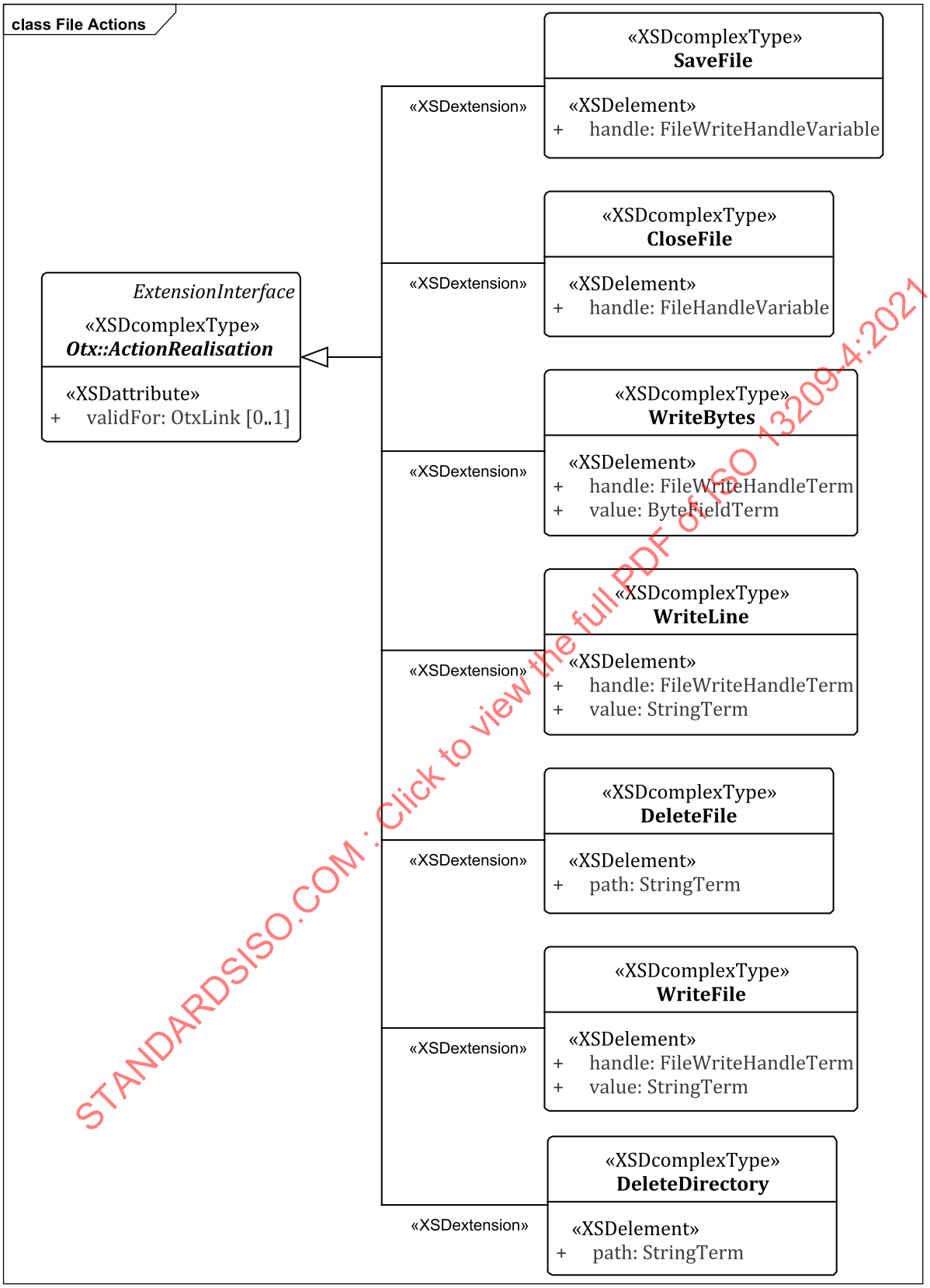
All actions introduced by this extension extend the **otx:ActionRealisation** extension interface.

18.5.2 Actions syntax

Figure 84 shows the syntax of all actions in the **OTX File extension**.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 84 — File actions

© ISO 2021 – All rights reserved

18.5.3 Actions general semantics

18.5.3.1 CloseFile

Semantic:

Closes the current file and releases any resources associated with the current file. In the case of a **FileWriteHandle**, the file will be saved before closing. The use of **CloseFile** is optional. An OTX runtime system shall automatically close any open FileHandles at the end of a diagnostic session, saving the file in case of a **FileWriteHandle**.

If a diagnostic sequence uses a **FileHandle** after it has been closed by a **CloseFile** action, the runtime system shall throw an **otx:InvalidReferenceException.**

Parameters:

— <handle> : file:FileHandleVariable [1..1]

The handle for the file that the related file object encapsulates.

Exceptions:

file:FileSaveException

— This exception will be thrown if saving failed.

otx:InvalidReferenceException

— This exception will be thrown if the variable value is not valid (no value was assigned to the variable before, or the file has already been closed).

18.5.3.2 DeleteDirectory

Semantic:

Deletes a Directory on the specified path. Its contents will be deleted recursively.

Parameters:

— <path> : otx:StringTerm [1..1]

The directory to delete. The path shall be defined by a URI (see ISO 13209-2:—, Annex G).

Exceptions:

file:FileLockException

— This exception will be thrown if a file is locked (e.g. for security reasons).

file:FileNotFoundException

— This exception will be thrown if the path does not point to an existing directory.

18.5.3.3 DeleteFile

Semantic:

Deletes an **OTX File extension** on the specified path.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <path> : otx:StringTerm [1..1]

The file to delete. The path shall be defined by a URI (see ISO 13209-2:—, Annex G).

Exceptions:

file:FileLockException

— This exception will be thrown if the file is locked (e.g. for security reasons).

file:FileNotFoundException

— This exception will be thrown if the path does not point to an existing file.

18.5.3.4 SaveFile

Semantic:

Saves all the content of the related file handle to the file. With **SaveFile** it is guaranteed that all changes are written to the file. After the save operation, further write operations are possible.

Parameters:

— <handle> : file:FileWriteHandleVariable [1..1]

The handle for the file that the related file object encapsulates.

Exceptions:

file:FileSaveException

— This exception will be thrown if saving failed.

otx:InvalidReferenceException

— This exception will be thrown if the variable value is not valid (no value was assigned to the variable before, or the file has already been closed).

18.5.3.5 WriteBytes

Semantic:

Appends the specified bytes to the file handle.

The first read or write activity on a file handle defines if access shall be realised by line, bytes or file. Mixed access will lead to a **FileFormatException**.

Parameters:

— <handle> : file:FileWriteHandleTerm [1..1]

The handle for the file that the related file object encapsulates.

— <value> : otx:ByteFieldTerm [1..1]

The ByteField to write to the file.

Exceptions:

file:FileAccessException

— This exception will be thrown if the file is not accessible.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

file:FileFormatException

— This exception will be thrown if **WriteLine** or **WriteFile** has been called on the file handle before.

file:FileLockException

— This exception will be thrown if a file is locked (e.g. for security reasons).

18.5.3.6 WriteFile

Semantic:

Appends the given string to the text file specified in the file handle.

The first read or write activity on a file handle defines if access shall be realised by line, bytes or file. Mixed access will lead to a **FileFormatException**.

Parameters:

— <handle> : file:FileWriteHandleTerm [1..1]

The handle for the file that the related file object encapsulates.

— <value> : otx:StringTerm [1..1]

The string to write to the file. The string can contain line breaks, which will be written without any operating system specific conversion.

Exceptions:

file:FileAccessException

— This exception will be thrown if the file is not accessible.

file:FileFormatException

— This exception will be thrown if **WriteBytes** or **WriteLine** has been called on the file handle before.

file:FileLockException

— This exception will be thrown if a file is locked (e.g. for security reasons).

18.5.3.7 WriteLine

Semantic:

Appends the specified string value followed by a line break to the file handle. The style of the line break shall be taken from the underlying operating system (Unix: '\n', Windows: '\r\n' or MacOS '\r'). The first read or write activity on a file handle defines if access shall be realised by line, bytes or file. Mixed access will lead to a **FileFormatException**.

Parameters:

— <handle> : file:FileWriteHandleTerm [1..1]

The handle for the file that the related file object encapsulates.

— <value> : otx:StringTerm [1..1]

The string to write to the file. The string can contain line breaks, which will be written without any operating system specific conversion.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

NOTE The OS specific new line will always be appended to the string before write.

Exceptions:

file:FileAccessException

— This exception will be thrown if the file is not accessible.

file:FileFormatException

— This exception will be thrown if **WriteBytes** or **WriteFile** has been called on the file handle before.

file:FileLockException

— This exception will be thrown if a file is locked (e.g. for security reasons).

18.6 Terms

18.6.1 Terms overview

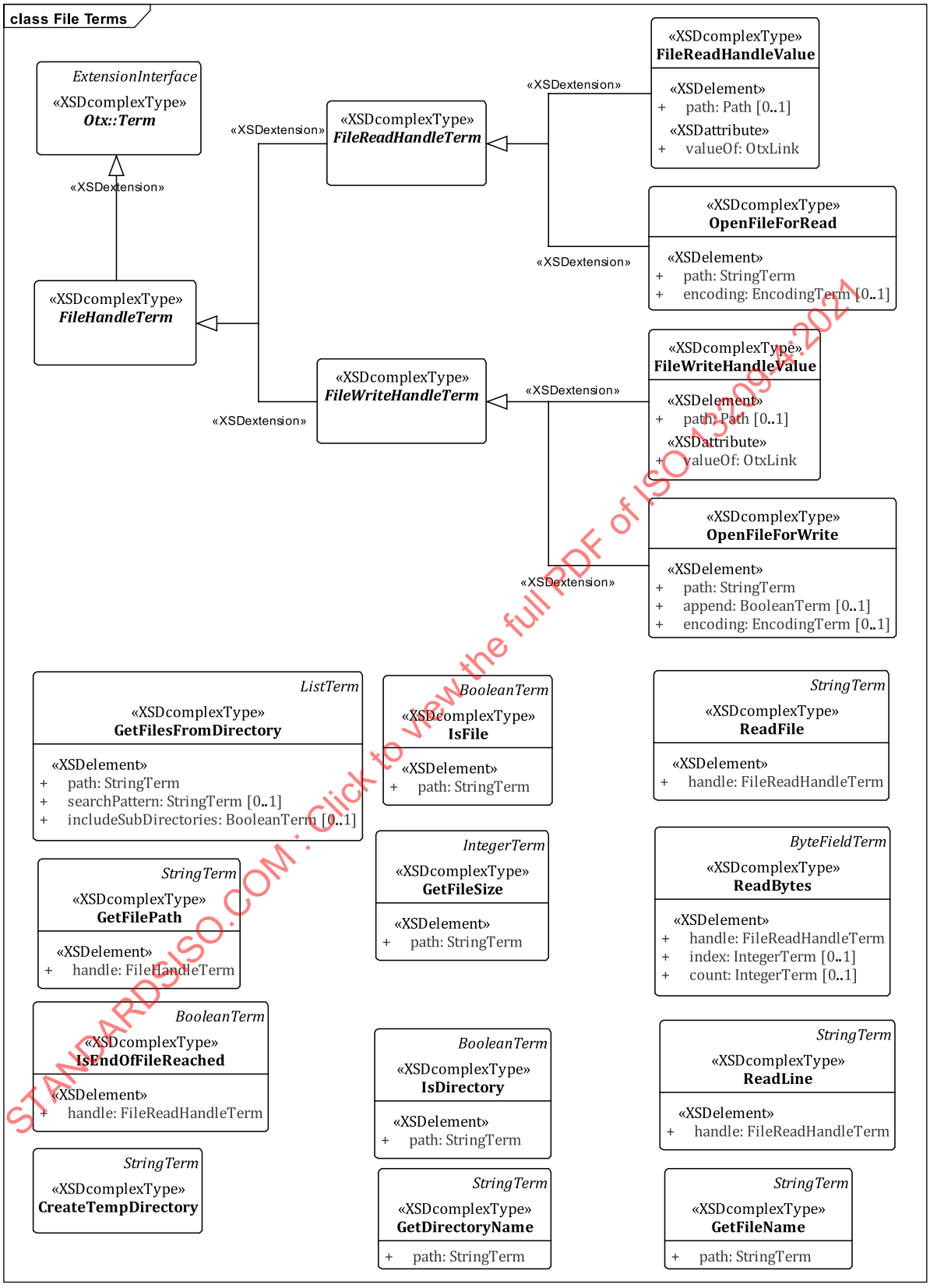
All terms introduced by this extension extend the **otx:Term** extension interface. Information about the specific super class of a term is provided in the individual term description clauses.

18.6.2 Terms syntax

Figure 85 shows the syntax of the **OTX File extension** related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 85 — File terms

© ISO 2021 – All rights reserved

18.6.3 Terms general semantics

18.6.3.1 CreateTempDirectory

Semantic:

Creates a new unique temporary directory and returns the URI of the path (see ISO 13209-2:—, Annex G). The OTX runtime environment has to provide this directory and has to ensure that the user has read and write access to the content of that directory. After finishing the execution of the OTX runtime, the temporary directory and its contents will be deleted automatically.

IMPORTANT — This term has a side effect.

Associated checker rules:

— File_Chk001 – no lazy evaluation on terms with side effects.

Return type:

otx:StringTerm

Parameters:

— None

Exceptions:

file:FileSaveException

— This exception will be thrown if creating a temporary directory failed.

18.6.3.2 FileHandleTerm

Semantic:

A **FileHandleTerm** is a Term that returns a handle to perform operations on a file.

18.6.3.3 FileReadHandleTerm

Semantic:

The handle for the file that the related file object encapsulates.

18.6.3.4 FileReadHandleValue

Semantic:

Returns the **FileReadHandleValue** stored in a **FileReadHandle** variable.

Return type:

file:FileReadHandleTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— This exception will be thrown if the variable value is not valid (no value was assigned to the variable before, or the file has already been closed).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location, which does not exist.

18.6.3.5 FileWriteHandleTerm

Semantic:

A **FileWriteHandleTerm** is a term that returns the handle for the file that the related file object encapsulates. It is unspecified whether write actions immediately influence the file system. **SaveFile** and **CloseFile** guarantee the writing to a file.

18.6.3.6 FileWriteHandleValue

Semantic:

Returns the **FileWriteHandle** stored in a **FileWriteHandle** variable.

