|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Document Title** | | | | Specification of Manifest | |
| **Document Owner** | | | | AUTOSAR | |
| **Document Responsibility** | | | | AUTOSAR | |
| **Document Identification No** | | | | 713 | |
| **Document Status** | | | | published | |
| **Part of AUTOSAR Standard** | | | | Adaptive Platform | |
| **Part of Standard Release** | | | | R21-11 | |
|  | | | | | |
| **Document Change History** | | | | | |
|  | | | | | |
| **Date** | **Release** | **Changed by** | | **Description** | |
| 2021-11-25 | R21-11 | AUTOSAR  Release  Management | | * Removal of REST support * Rework of signal-to-service conversion * Rework of raw data stream configuration * Rework of diagnostic mapping * minor corrections / clarifications / editorial changes | |
| 2020-11-30 | R20-11 | AUTOSAR  Release  Management | | * Remodeling of Phm contribution * Reporting of Security Events * Support for cryptographic Operations * Remodeling of Diagnostic Mapping * minor corrections / clarifications / editorial changes | |
| 2019-11-28 | R19-11 | AUTOSAR  Release  Management | | * Overhaul of Signal-to-Service Translation * Support for Raw Data Streams * Support for Vehicle Package * Support for Service Versioning * Changed Document Status from Final to published | |

|  |  |  |  |
| --- | --- | --- | --- |
| 2019-03-29 | 19-03 | AUTOSAR  Release  Management | * Introduction of Diagnostic Port Interfaces * Overhaul of Software Cluster and introduction of Software Package * Support for Identity and Access Management * Network Management Configuration |
| 2018-10-31 | 18-10 | AUTOSAR  Release  Management | * Finish introduction of   CppImplementationDataType   * Support for optional elements in structures * Rework configuration of adaptive platform modules |
| 2018-03-29 | 18-03 | AUTOSAR  Release  Management | * Time Synchronization * DDS Deployment |
| 2017-10-27 | 17-10 | AUTOSAR  Release  Management | * Optional elements in Service Interfaces * Interaction with web services * Secure Communication * Support for interaction with crypto and persistency * Signal-to-Service translation * Support for E2E communication * Platform Health Management * Uploadable Software Package |
| 2017-03-31 | 17-03 | AUTOSAR  Release  Management | • Initial release |

**Disclaimer**

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

# Table of Contents

1. Introduction 18
   1. Modeling Approach . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
   2. The Term Service . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
   3. Terms and Abbreviations . . . . . . . . . . . . . . . . . . . . . . . . . . 20
   4. Document Conventions . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
   5. Requirements Tracing . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
2. Big Picture of Manifest Definition 29
   1. Design vs. Deployment . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
      1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
      2. Relation between Design and Deployment Models . . . . . . 29
      3. Structure of the document . . . . . . . . . . . . . . . . . . . . 31
   2. About Manifest . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
   3. Serialization Format . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
   4. Scope . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
   5. Manifests described in this Document . . . . . . . . . . . . . . . . . . 33
3. Application Design 35
   1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
   2. Software Component . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
   3. Data Type . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
      1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
      2. ApplicationDataType . . . . . . . . . . . . . . . . . . . . . . . 37
         1. String Data Type . . . . . . . . . . . . . . . . . . . . 38
         2. Associative Map Data Type . . . . . . . . . . . . . . 40
         3. Attributes of SwDataDefProps . . . . . . . . . . . . . 46
      3. CppImplementationDataType . . . . . . . . . . . . . . . . . . 47
         1. Overview . . . . . . . . . . . . . . . . . . . . . . . . 47
         2. Attributes of SwDataDefProps . . . . . . . . . . . . . 61
         3. Primitive Data Types . . . . . . . . . . . . . . . . . . 63
         4. String Data Type . . . . . . . . . . . . . . . . . . . . 63
         5. Array Data Type . . . . . . . . . . . . . . . . . . . . . 64
         6. Vector Data Type . . . . . . . . . . . . . . . . . . . . 66
         7. Struct Data Type . . . . . . . . . . . . . . . . . . . . 69
         8. Enumeration Data Type . . . . . . . . . . . . . . . . 70
         9. Map Data Type . . . . . . . . . . . . . . . . . . . . . 72
         10. Variant Data Type . . . . . . . . . . . . . . . . . . . . 73
         11. Bitfield Data Type . . . . . . . . . . . . . . . . . . . . 74
      4. Compatibility of ApplicationDataType and CppImplementa-

tionDataType . . . . . . . . . . . . . . . . . . . . . . . . . . . 75

* 1. Service Interface . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
     1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
     2. Event . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
     3. Trigger . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 81
     4. Field . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 81
     5. Method . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 83
        1. Fire and Forget Method . . . . . . . . . . . . . . . . 84
     6. Versioning of ServiceInterfaces . . . . . . . . . . . . . . 85
        1. Versioning driven by transport layer . . . . . . . . . . 86
     7. Namespace . . . . . . . . . . . . . . . . . . . . . . . . . . . 87
     8. Error Handling . . . . . . . . . . . . . . . . . . . . . . . . . . 90
     9. Service Interface Data Type Mapping . . . . . . . . . . . . . 93
     10. Communication Group pattern . . . . . . . . . . . . . . . . . 95
  2. Service Interface Mapping . . . . . . . . . . . . . . . . . . . . . . . . . 96
  3. Service Interface Element Mapping . . . . . . . . . . . . . . . . . . . . 102
     1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 102
     2. Service Interface Event Mapping . . . . . . . . . . . . . . . . 106
     3. Service Interface Trigger Mapping . . . . . . . . . . . . . . . 107 3.6.4 Service Interface Field Mapping . . . . . . . . . . . . . . . . 108 3.6.5 Service Interface Method Mapping . . . . . . . . . . . . . . . 109
  4. Persistency Interface . . . . . . . . . . . . . . . . . . . . . . . . . . . . 111
     1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 111
        1. The big Picture . . . . . . . . . . . . . . . . . . . . . 111
        2. Modeling of Persistency Interface . . . . . . . . . . . 112
        3. Redundancy Handling . . . . . . . . . . . . . . . . . 113
        4. Update Handling . . . . . . . . . . . . . . . . . . . . 118
     2. Persistency Key Value Storage Interface . . . . . . . . . . . 119
     3. Persistency File Storage Interface . . . . . . . . . . . . . . . 121
  5. Time Synchronization Interface . . . . . . . . . . . . . . . . . . . . . . 123
  6. Platform Health Management Interface . . . . . . . . . . . . . . . . . . 126
     1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 126 3.9.2 Supervised Entities and Checkpoints . . . . . . . . . . . . . 127
     2. Health Channels . . . . . . . . . . . . . . . . . . . . . . . . . 129
     3. Recovery notification to State Management . . . . . . . . . . 131
  7. Diagnostic Interface . . . . . . . . . . . . . . . . . . . . . . . . . . . . 134
     1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 134 3.10.2 Diagnostic Routine Interface . . . . . . . . . . . . . . . . . . 136

3.10.3 Interface to Data Identifier and Element of Data Identifier . . 139 3.10.4 Interface to diagnostic Events . . . . . . . . . . . . . . . . . 142 3.10.5 Interface to diagnostic Condition . . . . . . . . . . . . . . . . 143 3.10.6 Indicator Interface . . . . . . . . . . . . . . . . . . . . . . . . 144

* + 1. Security Level Interface . . . . . . . . . . . . . . . . . . . . . 145
    2. Service Validation Interface . . . . . . . . . . . . . . . . . . . 145 3.10.9 Operation Cycle Interface . . . . . . . . . . . . . . . . . . . . 146
    3. Generic UDS Interface . . . . . . . . . . . . . . . . . . . . . 147
    4. DoIP Interfaces . . . . . . . . . . . . . . . . . . . . . . . . . 148 3.10.12 Diagnostic Interfaces for Upload and Download . . . . . . . 150

3.10.13 Interface to support managing the EcuReset . . . . . . . . . 151 3.10.14 Diagnostic Authentication Interface . . . . . . . . . . . . . . 152

* + 1. Diagnostic External Authentication Interface . . . . . . . . . 152
    2. Diagnostic Communication Control Interface . . . . . . . . . 153
  1. Crypto Interfaces . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 154
     1. Interaction with Crypto Software . . . . . . . . . . . . . . . . 154
     2. Crypto Key Slot Interface . . . . . . . . . . . . . . . . . . . . 155
     3. Crypto Certificate Interface . . . . . . . . . . . . . . . . . . . 160
     4. Crypto Provider Interface . . . . . . . . . . . . . . . . . . . . 160
     5. Crypto TrustMaster Interface . . . . . . . . . . . . . . . . . . 161
     6. Linking of Crypto Certificate to a Crypto Key Slot . . . . . . . 161
  2. Raw Data Stream Interface . . . . . . . . . . . . . . . . . . . . . . . . 164
  3. Security Event Report Interface . . . . . . . . . . . . . . . . . . . . . . 165
  4. Log And Trace Interface . . . . . . . . . . . . . . . . . . . . . . . . . . 167
  5. Interaction Endpoint for Application . . . . . . . . . . . . . . . . . . . . 168
     1. Service-oriented Communication . . . . . . . . . . . . . . . . 168
     2. Interaction with Persistent Key-Value Storage . . . . . . . . . 169 3.15.3 Interaction with Persistent File Storage . . . . . . . . . . . . 170
     3. Port Prototype Props . . . . . . . . . . . . . . . . . . . . . . 170
     4. Port Prototype ComSpec . . . . . . . . . . . . . . . . . . . . 172
        1. Port Prototypes typed by Service Interfaces . . . . . 174
        2. Port Prototypes typed by Persistency Data Interfaces 183
  6. Executable . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 184
  7. Optional Members in complex Data Structures . . . . . . . . . . . . . . 190
     1. Background . . . . . . . . . . . . . . . . . . . . . . . . . . . 190
     2. Definition of Optionality . . . . . . . . . . . . . . . . . . . . . 191
  8. Serialization Properties . . . . . . . . . . . . . . . . . . . . . . . . . . . 193
     1. Default Values for Serialization Properties . . . . . . . . . . . 194
     2. Individual Definition of Serialization Properties . . . . . . . . 199
     3. Assignment of TLV properties . . . . . . . . . . . . . . . . . 205
        1. Assignment of TLV Data IDs . . . . . . . . . . . . . . 205
        2. Assignment of Wire Type Selection . . . . . . . . . . 211
  9. Process Design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 211
     1. Deterministic Client Resource . . . . . . . . . . . . . . . . . 213
  10. Grant Design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 216
      1. Com Grant Design . . . . . . . . . . . . . . . . . . . . . . . . 217
      2. Grant Design for Raw Streaming Data . . . . . . . . . . . . . 223
      3. Remote access control . . . . . . . . . . . . . . . . . . . . . 224
         1. Remote subject in case of TLS . . . . . . . . . . . . 226 3.20.3.2 Remote subject in case of IPsec . . . . . . . . . . . 228
         2. Remote subject in case of IP communication . . . . 230
         3. Remote subject in case of SecOC communication . 231

1. Diagnostic Design 232
   1. Diagnostic Mapping . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 232
      1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 232
      2. Diagnostic Monitor to Port Mapping . . . . . . . . . . . . . . 235
      3. Diagnostic Event to Port Mapping . . . . . . . . . . . . . . . 238
      4. Diagnostic Operation Cycle to Port Mapping . . . . . . . . . 240
      5. Diagnostic Enable Condition to Port Mapping . . . . . . . . . 242 4.1.6 Diagnostic Clear Condition to Port Mapping . . . . . . . . . . 244
      6. Diagnostic Indicator to Port Mapping . . . . . . . . . . . . . . 246
      7. Diagnostic Memory Destination to Port Mapping . . . . . . . 248
      8. Diagnostic Security to Port Mapping . . . . . . . . . . . . . . 250
      9. Diagnostic Data Identifier to Port Mapping . . . . . . . . . . . 252 4.1.11 Diagnostic Generic UDS Service Handler to Port Mapping . 256
      10. Diagnostic Generic Mapping . . . . . . . . . . . . . . . . . . 256
      11. Diagnostic Service Validation Mapping . . . . . . . . . . . . 260 4.1.14 Diagnostic Authentication Mapping . . . . . . . . . . . . . . . 261

4.1.15 Diagnostic External Authentication Mapping . . . . . . . . . 264

* 1. Diagnostic Clear Condition . . . . . . . . . . . . . . . . . . . . . . . . . 266
  2. Security Access . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 267
  3. DiagnosticProvidedDataMapping . . . . . . . . . . . . . . . . . . . . . 268
  4. Diagnostic Miscellanea . . . . . . . . . . . . . . . . . . . . . . . . . . . 269

1. System Design 270
   1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 270
   2. Specification of Communication System Structure . . . . . . . . . . . . 272
      1. Network connection . . . . . . . . . . . . . . . . . . . . . . . 275
         1. Support of 10BASE-T1S Network Topologies . . . . 280
      2. TcpIp stack configuration properties . . . . . . . . . . . . . . 280
         1. IP configuration properties . . . . . . . . . . . . . . . 281 5.2.2.2 TCP and UDP configuration properties . . . . . . . . 287

5.2.2.3 ICMP configuration properties . . . . . . . . . . . . . 289

* + 1. Securing Communication with IPsec . . . . . . . . . . . . . . 290
    2. Service Discovery Configuration . . . . . . . . . . . . . . . . 298
       1. SOME/IP Service Discovery Configuration . . . . . . 298
    3. Partial Network . . . . . . . . . . . . . . . . . . . . . . . . . . 300
  1. Specification of Application Software System Structure . . . . . . . . . 302
  2. Modeling of service oriented communication between Classic and

Adaptive platform . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 304 5.4.1 MethodMapping . . . . . . . . . . . . . . . . . . . . . . . . . 306 5.4.2 EventMapping . . . . . . . . . . . . . . . . . . . . . . . . . . 306

* + 1. FieldMapping . . . . . . . . . . . . . . . . . . . . . . . . . . . 307
    2. FireAndForgetMapping . . . . . . . . . . . . . . . . . . . . . 309

1. Sub-System Design 311
   1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 311 6.2 Software Cluster Design . . . . . . . . . . . . . . . . . . . . . . . . . . 311 6.3 Provided and required Services of Software Cluster Design . . . . . . 317

6.4 Mapping of Services to Executables . . . . . . . . . . . . . . . . . . . 320

1. Machine Manifest 324
   1. Process To Machine Mapping . . . . . . . . . . . . . . . . . . . . . . . 328
      1. General Modeling Approach . . . . . . . . . . . . . . . . . . 328
      2. Core Affinity . . . . . . . . . . . . . . . . . . . . . . . . . . . 330
      3. Default Start-up and Termination Timeout . . . . . . . . . . . 331
2. Execution Manifest 333
   1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 333
   2. Startup Configuration . . . . . . . . . . . . . . . . . . . . . . . . . . . . 336
      1. State-dependent Startup Configuration . . . . . . . . . . . . 337
      2. Scheduling . . . . . . . . . . . . . . . . . . . . . . . . . . . . 340
      3. Process Arguments . . . . . . . . . . . . . . . . . . . . . . . 341
      4. Association with Resource Group . . . . . . . . . . . . . . . 341
      5. Execution Dependency . . . . . . . . . . . . . . . . . . . . . 342
      6. Assignment of Processes to Function Group states . . . . . 344
      7. Resource Consumption Boundaries . . . . . . . . . . . . . . 345
      8. Error and Termination Behavior . . . . . . . . . . . . . . . . . 347
   3. Deterministic Client . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 348
   4. Function Groups . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 349
      1. Semantics of Function Group . . . . . . . . . . . . . . . . . . 349
      2. Machine Function Group . . . . . . . . . . . . . . . . . . . . 352
   5. Reporting of Security Events . . . . . . . . . . . . . . . . . . . . . . . . 353
3. Platform Module Development 354
   1. OS Module configuration . . . . . . . . . . . . . . . . . . . . . . . . . . 356
   2. Persistency Deployment . . . . . . . . . . . . . . . . . . . . . . . . . . 358
      1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 358
         1. Redundancy Handling . . . . . . . . . . . . . . . . . 361
         2. Update Handling . . . . . . . . . . . . . . . . . . . . 362
         3. Size Handling . . . . . . . . . . . . . . . . . . . . . . 363
         4. Security Handling . . . . . . . . . . . . . . . . . . . . 364
      2. Deployment of Persistent Key-Value Storage . . . . . . . . . 364
      3. Deployment of File Storage . . . . . . . . . . . . . . . . . . . 368
   3. Platform Health Management Deployment . . . . . . . . . . . . . . . . 371
      1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 371
      2. Relation between design and deployment . . . . . . . . . . . 374
      3. Supervision deployment . . . . . . . . . . . . . . . . . . . . . 375
         1. Global supervision . . . . . . . . . . . . . . . . . . . 378
         2. Supervision Mode . . . . . . . . . . . . . . . . . . . 381
         3. Supervision Mode condition . . . . . . . . . . . . . . 383
         4. NoSupervision definition . . . . . . . . . . . . . . 386
         5. AliveSupervision definition . . . . . . . . . . . . 387
         6. CheckpointTransition definition . . . . . . . . . 389
         7. LogicalSupervision definition . . . . . . . . . . 390
         8. DeadlineSupervision definition . . . . . . . . . . 391
      4. Health channel deployment . . . . . . . . . . . . . . . . . . . 392
         1. Supervision health channel deployment . . . . . . . 393
         2. External health channel deployment . . . . . . . . . 393
      5. Recovery Notification . . . . . . . . . . . . . . . . . . . . . . 396
   4. Time Synchronization Deployment . . . . . . . . . . . . . . . . . . . . 397
      1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 397
      2. Time Synchronization functional cluster configuration . . . . 398
      3. Time Base . . . . . . . . . . . . . . . . . . . . . . . . . . . . 399
         1. Synchronized time base . . . . . . . . . . . . . . . . 400
         2. Persistent Time Base value storage . . . . . . . . . . 403
         3. Ethernet synchronized time . . . . . . . . . . . . . . 404
      4. Time Base to Port Prototype mapping . . . . . . . . . . . . . 406
   5. DoIP configuration . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 408
   6. Log and Trace module configuration . . . . . . . . . . . . . . . . . . . 414
      1. Log and trace deployment . . . . . . . . . . . . . . . . . . . . 416
      2. Relation between design and deployment . . . . . . . . . . . 419
   7. Network Management configuration . . . . . . . . . . . . . . . . . . . . 422
      1. UdpNm configuration constraints . . . . . . . . . . . . . . . . 429
   8. Update and Configuration Management . . . . . . . . . . . . . . . . . 430
   9. IAM configuration . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 432
      1. Com Grant Deployment . . . . . . . . . . . . . . . . . . . . . 433
      2. Grant Deployment for Raw Streaming Data . . . . . . . . . . 438
      3. Remote access control . . . . . . . . . . . . . . . . . . . . . 439
   10. Crypto Deployment . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 441
       1. Crypto Provider . . . . . . . . . . . . . . . . . . . . . . . . . 442
       2. Crypto Key Slot . . . . . . . . . . . . . . . . . . . . . . . . . 444
       3. Crypto Certificate . . . . . . . . . . . . . . . . . . . . . . . . 446
   11. IdsM Deployment . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 448
       1. IdsM Instantiation . . . . . . . . . . . . . . . . . . . . . . . . 448
       2. Obtaining custom Time Stamps for Security Events . . . . . 449
       3. Deployment for Security Events . . . . . . . . . . . . . . . . 450
4. Service Instance Manifest 452
   1. ApplicationEndpoint . . . . . . . . . . . . . . . . . . . . . . . . . . . . 452
      1. VLAN Priority . . . . . . . . . . . . . . . . . . . . . . . . . . . 454
   2. Service Interface Deployment . . . . . . . . . . . . . . . . . . . . . . . 455
      1. SOME/IP Service Interface Deployment . . . . . . . . . . . . 460
      2. DDS Service Interface Deployment . . . . . . . . . . . . . . 469
      3. User Defined Service Interface . . . . . . . . . . . . . . . . . 474
   3. Service Instance Deployment . . . . . . . . . . . . . . . . . . . . . . . 478
      1. SOME/IP Service Instance Deployment . . . . . . . . . . . . 484
         1. Provided Service Instance . . . . . . . . . . . . . . . 485
         2. Required Service Instance . . . . . . . . . . . . . . . 506
      2. DDS Service Instance Deployment . . . . . . . . . . . . . . . 517
         1. Provided DDS Service Instance . . . . . . . . . . . . 518
         2. Required DDS Service Instance . . . . . . . . . . . . 522
         3. DDS Service Instance to Machine mapping . . . . . 524
      3. User Defined Service Instance Deployment . . . . . . . . . . 525
      4. Service Deployment Relations . . . . . . . . . . . . . . . . . 527
   4. EndToEndProtection . . . . . . . . . . . . . . . . . . . . . . . . . . . . 528
   5. Secure Communication . . . . . . . . . . . . . . . . . . . . . . . . . . 536
      1. Secure Communication over TLS . . . . . . . . . . . . . . . 540
      2. Secure Communication over SecOC . . . . . . . . . . . . . . 548
      3. Secure Communication over DDS . . . . . . . . . . . . . . . 551
5. Raw Data Stream Manifest 555
   1. Raw Data Stream Deployment . . . . . . . . . . . . . . . . . . . . . . . 555
   2. Raw Data Stream Mapping . . . . . . . . . . . . . . . . . . . . . . . . 555
   3. Raw Data Streams over Ethernet . . . . . . . . . . . . . . . . . . . . . 556
      1. Ethernet Raw Data Stream Client Mapping . . . . . . . . . . 561 11.3.2 Ethernet Raw Data Stream Server Mapping . . . . . . . . . . 562

11.3.3 Ethernet Raw Data Stream Configuration Examples . . . . . 564

* + - 1. Preamble . . . . . . . . . . . . . . . . . . . . . . . . 564
      2. Fully formalized Model Example . . . . . . . . . . . 566 11.3.3.3 Only Client is formalized . . . . . . . . . . . . . . . . 568

11.3.3.4 Only Server is formalized . . . . . . . . . . . . . . . 568

1. Signal-based communication 570
   1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 570
   2. Signal-based prerequisites . . . . . . . . . . . . . . . . . . . . . . . . . 574
      1. Message Header . . . . . . . . . . . . . . . . . . . . . . . . . 575
   3. Signal-based Deployment . . . . . . . . . . . . . . . . . . . . . . . . . 577
   4. Signal/Service Translation Mapping . . . . . . . . . . . . . . . . . . . . 578
      1. Direction of Signal/Service Translation Mapping . . . . . . . 578
      2. SignalBasedEvent Mapping . . . . . . . . . . . . . . . . . . . 580
      3. SignalBasedTrigger Mapping . . . . . . . . . . . . . . . . . . 585
      4. SignalBasedField Mapping . . . . . . . . . . . . . . . . . . . 586
      5. SignalBasedMethod Mapping . . . . . . . . . . . . . . . . . . 590
   5. Service Discovery and Static Communication Configuration . . . . . . 592
      1. Service discovery control . . . . . . . . . . . . . . . . . . . . 593 12.5.2 Service control right after translation start . . . . . . . . . . . 595

12.5.3 Service control due to availability of related service instance 595

* + - 1. Signal-service-translation . . . . . . . . . . . . . . . 596 12.5.3.2 Service-signal-translation . . . . . . . . . . . . . . . 596

12.5.4 Static Communication Configuration . . . . . . . . . . . . . . 596

* + - 1. Static Communication Configuration for the Server . 596
      2. Static Communication Configuration for the Client . . 597
  1. Translation behavior . . . . . . . . . . . . . . . . . . . . . . . . . . . . 597
     1. ServiceInterface representation of translation data . . . 598
        1. Optional elements in case signal-servicetranslation . . . . . . . . . . . . . . . . . . . . . 598
        2. Optional elements in case service-signaltranslation . . . . . . . . . . . . . . . . . . . . . 599
     2. Event and update-bit semantics . . . . . . . . . . . . . . . . 600
     3. Data filtering . . . . . . . . . . . . . . . . . . . . . . . . . . . 603 12.6.4 Translation from one source . . . . . . . . . . . . . . . . . . 605

12.6.5 Translation from several sources . . . . . . . . . . . . . . . . 606

* 1. Translation pass-through composition . . . . . . . . . . . . . . . . . . . 607
  2. Expected features of Classic platform . . . . . . . . . . . . . . . . . . . 608
     1. Processing order . . . . . . . . . . . . . . . . . . . . . . . . . 608
     2. Reception data filter . . . . . . . . . . . . . . . . . . . . . . . 608
     3. Reception of invalid signal . . . . . . . . . . . . . . . . . . . 608
     4. Update Bit handling . . . . . . . . . . . . . . . . . . . . . . . 609 12.8.5 Transfer properties and transmission modes for Service-

Signal-Translation . . . . . . . . . . . . . . . . . . . . . . . . 609

* + 1. Deadline monitoring . . . . . . . . . . . . . . . . . . . . . . . 611
    2. Signal and IPdu Transmission . . . . . . . . . . . . . . . . . 611
    3. IPdu multiplexing . . . . . . . . . . . . . . . . . . . . . . . . . 611
  1. End-to-End considerations . . . . . . . . . . . . . . . . . . . . . . . . . 613
     1. Safety . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 613
        1. Signal-service-translation . . . . . . . . . . . . . . . 616
        2. Service-signal-translation . . . . . . . . . . . . . . . 616
     2. Security . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 617
        1. Restrictions on Classic Platform SecOC . . . . . . . 619
        2. Link between Classic Platform SecOC and Cryp-

toKeySlot . . . . . . . . . . . . . . . . . . . . . . . 619

1. Cross-FunctionalCluster interaction 621
   1. ComCertificateToCryptoCertificateMapping . . . . . . . . . . . . . . . 623
   2. ComKeyToCryptoKeySlotMapping . . . . . . . . . . . . . . . . . . . . . 624 13.3 ComSecOcToCryptoKeySlotMapping . . . . . . . . . . . . . . . . . . . 626
   3. PersistencyDeploymentToCryptoKeySlotMapping . . . . . . . . . . . . 628
   4. PersistencyDeploymentElementToCryptoKeySlotMapping . . . . . . . 629
   5. PersistencyDeploymentToDltLogSinkMapping . . . . . . . . . . . . . . 631
2. Software Distribution 632
   1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 632
   2. Software Cluster . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 633
      1. Software Cluster General Modeling . . . . . . . . . . . . . . 633
      2. Relevance of Software Cluster for Diagnostics . . . . . . . . 640
         1. Diagnostic Props . . . . . . . . . . . . . . . . . . . . 640
         2. Diagnostic Address . . . . . . . . . . . . . . . . . . . 642 14.2.2.3 Relation to the Diagnostic Contribution Set . . . . . 644

14.2.2.4 Diagnostic Service Validation . . . . . . . . . . . . . 645

* + 1. Sub Software Cluster . . . . . . . . . . . . . . . . . . . . . . 646
    2. Software Cluster Dependency . . . . . . . . . . . . . . . . . 647 14.2.5 References between Software Clusters . . . . . . . . . . . . 653 14.2.6 Software Cluster Artifact Checksum . . . . . . . . . . . . . . 654
  1. Software Package . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 656
  2. Vehicle Package . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 661
     1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 661 14.4.2 VehicleRolloutStep . . . . . . . . . . . . . . . . . . . . . . . . 664
     2. UcmStep . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 666
     3. SoftwarePackageStep . . . . . . . . . . . . . . . . . . . . . . 667
     4. Examples for the Usage of SoftwarePackageStep . . . . . . 669
        1. Examples for the Usage of transfer and process . . 669

1. Interoperability between Classic Platform and Adaptive Platform 673
   1. Usage of majorVersion in the SOME/IP network binding . . . . . . . . 673
2. Examples 675
   1. Service Instance Deployment by Service Interface Mapping . . . . . . 675
   2. Service Instance Deployment by Service Interface Element Mapping . 677
   3. Definition of Startup Configuration . . . . . . . . . . . . . . . . . . . . . 679
   4. Service Instance Mapping . . . . . . . . . . . . . . . . . . . . . . . . . 682
   5. Radar and Camera ServiceInterface example . . . . . . . . . . . . . . 685
   6. Definition of Persistent Data . . . . . . . . . . . . . . . . . . . . . . . . 690
   7. Definition of Persistent File . . . . . . . . . . . . . . . . . . . . . . . . . 691
   8. Definition of Phm interaction . . . . . . . . . . . . . . . . . . . . . . . . 692
      1. Phm Application Design example . . . . . . . . . . . . . . . 692
      2. Phm configuration example . . . . . . . . . . . . . . . . . . . 693
   9. Scenarios to define a Vector . . . . . . . . . . . . . . . . . . . . . . . . 695
3. Custom Model Extension 696
   1. Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 696
   2. Custom Attribute Definition . . . . . . . . . . . . . . . . . . . . . . . . . 698
      1. Custom Primitive Attribute Definition . . . . . . . . . . . . . . 698
      2. Custom Complex Attribute Definition . . . . . . . . . . . . . . 700
   3. Custom Foreign Reference Definition . . . . . . . . . . . . . . . . . . . 702
   4. Custom Subclass Configuration . . . . . . . . . . . . . . . . . . . . . . 705
   5. Custom Constraints . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 707
   6. Definition of Reference from SdgClass to SdgClass . . . . . . . . . . . 708
4. General Modeling 711
   1. Reference to a DataPrototype in a PortInterface . . . . . . . . . . . . . 711
      1. Reference to the inside of an ApplicationDataType . . . 711
      2. Reference to the inside of a CppImplementationDataType 714
   2. Reference to a AutosarDataPrototype in an Executable . . . . . . . . . 716
   3. Reference to a PortPrototype in an Executable . . . . . . . . . . . . . 719
   4. Modeling of a Method in an Executable . . . . . . . . . . . . . . . . . . 725
   5. Modeling of Mode-related InstanceRefs . . . . . . . . . . . . . . . . . 728
   6. Modeling of Diagnostic-related InstanceRefs . . . . . . . . . . . . . . . 730
   7. Modeling of PHM-related InstanceRefs . . . . . . . . . . . . . . . . . . 731
   8. Modeling of Time-related InstanceRefs . . . . . . . . . . . . . . . . . . 735
   9. Modeling of Persistency-related InstanceRefs . . . . . . . . . . . . . . 736
   10. Modeling of SoftwareClusterDesign-related InstanceRefs . . . . . . . . 736
5. Mentioned Class Tables 740
6. History of Constraints and Specification Items 806
   1. Constraint and Specification Item History of this document according

to AUTOSAR Release R17-03 (original version) . . . . . . . . . . . . . 806

* + 1. Created Constraints in R17-03 . . . . . . . . . . . . . . . . . 806
    2. Created Specification Items in R17-03 . . . . . . . . . . . . . 808
  1. Constraint and Specification Item History of this document according

to AUTOSAR Release R17-10 . . . . . . . . . . . . . . . . . . . . . . . 813

* + 1. Added Traceables in R17-10 . . . . . . . . . . . . . . . . . . 813
    2. Changed Traceables in R17-10 . . . . . . . . . . . . . . . . . 817
    3. Deleted Traceables in R17-10 . . . . . . . . . . . . . . . . . 818
    4. Added Constraints in R17-10 . . . . . . . . . . . . . . . . . . 818
    5. Changed Constraints in R17-10 . . . . . . . . . . . . . . . . 820
    6. Deleted Constraints in R17-10 . . . . . . . . . . . . . . . . . 820
  1. Constraint and Specification Item History of this document according

to AUTOSAR Release R18-03 . . . . . . . . . . . . . . . . . . . . . . . 820

* + 1. Added Traceables in R18-03 . . . . . . . . . . . . . . . . . . 820
    2. Changed Traceables in R18-03 . . . . . . . . . . . . . . . . . 824
    3. Deleted Traceables in R18-03 . . . . . . . . . . . . . . . . . 826
    4. Added Constraints in R18-03 . . . . . . . . . . . . . . . . . . 826
    5. Changed Constraints in R18-03 . . . . . . . . . . . . . . . . 829
    6. Deleted Constraints in R18-03 . . . . . . . . . . . . . . . . . 829
  1. Constraint and Specification Item History of this document according

to AUTOSAR Release R18-10 . . . . . . . . . . . . . . . . . . . . . . . 830

* + 1. Added Traceables in R18-10 . . . . . . . . . . . . . . . . . . 830
    2. Changed Traceables in R18-10 . . . . . . . . . . . . . . . . . 832
    3. Deleted Traceables in R18-10 . . . . . . . . . . . . . . . . . 834
    4. Added Constraints in R18-10 . . . . . . . . . . . . . . . . . . 836
    5. Changed Constraints in R18-10 . . . . . . . . . . . . . . . . 838
    6. Deleted Constraints in R18-10 . . . . . . . . . . . . . . . . . 839
  1. Constraint and Specification Item History of this document according

to AUTOSAR Release R19-03 . . . . . . . . . . . . . . . . . . . . . . . 840

* + 1. Added Traceables in R19-03 . . . . . . . . . . . . . . . . . . 840
    2. Changed Traceables in R19-03 . . . . . . . . . . . . . . . . . 842
    3. Deleted Traceables in R19-03 . . . . . . . . . . . . . . . . . 843
    4. Added Constraints in R19-03 . . . . . . . . . . . . . . . . . . 843
    5. Changed Constraints in R19-03 . . . . . . . . . . . . . . . . 845
    6. Deleted Constraints in R19-03 . . . . . . . . . . . . . . . . . 845
  1. Constraint and Specification Item History of this document according

to AUTOSAR Release R19-11 . . . . . . . . . . . . . . . . . . . . . . . 846

* + 1. Added Traceables in R19-11 . . . . . . . . . . . . . . . . . . 846
    2. Changed Traceables in R19-11 . . . . . . . . . . . . . . . . . 849
    3. Deleted Traceables in R19-11 . . . . . . . . . . . . . . . . . 850
    4. Added Constraints in R19-11 . . . . . . . . . . . . . . . . . . 850
    5. Changed Constraints in R19-11 . . . . . . . . . . . . . . . . 852
    6. Deleted Constraints in R19-11 . . . . . . . . . . . . . . . . . 853
  1. Constraint and Specification Item History of this document according

to AUTOSAR Release R20-11 . . . . . . . . . . . . . . . . . . . . . . . 853

* + 1. Added Traceables in R20-11 . . . . . . . . . . . . . . . . . . 853
    2. Changed Traceables in R20-11 . . . . . . . . . . . . . . . . . 856
    3. Deleted Traceables in R20-11 . . . . . . . . . . . . . . . . . 859
    4. Added Constraints in R20-11 . . . . . . . . . . . . . . . . . . 860
    5. Changed Constraints in R20-11 . . . . . . . . . . . . . . . . 862
    6. Deleted Constraints in R20-11 . . . . . . . . . . . . . . . . . 863
  1. Constraint and Specification Item History of this document according

to AUTOSAR Release R21-11 . . . . . . . . . . . . . . . . . . . . . . . 864

* + 1. Added Traceables in R21-11 . . . . . . . . . . . . . . . . . . 864
    2. Changed Traceables in R21-11 . . . . . . . . . . . . . . . . . 867
    3. Deleted Traceables in R21-11 . . . . . . . . . . . . . . . . . 869
    4. Added Constraints in R21-11 . . . . . . . . . . . . . . . . . . 870
    5. Changed Constraints in R21-11 . . . . . . . . . . . . . . . . 874
    6. Deleted Constraints in R21-11 . . . . . . . . . . . . . . . . . 875

1. Splitable Elements in the Scope of this Document 876
2. Variation Points in the Scope of this Document 878

# References

1. Software Component Template

AUTOSAR\_TPS\_SoftwareComponentTemplate

1. Layered Software Architecture

AUTOSAR\_EXP\_LayeredSoftwareArchitecture

1. Reference Model for Service Oriented Architecture 1.0 <https://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf>
2. Glossary

AUTOSAR\_TR\_Glossary

1. Standardization Template

AUTOSAR\_TPS\_StandardizationTemplate

1. Generic Structure Template

AUTOSAR\_TPS\_GenericStructureTemplate

1. SOME/IP Protocol Specification

AUTOSAR\_PRS\_SOMEIPProtocol

1. Specification of Communication Management

AUTOSAR\_SWS\_CommunicationManagement

1. Specification of Persistency

AUTOSAR\_SWS\_Persistency

1. IEEE Standard for Information Technology- Standardized Application Environment Profile (AEP)-POSIX Realtime and Embedded Application Support <https://standards.ieee.org/findstds/standard/1003.13-2003.html>
2. Specification of Time Synchronization

AUTOSAR\_SWS\_TimeSynchronization

1. Specification of Platform Health Management

AUTOSAR\_SWS\_PlatformHealthManagement

1. Specification of Cryptography

AUTOSAR\_SWS\_Cryptography

1. Specification of Identity and Access Management

AUTOSAR\_SWS\_IdentityAndAccessManagement

1. Explanation of ara::com API

AUTOSAR\_EXP\_ARAComAPI

1. Information technology – Universal Coded Character Set (UCS) [http://www.iso.org](http://www.iso.org/)
2. System Template

AUTOSAR\_TPS\_SystemTemplate

1. Specification of Execution Management

AUTOSAR\_SWS\_ExecutionManagement

1. Diagnostic Extract Template

AUTOSAR\_TPS\_DiagnosticExtractTemplate

1. Specification of Diagnostics

AUTOSAR\_SWS\_Diagnostics

1. Security Extract Template

AUTOSAR\_TPS\_SecurityExtractTemplate

1. Road vehicles – Diagnostic communication over Internet Protocol (DoIP) [http://www.iso.org](http://www.iso.org/)
2. Log and Trace Protocol Specification

AUTOSAR\_PRS\_LogAndTraceProtocol

1. SOME/IP Service Discovery Protocol Specification

AUTOSAR\_PRS\_SOMEIPServiceDiscoveryProtocol

1. Data Distribution Service (DDS), Version 1.4 <http://www.omg.org/spec/DDS/1.4>
2. RPC over DDS, Version 1.0 <https://www.omg.org/spec/DDS-RPC/1.0>
3. DDS Consolidated XML Syntax, Version 1.0 <https://www.omg.org/spec/DDS-XML>
4. Specification of SW-C End-to-End Communication Protection Library

AUTOSAR\_SWS\_E2ELibrary

1. E2E Protocol Specification

AUTOSAR\_PRS\_E2EProtocol

1. DDS Security, Version 1.1 <https://www.omg.org/spec/DDS-SECURITY/1.1>
2. Specification of Secure Onboard Communication

AUTOSAR\_SWS\_SecureOnboardCommunication

# 1 Introduction

This document contains the specification of the so-called the *Manifest* on the *AUTOSAR adaptive platform*. A description of the overall modeling approach can be found in section 1.1. A reference to the definition of the term *service* is given in section 1.2.

The term *Manifest* is used in this specification in the meaning of a formal specification of configuration content. Please find a more detailed description of the term and the implications for the *AUTOSAR adaptive platform* in section 2.

Please note that the content of the document (despite the name) extends to the description of design elements necessary to develop software for the *AUTOSAR adaptive platform*.

The design-related modeling mainly is focused on the development of application software on the *AUTOSAR adaptive platform* as well as the connection between application and diagnostics and is described in detail[[1]](#footnote-1) in section 3 and section 4.

Section 5, in particular, describes the big picture of *AUTOSAR classic platform* and *AUTOSAR adaptive platform* communicating via service-oriented communication.

Section 7 describes the options for configuring a machine by means of a *manifest*.

Section 8 represents that counterpart to section 3 on deployment level, it describes the content of the so-called *execution manifest*.

Section 9 contains a string of sub-sections that explain the manifest content of platform module functionality.

Section 10 provides a detailed description of how service-oriented communication shall be configured on *manifest* level.

Section 11 describes the deployment modeling for raw data stream communication.

Section 12 explains how signal-based communication can be transformed into serviceoriented communication and vice versa in order to participate in the communication between ECUs on the *AUTOSAR classic platform*.

Section 13 explains the modeling of interactions among functional clusters in the AUTOSAR stack.

Section 14 describes the idea behind and the configuration of the concept of an uploadable software package.

Finally, section 15 makes some remarks about the interaction of the *AUTOSAR adaptive platform* with software on the *AUTOSAR adaptive platform*.

## 1.1 Modeling Approach

The *AUTOSAR adaptive platform* has been introduced when the *AUTOSAR classic platform* was already a stable and well-established standard in the automotive domain.

And yet, the *AUTOSAR adaptive platform* is no successor of the *AUTOSAR classic platform*. Both platforms complement each other for specific use cases that can be better implemented by one or the other platform.

In this situation, two possible approaches for modeling on the *AUTOSAR adaptive platform* could have been taken:

* The *AUTOSAR adaptive platform* is based on different principles than the *AUTOSAR classic platform*, and hence the modeling approach could also **decouple from the canon of the AUTOSAR classic platform as much as possible** to advertise the fact that the two platforms have different purposes.

Consequentially, even if specific model elements have clear counterparts in the respective other platform, use a different terminology to not confuse the users of both platforms.

* Despite the undeniable differences between the two platforms, there is still a significant number of striking similarities that strongly encourage the **usage of existing modeling concepts** from the *AUTOSAR classic platform*, especially from the specification of the AUTOSAR Software-Component Template [1], as much as possible.

Consequentially, the conclusion is to use the identical meta-classes for similar purposes on both platforms. It will then be necessary to extend some of the affected meta-classes platform specific where applicable and add constraints that clarify the platform-specific usage of the mentioned extensions.

Without further ado, the modeling approach for the *AUTOSAR adaptive platform* follows the second alternative.

This means, for example, that a piece of application software on the *AUTOSAR adaptive platform* shall be represented by an SwComponentType. This includes the definition of CompositionSwComponentTypes that in turn aggregate SwComponentPrototypes typed by e.g. (in case of the *AUTOSAR adaptive platform*) AdaptiveApplicationSwComponentTypes.

The reuse of existing model-elements for the definition of the meta-model for the *AUTOSAR adaptive platform* has the side effect that the descriptions of existing model elements may contain references to technical details that only make sense on the *AUTOSAR classic platform*.

After all, the model elements were created when only the *AUTOSAR classic platform* existed.

These references shall be taken with a grain of salt. It is expected that readers can abstract from those details and extract the aspects of these model elements that create relevance for the description of the *AUTOSAR adaptive platform*.

## 1.2 The Term Service

It is essential to keep in mind that the term *service* is frequently used within this document in particular and the *AUTOSAR adaptive platform* in general.

This usage has its reasons despite the fact that the meaning of the term *service* on the *AUTOSAR adaptive platform* collides with other meanings used within AUTOSAR.

In summary, the following meaning of the term *service* exist in the scope of AUTOSAR:

* The Term *service* is used in the layered software architecture [2] to denote the highest layer of the AUTOSAR software architecture that interacts with the application. In this context, model elements like ServiceSwComponentType, SwcServiceDependency, ServiceNeeds, or PortInterface.isService have been created on the *AUTOSAR classic platform*.
* The term *service* is used to express that information is related or required in a workshop where a car is **serviced**. In this context, *service-only diagnostic trouble codes* (DTC) are defined.
* The term *service* is used to describe the handling of **diagnostic services**, e.g. UDS service *ReadDataByIdentifier*, for the communication between a diagnostic tester and a diagnostic stack on an (AUTOSAR) ECU.
* the term *service* is used in the meaning defined by the **service-oriented architecture** (SOA) [3]. This meaning has the strongest relation to the usage of the term *service* on the *AUTOSAR adaptive platform*.