Return type:

file:FileWriteHandleTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception will be thrown if the variable value is not valid (no value was assigned to the variable before, or the file has already been closed).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location, which does not exist.

18.6.3.7 GetDirectoryName

Semantic:

Returns the directory name of the given path.

IMPORTANT — The term performs a pure string operation without directory access.

IMPORTANT — The result of the term can reformatted to a certain format of the operation system. For example, at Windows OS the URI path "/C:/MyFolder/MyFile.txt" can return "\C:\ MyFolder". Therefore, it should not be used for string comparisons within the test logic.

Return type:

otx:StringTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— path : otx:StringTerm [1..1]

The file path to get the directory name from. The path shall be defined by a URI (see ISO 13209-2:—, Annex G).

The information in term **GetFileName** gives some samples of well formed and invalid URI.

Exceptions:

— None

18.6.3.8 GetFileName

Semantic:

Returns the file name of the given path including a possible file extension.

IMPORTANT — The term performs a pure string operation without file access.

The following information gives some samples of well formed URI:

StringTerm path	*GetFileName*	*GetDirectoryName*
"file:///C:/MyFolder/MyFile.txt"	"MyFile.txt"	"file:///C:/MyFolder"
"file:///./MyFile.txt"	"MyFile.txt"	"file:///."
"file:///C:/MyFolder/./MyFile.txt"	"MyFile.txt"	"file:///C:/MyFolder/."
"file:///C:/MyFolder/MyFile"	"MyFile"	"file:///C:/MyFolder"
"file:///C:/MyFolder/"	""	"file:///C:/MyFolder"
"file:///C:/MyFolder"	"MyFolder"	"file:///C:"
"file:///C:/"	""	"file:///C:"
"/C:/MyFolder/MyFile.txt"	"MyFile.txt"	"/C:/MyFolder"
"./MyFile.txt"	"MyFile.txt"	"."
"../MyFile.txt"	"MyFile.txt"	".."
"./"	""	"."
"."	"."	""
"http://www.myDomain.com/MyFile.txt"	"MyFile.txt"	"http://www.myDomain.com"
"www.myDomain.com/MyFile.txt"	"MyFile.txt"	"www.myDomain.com"
"www.myDomain.com/"	""	"www.myDomain.com"
"www.myDomain.com"	""	""
""	""	**otx:OutOfBoundsException**

The following information gives some samples of invalid URI, which throws an **otx:OutOfBoundsException:**

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Path	*Error*
"file:///C:/MyFolder/My File.txt"	The string is not correctly escaped
"file:\\\C:\MyFolder\MyFile.txt"	Backslash instead of slash
"file:///C:\MyFolder\MyFile.txt"	Backslash instead of slash
"file://C:/MyFolder/MyFile.txt"	Missing a slash before the path
"C:/MyFolder/MyFile.txt"	Missing leading slash to identify the absolute URI
"file:///C:"	Missing trailing slash

Return type:

otx:StringTerm

Parameters:

— path : otx:StringTerm [1..1]

The file path to get the file name from. The path shall be defined by a URI (see ISO 13209-2:—, Annex G). If the last character of the given path is a slash separator ("/"), then the term returns an empty string.

Exceptions:

— None

18.6.3.9 GetFilePath

Semantic:

Returns the URI of the path of the specified file (see ISO 13209-2:—, Annex G).

Return type:

otx:StringTerm

Parameters:

— <handle> : file:FileHandleTerm [1..1]

The file handle to get the file path from.

Exceptions:

— None

18.6.3.10 GetFilesFromDirectory

Semantic:

Returns a list of strings specifying the paths as URIs of the files in the specified directory that match the specified search pattern (see ISO 13209-2:—, Annex G).

Given the following example file system contents:

file:///usr/user1/example1.txt

file:///usr/user1/example/example.txt

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

file:///usr/user1/example/example1.txt

file:///usr/user1/example/example2.txt

GetFilesFromDirectory("file:///usr/user1", "*.txt", true) will return the paths of all four files.

GetFilesFromDirectory("file:///usr/user1/", "example?.txt", true) will return the paths of example0.txt, example1.txt and example2.txt.

GetFilesFromDirectory("file:///usr/user1", "example/*.txt", true) will return an empty list, as the search pattern may only be used to match file names, not directory names.

Return type:

otx:ListTerm

Parameters:

— <path> : otx:StringTerm [1..1]

The directory to scan. The path shall be defined by a URI (see ISO 13209-2:—, Annex G).

— <searchPattern> : otx:StringTerm [0..1]

The search string to match against file names. It shall only be matched against the file name itself, not against directory names within the path. The following wildcard specifiers are permitted in the searchPattern: "*" = zero or more characters; "?" = exactly one character. Characters other than the wildcard specifiers represent themselves.For example, the searchPattern string "*t" searches for all names in path ending with the letter "t". The searchPattern string "s*" searches for all names in path beginning with the letter "s".

— <includeSubDirectories> : otx:BooleanTerm [0..1]

If the parameter is true, all files within subdirectories in any depth shall also returned. The default is true if not specified.

Exceptions:

file:FileAccessException

— This exception will be thrown if the directory is not accessible.

file:FileNotFoundException

— This exception will be thrown if the path (file or directory) does not point to an existing directory.

18.6.3.11 GetFileSize

Semantic:

Retrieves the size of the specified file in bytes.

Return type:

otx:IntegerTerm

Parameters:

— <path> : otx:StringTerm [1..1]

The file whose size is to be determined. The path shall be defined by a URI (see ISO 13209-2:—, Annex G).

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

Exceptions:

file:FileAccessException

— This exception will be thrown if the file is not accessible.

file:FileLockException

— This exception will be thrown if a file is locked (e.g. for security reasons).

file:FileNotFoundException

— This exception will be thrown if the path (file or directory) does not point to an existing file.

18.6.3.12 IsDirectory

Semantic:

Determines whether the specified path exists and is a directory.

Return type:

otx:BooleanTerm

Parameters:

— <path> : otx:StringTerm [1..1]

The directory to check. The path shall be defined by a URI (see ISO 13209-2:—, Annex G).

Exceptions:

— None

18.6.3.13 IsEndOfFileReached

Semantic:

If the End-Of-File has been reached by the **ReadLine** or **ReadBytes** term or if the file is empty, the term returns true, otherwise false.

Return type:

otx:BooleanTerm

Parameters:

— <handle> : file:FileReadHandleTerm [1..1]

The handle for the file that the related file object encapsulates.

Exceptions:

— None

18.6.3.14 IsFile

Semantic:

Determines whether the specified path exists and is a file.

Return type:

otx:BooleanTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <path> : otx:StringTerm [1..1]

The file to check. The path shall be defined by a URI (see ISO 13209-2:—, Annex G).

Exceptions:

— None

18.6.3.15 OpenFileForRead

Semantic:

Opens an **OTX File extension** on the specified path with read access and returns a file handle to this file. If a file is opened for reading it shall be automatically locked against writing. Simultaneous reading of the same file is possible.

IMPORTANT — This term has a side effect.

Associated checker rules:

— File_Chk001 – no lazy evaluation on terms with side effects.

Return type:

file:FileReadHandleTerm

Parameters:

— <path> : otx:StringTerm [1..1]

The file to open. The path shall be defined by a URI (see ISO 13209-2:—, Annex G).

— <encoding> : string:EncodingTerm [0..1]

This represents the encoding to be used. The set of standard encodings which shall be supported by any runtime system is given by the encoding enumeration in StringUtil extension, see ISO 13209-3. If the encoding is not specified, it assumes that Strings will be encoded using UTF-8. The encodings HEX, BIN, OCT are not allowed.

Exceptions:

file:FileLockException

— This exception will be thrown if the file is locked (e.g. for security reasons).

file:FileNotFoundException

— This exception will be thrown if the path (file or directory) does not point to an existing file.

file:FileOpenException

— This exception will be thrown if the open operation failed, or an encoding HEX, BIN, OCT is used.

18.6.3.16 OpenFileForWrite

Semantic:

Opens an **OTX File extension** on the specified path with write access and returns a file handle to this file. If the directory does not exist, the directory and all ancestors shall be created automatically. The file shall be automatically locked to prevent simultaneous reading or writing.

IMPORTANT — This term has a side effect.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Associated checker rules:

— File_Chk001 – no lazy evaluation on terms with side effects.

Return type:

file:FileWriteHandleTerm

Parameters:

— <path> : otx:StringTerm [1..1]

The file to open. The path shall be defined by a URI (see ISO 13209-2:—, Annex G).

— <append> : otx:BooleanTerm [0..1]

This parameter is only effective when opening an existing file. If true, the resulting **FileWriteHandle** will initially contain the original contents of the file. If false, the **FileWriteHandle** will be created without initial content. If the element is not set, the default value false applies.

The append flag has no influence on other actions that use the opened **FileWriteHandle**.

— <encoding> : string:EncodingTerm [0..1]

This represents the encoding to be used. The set of standard encodings, which shall be supported by any runtime system is given by the encoding enumeration in StringUtil extension, see ISO 13209-3. If the encoding is not specified, it assumes that Strings will be encoded using UTF-8. The encodings HEX, BIN, OCT are not allowed.

Exceptions:

file:FileLockException

— This exception will be thrown if the file is locked (e.g. for security reasons).

file:FileNotFoundException

— This exception will be thrown if the path (file or directory) does not point to an existing file.

file:FileOpenException

— This exception will be thrown if the open operation failed, or an encoding HEX, BIN, OCT is used.

18.6.3.17 ReadBytes

Semantic:

Reads bytes from the file specified in the file handle and returns a ByteField.

The first read or write activity on a file handle defines if access shall be realised by line, bytes or file. Mixed access will lead to a **FileFormatException**.

If index + count > file size, a ByteField containing the bytes up to the file size shall be returned.

IMPORTANT — This term has a side effect.

Associated checker rules:

— File_Chk001 – no lazy evaluation on terms with side effects.

Return type:

otx:ByteFieldTerm

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Parameters:

— <handle> : file:FileReadHandleTerm [1..1]

The handle to the file that the related file object encapsulates.

— <index> : otx:IntegerTerm [0..1]

Index for reading. If index is omitted, start reading at index of the first unread byte.

— <count> : otx:IntegerTerm [0..1]

The number of bytes to read. If count is omitted, read until the end of the file.

Exceptions:

file:FileAccessException

— This exception will be thrown if the file is not accessible.

file:FileFormatException

— This exception will be thrown if **ReadLine** or **ReadFile** has been called on the file handle before.

file:FileLockException

— This exception will be thrown if a file is locked (e.g. for security reasons).

otx:OutOfBoundsException

— This exception will be thrown if index is negative, or count is negative, or index is greater than or equal to the file size.

18.6.3.18 ReadFile

Semantic:

Reads all lines of a text file specified in the file handle and returns a String. The resulting string will contain line break characters as they were found in the file; no operating system specific conversion shall take place. Subsequent calls to **ReadFile** on the same handle will re-read the contents of the file. The first read or write activity on a file handle defines if an access shall be realised by line, bytes or file. Mixed access will lead to a **FileFormatException**.

IMPORTANT — This term has a side effect.

Associated checker rules:

— File_Chk001 – no lazy evaluation on terms with side effects.

Return type:

otx:StringTerm

Parameters:

— <handle> : file:FileReadHandleTerm [1..1]

The handle for the file that the related file object encapsulates.

Exceptions:

file:FileAccessException

— This exception will be thrown if the file is not accessible.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

file:FileFormatException

— This exception will be thrown if **ReadBytes** or **ReadLine** has been called on the file handle before.

file:FileLockException

— This exception will be thrown if a file is locked (e.g. for security reasons).

18.6.3.19 ReadLine

Semantic:

Reads the next line of characters from the file and returns the data as a String. The line is terminated either by a recognised line break (any of '\r\n', '\r' or '\n') or by reaching the end of the file. The line break is not included in the returned string.

The first read activity on a file handle defines if an access shall be realised by line, bytes or file. Mixed access will lead to a **FileFormatException**.

IMPORTANT — This term has a side effect.

Associated checker rules:

— File_Chk001 – no lazy evaluation on terms with side effects.

Return type:

otx:StringTerm

Parameters:

— <handle> : file:FileReadHandleTerm [1..1]

The handle for the file that the related file object encapsulates.

Exceptions:

file:FileAccessException

— This exception will be thrown if the file is not accessible.

file:FileFormatException

— This exception will be thrown if **ReadBytes** or **ReadLine** has been called on the file handle before.

file:FileLockException

— This exception will be thrown if a file is locked (e.g. for security reasons).

otx:OutOfBoundsException

— This exception will be thrown if the end of file has already been reached by a previous call of the **ReadLine** action.

19 OTX FlashPlus extension

19.1 General

In ISO 13209-3 a flash extension is specified. This extension does not enable late-bound flashing of ECUs in OTX. The **OTX FlashPlus extension** is introduced to enable late-binding of flash files.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

19.2 Exceptions

19.2.1 Exceptions overview

All exceptions introduced by this extension are derived from the **otx:Exception** type. They represent the full set of exceptions added by the **OTX FlashPlus extension**.

19.2.2 Exceptions syntax

The syntax of all **OTX FlashPlus extension** exception-type declarations is shown in Figure 86.

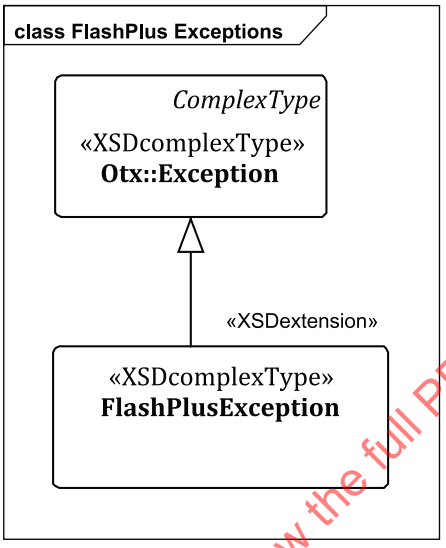


Figure 86 — FlashPlus exceptions

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

19.2.3 Exceptions semantics

19.2.3.1 General

Since all **OTX FlashPlus extension** exception types are implicit exceptions without initialisation parts, they cannot be declared constant.

19.2.3.2 FlashPlusException

The **FlashPlusException** is introduced to allow a specific exception to be thrown if there is an issue during late-bound flashing.

IMPORTANT — All terms and action realisations in this extension may potentially throw this exception.

19.3 Actions

19.3.1 Actions overview

All actions introduced by this extension extend the **otx:ActionRealisation** extension interface.

19.3.2 Actions syntax

Figure 87 shows the syntax of all actions in the **OTX FlashPlus extension**.

© ISO 2021 – All rights reserved

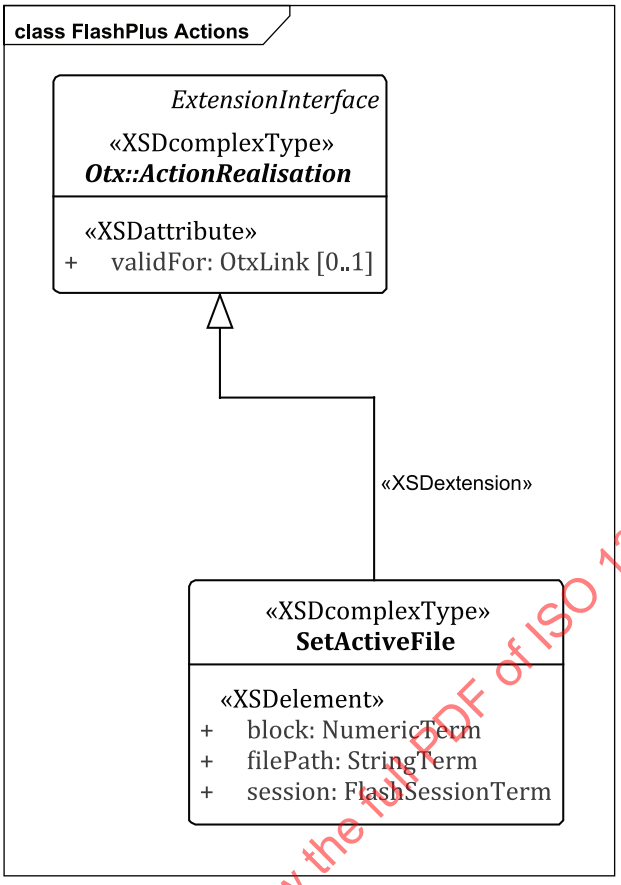


Figure 87 — FlashPlus actions

19.3.3 Actions general semantics

19.3.3.1 SetActiveFile

Semantic:

Sets the flash file name to a temporarily defined flash file name for the given block of the given **flash:FlashSession.**

NOTE The appropriate MVCI system operation is MCDDbFlashData.setActiveFileName.

Parameters:

— <block> : otx:NumericTerm [1..1]

Specifies a block in the **flash:FlashSession** context that shall be addressed. The value shall be in the range of the existing blocks. Float values shall be truncated.

— <filePath> : otx:StringTerm [1..1]

Specifies the path to the file.

— <session> : flash:FlashSessionTerm [1..1]

Specifies the **flash:FlashSession** which provides the data that shall be addressed.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

19.4 Terms

19.4.1 Terms overview

All terms introduced by this extension extend the **otx:Term** extension interface. Information about the specific super class of a term is provided in the individual term description clauses.

19.4.2 Terms syntax

Figure 88 shows the syntax of the **OTX FlashPlus extension** related terms.

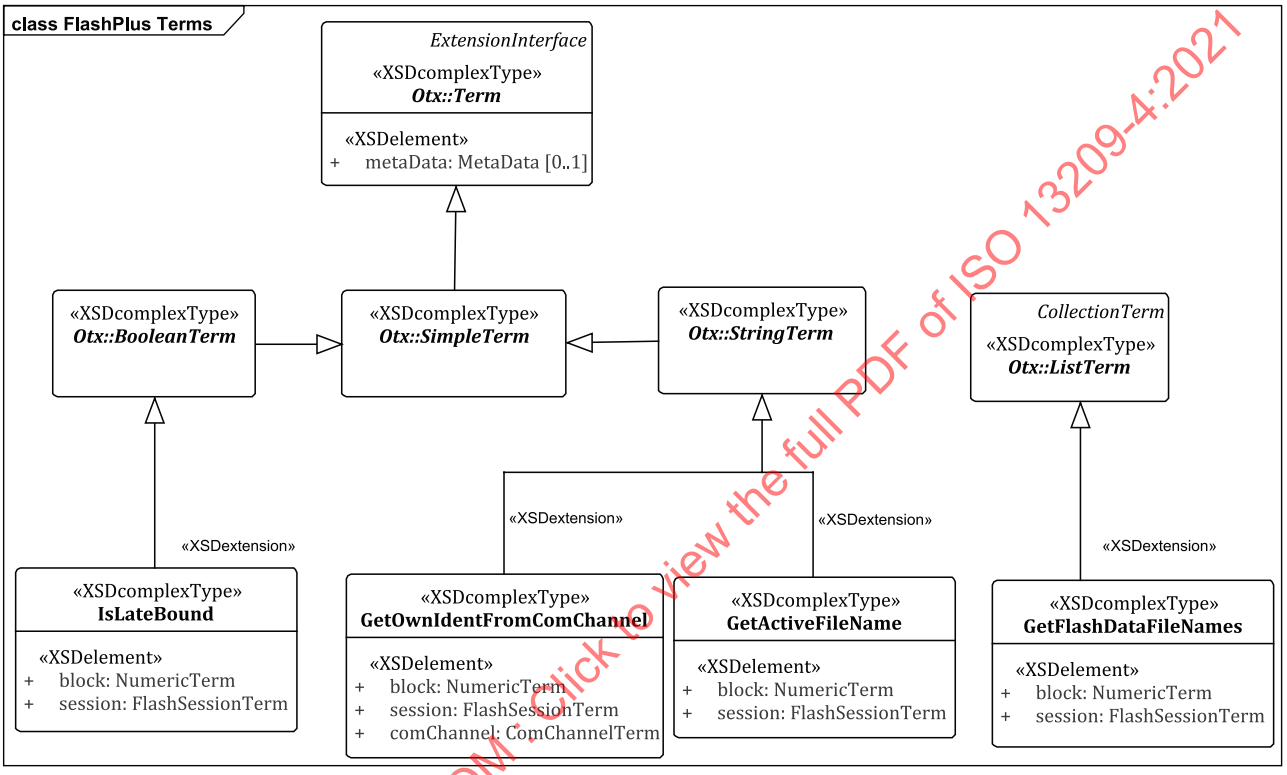


Figure 88 — FlashPlus terms

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

19.4.3 Terms general semantics

19.4.3.1 GetActiveFileName

Semantic:

Returns the active flash file name for the given block of the given **flash:FlashSession** as an **otx: String.**

NOTE The appropriate MVCI system operation is MCDDbFlashData.isLatebound.

Return type:

otx:StringTerm

Parameters:

— <block> : otx:NumericTerm [1..1]

Specifies a block in the **flash:FlashSession** context that shall be addressed. The value shall be in the range of the existing blocks. Float values shall be truncated.

© ISO 2021 – All rights reserved

— <session> : flash:FlashSessionTerm [1..1]

Specifies the **flash:FlashSession** which provides the data that shall be addressed.

Exceptions:

flashPlus:FlashPlusException

— This exception will be thrown if the flash data file name cannot be retrieved.

19.4.3.2 GetFlashDataFileNames

Semantic:

Returns a possibly empty list of strings containing file names which match the pattern obtainable via MCDDbFlashData::getDataFileName().

NOTE The appropriate MVCI system operation is MCDDbFlashData.getMatchingFileNames.

Return type:

otx:ListTerm

Parameters:

— <block> : otx:NumericTerm [1..1]

Specifies a block in the **flash:FlashSession** context that shall be addressed. The value shall be in the range of the existing blocks. Float values shall be truncated.

— <session> : flash:FlashSessionTerm [1..1]

Specifies the **flash:FlashSession** which provides the data that shall be addressed.

Exceptions:

flashPlus:FlashPlusException

— This exception is thrown if the flash data file names cannot be retrieved.

19.4.3.3 GetOwnIdentFromComChannel

Semantic:

The **GetOwnIdentFromComChannel** term shall return an identification string which shall be read from an ECU that is accessed by the given **diag:ComChannel.**

Return type:

otx:StringTerm

Parameters:

— <block> : otx:NumericTerm [1..1]

Specifies a block in the **FlashSession** context that shall be addressed. The value shall be in the range of the existing blocks. Float values shall be truncated.

— <session> : flash:FlashSessionTerm [1..1]

Specifies the **FlashSession,** which provides the data that shall be addressed.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— <comChannel> : diag:ComChannelTerm [1..1]

The communication channel which shall be used for communication with the ECU to get the identification string.

Exceptions:

otx:OutOfBoundsException

— This exception will be thrown if there was no block or own identification number found with the requested block.

19.4.3.4 IsLateBound

Semantic:

Returns true if the given block of the given **FlashSession** can be flashed late-bound. Otherwise returns false.

NOTE The appropriate MVCI system operation is MCDDbFlashData.isLatebound.

Return type:

otx:BooleanTerm

Parameters:

— <block> : otx:NumericTerm [1..1]

Specifies a block in the **FlashSession** context that shall be addressed. The value shall be in the range of the existing blocks. Float values shall be truncated.

— <session> : flash:FlashSessionTerm [1..1]

Specifies the **FlashSession,** which provides the data that shall be addressed.

Exceptions:

flashPlus:FlashPlusException

— This exception will be thrown if it is not possible to find out if the block is late-bound.

20 OTX Persistence extension

20.1 General

The **OTX Persistence extension** is used to store and retrieve runtime information, for example information about ECU variants. 'Persistent' means that the value can be stored in non-volatile storage (for example, a hard drive) and can be retrieved in a different process from the one that created it.

The **OTX Persistence extension** is used for sessions of the same runtime system and not for exchange between different runtimes. The save and load activities can only be performed by the same OTX runtime system.

Each variable can store and recover its own value. The runtime system guarantees the data consistency and is responsible for storing this value. It is also responsible for ensuring that a variable can only be restored with a value previously stored of the same identical variable. To be identical the variable has to be the same data type, package name, document name, procedure name (if applicable) and variable name.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

The **OTX Persistence extension** does not provide a mechanism to delete persisted data – it is the responsibility of the runtime system to provide a mechanism (for example, by a user action or a timed expiry).

20.2 Exceptions

20.2.1 Exceptions overview

All exceptions introduced by this extension are derived from the **otx:Exception** type. They represent the full set of exceptions added by the **OTX Persistence extension**.

20.2.2 Exceptions syntax

The syntax of all **OTX Persistence extension** exception-type declarations is shown in Figure 89.

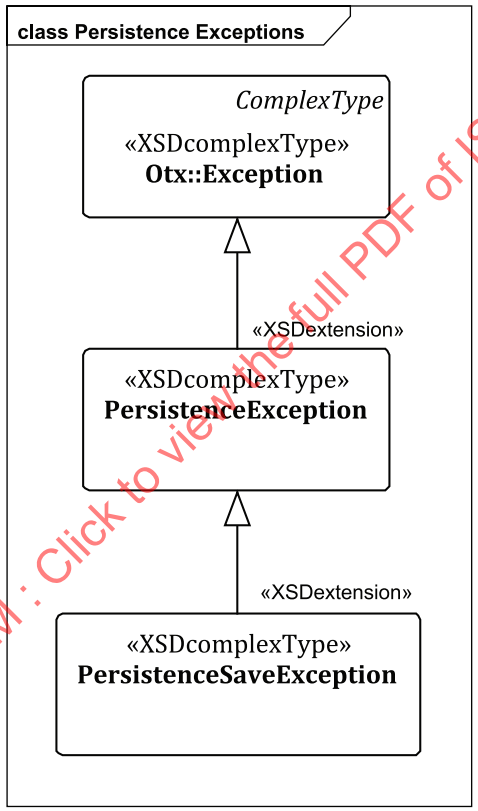


Figure 89 — Persistence exceptions

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

20.2.3 Exceptions semantics

20.2.3.1 General

Since all **OTX Persistence extension** exception types are implicit exceptions without initialisation parts, they cannot be declared constant.

20.2.3.2 PersistenceException

A **PersistenceException** shall be used in case the more specific exception types described in the remainder of this section do not apply to the problem at hand.

IMPORTANT — All terms and action realisations in this extension may potentially throw this exception.

© ISO 2021 – All rights reserved

20.2.3.3 PersistenceSaveException

This exception is thrown if storage of the persistence information was not possible, e.g. because of not enough storage capacity.

20.3 Actions

20.3.1 Actions overview

All actions introduced by this extension extend the **otx:ActionRealisation** extension interface.

20.3.2 Actions syntax

Figure 90 shows the syntax of all actions in the **OTX Persistence extension**.

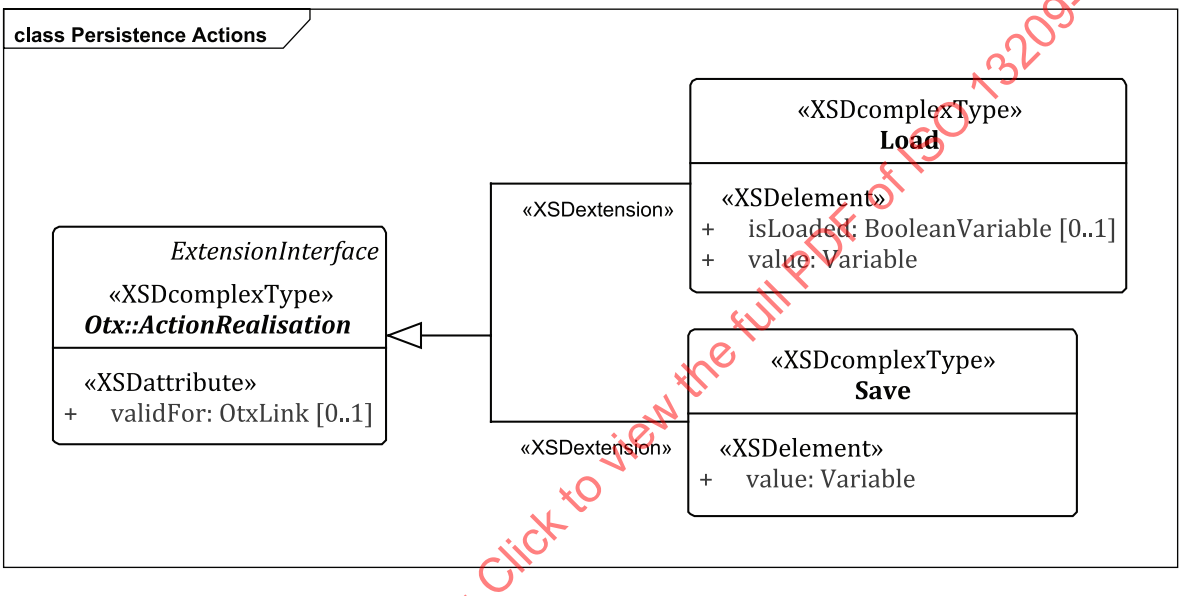


Figure 90 — Persistence actions

20.3.3 Actions general semantics

20.3.3.1 Load

Semantic:

Loads the value of a previously stored OTX variable from persistent storage. If the variable cannot be retrieved, the value of the variable shall be unchanged.