## 1.3 Terms and Abbreviations

The main list of terms and abbreviations are defined in [4]. The following table contains the list of terms and abbreviations used in the scope of this document which are not already defined in [4] along with the spelled-out meaning of each of the abbreviations.

|  |  |
| --- | --- |
| *Abbreviation* | *Meaning* |
| AES | Advanced Encryption Standard |
| ATP | AUTOSAR Template Profile |
| ARXML | AUTOSAR XML |
| CTM | Counter Mode |

5

4

|  |  |
| --- | --- |
| *Abbreviation* | *Meaning* |
| DDS | Data Distribution Service |
| DES | Data Encryption Standard |
| DM | Diagnostic Manager |
| DTC | Diagnostic Trouble Code |
| ECB | Electronic Code Book |
| ECC | Elliptic Curve Cryptography |
| ECDSA | Elliptic Curve Digital Signature Algorithm |
| ECIES | Elliptic Curve Integrated Encryption Scheme |
| EDDSA | Edwards-Curve Digital Signature Algorithm |
| FQDN | Fully-Qualified Domain Name |
| GCM | Galios/Counter Mode |
| HMAC | Hash-based Message Authentication Code |
| HTTP | Hypertext Transport Protocol |
| ID | Identifier |
| IP | Internet Protocol |
| ISO | International Standardization Organization |
| JSON | JavaScript Object Notation |
| LAN | Local Area Network |
| MAC | Media Access Control |
| MAC | Message Authentication Code |
| MD | Message Digest |
| MTU | Maximum Transmission Unit |
| NM | Network Management |
| NV | Non-Volatile |
| PHM | Platform Health Management |
| PKCS | Public Key Cryptography Standards |
| POSIX | Portable Operating System Interface |
| PSK | Pre-Shared Key |
| ROM | Read-Only Memory |
| RSA | Cryptographic approach according to Rivest, Shamir, and  Adleman |
| SD | Service Discovery |
| SDG | Special Data Group |
| SHA | Secure Hash Algorithm |
| SOME/IP | Scalable service-Oriented MiddlewarE over IP |
| SWC | Software Component |

5 4

|  |  |
| --- | --- |
| *Abbreviation* | *Meaning* |
| TLS | Transport Layer Security |
| TLV | Tag Length Value |
| TTL | Time to Live |
| UDS | Unified Diagnostic Services |
| UML | Unified Modeling Language |
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator |
| UUID | Universally Unique Identifier |
| VLAN | Virtual Local Area Network |
| VSA | Variable Size Array |

**Table 1.1: Terms and Abbreviations used in the scope of this Document**

## 1.4 Document Conventions

Technical terms are typeset in mono spaced font, e.g. PortPrototype. As a general rule, plural forms of technical terms are created by adding "s" to the singular form, e.g. PortPrototypes. By this means the document resembles terminology used in the AUTOSAR XML Schema.

This document contains constraints in textual form that are distinguished from the rest of the text by a unique numerical constraint ID, a headline, and the actual constraint text starting after the d character and terminated by the c character.

The purpose of these constraints is to literally constrain the interpretation of the AUTOSAR meta-model such that it is possible to detect violations of the standardized behavior implemented in an instance of the meta-model (i.e. on M1 level).

Makers of AUTOSAR tools are encouraged to add the numerical ID of a constraint that corresponds to an M1 modeling issue as part of the diagnostic message issued by the tool.

The attributes of the classes introduced in this document are listed in form of class tables. They have the form shown in the example of the top-level element AUTOSAR:

Please note that constraints are not supposed to be enforceable at any given time in an AUTOSAR workflow. During the development of a model, constraints may legitimately be violated because an incomplete model will obviously show inconsistencies.

However, at specific points in the workflow, constraints shall be enforced as a safeguard against misconfiguration.

The points in the workflow where constraints shall be enforced, sometimes also known as the "binding time" of the constraint, are different for each model category, e.g. on the classic platform, the constraints defined for software-components are typically enforced prior to the generation of the RTE while the constraints against the definition of an Ecu extract shall be applied when the Ecu configuration for the Com stack is created.

For each document, possible binding times of constraints are defined and the binding times are typically mentioned in the constraint themselves to give a proper orientation for implementers of AUTOSAR authoring tools.

Let AUTOSAR be an example of a typical class table. The first rows in the table have the following meaning:

**Class**: The name of the class as defined in the UML model.

**Package**: The UML package the class is defined in. This is only listed to help locating the class in the overall meta model.

**Note**: The comment the modeler gave for the class (class note). Stereotypes and UML tags of the class are also denoted here.

**Base Classes**: If applicable, the list of direct base classes.

The headers in the table have the following meaning:

**Attribute**: The name of an attribute of the class. Note that AUTOSAR does not distinguish between class attributes and owned association ends.

**Type**: The type of an attribute of the class.

**Mul.**: The assigned multiplicity of the attribute, i.e. how many instances of the given data type are associated with the attribute.

**Kind**: Specifies, whether the attribute is aggregated in the class (aggr aggregation), an UML attribute in the class (attr primitive attribute), or just referenced by it (ref reference). Instance references are also indicated (iref instance reference) in this field.

**Note**: The comment the modeler gave for the class attribute (role note). Stereotypes and UML tags of the class are also denoted here.

Please note that the chapters that start with a letter instead of a numerical value represent the appendix of the document. The purpose of the appendix is to support the explanation of certain aspects of the document and does not represent binding conventions of the standard.

The verbal forms for the expression of obligation specified in [TPS\_STDT\_00053] shall be used to indicate requirements, see Standardization Template, chapter Support for

Traceability ([5]).

The representation of requirements in AUTOSAR documents follows the table specified in [TPS\_STDT\_00078], see Standardization Template, chapter Support for Traceability

([5]).

## 1.5 Requirements Tracing

Requirements against this document are exclusively stated in the corresponding requirements document.

The following table 1.2 references the requirements specified in the corresponding requirements document and provides information about individual specification items that fulfill a given requirement.

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Description** | **Satisfied by** |
| **[RS\_MANI\_00001]** | Adaptive AUTOSAR Application | [TPS\_MANI\_01001] [TPS\_MANI\_01010] |
| **[RS\_MANI\_00002]** | Declaration of provided and required services in an application | [TPS\_MANI\_01039] [TPS\_MANI\_01040] [TPS\_MANI\_01053]  [TPS\_MANI\_01057] [TPS\_MANI\_01190] [TPS\_MANI\_03210]  [TPS\_MANI\_03211] [TPS\_MANI\_03212] |
| **[RS\_MANI\_00003]** | Specification of service interfaces | [TPS\_MANI\_01001] [TPS\_MANI\_01004] [TPS\_MANI\_01005]  [TPS\_MANI\_01006] [TPS\_MANI\_01007] [TPS\_MANI\_01033]  [TPS\_MANI\_01034] [TPS\_MANI\_01035] [TPS\_MANI\_01064]  [TPS\_MANI\_03118] [TPS\_MANI\_03119] [TPS\_MANI\_03223] [TPS\_MANI\_03291] |
| **[RS\_MANI\_00004]** | Support of application design | [TPS\_MANI\_01010] [TPS\_MANI\_01228] [TPS\_MANI\_01229] |
| **[RS\_MANI\_00005]** | Configuration of diagnostic capabilities of an application | [TPS\_MANI\_01048] [TPS\_MANI\_01049] [TPS\_MANI\_01050]  [TPS\_MANI\_01230] [TPS\_MANI\_01259] [TPS\_MANI\_01260]  [TPS\_MANI\_01261] [TPS\_MANI\_01262] [TPS\_MANI\_01263]  [TPS\_MANI\_01326] [TPS\_MANI\_01350] [TPS\_MANI\_01351]  [TPS\_MANI\_01352] [TPS\_MANI\_01358] [TPS\_MANI\_01360]  [TPS\_MANI\_01361] [TPS\_MANI\_01362] |
| **[RS\_MANI\_00006]** | Support of application deployment | [TPS\_MANI\_01011] [TPS\_MANI\_01308] [TPS\_MANI\_01337]  [TPS\_MANI\_03147] |
| **[RS\_MANI\_00007]** | Configuration of application startup behavior | [TPS\_MANI\_01012] [TPS\_MANI\_01013] [TPS\_MANI\_01017]  [TPS\_MANI\_01041] [TPS\_MANI\_01046] [TPS\_MANI\_01061]  [TPS\_MANI\_01188] [TPS\_MANI\_01209] [TPS\_MANI\_01277]  [TPS\_MANI\_01278] [TPS\_MANI\_01328] [TPS\_MANI\_01334] [TPS\_MANI\_03151] |
| **[RS\_MANI\_00008]** | Service interface deployment to a transport layer mechanism | [TPS\_MANI\_01136] [TPS\_MANI\_01137] [TPS\_MANI\_01210]  [TPS\_MANI\_03036] [TPS\_MANI\_03037] [TPS\_MANI\_03038]  [TPS\_MANI\_03039] [TPS\_MANI\_03070] [TPS\_MANI\_03071]  [TPS\_MANI\_03072] [TPS\_MANI\_03073] [TPS\_MANI\_03074]  [TPS\_MANI\_03101] [TPS\_MANI\_03103] [TPS\_MANI\_03104]  [TPS\_MANI\_03105] [TPS\_MANI\_03106] [TPS\_MANI\_03107]  [TPS\_MANI\_03108] [TPS\_MANI\_03116] [TPS\_MANI\_03117]  [TPS\_MANI\_03217] [TPS\_MANI\_03235] [TPS\_MANI\_03278] [TPS\_MANI\_03288] |
| **[RS\_MANI\_00009]** | Service instance configuration on the network-level | [TPS\_MANI\_01316] [TPS\_MANI\_01317] [TPS\_MANI\_03001]  [TPS\_MANI\_03002] [TPS\_MANI\_03003] [TPS\_MANI\_03004]  [TPS\_MANI\_03005] [TPS\_MANI\_03006] [TPS\_MANI\_03007]  [TPS\_MANI\_03008] [TPS\_MANI\_03009] [TPS\_MANI\_03010]  [TPS\_MANI\_03022] [TPS\_MANI\_03023] [TPS\_MANI\_03024]  [TPS\_MANI\_03049] [TPS\_MANI\_03061] [TPS\_MANI\_03236] [TPS\_MANI\_03237] [TPS\_MANI\_03554] [TPS\_MANI\_03555] [TPS\_MANI\_03619] |
| **[RS\_MANI\_00011]** | Instantiation of provided and required services in an application | [TPS\_MANI\_01275] [TPS\_MANI\_01276] [TPS\_MANI\_01282]  [TPS\_MANI\_03000] |
| **[RS\_MANI\_00014]** | User defined transport layer mechanisms | [TPS\_MANI\_01165] [TPS\_MANI\_03032] [TPS\_MANI\_03045]  [TPS\_MANI\_03046] [TPS\_MANI\_03047] [TPS\_MANI\_03048]  [TPS\_MANI\_03102] [TPS\_MANI\_03280] [TPS\_MANI\_03281] |
| **[RS\_MANI\_00015]** | Definition of the nature of a manifest | [TPS\_MANI\_01000] [TPS\_MANI\_01019] [TPS\_MANI\_01020]  [TPS\_MANI\_01021] |

5

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Description** | **Satisfied by** |
| **[RS\_MANI\_00016]** | Usage of data types specifically on the AUTOSAR adaptive platform | [TPS\_MANI\_01016] [TPS\_MANI\_01027] [TPS\_MANI\_01047]  [TPS\_MANI\_01100] [TPS\_MANI\_01309] |
| **[RS\_MANI\_00017]** | Specification of the mapping of  Service Interfaces | [TPS\_MANI\_01002] [TPS\_MANI\_01003] [TPS\_MANI\_01022]  [TPS\_MANI\_01024] [TPS\_MANI\_01025] [TPS\_MANI\_01026]  [TPS\_MANI\_01032] [TPS\_MANI\_03277] [TPS\_MANI\_03289] |
| **[RS\_MANI\_00018]** | Network connections of the machine | [TPS\_MANI\_03052] [TPS\_MANI\_03053] |
| **[RS\_MANI\_00019]** | Service discovery message exchange configuration | [TPS\_MANI\_03064] |
| **[RS\_MANI\_00020]** | Hardware resources of the machine | [TPS\_MANI\_01269] [TPS\_MANI\_03035] [TPS\_MANI\_03148] |
| **[RS\_MANI\_00021]** | Description of machine states | [TPS\_MANI\_03035] |
| **[RS\_MANI\_00022]** | Adaptive Platform configuration | [TPS\_MANI\_01208] [TPS\_MANI\_01273] [TPS\_MANI\_03035] |
| **[RS\_MANI\_00023]** | Adaptive Module configuration | [TPS\_MANI\_01208] [TPS\_MANI\_01226] [TPS\_MANI\_01227]  [TPS\_MANI\_01271] [TPS\_MANI\_01279] [TPS\_MANI\_01343]  [TPS\_MANI\_02384] [TPS\_MANI\_02385] [TPS\_MANI\_02386]  [TPS\_MANI\_02387] [TPS\_MANI\_02388] [TPS\_MANI\_03035]  [TPS\_MANI\_03056] [TPS\_MANI\_03096] [TPS\_MANI\_03098]  [TPS\_MANI\_03162] [TPS\_MANI\_03163] [TPS\_MANI\_03164]  [TPS\_MANI\_03165] [TPS\_MANI\_03166] [TPS\_MANI\_03167]  [TPS\_MANI\_03218] [TPS\_MANI\_03219] [TPS\_MANI\_03220]  [TPS\_MANI\_03221] [TPS\_MANI\_03222] [TPS\_MANI\_03226]  [TPS\_MANI\_03260] [TPS\_MANI\_03261] [TPS\_MANI\_03262]  [TPS\_MANI\_03263] [TPS\_MANI\_03264] [TPS\_MANI\_03265]  [TPS\_MANI\_03266] [TPS\_MANI\_03267] [TPS\_MANI\_03268]  [TPS\_MANI\_03269] [TPS\_MANI\_03270] [TPS\_MANI\_03271]  [TPS\_MANI\_03272] [TPS\_MANI\_03273] [TPS\_MANI\_03274]  [TPS\_MANI\_03276] [TPS\_MANI\_03279] [TPS\_MANI\_03282]  [TPS\_MANI\_03283] [TPS\_MANI\_03285] [TPS\_MANI\_03286]  [TPS\_MANI\_03502] [TPS\_MANI\_03503] [TPS\_MANI\_03505]  [TPS\_MANI\_03506] [TPS\_MANI\_03508] [TPS\_MANI\_03509]  [TPS\_MANI\_03510] [TPS\_MANI\_03511] [TPS\_MANI\_03512]  [TPS\_MANI\_03513] [TPS\_MANI\_03514] [TPS\_MANI\_03515]  [TPS\_MANI\_03516] [TPS\_MANI\_03517] [TPS\_MANI\_03544]  [TPS\_MANI\_03545] [TPS\_MANI\_03546] [TPS\_MANI\_03553]  [TPS\_MANI\_03573] [TPS\_MANI\_03574] [TPS\_MANI\_03575]  [TPS\_MANI\_03576] [TPS\_MANI\_03625] [TPS\_MANI\_03626]  [TPS\_MANI\_03633] [TPS\_MANI\_03651] |
| **[RS\_MANI\_00024]** | SOME/IP transport layer mechanisms | [TPS\_MANI\_01136] [TPS\_MANI\_01137] [TPS\_MANI\_03002]  [TPS\_MANI\_03003] [TPS\_MANI\_03004] [TPS\_MANI\_03005]  [TPS\_MANI\_03006] [TPS\_MANI\_03007] [TPS\_MANI\_03008]  [TPS\_MANI\_03009] [TPS\_MANI\_03010] [TPS\_MANI\_03011]  [TPS\_MANI\_03012] [TPS\_MANI\_03013] [TPS\_MANI\_03014]  [TPS\_MANI\_03015] [TPS\_MANI\_03016] [TPS\_MANI\_03017]  [TPS\_MANI\_03018] [TPS\_MANI\_03020] [TPS\_MANI\_03021]  [TPS\_MANI\_03022] [TPS\_MANI\_03023] [TPS\_MANI\_03024]  [TPS\_MANI\_03025] [TPS\_MANI\_03026] [TPS\_MANI\_03027]  [TPS\_MANI\_03028] [TPS\_MANI\_03029] [TPS\_MANI\_03030]  [TPS\_MANI\_03031] [TPS\_MANI\_03040] [TPS\_MANI\_03041]  [TPS\_MANI\_03042] [TPS\_MANI\_03043] [TPS\_MANI\_03044]  [TPS\_MANI\_03049] [TPS\_MANI\_03050] [TPS\_MANI\_03051]  [TPS\_MANI\_03057] [TPS\_MANI\_03059] [TPS\_MANI\_03061]  [TPS\_MANI\_03067] [TPS\_MANI\_03068] [TPS\_MANI\_03069]  [TPS\_MANI\_03070] [TPS\_MANI\_03071] [TPS\_MANI\_03072]  [TPS\_MANI\_03073] [TPS\_MANI\_03074] [TPS\_MANI\_03116]  [TPS\_MANI\_03154] [TPS\_MANI\_03155] [TPS\_MANI\_03156]  [TPS\_MANI\_03157] [TPS\_MANI\_03158] [TPS\_MANI\_03159]  [TPS\_MANI\_03168] [TPS\_MANI\_03217] [TPS\_MANI\_03227]  [TPS\_MANI\_03230] [TPS\_MANI\_03231] [TPS\_MANI\_03235] [TPS\_MANI\_03237] [TPS\_MANI\_03278] [TPS\_MANI\_03554] [TPS\_MANI\_03555] |

5

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Description** | **Satisfied by** |
| **[RS\_MANI\_00025]** | Definition and configuration of serialization | [TPS\_MANI\_01210] [TPS\_MANI\_03101] [TPS\_MANI\_03102]  [TPS\_MANI\_03103] [TPS\_MANI\_03104] [TPS\_MANI\_03105]  [TPS\_MANI\_03106] [TPS\_MANI\_03107] [TPS\_MANI\_03108]  [TPS\_MANI\_03109] [TPS\_MANI\_03117] [TPS\_MANI\_03288] |
| **[RS\_MANI\_00026]** | Software Component System Design | [TPS\_MANI\_01191] [TPS\_MANI\_01192] [TPS\_MANI\_01198]  [TPS\_MANI\_03110] [TPS\_MANI\_03111] [TPS\_MANI\_03112]  [TPS\_MANI\_03113] [TPS\_MANI\_03114] [TPS\_MANI\_03115] |
| **[RS\_MANI\_00027]** | Support for access to persistent data | [TPS\_MANI\_01065] [TPS\_MANI\_01067] [TPS\_MANI\_01068]  [TPS\_MANI\_01073] [TPS\_MANI\_01078] [TPS\_MANI\_01079]  [TPS\_MANI\_01080] [TPS\_MANI\_01081] [TPS\_MANI\_01135]  [TPS\_MANI\_01138] [TPS\_MANI\_01139] [TPS\_MANI\_01140]  [TPS\_MANI\_01142] [TPS\_MANI\_01144] [TPS\_MANI\_01146]  [TPS\_MANI\_01147] [TPS\_MANI\_01148] [TPS\_MANI\_01149]  [TPS\_MANI\_01150] [TPS\_MANI\_01155] [TPS\_MANI\_01156]  [TPS\_MANI\_01157] [TPS\_MANI\_01159] [TPS\_MANI\_01160]  [TPS\_MANI\_01179] [TPS\_MANI\_01180] [TPS\_MANI\_01182]  [TPS\_MANI\_01187] [TPS\_MANI\_01194] [TPS\_MANI\_01196]  [TPS\_MANI\_01197] [TPS\_MANI\_01204] [TPS\_MANI\_01205]  [TPS\_MANI\_01206] [TPS\_MANI\_01207] [TPS\_MANI\_01313]  [TPS\_MANI\_01314] [TPS\_MANI\_01315] [TPS\_MANI\_01319]  [TPS\_MANI\_01320] [TPS\_MANI\_01321] [TPS\_MANI\_01322] [TPS\_MANI\_01323] |
| **[RS\_MANI\_00028]** | Configuration of Safety protection | [TPS\_MANI\_01324] [TPS\_MANI\_01325] [TPS\_MANI\_01327]  [TPS\_MANI\_03127] [TPS\_MANI\_03128] [TPS\_MANI\_03129]  [TPS\_MANI\_03130] [TPS\_MANI\_03131] [TPS\_MANI\_03132]  [TPS\_MANI\_03228] [TPS\_MANI\_03229] [TPS\_MANI\_03252] |
| **[RS\_MANI\_00029]** | Mapping description between  Signal-based communication and  Service-Oriented communication | [TPS\_MANI\_03124] [TPS\_MANI\_03125] [TPS\_MANI\_03126]  [TPS\_MANI\_03627] [TPS\_MANI\_03629] [TPS\_MANI\_03635] |
| **[RS\_MANI\_00030]** | Definition of optional elements in composite data structures | [TPS\_MANI\_01097] [TPS\_MANI\_01184] [TPS\_MANI\_01185]  [TPS\_MANI\_01186] [TPS\_MANI\_01270] [TPS\_MANI\_01333] |
| **[RS\_MANI\_00031]** | Interaction with Crypto Software | [TPS\_MANI\_03253] [TPS\_MANI\_03254] [TPS\_MANI\_03255]  [TPS\_MANI\_03256] [TPS\_MANI\_03257] [TPS\_MANI\_03258] [TPS\_MANI\_03259] |
| **[RS\_MANI\_00032]** | Support for platform health management | [TPS\_MANI\_01280] [TPS\_MANI\_03500] [TPS\_MANI\_03502]  [TPS\_MANI\_03503] [TPS\_MANI\_03505] [TPS\_MANI\_03506]  [TPS\_MANI\_03508] [TPS\_MANI\_03509] [TPS\_MANI\_03510]  [TPS\_MANI\_03511] [TPS\_MANI\_03512] [TPS\_MANI\_03513]  [TPS\_MANI\_03514] [TPS\_MANI\_03515] [TPS\_MANI\_03516]  [TPS\_MANI\_03517] [TPS\_MANI\_03534] [TPS\_MANI\_03544]  [TPS\_MANI\_03545] [TPS\_MANI\_03546] [TPS\_MANI\_03553]  [TPS\_MANI\_03573] [TPS\_MANI\_03574] [TPS\_MANI\_03575]  [TPS\_MANI\_03576] [TPS\_MANI\_03623] [TPS\_MANI\_03624]  [TPS\_MANI\_03625] [TPS\_MANI\_03626] [TPS\_MANI\_03630]  [TPS\_MANI\_03631] [TPS\_MANI\_03633] [TPS\_MANI\_03651] |
| **[RS\_MANI\_00034]** | Specification of intents | [TPS\_MANI\_01106] [TPS\_MANI\_01107] [TPS\_MANI\_01108]  [TPS\_MANI\_03209] |
| **[RS\_MANI\_00035]** | Definition of an uploadable software package | [TPS\_MANI\_01109] [TPS\_MANI\_01110] [TPS\_MANI\_01111]  [TPS\_MANI\_01112] [TPS\_MANI\_01113] [TPS\_MANI\_01114]  [TPS\_MANI\_01115] [TPS\_MANI\_01116] [TPS\_MANI\_01117]  [TPS\_MANI\_01118] [TPS\_MANI\_01119] [TPS\_MANI\_01161]  [TPS\_MANI\_01164] [TPS\_MANI\_01189] [TPS\_MANI\_01202]  [TPS\_MANI\_01211] [TPS\_MANI\_01213] [TPS\_MANI\_01214]  [TPS\_MANI\_01215] [TPS\_MANI\_01216] [TPS\_MANI\_01217]  [TPS\_MANI\_01218] [TPS\_MANI\_01219] [TPS\_MANI\_01220]  [TPS\_MANI\_01221] [TPS\_MANI\_01222] [TPS\_MANI\_01223]  [TPS\_MANI\_01225] [TPS\_MANI\_01310] [TPS\_MANI\_01329]  [TPS\_MANI\_01331] [TPS\_MANI\_01335] [TPS\_MANI\_01344]  [TPS\_MANI\_01345] [TPS\_MANI\_01346] [TPS\_MANI\_01349] |

5

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Description** | **Satisfied by** |
| **[RS\_MANI\_00036]** | Configuration of security protection | [TPS\_MANI\_03133] [TPS\_MANI\_03134] [TPS\_MANI\_03137]  [TPS\_MANI\_03138] [TPS\_MANI\_03139] [TPS\_MANI\_03140]  [TPS\_MANI\_03199] [TPS\_MANI\_03200] [TPS\_MANI\_03203]  [TPS\_MANI\_03204] [TPS\_MANI\_03205] [TPS\_MANI\_03206]  [TPS\_MANI\_03207] [TPS\_MANI\_03208] [TPS\_MANI\_03213]  [TPS\_MANI\_03214] [TPS\_MANI\_03215] [TPS\_MANI\_03216]  [TPS\_MANI\_03232] [TPS\_MANI\_03233] [TPS\_MANI\_03234]  [TPS\_MANI\_03240] [TPS\_MANI\_03241] [TPS\_MANI\_03242] [TPS\_MANI\_03661] |
| **[RS\_MANI\_00037]** | Configuration of logging and tracing | [TPS\_MANI\_01271] [TPS\_MANI\_03284] |
| **[RS\_MANI\_00038]** | DDS transport layer mechanisms | [TPS\_MANI\_03525] [TPS\_MANI\_03526] [TPS\_MANI\_03527]  [TPS\_MANI\_03528] [TPS\_MANI\_03529] [TPS\_MANI\_03530]  [TPS\_MANI\_03531] [TPS\_MANI\_03532] [TPS\_MANI\_03533]  [TPS\_MANI\_03556] [TPS\_MANI\_03557] [TPS\_MANI\_03558]  [TPS\_MANI\_03561] [TPS\_MANI\_03562] [TPS\_MANI\_03567]  [TPS\_MANI\_03568] [TPS\_MANI\_03622] [TPS\_MANI\_03650] [TPS\_MANI\_03662] |
| **[RS\_MANI\_00039]** | Usage of implementation specific data types | [TPS\_MANI\_01166] [TPS\_MANI\_01167] [TPS\_MANI\_01168]  [TPS\_MANI\_01169] [TPS\_MANI\_01171] [TPS\_MANI\_01172]  [TPS\_MANI\_01173] [TPS\_MANI\_01174] [TPS\_MANI\_01175]  [TPS\_MANI\_01176] [TPS\_MANI\_01177] [TPS\_MANI\_01201]  [TPS\_MANI\_01212] [TPS\_MANI\_03169] [TPS\_MANI\_03170]  [TPS\_MANI\_03171] [TPS\_MANI\_03172] [TPS\_MANI\_03173]  [TPS\_MANI\_03174] [TPS\_MANI\_03175] [TPS\_MANI\_03176]  [TPS\_MANI\_03177] [TPS\_MANI\_03178] [TPS\_MANI\_03179]  [TPS\_MANI\_03180] [TPS\_MANI\_03181] [TPS\_MANI\_03183]  [TPS\_MANI\_03184] [TPS\_MANI\_03185] [TPS\_MANI\_03186]  [TPS\_MANI\_03187] [TPS\_MANI\_03188] [TPS\_MANI\_03189]  [TPS\_MANI\_03190] [TPS\_MANI\_03191] [TPS\_MANI\_03192]  [TPS\_MANI\_03193] [TPS\_MANI\_03196] [TPS\_MANI\_03197]  [TPS\_MANI\_03198] [TPS\_MANI\_03201] [TPS\_MANI\_03202] |
| **[RS\_MANI\_00040]** | Support for access to synchronized time | [TPS\_MANI\_03535] [TPS\_MANI\_03536] [TPS\_MANI\_03537]  [TPS\_MANI\_03539] [TPS\_MANI\_03541] [TPS\_MANI\_03542]  [TPS\_MANI\_03543] [TPS\_MANI\_03547] [TPS\_MANI\_03548]  [TPS\_MANI\_03549] [TPS\_MANI\_03551] [TPS\_MANI\_03632] |
| **[RS\_MANI\_00041]** | Configuration of function groups | [TPS\_MANI\_01330] [TPS\_MANI\_03145] [TPS\_MANI\_03152]  [TPS\_MANI\_03194] [TPS\_MANI\_03195] |
| **[RS\_MANI\_00050]** | Support of Deterministic Client | [TPS\_MANI\_01199] [TPS\_MANI\_01200] [TPS\_MANI\_01203] |
| **[RS\_MANI\_00060]** | Support of Identity and Access  Management | [TPS\_MANI\_01231] [TPS\_MANI\_01232] [TPS\_MANI\_01233]  [TPS\_MANI\_01234] [TPS\_MANI\_01235] [TPS\_MANI\_01236]  [TPS\_MANI\_01237] [TPS\_MANI\_01238] [TPS\_MANI\_01239]  [TPS\_MANI\_01240] [TPS\_MANI\_01241] [TPS\_MANI\_01284]  [TPS\_MANI\_01307] [TPS\_MANI\_03238] [TPS\_MANI\_03239]  [TPS\_MANI\_03240] [TPS\_MANI\_03241] [TPS\_MANI\_03242]  [TPS\_MANI\_03244] [TPS\_MANI\_03245] [TPS\_MANI\_03246]  [TPS\_MANI\_03247] [TPS\_MANI\_03248] [TPS\_MANI\_03249]  [TPS\_MANI\_03250] [TPS\_MANI\_03251] [TPS\_MANI\_03290] |
| **[RS\_MANI\_00061]** | Support of Diagnostic Interfaces | [TPS\_MANI\_01048] [TPS\_MANI\_01049] [TPS\_MANI\_01050]  [TPS\_MANI\_01242] [TPS\_MANI\_01243] [TPS\_MANI\_01244]  [TPS\_MANI\_01245] [TPS\_MANI\_01246] [TPS\_MANI\_01247]  [TPS\_MANI\_01248] [TPS\_MANI\_01249] [TPS\_MANI\_01250]  [TPS\_MANI\_01251] [TPS\_MANI\_01252] [TPS\_MANI\_01253]  [TPS\_MANI\_01254] [TPS\_MANI\_01255] [TPS\_MANI\_01259]  [TPS\_MANI\_01260] [TPS\_MANI\_01261] [TPS\_MANI\_01262]  [TPS\_MANI\_01263] [TPS\_MANI\_01265] [TPS\_MANI\_01326]  [TPS\_MANI\_01332] [TPS\_MANI\_01347] [TPS\_MANI\_01348]  [TPS\_MANI\_01351] [TPS\_MANI\_01352] [TPS\_MANI\_01353] [TPS\_MANI\_01359] [TPS\_MANI\_01360] [TPS\_MANI\_01361]  [TPS\_MANI\_01362] [TPS\_MANI\_01363] |
| **[RS\_MANI\_00062]** | Support for Partial Networking | [TPS\_MANI\_03224] [TPS\_MANI\_03225] |

5

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Description** | **Satisfied by** |
| **[RS\_MANI\_00063]** | The Manifest specification shall support the translation between signal-based and service-oriented communication | [TPS\_MANI\_03287] [TPS\_MANI\_03577] [TPS\_MANI\_03578]  [TPS\_MANI\_03579] [TPS\_MANI\_03580] [TPS\_MANI\_03581]  [TPS\_MANI\_03582] [TPS\_MANI\_03583] [TPS\_MANI\_03585]  [TPS\_MANI\_03589] [TPS\_MANI\_03590] [TPS\_MANI\_03591]  [TPS\_MANI\_03592] [TPS\_MANI\_03593] [TPS\_MANI\_03594]  [TPS\_MANI\_03595] [TPS\_MANI\_03597] [TPS\_MANI\_03598]  [TPS\_MANI\_03599] [TPS\_MANI\_03600] [TPS\_MANI\_03601]  [TPS\_MANI\_03602] [TPS\_MANI\_03603] [TPS\_MANI\_03604]  [TPS\_MANI\_03605] [TPS\_MANI\_03606] [TPS\_MANI\_03607]  [TPS\_MANI\_03608] [TPS\_MANI\_03609] [TPS\_MANI\_03610]  [TPS\_MANI\_03611] [TPS\_MANI\_03612] [TPS\_MANI\_03614]  [TPS\_MANI\_03615] [TPS\_MANI\_03620] [TPS\_MANI\_03621]  [TPS\_MANI\_03636] [TPS\_MANI\_03637] [TPS\_MANI\_03638]  [TPS\_MANI\_03639] [TPS\_MANI\_03640] [TPS\_MANI\_03641]  [TPS\_MANI\_03642] [TPS\_MANI\_03643] [TPS\_MANI\_03644]  [TPS\_MANI\_03645] [TPS\_MANI\_03646] [TPS\_MANI\_03647]  [TPS\_MANI\_03648] [TPS\_MANI\_03649] [TPS\_MANI\_03652]  [TPS\_MANI\_03653] [TPS\_MANI\_03654] [TPS\_MANI\_03655]  [TPS\_MANI\_03656] [TPS\_MANI\_03657] [TPS\_MANI\_03658]  [TPS\_MANI\_03659] [TPS\_MANI\_03660] |
| **[RS\_MANI\_00064]** | Service contract version for a service interface | [TPS\_MANI\_03616] |
| **[RS\_MANI\_00065]** | Service contract versioning for all  Transport Deployment Protocols | [TPS\_MANI\_03617] |
| **[RS\_MANI\_00066]** | Service Versioning Blacklist | [TPS\_MANI\_03618] |
| **[RS\_MANI\_00067]** | Raw data stream deployment | [TPS\_MANI\_01285] [TPS\_MANI\_01287] [TPS\_MANI\_01307]  [TPS\_MANI\_01354] [TPS\_MANI\_01355] [TPS\_MANI\_01356] [TPS\_MANI\_01357] |
| **[RS\_MANI\_00068]** | Support for security event reporting interface definitions | [TPS\_MANI\_01338] [TPS\_MANI\_01339] [TPS\_MANI\_01340] |
| **[RS\_MANI\_00069]** | Support for deployment definition of the Intrusion Detection System Manager | [TPS\_MANI\_01341] [TPS\_MANI\_01342] |

**Table 1.2: RequirementsTracing**

# 2 Big Picture of Manifest Definition

## 2.1 Design vs. Deployment

**2.1.1 Overview**

Despite the name, this document contains the description of model elements that are clearly bound to a *design* workflow **and** model elements that have a strong relation to the *deployment* aspect.

Model elements discussed in this document are either related to *design* or *deployment*, there is no overlap between the two groups.

Model elements that are related to *deployment* will be used in models that are uploaded to a target platform, see [TPS\_MANI\_01000]. These model elements are mainly described in sections of this document where the term “Manifest” is part of the section title.

**2.1.2 Relation between Design and Deployment Models**

Please note that in many cases the part of the meta-model related to *deployment* reflects a similar modeling in the *design* domain, e.g. the definition of E2E profile parameters.

There is currently no clearly defined preference about how the relation between *design* and *deployment* may impact a concrete development project. The following scenarios for the example of *E2E properties* might occur:

* An OEM delivers the description of AdaptivePlatformServiceInstances including the definition of *E2E properties*.

It is safe to assume that subsequent processing of the model shall take the *E2E properties* as granted and develop the software with respect to the given properties.

* Software exists that has defined *E2E properties* by means of ComSpecs. For various reasons, it may happen that the software cannot be updated and therefore takes the “lead” in terms of the definition of *E2E properties*.

The definition of AdaptivePlatformServiceInstances may then have to respect the existing modeling on the software side.

* It could also happen that existing definitions can be **partly** overwritten by engineers who **really** know what they are doing.

In addition, it should be noted that some model elements are used in both design and deployment steps of the workflow, which is another indication that the border between design and deployment is not as easily defined as on the *AUTOSAR classic platform*.

In contrast, other model elements described in this document that are part of the content that gets uploaded to the target platform have direct counterparts on the design level.

One example for such a relation is the definition of ProcessDesign and Process. In these cases it would be easier to draw the line between design and deployment aspects of the model.

A consequence of the (at least intended) separation between design and deployment model elements is that content that is primarily related to design objects needs to be duplicated on the deployment level as a measure to keep the actual manifest content as lean as possible.

For example, if a deployment element needs to refer to a PortPrototype and information from e.g. the ComSpec attached to the PortPrototype is required in the manifest to properly define the intended semantics, then the owner of the PortPrototype (i.e. a SwComponentPrototype typed by an SwComponentType) would also have to appear in the manifest.

It is obvious that, this way, the entire software model would make it into the manifest and inflate the manifest content unnecessarily.

However, in such a situation, there is a tendency to duplicate model content from the design domain in the deployment to at least keep the manifest content as compact as possible.

There are cases, however, where the mere existence of a reference into the design model is already representing valuable information. References that expose this capability are decorated with the stereotype atpUriDef.

Specifically, it is possible for the software on the platform to derive the value of an InstanceSpecifier from the content of an instanceRef decorated with atpUriDef. This is a very important mechanism for the platform software to interact with the application layer.

Another example is the definition of the checkpointId in the context of the configuration of the Platform Health Manager.

The modeling of the PHM interaction on the application layer involves the definition of the value of the PhmCheckpoint.checkpointId and the PortPrototype where a specific PhmCheckpoint.checkpointId is used is identified by means of a reference stereotyped as atpUriDef.

And because the target of the reference is not necessarily existing on the platform, the SupervisionCheckpoint.checkpointId is replicated in the manifest model so that the platform software has access to this important piece of information (this aspect is also explained in Figure 9.9).

**2.1.3 Structure of the document**

The structure of the document maps to the division between *design* and *deployment* such that the *design* aspect is mostly described in sections 3, 4, 5, 6, and most of 12.

In contrast, chapters 7, 8, 9, 10, 12.3, 13, and 14 focus on *deployment*-related content.

## 2.2 About Manifest

This chapter shall clarify the definition of the term Manifest in the context of the *AUTOSAR adaptive platform*.

**[TPS\_MANI\_01000]**{DRAFT} **Definition of the term Manifest** dA Manifest represents a piece of AUTOSAR model description that is created to support the configuration of an *AUTOSAR adaptive platform* product and which is uploaded to the *AUTOSAR adaptive platform* product, potentially in combination with other artifacts (like binary files) that contain executable code to which the Manifest applies.c*(RS\_MANI\_00015)*

It is important to stress the fact that the usage of a Manifest is indeed strictly limited to the *AUTOSAR adaptive platform* and that there is no use case to port the concept to the *AUTOSAR classic platform*.

## 2.3 Serialization Format

One aspect that the definition of a Manifest has in common with other AUTOSAR model content is the standardized serialization format.

**[TPS\_MANI\_01020]**{DRAFT} **Serialization format of the Manifest in AUTOSAR** dThe standardized serialization format of Manifest content in AUTOSAR is ARXML.

Consequently, Manifest model content can be validated against the AUTOSAR XML Schema.c*(RS\_MANI\_00015)*

An important consequence of [TPS\_MANI\_01020] is that there is no limitation to just one “manifest file” a.k.a. “the manifest”.

Content may be distributed among several physical files according to the rules given in the specification of the AUTOSAR Generic Structure Template [6].

**[TPS\_MANI\_01021]**{DRAFT} **Serialization format of Manifest content on a machine** dThe serialization format used to actually upload a manifest on a machine may be freely chosen by a platform supplier.

However, the content and semantics of the original ARXML Manifest needs to be **fully preserved**.c*(RS\_MANI\_00015)*

It can be expected that in many cases the best option for the upload of the Manifest will still be ARXML because a custom format obviously has to support the full complexity of the Manifest meta-model.

Application1



«manifest»

Application1Service.arxml



«manifest»

Application1Startup.arxml



«binary»

Application1Executable

**Figure 2.1: Example usage of several manifest files within one software delivery**

Please note that the meta-model foresees the existence of references from manifestrelated meta-classes to design-related meta-classes.

These references are created for the sake of clarity but it is not mandatory that the content of the reference actually needs to be resolvable.

In terms of the AUTOSAR modeling approach, this translates to a decoration of these references with the stereotype atpUriDef. More information can be found in [6].

If the referenced meta-classes contain information that is relevant for the manifest level then this information is replicated on the manifest level (such that the manifest-level model does not have to rely on the availability of design-level information).

## 2.4 Scope

As mentioned before, the usage of a Manifest is limited to the *AUTOSAR adaptive platform*. This does not mean, however, that all ARXML produced in a development project that targets the *AUTOSAR adaptive platform* is automatically considered a Manifest.

In fact, the *AUTOSAR adaptive platform* is usually not exclusively used in a vehicle project.

A typical vehicle will most likely be also equipped with a number of ECUs developed on the *AUTOSAR classic platform* and the system design for the entire vehicle will therefore have to cover both ECUs built on top of the *AUTOSAR classic platform* and those created on top of the *AUTOSAR adaptive platform*.

**[TPS\_MANI\_01019]**{DRAFT}**Manifest content may apply to different aspects of the *AUTOSAR adaptive platform*** dManifest content can apply to different aspects of the model. At the moment, Manifest content can roughly be divided into three focus areas:

* Application-related Manifest content describes all aspects of the deployment of an application, including - but not limited to - the startup configuration and the configuration of service-oriented communication endpoints on application level.
* Machine-related Manifest content describes the deployment of just a machine, i.e. without any application (including platform modules) running on the machine.
* Service instance-related Manifest describes how service-oriented communication on transport layer level is bound to endpoints in the application and (in some cases) platform software.

c*(RS\_MANI\_00015)*

## 2.5 Manifests described in this Document

In principle, the term Manifest could be defined such that there is conceptually just one “manifest” and every deployment aspect would be handled in this context.

This does not seem appropriate because it became apparent that manifest-related model-elements exist that are relevant in entirely different phases of a typical development project.

This aspect is taken as the main motivation to subdivide the definition of the term Manifest in five different partitions:

**Execution Manifest** This kind of Manifest is used to specify the deployment-related information of applications running on the *AUTOSAR adaptive platform*.

An Execution Manifest is bundled with the actual executable code in order to support the integration of the executable code onto the machine.

Please find more information regarding this topic in section 8.

**Service Instance Manifest** This kind of Manifest is used to specify how serviceoriented communication is configured in terms of the requirements of the underlying transport protocols.

A Service Instance Manifest is bundled with the actual executable code that implements the respective usage of service-oriented communication.

Please find more information regarding this topic in section 10.

**Machine Manifest** This kind of Manifest is supposed to describe deploymentrelated content that applies to the configuration of just the underlying machine (i.e. without any applications running on the machine) that runs an *AUTOSAR adaptive platform*.

A Machine Manifest is bundled with the software taken to establish an instance of the *AUTOSAR adaptive platform*.

Please find more information regarding this topic in sections 7 and 9.

**Raw Data Stream Manifest** This kind of Manifest describes the configuration of client and server for the purpose of communicating via raw data streams.