Associated checker rules:

— Persistence_Chk001 – persistable data types,

— Persistence_Chk002 – no Path in persisted value.

Parameters:

— <isLoaded> : otx:BooleanVariable [0..1]

Optional return value. If the OTX variable was previously stored and can be loaded then it is set to true, otherwise false.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— <value> : otx:Variable [1..1]

The variable to be retrieved from the storage.

Exceptions:

— None

20.3.3.2 Save

Semantic:

Saves the value of an OTX variable of a valid data type to the storage.

Associated checker rules:

— Persistence_Chk001 – persistable data types,

— Persistence_Chk002 – no Path in persisted value.

Parameters:

— <value> : otx:Variable [1..1]

The variable to be stored persistently.

Exceptions:

PersistenceSaveException

— This exception will be thrown if saving the value failed.

21 OTX SQL extension

21.1 General

The purpose of the **OTX SQL extension** is to provide simple actions and terms to handle access to an **OTX SQL extension** database. It allows connections to a database and **OTX SQL extension** commands for manipulating the data within the database.

IMPORTANT — The extension shall be independent from a specific data provider. The selection of a specific data provider shall covered by the tool specific runtime implementation.

The types of **OTX SQL extension** should be mapped to OTX and vice versa as described in Table 3.

Table 3 — SQL – OTX Mapping

SQL	OTX
Integer	Integer
numeric, decimal	Float
float	Float
real	Float
double	Float
character, char	String
varchar	String
text	String
date	String
time	String

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Table 3 *(continued)*

SQL	OTX
timestamp	String
boolean	Boolean
blob, clob	ByteField
other data types	not specified

21.2 Datatypes

21.2.1 Datatypes overview

All data types introduced by this extension are derived from **otx:ComplexType**.

21.2.2 Datatypes syntax

The syntax of the data-type declarations of the **OTX SQL extension** is shown in Figure 91.

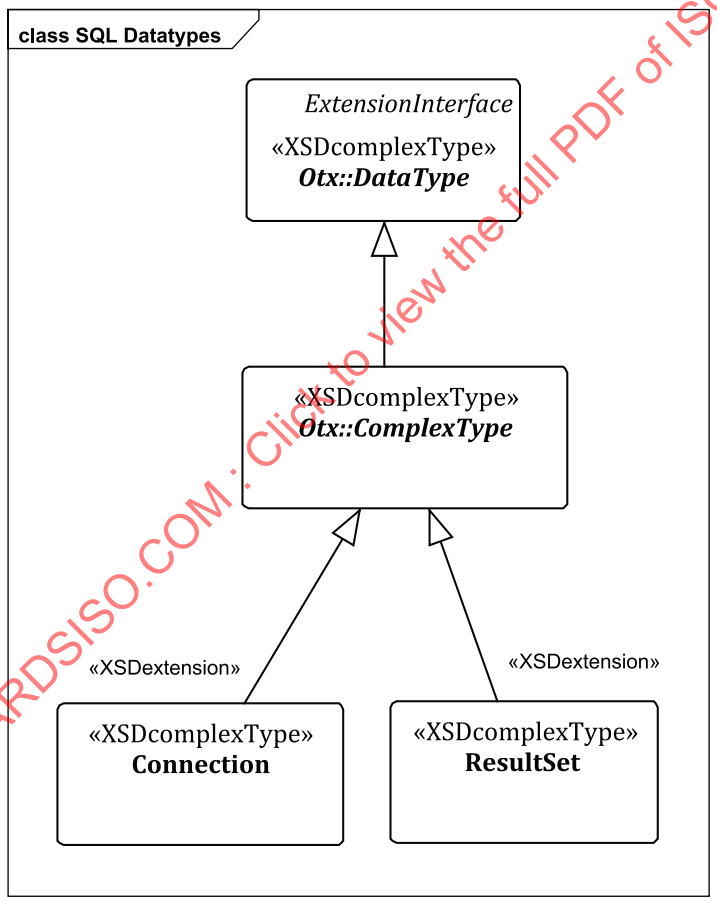


Figure 91 — SQL datatypes

21.2.3 Datatypes semantics

21.2.3.1 Connection

Connection is an **otx:ComplexType**.

A **Connection** represents an open connection to a database.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

21.2.3.2 ResultSet

ResultSet is an **otx:ComplexType**.

The **ResultSet** data type is used to handle the rows read from the database. A result set contains zero or more rows. Each row contains one or more values.

21.3 Exceptions

21.3.1 Exceptions overview

All exceptions introduced by this extension are derived from the **otx:Exception** type. They represent the full set of exceptions added by the **OTX SQL extension**.

21.3.2 Exceptions syntax

The syntax of all **OTX SQL extension** exception-type declarations is shown in Figure 92.

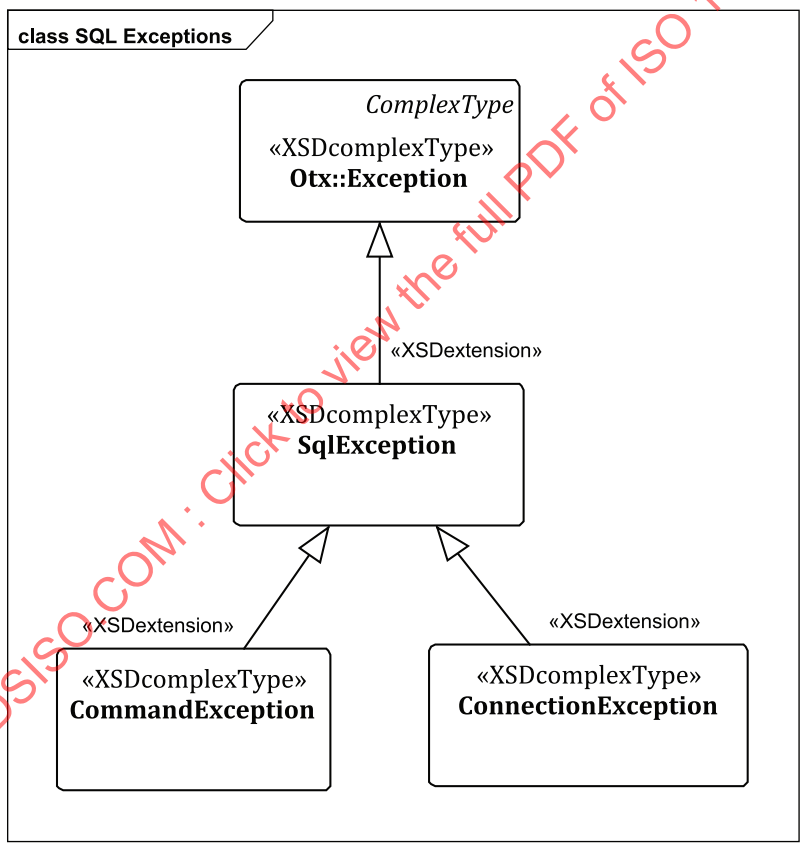


Figure 92 — SQL exceptions

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

21.3.3 Exceptions semantics

21.3.3.1 General

Since all **OTX SQL extension** exception types are implicit exceptions without initialisation parts, they cannot be declared constant.

© ISO 2021 – All rights reserved

21.3.3.2 CommandException

The **CommandException** shall be thrown when there is a problem with the command provided by the user to a database.

21.3.3.3 ConnectionException

A **ConnectionException** shall be thrown when an error occurs during database connection using the connection string provided.

21.3.3.4 SqlException

The **SqlException** is the super class for all exceptions in the **OTX SQL extension**. A **SqlException** shall be used in case the more specific exception types described in the remainder of this section do not apply to the problem at hand.

IMPORTANT — All terms and action realisations in this extension may potentially throw this exception.

21.4 Variable access

21.4.1 Variable-access overview

All variable-access types introduced by this extension are derived from the OTX core variable extension interface.

21.4.2 Variable-access syntax

The syntax of the **OTX SQL extension** variable-access types is shown in Figure 93.

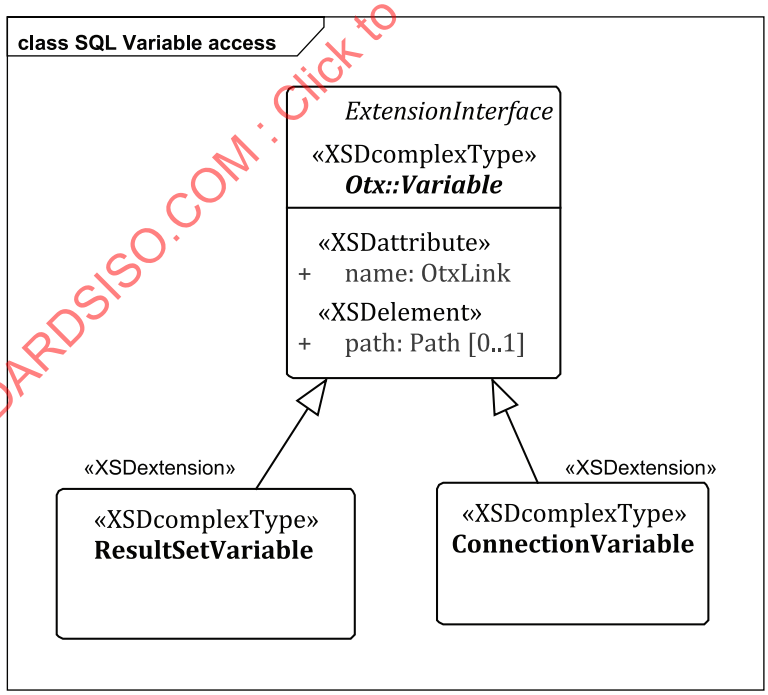


Figure 93 — SQL variable access

21.4.3 Variable-access semantics

The general semantics for all variable-access types shall apply. Please refer to ISO 13209-2 for details.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

21.5 Actions

21.5.1 Actions overview

All actions introduced by this extension extend the **otx:ActionRealisation** extension interface.

21.5.2 Actions syntax

Figure 94 shows the syntax of all actions in the **OTX SQL extension**.

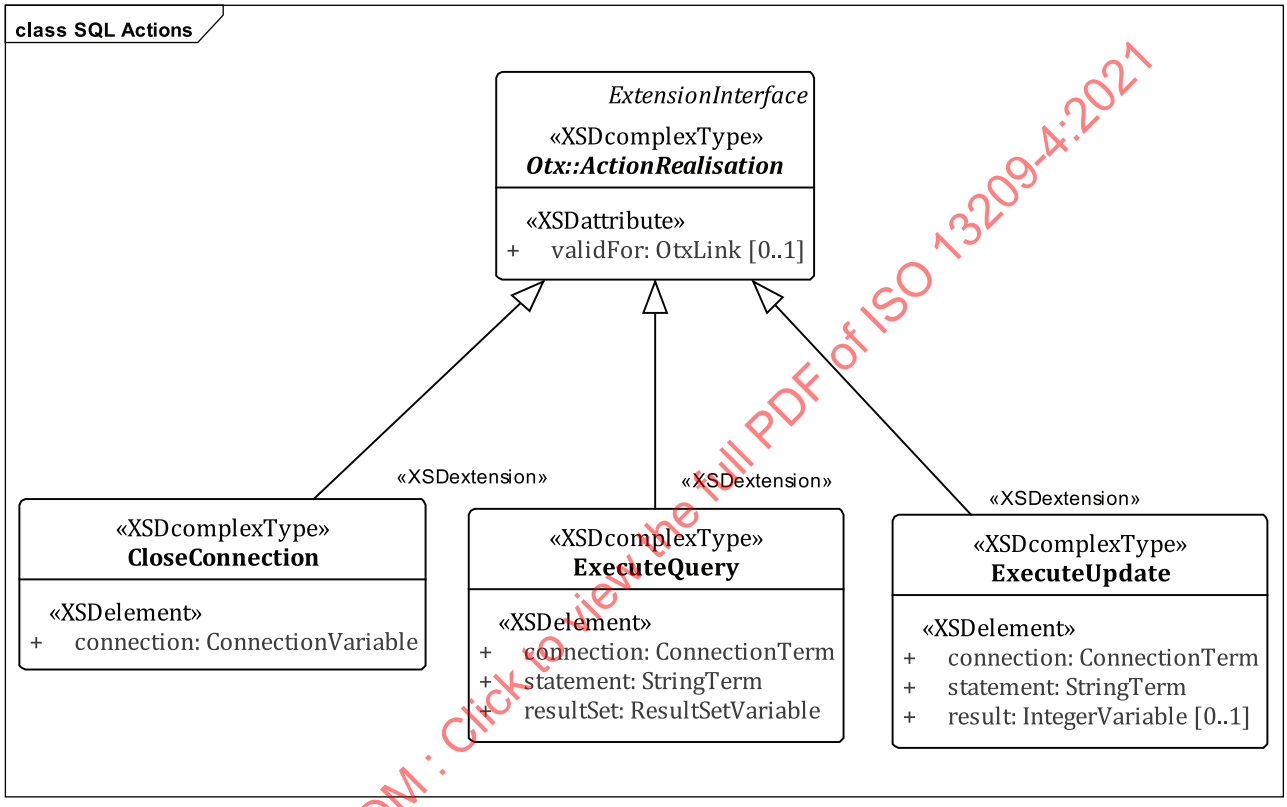


Figure 94 — SQL actions

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

21.5.3 Actions general semantics

21.5.3.1 CloseConnection

Semantic:

The **CloseConnection** action closes the current connection and releases any resources associated with the current connection. The use of **CloseConnection** is optional. An OTX runtime system shall automatically close any open connection at an appropriate time, e.g. at the end of the runtime execution. If a sequence uses a connection after it has been closed by a **CloseConnection** action, the runtime system shall throw an **otx:InvalidReferenceException.**

Parameters:

— <connection> : sql:ConnectionVariable [1..1]

This element represents the specified connection variable that shall be closed.

Exceptions:

sql:ConnectionException

© ISO 2021 – All rights reserved

— A **ConnectionException** shall be thrown if the connection is not valid.