**Software Distribution** This kind of Manifest describes the packaging and logistics aspects of software on the *AUTOSAR adaptive platform*.

Please find more information regarding this topic in section 14.

The temporal division between the definition (and usage) of different kinds of Manifest leads to the conclusion that in most cases different physical files will be used to store the content of the different kinds of Manifest.

However, as with all kinds of ARXML content, this is not a binding rule.

# 3 Application Design

## 3.1 Overview

This chapter describes all design-related modeling that applies to the creation of application software on the *AUTOSAR adaptive platform*.

This also extends to extensions of existing modeling used on the *AUTOSAR classic platform*, e.g. the introduction of new values of the attribute category.

In particular, this section of the document focuses on the following aspects:

* Definition of a dedicated subclass of SwComponentType for the *AUTOSAR adaptive platform* (section 3.2)
* Definition of data types specifically for the *AUTOSAR adaptive platform* (section

3.3)

* Service interface as the pivotal element for service-oriented communication (section 3.4)
* Definition of domain-specific PortInterfaces, e.g. for diagnostics (section

3.10), PHM (section 3.9), persistency (section 3.7), crypto (section 3.11).

* Service interface mapping as a mediator between internal and external communication (section 3.5)
* Service interface **element** mapping as a mediator between internal and external communication (section 3.6)
* Aspects of the fine-grained configuration of interaction with the “outside world” from the perspective of the inside of a software-component (section 3.15)
* Executable as the smallest executable unit (section 3.16)
* Handling of optional elements in data structures, see section 3.17
* Configuration of transformation properties (section 3.18)
* Description of how to front-load the configuration of a Process by means of ProcessDesign, see section 3.19.
* Description of the design-level IAM configuration by means of the GrantDesign, see section 3.20

## 3.2 Software Component

In principle, it would be possible to directly take over the definition of e.g. ApplicationSwComponentType for the usage on the *AUTOSAR adaptive platform*.

However, this would complicate the formulation of constraints regarding the existence of model elements (for example: data types, as explained in section 3.3) that are exclusive to the *AUTOSAR adaptive platform*.

Therefore, the AdaptiveApplicationSwComponentType is defined as a representation of software-components on the *AUTOSAR adaptive platform*.

The Existence of the AdaptiveApplicationSwComponentType allows for a convenient way (see [constr\_1492]) to lock out most kinds of software-component defined for the *AUTOSAR classic platform* from the usage on the *AUTOSAR adaptive platform*.

The clarification of the opposite direction (i.e. an erroneous use of an AdaptiveApplicationSwComponentType) is less obvious.

In other words, it may be possible to use an AdaptiveApplicationSwComponentType within a System as some sort of overall design model for software on both the *AUTOSAR classic platform* **and** the *AUTOSAR adaptive platform*.

This aspect, however, is not clarified so far nor is a restriction in place that prohibits AdaptiveApplicationSwComponentType to appear in the context of a System.

Later versions of this specification may fix the missing regulation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **AdaptiveApplicationSwComponentType** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure | | | |
| ***Note*** | This meta-class represents the ability to support the formal modeling of application software on the AUTOSAR adaptive platform. Consequently, it shall only be used on the AUTOSAR adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=AdaptiveApplicationSwComponentTypes | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| internalBehavior | AdaptiveSwcInternal  Behavior | 0..1 | aggr | This aggregation represents the internal behavior of the AdaptiveApplicationSwComponentType for the AUTOSAR adaptive platform.  **Stereotypes:** atpSplitable; atpVariation **Tags:**  atp.Splitkey=internalBehavior.shortName, internal Behavior.variationPoint.shortLabel atp.Status=draft  vh.latestBindingTime=preCompileTime |

**Table 3.1: AdaptiveApplicationSwComponentType**

|  |  |
| --- | --- |
| ***Class*** | **AdaptiveSwcInternalBehavior** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::AdaptiveInternalBehavior |
| ***Note*** | This meta-class represents the ability to define an internal behavior of an AtomicSwComponentType used on the AUTOSAR adaptive platform.  Please note that the model of internal behavior in this case, in stark contrast to the situation of the AUTOSAR classic platform, is very minimal.  **Tags:**atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **AdaptiveSwcInternalBehavior** | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| service  Dependency | SwcService  Dependency | \* | aggr | This represents the collection of SwcService  Dependencys owned by AdaptiveInternalBehavior.  **Tags:**atp.Status=draft |

**Table 3.2: AdaptiveSwcInternalBehavior**

## 3.3 Data Type

**3.3.1 Overview**

The specification of data types on the *AUTOSAR adaptive platform* follows the same pattern as the counterpart on the *AUTOSAR classic platform*: data types are defined on different levels of abstraction that complement each other.

In the context of this document, the focus is on the discussion of ApplicationDataTypes and CppImplementationDataTypes.

In general, most of the concepts regarding the definition of data types can be taken over from the existing specifications on the *AUTOSAR classic platform*.

However, some aspects are specific to the *AUTOSAR adaptive platform* and are consequently discussed in the scope of this document rather than the specification of the AUTOSAR Software Component Template [1].

One of the aspects that could be taken over from the *AUTOSAR classic platform* is the definition of initial values.

Although the utility of initial values is certainly limited on the *AUTOSAR adaptive platform*, there is an opportunity to utilize the definition of initial values in the context of the so-called Fields (see [TPS\_MANI\_01034]).

**3.3.2 ApplicationDataType**

The full range of the modeling of ApplicationDataTypes that is supported on the *AUTOSAR classic platform* can directly be used on the *AUTOSAR adaptive platform* as well.

In addition to the ApplicationDataTypes supported on the *AUTOSAR classic platform*, there are further ApplicationDataTypes that – while in principle also available on the *AUTOSAR classic platform* – are primarily used on and designed for the *AUTOSAR adaptive platform*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***ApplicationDataType*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes | | | |
| ***Note*** | ApplicationDataType defines a data type from the application point of view. Especially it should be used whenever something "physical" is at stake.  An ApplicationDataType represents a set of values as seen in the application model, such as measurement units. It does not consider implementation details such as bit-size, endianess, etc.  It should be possible to model the application level aspects of a VFB system by using ApplicationData Types only. | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*,  *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Subclasses*** | *ApplicationCompositeDataType*, ApplicationPrimitiveDataType | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.3: ApplicationDataType**

**3.3.2.1 String Data Type**

While the handling of data types that represent textual strings is very similar with respect to the definition of ApplicationDataTypes on the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform*, special regulations apply on the level of CppImplementationDataTypes on the *AUTOSAR adaptive platform*.

For more information about the modeling of string data types on the level of CppImplementationDataType please refer to section 3.3.3.4.

For the sake of consistency, this chapter summarizes the modeling of ApplicationDataTypes for the modeling of data types that represent textual strings as far as the *AUTOSAR adaptive platform* is concerned.

ApplicationPrimitiveDataType

SwTextProps

arraySizeSemantics: ArraySizeSemanticsEnum

[0..1]

+

swFillCharacter: Integer

[0..1]

+

«atpVariation»

[0..1]

swMaxTextSize: Integer

+

«atpVariation»

SwDataDefProps

*AtpBlueprint*

*AtpBlueprintable*

*ApplicationDataType*

*ARElement*

*AtpType*

*AutosarDataType*

*AtpBlueprint*

*AtpBlueprintable*

*BaseType*

SwBaseType

*ARElement*

SwRecordLayout

*CompositeRuleBasedValueArgument*

ApplicationValueSpecification

+

category: Identifier

[0..1]

*ValueSpecification*

shortLabel: Identifier

+

[0..1]

+

baseType

0..1

swDataDefProps

+

0..1

swRecordLayout

+

0..1

swTextProps

+

0..1

invalidValue

+

0..1

**Figure 3.1: Specification of textual strings**

The meta-classes used to define an ApplicationPrimitiveDataType of categorySTRING are summarized in Figure 3.1.

Please note that thanks to the usage of programming languages with richer data types than plain C, the implementation of an ApplicationPrimitiveDataType of categorySTRING on the *AUTOSAR adaptive platform* is predefined for a given *language binding*.

**[TPS\_MANI\_01047]**{DRAFT} **Existence of SwRecordLayout for an ApplicationPrimitiveDataType of category STRING** dFor the usage of an ApplicationPrimitiveDataType of category STRING on the *AUTOSAR adaptive platform*, the existence of ApplicationPrimitiveDataType.swDataDefProps. swRecordLayout shall be ignored.c*(RS\_MANI\_00016)*

Please note that [TPS\_MANI\_01047] intentionally does not forbid the existence of

SwRecordLayout because the same ApplicationPrimitiveDataType of categorySTRING could rightfully be used **on both** the *AUTOSAR adaptive platform* and the *AUTOSAR classic platform*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ApplicationPrimitiveDataType** | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes | | | |
| ***Note*** | A primitive data type defines a set of allowed values.  **Tags:**atp.recommendedPackage=ApplicationDataTypes | | | |
| ***Base*** | *ARElement*, *ARObject*, *ApplicationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*,  *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.4: ApplicationPrimitiveDataType**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SwTextProps** | | | |
| ***Package*** | M2::MSR::DataDictionary::DataDefProperties | | | |
| ***Note*** | This meta-class expresses particular properties applicable to strings in variables or calibration parameters. | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| arraySize Semantics | ArraySizeSemantics  Enum | 0..1 | attr | This attribute controls the semantics of the arraysize for the array representing the string in an Implementation DataType.  It is there to support a safe conversion between ApplicationDatatype and ImplementationDatatype, even for variable length strings as required e.g. for Support of SAE J1939. |
| baseType | SwBaseType | 0..1 | ref | This is the base type of one character in the string. In particular this baseType denotes the intended encoding of the characters in the string on level of ApplicationData Type.  **Tags:**xml.sequenceOffset=30 |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SwTextProps** |  |  |  |
| swFillCharacter | Integer | 0..1 | attr | Filler character for text parameter to pad up to the maximum length swMaxTextSize.  The value will be interpreted according to the encoding specified in the associated base type of the data object, e.g. 0x30 (hex) represents the ASCII character zero as filler character and 0 (dec) represents an end of string as filler character.  The usage of the fill character depends on the arraySize Semantics.  **Tags:**xml.sequenceOffset=40 |
| swMaxTextSize | Integer | 0..1 | attr | Specifies the maximum text size in characters. Note the size in bytes depends on the encoding in the corresponding baseType.  **Stereotypes:** atpVariation **Tags:**  vh.latestBindingTime=preCompileTime xml.sequenceOffset=20 |

**Table 3.5: SwTextProps**

|  |  |
| --- | --- |
| ***Enumeration*** | **ArraySizeSemanticsEnum** |
| ***Package*** | M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes |
| ***Note*** | This type controls how the information about the number of elements in an ApplicationArrayDataType is to be interpreted. |
| ***Literal*** | ***Description*** |
| fixedSize | This means that the ApplicationArrayDataType will always have a fixed number of elements.  **Tags:**atp.EnumerationLiteralIndex=0 |
| variableSize | This implies that the actual number of elements in the ApplicationArrayDataType might vary at run-time. The value of arraySize represents the maximum number of elements in the array.  **Tags:**atp.EnumerationLiteralIndex=1 |

**Table 3.6: ArraySizeSemanticsEnum**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SwBaseType** | | | |
| ***Package*** | M2::MSR::AsamHdo::BaseTypes | | | |
| ***Note*** | This meta-class represents a base type used within ECU software.  **Tags:**atp.recommendedPackage=BaseTypes | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *BaseType*, *CollectableElement*, *Identifiable*,  *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.7: SwBaseType**

**3.3.2.2 Associative Map Data Type**

**[TPS\_MANI\_01027]**{DRAFT} **Semantics of ApplicationAssocMapDataType** dAn ApplicationAssocMapDataType represents an associative data structure,

i.e. a data structure where so-called *key*s (formalized as ApplicationAssocMapDataType.key that are in turn typed by an ApplicationDataType) are associated with *value*s (formalized as ApplicationAssocMapDataType.value that are also in turn typed by an ApplicationDataType).c*(RS\_MANI\_00016)*

**[constr\_3349]**{DRAFT} **Usage of ApplicationAssocMapDataType is limited** dThe usage of an ApplicationAssocMapDataType is limited to the context of AdaptiveApplicationSwComponentTypes and CompositionSwComponentTypes defined in the context of an Executable, i.e. such a data type shall not be used on the *AUTOSAR classic platform*.c*()*

[constr\_3349] is a formal approach to express that an ApplicationAssocMapDataType shall only be used on the *AUTOSAR adaptive platform*.

**[TPS\_MANI\_01016]**{DRAFT} **Category of ApplicationAssocMapDataType**dThe value ApplicationAssocMapDataType.category shall be set to ASSOCIATIVE\_MAP for attribute.c*(RS\_MANI\_00016)*

Figure 3.2 depicts an example of the structure of an ApplicationAssocMapDataType.

ApplicationAssocMapDataType

Key

1

Value

1

Key

2

Value

2

Key

3

Value

3

Key

4

Value

4

Key

n

Value

n

......

......

**Figure 3.2: Example ApplicationAssocMapDataType on the *AUTOSAR adaptive platform***

As can be deduced from looking at Figure 3.2, the concept of an Application-

DataType of categoryMAP shall not be confused with an ApplicationAssocMapDataType[[2]](#footnote-2).

There are a number of technical implications on the usage of an associative data structure at run-time, e.g. that the content of each *key* shall be unique within the context of the overall data structure.

On the other hand, it is totally no problem if content on the value-side contain duplicates, e.g. two unique keys are associated with values that have a completely identical content.

However, these aspects have no implication on the formal model of the ApplicationAssocMapDataType and are therefore not considered in this document.

The modeling of the ApplicationAssocMapDataType is somewhat minimalistic and motivated mainly be the fact that data types for both key and value need to be defined.

There is no assumption how the structure of an implementation of an associative map may look like. For example, in C++ (which is currently the only supported language binding on the *AUTOSAR adaptive platform*) the straightforward way to use an associative map is to utilize the container ara::core::Map (where the implementation is opaque to the client programmer).

*AtpBlueprint*

*AtpBlueprintable*

*ApplicationDataType*

*ARElement*

*AtpType*

*AutosarDataType*

*ApplicationCompositeDataType*

ApplicationPrimitiveDataType

«atpVariation»

SwDataDefProps

ApplicationRecordDataType

ApplicationArrayDataType

dynamicArraySizeProfile: String

[0..1]

+

ApplicationAssocMapDataType

*DataPrototype*

*ApplicationCompositeElementDataPrototype*

ApplicationAssocMapElement

+

value

1

+

swDataDefProps

0..1

+

key

1

«isOfType»

+

type

0..1

{

redefines atpType

}

**Figure 3.3: Formal model of ApplicationAssocMapDataType**

Figure 3.4 contains a graphical representation of an example model for ApplicationAssocMapDataType.

MyMap: StdCppImplementationDataType

category = ASSOCIATIVE\_MAP

uint16\_t:

StdCppImplementationDataType

category = VALUE

MyAssociativeMap: ApplicationAssocMapDataType

category = ASSOCIATIVE\_MAP

key:

ApplicationAssocMapElement

value:

ApplicationAssocMapElement

myDataTypeMap

keyType:

ApplicationPrimitiveDataType

category = VALUE

valueType:

ApplicationPrimitiveDataType

category = VALUE

uint8\_t:

StdCppImplementationDataType

category = VALUE

valueDataTypeMap: DataTypeMap

keyDataTypeMap: DataTypeMap

CppTemplateArgument

:

CppTemplateArgument

:

+

implementationDataType

+

value

applicationDataType

+

+

templateType

applicationDataType

+

+

templateArgument

applicationDataType

+

+

implementationDataType

type

+

+

type

+

implementationDataType

key

+

templateArgument

+

+

templateType

**Figure 3.4: Example of the model of an associative map**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ApplicationAssocMapDataType** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType | | | |
| ***Note*** | An application data type which is a map and consists of a key and a value  **Tags:**  atp.Status=draft  atp.recommendedPackage=ApplicationDataTypes | | | |
| ***Base*** | *ARElement*, *ARObject*, *ApplicationCompositeDataType*, *ApplicationDataType*, *AtpBlueprint*, *Atp*  *Blueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| key | ApplicationAssocMap  Element | 1 | aggr | Key element of the map that is used to uniquely identify the value of the map.  **Tags:**atp.Status=draft |
| value | ApplicationAssocMap  Element | 1 | aggr | Value element of the map that stores the content associated to a key.  **Tags:**atp.Status=draft |

**Table 3.8: ApplicationAssocMapDataType**

|  |  |
| --- | --- |
| ***Class*** | **ApplicationAssocMapElement** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType |
| ***Note*** | Describes the properties of the elements of an application map data type.  **Tags:**atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ApplicationAssocMapElement** | | | |
| ***Base*** | *ARObject*, *ApplicationCompositeElementDataPrototype*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.9: ApplicationAssocMapElement**

Listing 3.1 provides the corresponding ARXML serialization of the example model of an ApplicationAssocMapDataType depicted in Figure 3.4.

**Listing 3.1: Example for the definition of an ApplicationAssocMapDataType**

**<APPLICATION-ASSOC-MAP-DATA-TYPE>**

**<SHORT-NAME>**MyAssociativeMap**</SHORT-NAME>**

**<KEY>**

**<SHORT-NAME>**MyKey**</SHORT-NAME>**

**<TYPE-TREF DEST=**"APPLICATION-PRIMITIVE-DATA-TYPE"**>**keyType**</TYPE-TREF>**

**</KEY>**

**<VALUE>**

**<SHORT-NAME>**MyValue**</SHORT-NAME>**

**<TYPE-TREF DEST=**"APPLICATION-PRIMITIVE-DATA-TYPE"**>**valueType**</TYPE-TREF> </VALUE>**

**</APPLICATION-ASSOC-MAP-DATA-TYPE>**

**<APPLICATION-PRIMITIVE-DATA-TYPE>**

**<SHORT-NAME>**keyType**</SHORT-NAME>**

**<CATEGORY>**VALUE**</CATEGORY>**

**</APPLICATION-PRIMITIVE-DATA-TYPE>**

**<APPLICATION-PRIMITIVE-DATA-TYPE>**

**<SHORT-NAME>**valueType**</SHORT-NAME>**

**<CATEGORY>**VALUE**</CATEGORY>**

**</APPLICATION-PRIMITIVE-DATA-TYPE>**

The initialization of an ApplicationAssocMapDataType, however, needs to be clarified because it would (using a combination of RecordValueSpecification and ArrayValueSpecification) in general be technically possible to define a number of differently structured ValueSpecifications that are semantically identical.

In order to keep this element of uncertainty out of the AUTOSAR standard, the initialization of a DataPrototype typed by ApplicationAssocMapDataType is clarified by means of [constr\_1488].

**[constr\_1488]**{DRAFT} **Initialization of a DataPrototype typed by an ApplicationAssocMapDataType** dA DataPrototype typed by an ApplicationAssocMapDataType shall only be initialized by an ApplicationAssocMapValueSpecification.c*()*

*ValueSpecification*

shortLabel: Identifier

+

[0..1]

*ARElement*

ConstantSpecification

*CompositeValueSpecification*

ApplicationAssocMapValueSpecification

ApplicationAssocMapElementValueSpecification

+

key

1

+

value

1

+

mapElementTuple

0..\* {

ordered

}

+

valueSpec

0..1

**Figure 3.5: Formal model of the initialization of an ApplicationAssocMapDataType**

As already mentioned, there is a semantic requirement that the *key* elements of an *associative map* need to the unique in the context of one *associative map* container.

Obviously, the model has no influence on what happens at run-time. On the other hand, there is an implication onto the initialization of an ApplicationAssocMapDataType, see [constr\_1489].

**[constr\_1489]**{DRAFT} **Uniqueness of ApplicationAssocMapValueSpecification.mapElementTuple.key**dThe value of all mapElementTuple.key elements in the context of a given ApplicationAssocMapValueSpecification shall be unique.c*()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ApplicationAssocMapValueSpecification** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType | | | |
| ***Note*** | This meta-class represents the ability to define the initialization of an ApplicationAssocMapDataType.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *CompositeValueSpecification*, *ValueSpecification* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| mapElement  Tuple (ordered) | ApplicationAssocMap  ElementValue Specification | \* | aggr | This aggregation represents the initial values for the elements of the ApplicationAssocMapValueSpecification.  **Tags:**atp.Status=draft |

**Table 3.10: ApplicationAssocMapValueSpecification**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ApplicationAssocMapElementValueSpecification** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType | | | |
| ***Note*** | This meta-class represents the ability to define the initialization of the elements of an ApplicationAssoc MapDataType.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| key | ValueSpecification | 1 | aggr | This aggregation represents the initialization of the key part of an AssociativeElementValueSpecification.  **Tags:**atp.Status=draft |
| value | ValueSpecification | 1 | aggr | This aggregation represents the initialization of the value part of an AssociativeElementValueSpecification.  **Tags:**atp.Status=draft |

**Table 3.11: ApplicationAssocMapElementValueSpecification**

**3.3.2.3 Attributes of SwDataDefProps**

**[constr\_1478]**{DRAFT}**SwDataDefProps applicable to ApplicationDataTypes exclusive to the *AUTOSAR adaptive platform*** dA complete list of the SwDataDefProps and other attributes and their multiplicities which are allowed for a given category is shown in table 3.12.c*()*

A consequence of [constr\_1478] is that the Table 3.12 shows only the values of category that are limited to the *AUTOSAR adaptive platform*. For all other values of category that are also supported on the *AUTOSAR classic platform* please refer to a similar table contained in the specification of the Software Component Template [1].

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes of SwDataDefProps** | **Root Elem.** | | **Attribute Existence per Category** |
|  | **ApplicationAssocMapDataType** | **ApplicationAssocMapElement** | **ASSOCIATIVE\_MAP** |
| **additionalNativeTypeQualifier** |  |  |  |
| **annotation** | x | x | \* |
| **baseType** |  |  |  |
| **compuMethod** |  |  |  |
| **dataConstr** |  |  |  |

5

4

|  |  |  |  |
| --- | --- | --- | --- |
| **displayFormat** | x | x | 0..1 |
| **implementationDataType** |  |  |  |
| **invalidValue** |  |  |  |
| **stepSize** |  |  |  |
| **swAddrMethod** |  |  |  |
| **swAlignment** |  |  |  |
| **swBitRepresentation** |  |  |  |
| **swCalibrationAccess** |  |  |  |
| **swCalprmAxisSet** |  |  |  |
| **swComparisonVariable** |  |  |  |
| **swDataDependency** |  |  |  |
| **swHostVariable** |  |  |  |
| **swImplPolicy** |  |  |  |
| **swIntendedResolution** |  |  |  |
| **swInterpolationMethod** |  |  |  |
| **swIsVirtual** |  |  |  |
| **swPointerTargetProps** |  |  |  |
| **swRecordLayout** |  |  |  |
| **swRefreshTiming** |  |  |  |
| **swTextProps** |  |  |  |
| **swValueBlockSize** |  |  |  |
| **unit** |  |  |  |
| **valueAxisDataType** |  |  |  |
| **Other Attributes below the Root Element** | | | |
| **key: ApplicationAssocMapElement** | x |  | 1 |
| **value: ApplicationAssocMapElement** | x |  | 1 |

**Table 3.12: Allowed Attributes vs. category for ApplicationDataTypes**

**3.3.3 CppImplementationDataType**

**3.3.3.1 Overview**

In the AUTOSAR standard, data types represent assets of paramount prominence for the entire development approach.

Therefore, AUTOSAR implements[[3]](#footnote-3) a multi-level approach for the modeling of data types. One of the described levels, the so-called *Implementation Data Level* aims at a modeling on a level that could be described as “language binding” in the parlor of the *AUTOSAR adaptive platform*.

For the *AUTOSAR classic platform*, the *Implementation Data Level* has been addressed by the creation of the ImplementationDataType that specifically aims at covering the data type behavior of the C programming language.

In contrast to the *AUTOSAR classic platform*, the *AUTOSAR adaptive platform* currently does not foresee the usage of the C language and instead (at least for the foreseeable future) defines language binding to the C++ language.

It is therefore necessary to provide a modeling approach on the *Implementation Data Level* with a proper support for the capabilities of the C++ language.

While it would technically be feasible to extend the semantics of ImplementationDataType for a support of a C++ language binding this would significantly water down the clarity and expressiveness of ImplementationDataType[[4]](#footnote-4).

It therefore seems reasonable to add a system of meta-classes that specifically supports the usage of data types with an intended binding to the C++ language.

**[TPS\_MANI\_01166]**{DRAFT} **Semantics of CppImplementationDataType** dThe abstract meta-class CppImplementationDataType supports the modeling of data types specifically tailored towards a support for a C++ language binding.c*(RS\_MANI\_00039)*

**[TPS\_MANI\_03197]**{DRAFT} **Semantics of StdCppImplementationDataType** dMeta-class StdCppImplementationDataType supports the modeling of data types that will be mapped to C++ Standard Library features in the C++ language binding.c*(RS\_MANI\_00039)*

Please note that Structures (category = STRUCTURE) and type aliases (category

= TYPE\_REFERENCE) are also modeled as StdCppImplementationDataTypes for simplification reasons.

**[TPS\_MANI\_03198]**{DRAFT} **Semantics of CustomCppImplementationDataType** dMeta-class CustomCppImplementationDataType supports the modeling of data types that will mapped to a custom implementation in the C++ language binding that is declared in the headerFile.c*(RS\_MANI\_00039)*

Please note that the category values for a CustomCppImplementationDataType are restricted by [constr\_1578].

*AbstractImplementationDataType*

*CppImplementationDataTypeContextTarget*

*CppImplementationDataType*

+

headerFile: String

[0..1]

+

typeEmitter: NameToken

[0..1]

«atpVariation»

+

arraySize: PositiveInteger

[0..1]

CustomCppImplementationDataType

StdCppImplementationDataType

**Figure 3.6: Specializations of CppImplementationDataType**

This means that the modeling of primitive data types and strings is only possible with StdCppImplementationDataTypes. The reason is that the serialization rules that are defined in AUTOSAR for SOME/IP and DDS are based on the defined types of the standard library.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***CppImplementationDataType*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType | | | |
| ***Note*** | This meta-class represents the way to specify a reusable data type definition taken as a the basis for a C++ language binding **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*,  *AtpType*, *AutosarDataType*, *CollectableElement*, *CppImplementationDataTypeContextTarget*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Subclasses*** | CustomCppImplementationDataType, StdCppImplementationDataType | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| arraySize | PositiveInteger | 0..1 | attr | This attribute can be used to specify the array size if the enclosing CppImplementationDataType has array semantics.  **Stereotypes:** atpVariation **Tags:**  atp.Status=draft  vh.latestBindingTime=preCompileTime |
| headerFile | String | 0..1 | attr | Configuration of the Header File with the custom class declaration.  **Tags:**atp.Status=draft |
| namespace  (ordered) | SymbolProps | \* | aggr | This aggregation allows for the definition an own namespace for the enclosing CppImplementationData Type.  **Tags:**atp.Status=draft |
| subElement (ordered) | CppImplementation  DataTypeElement | \* | aggr | This represents the collection of sub-elements of the enclosing CppImplementationDataType  **Tags:**atp.Status=draft |
| template Argument  (ordered) | CppTemplateArgument | \* | aggr | This aggregation allows for the specification of properties of template arguments **Tags:**atp.Status=draft |
| typeEmitter | NameToken | 0..1 | attr | This attribute can be taken to control how the respective CppImplementationDataType is contributed to the language binding.  **Tags:**atp.Status=draft |
| typeReference | CppImplementation  DataType | 0..1 | ref | This reference shall be defined to define a type reference (a.k.a. typedef).  **Tags:**atp.Status=draft |

**Table 3.13: CppImplementationDataType**

|  |  |
| --- | --- |
| ***Class*** | **StdCppImplementationDataType** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType |
| ***Note*** | This meta-class represents the way to specify a data type definition that is taken as the basis for a C++ language binding to a C++ Standard Library feature.  **Tags:**  atp.Status=draft  atp.recommendedPackage=CppImplementationDataTypes |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **StdCppImplementationDataType** | | | |
| ***Base*** | *ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*,  *AtpType*, *AutosarDataType*, *CollectableElement*, *CppImplementationDataType*, *CppImplementationData*  *TypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.14: StdCppImplementationDataType**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CustomCppImplementationDataType** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType | | | |
| ***Note*** | This meta-class represents the way to specify a data type definition that is taken as the basis for a C++ language binding to a custom implementation that is declared in the configured header file. The Short Name of this CustomCppImplementationDataType defines the Class-Name of the custom implementation.  **Tags:**  atp.Status=draft  atp.recommendedPackage=CppImplementationDataTypes | | | |
| ***Base*** | *ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*,  *AtpType*, *AutosarDataType*, *CollectableElement*, *CppImplementationDataType*, *CppImplementationData*  *TypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.15: CustomCppImplementationDataType**

**[constr\_1571]**{DRAFT} **CppImplementationDataType is limited** dThe usage of a CppImplementationDataType is limited to the context of AdaptiveApplicationSwComponentTypes and CompositionSwComponentTypes defined in the context of an Executable.c*()*

**[TPS\_MANI\_01167]**{DRAFT} **AbstractImplementationDataType** dMeta-class CppImplementationDataType inherits from abstract base class AbstractImplementationDataType in order to become a valid target for specific references from other meta-classes that want to refer to “ImplementationDataType in general”.c *(RS\_MANI\_00039)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***AbstractImplementationDataType*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes | | | |
| ***Note*** | This meta-class represents an abstract base class for different flavors of ImplementationDataType. | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*,  *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Subclasses*** | *CppImplementationDataType*, ImplementationDataType | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.16: AbstractImplementationDataType**

A prominent example for the idea of referring to “ImplementationDataType in general” can be found in meta-class DataTypeMap. The intention behind the existence of DataTypeMap is to map an ApplicationDataType to either an ImplementationDataType or CppImplementationDataType.

By means of modeling the reference DataTypeMap.implementationDataType as a reference to AbstractImplementationDataType both options are possible in a single role.

In contrast to the C language, C++ supports the definition of namespaces in programs. This feature is also cleared for development on the *AUTOSAR adaptive platform* and therefore needs to be represented in the modeling approach.

**[TPS\_MANI\_01168]**{DRAFT} **Specification of a namespace for a CppImplementationDataType** dThe ability to define a namespace for a CppImplementationDataType is expressed by means of the aggregation of SymbolProps at CppImplementationDataType in the role namespace.c*(RS\_MANI\_00039)*

**[constr\_3443]**{DRAFT} **Specification of a namespace for a StdCppImplementationDataType** dThe definition of a namespace for a StdCppImplementationDataType of category VALUE is not allowed. For this value of category the std namespace is already assumed by the usage of the StdCppImplementation-

DataType.c*()*

*AbstractImplementationDataType*

*CppImplementationDataTypeContextTarget*

*CppImplementationDataType*

+

headerFile: String

[0..1]

typeEmitter: NameToken

[0..1]

+

«atpVariation»

+

arraySize: PositiveInteger

[0..1]

*AbstractImplementationDataTypeElement*

*CppImplementationDataTypeContextTarget*

CppImplementationDataTypeElement

[0..1]

isOptional: Boolean

+

SymbolProps

*Referrable*

*ImplementationProps*

+

[0..1]

symbol: CIdentifier

*ARElement*

Allocator

[0..1]

headerFile: String

+

CppTemplateArgument

+

[0..1]

category: CategoryString

inplace: Boolean

+

[0..1]

CppImplementationDataTypeElementQualifier

[0..1]

inplace: Boolean

+

allocator

+

0..1

+

typeReference

0..1

0..1

+

typeReference

+

templateType

0..1

+

templateArgument

0..\* {

ordered

}

namespace

+

0..\* {

}

ordered

subElement

+

0..\*

ordered

}

{

typeReference

+

1

+

namespace

}

ordered

0..\* {

**Figure 3.7: CppImplementationDataType overview**

**[TPS\_MANI\_01309]**{DRAFT} **Semantics of attribute CppImplementation-**

**DataType.headerFile** dThe attribute CppImplementationDataType.headerFile shall be used to specify the name of the corresponding header file in two cases:

* A CustomCppImplementationDataType shall set the value of the attribute to the name of the header file that defines the C++ code for the CustomCppImplementationDataType.
* A platform data type (modeled as a StdCppImplementationDataType) shall set the attribute to the name of the applicable header file (e.g. "cstdint") from the C++ standard library.

c*(RS\_MANI\_00016)*

**[constr\_1743]**{DRAFT} **CppImplementationDataType.headerFile vs. CppImplementationDataType.typeEmitter**dThe two attributes CppImplementationDataType.headerFile and CppImplementationDataType.typeEmitter shall always be used mutually exclusive.

In other words, a subclass of CppImplementationDataType shall either use headerFile or typeEmitter. The simultaneous usage of both attributes is not supported.c*()*

**[TPS\_MANI\_01176]**{DRAFT} **Standardized value for attribute CppImplementationDataType.typeEmitter** dThe AUTOSAR Standard reserves the following value for attribute CppImplementationDataType.typeEmitter:

* TYPE\_EMITTER\_ARA
* FUNDAMENTAL\_TYPE: this value is only applicable for the platform types bool, float, and double.

c*(RS\_MANI\_00039)*

**[TPS\_MANI\_01177]**{DRAFT} **Semantics of attribute CppImplementationDataType.typeEmitter** dThe following set of rules applies for the usage of the attribute CppImplementationDataType.typeEmitter:

* If the attribute typeEmitter is set to the value TYPE\_EMITTER\_ARA, the ARA generator shall generate the corresponding data type definition.
* If the attribute typeEmitter is set to any value other than TYPE\_EMITTER\_ARA, the ARA generator shall silently **not** generate the corresponding data type definition.

c*(RS\_MANI\_00039)*

In the context of [TPS\_MANI\_01177], [TPS\_MANI\_01309] and [constr\_1743] apply.

**[TPS\_MANI\_01212]**{DRAFT} **Usage of attribute typeEmitter in the context of a**

**CustomCppImplementationDataType** dAttribute typeEmitter does not have to be used in the context of a CustomCppImplementationDataType. If the typeEmitter is used regardless then the value of the attribute shall be set to the name of the header file that contains the language binding of the respective CustomCppImplementationDataType.c*(RS\_MANI\_00039)*

**[TPS\_MANI\_01169]**{DRAFT} **Support for template data types** dMeta-class CppImplementationDataType supports the usage of templates for the definition of data types in C++ programs by means of the reference CppImplementationDataType. templateArgument.

The order of arguments in templates is significant, therefore templateArgument is modeled as an **ordered** collection.c*(RS\_MANI\_00039)*

**[TPS\_MANI\_01174]**{DRAFT} **Semantics of reference in the role CppTemplateArgument.templateType** dAttribute CppTemplateArgument.templateType specifies the data type to be filled in the respective position of the template in the language binding.c*(RS\_MANI\_00039)*

**[TPS\_MANI\_01175]**{DRAFT} **Semantics of reference in the role CppTemplateArgument.allocator** dAttribute CppTemplateArgument.allocator specifies the behavior of an allocator class to be filled in the respective position of the template in the language binding.c*(RS\_MANI\_00039)*

**[constr\_1576]**{DRAFT} **Existence of CppTemplateArgument.templateType vs. CppTemplateArgument.allocator** dFor any given CppTemplateArgument, **at most one of** the references

* CppTemplateArgument.templateType or
* CppTemplateArgument.allocator

**may** exist.c*()*

**[TPS\_MANI\_01201]**{DRAFT} **Standardized values for attribute CppTemplateArgument.category** dAUTOSAR reserves the following values for attribute CppTemplateArgument.category:

**ASSOC\_MAP\_KEY** : the specific CppTemplateArgument represents the *key* datatype of an associative map.

**ASSOC\_MAP\_VALUE** : the specific CppTemplateArgument represents the *value* data-type of an associative map.

c*(RS\_MANI\_00039)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CppTemplateArgument** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType | | | |
| ***Note*** | This meta-class has the ability to define properties for template arguments.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CppTemplateArgument** |  |  |  |
| allocator | Allocator | 0..1 | ref | This reference identifies the applicable allocator.  **Tags:**atp.Status=draft |
| category | CategoryString | 0..1 | attr | This attribute shall be used to contribute further clarification regarding the semantics of the enclosing Cpp TemplateArgument.  **Tags:**atp.Status=draft |
| inplace | Boolean | 0..1 | attr | This attribute specifies whether the shortName of the referenced templateType is used in the code generation and the type declaration is defined outside of the enclosing CppImplementationDataType (true) or whether the type definition is embedded inside of the enclosing CppImplementationDataType and the shortName is ignored (false).  **Tags:**atp.Status=draft |
| templateType | CppImplementation  DataType | 0..1 | ref | This reference identifies the data type of the specific template argument required for the language binding.  **Tags:**atp.Status=draft |

**Table 3.17: CppTemplateArgument**

**[TPS\_MANI\_01171]**{DRAFT} **Modeling of structured data types** dMeta-class CppImplementationDataType supports the creation of nested data types by means of the aggregation of CppImplementationDataTypeElement in the role subElement.

Because the order of sub-elements in a structured data type is significant the aggregation subElement is modeled as an **ordered** collection.c*(RS\_MANI\_00039)*

Please note that although the modeling of structures is formally done by way of using

CppImplementationDataType it is actually only possible to use StdCppImplementationDataType for this purpose (see [constr\_1578]).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CppImplementationDataTypeElement** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType | | | |
| ***Note*** | Declares a data object which is locally aggregated. Such an element can only be used within the scope where it is aggregated. A CppImplementationDataTypeElement is used to represent an element of a structure, defining its type.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *AbstractImplementationDataTypeElement*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CppImplementationDataTypeElement** | | |  |
| isOptional | Boolean | 0..1 | attr | This attribute represents the ability to declare the enclosing CppImplementationDataTypeElement as optional. This means the that, at runtime, the Cpp ImplementationDataTypeElement may or may not have a valid value and shall therefore be ignored.  The underlying runtime software provides means to set the CppImplementationDataTypeElement as not valid at the sending end of a communication and determine its validity at the receiving end.  **Tags:**atp.Status=draft |
| typeReference | CppImplementation  DataTypeElement  Qualifier | 0..1 | aggr | This aggregation defines the type of the Cpp ImplementationDataTypeElement and determines whether in C++ the CppImplementationDataTypeElement  is defined inside or outside of the enclosing Cpp ImplementationDataType.  **Tags:**atp.Status=draft |

**Table 3.18: CppImplementationDataTypeElement**

Please note that there is no intention to support a “mixed” modeling of structured data types such that the resulting data type on C++ level would be composed of data types that are native to C++ and data types from the C subsystem.

While this would technically be possible on code level it would impose a huge effort on modeling level and the consensus is that there is no real use case for such a "mixed" data type.

The C++ data type system can, as far as the implementation of the *AUTOSAR adaptive platform* is concerned, fully replace the “legacy” C data types in C++.

**[constr\_1572]**{DRAFT} **Usage of SwDataDefProps.implementationDataType within a CppImplementationDataType**dWithin the scope of a CppImplementationDataType the reference CppImplementationDataType.swDataDefProps. implementationDataType **shall not exist**.c*()*

This aspect is also expressed in a more general form by [constr\_1579].

As a consequence of [constr\_1572], type-references have to be done differently on the *AUTOSAR adaptive platform*. For this purpose dedicated references are available.

**[TPS\_MANI\_01172]**{DRAFT} **Description of type references in the scope of**

**CppImplementationDataType** dThe reference CppImplementationDataType. typeReference can be used to create a type reference from the enclosing CppImplementationDataType to another CppImplementationDataType.c*(RS\_MANI\_00039)*

**[TPS\_MANI\_01173]**{DRAFT} **Description of type references in the scope of**

**CppImplementationDataTypeElement** dCppImplementationDataTypeElement.typeReference can be used to create a reference to the CppImplementationDataType that shall apply for the enclosing CppImplementationDataType-

Element.c*(RS\_MANI\_00039)*

Please note that the CppImplementationDataTypeElement.typeReference is realized as an Association Class that allows to add the inplace attribute to the typeReference.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CppImplementationDataTypeElementQualifier** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType | | | |
| ***Note*** | This element qualifies the typeReference of the CppImplementationDataTypeElement to the Cpp ImplementationDataType.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| inplace | Boolean | 0..1 | attr | This attribute defines whether the member type of the CppImplementationDataTypeElement in C++ is an embedded type element inside of the enclosing struct (true) or whether the type declaration is defined outside of the struct.  **Tags:**atp.Status=draft |
| typeReference | CppImplementation  DataType | 1 | ref | This reference defines a type reference.  **Tags:**atp.Status=draft |

**Table 3.19: CppImplementationDataTypeElementQualifier**

**[TPS\_MANI\_03196]**{DRAFT} **Semantics of CppImplementationDataTypeElementQualifier.inplace attribute** dThe CppImplementationDataTypeElementQualifier.inplace attribute defines whether the data type of the CppImplementationDataTypeElement in the C++ language binding is derived from the name or the properties of the referenced CppImplementationDataType.

Specifically, the following rules shall apply:

* if CppImplementationDataTypeElement.typeReference.inplace is set to False then the **shortName** of the CppImplementationDataType referenced in the role CppImplementationDataTypeElement.typeReference. typeReference shall be used in the C++ language binding.
* if CppImplementationDataTypeElement.typeReference.inplace is set to True then only the **properties** of the CppImplementationDataType referenced in the role CppImplementationDataTypeElement.typeReference. typeReference shall be used in the C++ language binding and the shortName is ignored.

c*(RS\_MANI\_00039)*

Please note that Figure 3.13 shows an example of a Structure where the typeReference of one subElement is classified as inplace.

**[constr\_1659]**{DRAFT} **Restriction for the usage of CppImplementation-**

**DataTypeElementQualifier.inplace** dThe attribute CppImplementationDataTypeElementQualifier.inplace shall only exist if the target referenced in the role CppImplementationDataTypeElementQualifier.typeReference is an StdCppImplementationDataType that has the attribute category set to either of the values

* ARRAY
* VECTOR
* ASSOCIATIVE\_MAP
* VARIANT
* STRUCTURE
* STRING
* TYPE\_REFERENCE, if the CppImplementationDataType refers to a CompuMethod of categoryTEXTTABLE

### c()

Rationale for the existence of [constr\_1659]: by application of the exclusion principle, there are three cases where attribute CppImplementationDataTypeElementQualifier.inplace shall not exist:

* StdCppImplementationDataType of categoryVALUE
* CustomCppImplementationDataType
* CppImplementationDataType of categoryTYPE\_REFERENCE, unless the CppImplementationDataType refers to a CompuMethod of category TEXTTABLE

Neither of them can be used as a target of CppImplementationDataTypeElementQualifier.typeReference where CppImplementationDataTypeElementQualifier.inplace is set to True because in these cases there is already a valid name that is directly usable for the language binding and a possible indirection via a using clause would obviously require an additional name that is not available from the model.

After all, the motivation for the definition of a TYPE\_REFERENCE is the direct opposite of the motivation behind using the attribute CppImplementationDataTypeElementQualifier.inplace to control the language binding. Therefore, this case is also excluded.

**[TPS\_MANI\_03201]**{DRAFT} **Semantics of CppTemplateArgument.inplace attribute** dThe CppTemplateArgument.inplace attribute defines whether the data type that is referenced by the templateType in the C++ language binding is derived from the name or the properties of the referenced CppImplementationDataType.

Specifically, the following rules shall apply:

* if CppTemplateArgument.inplace is set to False then the **shortName** of the CppImplementationDataType referenced in the role CppTemplateArgument.templateType shall be used in the C++ language binding.
* if CppTemplateArgument.inplace is set to True then only the **properties** of the CppImplementationDataType referenced in the role CppTemplateArgument.templateType shall be used in the C++ language binding and the shortName is ignored.

c*(RS\_MANI\_00039)*

**[constr\_1660]**{DRAFT} **Restriction for the usage of CppTemplateArgument.inplace** dThe attribute CppTemplateArgument.inplace shall only exist if the target referenced in the role CppTemplateArgument.templateType is an StdCppImplementationDataType that has the attribute category set to either of the values

* ARRAY
* VECTOR
* ASSOCIATIVE\_MAP
* VARIANT
* STRUCTURE
* STRING

### c()

Rationale for the existence of [constr\_1660]: by application of the exclusion principle, there are three cases where attribute CppTemplateArgument.inplace shall not exist:

* StdCppImplementationDataType of categoryVALUE
* CustomCppImplementationDataType
* CppImplementationDataType of categoryTYPE\_REFERENCE

Neither of them can be used as a target of CppTemplateArgument.templateType where CppTemplateArgument.inplace is set to True because in these cases there is already a valid name that is directly usable for the language binding and a possible indirection via a using clause would obviously require an additional name that is not available from the model.

After all, the motivation for the definition of a TYPE\_REFERENCE is the direct opposite of the motivation behind using the attribute CppTemplateArgument.inplace to control the language binding. Therefore, this case is also excluded.

Please note that the question of the value of attribute CppTemplateArgument.inplace for the case of CppTemplateArgument.templateType referring to StdCppImplementationDataType of category STRUCTURE is regulated by [constr\_3462].

**[constr\_1708]**{DRAFT} **Combination of CppImplementationDataTypeElement.isOptional and CppImplementationDataTypeElementQualifier.inplace** dIf a CppImplementationDataTypeElement is typed by a CppImplementationDataType of categorySTRUCTURE then the combination of attribute CppImplementationDataTypeElement.isOptional set to True and CppImplementationDataTypeElement.typeReference.inplace set to True is not allowed.c*()*

Rationale for the existence of [constr\_1708]: the “optional” semantics is implemented via a template and it is not possible to pass an “inplace” structure as a template argument.

**[constr\_3462]**{DRAFT} **CppTemplateArgument.templateType reference to**

**StdCppImplementationDataType of categorySTRUCTURE and the inplace flag** dCppTemplateArgument.templateType that points to a StdCppImplementationDataType of categorySTRUCTURE shall have the inplace attribute set to **false**.c*()*

The reason for [constr\_3462] is that the usage of an unnamed struct as template argument is not permitted by ISO C++11/14/17.

**[constr\_3446]**{DRAFT} **CppTemplateArgument with allocator reference and the inplace flag** dA CppTemplateArgument that points with an allocator reference to an Allocator shall not have the inplace flag set to a value.c*()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Allocator** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType | | | |
| ***Note*** | This meta-class represents the ability to take influence on the way objects are allocated in memory, for example it can be controlled whether an objects is allocated on the heap or on the stack.  **Tags:**  atp.Status=draft  atp.recommendedPackage=Allocators | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| headerFile | String | 0..1 | attr | Configuration of the Header File with the custom class declaration  **Tags:**atp.Status=draft |
| namespace  (ordered) | SymbolProps | \* | aggr | This aggregation allows for the definition of a namespace of an Allocator.  **Tags:**atp.Status=draft |

**Table 3.20: Allocator**

**[TPS\_MANI\_01100]**{DRAFT} **Semantics of Allocator** dMeta-class Allocator carries the ability to define the properties of an allocation of memory. The general approach for memory allocation is expressed by means of the attribute category.

The following values of Allocator.category are standardized by AUTOSAR:

* MAX\_SIZE\_HEAP: when using this allocator there is the intention to allocate a fixed-size chunk on the heap. This allocator adds the ability to define a maximum number of elements to the semantics of the default allocator of ara::core::Vector.
* MAX\_SIZE\_STACK: when using this allocator there is the intention to allocate a fixed-size chunk on the stack. Memory on the stack always needs to be constrained in terms of the maximum size. In other words, there is hardly any case where an unbounded amount of memory should be allocated on the stack.
* MAX\_SIZE\_DATASEGMENT: when using this allocator there is the intention to allocate a fixed-size chunk in the data segment.

c*(RS\_MANI\_00016)*

**[constr\_1578]**{DRAFT} **applicable data categories** dTable 3.21 defines the applicable categorys vs. meta-class.c*()*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Category** | **Applicable to ...** | | | | | | | | **Description** |
|  | **ApplicationArrayDataType** | **ApplicationRecordDataType** | **ApplicationPrimitiveDataType** | **ApplicationRecordElement** | **ApplicationArrayElement** | **ApplicationValueSpecification** | **StdCppImplementationDataType** | **CustomCppImplementationDataType** |  |
| **VALUE** |  |  | x | x | x | x | x |  | Contains a single value. See also [TPS\_MANI\_03192]. |
| **TYPE\_REFERENCE** |  |  |  |  |  |  | x |  | The element is defined via reference to another data type (via CppImplementationDataType.typeReference. |
| **STRUCTURE** |  | x |  | x | x |  | x |  | Holds one or several further elements which can have different AutosarDataTypes. See also [TPS\_MANI\_03180]. |
| **VARIANT** |  |  |  |  |  |  | x | x | Can hold values of different data types. It is similar to STRUCTURE  except that all of its members start at the same location in memory.  A VARIANT data prototype can contain only one of its elements at a time and represents a type-safe union. The size of the VARIANT is at least the size of the largest member. See also [TPS\_MANI\_03189]. |
| **ARRAY** | x |  |  | x | x |  | x | x | A fixed-sized array of sub-elements of the same data type. See also [TPS\_MANI\_03169]. |
| **VECTOR** |  |  |  |  |  |  | x | x | An array of elements of the same data type that is able to grow at run-time. See also [TPS\_MANI\_03174]. |
| **ASSOCIA-**  **TIVE\_MAP** |  |  |  |  |  |  | x | x | An associative array of key-value pairs. See also [TPS\_MANI\_03183]. |
| **STRING** |  |  | x | x | x | x | x |  | Contains a text string. See also [TPS\_MANI\_03178]. |
| **BOOLEAN** |  |  | x | x | x | x |  |  | Contains one boolean state. Depending on the CPU direct addressing of single bits may not be available.  So a byte or a word can be used to store only one logical state. |

**Table 3.21: Usage of category for Data Types**

**3.3.3.2 Attributes of SwDataDefProps**

**[constr\_1579]**{DRAFT} **SwDataDefProps applicable to CppImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*** dA complete list of the SwDataDefProps and other attributes and their multiplicities which are allowed for a given category is shown in table 3.22.c*()*

A consequence of [constr\_1578] is that the Table 3.22 shows only the values of category that are limited to the *AUTOSAR adaptive platform*. For all other values of category that are also supported on the *AUTOSAR classic platform* please refer to a similar table contained in the specification of the Software Component Template [1].

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Attributes of SwDataDefProps** | **Root**  **Ele-**  **ment** | **Attribute Existence per Category** | | | | | | | |
|  | **CppImplementationDataType** | **VALUE** | **TYPE\_REFERENCE** | **STRUCTURE** | **VARIANT** | **ARRAY** | **VECTOR** | **ASSOCIATIVE\_MAP** | **STRING** |
| **additionalNativeTypeQualifier** |  |  |  |  |  |  |  |  |  |
| **annotation** | x | \* | \* | \* | \* | \* | \* | \* | \* |
| **baseType** |  |  |  |  |  |  |  |  |  |
| **compuMethod** | x |  | 0..1 |  |  |  |  |  |  |
| **dataConstr.dataConstrRule.physConstrs** | x |  | d/c |  |  | d/c | d/c |  |  |
| **dataConstr.dataConstrRule.internalConstrs** | x |  | 0..1 |  |  | 0..1 | 0..1 |  |  |
| **displayFormat** | x | 0..1 | 0..1 | 0..1 | 0..1 | 0..1 | 0..1 | 0..1 | 0..1 |
| **implementationDataType** |  |  |  |  |  |  |  |  |  |
| **invalidValue** | x |  | 0..1 |  |  |  |  |  | 0..1 |
| **stepSize** |  |  |  |  |  |  |  |  |  |
| **swAddrMethod** |  |  |  |  |  |  |  |  |  |
| **swAlignment** |  |  |  |  |  |  |  |  |  |
| **swBitRepresentation** |  |  |  |  |  |  |  |  |  |
| **swCalibrationAccess** |  |  |  |  |  |  |  |  |  |
| **swCalprmAxisSet** |  |  |  |  |  |  |  |  |  |
| **swComparisonVariable** |  |  |  |  |  |  |  |  |  |
| **swDataDependency** |  |  |  |  |  |  |  |  |  |
| **swHostVariable** |  |  |  |  |  |  |  |  |  |
| **swImplPolicy** |  |  |  |  |  |  |  |  |  |
| **swIntendedResolution** |  |  |  |  |  |  |  |  |  |
| **swInterpolationMethod** |  |  |  |  |  |  |  |  |  |
| **swIsVirtual** |  |  |  |  |  |  |  |  |  |
| **swPointerTargetProps** |  |  |  |  |  |  |  |  |  |
| **swPointerTargetProps**.**swDataDefProps** |  |  |  |  |  |  |  |  |  |
| **swRecordLayout** |  |  |  |  |  |  |  |  |  |

5

4

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Attributes of SwDataDefProps** | **Root**  **Ele-**  **ment** |  | **Attribute Existence per Category** | | | | | |  |
|  | **CppImplementationDataType** | **VALUE** | **TYPE\_REFERENCE** | **STRUCTURE** | **VARIANT** | **ARRAY** | **VECTOR** | **ASSOCIATIVE\_MAP** | **STRING** |
| **swRefreshTiming** | x | 0..1 | 0..1 | 0..1 | 0..1 | 0..1 | 0..1 | 0..1 | 0..1 |
| **swTextProps** |  |  |  |  |  |  |  |  |  |
| **swValueBlockSize** |  |  |  |  |  |  |  |  |  |
| **unit** |  |  |  |  |  |  |  |  |  |
| **valueAxisDataType** |  |  |  |  |  |  |  |  |  |
| **Other Attributes** | | |  | | | | | |  |
| **subElement: CppImplementationDataTypeElement** | x |  |  | 1..\* |  |  |  |  |  |
| **templateArgument** | x |  |  |  | 1..\* | 1 | 1..\* | 2..\* | 0..1 |
| **typeReference** | x |  | 1 |  |  |  |  |  |  |

**Table 3.22: Allowed Attributes vs. category for CppImplementationDataType**

The invalidValue is applicable to Primitive Data Types and defines one specific value (in the range of that Primitive Data Type) which indicates that the respective value is not valid.

A typical use case is a composite data type that contains the values of all 4 wheel speeds. If one of the wheel speed sensors fails, and is no longer able to provide useful data, it does still make sense to provide the other 3 wheel speed values.

In such a scenario the one wheel speed value would then be set to the invalidValue. The receivers are able to check for each individual element of the data composition whether the value corresponds to the invalidValue and take corresponding actions.

**[constr\_3569]**{DRAFT} **Applicability of attribute invalidValue on CppImplementationDataType of category TYPE\_REFERENCE** dIf a CppImplementationDataType of category TYPE\_REFERENCE has an invalidValue defined, then the referenced CppImplementationDataType (via typeReference) shall eventually be of categoryVALUE.c*()*

Please note that the following rationale exists for the support of invalidValue for specific categorys of data types:

* The usage of categoryVALUE on the *AUTOSAR adaptive platform* boils down to the usage of the standard types. There is no use case to define an invalidValue for a standard data type because **all usages** of the standard data type would be characterized by the same invalidValue.
* The definition of an invalidValue for a container (except STRING) is not supported because there are no known use cases for supporting an invalidValue.
* The definition of an invalidValue on a data type of category STRING is accepted because it is also supported on the *AUTOSAR classic platform* and it is necessary to sustain interoperability between the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform*.
* The definition of an invalidValue on an StdCppImplementationDataType of categoryTYPE\_REFERENCE represents the main-stream use case for the definition of an invalidValue.

**3.3.3.3 Primitive Data Types**

**[TPS\_MANI\_03192]**{DRAFT} **CppImplementationDataType of category VALUE** dThe primitive data types like Boolean, fixed-width integer data types and floating-point data types are described as CppImplementationDataTypes of categoryVALUE.c*(RS\_MANI\_00039)*

**[TPS\_MANI\_03193]**{DRAFT} **CppImplementationDataType of category TYPE\_REFERENCE** dThe definition of a CppImplementationDataType of category TYPE\_REFERENCE creates an alias for another CppImplementationDataType that is referenced by the typeReference.c*(RS\_MANI\_00039)*

**3.3.3.4 String Data Type**

**[TPS\_MANI\_03178]**{DRAFT} **StdCppImplementationDataType of category**

**STRING** dA StdCppImplementationDataType of categorySTRING represents a container data type for a sequence of characters.

AUTOSAR demands that the C++ binding of a StdCppImplementationDataType of category STRING is implemented by a ara::core::String.c*(RS\_MANI\_00039)*

**[constr\_1674]**{DRAFT} **Supported encoding of StdCppImplementationDataType of categorySTRING**dOn the level of the meta-model (and, by extension, the language binding), the only supported encoding of StdCppImplementationDataType of categorySTRING is UTF-8.c*()*

Please note that it is nonetheless possible to use a different encoding, e.g. UTF-16 on the level of a SOME/IP message. This behavior can be configured by means of ApSomeipTransformationProps. As a consequence, a transcoding may have to be applied between the representation of a string on the wire and in the software.

**[TPS\_MANI\_03179]**{DRAFT} **C++ language binding of StdCppImplementation-**

**DataTypes of categorySTRING**dA CppImplementationDataType of category

STRING shall be implemented as ara::core::String.c*(RS\_MANI\_00039)*

The formulation of [TPS\_MANI\_03179] leaves room for potential later extensions towards the support for other storage formats.

The example depicted in Figure 3.8 contains the definition of both an ApplicationDataType as well as the definition of the corresponding CppImplementationDataType.

The latter obviously becomes significantly lighter to model thanks to the restriction that, as far as the C++ language binding is concerned, a CppImplementationDataType of categorySTRING shall only be implemented on the basis of an ara::core::String.

Another aspect of the example in Figure 3.8 is that it defines the intended encoding of the modeled data type in the scope of the ApplicationPrimitiveDataType.

**[TPS\_MANI\_03188]**{DRAFT} **Usage of an Allocator for a StdCppImplementationDataType of categorySTRING** dA StdCppImplementationDataType of categorySTRING is allowed to aggregate a CppTemplateArgument that refers to an Allocator with the allocator reference.c*(RS\_MANI\_00039)*

MyStringApplicationDataType:

ApplicationPrimitiveDataType

category = STRING

dataTypeMap: DataTypeMap

string:

StdCppImplementationDataType

category = STRING

+

implementationDataType

+

applicationDataType

**Figure 3.8: Example of the model of a string with UTF-8 encoding**

**3.3.3.5 Array Data Type**

**[TPS\_MANI\_03169]**{DRAFT} **CppImplementationDataType with fixed size array semantics** dA CppImplementationDataType of categoryARRAY represents a container data type that encapsulates fixed size arrays.c*(RS\_MANI\_00039)*

**[TPS\_MANI\_03170]**{DRAFT} **CppImplementationDataType of category ARRAY**dFor a C++ binding, a CppImplementationDataType of categoryARRAY can be implemented as

* an ara::core::Array if StdCppImplementationDataType subclass is used for modeling or as
* an array type in a custom namespace (e.g. my::array) if CustomCppImplementationDataType subclass is used (provided that the type in the custom namespace can be configured with the available modeling capabilities).

c*(RS\_MANI\_00039)*

**[TPS\_MANI\_03171]**{DRAFT} **Value type of a CppImplementationDataType of category ARRAY** dThe type of elements contained in a CppImplementationDataType of categoryARRAY is defined by the aggregated templateArgument and the corresponding templateType that defines the data type of the CppTemplateArgument.c*(RS\_MANI\_00039)*

**[constr\_3433]**{DRAFT} **Aggregation of templateArguments for an ARRAY** dCppImplementationDataType of categoryARRAY that boils down to ara::core::Array shall aggregate exactly one templateArgument that defines the type of elements contained in the CppImplementationDataType of categoryARRAY.c*()*

**[TPS\_MANI\_03172]**{DRAFT} **Size of a CppImplementationDataType of category ARRAY** dThe primitive attribute arraySize of a CppImplementationDataType of categoryARRAY shall be used to define the size of the array.c*(RS\_MANI\_00039)*

Figure 3.9 shows an example of an one-dimensional array of uint16 elements with arraySize = 5.

DataTypeMapping: DataTypeMap

ArrayLinear:

ApplicationArrayDataType

category = ARRAY

ArrayElementLinear:

ApplicationArrayElement

category = VALUE

maxNumberOfElements = 5

arraySizeSemantics = fixedSize

ArrayLinearElementDataType:

ApplicationPrimitiveDataType

category = VALUE

ArrayLinearImpl:

StdCppImplementationDataType

arraySize = 5

category = ARRAY

uint16\_t:

StdCppImplementationDataType

category = VALUE

CppTemplateArgument

:

templateArgument

+

+

type

applicationDataType

+

implementationDataType

+

+

element

+

templateType

**Figure 3.9: Example of the model of a one-dimensional array**

**[TPS\_MANI\_03173]**{DRAFT} **Definition of a multidimensional Array** dA multidimensional CppImplementationDataType of category ARRAY contains nested CppImplementationDataTypes of categoryARRAY.

The CppImplementationDataType of categoryARRAY that represents the outer array will refer to a CppImplementationDataType of categoryARRAY that represents the inner array via the aggregated templateArgument. Such a definition describes a two-dimensional Array; consequently a type with more dimensions is described by just nesting more CppImplementationDataTypes of categoryARRAY.

The array element itself is specified by the innermost CppImplementationDataType with category different from ARRAY.c*(RS\_MANI\_00039)*

Figure 3.10 shows an example of a multidimensional array where a CppImplementationDataType of categoryARRAY with arraySize = 5 has a templateArgument that points to the inner CppImplementationDataType of categoryARRAY in the role templateType.

The inner CppImplementationDataType has a templateArgument that finally points with the templateType reference to a primitive type.

unit16\_t: StdCppImplementationDataType

category = VALUE

ArrayLinearImpl: StdCppImplementationDataType

arraySize = 10

category = ARRAY

ArrayLinearElementDataType: ApplicationPrimitiveDataType

category = VALUE

ArrayElementRectangular: ApplicationArrayElement

category = ARRAY

maxNumberOfElements = 5

arraySizeSemantics = fixedSize

ArrayRectangular: ApplicationArrayDataType

category = ARRAY

DataTypeMapping: DataTypeMap

ArrayLinear: ApplicationArrayDataType

category = ARRAY

ArrayLinearElement: ApplicationArrayElement

category = VALUE

maxNumberOfElements = 10

arraySizeSemantics = fixedSize

ArrayRectangularImpl: StdCppImplementationDataType

arraySize = 5

category = ARRAY

CppTemplateArgument

:

inplace = True

CppTemplateArgument

:

type

+

templateArgument

+

+

templateType

+

applicationDataType

+

element

templateType

+

+

implementationDataType

+

templateArgument

+

element

type

+

**Figure 3.10: Example of the model of a multidimensional array**

**3.3.3.6 Vector Data Type**

**[TPS\_MANI\_03174]**{DRAFT} **CppImplementationDataType with variable size array semantics** dA CppImplementationDataType of categoryVECTOR represents a container data type that encapsulates variable size arrays.c*(RS\_MANI\_00039)* **[TPS\_MANI\_03175]**{DRAFT} **CppImplementationDataType of categoryVEC-**

**TOR** dFor a C++ binding, a CppImplementationDataType of categoryVECTOR can be implemented as

* an ara::core::Vector if StdCppImplementationDataType subclass is used or as
* a vector type in a custom namespace (e.g. my::vector) if CustomCppImplementationDataType subclass is used (provided that the type in the custom namespace can be configured with the available modeling capabilities).

c*(RS\_MANI\_00039)*

DynamicDataArray\_Linear: ApplicationArrayDataType

dynamicArraySizeProfile = VSA\_LINEAR

category = ARRAY

DynamicDataArrayDim1: ApplicationArrayElement

arraySizeSemantics = variableSize

maxNumberOfElements = 5

arraySizeHandling = AllIndicesSameArraySize

category = VALUE

PrimitiveDataElementType:

ApplicationPrimitiveDataType

category = VALUE

DynamicDataArrayImplLinear:

StdCppImplementationDataType

category = VECTOR

DataTypeMapping: DataTypeMap

uint16\_t: StdCppImplementationDataType

category = VALUE

:

CppTemplateArgument

element

+

implementationDataType

+

templateType

+

+

applicationDataType

+

templateArgument

+

type

**Figure 3.11: Example of the model of a one-dimensional vector**

**[TPS\_MANI\_03176]**{DRAFT} **Value type of a CppImplementationDataType of category VECTOR** dThe type of elements contained in a CppImplementationDataType of categoryVECTOR is defined by the aggregated templateArgument and the corresponding templateType that defines the data type of the CppTemplateArgument.c*(RS\_MANI\_00039)*

**[constr\_3434]**{DRAFT} **Aggregation of templateArguments for a VECTOR** dCppImplementationDataType of category VECTOR that boils down to ara::core::Vector shall aggregate

* one templateArgument that defines the type of elements contained in the CppImplementationDataType of categoryVECTOR with the templateType reference.
* optionally one additional templateArgument that defines the Allocator with the allocator reference.

### c()

**[TPS\_MANI\_03186]**{DRAFT} **Usage of arraySize in case of a Vector** dIf the CppImplementationDataType of categoryVECTOR aggregates a templateArgument that defines the Allocator with the allocator reference then the attribute arraySize that defines the maximum size of the vector is allowed to be used.c*(RS\_MANI\_00039)*

Figure 3.11 shows an example of an one-dimensional vector of uint16 elements.

DynamicDataArrayDim2Element:

ApplicationArrayElement

arraySizeSemantics = variableSize

maxNumberOfElements = 4

arraySizeHandling = AllIndicesSameArraySize

category = VALUE

PrimitiveDataElementType:

ApplicationPrimitiveDataType

category = VALUE

DynamicDataArrayImplRectangular:

StdCppImplementationDataType

category = VECTOR

templateArgumentDim1:

StdCppImplementationDataType

category = VECTOR

DynamicDataArrayDim1: ApplicationArrayElement

arraySizeSemantics = variableSize

arraySizeHandling = allIndicesDifferentArraySize

maxNumberOfElements = 3

category = ARRAY

DynamicDataArrayDim2: ApplicationArrayDataType

category = ARRAY

dynamicArraySizeProfile = VSA\_LINEAR

DataTypeMapping: DataTypeMap

uint16\_t: StdCppImplementationDataType

category = VALUE

DynamicDataArray\_Rectangular:

ApplicationArrayDataType

category = ARRAY

dynamicArraySizeProfile = VSA\_RECTANGULAR

:

CppTemplateArgument

CppTemplateArgument

:

element

+

+

implementationDataType

+

templateType

type

+

templateArgument

+

templateType

+

+

applicationDataType

+

type

+

templateArgument

+

element

**Figure 3.12: Example of the model of a multidimensional vector**

**[TPS\_MANI\_03177]**{DRAFT} **Definition of a multidimensional Vector** dA multidimensional CppImplementationDataType of categoryVECTOR contains nested CppImplementationDataTypes of categoryVECTOR.

The CppImplementationDataType of categoryVECTOR that represents the outer vector will refer to a CppImplementationDataType of categoryVECTOR that represents the inner vector via the aggregated templateArgument.

Such a definition describes a two-dimensional Vector; consequently a type with more dimensions is described by just nesting more CppImplementationDataTypes of categoryVECTOR.

The vector element itself is specified by the innermost CppImplementationDataType with category different from VECTOR.c*(RS\_MANI\_00039)*

Figure 3.12 shows an example of a multidimensional vector where a CppImplementationDataType of categoryVECTOR has a templateArgument that points to the inner CppImplementationDataType of category VECTOR in the role templateType. The inner CppImplementationDataType has a templateArgument that finally points with the templateType reference to a primitive type.

Please note that the meta-model supports the creation of a reference to a specific element (identified by means of the index) of a CppImplementationDataType of categoryVECTOR.

However, this may lead to a problem at run-time if the specific element does not exist at the respective point in time. Any software using such data types needs to be prepared for the potential non-existence of vector elements.

Alternatively, it could be an option to simply avoid a situation where an element of a CppImplementationDataType of categoryVECTOR becomes the target of a reference in the model.

**3.3.3.7 Struct Data Type**

**[TPS\_MANI\_03180]**{DRAFT} **Definition of Structures** dA StdCppImplementationDataType of categorySTRUCTURE represents a data type for holding an ordered collection of variables of arbitrary data types.c*(RS\_MANI\_00039)*

**[TPS\_MANI\_03181]**{DRAFT} **Definition of members in StdCppImplementation-**

**DataType of category STRUCTURE** dMembers in a StdCppImplementation-

DataType of categorySTRUCTURE are defined by ordered CppImplementationDataTypeElements that are aggregated in the role subElement by the enclosing StdCppImplementationDataType of categorySTRUCTURE.

The name of each member is defined by the shortName of the CppImplementationDataTypeElement.

The type of each member is defined by the typeReference to a CppImplementationDataType.c*(RS\_MANI\_00039)*

Please note that the inplace flag that is able to classify a CppImplementationDataTypeElement.typeReference is documented in [TPS\_MANI\_03196].

The example depicted in Figure 3.13 shows the definition of a Structure, called MyStruct, that has two members. The typeReference of the subElements with the shortNameArrayElement is classified with inplace = **True**.

In case that the inplace attribute in the typeReference to the array is set to **False** the model results in a using-declaration of ArrayDataType that is defined outside MyStruct.

MyStruct: CppImplementationDataType

category = STRUCTURE

PrimitiveElement:

CppImplementationDataTypeElement

ArrayElement:

CppImplementationDataTypeElement

ArrayDataType:

StdCppImplementationDataType

category = ARRAY

arraySize = 5

uint8\_t: StdCppImplementationDataType

category = VALUE

:

CppTemplateArgument

CppImplementationDataTypeElementQualifier

:

inplace = True

:

CppImplementationDataTypeElementQualifier

+

typeReference

templateArgument

+

+

templateType

subElement

+

typeReference

+

subElement

+

+

templateType

typeReference

+

**Figure 3.13: Example of the model of a Struct**

**3.3.3.8 Enumeration Data Type**

**[TPS\_MANI\_03187]**{DRAFT} **Definition of enumeration types** dIn the AUTOSAR meta-model, an enumeration is not implemented by means of a CppImplementationDataType with an own category.

Instead, a discrete set of integer numbers can be used as a structural description for a single fundamental CppImplementationDataType of categoryTYPE\_REFERENCE that boils down to a CppImplementationDataType of category VALUE.

The mapping of the integer numbers to labels in the scope of the definition of an enumeration is considered part of the semantical definition via an attached CompuMethod with categoryTEXTTABLE rather than part of the structural description.c*(RS\_MANI\_00039)*

The rules for the usage of a CompuMethod with categoryTEXTTABLE are the same as in the AUTOSAR Classic Platform and are described in the Software Component Template [1].

To summarize, an enumeration value in the CompuMethod with category TEXTTABLE can be provided as a text value in the vt of the CompuConst, in the shortLabel or symbol of the applicable CompuScale of the CompuMethod.

Each CompuScale shall be defined as compuInternalToPhys computation in the CompuMethod and shall contain an upperLimit and lowerLimit.

The following example illustrates how an enumeration is specified using a CompuMethod.

**Listing 3.2: example for enumeration**

**<COMPU-METHOD>**

**<SHORT-NAME>**cylinders**</SHORT-NAME>**

**<CATEGORY>**TEXTTABLE**</CATEGORY>**

**<COMPU-INTERNAL-TO-PHYS>**

**<COMPU-SCALES>**

**<COMPU-SCALE>**

**<LOWER-LIMIT INTERVAL-TYPE=**"CLOSED"**>**0**</LOWER-LIMIT>**

**<UPPER-LIMIT INTERVAL-TYPE=**"CLOSED"**>**0**</UPPER-LIMIT>**

**<COMPU-CONST>**

**<VT>**Cylinder1**</VT>**

**</COMPU-CONST>**

**</COMPU-SCALE>**

**<COMPU-SCALE>**

**<LOWER-LIMIT INTERVAL-TYPE=**"CLOSED"**>**1**</LOWER-LIMIT> <UPPER-LIMIT INTERVAL-TYPE=**"CLOSED"**>**1**</UPPER-LIMIT>**

**<COMPU-CONST>**

**<VT>**Cylinder2**</VT>**

**</COMPU-CONST>**

**</COMPU-SCALE>**

**<COMPU-SCALE>**

**<LOWER-LIMIT INTERVAL-TYPE=**"CLOSED"**>**2**</LOWER-LIMIT> <UPPER-LIMIT INTERVAL-TYPE=**"CLOSED"**>**2**</UPPER-LIMIT>**

**<COMPU-CONST>**

**<VT>**Cylinder3**</VT>**

**</COMPU-CONST>**

**</COMPU-SCALE>**

**<COMPU-SCALE>**

**<LOWER-LIMIT INTERVAL-TYPE=**"CLOSED"**>**3**</LOWER-LIMIT> <UPPER-LIMIT INTERVAL-TYPE=**"CLOSED"**>**3**</UPPER-LIMIT> <COMPU-CONST>**

**<VT>**Cylinder4**</VT>**

**</COMPU-CONST>**

**</COMPU-SCALE>**

**</COMPU-SCALES>**

**</COMPU-INTERNAL-TO-PHYS>**

**</COMPU-METHOD>**

**3.3.3.9 Map Data Type**

**[TPS\_MANI\_03183]**{DRAFT} **CppImplementationDataType of category AS-**

**SOCIATIVE\_MAP** dA CppImplementationDataType of category ASSOCIATIVE\_MAP represents a container that contains key-value pairs with unique keys.c *(RS\_MANI\_00039)*

**[TPS\_MANI\_03184]**{DRAFT} **CppImplementationDataType of category ASSOCIATIVE\_MAP** dFor a C++ binding, a CppImplementationDataType of categoryASSOCIATIVE\_MAP can be implemented as

* an ara::core::Map if StdCppImplementationDataType subclass is used or as
* a map type in a custom namespace (e.g. my::map) if CustomCppImplementationDataType subclass is used (provided that the type in the custom namespace can be configured with the available modeling capabilities).

c*(RS\_MANI\_00039)*

MyVariant: StdCppImplementationDataType

category = ASSOCIATIVE\_MAP

CppTemplateArgument

:

category = ASSOC\_MAP\_KEY

CppTemplateArgument

:

category = ASSOC\_MAP\_VALUE

uint16\_t:

StdCppImplementationDataType

category = VALUE

uint8\_t:

StdCppImplementationDataType

category = VALUE

+

templateType

+

templateType

+

templateArgument

+

templateArgument

**Figure 3.14: Example of the model of an ASSOCIATIVE\_MAP**

**[TPS\_MANI\_03185]**{DRAFT} **Structure of a CppImplementationDataType of categoryASSOCIATIVE\_MAP** dA CppImplementationDataType of category ASSOCIATIVE\_MAP that boils down to a ara::core::Map shall aggregate the following CppTemplateArguments:

* one CppTemplateArgument shall have the categoryASSOC\_MAP\_KEY and shall reference a CppImplementationDataType with the templateType reference. This CppTemplateArgument represents the role that corresponds to ApplicationAssocMapDataType.key and defines the respective data type details.
* one CppTemplateArgument shall shall have the category ASSOC\_MAP\_VALUE and shall reference a CppImplementationDataType with the templateType reference. This CppTemplateArgument represents the role that corresponds to ApplicationAssocMapDataType.value and defines the respective data type details.
* one additional optional CppTemplateArgument is allowed to reference an Allocator with the allocator reference.

c*(RS\_MANI\_00039)*

The example depicted in Figure 3.14 shows the definition of a ASSOCIATIVE\_MAP that has two CppTemplateArguments, one for the key and one for the value.

**3.3.3.10 Variant Data Type**

**[TPS\_MANI\_03189]**{DRAFT} **Definition of CppImplementationDataType of category VARIANT** dA CppImplementationDataType of category VARIANT

represents a type safe union.c*(RS\_MANI\_00039)*

**[TPS\_MANI\_03190]**{DRAFT} **CppImplementationDataType of category**

**VARIANT** dFor a C++ binding, a CppImplementationDataType of category

VARIANT can be implemented as

* an ara::core::Variant if StdCppImplementationDataType subclass is used or as
* a variant type in a custom namespace (e.g. my::variant) if CustomCppImplementationDataType subclass is used (provided that the type in the custom namespace can be configured with the available modeling capabilities).

c*(RS\_MANI\_00039)*

**[TPS\_MANI\_03191]**{DRAFT} **Definition of type alternatives stored in a VARIANT** dA type alternative that is stored in a CppImplementationDataType of category VARIANT is defined by the aggregated templateArgument and the corresponding templateType that defines the data type of the CppTemplateArgument.c*(RS\_MANI\_00039)*

**[constr\_3429]**{DRAFT} **No allocator usage for CppImplementationDataTypes of categoryVARIANT**dCppImplementationDataType of categoryVARIANT is not allowed to aggregate a templateArgument that points to an Allocator in the role allocator.c*()*

MyVariant: StdCppImplementationDataType

category = VARIANT

CppTemplateArgument

:

CppTemplateArgument

:

float:

StdCppImplementationDataType

category = VALUE

uint8\_t:

StdCppImplementationDataType

category = VALUE

+

templateType

+

templateType

+

templateArgument

+

templateArgument

**Figure 3.15: Example of the model of an VARIANT**

The example depicted in Figure 3.15 shows the definition of a VARIANT that has two CppTemplateArguments. Each one represents one alternative type. Please note that the CppTemplateArguments of a CppImplementationDataType are ordered in ARXML and this order is not visible in the object diagram.

**3.3.3.11 Bitfield Data Type**

**[TPS\_MANI\_03202]**{DRAFT} **Definition of bitfield types** dIn the AUTOSAR metamodel, a bitfield is not implemented by means of a CppImplementationDataType with an own category.

A bitfield is defined in the context of a primitive StdCppImplementationDataType of category TYPE\_REFERENCE that boils down to a StdCppImplementationDataType of categoryVALUE.

A CompuMethod of category BITFIELD\_TEXTTABLE is used to assign a special meaning to each bit of the primitive StdCppImplementationDataType.c*(RS\_MANI\_00039)*

CompuScales with a mask inside of the CompuMethod of category BITFIELD\_TEXTTABLE are defining isolated parts that can be independent from each other with respect to the semantics of the data that match the mask.

The rules for the usage of a CompuMethod with categoryBITFIELD\_TEXTTABLE are the same as in the AUTOSAR Classic Platform and are described in the Software Component Template [1].

**3.3.4 Compatibility of ApplicationDataType and CppImplementationDataType**

The usage of ApplicationDataTypes implies that also a corresponding CppImplementationDataType exists at a certain point in time. The usage of CppImplementationDataTypes in a ServiceInterface is required as the basis for generating the ara::com proxies and skeletons and as basis for the serialization of the payload in the network binding.

**[TPS\_MANI\_03223]**{DRAFT} **Existence of CppImplementationDataType** dThe existence of CppImplementationDataTypes is **not** required until the methodology step of generating the Service header files for a ServiceInterface. Before arriving at this step in the methodology, it is perfectly feasible to use only ApplicationDataTypes for describing the semantics of ServiceInterfaces.c*(RS\_MANI\_00003)*

As a consequence, it is necessary to define compatibility rules that unambiguously clarify the conformance of an ApplicationDataType with a CppImplementationDataType and vice versa.

Several rules depend on the category of the data types:

1. As a general rule, if a CppImplementationDataType of categoryTYPE\_-

REFERENCE is targeted by a type mapping all the rules given below apply to the CppImplementationDataType which is finally valid after resolving all such references.

This is not repeated in all rules. For example, if the document states that a given

ApplicationDataType can be mapped to a CppImplementationDataType of categoryVALUE this shall include the possibility of mapping to a CppImplementationDataType of categoryTYPE\_REFERENCE which refers to another CppImplementationDataType of categoryVALUE.

1. **[constr\_5033]**{DRAFT} **Compatibility of data types with categoryVALUE** dAn ApplicationDataType of categoryVALUE can only be mapped to a CppImplementationDataType which also has categoryVALUE.c*()*

In this case, the C++ data type resulting from the CppImplementationDataType shall be able to express all the numerical values required by the ApplicationDataType.

This condition is fulfilled if the numerical range which can be expressed by the

C++ data type at least covers the range defined by the limits in ApplicationDataType.swDataDefProps.dataConstr (which are either internal limits or physical limits to be converted via the CompuMethod which also has to be provided by the ApplicationDataType).

The condition is also fulfilled if the C++ data type covers the range defined in the CompuMethod for an enumeration.

1. **[constr\_5034]**{DRAFT} **Compatibility of data types with categoryBOOLEAN** dAn ApplicationDataType of categoryBOOLEAN can only be mapped to a CppImplementationDataType of categoryVALUE.c*()*
2. **[constr\_5035]**{DRAFT} **Compatibility of data types with categorySTRING** dAn ApplicationDataType of categorySTRING can only be mapped to a CppImplementationDataType of categorySTRING.c*()*
3. **[constr\_5036]**{DRAFT} **Compatibility of data types with categoryARRAY** dAn ApplicationDataType of categoryARRAY can only be mapped to
   * a CppImplementationDataType of categoryARRAY **or**
   * a CppImplementationDataType of categoryVECTOR. c*()*

In this case, the array size and the type of the array elements of the CppImplementationDataType shall be such that they can be mapped/transferred 1:1 by order to the corresponding application data and vice versa.

1. **[constr\_5037]**{DRAFT} **Compatibility of data types with categoryARRAY with variableSize** dAn ApplicationDataType of categoryARRAY that includes one ApplicationArrayElement with arraySizeSemantics set to variableSize in one of the defined dimensions shall be mapped to
   * a CppImplementationDataType of categoryVECTOR c*()*
2. **[constr\_5038]**{DRAFT} **Compatibility of data types with categoryARRAY with fixedSize** dAn ApplicationDataType of category ARRAY that includes only ApplicationArrayElements with arraySizeSemantics set to fixedSize in all defined dimensions shall be mapped to
   * a CppImplementationDataType of categoryARRAY c*()*
3. **[constr\_5039]**{DRAFT} **Compatibility of data types with categorySTRUC-**

**TURE** dAn ApplicationDataType of category STRUCTURE can only be mapped to a CppImplementationDataType of categorySTRUCTURE.c*()*

This means, that the corresponding pairs of elements shall also have compatible types.

1. **[constr\_5040]**{DRAFT} **Compatibility of ApplicationRecordDataType and CppImplementationDataType that both represent an Optional Element Structure** dAn ApplicationRecordDataType that represents an Optional Element Structure can only be mapped to a CppImplementationDataType of categorySTRUCTURE that represents an Optional Element Structure if corresponding pairs of elements have the same value of the attribute isOptional.c*()*
2. **[constr\_5041]**{DRAFT} **Compatibility of data types with categoryASSO-**

**CIATIVE\_MAP**dAn ApplicationDataType of categoryASSOCIATIVE\_MAP can only be mapped to a CppImplementationDataType of categoryASSOCIATIVE\_MAP.c*()*

1. **[constr\_5042]**{DRAFT} **No data type mapping for CppImplementation-**

**DataType of categoryVARIANT**dAn ApplicationDataType shall never be mapped to a CppImplementationDataType of categoryVARIANT.c*()*

1. **[constr\_5043]**{DRAFT} **Forbidden mappings to CppImplementation-**

**DataType**dAn ApplicationDataType of categoryCOM\_AXIS, RES\_AXIS, CURVE, MAP, CUBOID, CUBE\_4, CUBE\_5 is not supported by the Adaptive Platform and can therefore not be mapped to a CppImplementationDataType.c

### ()

Please note that the categories listed in [constr\_5043] are not supported because there is no use case for the usage in Adaptive Platform.

On the AUTOSAR classic Platform, elements of a composite data type are not required to be considered in a DataTypeMap. This regulation is motivated by the fact that an element of a composite data type on the AUTOSAR classic Platform does not necessarily have a reference to an ImplementationDataType.

On the AUTOSAR adaptive Platform the situation is different. The CppImplementationDataTypeElement always requires a reference to a formalized CppImplementationDataType.

Since the processing of the data type definition becomes much easier if all the relevant data types are mentioned in a DataTypeMap the existence of [constr\_5044] is motivated.