21.5.3.2 ExecuteQuery

Semantic:

The ExecuteSqlQuery action shall execute the given **OTX SQL extension** statement, which returns a single **ResultSet** object.

NOTE After executing a command the **ResultSet** cursor is initially positioned before the first row.

Parameters:

— <connection> : sql:ConnectionTerm [1..1]

This element represents a valid **Connection**.

— <statement> : otx:StringTerm [1..1]

Mandatory **OTX SQL extension** statement string which is sent to the database, typically a static **OTX SQL extension** SELECT statement.

— <resultSet> : sql:ResultSetVariable [1..1]

Returns a **ResultSet** which contains the result of the executed reading command.

Exceptions:

sql:CommandException

— A **CommandException** shall be thrown if an error occurs during execution of the command.

sql:ConnectionException

— A **ConnectionException** shall be thrown if the connection is not valid.

21.5.3.3 ExecuteUpdate

Semantic:

The ExecuteSqlQuery action executes the given **OTX SQL extension** statement, which may be an INSERT, UPDATE or DELETE statement or an **OTX SQL extension** statement that returns nothing, such as an **OTX SQL extension** DDL statement.

Parameters:

— <connection> : sql:ConnectionTerm [1..1]

This element represents a valid **Connection**.

— <statement> : otx:StringTerm [1..1]

Mandatory **OTX SQL extension** statement string which is sent to the database.

— <result> : otx:IntegerVariable [0..1]

Returns the row count for **OTX SQL extension** data manipulation statements or 0 for **OTX SQL extension** statements that return nothing.

Exceptions:

sql:CommandException

— A **CommandException** shall be thrown if an error occurs during execution of the command.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

sql:ConnectionException

— A **ConnectionException** shall be thrown if the connection is not valid.

21.6 Terms

21.6.1 Terms overview

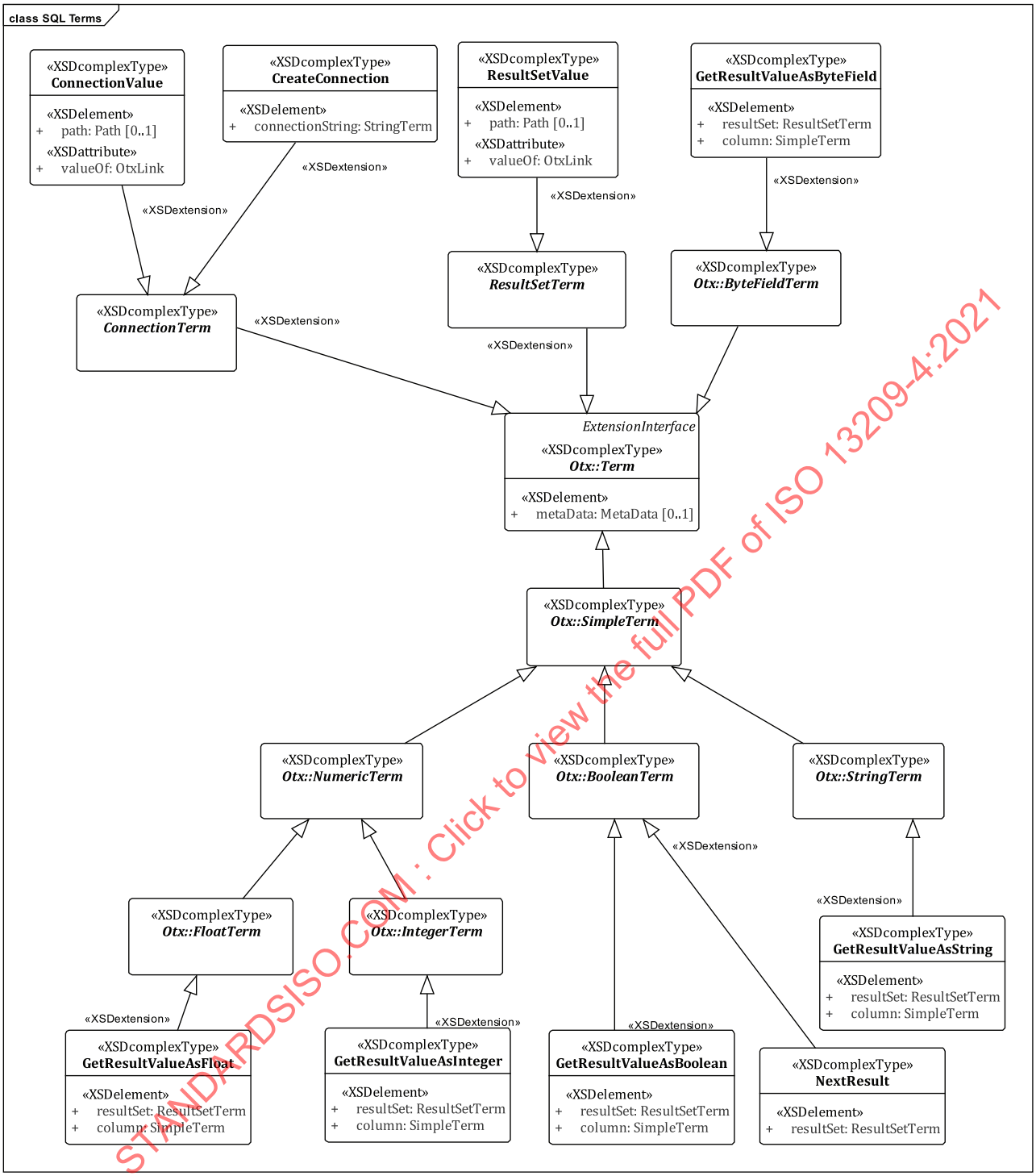
All terms introduced by this extension extend the **otx:Term** extension interface. Information about the specific super class of a term is provided in the individual term description clauses.

21.6.2 Terms syntax

Figure 95 shows the syntax of the **OTX SQL extension** related terms.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 95 — SQL terms

21.6.3 Terms general semantics

21.6.3.1 ConnectionTerm

Semantic:

A **ConnectionTerm** is a term that returns a connection to perform operations on a database.

© ISO 2021 – All rights reserved

21.6.3.2 ConnectionValue

Semantic:

Returns the **ConnectionValue** stored in a **Connection** variable.

Return type:

sql:ConnectionTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

21.6.3.3 CreateConnection

Semantic:

The **CreateConnection** term shall open a connection using the specified connection string for access and returns the **Connection** for the application.

Return type:

sql:ConnectionTerm

Parameters:

— <connectionString>: otx:StringTerm [1..1]

Specifies the location of the data source that is required for the connection. ConnectionString is used to create the connection that passes for the creation of command variable. The connection string includes the source database name and other parameters needed to establish the initial connection.

Exceptions:

sql:ConnectionException

— A **ConnectionException** shall be thrown if the connection string is invalid, or if there is an attempt to create a new connection with a wrong data source path.

21.6.3.4 GetResultValueAsBoolean

Semantic:

The **GetResultValueAsBoolean** returns the value of the given column (via index or name) inside the row as a Boolean.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Associated checker rules:

— SQL_Chk001 – right column data type.

Return type:

otx:BooleanTerm

Parameters:

— <resultSet> : sql:ResultSetTerm [1..1]

This element represents the **ResultSet** from which the value of the current row shall be read.

— <column> : otx:SimpleTerm [1..1]

This element represents the column from which the value shall be read. A column can be accessed via name or via index. Therefore, the data type of the column property shall be String (name) or Integer (index).

Exceptions:

otx:OutOfBoundsException

— This exception is thrown if the name of the given column does not exists in the data record.

otx:OutOfBoundsException

— This exception is thrown if the index of the given column is not within the range [0, n-1], where n is the number of the data record columns.

otx:OutOfBoundsException

— This exception is thrown if result is empty.

otx:TypeMismatchException

— This exception is thrown if the data type of the column value is not a Boolean.

21.6.3.5 GetResultValueAsByteField

Semantic:

The **GetResultValueAsByteField** returns the value of the given column (via index or name) inside the row as a ByteField.

Associated checker rules:

— SQL_Chk001 – right column data type.

Return type:

otx:ByteFieldTerm

Parameters:

— <resultSet> : sql:ResultSetTerm [1..1]

This element represents the **ResultSet** from which the value of the current row shall be read.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— <column> : otx:SimpleTerm [1..1]

This element represents the column from which the value shall be read. A column can be accessed via name or via index. Therefore the data type of the column property shall be String (name) or Integer (index).

Exceptions:

otx:OutOfBoundsException

— This exception is thrown if the name of the given column does not exists in the data record.

otx:OutOfBoundsException

— This exception is thrown if the index of the given column is not within the range [0, n-1], where n is the number of the data record columns.

otx:OutOfBoundsException

— This exception is thrown if result is empty.

otx:TypeMismatchException

— This exception is thrown if the data type of the column value is not a ByteField.

21.6.3.6 GetResultValueAsFloat

Semantic:

The **GetResultValueAsFloat** returns the value of the given column (via index or name) inside the row as a Float.

Associated checker rules:

— SQL_Chk001 – right column data type.

Return type:

otx:FloatTerm

Parameters:

— <resultSet> : sql:ResultSetTerm [1..1]

This element represents the **ResultSet** from which the value of the current row shall be read.

— <column> : otx:SimpleTerm [1..1]

This element represents the column from which the value shall be read. A column can be accessed via name or via index. Therefore the data type of the column property shall be String (name) or Integer (index).

Exceptions:

otx:OutOfBoundsException

— This exception is thrown if the name of the given column does not exists in the data record.

otx:OutOfBoundsException

— This exception is thrown if the index of the given column is not within the range [0, n-1], where n is the number of the data record columns.

otx:OutOfBoundsException

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— This exception is thrown if result is empty.

otx:TypeMismatchException

— This exception is thrown if the data type of the column value is not a Float.

21.6.3.7 GetResultValueAsInteger

Semantic:

The **GetResultValueAsInteger** returns the value of the given column (via index or name) inside the row as an Integer.

Associated checker rules:

— SQL_Chk001 – right column data type.

Return type:

otx:IntegerTerm

Parameters:

— <resultSet> : sql:ResultSetTerm [1..1]

This element represents the **ResultSet** from which the value of the current row shall be read.

— <column> : otx:SimpleTerm [1..1]

This element represents the column from which the value shall be read. A column can be accessed via name or via index. Therefore the data type of the column property shall be String (name) or Integer (index).

Exceptions:

otx:OutOfBoundsException

— This exception is thrown if the name of the given column does not exists in the data record.

otx:OutOfBoundsException

— This exception is thrown if the index of the given column is not within the range [0, n-1], where n is the number of the data record columns.

otx:OutOfBoundsException

— This exception is thrown if result is empty.

otx:TypeMismatchException

— This exception is thrown if the data type of the column value is not an Integer.

21.6.3.8 GetResultValueAsString

Semantic:

The **GetResultValueAsString** returns the value of the given column (via index or name) inside the row as a String.

Associated checker rules:

— SQL_Chk001 – right column data type.

Return type:

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

otx:StringTerm

Parameters:

— <resultSet> : sql:ResultSetTerm [1..1]

This element represents the **ResultSet** from which the value of the current row shall be read.

— <column> : otx:SimpleTerm [1..1]

This element represents the column from which the value shall be read. A column can be accessed via name or via index. Therefore, the data type of the column property shall be String (name) or Integer (index).

Exceptions:

otx:OutOfBoundsException

— This exception is thrown if the name of the given column does not exists in the data record.

otx:OutOfBoundsException

— This exception is thrown if the index of the given column is not within the range [0, n-1], where n is the number of the data record columns.

otx:OutOfBoundsException

— This exception is thrown if result is empty.

otx:TypeMismatchException

— This exception is thrown if the data type of the column value is not a String.

21.6.3.9 NextResult

Semantic:

The **NextResult** term moves the cursor forward one row from its current position. A **ResultSet** cursor is initially positioned before the first row; the first call to the term **NextResult** makes the first row the current row; the second call makes the second row the current row, and so on. When a call to the **NextResult** term returns false, the cursor is positioned after the last row.

IMPORTANT — This term has a side effect.

Associated checker rules:

— SQL_Chk002 – no lazy evaluation on terms with side effects.

Return type:

otx:BooleanTerm

Parameters:

— <resultSet> : sql:ResultSetTerm [1..1]

This element represents the result set from which the next row shall be selected.

Exceptions:

— None

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

21.6.3.10 ResultSetTerm

Semantic:

A **ResultSetTerm** is a term that returns a result set of a database operation.

21.6.3.11 ResultSetValue

Semantic:

Returns the **ResultSetValue** stored in a **ResultSet** variable.

Return type:

sql:ResultSetTerm

Parameters:

— <path> : otx:Path [0..1]

This element addresses parts of complex structures. For more information please refer to ISO 13209-2.

Exceptions:

otx:InvalidReferenceException

— This exception is thrown if the variable value is not valid (no value is assigned to the variable).

otx:OutOfBoundsException

— Occurs only, if a <path> is set and the <path> points to a location which does not exist (like a list index exceeding list length, or a map key which is not part of the map).

22 OTX StateMachineProcedure extension

22.1 General

A finite-state machine (FSM) or simply a state machine, is conceived as an abstract machine that can be in one of a finite number of states. The machine is in only one state at a time; the state it is in at any given time is called the current state. It can change from one state to another when initiated by a triggering event or condition; this is called a transition. A particular FSM is defined by a list of its states, and the triggering condition for each transition.

Each state machine shall have one mandatory state which is marked as initial state (State I in Figure 96). It can have one optional state which is marked as completed state. The initial state and the completed state shall be different.

Every state has 1 to n triggers. Each trigger has 1 to n event sources. If an event source, specified inside the triggers of the current state fires an event, the wait state of the current state will be terminated.

If a trigger terminates its wait state, all transitions (in order of XML occurrence) which refer to this trigger will check their condition. The first transition whose condition is true or which has no condition will be executed. All further transitions will be ignored.