**[constr\_5044]**{DRAFT} **DataTypeMap for composite data types** dIn the context of a given ServiceInterface, all pairs of ApplicationDataType and CppImplementationDataType used in the context of the definition of an ApplicationCompositeDataType used in the context of an event, field, method shall be described in a DataTypeMap that is contained in one of the DataTypeMappingSets that are referenced in a PortInterfaceToDataTypeMapping that also references the mentioned ServiceInterface.c*()*

## 3.4 Service Interface

**3.4.1 Overview**

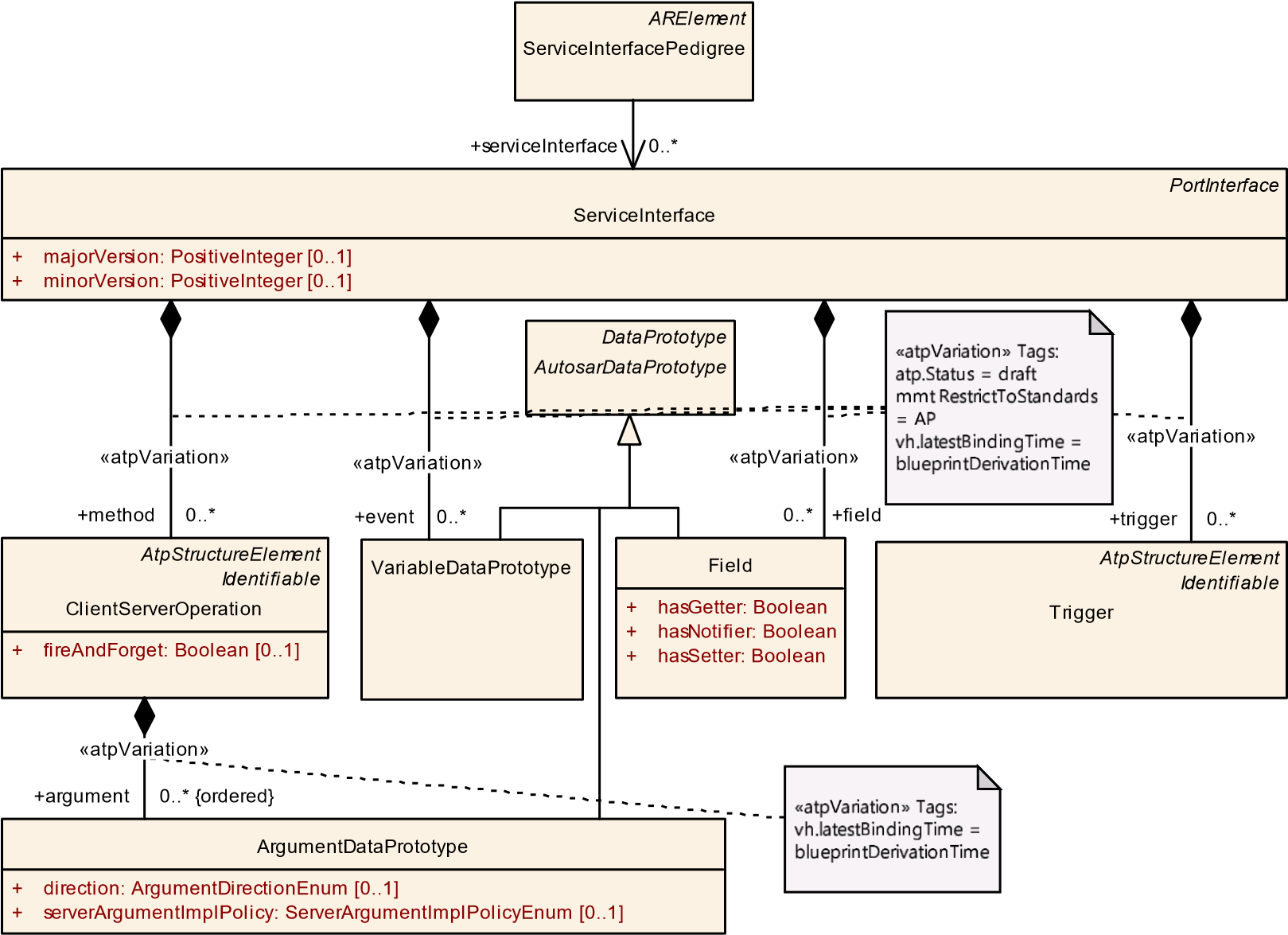
**[TPS\_MANI\_01001]**{DRAFT} **Meaning of ServiceInterface**dMeta-class ServiceInterface inherits from PortInterface and allows for a heterogeneous aggregation of elements, i.e. it is possible to mix

* aggregation of VariableDataPrototype in the role event with
* aggregation of meta-class Field in the role field with
* aggregation of ClientServerOperation in the role method
* aggregation of Trigger in the role trigger **within the same** ServiceInterface.c*(RS\_MANI\_00001, RS\_MANI\_00003)*

The purpose of this modeling is to embrace the concept of service-oriented communication [3] and better support this paradigm for communication on the *AUTOSAR adaptive platform*.

Please note that, in terms of semantics, the ApApplicationError represents a sort of second-class citizen (that only makes sense in the presence of ClientServerOperation in the role method) in the scope of the ServiceInterface.

More information can be found in section 3.4.8.



**Figure 3.16: Modeling of the ServiceInterface**

**[constr\_1483]**{DRAFT} **Applicability of a ServiceInterface**dThe applicability of a ServiceInterface shall be limited to the *AUTOSAR adaptive platform*, i.e. a ServiceInterface shall only be taken to type a PortPrototype if the latter is aggregated by an AdaptiveApplicationSwComponentType or by a CompositionSwComponentType defined in the context of an Executable.c*()*

Please note that on the *AUTOSAR adaptive platform* there are use-cases for the utilization of a ServiceInterface **without** the existence of a corresponding PortPrototype. For more explanation, please refer to [TPS\_MANI\_01032].

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ServiceInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This represents the ability to define a PortInterface that consists of a heterogeneous collection of methods, events and fields.  **Tags:**  atp.Status=draft  atp.recommendedPackage=ServiceInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| event | VariableDataPrototype | \* | aggr | This represents the collection of events defined in the context of a ServiceInterface.  **Stereotypes:** atpVariation **Tags:**  atp.Status=draft  vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30 |
| field | Field | \* | aggr | This represents the collection of fields defined in the context of a ServiceInterface.  **Stereotypes:** atpVariation **Tags:**  atp.Status=draft  vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=40 |
| majorVersion | PositiveInteger | 0..1 | attr | Major version of the service contract.  **Tags:**  atp.Status=draft xml.sequenceOffset=10 |
| method | ClientServerOperation | \* | aggr | This represents the collection of methods defined in the context of a ServiceInterface.  **Stereotypes:** atpVariation **Tags:**  atp.Status=draft  vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=50 |
| minorVersion | PositiveInteger | 0..1 | attr | Minor version of the service contract.  **Tags:**  atp.Status=draft xml.sequenceOffset=20 |
| trigger | Trigger | \* | aggr | This represents the collection of triggers defined in the context of a ServiceInterface.  **Stereotypes:** atpVariation **Tags:**  atp.Status=draft  vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=60 |

**Table 3.23: ServiceInterface**

As already described in [TPS\_SWCT\_01844], AUTOSAR does not support the existence of optional arguments in a ClientServerOperation.

This restriction is motivated by the lack of support for optional arguments in the APIs of the RTE on the *AUTOSAR classic platform*. For the sake of interoperability between the *classic platform* and the *adaptive platform*, this restriction is observed on the *AUTOSAR adaptive platform* as well.

**[TPS\_MANI\_01007]**{DRAFT} **Atomic unit of service discovery** dAs far as the application level is concerned, the atomic unit for **service discovery** on the *AUTOSAR adaptive platform* is the ServiceInterface.c*(RS\_MANI\_00003)*

Please note that there is no obligation to have any method, event, trigger, or field defined in the context of a given ServiceInterface. In other words, the existence of a ServiceInterface by itself represents a valid semantics that has a value on its own.

For example, a use case could exist where a given service instance that corresponds to such a ServiceInterface is offered with the mere intention to signal that the ECU that provides the service instance is becoming ready for something, e.g. being diagnosed.

A tester could then take the existence of the offer as an indication to initiate a connection to the respective ECU.

**3.4.2 Event**

**[TPS\_MANI\_01033]**{DRAFT} **Semantics of ServiceInterface.event** dAn event represents an update to a piece of data. The server decides when to send this update and makes sure that the event has full control over the value.

The occurrence of an event is transmitted from a server to one or more client(s).c *(RS\_MANI\_00003)*

**[constr\_1494]**{DRAFT} **Initial value for event** dAn ServiceInterface.event shall **not** have an initValue.c*()*

For the client, the only way to get access to the value of an event is to receive an update of the event from the server.

As mentioned in [constr\_1494], the Server always has full control over the value of the event and when it is sent to clients. Therefore, the definition of an initValue is not necessary.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **VariableDataPrototype** | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes | | | |
| ***Note*** | A VariableDataPrototype is used to contain values in an ECU application. This means that most likely a VariableDataPrototype allocates "static" memory on the ECU. In some cases optimization strategies might lead to a situation where the memory allocation can be avoided.  In particular, the value of a VariableDataPrototype is likely to change as the ECU on which it is used executes. | | | |
| ***Base*** | *ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *Multilanguage*  *Referrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| initValue | ValueSpecification | 0..1 | aggr | Specifies initial value(s) of the VariableDataPrototype |

**Table 3.24: VariableDataPrototype**

**3.4.3 Trigger**

**[TPS\_MANI\_03291]**{DRAFT} **Semantics of ServiceInterface.trigger** dA trigger represents a special kind of an event without any data that is transmitted from a server to one or more client(s) and at which occurrence the Service Consumer shall react in a particular manner.c*(RS\_MANI\_00003)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Trigger** | | | |
| ***Package*** | M2::AUTOSARTemplates::CommonStructure::TriggerDeclaration | | | |
| ***Note*** | The Trigger represents a special kind of an event (without data) at which occurrence the Service Consumer shall react in a particular manner. | | | |
| ***Base*** | *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.25: Trigger**

Please note that the trigger is processed in the queued manner, i.e. the triggers are stored in a queue and are processed in “first in first out” order.

**3.4.4 Field**

**[TPS\_MANI\_01034]**{DRAFT} **Semantics of ServiceInterface.field** dA field represents a piece of data hosted by a server that exposes to one or more client(s) a get accessor and/or a set mutator.

Clients can optionally receive notifications of changes of the field’s value.c*(RS\_MANI\_00003)*

In comparison to an event, a field has a concrete value at any time. This conceptual difference can be explained along the following examples:

Let a traffic-sign detection be an example for the semantics of an event. The detection of a traffic-sign represents a discrete event in time that would be raised by the service component any time a speed limit sign is detected.

On the other hand, let a temperature preset of the in-vehicle air-condition be an example for a field that has a concrete value at any given time. The concrete value can be set by a client, can be obtained on request of a client, and – at the same time – a change of the temperature preset represents relevant information by itself.

In summary, this means that if a field is defined with hasNotifier and a client subscribes to it then the current value of the field is sent back immediately to the subscriber in an event-like notification pattern as soon as the subscription to the field becomes effective.

Additional update notifications will be sent to subscribers whenever the value of the field gets updated.

In more technical terms, the get() accessor method the current field value can be retrieved by the client. By means of calling the set() mutator method the field value can be updated by the client.

Please note that all features that a field provides are optional, given a fulfillment of [constr\_1673]. In the ServiceInterface.field description it is defined whether the field supports the on-change-notification (hasNotifier), the get() accessor (hasGetter) or the set() mutator (hasSetter).

Admittedly, the concept of the field is roughly equivalent to an aggregation of an event with correlated get()/set()methods.

As far as the meta-model is concerned, the fact that a field shall have a concrete value at any time demands the **definition of an initial value** for the field. This aspect is clarified by [TPS\_MANI\_03212].

The existence of meta-class field as a first class citizen in the ServiceInterface expresses in addition to the existence of an individual event and individual methods that the two defined accessor/mutator methods get() and set() are applied to the **same data object** and that the defined field notifier reports each value change of this data object to subscribers.

In other words, the semantics of meta-class Field is fully determined by the attributes hasGetter, hasSetter, and hasNotifier.

Therefore, a Field where all of these attributes are set to False wouldn’t have any useful meaning and shall therefore not exist.

**[constr\_1673]**{DRAFT} **Existence of attributes hasGetter, hasSetter, and hasNotifier**dFor any given Field, all of the attributes

* hasGetter
* hasSetter
* hasNotifier

shall exist and at least one of the attributes shall be set to True.c*()*

Please note that [constr\_1673] allows that a Field may be defined with a notifier but without the two defined methods get() and set(). As described above a subscriber to a field notifier will get the current value of the Field immediately after the subscription. This functionality makes a Field without get()/set() methods useful in some functional cases compared to the usage of an event where the value would only be sent after the event is triggered.

|  |  |
| --- | --- |
| ***Class*** | **Field** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Field** | | | |
| ***Note*** | This meta-class represents the ability to define a piece of data that can be accessed with read and/or write semantics. It is also possible to generate a notification if the value of the data changes.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *Multilanguage*  *Referrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| hasGetter | Boolean | 1 | attr | This attribute controls whether read access is foreseen to this field.  **Tags:**atp.Status=draft |
| hasNotifier | Boolean | 1 | attr | This attribute controls whether a notification semantics is foreseen to this field.  **Tags:**atp.Status=draft |
| hasSetter | Boolean | 1 | attr | This attribute controls whether write access is foreseen to this field.  **Tags:**atp.Status=draft |

**Table 3.26: Field**

**3.4.5 Method**

**[TPS\_MANI\_01035]**{DRAFT} **Semantics of ServiceInterface.method** dA method represents a function that is executed by and in the scope of a server on request of one or more client(s).c*(RS\_MANI\_00003)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ClientServerOperation** | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::PortInterface | | | |
| ***Note*** | An operation declared within the scope of a client/server interface. | | | |
| ***Base*** | *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| argument (ordered) | ArgumentDataPrototype | \* | aggr | An argument of this ClientServerOperation  **Stereotypes:** atpVariation  **Tags:**vh.latestBindingTime=blueprintDerivationTime |
| fireAndForget | Boolean | 0..1 | attr | This attribute defines whether this method is a fire&forget method (true) or not (false).  **Tags:**atp.Status=draft |
| possibleApError | ApApplicationError | \* | ref | This reference identifies AdaptivePlatformApplication Errors as a possible error raised by the enclosing Client ServerOperation.  **Tags:**atp.Status=draft |
| possibleApError Set | ApApplicationErrorSet | \* | ref | This reference represents the ability to refer to an entire group of ApApplicationErrors as one model element instead of having to refer to all the represented Ap ApplicationErrors separately.  **Tags:**atp.Status=draft |

**Table 3.27: ClientServerOperation**

**3.4.5.1 Fire and Forget Method**

A so-called “fire & forget” method represents a special form of a method dedicated to the sole purpose of conveying information from a client to a server.

There is no expectation that the implementation of the method executes any kind of algorithm other than to merely accept the incoming data.

Spun from this angle, the semantics of a “fire & forget” method is comparable to the semantics of an event, only reverse.

In other words, the “fire & forget” method conveys the data and the occurrence of the data **from a client to a server**. For comparison, the event is used to convey information in combination with the occurrence of the information from **a server to a client**.

The *occurrence* aspect of this statement has the consequence that e.g. the number of “fire & forget” calls can be counted by the implementation of the server and this meta-information could be taken to convey additional semantics on top of the actual data.

**[TPS\_MANI\_01064]**{DRAFT} **Semantics of attribute method.fireAndForget** dThe activation of the “fire & forget” semantics of a given method is achieved by setting the value of attribute method.fireAndForget to value true.c*(RS\_MANI\_00003)*

**[TPS\_MANI\_03118]**{DRAFT} **Semantics of ServiceInterface.method with fireAndForget set to true** dA method with fireAndForget set to the value true represents a void-return-method where the client is not expecting any kind of acknowledge or handshake from the server side.c*(RS\_MANI\_00003)*

**[constr\_3374]**{DRAFT} **method with attribute fireAndForget set to true shall not have any inout or out arguments** dA method that has the value of attribute fireAndForget set to true is not allowed to have any arguments with direction inout or out.c*()*

**[constr\_3375]**{DRAFT} **method with attribute fireAndForget set to true shall not reference an ApApplicationError**dA method that has the value of attribute fireAndForget set to true is not allowed to reference

* an ApApplicationError in role possibleApError and/or
* an ApApplicationErrorSet in the role possibleApErrorSet. c*()*

**[TPS\_MANI\_03119]**{DRAFT} **Default value for the attribute fireAndForget of meta-class ClientServerOperation** dIf the attribute fireAndForget is not defined then it shall be assumed that no “fire & forget” semantics is intended.c*(RS\_MANI\_00003)*

**3.4.6 Versioning of ServiceInterfaces**

Using multiple versions of the same ServiceInterface supports an independent life cycle of services and allows to change and enhance ServiceInterfaces without affection of existing consumers. This chapter describes how different versions of the same ServiceInterface can be modeled.

A version of a ServiceInterface may be defined for example as ServiceInterface with an own shortName (e.g. Service\_Version1, Service\_Version2) or as ServiceInterface that is located in an own ARPackage (e.g. /Version1/Service, /Version2/Service).

It is also allowed to assign a different *namespace* to the different ServiceInterface versions to influence the generated code, e.g. to generate com::version1::Service and com::version2::Service.

It is expected that if using different versions of the same ServiceInterface in one Executable then different *namespace*s shall be used for each ServiceInterface version.

The attributes ServiceInterface.majorVersion and ServiceInterface.minorVersion provide the possibility to define version information at the level of the ServiceInterface.

**[TPS\_MANI\_03616]**{DRAFT} **Semantic versioning of ServiceInterface.majorVersion and ServiceInterface.minorVersion**dService contract versioning rules:

* for backwards-incompatible interface or behavior changes the majorVersion number shall be increased and the minorVersion number shall be set to 0
* for backwards-compatible interface or behavior changes the majorVersion number shall be unchanged and the minorVersion number shall be increased. c*(RS\_MANI\_00064)*

Note that it is expected that the decision about backwards compatibility is made by the service designer. In other words AUTOSAR does not define formal criteria for the backwards compatibility of ServiceInterfaces.

As for the modeling of several versions of a ServiceInterface, the fully qualified shortNames of the ServiceInterfaces have to be different. The ServiceInterfacePedigree allows to collect the set of ServiceInterfaces which form the collection of different versions of the same *Service*.

|  |  |
| --- | --- |
| ***Class*** | **ServiceInterfacePedigree** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ServiceInterfacePedigree** | | | |
| ***Note*** | Collection of ServiceInterfaces that belong to the same versioning.  **Tags:**  atp.Status=draft  atp.recommendedPackage=ServiceInterfacePedigrees | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| serviceInterface | ServiceInterface | \* | ref | Reference to the ServiceInterfaces which belong to the same versioning.  **Tags:**atp.Status=draft |

**Table 3.28: ServiceInterfacePedigree**

The other consumers of this service do not need to switch to using the latest version of this ServiceInterface, but can continue to use older versions of the ServiceInterface they were designed for and tested with.

**3.4.6.1 Versioning driven by transport layer**

Each transport layer mechanism (e.g. SOME/IP) may define its own compatibility rules. Therefore, for each individual transport layer an own impact assessment on the compatibility needs to be performed whether the changed service interface has an incompatible representation on this transport layer.

The compatibility depends on the features that are used on the transport layer. For example, in SOME/IP a length field that is put in front of a struct allows that during deserialization unknown elements at the end of an extensible data struct are skipped.

An additional option in SOME/IP is the usage of Data IDs in front of optional struct members. With this approach the receiver can skip unknown members of the struct, i.e. where the Data ID is unknown.

Therefore, on the Application Design level, all changes of ServiceInterfaces shall be handled carefully since only the used transport layer and the used features on the transport layer decide whether the change is compatible or not.

If one wants to make sure that two AutosarDataPrototypes inside a ServiceInterface are compatible then both AutosarDataPrototypes shall be typed by an identical AutosarDataType.

During the ServiceInterfaceDeployment the ServiceInterface is mapped to a middleware transport layer where the necessary middleware transport layer specific configuration settings are performed, as described in chapter 10.2.

For example, it is possible to assign the same SOME/IP serviceInterfaceId to different versions of the same ServiceInterface, but a different majorVersion or minorVersion.

This approach takes into account that the compatibility of ServiceInterfaces is heavily influenced by the used transport binding.

Version1

ServiceInterface

MyInterface

Event1

Event2

Version2

ServiceInterface

MyInterface

Event1

Event2

Method

Version3

ServiceInterface

MyInterface

Event1

Method

SomeipSIDeployment

ServiceInterfaceId =

46

-

1

majorVersion =

-

0

minorVersion =

-

SomeipSIDeployment

ServiceInterfaceId =

46

-

1

majorVersion =

-

1

minorVersion =

-

SomeipSIDeployment

-

ServiceInterfaceId =

46

-

majorVersion =

2

-

minorVersion =

0

**Figure 3.17: Example for different versions of the same ServiceInterface**

Please note that the compatibility rules for SOME/IP are described in [7].

**3.4.7 Namespace**

The definition of a ServiceInterface has a direct impact on the code of an application on the *AUTOSAR adaptive platform*.

Without going into too much detail at this point, it is necessary to support the definition of a *namespace* in the context of a ServiceInterface.

The namespace shall be used to encapsulate source code related to the ServiceInterface and thus avoid name clashes with the content of other definitions of ServiceInterfaces.

In principle, the definition of the namespace around a concrete ServiceInterface could be derived from the structure of ARPackages in which the definition of the ServiceInterface is contained. However, this approach puts some constraints of the package structure.

The same ServiceInterface may be used in different projects that may or may not demand the usage of a specific *different* package structure.

This placement of the same ServiceInterface in potentially different package hierarchies would lead to the definition of different namespaces, and thus the necessity to create or generate the code representing the ServiceInterface **plus** the code that uses this definition again and again.

One way to overcome this potential issue is to attach a dedicated namespace definition to the definition of the ServiceInterface itself.

This approach is documented in Figure 3.18.

SymbolProps

*Referrable*

*ImplementationProps*

+

symbol: CIdentifier

[0..1]

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

«atpSplitable»

+

namespace

0..\* {

ordered

}

**Figure 3.18: Specification of namespaces in PortInterfaces**

**[TPS\_MANI\_01004]**{DRAFT} **Semantics of ServiceInterface.namespace**dThe aggregation ServiceInterface.namespace shall be used to define the namespace to be used for the source code that corresponds to the given ServiceInterface.c *(RS\_MANI\_00003)*

**[TPS\_MANI\_01005]**{DRAFT} **The definition of the namespace of a ServiceInterface may follow a hierarchical pattern** dThe namespace of a ServiceInterface may follow a hierarchical pattern, as supported by many modern programming languages.

The separator between the elements of the hierarchical namespace definition depends on the used programming language and is not explicitly defined in the model.

The model only defines the elements of the hierarchical namespace pattern.c*(RS\_-*

*MANI\_00003)*

**Listing 3.3: Example for the definition of a namespace for a given ServiceInterface**

**<SERVICE-INTERFACE>**

**<SHORT-NAME>**MyServiceInterface**</SHORT-NAME>**

**<NAMESPACES>**

**<SYMBOL-PROPS>**

**<SHORT-NAME>**first**</SHORT-NAME>**

**<SYMBOL>**com**</SYMBOL>**

**</SYMBOL-PROPS>**

**<SYMBOL-PROPS>**

**<SHORT-NAME>**second**</SHORT-NAME>**

**<SYMBOL>**myCompany**</SYMBOL>**

**</SYMBOL-PROPS>**

**<SYMBOL-PROPS>**

**<SHORT-NAME>**third**</SHORT-NAME>**

**<SYMBOL>**software**</SYMBOL>**

**</SYMBOL-PROPS>**

**</NAMESPACES>**

**</SERVICE-INTERFACE>**

As the consequence of the ability to define a hierarchical namespace, the aggregation ServiceInterface.namespace is qualified as being ordered.

This means that the order of individual elements to the collection of namespaces has a semantical relevance[[5]](#footnote-5).

**[TPS\_MANI\_01006]**{DRAFT} **Ordered definition of ServiceInterface.namespace** dIn a hierarchical definition of ServiceInterface.namespace the order of namespace fragments shall be maintained in the translation of the namespace to source code.

In other words, the first namespace fragment shall appear first, followed by the second namespace fragment, and so on.c*(RS\_MANI\_00003)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***PortInterface*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::PortInterface | | | |
| ***Note*** | Abstract base class for an interface that is either provided or required by a port of a software component. | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Subclasses*** | *AbstractRawDataStreamInterface*, *AbstractSynchronizedTimeBaseInterface*, ClientServerInterface,  *CryptoInterface*, *DataInterface*, *DiagnosticPortInterface*, LogAndTraceInterface, ModeSwitchInterface,  *PersistencyInterface*, *PlatformHealthManagementInterface*, SecurityEventReportInterface, Service Interface, TriggerInterface | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| namespace  (ordered) | SymbolProps | \* | aggr | This represents the SymbolProps used for the definition of a hierarchical namespace applicable for the generation of code artifacts out of the definition of a ServiceInterface.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=namespace.shortName atp.Status=draft |

**Table 3.29: PortInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SymbolProps** | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Components | | | |
| ***Note*** | This meta-class represents the ability to contribute a part of a namespace. | | | |
| ***Base*** | *ARObject*, *ImplementationProps*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.30: SymbolProps**

The Listing 3.3 exemplifies the statement made by [TPS\_MANI\_01006], i.e. the resulting name space in e.g. C++ would look like sketched in Listing 3.4.

1. namespace com {
2. namespace myCompany {
3. namespace software {

4

1. }
2. }
3. }

**Listing 3.4: Resulting namespace for the example ServiceInterface**

**3.4.8 Error Handling**

The modeling of error handling on the *AUTOSAR adaptive platform* slightly differs from the approach implemented on the *AUTOSAR classic platform*.

In particular, the formal representation of an error during the execution of a method is done in a global scope, i.e. such a definition can be reused arbitrarily by any ServiceInterface.

**[TPS\_MANI\_01190]**{DRAFT} **Semantics of ApApplicationError** dMeta-class ApApplicationError represents the ability to define the existence of an error during the execution of a method independently of the scope of a ServiceInterface or ClientServerOperation.c*(RS\_MANI\_00002)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ApApplicationError** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class represents the ability to formally specify the semantics of an application error on the AUTOSAR adaptive platform  **Tags:**  atp.Status=draft  atp.recommendedPackage=ApplicationErrors | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| errorCode | Integer | 1 | attr | This attribute has the ability to specify the error code value within the enclosing AdaptivePlatformApplication Error.  **Tags:**atp.Status=draft |
| errorDomain | ApApplicationError  Domain | 1 | ref | This reference represents the error domain of the Ap ApplicationError.  **Tags:**atp.Status=draft |

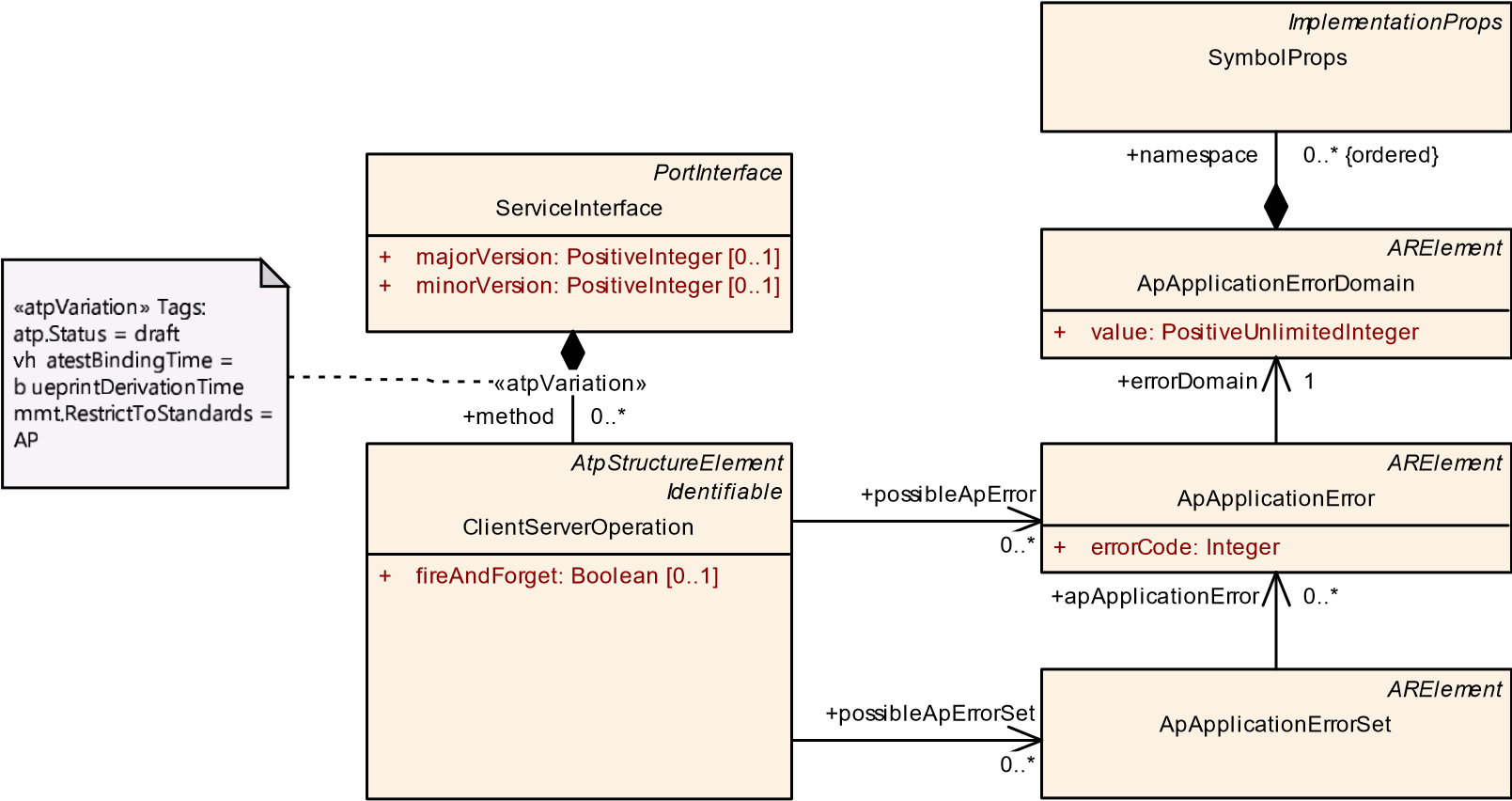
**Table 3.31: ApApplicationError**

**[TPS\_MANI\_01198]**{DRAFT} **Semantics of ApApplicationErrorSet** dMetaclass ApApplicationErrorSet has the ability to group references to ApApplicationError and thus represents a “proxy” to this group of references towards the ClientServerOperation.

The use case for this modeling ability is that some ClientServerOperations may have to reference an identical significant number of ApApplicationErrors.

Letting each of the ClientServerOperations repeat the same set of references to ApApplicationError is considered unnecessary and therefore the ability to refer to a group instead of individual references is provided as an alternative.c*(RS\_MANI\_00026)*

The decision whether an ApApplicationErrorSet is defined and referenced from specific ClientServerOperations has to be done on an individual basis. AUTOSAR just wants to make this business as straightforward as possible.



**Figure 3.19: Modeling of ApApplicationError on the *AUTOSAR adaptive platform***

Please note that it is also positively possible to mix the usage of ClientServer-

Operation.possibleApError and ClientServerOperation.possibleApErrorSet.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ApApplicationErrorSet** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class acts as a reference target that represents an entire collection of APApplicationErrors. This takes the burden from ClientServerOperations that reference a larger number of ApApplication Errors.  **Tags:**  atp.Status=draft  atp.recommendedPackage=ApplicationErrorSets | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| apApplication Error | ApApplicationError | \* | ref | This reference represents the collection of ApApplication  Error represented by the enclosing ApApplicationErrorSet  **Tags:**atp.Status=draft |

**Table 3.32: ApApplicationErrorSet**

As ApApplicationError is no longer defined within the scope of a ServiceInterface, there is no need to define a mapping between two ApApplicationErrors by means of a dedicated sub-class of ServiceInterfaceElementMapping.

**[TPS\_MANI\_01191]**{DRAFT} **Modeling of possible errors** dA ClientServerOperation aggregated by a ServiceInterface in the role method shall reference

* one or more ApApplicationError(s) in the role possibleApError
* one or more ApApplicationErrorSet(s) in the role possibleApErrorSet

to formally specify the existence of possible errors raised by the ClientServerOperation.c*(RS\_MANI\_00026)*

**[TPS\_MANI\_01192]**{DRAFT} **Semantics of ApApplicationErrorDomain**dMetaclass ApApplicationErrorDomain shall be used to define a specific error domain that can potentially be standardized by AUTOSAR.

Therefore, the definition of such an error domain is not defined in the scope of the

ApApplicationError itself. Instead, an ApApplicationError identifies the applicable error domain by means of a reference in the role errorDomain.

It is possible to attach the definition of a namespace to ApApplicationErrorDomain because this information is relevant for the language binding.c*(RS\_MANI\_00026)*

**[constr\_1627]**{DRAFT} **Supported value range for attribute ApApplication-**

**ErrorDomain.value** dThe supported value range of attribute ApApplication-

ErrorDomain.value is limited to the interval [0..18446744073709551616].c*()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ApApplicationErrorDomain** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class represents the ability to define a global error domain for an ApApplicationError.  **Tags:**  atp.Status=draft  atp.recommendedPackage=ApplicationErrorDomains | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| namespace  (ordered) | SymbolProps | \* | aggr | This aggregation defines the namespace of the Ap ApplicationErrorDomain **Tags:**atp.Status=draft |
| value | PositiveUnlimitedInteger | 1 | attr | This attribute identifies the error category.  **Tags:**atp.Status=draft |

**Table 3.33: ApApplicationErrorDomain**

**[constr\_1625]**{DRAFT} **Existence of reference ApApplicationError.errorDomain**dFor each ApApplicationError, the reference errorDomain shall exist.

In other words, the association of an ApApplicationError with a corresponding ApApplicationErrorDomain is mandatory.c*()*

**[constr\_1664]**{DRAFT} **Unique ApApplicationError.shortName**dWithin the set of all ApApplicationErrors that reference a given ApApplicationErrorDomain in the role errorDomain the attribute ApApplicationError.shortName shall have a unique value.c*()*

**[constr\_1665]**{DRAFT} **Unique ApApplicationError.errorCode**dWithin the set of all ApApplicationErrors that reference a given ApApplicationErrorDomain in the role errorDomain the attribute ApApplicationError.errorCode shall have a unique value.c*()*

Rationale for the existence of [constr\_1664] and [constr\_1665]: the language binding for C++ foresees the usage of attributes ApApplicationError.shortName and ApApplicationError.errorCode for the creation of an enum within the context of the ApApplicationErrorDomain.

Duplicates in terms of labels of enumerators or values of enumerators lead to compiletime errors.

**3.4.9 Service Interface Data Type Mapping**

An important step in the workflow of implementing software on the *AUTOSAR adaptive platform* is the creation of a code-based representation of a ServiceInterface to make it accessible for the application code.

This creation of a code-based representation is usually automatized and will be executed by a code generator. This code generator needs an input from the model. The main input for this purpose is obviously the definition of the ServiceInterface itself.

However, this is not sufficient. The designer of a ServiceInterface is free to use ApplicationDataTypes for the specification of the details of the ServiceInterface.

It is therefore necessary to provide the definition of an AbstractImplementationDataType for each of the used ApplicationDataType. In the meta-model, this correspondence is implemented by means of the meta-class DataTypeMappingSet[[6]](#footnote-6).

However, from the methodological point of view it is considered inappropriate to let ServiceInterface directly refer to one or more DataTypeMappingSet(s).

For clarification, this would mean that the mapping of ApplicationDataType to AbstractImplementationDataType becomes an integral part of the definition of the ServiceInterface although the mapping itself does not really contribute to the actual semantics of the ServiceInterface.

As a consequence, the ServiceInterface would have to be updated whenever the mapping between data types changes.

But since the definition of ServiceInterfaces are usually considered very stable a frequent update for the mere purpose of acknowledging a change in the data type mapping is not acceptable.

In this concrete case, the described problem can be circumvented by the definition of a mapping class that refers to both a ServiceInterface and a DataTypeMappingSet and therefore create the correspondence without the need to update the ServiceInterface.

Although the prelude into this chapter suggests the existence of a meta-class that maps a ServiceInterface to one or more DataTypeMappingSet(s) the actual meta-model is designed with a broader focus.

In the future, there could be further kinds of PortInterfaces beside the ServiceInterface that need to fulfill the same use case.

Consequently, the name of the meta-class created for this purpose is PortInterfaceToDataTypeMapping.

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

DataTypeMappingSet

*ARElement*

PortInterfaceToDataTypeMapping

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

+

dataTypeMappingSet

1..\*

+

portInterface

1

**Figure 3.20: Modeling of PortInterfaceToDataTypeMapping**

**[constr\_1507]**{DRAFT} **PortInterfaceToDataTypeMapping is only applicable to ServiceInterface or PersistencyKeyValueStorageInterface** dPortInterfaceToDataTypeMapping.portInterface shall only refer to **either** a ServiceInterface **or** a PersistencyKeyValueStorageInterface.c*()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PortInterfaceToDataTypeMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class represents the ability to associate a PortInterface with a DataTypeMappingSet. This association is needed for the generation of header files in the scope of a single PortInterface.  The association is intentionally made outside the scope of the PortInterface itself because the designers of a PortInterface most likely will not want to add details about the level of ImplementationDataType.  **Tags:**  atp.Status=draft  atp.recommendedPackage=PortInterfaceToDataTypeMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| dataType  MappingSet | DataTypeMappingSet | 1..\* | ref | This represents the reference to the applicable data TypemappingSet  **Tags:**  atp.Status=draft  atp.StatusComment=Reserved for adaptive platform |
| portInterface | PortInterface | 1 | ref | This represents the reference to the applicable Port Interface  **Tags:**  atp.Status=draft  atp.StatusComment=Reserved for adaptive platform |

**Table 3.34: PortInterfaceToDataTypeMapping**

|  |  |
| --- | --- |
| ***Class*** | **DataTypeMappingSet** |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DataTypeMappingSet** | | | |
| ***Note*** | This class represents a list of mappings between ApplicationDataTypes and ImplementationDataTypes.  In addition, it can contain mappings between ImplementationDataTypes and ModeDeclarationGroups.  **Tags:**atp.recommendedPackage=DataTypeMappingSets | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| dataTypeMap | DataTypeMap | \* | aggr | This is one particular association between an Application DataType and its AbstractImplementationDataType. |
| modeRequest TypeMap | ModeRequestTypeMap | \* | aggr | This is one particular association between an Mode DeclarationGroup and its AbstractImplementationData Type. |

**Table 3.35: DataTypeMappingSet**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DataTypeMap** | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes | | | |
| ***Note*** | This class represents the relationship between ApplicationDataType and its implementing Abstract ImplementationDataType. | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| applicationData Type | ApplicationDataType | 0..1 | ref | This is the corresponding ApplicationDataType |
| implementation DataType | AbstractImplementation  DataType | 0..1 | ref | This is the corresponding AbstractImplementationData Type. |

**Table 3.36: DataTypeMap**

**3.4.10 Communication Group pattern**

The Communication Group defines a specific pattern of the usage of a ServiceInterface in a bi-directional way.

The details can be found in SWS\_CommunicationManagement [8]. In order to define a Communication Group several ServiceInterface.category values are defined.

**[TPS\_MANI\_03628]**{DRAFT} **Standardized values of ServiceInterface.category** dThe AUTOSAR Standard reserves the following values for attribute ServiceInterface.category:

* COMMUNICATION\_GROUP
* COMMUNICATION\_GROUP\_SERVER
* COMMUNICATION\_GROUP\_CLIENT

It is possible to use a custom, non-standardized value for the attribute ServiceInterface.category but this option comes with the obligation to use a value that is guaranteed to not clash with possible future extensions of the collection of standardized values, e.g. use company name in the category value.c*()*

The general idea of the Communication Group pattern is that a ServiceInterface of categoryCOMMUNICATION\_GROUP is created to describe the information to be transported (the msg and responseMsg data types). There will not be any instance of this ServiceInterface of categoryCOMMUNICATION\_GROUP in the system, it is just a design artifact.

Out of the ServiceInterface of categoryCOMMUNICATION\_GROUP the two ServiceInterfaces for the server (categoryCOMMUNICATION\_GROUP\_SERVER) and the client (categoryCOMMUNICATION\_GROUP\_CLIENT) roles are created. The rules how this creation shall be done are defined in SWS\_CommunicationManagement [8].

**ServiceInterface**

category

:

**COMMUNICATION**

**\_**

**GROUP**

*-*

*method message*

*(*

*msg*

*)*

*-*

*event response*

*(*

*responseMsg*

*)*

**ServiceInterface**

category

:

**COMMUNICATION**

**\_**

**GROUP**

**\_**

**SERVER**

-

method broadcast

(

msg

)

-

method message

(

clientId

,

msg

)

-

event response

(

clientID

,

responseMsg

)

-

method

listClients

()

**ServiceInterface**

category

:

**COMMUNICATION**

**\_**

**GROUP**

**\_**

**CLIENT**

-

method message

(

msg

)

-

event response

(

responseMsg

)

<<

derive

>>

<<

derive

>>

**Figure 3.21: Example of Communication Group categories**

## 3.5 Service Interface Mapping

Please note that, according to [TPS\_MANI\_01007], the ServiceInterface becomes the single basis for both VFB-based and *external* (i.e. using communication networks) communication.

This concept is in stark contrast to the approach on the *AUTOSAR classic platform* where different model elements are used for the VFB-level (PortInterface) and the network-level (SystemSignal, ISignal, and ISignalIPdu).

The usage of different model elements optimally supports the existence of different granularity for VFB-based vs. network-based communication.

In other words, design of communication on the network level may be subject to different design restrictions, e.g. keep the bus load caused by service discovery manageable by defining coarse-grained communication packages.

Opposed to that, designers on the VFB level may want to define interface granularity to achieve maximum reusability.

ServiceInterfaceMapping

*PortInterface*

ServiceInterface

[0..1]

majorVersion: PositiveInteger

+

minorVersion: PositiveInteger

+

[0..1]

*AtpBlueprint*

*AtpBlueprintable*

*Identifiable*

*PortInterfaceMapping*

*AtpStructureElement*

*SwConnector*

PassThroughSwConnector

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

PortInterfaceMappingSet

+

compositeServiceInterface

1

+

sourceServiceInterface

1..\*

«atpVariation»

+

portInterfaceMapping

0..\*

+

mapping

0..1

**Figure 3.22: Modeling of the ServiceInterfaceMapping**

**[TPS\_MANI\_01002]**{DRAFT} **Semantics of meta-class ServiceInterfaceMapping**dIn order to sort out a potentially different motivation between the definition of

* ServiceInterfaces explicitly designed for VFB-based communication and
* ServiceInterfaces explicitly designed for network-based communication meta-class ServiceInterfaceMapping is available to map
* (fine-grained) ServiceInterfaces for the VFB-communication to
* (coarse-grained) ServiceInterfaces for network communication.

c*(RS\_MANI\_00017)*

**[TPS\_MANI\_01032]**{DRAFT} **Usage of ServiceInterfaceMapping**dIt is possible to derive a dedicated AdaptiveApplicationSwComponentType that implements the mapping functionality. A SwComponentPrototype derived from this so-called *facade* software-component would expose PortPrototypes for each of the ServiceInterfaces.

Other SwComponentPrototypes could then “connect” to the PortPrototypes typed by ServiceInterfaces referenced in the role sourceServiceInterface.

This means that the PortPrototype typed by the ServiceInterface referenced in the role compositeServiceInterface is used for external communication.

PassThroughSwConnectors can be used to describe in the modeled *facade* CompositionSwComponentType which “fine-grained” Ports are combined to a “coarsegrained” Port that is used for network communication. The mapping of Service Interface elements of the “fine-grained” Ports to the Service Interface elements of the “course-grained” Port is described with the ServiceInterfaceMapping or rather ServiceInterfaceElementMapping.c*(RS\_MANI\_00017)*

Please note that the modeling of a *facade* SwComponentType does not make any assumptions about the implementation and about the realization of such a *facade* functionality. The *facade* may be realized by an Adaptive Software Component/Application or it may be realized by a “Network-Daemon”. AUTOSAR does not define any instructions for the implementation of such a functionality and the decision is project specific. The behavioral aspects of such a “facade” (e.g. when is the coarse-grained ServiceInstance offered) are also project-specific and are not predefined by AUTOSAR.

Interface1

Event: event1

Interface2

Event: event2

Interface3

Event: event1, event2

SWC1

P

SWC2

P

SWC3

P

R

R

**Figure 3.23: Concept of a facade software-component**

Figure 3.23 summarizes the idea behind the creation of a *facade* software-component. The latter is able to “bundle” the communication of different PortPrototypes owned by potentially different SwComponentTypes for external communication.

In other words, elements event1 owned by SWC1 and event2 owned by SWC1 are combined into one ServiceInterface used to type one PortPrototype of the *facade* software-component.

From the communication-related outside point-of-view, SWC3 acts like a facade to the “inner structure” created by SWC1 and SWC2 that is, by way of the existence of SWC3, abstracted away.

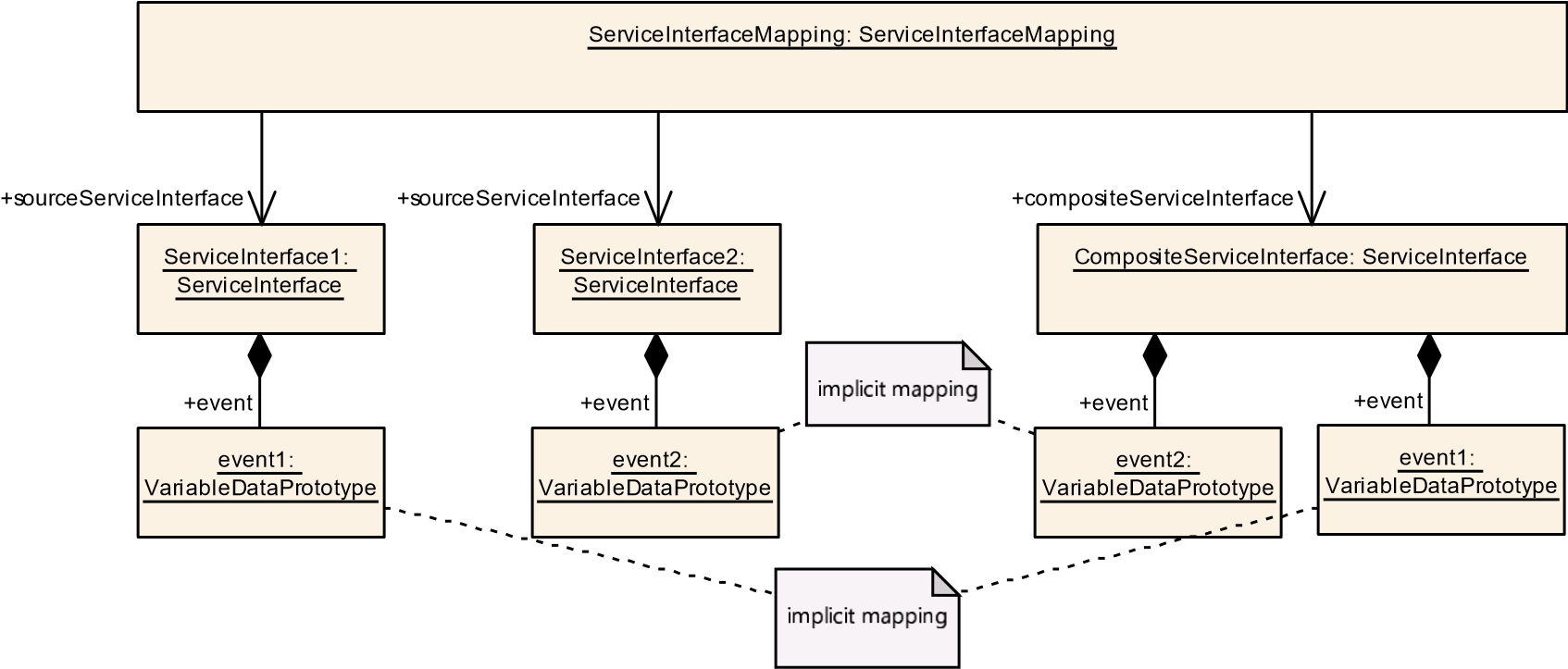
**[constr\_5056]**{DRAFT} **Restriction of sub-class of CompositionSwComponent-**

**Type.connector**dIn the context of a CompositionSwComponentType.connector (transitively) referenced by a Executable.rootSwComponentPrototype, the only supported sub-class of SwConnector is PassThroughSwConnector.c*()*

**[constr\_5057]**{DRAFT} **PassThroughSwConnector and ServiceInterfaceMapping** dIf a PassThroughSwConnector is defined between two Ports in a CompositionSwComponentType either:

* a ServiceInterfaceMapping between the ServiceInterfaces of these two Ports shall be defined and the PassThroughSwConnector shall reference the relevant ServiceInterfaceMapping in the role mapping or
* ServiceInterfaceElementMappings for elements of ServiceInterfaces of the two Ports shall be defined and the PassThroughSwConnector shall reference the relevant ServiceInterfaceElementMappings in the role serviceInterfaceElementMapping.

### c()



**Figure 3.24: Example for the application of a ServiceInterfaceMapping**

**[TPS\_MANI\_01022]**{DRAFT} **Concept behind ServiceInterfaceMapping** dThe concept behind the definition of a ServiceInterfaceMapping is that **all elements** of the sourceServiceInterface are required to have a **counterpart of the same kind** (ServiceInterface.event, ServiceInterface.field, or ServiceInterface.method) and with the identical shortName.c*(RS\_MANI\_00017)*

The regulation stated in [TPS\_MANI\_01022] is exemplified in Figure 3.24.

Please note that the creation of a ServiceInterfaceMapping is considered an atomic step, it is unlikely that such a ServiceInterfaceMapping is partially created and then later finished by a different party.

After all, there are mutually exclusive ways to specify the mapping, and any creator of a partial mapping of ServiceInterfaces could not be sure which of the alternatives apply for a specific pairing of one ServiceInterface with another without already knowing the other ServiceInterface (in which case the mapping can already be completed).

Therefore, there is no need to set the lower multiplicity of the references to ServiceInterface to 0.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ServiceInterfaceMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping | | | |
| ***Note*** | Specifies one ServiceInterfaceMapping that allows to define that a ServiceInterface is composite of several other ServiceInterfaces.  **Tags:**  atp.Status=draft  atp.recommendedPackage=ServiceInterfaceMappings | | | |
| ***Base*** | *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *PortInterfaceMapping*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| composite  ServiceInterface | ServiceInterface | 1 | ref | This represents the composite ServiceInterface.  **Tags:**atp.Status=draft |
| sourceService Interface | ServiceInterface | 1..\* | ref | ServiceInterface that is mapped into the composite ServiceInterface.  **Tags:**atp.Status=draft |

**Table 3.37: ServiceInterfaceMapping**

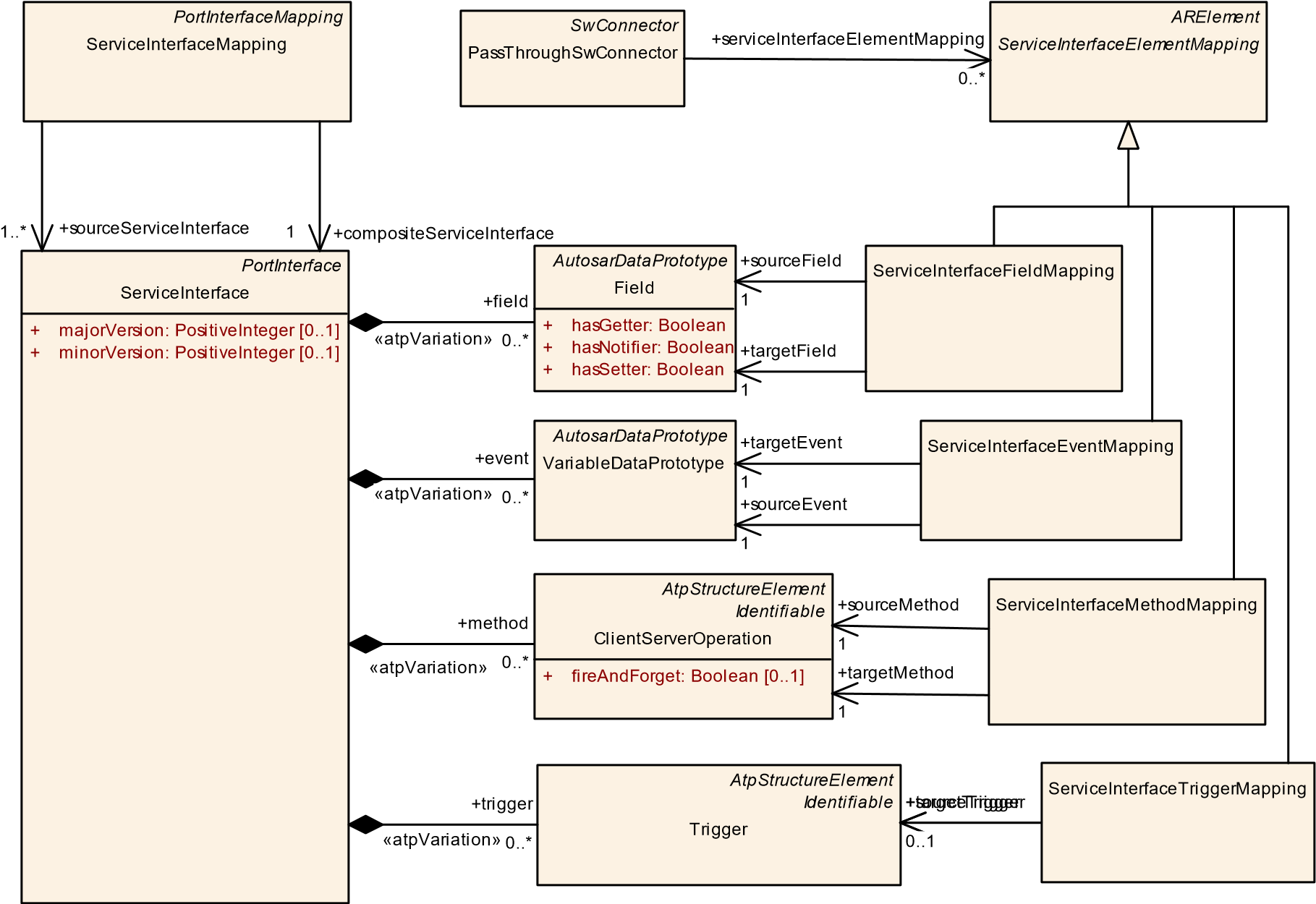
**[TPS\_MANI\_01003]**{DRAFT} **Limitation of the applicability of ServiceInterfaceMapping** dThe applicability of the ServiceInterfaceMapping is limited to cases where the shortNames of the elements of the compositeServiceInterface are **unique** in the context of the compositeServiceInterface.c*(RS\_MANI\_00017)*

Note that the ServiceInterfaceMapping is not an up-front association (by means of SwConnectors) between communication ends in the sense of section 3.4.6.

As stated in [TPS\_MANI\_01032], the ServiceInterfaceMapping allows for the derivation of a facade software-component or a proper configuration of the communication middleware.

The compatibility between the sourceServiceInterfaces and the compositeServiceInterface is achieved by an adequate transformation implemented in the facade software-component or the configuration of the middleware.

Thus, connecting ServiceInterfaces (or parts of them) via ServiceInterfaceMappings is not constrained by any compatibility rules apart from the ones stated in [TPS\_MANI\_01022].



**Figure 3.25: Overview of the modeling of the ServiceInterfaceMapping and ServiceInterfaceElementMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PassThroughSwConnector** | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Composition | | | |
| ***Note*** | This kind of SwConnector can be used inside a CompositionSwComponentType to connect two delegation PortPrototypes. | | | |
| ***Base*** | *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SwConnector* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| providedOuter Port | AbstractProvidedPort  Prototype | 0..1 | ref | This represents the provided outer delegation Port Prototype of the PassThroughSwConnector. |
| requiredOuter Port | AbstractRequiredPort  Prototype | 0..1 | ref | This represents the required outer delegation Port Prototype of the PassThroughSwConnector. |
| serviceInterface  Element  Mapping | ServiceInterface  ElementMapping | \* | ref | Reference to a ServiceInterfaceElementMapping specifying the mapping of unequal named Service Interface elements of the two different ServiceInterfaces typing the two PortPrototypes which are referenced by the PassThroughSwConnector.  **Tags:**atp.Status=draft |

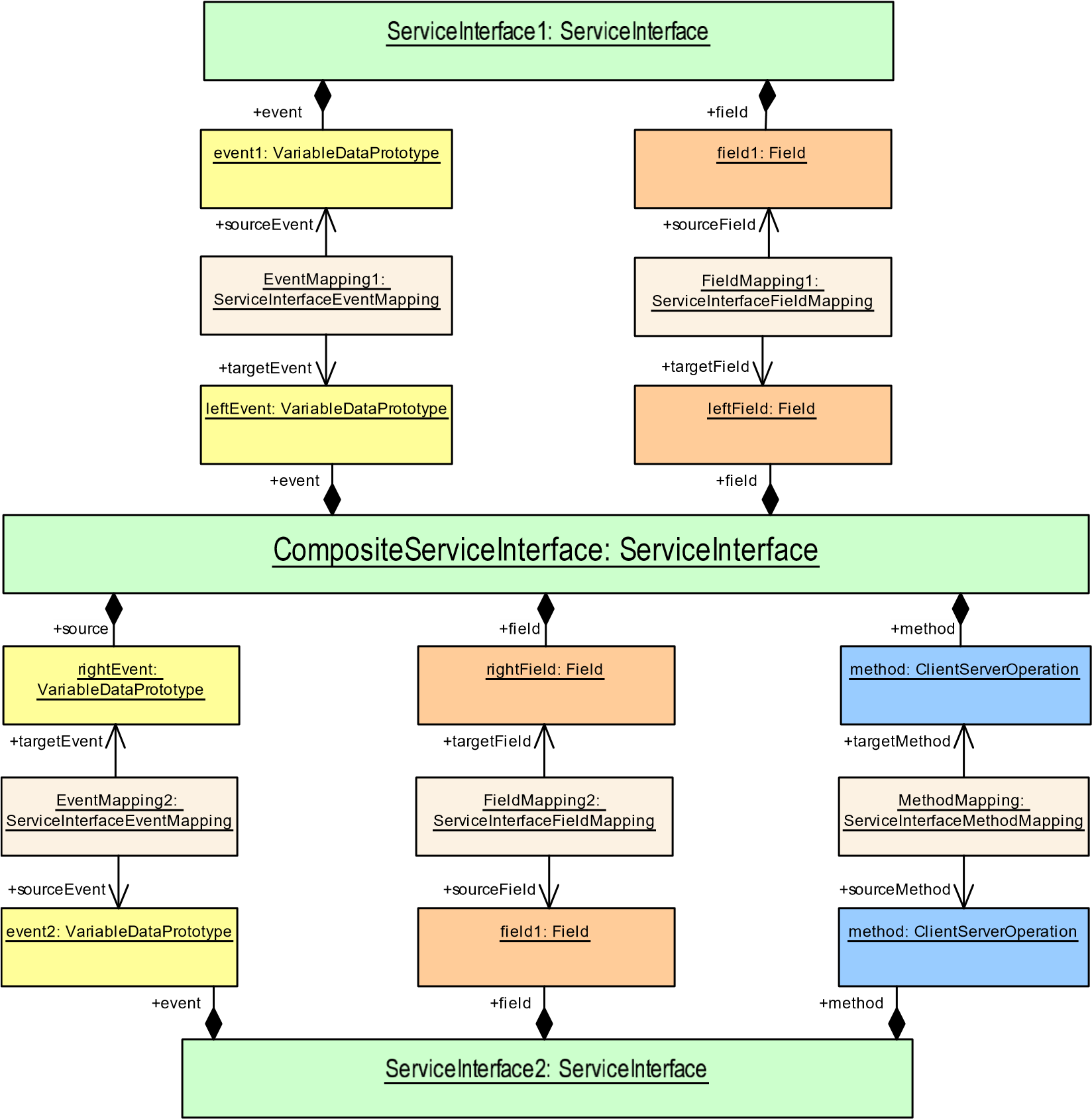
**Table 3.38: PassThroughSwConnector**

## 3.6 Service Interface Element Mapping

**3.6.1 Overview**

The existence of the ServiceInterfaceMapping leaves the question about how ServiceInterfaces where elements have non-matching shortName can be mapped.

The answer to this question is provided by the ability to create an element-wise mapping of elements of the same kind.



**Figure 3.26: Example for a mapping of elements of ServiceInterface**

Figure 3.26 provides an example of how such a mapping on element basis looks like.

Note that, in this example, both ServiceInterface1 and ServiceInterface2 aggregate a field with the shortNamefield1.

This configuration disqualifies the scenario from the application of the ServiceInterfaceMapping, as of [TPS\_MANI\_01003]. The element-wise mapping, however, is able to work around the existence of the shortNamefield1 in both “source” ServiceInterfaces quite nicely:

* ServiceInterface1.field1 is mapped to CompositeServiceInterface.leftField
* ServiceInterface2.field1 is mapped to CompositeServiceInterface.rightField

The formal modeling of the individual mappings is described in section 3.6.

Please note that it is **not intended** to mix a mapping of ServiceInterfaces with a mapping of elements of a ServiceInterface.

In other words, as soon as a mapping between two ServiceInterfaces exists, it is not supported that a mapping between elements of the same pair of ServiceInterfaces exists. This important restriction is formalized by [constr\_1482].

**[constr\_1482]**{DRAFT} **Mapping of service interfaces vs. mapping of service interface elements** dIn order to establish a mapping between a given pair of ServiceInterfaces, at most **one of** the following alternatives can exist:

* the given pair of ServiceInterfaces is referenced by a ServiceInterfaceMapping, where one ServiceInterface is referenced in the role sourceServiceInterface and the other ServiceInterface is referenced in the role compositeServiceInterface.
* an arbitrary mixture of the following options exists:
  + an event aggregated by one of the given ServiceInterfaces is referenced by a ServiceInterfaceEventMapping in the role sourceEvent and one events aggregated by the other given ServiceInterface is referenced by the same ServiceInterfaceEventMapping in the role targetEvent.
  + a trigger aggregated by one of the given ServiceInterfaces is referenced by a ServiceInterfaceTriggerMapping in the role sourceTrigger and one trigger aggregated by the other given ServiceInterface is referenced by the same ServiceInterfaceTriggerMapping in the role targetTrigger.
  + a field aggregated by one of the given ServiceInterfaces is referenced by a ServiceInterfaceFieldMapping in the role sourceField and one fields aggregated by the other given ServiceInterface is referenced by the same ServiceInterfaceFieldMapping in the role targetField.
  + a method aggregated by one of the given ServiceInterfaces is referenced by a ServiceInterfaceMethodMapping in the role sourceMethod and one methods aggregated by the other given ServiceInterface is referenced by the same ServiceInterfaceMethodMapping in the role targetMethod.

### c()

Of course, it is possible that the same ServiceInterface is referenced by mappings to elements and mappings to entire ServiceInterfaces. The limitation formalized in [constr\_1482] always applies to a **pair** of ServiceInterfaces.

A mapping between elements of ServiceInterfaces is modeled by means of a subclass of the abstract meta-class ServiceInterfaceElementMapping.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***ServiceInterfaceElementMapping*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping | | | |
| ***Note*** | This abstract meta-class acts as base class for the mapping of specific elements of a ServiceInterface.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Subclasses*** | ServiceInterfaceEventMapping, ServiceInterfaceFieldMapping, ServiceInterfaceMethodMapping, Service  InterfaceTriggerMapping | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.39: ServiceInterfaceElementMapping**

Please note that the creation of a ServiceInterfaceElementMapping is considered an atomic step, i.e. it is unlikely that such a ServiceInterfaceElementMapping is partially created, handed over to a different party and then later finished by that different party.

After all, there are mutually exclusive ways to specify the mapping, and any creator of a partial mapping of ServiceInterfaces could not be sure which of the alternatives apply for a specific pairing of one ServiceInterface with another without already knowing the other ServiceInterface (in which case the mapping can already be completed).

Therefore, there is no need to set the lower multiplicity of the references to elements of the ServiceInterface to 0.

**[TPS\_MANI\_03277]**{DRAFT}**ServiceInterfaceElementMappings for a subset of elements of a single ServiceInterface**dIf elements of a source ServiceInterface are mapped to target ServiceInterface elements by ServiceInterfaceElementMappings it is allowed that only a subset of those source ServiceInterface elements are mapped to the target ServiceInterface elements.c*(RS\_MANI\_00017)*

With [TPS\_MANI\_03277] use cases are supported as shown in Figure 3.27 where the event4 and method2 of ServiceInterface2 that is provided by AppM1\_2 are used only locally on the Machine1 and event3 of the same ServiceInterface is provided together with the ServiceInterface1 elements as new SOME/IP Service over the network.

Please note that the modeling of a facade SwComponentType does not make any assumptions about the implementation and about the realization of such a facade functionality.

AUTOSAR does not define any instructions for the implementation of such a functionality and the decision is project specific.

The behavioral aspects of such a “facade” (e.g. when is the coarse-grained ProvidedApServiceInstance offered) are also project-specific and are not predefined by AUTOSAR.

AppM1\_1

P

AppM1\_2

P

AppM2\_1

R

Facade

R

R

P

Machine 1

Machine 2

SOME/IP Service with

Event1

Event2

Event3

Method1

Facade

R

P

AppM2\_2

R

P

AppM2\_3

R

Service 4 with

Event2

Service 3 with

Event1, Event3,

Method1

Service 1 with Event1,

Event2, Method1

Service 2 with Event3,

Event4, Method2

**Figure 3.27: Example for ServiceInterfaceElementMapping for a subset of elements of a single ServiceInterface on the provider side**

Similar use cases are applicable on the Consumer side as well as shown in Figure 3.28 where the Facade component is the service consumer of a SOME/IP service with event1, event2 and event3, and method1 but only event1, event3 and method1 are used by Software that is deployed on Machine2.

AppM11

P

AppM12

P

AppM22

R

Facade

R

R

P

Machine 1

Machine 2

SOME/IP Service with

Event1

Event2

Event3

Method1

Facade

R

P

Service 3 with

Event1, Event3,Method1

Service 2 with Event3,

Event4, Method2

Service 1 with

Event1, Event2,

Method1

**Figure 3.28: Example for ServiceInterfaceElementMapping for a subset of elements of a single ServiceInterface on the consumer side**

**3.6.2 Service Interface Event Mapping**

**[TPS\_MANI\_01024]**{DRAFT} **Semantics of ServiceInterfaceEventMapping** dMeta-class ServiceInterfaceEventMapping has the ability to map a ServiceInterface.event referenced in the role sourceEvent explicitly to another ServiceInterface.event referenced in the role targetEvent.c*(RS\_MANI\_00017)*

ServiceInterfaceEventMapping

*AutosarDataPrototype*

VariableDataPrototype

*ARElement*

*ServiceInterfaceElementMapping*

+

targetEvent

1

+

sourceEvent

1

**Figure 3.29: Modeling of the ServiceInterfaceEventMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ServiceInterfaceEventMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping | | | |
| ***Note*** | This meta-class allows to define a mapping between events of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping.  **Tags:**  atp.Status=draft  atp.recommendedPackage=ServiceInterfaceElementMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable*  *Element*, *Referrable*, *ServiceInterfaceElementMapping* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| sourceEvent | VariableDataPrototype | 1 | ref | Reference to an event that is contained in the source ServiceInterface.  **Tags:**atp.Status=draft |
| targetEvent | VariableDataPrototype | 1 | ref | Reference to an event that is contained in the composite ServiceInterface.  **Tags:**atp.Status=draft |

**Table 3.40: ServiceInterfaceEventMapping**

The explicit mapping implemented by ServiceInterfaceEventMapping does **not** require equal shortNames on both sides of the mapping.

It is also possible to map a given event of a given ServiceInterface multiple times in different roles to the ServiceInterface that aggregates the targetEvent, as exemplified by Figure 3.30.

Please note that the mapping of one sourceEvent to different targetEvents does **not** represent a *fan-out* of any kind.

It only means that the sourceEvent will be used in different roles, as specified in the deployment. For more explanation, please find an example of how the role-based mapping of elements of ServiceInterfaces works in Figure A.5.

ServiceInterface1:

ServiceInterface

event1:

VariableDataPrototype

event11:

VariableDataPrototype

CompositeServiceInterface:

ServiceInterface

event10:

VariableDataPrototype

ServiceInterfaceEventMapping1:

ServiceInterfaceEventMapping

ServiceInterfaceEventMapping2:

ServiceInterfaceEventMapping

+

event

targetEvent

+

+

event

+

sourceEvent

targetEvent

+

+

sourceEvent

event

+

**Figure 3.30: Example for the application of a ServiceInterfaceEventMapping**

**3.6.3 Service Interface Trigger Mapping**

**[TPS\_MANI\_03289]**{DRAFT} **Semantics of ServiceInterfaceTriggerMapping** dMeta-class ServiceInterfaceTriggerMapping has the ability to map a ServiceInterface.trigger referenced in the role sourceTrigger explicitly to another ServiceInterface.trigger referenced in the role targetTrigger.c *(RS\_MANI\_00017)*

*AtpStructureElement*

*Identifiable*

Trigger

ServiceInterfaceTriggerMapping

*ARElement*

*ServiceInterfaceElementMapping*

+

sourceTrigger

0..1

+

targetTrigger

0..1

**Figure 3.31: Modeling of the ServiceInterfaceTriggerMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ServiceInterfaceTriggerMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping | | | |
| ***Note*** | This meta-class allows to define a mapping between triggers of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping.  **Tags:**  atp.Status=draft  atp.recommendedPackage=ServiceInterfaceElementMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable*  *Element*, *Referrable*, *ServiceInterfaceElementMapping* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| sourceTrigger | Trigger | 0..1 | ref | Reference to a trigger that is contained in the source ServiceInterface.  **Tags:**atp.Status=draft |
| targetTrigger | Trigger | 0..1 | ref | Reference to a trigger that is contained in the target ServiceInterface.  **Tags:**atp.Status=draft |

**Table 3.41: ServiceInterfaceTriggerMapping**

The explicit mapping implemented by ServiceInterfaceTriggerMapping does **not** require equal shortNames on both sides of the mapping.

It is also possible to map a given trigger of a given ServiceInterface multiple times in different roles to the ServiceInterface that aggregates the targetTrigger.

Please note that the mapping of one sourceTrigger to different targetTriggers does **not** represent a *fan-out* of any kind.

It only means that the sourceTrigger will be used in different roles, as specified in the deployment.

**3.6.4 Service Interface Field Mapping**

**[TPS\_MANI\_01025]**{DRAFT} **Semantics of ServiceInterfaceFieldMapping** dMeta-class ServiceInterfaceFieldMapping has the ability to map a ServiceInterface.field referenced in the role sourceField explicitly to another ServiceInterface.field referenced in the role targetField.c*(RS\_MANI\_00017)*

ServiceInterfaceFieldMapping

*AutosarDataPrototype*

Field

+

hasGetter: Boolean

+

hasNotifier: Boolean

+

hasSetter: Boolean

*ARElement*

*ServiceInterfaceElementMapping*

+

sourceField

1

+

targetField

1

**Figure 3.32: Modeling of the ServiceInterfaceFieldMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ServiceInterfaceFieldMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping | | | |
| ***Note*** | This meta-class allows to define a mapping between fields of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping.  **Tags:**  atp.Status=draft  atp.recommendedPackage=ServiceInterfaceElementMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable*  *Element*, *Referrable*, *ServiceInterfaceElementMapping* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| sourceField | Field | 1 | ref | Reference to a field that is contained in the source ServiceInterface.  **Tags:**atp.Status=draft |
| targetField | Field | 1 | ref | Reference to a field that is contained in the composite ServiceInterface.  **Tags:**atp.Status=draft |

**Table 3.42: ServiceInterfaceFieldMapping**

The explicit mapping implemented by ServiceInterfaceFieldMapping does **not** require equal shortNames on both sides of the mapping.

It is also possible to map a given field of a given ServiceInterface multiple times in different roles to the ServiceInterface that aggregates the targetField, as exemplified by Figure 3.33.

ServiceInterface1:

ServiceInterface

CompositeServiceInterface:

ServiceInterface

field1: Field

field10: Field

field11: Field

ServiceInterfaceFieldMapping1:

ServiceInterfaceFieldMapping

ServiceInterfaceFieldMapping2:

ServiceInterfaceFieldMapping

+

field

sourceField

+

+

targetField

targetField

+

+

sourceField

+

field

+

field

**Figure 3.33: Example for the application of a ServiceInterfaceFieldMapping**

Please note that the mapping of one sourceField to different targetFields does **not** represent a *fan-out* of any kind.

It only means that the sourceField will be used in different roles, as specified in the deployment. For more explanation, please find an example of how the role-based mapping of elements of ServiceInterfaces works in Figure A.5.

**3.6.5 Service Interface Method Mapping**

**[TPS\_MANI\_01026]**{DRAFT} **Semantics of ServiceInterfaceMethodMapping** dMeta-class ServiceInterfaceMethodMapping has the ability to map a ServiceInterface.method referenced in the role sourceMethod explicitly to another ServiceInterface.method referenced in the role targetMethod.c*(RS\_MANI\_-*

*00017)*

ServiceInterfaceMethodMapping

*AtpStructureElement*

*Identifiable*

ClientServerOperation

+

fireAndForget: Boolean

[0..1]

*ARElement*

*ServiceInterfaceElementMapping*

+

sourceMethod

1

+

targetMethod

1

**Figure 3.34: Modeling of the ServiceInterfaceMethodMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ServiceInterfaceMethodMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping | | | |
| ***Note*** | This meta-class allows to define a mapping between methods of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping.  **Tags:**  atp.Status=draft  atp.recommendedPackage=ServiceInterfaceElementMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable*  *Element*, *Referrable*, *ServiceInterfaceElementMapping* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| sourceMethod | ClientServerOperation | 1 | ref | Reference to a method that is contained in the source ServiceInterface.  **Tags:**atp.Status=draft |
| targetMethod | ClientServerOperation | 1 | ref | Reference to a method that is contained in the composite ServiceInterface.  **Tags:**atp.Status=draft |

**Table 3.43: ServiceInterfaceMethodMapping**

The explicit mapping implemented by ServiceInterfaceMethodMapping does **not** require equal shortNames on both sides of the mapping.

It is also possible to map a given method of a given ServiceInterface multiple times in different roles to the ServiceInterface that aggregates the targetMethod, as exemplified by Figure 3.35.

ServiceInterface1:

ServiceInterface

CompositeServiceInterface:

ServiceInterface

ServicInterfaceMethodMapping1:

ServiceInterfaceMethodMapping

ServicInterfaceMethodMapping2:

ServiceInterfaceMethodMapping

method1:

ClientServerOperation

method11:

ClientServerOperation

method10:

ClientServerOperation

+

method

+

method

sourceMethod

+

+

targetMethod

sourceMethod

+

targetMethod

+

+

method

**Figure 3.35: Example for the application of a ServiceInterfaceMethodMapping**

Please note that the mapping of one sourceMethod to different targetMethods does **not** represent a *fan-out* of any kind.

It only means that the sourceMethod will be used in different roles, as specified in the deployment. For more explanation, please find an example of how the role-based mapping of elements of ServiceInterfaces works in Figure A.5.

## 3.7 Persistency Interface

**3.7.1 Overview**

**3.7.1.1 The big Picture**

The *AUTOSAR adaptive platform* foresees a support for access to persistent data by e.g. application software.

ara::com

ara::per

ServiceInstance

Mapping

Mapping

Manifest

Design

Key

-

Value

Storage

(

Configuration on the target hard

-

and software)

Access to individual elements

Software

Component

Access to entire

Key

-

value storage

**Figure 3.36: General approach for the modeling of persistency**

There are some similarities to the communication model in terms of the usage of PortPrototypes.

In contrast to the configuration of communication, however, the modeling approach is much less detailed (i.e. instead of providing access to individual elements of a keyvalue storage an entire key-value storage is accessible on the level of PortPrototype).

The aspect of deployment for the configuration of persistent data is explained in Figure 3.36.

Please note that the AUTOSAR meta-model actually defines two separate metaclasses (for more details, please refer to Figure 3.37) for the different use cases of access to persistent data (i.e. PersistencyKeyValueStorageInterface) and access to files on the file system, or maybe an emulation of one (by means of PersistencyFileStorageInterface).

**3.7.1.2 Modeling of Persistency Interface**

Abstract meta-class PersistencyInterface has been created as a means of categorization, i.e. it allows for easily referring to PortInterfaces dedicated to persistency in general.

As a counterpart to the abstract base class PersistencyInterface on interface level, meta-class PersistencyInterfaceElement has been defined as an abstract base class for elements of a PersistencyInterface.

*PersistencyInterface*

[0..1]

minimumSustainedSize: PositiveInteger

+

+

redundancy: PersistencyRedundancyEnum

[0..1]

[0..1]

updateStrategy: PersistencyCollectionLevelUpdateStrategyEnum

+

PersistencyKeyValueStorageInterface

PersistencyFileStorageInterface

maxNumberOfFiles: PositiveInteger

+

[0..1]

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

PersistencyDataElement

«enumeration»

PersistencyElementLevelUpdateStrategyEnum

overwrite

keepExisting

delete

*DataPrototype*

*AutosarDataPrototype*

*ARElement*

*AtpType*

*AutosarDataType*

«enumeration»

PersistencyCollectionLevelUpdateStrategyEnum

keepExisting

delete

PersistencyFileElement

contentUri: UriString

+

fileName: String

+

*AtpBlueprint*

*AtpBlueprintable*

*AbstractImplementationDataType*

«enumeration»

PersistencyRedundancyEnum

redundant

none

redundantPerElement

*Identifiable*

*PersistencyInterfaceElement*

updateStrategy: PersistencyElementLevelUpdateStrategyEnum

+

[0..1]

+

dataElement

0..\*

fileElement

+

0..\*

dataTypeForSerialization

+

0..\*

«isOfType»

+

type

0..1

{

}

redefines atpType

**Figure 3.37: Specification of PortInterfaces for persistency use cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***PersistencyInterfaceElement*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class provides the abstract ability to define an element of a PortInterface for the support of persistency use cases. **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Subclasses*** | PersistencyDataElement, PersistencyFileElement | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***PersistencyInterfaceElement*** (abstract) | | |  |
| updateStrategy | PersistencyElement  LevelUpdateStrategy  Enum | 0..1 | attr | This attribute can be used to specify the update strategy of the respective PersistencyInterfaceElement.  **Tags:**atp.Status=draft |

**Table 3.44: PersistencyInterfaceElement**

**[TPS\_MANI\_01194]**{DRAFT} **Semantics of PersistencyInterface.minimum-**

**SustainedSize** dAttribute PersistencyInterface.minimumSustainedSize can be used for the definition of a minimum amount of storage that the PersistencyInterface will need to allocate from the application designer’s point of view.c *(RS\_MANI\_00027)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***PersistencyInterface*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class provides the abstract ability to define a PortInterface for the support of persistency use cases.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable* | | | |
| ***Subclasses*** | PersistencyFileStorageInterface, PersistencyKeyValueStorageInterface | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| minimum  SustainedSize | PositiveInteger | 0..1 | attr | The value of this attribute represents the minimum size required at design time for the enclosing Persistency Interface.  **Tags:**atp.Status=draft |
| redundancy | PersistencyRedundancy  Enum | 0..1 | attr | This attribute represents a requirement towards the redundancy of storage. **Tags:**atp.Status=draft |
| redundancy Handling | PersistencyRedundancy  Handling | \* | aggr | This aggregation represents the chosen approaches to handle redundancy for the various use cases implemented by subclasses  **Tags:**atp.Status=draft |
| updateStrategy | PersistencyCollection  LevelUpdateStrategy  Enum | 0..1 | attr | This attribute can be used to specify the update strategy of the respective PersistencyInterface as a whole.  **Tags:**atp.Status=draft |

**Table 3.45: PersistencyInterface**

**3.7.1.3 Redundancy Handling**

**[TPS\_MANI\_01204]**{DRAFT} **Specification of redundancy of persistent data** dThe attribute PersistencyInterface.redundancy can be taken to specify whether the respective key-value storage or file storage shall store data redundantly from the perspective of the designer of the software-component.c*(RS\_MANI\_00027)*

The details are left to an integrator who may also decide to overrule the value of PersistencyInterface.redundancy entirely if there is a use case for that.

|  |  |
| --- | --- |
| ***Enumeration*** | **PersistencyRedundancyEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec |
| ***Note*** | This meta-class provides a way to specify in which way redundancy shall be applied on collection level.  **Tags:**atp.Status=draft |
| ***Literal*** | ***Description*** |
| none | This value represents the requirement that redundancy measures are not applied on persistency storage level.  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |
| redundant | This value represents the requirement that redundancy measures are applied on persistency storage level.  The nature of the redundant persistent storage is not further qualified and subject to integrator decisions.  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |
| redundantPer Element | This value represents the requirement that redundancy measures are applied on key-value level of a key-value storage or on file level of a file storage.  The nature of the redundancy used on the persistent storage is not further qualified and subject to integrator decisions.  **Tags:**  atp.EnumerationLiteralIndex=2 atp.Status=draft |

**Table 3.46: PersistencyRedundancyEnum**

**[TPS\_MANI\_01319]**{DRAFT} **Modeling of redundancy in the context of PersistencyInterface**dAs an alternative to the ability to use PersistencyInterface. redundancy for announcing the consideration of redundancy at all, the design level for persistency also provides the ability to provide a more detailed definition of redundant behavior for both key-value storage and files by means of the aggregation of PersistencyRedundancyHandling at PersistencyInterface.

This modeling is attached to the abstract base class PersistencyInterface in order to let both aspects of persistency (i.e. key-value storage and file storage) on the AUTOSAR adaptive platform benefit from the existence of meta-class PersistencyRedundancyHandling.c*(RS\_MANI\_00027)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***PersistencyRedundancyHandling*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***Note*** | This abstract base class represents a formal description of redundancy.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject* | | | |
| ***Subclasses*** | *PersistencyRedundancyChecksum*, PersistencyRedundancyMOutOfN | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***PersistencyRedundancyHandling*** (abstract) | | | |
| scope | PersistencyRedundancy  HandlingScopeEnum | 0..1 | attr | This attribute controls the scope in which the redundancy handling is applied.  **Tags:**atp.Status=draft |

**Table 3.47: PersistencyRedundancyHandling**

**[TPS\_MANI\_01320]**{DRAFT} **Definition of redundancy on interface level may be overruled in deployment** dThe modeling of redundancy by means of PersistencyInterface.redundancyHandling represents the intention of the designer of the PersistencyInterface.

While this is certainly a valuable input to the deployment phase, it is explicitly foreseen that an integrator may overrule the design decision regarding persistency based on superior knowledge only available at deployment time.c*(RS\_MANI\_00027)*

*PortInterface*

*PersistencyInterface*

+

[0..1]

minimumSustainedSize: PositiveInteger

redundancy: PersistencyRedundancyEnum

+

[0..1]

[0..1]

updateStrategy: PersistencyCollectionLevelUpdateStrategyEnum

+

*PersistencyRedundancyHandling*

[0..1]

scope: PersistencyRedundancyHandlingScopeEnum

+

PersistencyRedundancyCrc

PersistencyRedundancyMOutOfN

m: PositiveInteger

+

n: PositiveInteger

+

*PersistencyRedundancyChecksum*

algorithmFamily: String

+

length: PositiveInteger

+

PersistencyRedundancyHash

initializationVectorLength: PositiveInteger

[0..1]

+

redundancyHandling

+

0..\*

**Figure 3.38: Specification of redundancy on the level of PersistencyInterface**

**[constr\_1746]**{DRAFT} **Mutual exclusive existence of PersistencyInterface. redundancy and PersistencyInterface.redundancyHandling** dFor each PersistencyInterface, either the attribute redundancy or the aggregation of

PersistencyRedundancyHandling in the role redundancyHandling may exist.c

### ()

|  |  |
| --- | --- |
| ***Enumeration*** | **PersistencyRedundancyHandlingScopeEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency |

5

4

|  |  |
| --- | --- |
| ***Enumeration*** | **PersistencyRedundancyHandlingScopeEnum** |
| ***Note*** | This meta-class provides values to control the scope of redundancy measures in the persistency deployment  **Tags:**atp.Status=draft |
| ***Literal*** | ***Description*** |
| persistency  Redundancy  HandlingScope  Element | The redundancy handling shall be applied on element level (key-value pair and file).  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |
| persistency  Redundancy  HandlingScope  Storage | The redundancy handling shall be applied on storage (key-value storage and file storage) level.  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |

**Table 3.48: PersistencyRedundancyHandlingScopeEnum**

**[TPS\_MANI\_01207]**{DRAFT} **Standardized values of attribute PersistencyRedundancyChecksum.algorithmFamily**dThe following values of attribute PersistencyRedundancyChecksum.algorithmFamily are standardized by AUTOSAR:

* **CRC\_J1850**
* **CRC\_CCITT\_FALSE**
* **CRC\_ETHERNET**
* **CRC\_0x42F0E1EBA9EA3693**
* **CRC\_8H2F**
* **CRC\_16ARC**
* **CRC\_32P4**

c*(RS\_MANI\_00027)*

**[constr\_1668]**{DRAFT} **Allowed combinations of PersistencyRedundancyChecksum.length and algorithmFamily**dThe allowed combinations of PersistencyRedundancyChecksum.length and algorithmFamily are documented in

Table 3.49.c*()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **8** | **16** | **32** | **64** |
| **CRC\_J1850** | x |  |  |  |
| **CRC\_CCITT\_FALSE** |  | x |  |  |
| **CRC\_ETHERNET** |  |  | x |  |
| **CRC\_0x42F0E1EBA9EA3693** |  |  |  | x |
| **CRC\_8H2F** | x |  |  |  |
| **CRC\_16ARC** |  | x |  |  |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CRC\_32P4** |  |  | x |  |

**Table 3.49: Allowed combinations of PersistencyRedundancy-**

**Checksum.length and algorithmFamily**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***PersistencyRedundancyChecksum*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***Note*** | Abstract class that defines the common attributes for implementations of redundancy.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *PersistencyRedundancyHandling* | | | |
| ***Subclasses*** | PersistencyRedundancyCrc, PersistencyRedundancyHash | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| algorithmFamily | String | 1 | attr | This attribute identifies the algorithm family that is used to execute the CRC/Hash.  **Tags:**atp.Status=draft |
| length | PositiveInteger | 1 | attr | This attribute describes the length of the CRC/Hash in the unit bits.  **Tags:**atp.Status=draft |

**Table 3.50: PersistencyRedundancyChecksum**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PersistencyRedundancyCrc** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***Note*** | This meta-class formally describes the usage of a CRC for the implementation of redundancy.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *PersistencyRedundancyChecksum*, *PersistencyRedundancyHandling* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.51: PersistencyRedundancyCrc**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PersistencyRedundancyHash** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***Note*** | This meta-class formally describes the usage of a Hash for the implementation of redundancy.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *PersistencyRedundancyChecksum*, *PersistencyRedundancyHandling* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| initialization  VectorLength | PositiveInteger | 0..1 | attr | Length of the initialization vector.  **Tags:**atp.Status=draft |

**Table 3.52: PersistencyRedundancyHash**

**[constr\_10046]**{DRAFT} **Value of PersistencyRedundancyMOutOfN.n** dThe value of Value of PersistencyRedundancyMOutOfN.n shall be set at least to 2 and at most to 255, i.e. the allowed interval is [2..255].c*()*

**[constr\_1751]**{DRAFT} **Value of PersistencyRedundancyMOutOfN.m**dThe value of attribute PersistencyRedundancyMOutOfN.m shall be set at least to 1 and at most to the value of attribute PersistencyRedundancyMOutOfN.n, i.e. the allowed interval is [1..PersistencyRedundancyMOutOfN.n].c*()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PersistencyRedundancyMOutOfN** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***Note*** | This meta-class provides the ability to describe redundancy via an "M out of N" approach. In this case N is the number of copies created and M is the minimum number of identical copies to justify a reliable read access to the data.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *PersistencyRedundancyHandling* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| m | PositiveInteger | 1 | attr | This attribute represents the "M" coordinate in the "M out of N" scheme.  **Tags:**atp.Status=draft |
| n | PositiveInteger | 1 | attr | This attribute represents the "N" coordinate in the "M out of N" scheme.  **Tags:**atp.Status=draft |

**Table 3.53: PersistencyRedundancyMOutOfN**

**3.7.1.4 Update Handling**

**[TPS\_MANI\_01139]**{DRAFT} **Semantics of PersistencyInterface.updateStrategy** dThe attribute PersistencyInterface.updateStrategy can be used to specify the strategy for updating the actual persistent elements used in the context of the PersistencyDeployment that corresponds to PersistencyInterface.