If a transition is active the optional flow of that transition shall be executed first. Only then will the target state of the transition shall be reached.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

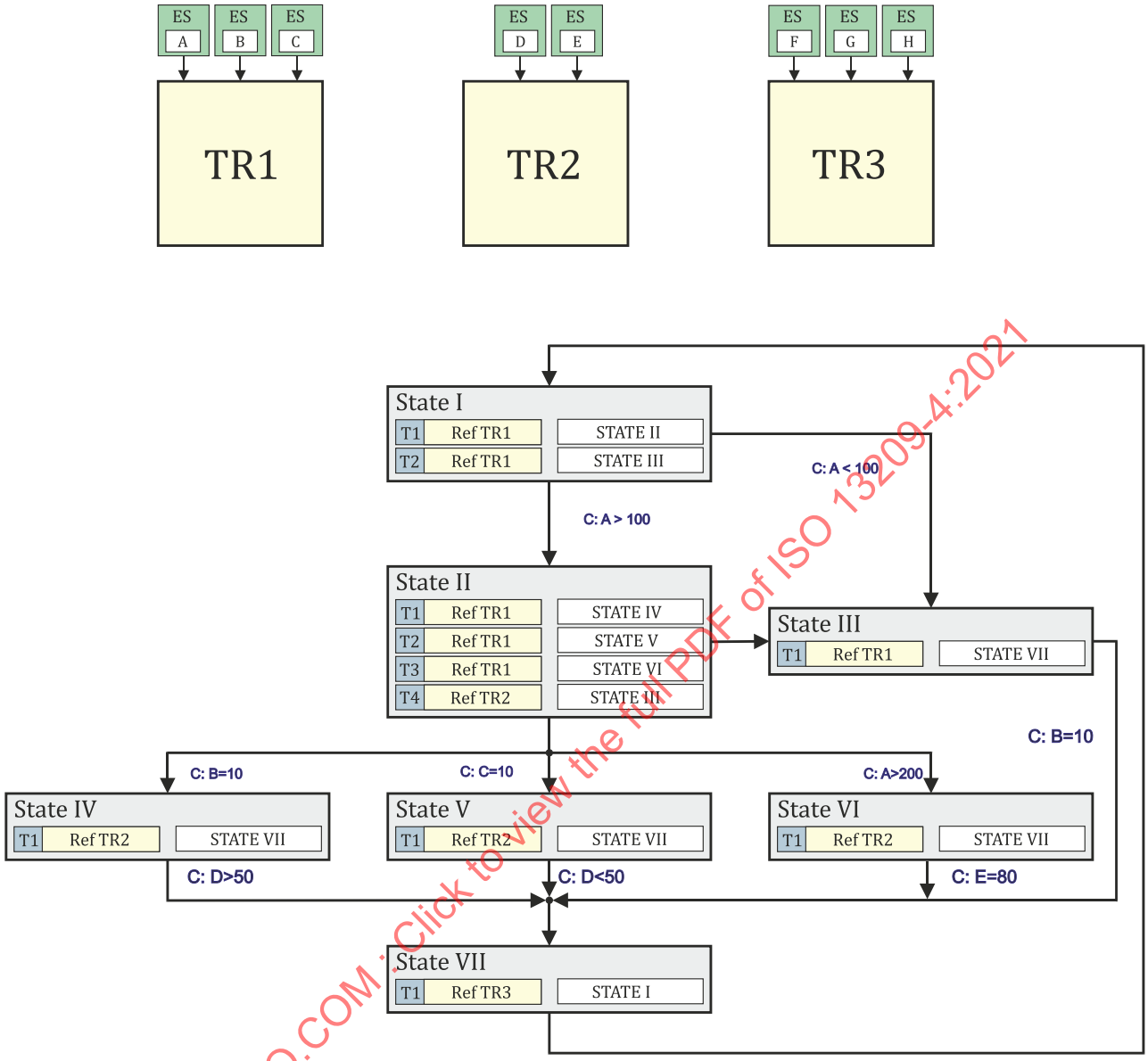


Figure 96 — Principal state machine

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Every state has an entry and an exit flow, which both are optional. On entering a state its entry flow is processed once. If a transition becomes active, the following processing order is defined:

— exit flow of the source state,

— flow of transition,

— entry flow of the target state.

This is also valid if the target state is identical to the source state (loopback transition).

IMPORTANT — - It is possible, that two or more triggers become active at the same time. Via the order of XML occurrence of the transitions, the author can specify the transition priority.

IMPORTANT — - A StateMachine is executed via a standalone ProcedureRealisation. In state machines no layout information shall be saved. A tool supplier is free to store layout information in the meta data. But the meta data shall not contain runtime relevant information.

© ISO 2021 – All rights reserved

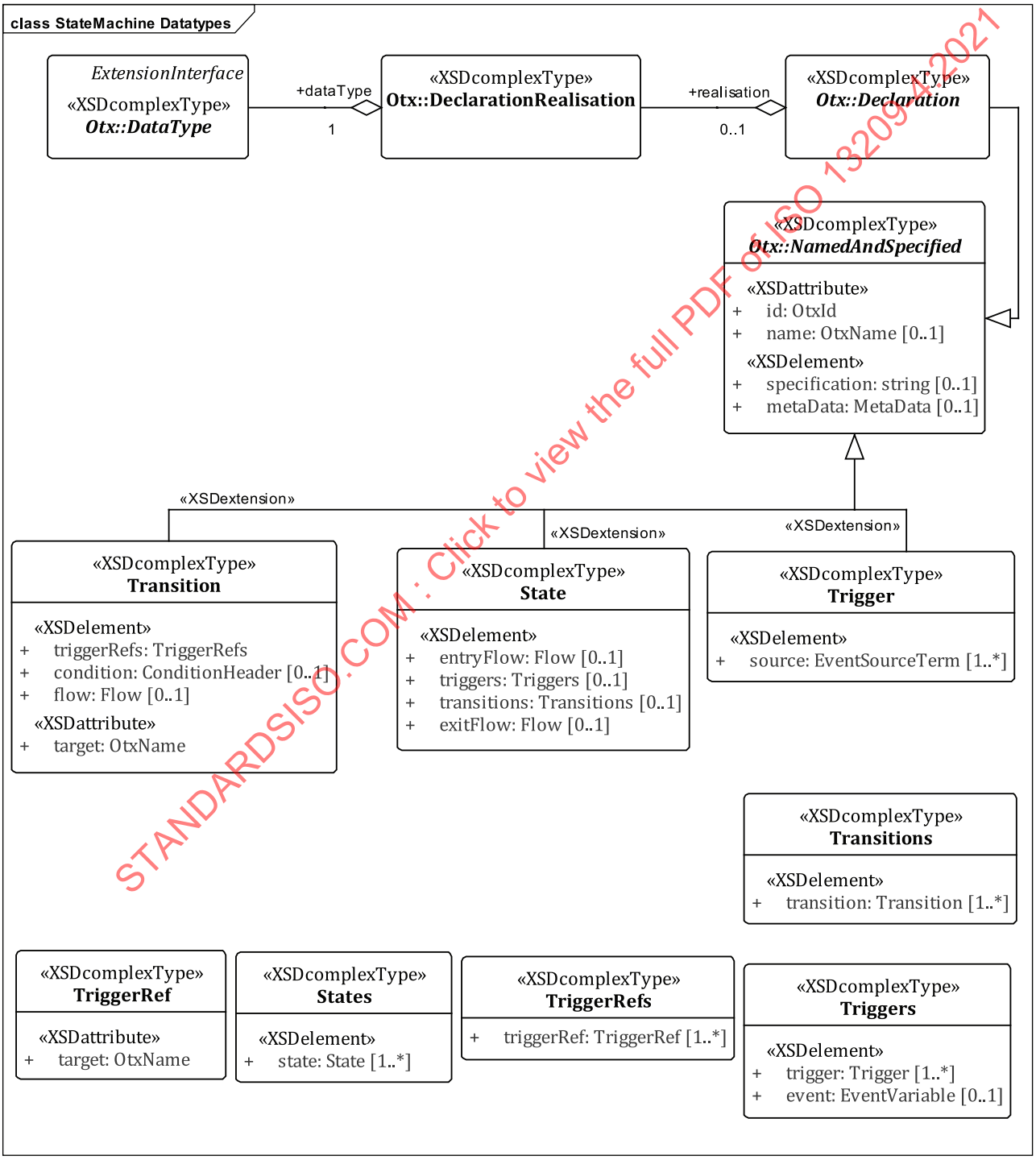
22.2 Datatypes

22.2.1 Datatypes overview

All data types introduced by this extension are derived from **otx:ComplexType**.

22.2.2 Datatypes syntax

The syntax of the data-type declarations of the **StateMachineProcedure** extension is shown in Figure 97.



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 97 — StateMachine datatypes

© ISO 2021 – All rights reserved

22.2.3 Datatypes semantics

22.2.3.1 State

State is an **otx:NamedAndSpecified**.

A **State** within a state machine.

IMPORTANT — A state can be referred by the target attribute of a transition. Therefore, the name attribute is mandatory for states (unlike the base type NamedAndSpecified where it is optional). This is enforced by <xsd:key> constraints specified in the OTX schema.

Its members have the following semantics:

— <entryFlow> : otx:Flow[0..1]

If the state machine switches to this state, if available, the entryFlow will execute first. For more information regarding from please refer to ISO 13209-2.

— <triggers> : smp:Triggers[0..1]

The collection of all triggers related to this state. The order of the triggers is not specified. This means that if two trigger event sources fire at the same time, it cannot be guaranteed that the transitions referred to by the first trigger (in XML order) are performed.

Each state except the completed (final) state shall have at least one trigger.

— <transitions> : smp:Transitions[0..1]

Collection of all transitions related to this state. If a trigger terminates its wait state, all transitions which refer to this trigger will be checked in order of XML occurrence.

Each state except the completed state shall have at least one transition.

— <exitFlow> : otx:Flow[0..1]

If the state machine leaves this state, if available, the exitFlow will execute last. For more information regarding from please refer to ISO 13209-2.

Associated checker rules:

— StateMachineProcedure_Chk004 – mandatory trigger,

— StateMachineProcedure_Chk005 – mandatory transition.

22.2.3.2 States

The collection of all states of the state machine. At least one state shall exist: the InitialState.

Its members have the following semantics:

— <state> : smp:State[1..*]

The single states of a state machine.

22.2.3.3 Transition

Transition is an **otx:NamedAndSpecified**.

A transition is a possible switch from one state to another. If the condition in a transition evaluates to true, no further transitions will be checked; the flow of this transition is executed, if available, it leaves the current state and it switches to the target state. The target state can be an arbitrary state inside

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

this ProcedureRealisation as well as the same state. All conditions of the transition collection should be checked according to Core_Chk055 "unreachable nodes".

Its members have the following semantics:

— <triggerRefs> : smp:TriggersRef[1..1]

  Refers to one or more triggers defined in this state.

— <condition> : otx:ConditionHeader[0..1]

  Optional boolean term which represents the condition of the trigger. The condition can be omitted. In this case the condition results in true and the transition has to be executed. For more information please refer to ISO 13209-2.

— <flow> : otx:Flow[0..1]

  Optional flow which is executed if the condition is true before switching to the new state. For more information please refer to ISO 13209-2.

— <target> : otx:OtxName[0..1]

  Defines the next state to which the state machine should be switched. The new state can be the same state.

Associated checker rules:

— Core_Chk055 - unreachable nodes,

— StateMachineProcedure_Chk002 – mandatory target state,

— StateMachineProcedure_Chk003 – no target state for completed state,

— StateMachineProcedure_Chk009 – correct nesting of transition target state,

— StateMachineProcedure_Chk011 – unreachable states.

22.2.3.4 Transitions

Collection of all transitions of the state machine.

Its members have the following semantics:

— <transition> : smp:Transition[1..*]

  Transition of a state machine.

22.2.3.5 Trigger

Trigger is an **otx:NamedAndSpecified**.

A trigger defines the wait action for a state. A trigger mainly consists of one or more event sources. It has the same behaviour as the **event:WaitForEvent** action in OTX EventHandling extension. An arbitrary number of triggers can be defined for each state. A trigger can be referred to by one or more transitions.

After execution of the entry flow the state blocks the thread of execution until it receives an event from one of its trigger event sources. As soon as an event becomes available in one of the sources event queues, the condition in the referenced transitions shall be checked in order of occurrence in XML.

IMPORTANT — A trigger can be referenced by the target attribute of a transition trigger. Therefore, the name attribute is mandatory for triggers (unlike the base type NamedAndSpecified where it is optional). This is enforced by <xsd:key> constraints specified in the OTX schema.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Its members have the following semantics:

— <source> : event:EventSourceTerm[1..*]

  This represents one or more event sources that the trigger shall wait for. The wait shall be terminated by the first source to fire an event.

22.2.3.6 TriggerRef

It is a reference to a trigger.

Its members have the following semantics:

— <target> : otx:OtxName[1..1]

  Name of the referenced trigger inside this state.

Associated checker rules:

— StateMachineProcedure_Chk010 – correct nesting of transition trigger references.

22.2.3.7 TriggerRefs

All referenced triggers.

Its members have the following semantics:

— <triggerRef> : smp:TriggerRef[1..*]

  Refers to a trigger defined in this state.

22.2.3.8 Triggers

Collection of all triggers of the state machine.

Its members have the following semantics:

— <trigger> : smp:Trigger[1..*]

  Trigger of a state machine.

— <event> : event:EventVariable[0..1]

  Represents an optional Event-type variable which shall receive the event that terminates the trigger.

22.3 Procedures

22.3.1 Procedures overview

All procedures introduced by this extension extend the **otx:procedure** or replace the **otx:procedure** realisation.