This update strategy shall be applied to the PersistencyInterface as a whole except for the explicitly modeled PersistencyInterfaceElements that define their own updateStrategy.c*(RS\_MANI\_00027)*

|  |  |
| --- | --- |
| ***Enumeration*** | **PersistencyCollectionLevelUpdateStrategyEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |
| ***Note*** | This enumeration provides possible values for the update strategy on interface/storage level.  **Tags:**atp.Status=draft |
| ***Literal*** | ***Description*** |
| delete | The update strategy is to delete all values on the level of the respective collection.  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |
| keepExisting | The update strategy is to keep the existing values on the level of the respective collection.  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |

**Table 3.54: PersistencyCollectionLevelUpdateStrategyEnum**

**[TPS\_MANI\_01140]**{DRAFT} **Semantics of PersistencyInterfaceElement. updateStrategy** dThe attribute PersistencyInterfaceElement.updateStrategy can be used to specify the strategy for updating the actual persistent element that corresponds to PersistencyInterfaceElement.c*(RS\_MANI\_00027)*

|  |  |
| --- | --- |
| ***Enumeration*** | **PersistencyElementLevelUpdateStrategyEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |
| ***Note*** | This enumeration provides possible values for the update strategy on element level.  **Tags:**atp.Status=draft |
| ***Literal*** | ***Description*** |
| delete | The update strategy is to delete the value of the respective data item.  **Tags:**  atp.EnumerationLiteralIndex=2 atp.Status=draft |
| keepExisting | The update strategy is to keep the existing value of the respective data item.  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |
| overwrite | The update strategy is to overwrite the respective data item.  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |

**Table 3.55: PersistencyElementLevelUpdateStrategyEnum**

The behavior of the software in terms of applying an update strategy is explained in detail in [9].

**3.7.2 Persistency Key Value Storage Interface**

**[TPS\_MANI\_01065]**{DRAFT} **Purpose of PersistencyKeyValueStorageInterface** dThe purpose of the PersistencyKeyValueStorageInterface is to support the persistent access to data in a key-value storage.c*(RS\_MANI\_00027)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PersistencyKeyValueStorageInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class provides the ability to implement a PortInterface for supporting persistency use cases for data.  **Tags:**  atp.Status=draft  atp.recommendedPackage=PersistencyKeyValueStorageInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyInterface*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| dataElement | PersistencyData  Element | \* | aggr | This aggregation represents the collection of Persistency DataElements in the context of the enclosing Persistency KeyValueStorageInterface.  **Tags:**atp.Status=draft |

5

**Table 3.56: PersistencyKeyValueStorageInterface**

**[TPS\_MANI\_01135]**{DRAFT} **Semantics of PersistencyKeyValueStorageInterface.dataTypeForSerialization** dThe reference PersistencyKeyValueStorageInterface.dataTypeForSerialization can be taken to get information about data types for which a serialization algorithm has to be generated in order to support the persistent storage of objects of such data type.c*(RS\_MANI\_00027)*

In contrast to other kinds of PortInterfaces it is **not required** to define elements of a PersistencyKeyValueStorageInterface. If this is intended, however, the aggregation PersistencyKeyValueStorageInterface.dataElement shall be used for this purpose.

**[TPS\_MANI\_01138]**{DRAFT} **Semantics of PersistencyKeyValueStorageInterface.dataElement** dBy aggregating PersistencyDataElement in the role dataElement, it is possible to explicitly model key-value pairs (and some of their properties) accessible to the application software within the context of a PersistencyKeyValueStorageInterface.c*(RS\_MANI\_00027)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PersistencyKeyValueStorageInterface** | | |  |
| dataTypeFor Serialization | AbstractImplementation  DataType | \* | ref | This reference identifies the AbstractImplementationData Types that shall be supported for storing in a key-value storage in addition to the types already determined from tha aggregation of PersistencyDataElement.  **Tags:**atp.Status=draft |

**[TPS\_MANI\_01180]**{DRAFT} **Collection of data types that requires serialization support** dThe collection of data types that requires serialization support consists of

* AbstractImplementationDataTypes referenced in the role PersistencyKeyValueStorageInterface.dataTypeForSerialization
* either
  + AbstractImplementationDataTypes taken to type a PersistencyKeyValueStorageInterface.dataElement or
  + AbstractImplementationDataTypes mapped to ApplicationDataTypes taken to type a PersistencyKeyValueStorageInterface.dataElement by means of PortInterfaceToDataTypeMapping. dataTypeMappingSet that also refers to the enclosing PersistencyKeyValueStorageInterface.

c*(RS\_MANI\_00027)*

|  |  |
| --- | --- |
| ***Class*** | **PersistencyDataElement** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PersistencyDataElement** | | | |
| ***Note*** | This meta-class represents the ability to formally specify a piece of data that is subject to persistency in the context of the enclosing PersistencyKeyValueStorageInterface.  PersistencyDataElement represents also a key-value pair of the deployed PersistencyKeyValueStorage and provides an initial value.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *Multilanguage Referrable*, *PersistencyInterfaceElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.57: PersistencyDataElement**

Please note that a PersistencyDataElement can be typed by either an ApplicationDataType or else a CppImplementationDataType.

**3.7.3 Persistency File Storage Interface**

**[TPS\_MANI\_01067]**{DRAFT} **Purpose of PersistencyFileStorageInterface** dThe purpose of meta-class PersistencyFileStorageInterface is to support access to an abstract representation of file storage.c*(RS\_MANI\_00027)*

As far as AUTOSAR persistency is concerned, a file can have binary or text content. If it has text content then the content of the file is expected to be encoded as UTF-8 encoding with UNIX line endings.

**[TPS\_MANI\_01068]**{DRAFT} **Semantics of PersistencyFileStorageInterface.maxNumberOfFiles** dAny PortPrototype typed by a PersistencyFileStorageInterface has the ability to access a number of files.

The upper bound of the number of files represented by a given PortPrototype typed by a PersistencyFileStorageInterface can be configured using the attribute PersistencyFileStorageInterface.maxNumberOfFiles.

The value of attribute PersistencyFileStorageInterface.maxNumberOfFiles **includes** the explicitly modeled PersistencyFileStorageInterface.fileElements.c*(RS\_MANI\_00027)*

Please note that the existence of the PersistencyFileStorageInterface does not violate the restrictions set by the POSIX subset PSE51 defined in IEEE1003.13

[10].

|  |  |
| --- | --- |
| ***Class*** | **PersistencyFileStorageInterface** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PersistencyFileStorageInterface** | | | |
| ***Note*** | This meta-class provides the ability to implement a PortInterface for supporting persistency use cases for files.  **Tags:**  atp.Status=draft  atp.recommendedPackage=PersistencyFileStorageInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyInterface*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| fileElement | PersistencyFileElement | \* | aggr | This aggregation represents the collection of Persistency FileStorages in the context of the enclosing Persistency FileStorageInterface.  **Tags:**atp.Status=draft |
| maxNumberOf  Files | PositiveInteger | 0..1 | attr | This attribute represents the definition of an upper bound for the handling of files at run-time in the context of the enclosing PersistencyFileStorageInterface.  **Tags:**atp.Status=draft |

**Table 3.58: PersistencyFileStorageInterface**

A PortPrototype typed by a PersistencyFileStorageInterface allows for abstracting the actual calls to the operating system away from the scope of the application software and into the modules of the *AUTOSAR adaptive platform*.

**[TPS\_MANI\_01142]**{DRAFT} **Semantics of PersistencyFileElement** dBy aggregating PersistencyFileElement in the role fileElement, it is possible to explicitly model files (and some of their properties) accessible to the application software within the context of a PersistencyFileStorageInterface.c*(RS\_MANI\_00027)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PersistencyFileElement** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class has the ability to represent a file at design time such that it is possible to configure the behavior for accessing the represented file at run-time.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *PersistencyInterfaceElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| contentUri | UriString | 1 | attr | This attribute represents the URI that identifies the initial content of the PersistencyFile.  **Tags:**atp.Status=draft |
| fileName | String | 1 | attr | This attribute holds filename part of the storage location for the PersistencyFileProxy, e.g. file on the file system.  **Tags:**atp.Status=draft |

**Table 3.59: PersistencyFileElement**

**[constr\_1581]**{DRAFT} **Value of fileElement.fileName** dWithin the scope of any given PersistencyFileStorageInterface, the value of all fileElement.

fileName shall be unique.c*()*

**[constr\_10080]**{DRAFT} **Existence of initial values for PersistencyFileElement**dFor each PersistencyFileElement, if the value of attribute updateStrategy is set to the value delete, then attribute PersistencyFileElement.contentUri shall not exist.c*()*

## 3.8 Time Synchronization Interface

The Time Synchronization functional cluster within the Adaptive Platform is responsible to provide various Time-Base Resources for the application to read from or to write to.

In order to interface with the Time Synchronization foundation software an application developer needs to declare which kind of Time-Base Resource this application will interact with.

The interface towards the Time Synchronization follows the generic pattern of PortPrototypes and PortInterfaces which are applied to many use-cases concerning the interaction of application software with platform software.

In contrast to the service based communication, the modeling of platform software interaction using PortPrototypes and PortInterfaces is less detailed. The PortPrototype is a placeholder for the interaction with platform software, it does not model the actually used APIs available for the interaction. The APIs to be used are formally specified in the platform software SWS document, i.e. SWS\_TimeSync [11].

**[TPS\_MANI\_03535]**{DRAFT} **Definition of Time Synchronization interaction** dThe meta-class AbstractSynchronizedTimeBaseInterface together with its sub classes are used to define the interaction of the application software with a Time Synchronization Time Base.c*(RS\_MANI\_00040)*

For more information, please refer to Figure 3.39.

By defining an RPortPrototype which is typed by one of the AbstractSynchronizedTimeBaseInterface sub classes the application indicates that it will access a specific Time Base.

**[TPS\_MANI\_03549]**{DRAFT} **Usage of PortPrototype for the interaction with Time Synchronization** dDepending on the use-case the usage of RPortPrototype or PPortPrototype typed by one of the sub-classes of AbstractSynchronizedTimeBaseInterface shall be used for the interaction with the Time Synchronization.c*(RS\_MANI\_00040)*

The application software may take the active or the passive role in the interaction with functional cluster, thus either a RPortPrototype or a PPortPrototype shall be used to represent this interaction from the application software point of view. The TimeBase Resource instance is identified using the InstanceSpecifier of the respective PortPrototype.

**[TPS\_MANI\_03536]**{DRAFT} **Time Synchronization interaction in a provider role** dThe meta-class SynchronizedTimeBaseProviderInterface is used to indicate the intended interaction with a synchronized global Time Base in a *provider* role.c*(RS\_-*

*MANI\_00040)*

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*AbstractSynchronizedTimeBaseInterface*

SynchronizedTimeBaseProviderInterface

+

timeBaseKind: TimeSynchronizationKindEnum

SynchronizedTimeBaseConsumerInterface

«enumeration»

TimeSynchronizationKindEnum

offset

synchronized

**Figure 3.39: Modeling of Time Synch Interfaces**

When interacting with a synchronized global Time Base in a *provider* role, the application is able to *set* (and *get*) the value of the synchronized global Time Base which is then propagated to the time value on the network.

**[TPS\_MANI\_03537]**{DRAFT} **Time Synchronization interaction in a consumer role** dThe meta-class SynchronizedTimeBaseConsumerInterface is used to indicate the intended interaction with a synchronized global Time Base in a *consumer* role.c*(RS\_MANI\_00040)*

When interacting with a synchronized global Time Base in a *consumer* role, the application is able to only *get* the value of the synchronized global Time Base which is synchronized from a time value coming from the network.

**[TPS\_MANI\_03551]**{DRAFT} **Definition of Time Base kind** dThe attributes SynchronizedTimeBaseProviderInterface.timeBaseKind defines whether the Time Base shall be a synchronized or an offset Time Base.c*(RS\_MANI\_00040)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SynchronizedTimeBaseProviderInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class provides the ability to define a PortInterface for the interaction with a Time Synchronization Provider.  **Tags:**  atp.Status=draft  atp.recommendedPackage=TimeSynchronizationInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AbstractSynchronizedTimeBaseInterface*, *AtpBlueprint*, *AtpBlueprintable*, *Atp*  *Classifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*,  *PortInterface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SynchronizedTimeBaseProviderInterface** | | |  |
| timeBaseKind | TimeSynchronization  KindEnum | 1 | attr | Defines which kind of time base is requested at this interface.  **Tags:**atp.Status=draft |

**Table 3.60: SynchronizedTimeBaseProviderInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SynchronizedTimeBaseConsumerInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class provides the ability to define a PortInterface for the interaction with a Time Synchronization Consumer.  **Tags:**  atp.Status=draft  atp.recommendedPackage=TimeSynchronizationInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AbstractSynchronizedTimeBaseInterface*, *AtpBlueprint*, *AtpBlueprintable*, *Atp*  *Classifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*,  *PortInterface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.61: SynchronizedTimeBaseConsumerInterface**

|  |  |
| --- | --- |
| ***Enumeration*** | **TimeSynchronizationKindEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |
| ***Note*** | Defines the possible kinds of TimeSynchronizationInterfaces.  **Tags:**atp.Status=draft |
| ***Literal*** | ***Description*** |
| offset | Defines that the requested time base shall be an offset time based.  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |
| synchronized | Defines that the requested time base shall be a synchronized time based.  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |

**Table 3.62: TimeSynchronizationKindEnum**

In the example in figure 3.40 the interaction of one Application with several time sync aspects are illustrated.

Application

1

tb

P

tb

C1

SynchronizedTimeBase

ProviderInterface

P

tb

C2

SynchronizedTimeBase

ConsumerInterface

C

**Figure 3.40: Example Application and Time Sync interaction**

The interaction approach is that, for each PortPrototype typed by a sub-class of AbstractSynchronizedTimeBaseInterface, the application developer gains access to the respective kind of Time-Base Resource.

In the application code, the respective Time Base class (as defined in [11]) is constructed using the InstanceSpecifier representing the PortPrototype name.

During application deployment, those PortPrototypes are mapped to actual TimeBase Resources in the Time-Sync Management (see figure 9.24).

## 3.9 Platform Health Management Interface

**3.9.1 Overview**

Platform Health Management functional cluster within the Adaptive Platform is responsible to supervise the execution of applications, monitor their status, and triggering the State Management for respective actions.

In order to interface with the Platform Health Management foundation software an application developer needs to declare which supervisions and status information is provided by the application software and shall be observed by the Platform Health Management.

The interface towards the Platform Health Management follows the generic pattern of PortPrototypes and PortInterfaces which are applied to many use-cases concerning the interaction of application software with platform software.

In contrast to the service based communication, the modeling of platform software interaction using PortPrototypes and PortInterfaces is less detailed. The PortPrototype is a placeholder for the interaction with platform software, it does not model the actually used APIs available for the interaction. The APIs to be used are formally specified in the platform software SWS document [12].

**3.9.2 Supervised Entities and Checkpoints**

The interaction of supervision with the Platform Health Management is defined by PhmSupervisedEntityInterface and PhmCheckpoints.

**[TPS\_MANI\_03500]**{DRAFT} **Definition of Platform Health Management Supervision and Checkpoints** dThe meta-class PhmSupervisedEntityInterface together with the aggregated PhmCheckpoint are used to define the interaction of one Supervised Entity with the Platform Health Management supervision.c*(RS\_MANI\_00032)*

By defining an RPortPrototype which is typed by the PhmSupervisedEntityInterface the application indicates that it wants to report the checkpoints of this PhmSupervisedEntityInterface.

|  |
| --- |
| **Legend** /u/v/t/Swc2 = ShortName Path of Model Element in AUTOSAR Model  = relevant for the creation of InstanceSpecifier |

Executable

„/

a/b/c/

**Exe**

“

RootSwComponentPrototype

„/

a/b/c/Exe/

**Root**

“

SwComponentPrototype

„/

u/v/t/Swc2/

**C2**

"

SwComponentPrototype

„/

q/r/s/Swc1/

**C1**

"

CompositionSwComponent

Type

„/

q/r/s/Swc1"

CompositionSwComponent

Type

„/

u/v/t/Swc2"

AdaptiveApplicationSw

ComponentType

„/

u/v/t/Swc3"

RPortPrototype

„/

u/v/t/Swc3/

**RP**

“

«isOfType»

«isOfType»

InstanceSpecifier: Exe/Root/C1/C2/RP

InstanceRef

phmCheckpoint

contextRootSwComponentPrototype

ContextComponentPrototype[0]

contextComponentPrototype[1]

contextRPortPrototype

PhmSupervisedEntityInterface

„/

f/h/MyPhmInt

“

«isOfType»

PhmCheckpoint

„/

f/h/MyPhmInt/CP

“

targetPhmCheckpoint

«isOfType»

**Figure 3.41: Example for the creation of an InstanceSpecifier of a *SupervisedEntity***

**[TPS\_MANI\_03623]**{DRAFT} **Usage of checkpointId in application code** dThe application code shall only use those PhmCheckpoint.checkpointId values which are defined as members of the PhmSupervisedEntityInterface.checkpoint.c *(RS\_MANI\_00032)*

**[constr\_1727]**{DRAFT} **Qualified combinations of PortPrototypes and PhmSupervisedEntityInterface on application software level** dWithin the context of an Executable of category APPLICATION\_LEVEL the usage of PhmSupervisedEntityInterface is **only** supported for an RPortPrototype.c*()*

The application software takes the active role in the interaction with foundation platform software thus a RPortPrototype is used to represent this interaction from the application software point of view. The *SupervisedEntity* instance is constructed using the InstanceSpecifier of the respective RPortPrototype.

The application code then calls the *ReportCheckpoint* API (defined in [12]) of the *SupervisedEntity* (which has been constructed in the context of the respective RPort-

Prototype typed by the PhmSupervisedEntityInterface) in order to notify the Platform Health Management that a specific PhmCheckpoint has been reached in the program flow.

**[constr\_3530]**{DRAFT} **Mandatory definition of checkpointId** dThe checkpointId shall be defined for every PhmCheckpoint element.c*()*

The checkpointId is used during the call to the *ReportCheckpoint* API as a representation of the PhmCheckpoint.

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*PlatformHealthManagementInterface*

PhmSupervisedEntityInterface

*AtpFeature*

PhmCheckpoint

+

checkpointId: PositiveInteger

+

checkpoint

0..\*

**Figure 3.42: Modeling of Supervised Entities and Checkpoints**

Note that from the application design point of view there are no relations defined between the checkpoints (as to indicate a specific observed order in reporting). The possible transitions between the checkpoints and their timing aspects are defined in the context of the PlatformHealthManagementContribution and described in chapter 9.3.3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PhmSupervisedEntityInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class provides the ability to implement a PortInterface for interaction with the Platform Health Management Supervised Entity.  **Tags:**  atp.Status=draft  atp.recommendedPackage=PlatformHealthManagementInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PlatformHealthManagementInterface*, *Port*  *Interface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| checkpoint | PhmCheckpoint | \* | aggr | Defines the set of checkpoints which can be reported on this supervised entity.  **Tags:**atp.Status=draft |

**Table 3.63: PhmSupervisedEntityInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PhmCheckpoint** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class provides the ability to implement a checkpoint for interaction with the Platform Health Management Supervised Entity.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *AtpFeature*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| checkpointId | PositiveInteger | 1 | attr | Defines the numeric value which is used to indicate the reporting of this Checkpoint to the Phm.  **Tags:**atp.Status=draft |

**Table 3.64: PhmCheckpoint**

**3.9.3 Health Channels**

The interaction of Health Channels with the Platform Health Management is defined by PhmHealthChannelInterface and PhmHealthChannelStatus states.

**[TPS\_MANI\_03534]**{DRAFT} **Definition of Platform Health Management Health**

**Channel** dThe meta-class PhmHealthChannelInterface together with the aggregated PhmHealthChannelStatus are used to define the interaction of one Health Channel with the Platform Health Management.c*(RS\_MANI\_00032)*

By defining a RPortPrototype which is typed by the PhmHealthChannelInterface (see [constr\_1728]) the application indicates that it wants to report the status of this PhmHealthChannelInterface.

The application software takes the active role in the interaction with foundation platform software thus a RPortPrototype is used to represent this interaction from the application software point of view. The *HealthChannel* instance is constructed using the InstanceSpecifier of the respective RPortPrototype.

The application code then calls the *ReportHealthStatus* API (defined in [12]) of the *HealthChannel* (which has been constructed in the context of the respective RPortPrototype typed by the PhmHealthChannelInterface) in order to notify the Platform Health Management that the Health Channel defined by the RPortPrototype has changed its status.

**[constr\_3532]**{DRAFT} **Mandatory definition of statusId** dThe statusId shall be defined for every PhmHealthChannelStatus element.c*()*

**[TPS\_MANI\_03624]**{DRAFT} **Usage of statusId in application code** dThe application code shall only use those PhmHealthChannelStatus.statusId values which are defined as members of the PhmHealthChannelInterface.status.c *(RS\_MANI\_00032)*

**[TPS\_MANI\_03630]**{DRAFT} **Semantics of triggersRecoveryNotification** dThe attribute triggersRecoveryNotification defines whether this specific PhmHealthChannelStatus shall be considered by the PHM as triggering the recovery notification.c*(RS\_MANI\_00032)*

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*PlatformHealthManagementInterface*

PhmHealthChannelInterface

*AtpFeature*

PhmHealthChannelStatus

+

statusId: PositiveInteger

[0..1]

+

triggersRecoveryNotification: Boolean

[0..1]

+

status

0..\*

**Figure 3.43: Modeling of Health Channel**

**[constr\_1728]**{DRAFT} **Qualified combinations of PortPrototypes and PhmHealthChannelInterface on application software level** dWithin the context of an Executable of category APPLICATION\_LEVEL the usage of PhmHealthChannelInterface is **only** supported for a RPortPrototype.c*()*

|  |  |
| --- | --- |
| ***Class*** | **PhmHealthChannelInterface** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |
| ***Note*** | This meta-class provides the ability to implement a PortInterface for interaction with the Platform Health Management Health Channel.  **Tags:**  atp.Status=draft  atp.recommendedPackage=PlatformHealthManagementInterfaces |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PhmHealthChannelInterface** | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PlatformHealthManagementInterface*, *Port*  *Interface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| status | PhmHealthChannel  Status | \* | aggr | Defines the possible set of status information available to the health channel.  **Tags:**atp.Status=draft |

**Table 3.65: PhmHealthChannelInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PhmHealthChannelStatus** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | The PhmHealthChannelStatus specifies one possible status of the health channel.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *AtpFeature*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| statusId | PositiveInteger | 0..1 | attr | Defines the numeric value which is used to indicate the indication of this status the Phm.  **Tags:**atp.Status=draft |
| triggers  Recovery  Notification | Boolean | 0..1 | attr | Defines whether this PhmHealthChannelStatus shall cause the Phm to trigger the Health Channel recovery notification.  True: Indicates unhealthy state. Phm to trigger the Health Channel recovery notification when the Health channel status changes to this state.  False: Indicates healthy state. Phm not to trigger the Health Channel recovery notification when the Health channel status changes to this state.  **Tags:**atp.Status=draft |

**Table 3.66: PhmHealthChannelStatus**

**3.9.4 Recovery notification to State Management**

The Phm monitors the reporting of Supervised Entities and Checkpoints as well as the reported Health Channel status information. In case of violations the Phm can be configured to report the violation to the State Management and let the State Management deal with the recovery activities.

The example in figure 3.44 illustrates the reporting of Supervised Entities by Application 1 and 2. The Phm is configured to perform the supervision of these reported elements. In case of violations the Phm is configured to notify the State Management application to deal with the situation.

**[TPS\_MANI\_01280]**{DRAFT} **Semantics of meta-class PhmSupervisionRecoveryNotificationInterface** dThe recovery notification of a failed Supervision by PHM does issue is to call a piece of code on State Management software level.

The mechanism for activating the code on the level of State Management software is to model a PPortPrototype typed by a PhmSupervisionRecoveryNotificationInterface.c*(RS\_MANI\_00032)*

**[TPS\_MANI\_03631]**{DRAFT} **Semantics of meta-class PhmHealthChannelRecoveryNotificationInterface**dThe recovery notification of a failed HealthChannel monitoring by PHM does issue is to call a piece of code on State Management software level.

The mechanism for activating the code on the level of State Management software is to model a PPortPrototype typed by a PhmHealthChannelRecoveryNotificationInterface.c*(RS\_MANI\_00032)*

The operation to be called by Phm in the context of [TPS\_MANI\_01280] and [TPS\_MANI\_03631] are defined in the Platform Health Management specification document [12].

As already mentioned, the State Management is supposed to implement the recovery actions. This implies that the PhmSupervisionRecoveryNotificationInterface and PhmHealthChannelRecoveryNotificationInterface can only be used in combination with a PPortPrototype. This aspect is clarified by [constr\_1729].

Platform

Health

Manager

Application

2

SE

A

Phm

SupervisedEntity

Interface

Application

1

SE

D

State

Management

Global

Supervision

Supervision

U

Supervision

V

Supervision

W

Supv

Notf

PhmSupervision

RecoveryNotification

Interface

SE

A

**Figure 3.44: Example of a Phm monitoring and recovery setup**

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*PlatformHealthManagementInterface*

*PhmAbstractRecoveryNotificationInterface*

PhmSupervisionRecoveryNotificationInterface

PhmHealthChannelRecoveryNotificationInterface

**Figure 3.45: Modeling of the PhmAbstractRecoveryNotificationInterface**

**[constr\_1729]**{DRAFT} **Qualified combinations of PortPrototypes and PhmSupervisionRecoveryNotificationInterface / PhmHealthChannelRecoveryNotificationInterface on State Management software level** dWithin the context of an Executable of categoryAPPLICATION\_LEVEL the usage of Phm-

SupervisionRecoveryNotificationInterface and PhmHealthChannelRecoveryNotificationInterface is **only** supported for a PPortPrototype.c*()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PhmSupervisionRecoveryNotificationInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class represents a PortInterface that can be taken for implementing a PHM Supervision notification.  **Tags:**  atp.Status=draft  atp.recommendedPackage=PlatformHealthManagementInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PhmAbstractRecoveryNotificationInterface*, *PlatformHealthManagementInterface*, *PortInterface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.67: PhmSupervisionRecoveryNotificationInterface**

|  |  |
| --- | --- |
| ***Class*** | **PhmHealthChannelRecoveryNotificationInterface** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |
| ***Note*** | This meta-class represents a PortInterface that can be taken for implementing a PHM HealthChannel notification.  **Tags:**  atp.Status=draft  atp.recommendedPackage=PlatformHealthManagementInterfaces |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PhmHealthChannelRecoveryNotificationInterface** | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PhmAbstractRecoveryNotificationInterface*, *PlatformHealthManagementInterface*, *PortInterface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.68: PhmHealthChannelRecoveryNotificationInterface**

## 3.10 Diagnostic Interface

**3.10.1 Overview**

On the *AUTOSAR adaptive platform*, dedicated PortInterfaces are defined for the interaction of application-layer software with the AUTOSAR Diagnostic Manager.

In contrast to the conventions on the AUTOSAR classic Platform, these PortInterfaces and, by extension, the standardized ara::diag API **are only used on the application side** of this communication relation.

The interfaces on the side of the AUTOSAR Diagnostic Manager (and thus the part of the implementation of the PortPrototype that faces the AUTOSAR Diagnostic Manager) are **entirely proprietary**. This aspect is depicted in Figure 3.46.

Diagnostic Manager

Application Software

Standardized

ara::diag

proprietary

proprietary

PortPrototype

DiagnosticPortInterface

«isOfType»

**Figure 3.46: Standardized vs. proprietary parts in the implementation of ara::diag**

This arrangement tries to provide the application programmer with the simplest possible API from the application’s point of view. At the same time it hides a lot of the complexity of the interaction between application and Diagnostic Manager behind a solid abstraction layer.

*PortInterface*

*DiagnosticPortInterface*

DiagnosticServiceValidationInterface

DiagnosticSecurityLevelInterface

DiagnosticOperationCycleInterface

*DiagnosticAbstractDataIdentifierInterface*

*DiagnosticAbstractRoutineInterface*

DiagnosticConditionInterface

DiagnosticDTCInformationInterface

DiagnosticDoIPGroupIdentificationInterface

DiagnosticDoIPPowerModeInterface

DiagnosticEventInterface

DiagnosticGenericUdsInterface

DiagnosticIndicatorInterface

DiagnosticMonitorInterface

DiagnosticDownloadInterface

DiagnosticUploadInterface

DiagnosticDoIPActivationLineInterface

DiagnosticAuthenticationInterface

DiagnosticExternalAuthenticationInterface

**Figure 3.47: Modeling of PortInterfaces for diagnostic purposes**

**[TPS\_MANI\_01242]**{DRAFT}**PortInterfaces used for communication with the AUTOSAR Diagnostic Manager**dAll PortInterfaces used for this purpose are derived from the abstract meta-class DiagnosticPortInterface. A DiagnosticPortInterface does not implement a service-oriented communication pattern, in particular there is no explicit service discovery on the API level involved.c*(RS\_MANI\_00061)*

The specializations of DiagnosticPortInterface cover the various aspects of diagnostic communication, e.g. the implementation of diagnostic routines, the reporting of diagnostic events or the access to a Diagnostic Data Identifier (DID).

Figure 3.47 depicts all meta-classes that directly inherit from DiagnosticPortInterface.

|  |  |
| --- | --- |
| ***Class*** | ***DiagnosticPortInterface*** (abstract) |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface |
| ***Note*** | This meta-class serves as an abstract base-class for all diagnostics-related PortInterfaces.  **Tags:**atp.Status=draft |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable* |
| ***Subclasses*** | *DiagnosticAbstractDataIdentifierInterface*, *DiagnosticAbstractRoutineInterface*, DiagnosticAuthentication  Interface, DiagnosticComControlInterface, DiagnosticConditionInterface, DiagnosticDTCInformation  Interface, DiagnosticDoIPActivationLineInterface, DiagnosticDoIPGroupIdentificationInterface, Diagnostic  DoIPPowerModeInterface, DiagnosticDoIPTriggerVehicleAnnouncementInterface, DiagnosticDownload Interface, DiagnosticEcuResetInterface, DiagnosticEventInterface, DiagnosticExternalAuthentication  Interface, DiagnosticGenericUdsInterface, DiagnosticIndicatorInterface, DiagnosticMonitorInterface, DiagnosticOperationCycleInterface, DiagnosticSecurityLevelInterface, DiagnosticServiceValidation Interface, DiagnosticUploadInterface |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***DiagnosticPortInterface*** (abstract) | | |  |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.69: DiagnosticPortInterface**

**3.10.2 Diagnostic Routine Interface**

The convention for the creation of diagnostic routines is to establish at most three methods for each diagnostic routine:

* Start the execution of the routine.
* Stop the execution of the routine.
* Request the results of the routine’s execution.

In response to this convention the DiagnosticRoutineInterface is modeled to aggregate ClientServerOperation in three dedicated roles: start, stop, and requestResult.

**[constr\_1696]**{DRAFT} **ClientServerOperation aggregated by DiagnosticRoutineInterface**dAny ClientServerOperation aggregated by a DiagnosticRoutineInterface shall not define the following attributes:

* fireAndForget
* possibleApError
* possibleApErrorSet

### c()

The arguments to the diagnostic routine shall be modeled as the arguments of the respective ClientServerOperations aggregated in the roles start, stop, and requestResult.

*PortInterface*

*DiagnosticPortInterface*

DiagnosticRoutineInterface

*AtpStructureElement*

*Identifiable*

ClientServerOperation

[0..1]

fireAndForget: Boolean

+

ArgumentDataPrototype

[0..1]

direction: ArgumentDirectionEnum

+

[0..1]

serverArgumentImplPolicy: ServerArgumentImplPolicyEnum

+

*DataPrototype*

*AutosarDataPrototype*

DiagnosticRoutineGenericInterface

*DiagnosticAbstractRoutineInterface*

start

+

0..1

stop

+

0..1

«atpVariation»

+

argument

ordered

}

0..\* {

+

requestResult

0..1

**Figure 3.48: Modeling of DiagnosticRoutineInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticRoutineInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a routine-focused PortInterface for diagnostics on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticAbstractRoutineInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| requestResult | ClientServerOperation | 0..1 | aggr | This represents the request result method of the diagnostic routine.  **Tags:**atp.Status=draft |
| start | ClientServerOperation | 0..1 | aggr | This represents the start method of the diagnostic routine.  **Tags:**atp.Status=draft |
| stop | ClientServerOperation | 0..1 | aggr | This represents the stop method of the diagnostic routine.  **Tags:**atp.Status=draft |

**Table 3.70: DiagnosticRoutineInterface**

In addition to the modeling of "typed" diagnostic routines using the DiagnosticRoutineInterface it is possible to use the DiagnosticRoutineGenericInterface to define a diagnostic routine for which no further formalization is provided.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticRoutineGenericInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a generic Routine-focused PortInterface for diagnostics on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractRoutineInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.71: DiagnosticRoutineGenericInterface**

This means that implicitly there are still up to three methods defined for the already mentioned roles of a diagnostic routine.

However, the methods inside the context of such a generic diagnostic routine would always use plain byte arrays as the arguments and therefore a formalization within the AUTOSAR meta-model does not make sense any longer.

Meta-class DiagnosticAbstractRoutineInterface serves as the abstract base class to all routine-related DiagnosticPortInterfaces on the *AUTOSAR adaptive platform*.

**[constr\_10031]**{DRAFT} **Existence of DiagnosticRoutineInterface.start** dAttribute DiagnosticRoutineInterface.start shall exist **at the time when the creation of the manifest is finished**.c*()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***DiagnosticAbstractRoutineInterface*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class serves as the abstract base class of PortInterfaces dedicated to routine execution on the AUTOSAR adaptive platform.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Subclasses*** | DiagnosticRoutineGenericInterface, DiagnosticRoutineInterface | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.72: DiagnosticAbstractRoutineInterface**

**3.10.3 Interface to Data Identifier and Element of Data Identifier**

The ability to access diagnostic-relevant **data** in the application software is formalized in another abstract sub-class of DiagnosticPortInterface: DiagnosticAbstractDataIdentifierInterface.

Meta-class DiagnosticAbstractDataIdentifierInterface, in turn, defines three concrete subclasses that represent the concrete abilities to access diagnosticrelated data in the application software.

**[TPS\_MANI\_01243]**{DRAFT} **Semantics of DiagnosticDataIdentifierInterface**dDiagnosticDataIdentifierInterface is used to access the content of an entire DID at once.

For this purpose up to two ClientServerOperations are aggregated in the roles read and write, depending on the concrete use case for a specific DiagnosticDataIdentifierInterface.c*(RS\_MANI\_00061)*

**[constr\_10030]**{DRAFT} **Existence of DiagnosticDataIdentifierInterface. read** dAttribute DiagnosticDataIdentifierInterface.read shall exist **at the time when the creation of the manifest is finished**.c*()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticDataIdentifierInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a DID-focused PortInterface for diagnostics on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticAbstractDataIdentifierInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| read | ClientServerOperation | 0..1 | aggr | This represents the method to read the content of a diagnostic data identifier. **Tags:**atp.Status=draft |
| write | ClientServerOperation | 0..1 | aggr | This represents the method to write the contents of a diagnostic data identifier. **Tags:**atp.Status=draft |

**Table 3.73: DiagnosticDataIdentifierInterface**

*PortInterface*

*DiagnosticPortInterface*

DiagnosticDataElementInterface

DiagnosticDataIdentifierGenericInterface

DiagnosticDataIdentifierInterface

*AtpStructureElement*

*Identifiable*

ClientServerOperation

fireAndForget: Boolean

+

[0..1]

ArgumentDataPrototype

+

direction: ArgumentDirectionEnum

[0..1]

[0..1]

serverArgumentImplPolicy: ServerArgumentImplPolicyEnum

+

*DataPrototype*

*AutosarDataPrototype*

*DiagnosticAbstractDataIdentifierInterface*

read

+

0..1

+

write

0..1

read

+

0..1

«atpVariation»

+

argument

0..\* {

ordered

}

**Figure 3.49: Modeling of DiagnosticDataIdentifierInterface**

**[TPS\_MANI\_01244]**{DRAFT} **Semantics of DiagnosticDataElementInterface** dDiagnosticDataElementInterface is used to access the content of an element within a given DID.

For this purpose, a ClientServerOperations is aggregated in the role read.c *(RS\_MANI\_00061)*

Please note that the DiagnosticDataElementInterface *intentionally* does not support a write operation because the consistency of the data in principle cannot be ensured if it is send to the application software piecemeal. Different Processes may be configured to receive the data but it cannot be ensured that all processes are up and running when data is transmitted.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticDataElementInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a element-of-DID-focused PortInterface for diagnostics on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticAbstractDataIdentifierInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| read | ClientServerOperation | 0..1 | aggr | This represents the method to read the content of an element of a diagnostic data identifier.  **Tags:**atp.Status=draft |

**Table 3.74: DiagnosticDataElementInterface**

**[TPS\_MANI\_01245]**{DRAFT} **Semantics of DiagnosticDataIdentifierGenericInterface**dDiagnosticDataIdentifierInterface is used to access the content of an entire DID at once.

For this purpose methods will be defined with a read and write semantics, but these methods will always only provide arguments that are byte-arrays.

Therefore, a further formalization of these methods for reading and writing data within the context of the AUTOSAR meta-model does not make sense and is therefore omitted.c*(RS\_MANI\_00061)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticDataIdentifierGenericInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a generic DID-focused PortInterface for diagnostics on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticAbstractDataIdentifierInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.75: DiagnosticDataIdentifierGenericInterface**

Please note that it is necessary to put some restrictions on the argument **unless** a given DiagnosticDataIdentifierInterface or DiagnosticDataElementInterface aggregates **only one** ClientServerOperation in **either** the role read **or** write.

**[constr\_1697]**{DRAFT} **Restriction for ClientServerOperation aggregated by a DiagnosticDataIdentifierInterface or DiagnosticDataElementInterface**dIf meta-classes DiagnosticDataIdentifierInterface or DiagnosticDataElementInterface aggregate two ClientServerOperations then

* The two ClientServerOperations shall have the same number of arguments.
* The arguments on the nth position in the collection of arguments shall have identical properties, except the direction. In particular, the following conditions shall be fulfilled with respect to attribute direction:
  + Any ArgumentDataPrototype aggregated by a ClientServerOperation that is itself aggregated in either the role DiagnosticDataIdentifierInterface.read or DiagnosticDataElementInterface.read shall set attribute direction to out.
  + Any ArgumentDataPrototype aggregated by a ClientServerOperation that is itself aggregated in the role DiagnosticDataIdentifierInterface.write shall set attribute direction to in.

### c()

**3.10.4 Interface to diagnostic Events**

AUTOSAR defines several subclasses of DiagnosticPortInterface that are dedicated to the handling of diagnostic events.

**[TPS\_MANI\_01246]**{DRAFT} **Semantics of DiagnosticMonitorInterface** dMeta-class DiagnosticMonitorInterface represents the ability to report diagnostic events to the AUTOSAR Diagnostic Manager.c*(RS\_MANI\_00061)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticMonitorInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a monitor-focused PortInterface for diagnostics on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.76: DiagnosticMonitorInterface**

*PortInterface*

*DiagnosticPortInterface*

DiagnosticEventInterface

DiagnosticDTCInformationInterface

DiagnosticMonitorInterface

**Figure 3.50: Modeling of DiagnosticEventInterface**

**[TPS\_MANI\_01247]**{DRAFT} **Semantics of DiagnosticDTCInformationInterface** dMeta-class DiagnosticDTCInformationInterface represents the ability to retrieve information about a given diagnostic trouble code.c*(RS\_MANI\_00061)*

|  |  |
| --- | --- |
| ***Class*** | **DiagnosticDTCInformationInterface** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface |
| ***Note*** | This meta-class represents the ability to implement a PortInterface to access the properties of DTCs on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticDTCInformationInterface** | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.77: DiagnosticDTCInformationInterface**

**[TPS\_MANI\_01248]**{DRAFT} **Semantics of DiagnosticEventInterface**dMetaclass DiagnosticEventInterface represents the ability to retrieve information about a given diagnostic event.c*(RS\_MANI\_00061)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticEventInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a PortInterface to access the properties of diagnostic events on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.78: DiagnosticEventInterface**

**3.10.5 Interface to diagnostic Condition**

**[TPS\_MANI\_01249]**{DRAFT} **Semantics of DiagnosticConditionInterface** dAUTOSAR supports different diagnostic conditions, i.e. enable condition and clear condition. This aspect is represented in the definition of the DiagnosticConditionInterface for the *AUTOSAR adaptive platform*.c*(RS\_MANI\_00061)*

The DiagnosticConditionInterface does not require any further details in its formalization.

*PortInterface*

*DiagnosticPortInterface*

DiagnosticConditionInterface

**Figure 3.51: Modeling of DiagnosticConditionInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticConditionInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a PortInterface to process requests for diagnostic conditions on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.79: DiagnosticConditionInterface**

**3.10.6 Indicator Interface**

**[TPS\_MANI\_01250]**{DRAFT} **Semantics of DiagnosticIndicatorInterface** dThe usage of the DiagnosticIndicatorInterface is foreseen for software that implements a diagnostic indicator (i.e. a warning light on the dashboard).c*(RS\_MANI\_-*

*00061)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticIndicatorInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a PortInterface to implement indicator functionality on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.80: DiagnosticIndicatorInterface**

*PortInterface*

*DiagnosticPortInterface*

DiagnosticIndicatorInterface

**Figure 3.52: Modeling of DiagnosticIndicatorInterface**

The DiagnosticIndicatorInterface does not require any further details in its formalization.