22.3.2 Procedures syntax

Figure 98 shows the syntax of the **StateMachineProcedure** related procedures.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

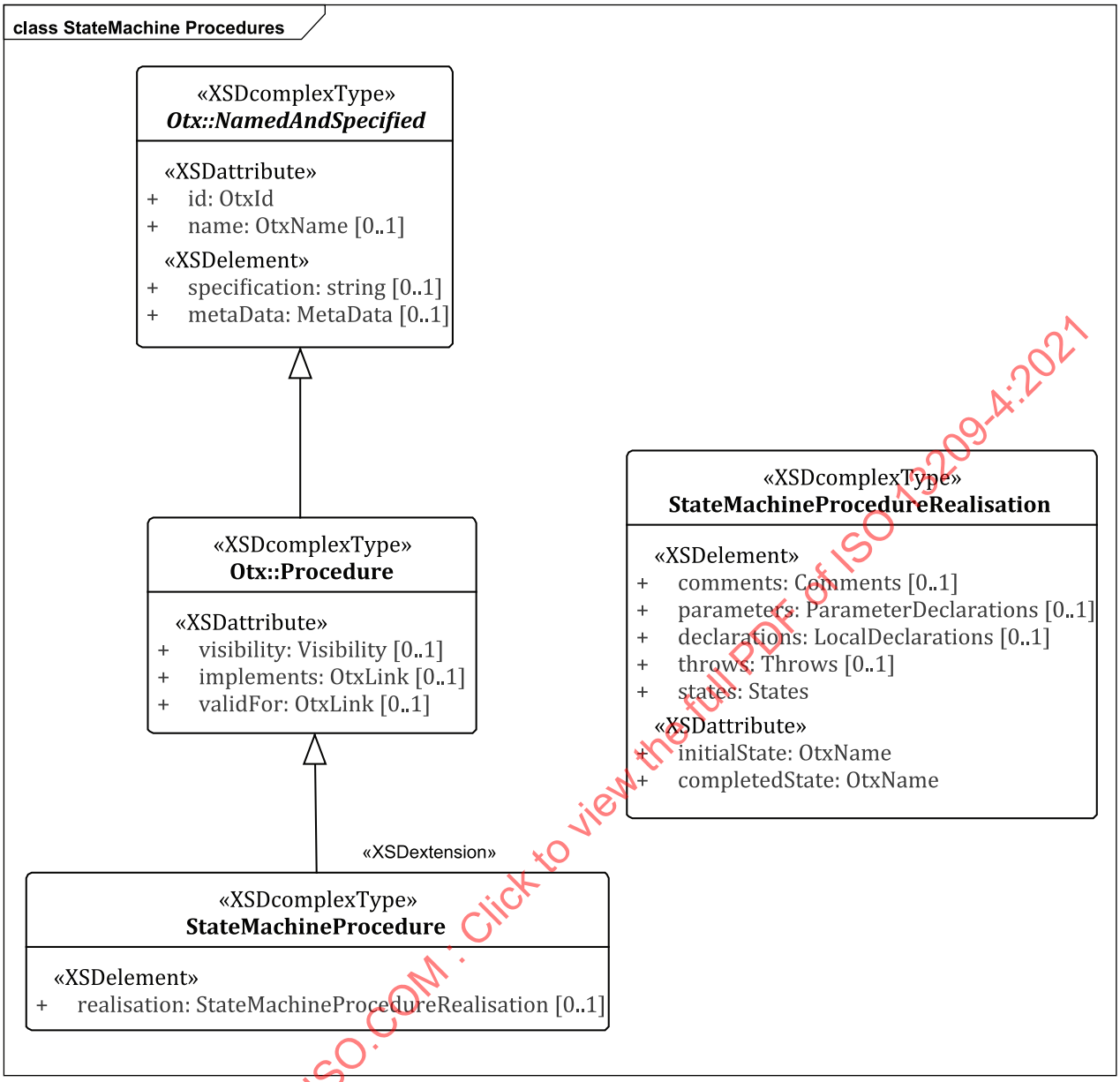


Figure 98 — StateMachine procedures

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

22.3.3 Procedures general semantics

22.3.3.1 StateMachineProcedure

Semantic:

A **StateMachineProcedure** replaces the ProcedureRealisation by the **StateMachineProcedureRealisation**. A **StateMachineProcedure** shall not have a ProcedureRealisation. This syntax cannot be expressed by an XSD schema see.

Associated checker rules:

— StateMachineProcedure_Chk001 – no procedure realisation,

— Core_Chk058 – unique OtxIds.

© ISO 2021 – All rights reserved

Parameters:

— visibility : otx:Visibility [0..1] (derived from Procedure)

This value represents the visibility modifier. For more information please refer to ISO 13209-2.

— implements : otx:OtxLink [0..1] (derived from Procedure)

This value represents the qualified name of the signature that a procedure implements. For more information please refer to ISO 13209-2.

— id : otx:OtxId [1..1] (derived from NamedAndSpecified)

This value represents the OtxId of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— name : otx:OtxName [0..1] (derived from NamedAndSpecified)

This value represents the OtxName of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

— <realisation> : smp:StateMachineProcedureRealisation [0..1]

Specifies the procedure implementation parts.

IMPORTANT — The added key constraints for unique OtxIds related to state machines cannot cover OtxIds of an other state machine procedure inside the same document. Therefore, it should be checked via the checker rule Core_Chk058.

```
<xsd:key name="UniqueStateMachineLocalIdentifierName">
<xsd:selector
xpath="smp:parameters/otx:*|smp:declarations/otx:*|smp:states/smp:state|smp:states/smp:state
/smp:triggers/smp:trigger|smp:states/smp:state/smp:transitions/smp:transition" />
<xsd:field xpath="@name" /></xsd:key>
<xsd:key name="StateKeyForReferences"><xsd:selector
xpath=".//smp:state" /><xsd:field xpath="@name" /></xsd:key>
<xsd:keyref name="TransitionStateRef"
refer="smp:StateKeyForReferences"><xsd:selector xpath=".//smp:transition" /><xsd:field
xpath="@target"
/></xsd:keyref>
<xsd:keyref name="InitialStateRef"
refer="smp:StateKeyForReferences"><xsd:selector xpath="."
/><xsd:field xpath="@initialState" /></xsd:keyref>
<xsd:keyref name="CompletedStateRef"
refer="smp:StateKeyForReferences"><xsd:selector xpath="."
/><xsd:field xpath="@completedState" /></xsd:keyref>
```

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

<xsd:key name="TriggerKeyForTransitions"><xsd:selector

xpath=".//smp:trigger" /><xsd:field xpath="@name" /></xsd:key>

<xsd:keyref name="TriggerRef"

refer="smp:TriggerKeyForTransitions"><xsd:selector xpath=".//smp:triggerRef" /><xsd:field

xpath="@target"

/></xsd:keyref>

<xsd:keyref name="StateMachineCommentTargetId"

refer="otx:UniqueNodeIds"><xsd:selector xpath="smp:comments/otx:comment/otx:links/otx:link"

/><xsd:field xpath="@targetId" /></xsd:keyref>

— <specification> : xs:string [0..1] (derived from NamedAndSpecified)

This value represents the string of the **otx:NamedAndSpecified** element. For more information please refer to ISO 13209-2.

Exceptions:

— None

22.3.3.2 StateMachineProcedureRealisation

Semantic:

Specifies the procedure implementation parts.

IMPORTANT — The added key constraints for unique OtxIds related to state machines cannot cover OtxIds of an other state machine procedure inside the same document.

Associated checker rules:

— Core_Chk058 – unique OtxIds,

— StateMachineProcedure_Chk006 – initial and completed state shall be distinguished,

— StateMachineProcedure_Chk007 – correct nesting of initial state,

— StateMachineProcedure_Chk008 – correct nesting of completed state.

Parameters:

— initialState : otx:OtxName [1..1]

Defines the initial state of the state machine. The value shall exist and shall be distinguished from the value of attribute CompletedState.

The initial state is the first state after starting the procedure execution.

— completedState : otx:OtxName [0..1]

Defines the completed state of the statemachine. The value is optional and shall be distinguished from the value of attribute InitialState.

After finishing the completed state the procedure is finished and shall return to the caller.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

 © ISO 2021 – All rights reserved

— <comments> : otx:Comments [0..1]

This is a container for an arbitrary-length list comments. They shall be used for commenting parts of the procedure flow implementation for the human reader. For more information please refer to ISO 13209-2.

NOTE Entities to which a comment's <link> may point to are described in ISO 13209-2 (e.g. to flow nodes, branch case conditions, loop conditions). In addition, state machine comments can refer to states, transitions, transition conditions, triggers as well as state machine comments themselves.

— <parameters> : otx:ParameterDeclarations [0..1]

Parameters of the procedure. For more information please refer to ISO 13209-2.

— <declarations> : otx:LocalDeclarations [0..1]

Local variables and constants of the procedure. For more information please refer to ISO 13209-2.

— <throws> : otx:Throws [0..1]

Exceptions of the procedure. For more information please refer to ISO 13209-2.

— <states> : smp:States [1..1]

States of a state machine. A state machine consists of at least one state. There are two special states, the initial state and the (optional) completed state. The mandatory initial state is the first state and the optional completed state is the last state of the state machine.

Exceptions:

— None

23 OTX StateVariable extension

23.1 General

Test sequences require a technique to provide different status information to the environment, e.g. progress of execution or currently selected ECU. This is the intention of state variables. State variables are the counterpart of context variables. They provide the mechanism to transport status information from inside a sequence to the environment, whilst context variables are the mechanism to transport environment information to the sequence.

IMPORTANT — State variables can only be set and cannot read inside OTX. In other words: the test logic shall not be based on the value of a state variable.

IMPORTANT — OTX documents declaring state variables should be connected to corresponding setting routines which can set the current value of the status in scope. Therefore, test applications require a mapping between state variables and according setting routines. The mapping may be kept in application specific places, but it may also be contained in meta-data compartments of an OTX document (see ISO 13209-2:—, 7.16.5). The recommendations for meta-data compartments described in ISO 13209-2:—, 7.5.3 shall be applied also for state variables.

NOTE For technical reasons, state:StateVariableDeclaration is defined as a sub type of otx:ContextVariableDeclaration. Regardless, state variables are not context variables. Statements about context variables do not automatically apply to state variables.

Changes of a state variable shall be immediately reported to the runtime environment.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

23.2 Declaration and arguments

23.2.1 Declaration and arguments syntax

Figure 99 shows the syntax of the StateVariable extension's declarations and arguments.

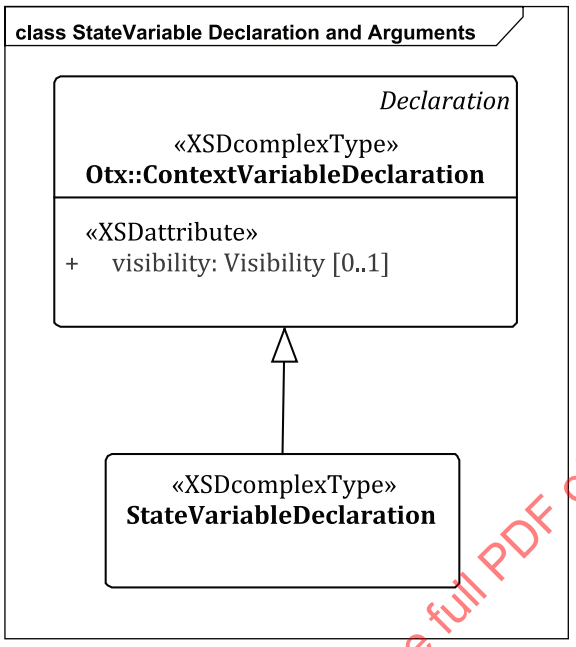


Figure 99 — StateVariable declaration and arguments

23.2.2 Declaration and arguments semantics

The semantic of the declarations and arguments is shown at the actions and terms, where these are used.

Associated checker rules for StateVariableDeclaration:

— StateVariable_Chk001 – write-only state variables.

24 OTX TestResultHandling extension

24.1 General

The OTX TestResultHandling extension provides functionality which allows for capturing, evaluating and persistently writing test results of test sequences, e.g. measured values, in a structured way. The following use cases are considered:

— capturing of test results in relation to the environment (e.g. test object , test identifier, date/time),

— evaluation of test result data against defined conditions,

— using a data model for structured persisting of test results,

— summarizing of test results in groups according to the data model,

— defining different test result levels (e.g. test results exclusively for debugging tasks),

— multilingual test result texts,

— simultaneous working with several test instances (e.g. several vehicles).

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

Similar to the OTX Logging extension, the OTX TestResultHandling extension uses so called **TestResultSeverities** which are associated to test results, and **TestResultHandlingLevels** which can be set in the test result handling framework. Depending on the currently set **TestResultHandlingLevel** and the **TestResultSeverity** of a test result, the test result gets tracked or is discarded. For that reason, a **TestResultHandlingLevel** represents a certain threshold which has to be exceeded by the **TestResultSeverity** of a test result in order to be written into the persistence layer. Initially, the **TestResultHandlingLevel** shall be ALL.

Each test result has a **TestResultState** which indicates whether the evaluation of the test result against the defined condition was successful.

24.2 Datatypes

24.2.1 Datatypes overview

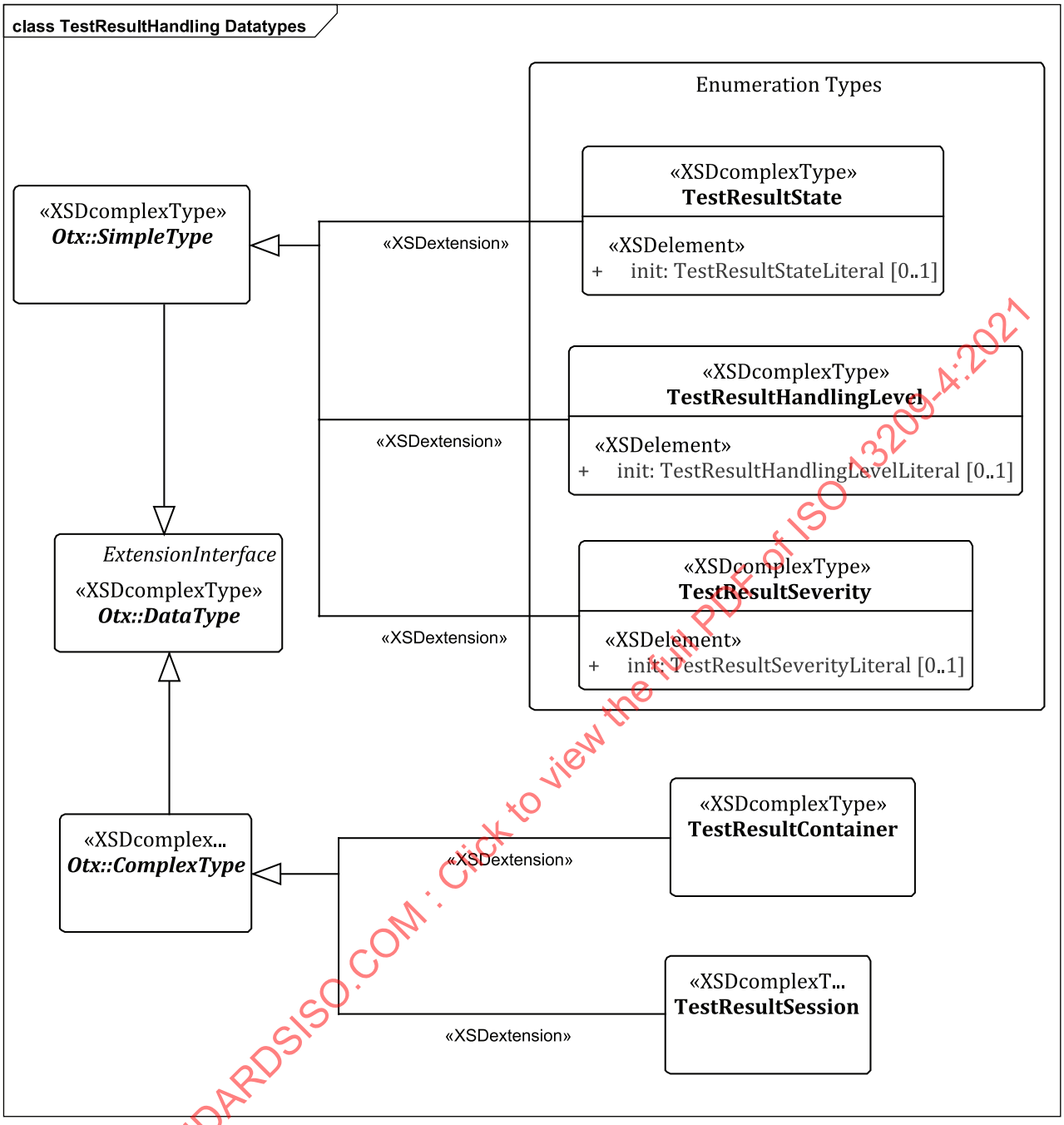
All data types introduced by this extension are derived from **otx:ComplexType** or **otx:SimpleType**.

24.2.2 Datatypes syntax

The syntax of the data-type declarations of the TestResultHandling extension is shown in Figure 100.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved



STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

Figure 100 — TestResultHandling datatypes

24.2.3 Datatypes semantics

24.2.3.1 TestResultContainer

TestResultContainer is an **otx:ComplexType**.

A **TestResultContainer** represents a node in the test result data model and is defined by its path.

24.2.3.2 TestResultHandlingLevel

TestResultHandlingLevel is an **otx:SimpleType**.

© ISO 2021 – All rights reserved

TestResultHandlingLevel is an enumeration type describing test result handling thresholds used by the **SetTestResultHandlingLevel** action. The allowed values are specified in the **TestResultHandlingLevels** enumeration.

IMPORTANT — TestResultHandlingLevel values may occur as operands of comparisons (see ISO 13209-2:—, 7.15.8). For this case, the following order relation shall apply: ALL < TRACK < OFF.

IMPORTANT — When applying otx:ToString on a TestResultHandlingLevel value, the resulting string shall be the name of the enumeration value, e.g. otx:ToString(TRACK)= "TRACK". Furthermore, applying otx:ToInteger shall return the index of the value in the TestResultHandlingLevels enumeration (smallest index is 0). The behaviour is undefined for other conversion terms (see ISO 13209-2).

Its members have the following semantics:

— <init> : testResult:TestResultHandlingLevelLiteral[0..1]

  This optional element stands for the hard-coded initialisation value of the identifier at declaration time.

IMPORTANT — If the TestResultHandlingLevel declaration is not explicitly initialised (omitted <init> element), the default value shall be ALL.

24.2.3.3 TestResultSession

TestResultSession is an **otx:ComplexType**.

A **TestResultSession** represents a test instance and is defined by a test object and a test identifier.

24.2.3.4 TestResultSeverity

TestResultSeverity is an **otx:SimpleType**.

TestResultSeverity is an enumeration type describing the severity of a test result. The allowed values are specified in the **TestResultSeverities** enumeration.

IMPORTANT — TestResultSeverity values may occur as operands of comparisons (see ISO 13209-2:—, 7.15.8). For this case, the following order relation shall apply: DEBUG < TRACK.

IMPORTANT — When applying otx:ToString on a TestResultSeverity value, the resulting string shall be the name of the enumeration value, e.g. otx:ToString(TRACK)="TRACK". Furthermore, applying otx:ToInteger shall return the index of the value in the TestResultSeveritys enumeration (smallest index is 0). The behaviour is undefined for other conversion terms (see ISO 13209-2).

Its members have the following semantics:

— <init> : testResult:TestResultSeverityLiteral[0..1]

  This optional element stands for the hard-coded initialisation value of the identifier at declaration time.

IMPORTANT — If the TestResultSeverity declaration is not explicitly initialised (omitted <init> element), the default value shall be TRACK.

24.2.3.5 TestResultState

TestResultState is an **otx:SimpleType**.

TestResultState is an enumeration type describing the state of the test result evaluation. The allowed values are specified in the **TestResultStates** enumeration.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

IMPORTANT — TestResultState values may occur as operands of comparisons (see ISO 13209-2:—, 7.15.8). For this case, the following order relation shall apply: NOT_TESTED < ERROR < NOT_OK < OK < INFO < NOT_AVAILABLE.

IMPORTANT — When applying otx:ToString on a TestResultState value, the resulting string shall be the name of the enumeration value, e.g. otx:ToString(NOT_OK)="NOT_OK". Furthermore, applying otx:ToInteger shall return the index of the value in the TestResultStates enumeration (smallest index is 0). The behaviour is undefined for other conversion terms (see ISO 13209-2).

All TestResultContainers in the TestResultHandling data model shall have a defined **TestResultState** with NOT_TESTED as the initial value.

It is possible, that there are TestResultContainers, which do not have their own test result but consist of child TestResultContainers. In this case the **TestResultContainer** gets the highest-order **TestResultState** of its children. This rule leads to the following behaviour.

The parent **TestResultContainer**'s **TestResultState** shall reflect the most critical **TestResultState** of its child **TestResultContainer**s **TestResultStates**. For example, the parent **TestResultContainer TestResultState** is OK, if at least one child **TestResultContainer TestResultState** is OK and no one is NOT_OK or ERROR or NOT_TESTED.

IMPORTANT — If a parent TestResultContainer gets assigned a TestResultState by the SetTestResultState action, the inherited TestResultState from its children will be overridden.

IMPORTANT — If a TestResultContainer cannot be tested (e.g. because of a low equipped vehicle), the TestResultState of the highest-order untestable TestResultContainer should be set manually by the SetTestResultState action.

Its members have the following semantics:

— <init> : testResult:TestResultStateLiteral[0..1]

  This optional element stands for the hard-coded initialisation value of the identifier at declaration time.

  This attribute shall contain one of the values defined in the **TestResultStates** enumeration.

IMPORTANT — If the TestResultState declaration is not explicitly initialised (omitted <init> element), the default value shall be NOT_TESTED.

24.3 Enumerations

24.3.1 Enumerations syntax

The syntax of the enumeration declarations of the TestResultHandling extension is shown in Figure 101.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

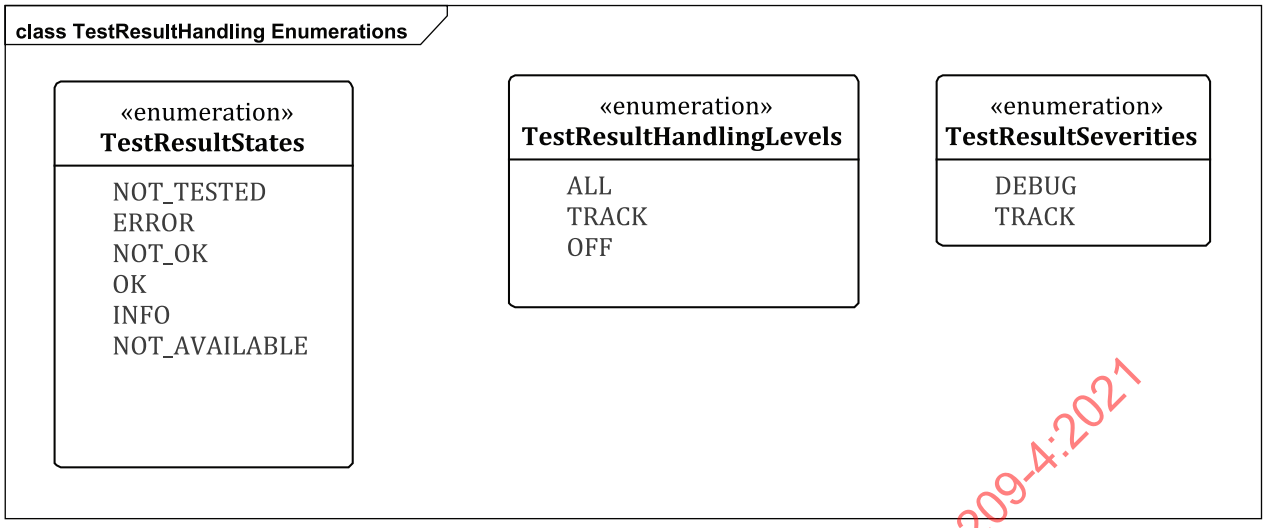


Figure 101 — TestResultHandling enumerations

24.3.2 Enumerations semantics

24.3.2.1 TestResultHandlingLevels

TestResultHandlingLevels has the following enumeration values:

— ALL

Test results with severity DEBUG or TRACK will be captured.

— TRACK

Test results with severity TRACK will be captured.

— OFF

No test results will be captured.

24.3.2.2 TestResultSeverities

TestResultSeverities has the following enumeration values:

— DEBUG

Temporary test result used only for debugging tasks.

— TRACK

It is the necessary test result for the user-defined use cases.

24.3.2.3 TestResultStates

TestResultStates has the following enumeration values:

— NOT_TESTED

The **TestResultContainer** has not been tested yet. This is the most critical state.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

— ERROR

An error has occurred while capturing or evaluating the test result, e.g. evaluation could not take place because a diagService gave a negative response.

— NOT_OK

The test result evaluation failed.

— OK

The test result evaluation succeeded.

— INFO

The test result is only for documentation purposes without an evaluation.

— NOT_AVAILABLE

The test result evaluation was not performed because the feature is missing. This is the least critical state.

24.4 Exceptions

24.4.1 Exceptions overview

All exceptions introduced by this extension are derived from the **otx:Exception** type. They represent the full set of exceptions added by the TestResultHandling extension.

24.4.2 Exceptions syntax

The syntax of all TestResultHandling exception-type declarations is shown in Figure 102.

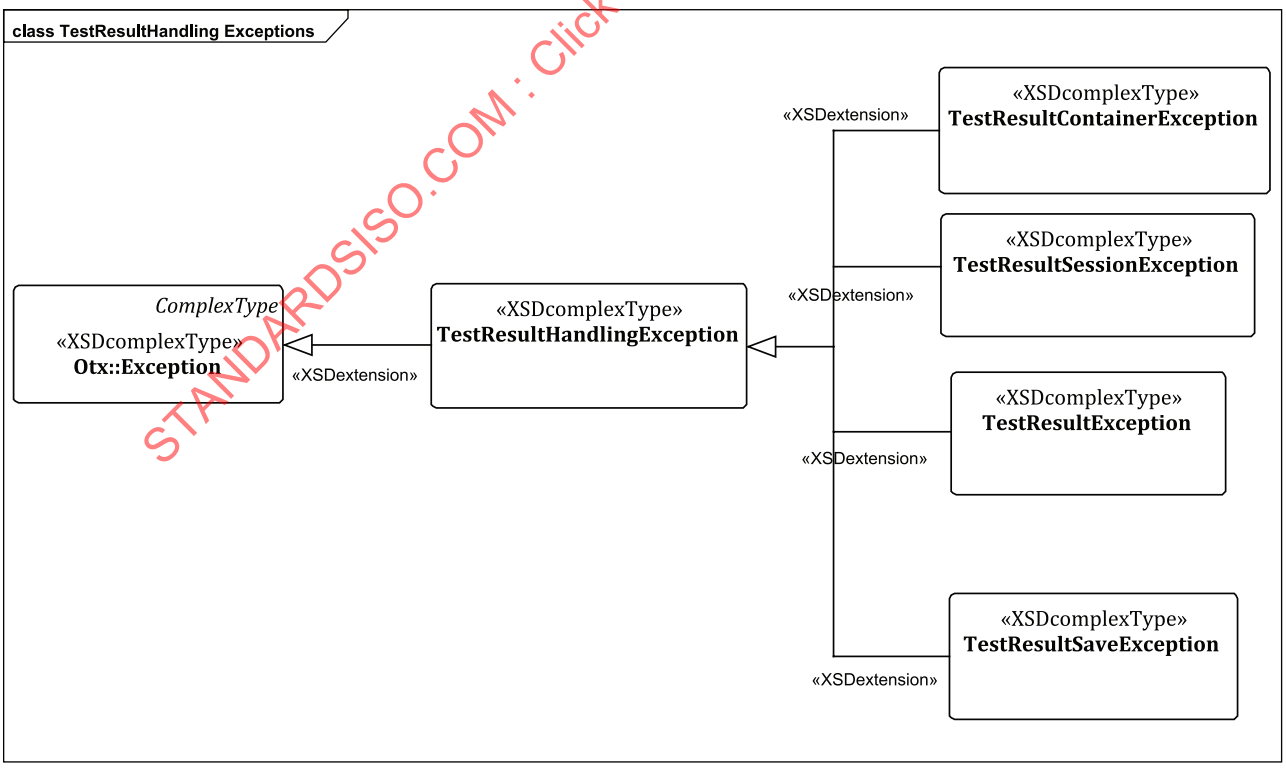


Figure 102 — TestResultHandling exceptions

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved

24.4.3 Exceptions semantics

24.4.3.1 General

Since all TestResultHandling exception types are implicit exceptions without initialisation parts, they cannot be declared constant.

24.4.3.2 TestResultContainerException

This exception is thrown if the **TestResultContainer** is not available for the operation, e.g. if a parent **TestResultContainer** should get an own test result by a **SetTestResult** action.

24.4.3.3 TestResultException

This exception is thrown if the test result cannot be set.

24.4.3.4 TestResultHandlingException

The **TestResultHandlingException** is the super class for all exceptions in the TestResultHandling extension. A **TestResultHandlingException** shall be used in case the more specific exception types described in the remainder of this section do not apply to the problem at hand.

IMPORTANT — All terms and action realisations in this extension may potentially throw this exception.

24.4.3.5 TestResultSaveException

The exception that is thrown if the changes within the **TestResultSession** cannot be written into the persistence layer.

24.4.3.6 TestResultSessionException

The exception that is thrown if the **TestResultSession** is not available.

24.5 Variable access

24.5.1 Variable-access overview

All variable-access types introduced by this extension are derived from the OTX core variable extension interface.

24.5.2 Variable-access syntax

The syntax of the TestResultHandling extension variable-access types is shown in Figure 103.

STANDARDSISO.COM : Click to view the full PDF of ISO 13209-4:2021

© ISO 2021 – All rights reserved