**3.10.7 Security Level Interface**

**[TPS\_MANI\_01251]**{DRAFT} **Semantics of DiagnosticSecurityLevelInterface** dThe usage of the DiagnosticSecurityLevelInterface is foreseen for software that implements the checks for the clearance of a given security level.c*(RS\_-*

*MANI\_00061)*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticSecurityLevelInterface

**Figure 3.53: Modeling of DiagnosticSecurityLevelInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticSecurityLevelInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a security-level-focused PortInterface for diagnostics on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.81: DiagnosticSecurityLevelInterface**

The DiagnosticSecurityLevelInterface does not require any further details in its formalization.

**3.10.8 Service Validation Interface**

**[TPS\_MANI\_01252]**{DRAFT} **Semantics of DiagnosticServiceValidationInterface** dThe usage of the DiagnosticServiceValidationInterface is foreseen for software that implements the checks for clearance on manufacturer or supplier level.c*(RS\_MANI\_00061)*

The DiagnosticServiceValidationInterface does not require any further details in its formalization.

*PortInterface*

*DiagnosticPortInterface*

DiagnosticServiceValidationInterface

**Figure 3.54: Modeling of DiagnosticServiceValidationInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticServiceValidationInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a PortInterface to process requests for service validation on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.82: DiagnosticServiceValidationInterface**

**3.10.9 Operation Cycle Interface**

**[TPS\_MANI\_01253]**{DRAFT} **Semantics of DiagnosticOperationCycleInterface** dThe usage of the DiagnosticOperationCycleInterface is foreseen for software that implements the manages the operation cycles.c*(RS\_MANI\_00061)*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticOperationCycleInterface

**Figure 3.55: Modeling of DiagnosticOperationCycleInterface**

|  |  |
| --- | --- |
| ***Class*** | **DiagnosticOperationCycleInterface** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface |
| ***Note*** | This meta-class represents the ability to implement a PortInterface to process requests for operation cycles on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticOperationCycleInterface** | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.83: DiagnosticOperationCycleInterface**

The DiagnosticOperationCycleInterface does not require any further details in its formalization.

**3.10.10 Generic UDS Interface**

**[TPS\_MANI\_01254]**{DRAFT} **Semantics of DiagnosticGenericUdsInterface** dThe AUTOSAR diagnostic communication API also foresees the existence of one DiagnosticPortInterface that support the implementation of a completely generic handler of a UDS service.c*(RS\_MANI\_00061)*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticGenericUdsInterface

**Figure 3.56: Modeling of DiagnosticGenericUdsInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticGenericUdsInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a generic UDS PortInterface for diagnostics on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.84: DiagnosticGenericUdsInterface**

The DiagnosticGenericUdsInterface does not require any further details in its formalization.

**3.10.11 DoIP Interfaces**

**[TPS\_MANI\_01255]**{DRAFT} **Semantics of DoIP DiagnosticPortInterfaces** dThe AUTOSAR diagnostic communication API also foresees the existence of DiagnosticPortInterfaces to implement functionalities in the context of DoIP operation.

Specifically, the following concrete sub-classes of DiagnosticPortInterface are defined to support the implementation of functionalities in the context of DoIP:

* DiagnosticDoIPGroupIdentificationInterface
* DiagnosticDoIPPowerModeInterface
* DiagnosticDoIPActivationLineInterface
* DiagnosticDoIPTriggerVehicleAnnouncementInterface c*(RS\_MANI\_00061)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticDoIPGroupIdentificationInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a PortInterface to implement the DoIP Group Identification on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.85: DiagnosticDoIPGroupIdentificationInterface**

*PortInterface*

*DiagnosticPortInterface*

DiagnosticDoIPGroupIdentificationInterface

DiagnosticDoIPPowerModeInterface

DiagnosticDoIPActivationLineInterface

DiagnosticDoIPTriggerVehicleAnnouncementInterface

**Figure 3.57: Modeling of DoIP DiagnosticPortInterfaces**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticDoIPPowerModeInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a PortInterface to implement the DoIP Power Mode on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.86: DiagnosticDoIPPowerModeInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticDoIPActivationLineInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a PortInterface to implement the DoIPActivationLine on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.87: DiagnosticDoIPActivationLineInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticDoIPTriggerVehicleAnnouncementInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a PortInterface to implement the DoIPTriggerVehicle Announcement on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.88: DiagnosticDoIPTriggerVehicleAnnouncementInterface**

The DiagnosticDoIPGroupIdentificationInterface, DiagnosticDoIPPowerModeInterface, DiagnosticDoIPActivationLineInterface and DiagnosticDoIPTriggerVehicleAnnouncementInterface do not require any further details in its formalization.

**3.10.12 Diagnostic Interfaces for Upload and Download**

**[TPS\_MANI\_01265]**{DRAFT} **Semantics of DiagnosticDownloadInterface and DiagnosticDownloadInterface** dThe AUTOSAR diagnostic communication API also foresees the existence of DiagnosticPortInterfaces to implement upload and download via diagnostic channels.

Specifically, the following concrete sub-classes of DiagnosticPortInterface are defined to support the implementation of upload and download:

* DiagnosticUploadInterface
* DiagnosticDownloadInterface c*(RS\_MANI\_00061)*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticDownloadInterface

DiagnosticUploadInterface

**Figure 3.58: Modeling of DiagnosticUploadInterface and DiagnosticDownloadInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticUploadInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a PortInterface to process requests for uploading data using diagnostic channels on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.89: DiagnosticUploadInterface**

|  |  |
| --- | --- |
| ***Class*** | **DiagnosticDownloadInterface** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface |
| ***Note*** | This meta-class represents the ability to implement a PortInterface to process requests for downloading data using diagnostic channels on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticDownloadInterface** | |  |  |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.90: DiagnosticDownloadInterface**

The DiagnosticUploadInterface and DiagnosticDownloadInterface do not require any further details in its formalization.

**3.10.13 Interface to support managing the EcuReset**

**[TPS\_MANI\_01332]**{DRAFT} **Semantics of DiagnosticEcuResetInterface** dMeta-class DiagnosticEcuResetInterface represents the ability to support the handling of a request to reset the machine.

This interface will typically be used by the state manager on the AUTOSAR adaptive platform.c*(RS\_MANI\_00061)*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticEcuResetInterface

**Figure 3.59: Modeling of DiagnosticEcuResetInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticEcuResetInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a focused PortInterface for handling the diagnostic service EcuReset on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.91: DiagnosticEcuResetInterface**

**3.10.14 Diagnostic Authentication Interface**

**[TPS\_MANI\_01359]**{DRAFT} **Semantics of DiagnosticAuthenticationInterface** dThe ability to support the diagnostic authentication in the application software is formalized in another sub-class of DiagnosticPortInterface: DiagnosticAuthenticationInterface.c*(RS\_MANI\_00061)*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticAuthenticationInterface

**Figure 3.60: Modeling of DiagnosticAuthenticationInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticAuthenticationInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a focused PortInterface for handling the diagnostic service communication control on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.92: DiagnosticAuthenticationInterface**

**3.10.15 Diagnostic External Authentication Interface**

**[TPS\_MANI\_01353]**{DRAFT} **Semantics of DiagnosticExternalAuthenticationInterface**dThe ability to support the authentication of a diagnostic client in the application software is formalized in another sub-class of DiagnosticPortInterface: DiagnosticExternalAuthenticationInterface.c*(RS\_MANI\_00061)*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticExternalAuthenticationInterface

**Figure 3.61: Modeling of DiagnosticExternalAuthenticationInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticExternalAuthenticationInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a focused PortInterface for handling the diagnostic client authentication on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.93: DiagnosticExternalAuthenticationInterface**

**3.10.16 Diagnostic Communication Control Interface**

**[TPS\_MANI\_01363]**{DRAFT} **Semantics of DiagnosticComControlInterface** dThe ability to support the activation and deactivation of communication with a diagnostic client is formalized in another sub-class of DiagnosticPortInterface: DiagnosticComControlInterface.c*(RS\_MANI\_00061)*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticComControlInterface

**Figure 3.62: Modeling of DiagnosticComControlInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticComControlInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***Note*** | This meta-class represents the ability to implement a focused PortInterface for handling the diagnostic service communication control on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.94: DiagnosticComControlInterface**

## 3.11 Crypto Interfaces

**3.11.1 Interaction with Crypto Software**

**[TPS\_MANI\_03253]**{DRAFT} **Interaction with crypto software** dInteraction with crypto software on an instance of the application software shall be modeled on the basis of the existence of RPortPrototypes typed by a PortInterface that is derived from the abstract meta-class CryptoInterface.c*(RS\_MANI\_00031)*

In contrast to the conventions on the AUTOSAR classic Platform, these CryptoInterfaces are only used on the application side of this communication relation.

The Crypto API is described in [13]. The model-path to an RPortPrototype that is referencing a CryptoInterface is provided by the ara::core::InstanceSpecifier that defines the logical local name used by the application developer in the API call. This local ara::core::InstanceSpecifier is translated at runtime with the information from the deployment model to a specific crypto object, e.g. CryptoKeySlot in a CryptoKeyStorage.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***CryptoInterface*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***Note*** | This meta-class provides the abstract ability to define a PortInterface for the support of crypto use cases.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable* | | | |
| ***Subclasses*** | CryptoCertificateInterface, CryptoKeySlotInterface, CryptoProviderInterface, CryptoTrustMasterInterface | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.95: CryptoInterface**

*AbstractRequiredPortPrototype*

RPortPrototype

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

CryptoKeySlotInterface

+

allocateShadowCopy: Boolean

[0..1]

[0..1]

+

cryptoAlgId: String

+

cryptoObjectType: CryptoObjectTypeEnum

[0..1]

slotCapacity: PositiveInteger

[0..1]

+

+

slotType: CryptoKeySlotTypeEnum

[0..1]

CryptoProviderInterface

CryptoCertificateInterface

isPrivate: Boolean

[0..1]

+

+

writeAccess: Boolean

[0..1]

CryptoTrustMasterInterface

«enumeration»

CryptoObjectTypeEnum

Undefined

SymmetricKey

PrivateKey

PublicKey

Signature

SecretSeed

CryptoKeySlotAllowedModification

+

allowContentTypeChange: Boolean

[0..1]

exportability: Boolean

+

[0..1]

[0..1]

+

maxNumberOfAllowedUpdates: PositiveInteger

[0..1]

restrictUpdate: Boolean

+

CryptoKeySlotContentAllowedUsage

[0..1]

+

allowedKeyslotUsage: String

*CryptoInterface*

«enumeration»

CryptoKeySlotTypeEnum

machine

application

«enumeration»

CryptoKeySlotUsageEnum

verification

encryption

+

keySlotContentAllowedUsage

0..\*

«isOfType»

requiredInterface

+

0..1

}

redefines atpType

{

keySlotAllowedModification

+

0..1

**Figure 3.63: CryptoInterfaces for modeling of the interaction of the Application with the Crypto software**

Figure 3.63 depicts all meta-classes that directly inherit from CryptoInterface.

**3.11.2 Crypto Key Slot Interface**

**[TPS\_MANI\_03254]**{DRAFT} **Modeling of application that uses and modifies a Crypto Key** dApplication software that uses and modifies a Crypto Key is modeled as a AdaptiveApplicationSwComponentType with an RPortPrototype that is typed by a CryptoKeySlotInterface that has the slotType value set to application.c*(RS\_MANI\_00031)*

**[TPS\_MANI\_03255]**{DRAFT} **Modeling of Key Manager application that manages a Crypto Key that is used by Stack Services** dAn Key Manager Application that manages a Crypto Key that is used by Stack Services like COM, Persistency or Diagnostic is modeled as a AdaptiveApplicationSwComponentType with an RPortPrototype that is typed by a CryptoKeySlotInterface that has the slotType value set to machine.c*(RS\_MANI\_00031)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CryptoKeySlotInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***Note*** | This meta-class provides the ability to define a PortInterface for Crypto Key Slots.  **Tags:**  atp.Status=draft  atp.recommendedPackage=CryptoInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| allocateShadow Copy | Boolean | 0..1 | attr | This attribute defines whether a shadow copy of this Key Slot shall be allocated to enable rollback of a failed Key Slot update campaign (see interface BeginTransaction).  **Tags:**atp.Status=draft |
| cryptoAlgId | String | 0..1 | attr | This attribute defines a crypto algorithm restriction (kAlgId Any means without restriction). The algorithm can be specified partially: family & length, mode, padding.  Future Crypto Providers can support some crypto algorithms that are not well known/ standardized today, therefore AUTOSAR doesn’t provide a concrete list of crypto algorithms’ identifiers and doesn’t suppose usage of numerical identifiers. Instead of this a provider supplier should provide string names of supported algorithms in accompanying documentation. The name of a crypto algorithm shall follow the rules defined in the specification of cryptography for Adaptive Platform.  **Tags:**atp.Status=draft |
| cryptoObject Type | CryptoObjectTypeEnum | 0..1 | attr | Object type that can be stored in the slot. If this field contains "Undefined" then mSlotCapacity must be provided and larger then 0  **Tags:**atp.Status=draft |
| keySlotAllowed Modification | CryptoKeySlotAllowed  Modification | 0..1 | aggr | Restricts how this keySlot may be used  **Tags:**atp.Status=draft |
| keySlotContent AllowedUsage | CryptoKeySlotContent  AllowedUsage | \* | aggr | Restriction of allowed usage of a key stored to the slot.  **Tags:**atp.Status=draft |
| slotCapacity | PositiveInteger | 0..1 | attr | Capacity of the slot in bytes to be reserved by the stack vendor. One use case is to define this value in case that the cryptoObjectType is undefined and the slot size can not be deduced from cryptoObjectType and cryptoAlgId.  "0" means slot size can be deduced from cryptoObject Type and cryptoAlgId. **Tags:**atp.Status=draft |
| slotType | CryptoKeySlotType  Enum | 0..1 | attr | This attribute defines whether the keySlot is exclusively used by the Application; or whether it is used by Stack Services and managed by a Key Manager Application.  **Tags:**atp.Status=draft |

**Table 3.96: CryptoKeySlotInterface**

|  |  |
| --- | --- |
| ***Enumeration*** | **CryptoKeySlotTypeEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign |
| ***Note*** | This enumeration defines the options for the usage of a Key Slot in the platform.  **Tags:**atp.Status=draft |

5

4

|  |  |
| --- | --- |
| ***Enumeration*** | **CryptoKeySlotTypeEnum** |
| ***Literal*** | ***Description*** |
| application | KeySlot is used and modified exclusively by the Application.  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |
| machine | Key slot is used by platform modules only. The application manages the key but is not able to use the key.  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |

**Table 3.97: CryptoKeySlotTypeEnum**

|  |  |
| --- | --- |
| ***Enumeration*** | **CryptoObjectTypeEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign |
| ***Note*** | Enumeration of all types of crypto objects, i.e. types of content that can be stored to a key slot.  **Tags:**atp.Status=draft |
| ***Literal*** | ***Description*** |
| PrivateKey | cryp::PrivateKey object  **Tags:**  atp.EnumerationLiteralIndex=2 atp.Status=draft |
| PublicKey | cryp::PublicKey object  **Tags:**  atp.EnumerationLiteralIndex=3 atp.Status=draft |
| SecretSeed | cryp::SecretSeed object  **Tags:**  atp.EnumerationLiteralIndex=5 atp.Status=draft |
| Signature | cryp::Signature object (asymmetric digital signature or symmetric MAC/HMAC)  **Tags:**  atp.EnumerationLiteralIndex=4 atp.Status=draft |
| SymmetricKey | cryp::SymmetricKey object  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |
| Undefined | Object type unknown  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |

**Table 3.98: CryptoObjectTypeEnum**

Please note that the assignment of a CryptoKeySlot to a CryptoProvider is described in the deployment model (Machine Manifest). With this mapping also the assignment of the CryptoKeySlot to a CryptoPrimitive of a CryptoProvider is established.

But the application developer is able to restrict the usage of the CryptoKeySlot to a specific cryptographic algorithm with the attribute cryptoAlgId.

To support crypto algorithms that are not well known/ standardized today, AUTOSAR doesn’t provide a concrete list of crypto algorithm’s identifiers and doesn’t suppose usage of numerical identifiers.

Instead of this a provider supplier should provide string names of supported algorithms in accompanying documentation. The name of a crypto algorithm shall follow the rules defined in the specification of cryptography for Adaptive Platform.

In addition the application developer is able to define further requirements for the usage of the CryptoKeySlot. With the attribute cryptoObjectType the crypto objects that are allowed to be stored in the key slot can be specified.

The allowed modifications of the key slot can be specified by keySlotAllowedModification. The allowed usage of the key slot content can be specified by keySlotContentAllowedUsage.

The Integrator needs to take the defined settings in the Application Design model into account if the assignment to the CryptoKeySlot in the Crypto Storage is performed. Please note that the Application Design model settings are transferred into the deployment model and are therefore are also available at run-time as described in chapter 9.10.2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CryptoKeySlotAllowedModification** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***Note*** | This meta-class restricts the allowed modification of a key stored in the key slot.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| allowContent TypeChange | Boolean | 0..1 | attr | This attribute describes whether the key content type can be changed (true) or not (false), e.g. changing the key from symmetric to RSA.  **Tags:**atp.Status=draft |
| exportability | Boolean | 0..1 | attr | This attribute describes whether the key slot content is allowed to be exported or not.  **Tags:**atp.Status=draft |
| maxNumberOf  AllowedUpdates | PositiveInteger | 0..1 | attr | This attribute describes the maximum updates that are allowed to the slot.  **Tags:**atp.Status=draft |
| restrictUpdate | Boolean | 0..1 | attr | This attribute defines whether restrictions on the number of updates are defined or not.  False: no restriction is placed on the number of updates. True: restrictions are placed on the number of updates with the attribute maxNumberOfAllowedUpdates.  **Tags:**atp.Status=draft |

**Table 3.99: CryptoKeySlotAllowedModification**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CryptoKeySlotContentAllowedUsage** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***Note*** | This meta-class restricts the allowed usage of a key stored in the key slot.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| allowedKeyslot Usage | String | 0..1 | attr | This attribute defines for which operations the KeySlot may be used.  **Tags:**atp.Status=draft |

**Table 3.100: CryptoKeySlotContentAllowedUsage**

**[constr\_5238]**{DRAFT} **CryptoKeySlotAllowedModification.restrictUpdate and the relationship to maxNumberOfAllowedUpdates** dIf the CryptoKeySlotAllowedModification.restrictUpdate is set to true then

CryptoKeySlotAllowedModification.maxNumberOfAllowedUpdates shall be set to a value.c*()*

**[constr\_5239]**{DRAFT} **Predefined values for CryptoKeySlotContentAllowedUsage.allowedKeyslotUsage** dThe following values for CryptoKeySlotContentAllowedUsage.allowedKeyslotUsage are predefined by AUTOSAR:

* ALLOW-DATA-ENCRYPTION,
* ALLOW-DATA-DECRYPTION,
* ALLOW-SIGNATURE,
* ALLOW-VERIFICATION,
* ALLOW-KEY-AGREEMENT,
* ALLOW-KEY-DIVERSIFY,
* ALLOW-DRNG-INIT,
* ALLOW-KDF-MATERIAL,
* ALLOW-KEY-EXPORTING,
* ALLOW-KEY-IMPORTING,
* ALLOW-EXACT-MODE-ONLY,
* ALLOW-DERIVED-DATA-ENCRYPTION,
* ALLOW-DERIVED-DATA-DECRYPTION,
* ALLOW-DERIVED-SIGNATURE,
* ALLOW-DERIVED-VERIFICATION,
* ALLOW-DERIVED-DIVERSIFY,
* ALLOW-DERIVED-DRNG-INIT,
* ALLOW-DERIVED-KDF-MATERIAL,
* ALLOW-DERIVED-KEY-EXPORTING,
* ALLOW-DERIVED-KEY-IMPORTING,
* ALLOW-DERIVED-EXACT-MODE-ONLY

### c()

**3.11.3 Crypto Certificate Interface**

**[TPS\_MANI\_03256]**{DRAFT} **Modeling of application that accesses a Crypto Certificate** dApplication software that accesses a Crypto Certificate is modeled as a

AdaptiveApplicationSwComponentType with an RPortPrototype that is typed by a CryptoCertificateInterface.c*(RS\_MANI\_00031)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CryptoCertificateInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***Note*** | This meta-class provides the ability to define a PortInterface for a CryptoCertificate.  **Tags:**  atp.Status=draft  atp.recommendedPackage=CryptoInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| isPrivate | Boolean | 0..1 | attr | This attribute controls the possibility to access the content of the CryptoCertificateSlot by Find() interfaces of the X509 Provider.  **Tags:**atp.Status=draft |
| writeAccess | Boolean | 0..1 | attr | This attribute defines whether the application has write-access to the CryptoCertificate (True) or only read-access (False).  **Tags:**atp.Status=draft |

**Table 3.101: CryptoCertificateInterface**

**3.11.4 Crypto Provider Interface**

**[TPS\_MANI\_03257]**{DRAFT} **Modeling of application that accesses a Crypto Provider** dApplication software that accesses a Crypto Provider is modeled as a AdaptiveApplicationSwComponentType with an RPortPrototype that is typed by a

CryptoProviderInterface.c*(RS\_MANI\_00031)*

Please note that the CryptoProviderInterface shall be used if the Application needs to access a Crypto Provider to execute keyless operations, e.g. Hashing, Random Number Generation. For cryptographic transformations that require keys the CryptoKeySlotInterface may be used.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CryptoProviderInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***Note*** | This meta-class provides the ability to define a PortInterface for a CryptoProvider.  **Tags:**  atp.Status=draft  atp.recommendedPackage=CryptoInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.102: CryptoProviderInterface**

**3.11.5 Crypto TrustMaster Interface**

**[TPS\_MANI\_03258]**{DRAFT} **Modeling of application designed as trust-master** dApplication software designed as trust-master is modeled as a AdaptiveApplicationSwComponentType with an RPortPrototype that is typed by a CryptoTrustMasterInterface.c*(RS\_MANI\_00031)*

An Application requires TrustMaster privileges to set global (machine-wide) root-oftrust certificates. Note: such a certificate may not be private.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CryptoTrustMasterInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***Note*** | This meta-class provides the ability to define a PortInterface for TrustMaster.  **Tags:**  atp.Status=draft  atp.recommendedPackage=CryptoInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.103: CryptoTrustMasterInterface**

**3.11.6 Linking of Crypto Certificate to a Crypto Key Slot**

It is possible to model a link between a Crypto Certificate and a Crypto KeySlot in the

Application Design with the meta-class SwcServiceDependency that aggregates CryptoCertificateKeySlotNeeds in the role serviceNeeds and RoleBasedPortAssignments that refer to an RPortPrototype that is typed by a CryptoCertificateInterface and an RPortPrototype that is typed by a CryptoKeySlotInterface.

*Identifiable*

*ServiceNeeds*

*Identifiable*

AdaptiveSwcInternalBehavior

*AtpStructureElement*

*Identifiable*

*ServiceDependency*

SwcServiceDependency

*SwComponentType*

AdaptiveApplicationSwComponentType

*CryptoNeeds*

CryptoCertificateKeySlotNeeds

RoleBasedPortAssignment

+

[0..1]

role: Identifier

*AtpBlueprintable*

*AtpPrototype*

*PortPrototype*

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

CryptoKeySlotInterface

allocateShadowCopy: Boolean

[0..1]

+

+

cryptoAlgId: String

[0..1]

[0..1]

+

cryptoObjectType: CryptoObjectTypeEnum

[0..1]

+

slotCapacity: PositiveInteger

slotType: CryptoKeySlotTypeEnum

[0..1]

+

CryptoCertificateInterface

+

isPrivate: Boolean

[0..1]

[0..1]

+

writeAccess: Boolean

RPortPrototype

*AbstractRequiredPortPrototype*

*CryptoInterface*

portPrototype

+

0..1

serviceNeeds

+

0..1

«atpVariation,atpSplitable»

assignedPort

+

0..\*

«isOfType»

+

requiredInterface

0..1

redefines atpType

}

{

«atpVariation,atpSplitable»

internalBehavior

+

0..1

serviceDependency

+

0..\*

**Figure 3.64: Linking of Crypto Certificate with Crypto Key Slot in Application Design**

**[TPS\_MANI\_03259]**{DRAFT} **Linking of Crypto Certificate to a Crypto Key Slot** d

**ServiceNeeds kind** CryptoCertificateKeySlotNeeds **RoleBasedPortAssignment** valid roles:

* CryptoKeySlotInterface [1]
* CryptoCertificateInterface [1]

**RoleBasedDataAssignment**

N/A

**RepresentedPortGroups** N/A c*(RS\_MANI\_00031)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***CryptoNeeds*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***Note*** | Specifies the abstract needs on the configuration of Crypto.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceNeeds* | | | |
| ***Subclasses*** | CryptoCertificateKeySlotNeeds | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.104: CryptoNeeds**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CryptoCertificateKeySlotNeeds** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***Note*** | This meta-class shall be taken to indicate that the SwcServiceDependecy modeled with this kind of ServiceNeeds defines a relationship between a CryptoKeySlot and a CryptoCertificate.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *CryptoNeeds*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceNeeds* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.105: CryptoCertificateKeySlotNeeds**

The following figure 3.65 shows an example how the SwcServiceDependency is used to create a relation between a Crypto Certificate and a Crypto KeySlot.

MyApplication: AdaptiveApplicationSwComponentType

MyCryptoKeyPort:

RPortPrototype

MyCryptoKeyInterface:

CryptoKeySlotInterface

objectType = SymmetricKey

slotType = application

AdaptiveSwcInternalBehavior

:

SwcServiceDependency

:

CryptoCertificateKeySlotNeeds

:

RoleBasedPortAssignment

:

role = CryptoCertificateInterface

RoleBasedPortAssignment

:

role = CryptoKeySlotInterface

MyCryptoCertificateInterface:

CryptoCertificateInterface

MyCryptoCertificatePort:

RPortPrototype

port

+

+

requiredInterface

port

+

+

requiredInterface

**Figure 3.65: Example that shows a link between a Port typed by CryptoKeySlotInterface and a Port typed by CryptoCertificateInterface**

## 3.12 Raw Data Stream Interface

In some cases it is necessary for the application software to be able to process raw binary data streams sent over a communication channel. Obviously, SOME/IP serialization does not make sense in such a scenario, as would the modeling of AutosarDataTypes, i.e. the creation of a ServiceInterface.

Therefore, a different mechanism that actively supports the requirements of raw data streaming is available on the *AUTOSAR adaptive platform*.

As far as the application software is concerned, the interaction with a raw data stream is based on the usage of an RPortPrototype typed by either a RawDataStreamClientInterface or a RawDataStreamServerInterface.

This kind of PortInterface does neither support nor require any elements with a modeled data type, i.e. an AutosarDataType.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***AbstractRawDataStreamInterface*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class serves as an abstract base class for PortInterfaces related to raw data streams.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable* | | | |
| ***Subclasses*** | RawDataStreamClientInterface, RawDataStreamServerInterface | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.106: AbstractRawDataStreamInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **RawDataStreamClientInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class represents the necessary capabilities for raw data streaming on the client side, i.e. the streaming of data that do not undergo any serialization. Each RawDataStreamClientInterface supports the following capabilities without further modeling:   * connect: set up the communication channel * shutdown: close the communication channel * write: send data down the communication channel * read: access incoming data on the communication channel   **Tags:**  atp.Status=draft  atp.recommendedPackage=RawDataStreamInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AbstractRawDataStreamInterface*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*,  *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.107: RawDataStreamClientInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **RawDataStreamServerInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class represents the necessary capabilities for raw data streaming on the server side, i.e. the streaming of data that do not undergo any serialization.  Each RawDataStreamServerInterface supports the following capabilities without further modeling:   * waitForConnection: wait until a communication channel is set up. * shutdown: close the communication channel * write: send data down the communication channel * read: access incoming data on the communication channel   **Tags:**  atp.Status=draft  atp.recommendedPackage=RawDataStreamInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AbstractRawDataStreamInterface*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*,  *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.108: RawDataStreamServerInterface**

## 3.13 Security Event Report Interface

On the *AUTOSAR adaptive platform*, a dedicated PortInterface for the interaction of application-layer software with the AUTOSAR Intrusion Detection System Manager is defined.

The name of this sub-class of abstract meta-class PortInterface is SecurityEventReportInterface.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SecurityEventReportInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class provides the ability to define a PortInterface for the reporting of security events in the context of the intrusion detection system.  **Tags:**  atp.Status=draft  atp.recommendedPackage=SecurityEventReportInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.109: SecurityEventReportInterface**

**[TPS\_MANI\_01340]**{DRAFT} **Semantics of SecurityEventReportInterface** dEach RPortPrototype typed by a SecurityEventReportInterface is able to report exactly one security event.c*(RS\_MANI\_00068)*

**[TPS\_MANI\_01338]**{DRAFT} **Semantics of SecurityEventReportToSecurityEventDefinitionMapping**dThe modeling of the association between a specific security event and the corresponding RPortPrototype typed by a SecurityEventReportInterface is created by means of the SecurityEventReportToSecurityEventDefinitionMapping.c*(RS\_MANI\_00068)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SecurityEventReportToSecurityEventDefinitionMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class represents the ability to map a PortPrototype for reporting a security event to the actual security event that shall be reported by this PortPrototype.  **Tags:**  atp.Status=draft  atp.recommendedPackage=SecurityEventReportToSecurityEventDefinitionMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| reported  SecurityEvent | AbstractRequiredPort  Prototype | 0..1 | iref | This identifies the mapped security event.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**RPortInComposition InstanceRef |
| securityEvent Definition | SecurityEventDefinition | 0..1 | ref | This reference identifies the definition of the security event.  **Tags:**atp.Status=draft |

**Table 3.110: SecurityEventReportToSecurityEventDefinitionMapping**

This meta-class maps the RPortPrototype to a SecurityEventDefinition that itself is part of the so-called Security Extract.

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

SecurityEventReportInterface

SecurityEventDefinition

id: PositiveInteger

+

[0..1]

*ARElement*

IdsDesign

*ARElement*

*IdsCommonElement*

*ARElement*

SecurityEventReportToSecurityEventDefinitionMapping

RPortPrototype

*PortInCompositionTypeInstanceRef*

RPortInCompositionInstanceRef

*PortPrototype*

*AbstractRequiredPortPrototype*

reportedSecurityEvent

+

0..1

+

securityEventDefinition

0..1

+

targetRPort

0..1

{

redefines targetPort

}

«isOfType»

+

requiredInterface

0..1

}

redefines atpType

{

«atpVariation,atpSplitable»

element

+

0..\*

«instanceRef»

+

reportedSecurityEvent

0..1

**Figure 3.66: Specification of the SecurityEventReportInterface and SecurityEventReportToSecurityEventDefinitionMapping**

**[TPS\_MANI\_01339]**{DRAFT} **Existence of the SecurityEventReportToSecurityEventDefinitionMapping is motivated by the AUTOSAR methodology** dThe existence of the SecurityEventReportToSecurityEventDefinitionMapping is motivated by the AUTOSAR methodology. At the point in time when a given SecurityEventReportInterface is defined it could be that the corresponding SecurityEventDefinition is not yet defined.

So it is possible to add this association later. Another reason for the existence of the mapping class is that a specific piece of application software may report different specific security events defined by different OEMs, depending on the deployment of the application software.

Of course, the semantics of the security event all always be either identical or at least comparable, it could still happen that the Id of a security event might change depending on the specific project or simply because different OEMs use different Ids for semantically identical security events.c*(RS\_MANI\_00068)*

## 3.14 Log And Trace Interface

On the *AUTOSAR adaptive platform*, a dedicated PortInterface named LogAndTraceInterface is defined for the interaction of application-layer software with the AUTOSAR Logging and Tracing Functional Cluster.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **LogAndTraceInterface** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***Note*** | This meta-class provides the ability to implement a PortInterface for support of Logging or Tracing.  **Tags:**  atp.Status=draft  atp.recommendedPackage=PortInterfaces | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.111: LogAndTraceInterface**

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

LogAndTraceInterface

**Figure 3.67: Specification of the LogAndTraceInterface**

**[TPS\_MANI\_03284]**{DRAFT} **Semantics of LogAndTraceInterface** dEach

RPortPrototype typed by a LogAndTraceInterface is able to forward logging information onto the external Dlt Log Viewer.c*(RS\_MANI\_00037)* **[constr\_5290]**{DRAFT}**PPortPrototype is not allowed to be typed by LogAndTraceInterface** dA PPortPrototype is not allowed to reference a LogAndTraceInterface in the role providedInterface.c*()*

Please note that the mapping of an RPortPrototype typed by a LogAndTraceInterface to a DltLogSink is described in the deployment model in chapter 9.6.2.

**[constr\_5291]**{DRAFT} **Allowed usage of PortPrototype.logAndTraceMessageCollectionSet**dOnly a PortPrototype that is typed by a LogAndTraceInterface is allowed to reference a LogAndTraceMessageCollectionSet in the role logAndTraceMessageCollectionSet.c*()*

**[TPS\_MANI\_03285]**{DRAFT} **Semantics of PortPrototype.logAndTraceMessageCollectionSet**dWith the PortPrototype.logAndTraceMessageCollectionSet reference it is possible to describe at application design time a collection of Log or Trace messages that will be used by the application.c*(RS\_MANI\_00023)*

## 3.15 Interaction Endpoint for Application

The interaction of software-components with the outside world can take several forms, e.g. service-oriented communication or the interaction with a persistent data storage.

A formal representation of the interaction needs to be described as an anchor point for adding various additional configuration attributes that make sense in this context but would not make sense in the context of a PortInterface.

There is a model element that already has a long-standing tradition in the AUTOSAR meta-model for exactly the described purpose: the PortPrototype.

The following sub-chapters discuss the interaction by means of PortPrototypes with software “outside” a given software-component with the focus on different kinds of interaction that require different ways to further contribute model elements for configuration.

**3.15.1 Service-oriented Communication**

The service-oriented communication by means of PortPrototypes does **not** support the concept of a communication endpoint that is both required and provided **at the same time**. This motivates the existence of [constr\_1473].

**[constr\_1473]**{DRAFT} **No support for PRPortPrototype** dA ServiceInterface shall not be referenced by a PRPortPrototype in the role providedRequiredInterface.c*()*

**[TPS\_MANI\_01039]**{DRAFT} **Representation of provided service** dA **provided service** shall be modeled by means of an PPortPrototype that is typed by a ServiceInterface.c*(RS\_MANI\_00002)*

**[TPS\_MANI\_01040]**{DRAFT} **Representation of required service** dA **required service** shall be modeled by means of an RPortPrototype that is typed by a ServiceInterface.c*(RS\_MANI\_00002)*

For more background regarding the rationale of [constr\_1473], please refer to [1].

Please note that the utilization of service discovery on the *AUTOSAR adaptive platform* means that opposite communication ends **are by design not known upfront**.

As a consequence, it is in general not possible to use AssemblySwConnectors to model a pre-defined relation between two communication endpoints modeled as PortPrototypes.

Independent of the issue described above, it is still necessary to provide means for configuration of a given PortPrototype on different levels:

* The PortPrototype itself (i.e. as a whole) may need to be customized, independently of the kind or number of elements aggregated by the corresponding ServiceInterface. This aspect is discussed in section 3.15.4.
* The usage of elements of the corresponding ServiceInterface may need to be configured for a given PortPrototype. This aspect is discussed in section 3.15.5.

**3.15.2 Interaction with Persistent Key-Value Storage**

The usage of PortPrototypes for the purpose of interacting with *persistent key-value storage* is less restricted than in the case of service-oriented communication. In other words, it is perfectly valid to use a PRPortPrototype where applicable.

**[TPS\_MANI\_01073]**{DRAFT} **Semantics of PortPrototype typed by PersistencyKeyValueStorageInterface** dThe usage of a specific sub-class of PortPrototype typed by PersistencyKeyValueStorageInterface indicates the intended semantics of interaction:

* The usage of a RPortPrototype indicates that the persistent data can only be **read from** the persistent storage.
* The usage of a PPortPrototype indicates that the persistent data can only be **written to** the persistent storage.
* The usage of a PRPortPrototype indicates that the persistent data can be **read from** as well as **written to** the persistent storage.

c*(RS\_MANI\_00027)*

Please note that the PersistencyKeyValueStorageInterface is described in chapter 3.7.2.

**3.15.3 Interaction with Persistent File Storage**

Interaction with **persistent file storage** can involve the ability to read from and write to a file by the same application. Therefore, the existence of a PRPortPrototype typed by a PersistencyFileStorageInterface shall be supported.

**[TPS\_MANI\_01081]**{DRAFT} **Semantics of PortPrototype typed by PersistencyFileStorageInterface**dThe usage of a specific sub-class of PortPrototype typed by PersistencyFileStorageInterface indicates the intended semantics of interaction:

* The usage of a RPortPrototype indicates that the corresponding file(s) can be **opened for read access**.
* The usage of a PPortPrototype indicates that the corresponding file(s) can be **opened or created for write access**. Also, there is the ability to **delete** a file.
* The usage of a PRPortPrototype indicates that the corresponding file(s) can be **opened or created for read and write access**. Also, there is the ability to **delete** a file.

c*(RS\_MANI\_00027)*

Please note that the PersistencyFileStorageInterface is described in chapter

3.7.3.

**3.15.4 Port Prototype Props**

As mentioned before, in some cases a qualification of the semantics of PortPrototypes is necessary. For this purpose, AUTOSAR typically defines a *props* class of some kind. The same approach applies in this situation as well.

In particular, PortPrototype aggregates the abstract meta-class PortPrototypeProps, that in turn starts an inheritance tree of derived meta-classes that have the ability to qualify sub-classes of PortPrototype accordingly.

**[constr\_3642]**{DRAFT} **Restriction of aggregation of PortPrototypeProps to the Adaptive Platform** dThe aggregation of PortPrototypeProps is only supported in the context of a SwComponentType that is (transitively) referenced by rootSwComponentPrototype.c*()*

One example for this approach is the definition of the meta-class RPortPrototypeProps, sketched in Figure 3.68.

**[constr\_3359]**{DRAFT}**RPortPrototypeProps are related only to RPortPrototypes** dThe RPortPrototypeProps shall be aggregated only by a RPortPrototype in the role portPrototypeProps.c*()*

**[TPS\_MANI\_01057]**{DRAFT} **Semantics of RPortPrototypeProps.searchIntention** dThe value of the attribute RPortPrototypeProps.searchIntention clarifies whether the search for a corresponding offer shall be done as a search for all or else as a search for a specific ID.

Typically, a search for any results in a collection of offers while the search for a given id results in just a single offer.c*(RS\_MANI\_00002)*

*AtpBlueprintable*

*AtpPrototype*

*PortPrototype*

RPortPrototypeProps

searchIntention: SearchIntentionEnum

+

[0..1]

*PortPrototypeProps*

*AbstractRequiredPortPrototype*

RPortPrototype

«enumeration»

SearchIntentionEnum

searchForAllInstances

searchForSpecificInstance

+

portPrototypeProps

0..1

**Figure 3.68: Modeling of the RPortPrototypeProps for RPortPrototype**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***PortPrototypeProps*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure | | | |
| ***Note*** | This meta-class represents the ability to define a further qualification of semantics of sub-classes of Port Prototype.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject* | | | |
| ***Subclasses*** | RPortPrototypeProps | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.112: PortPrototypeProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **RPortPrototypeProps** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure | | | |
| ***Note*** | PortPrototypeProps for a RPort.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *PortPrototypeProps* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| searchIntention | SearchIntentionEnum | 0..1 | attr | This attribute is used to specify the intention of the developer of the enclosing software-component in terms of whether the respective PortPrototype shall be use to search for a specific service instance or all instances of the given service.  Please note that the value of this attribute does not create a binding contract. The actual search behavior is defined as part of the service instance manifest.  **Tags:**atp.Status=draft |

**Table 3.113: RPortPrototypeProps**

|  |  |
| --- | --- |
| ***Enumeration*** | **SearchIntentionEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign |
| ***Note*** | This meta-class allows for the definition of a dedicated search intention from the application’s point of view.  **Tags:**atp.Status=draft |
| ***Literal*** | ***Description*** |
| searchForAll Instances | This value represents the intention to search for all instances of the given service.  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |
| searchForSpecific Instance | This value represents the intention to search for a specific instance of the given service.  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |

**Table 3.114: SearchIntentionEnum**

**3.15.5 Port Prototype ComSpec**

**[TPS\_MANI\_01053]**{DRAFT} **Usage of ComSpecs on the *AUTOSAR adaptive platform*** dThe aspect of further qualification of elements of the ServiceInterface used to type given PortPrototype is implemented by means of ComSpecs, i.e. specific sub-classes of the abstract meta-classes RPortComSpec and PPortComSpec.

However, the support for ComSpecs on the *AUTOSAR adaptive platform* only covers a **limited selection** of attributes of a specific ComSpec.c*(RS\_MANI\_00002)*

The details about supported attributes of either a RPortComSpec or PPortComSpec are described in this chapter.

The configuration of transformation capabilities in the context of a ComSpec is possible by means of subclasses of meta-class TransformationComSpecProps.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***TransformationComSpecProps*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Communication | | | |
| ***Note*** | TransformationComSpecProps holds all the attributes for transformers that are port specific. | | | |
| ***Base*** | *ARObject*, *Describable* | | | |
| ***Subclasses*** | EndToEndTransformationComSpecProps, UserDefinedTransformationComSpecProps | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.115: TransformationComSpecProps**

|  |  |
| --- | --- |
| ***Class*** | **UserDefinedTransformationComSpecProps** |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Communication |
| ***Note*** | The UserDefinedTransformationComSpecProps is used to specify port specific configuration properties for custom transformers. |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **UserDefinedTransformationComSpecProps** | | | |
| ***Base*** | *ARObject*, *Describable*, *TransformationComSpecProps* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.116: UserDefinedTransformationComSpecProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **EndToEndTransformationComSpecProps** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Transformer | | | |
| ***Note*** | The class EndToEndTransformationIComSpecProps specifies port specific configuration properties for EndToEnd transformer attributes. | | | |
| ***Base*** | *ARObject*, *Describable*, *TransformationComSpecProps* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| clearFromValid ToInvalid | Boolean | 0..1 | attr | Clear monitoring window on transition from state Valid to state Invalid. |
| disableEndTo EndCheck | Boolean | 0..1 | attr | Disables/Enables the E2E check. The E2Eheader is removed from the payload independent from the setting of this attribute. |
| disableEndTo  EndState  Machine | Boolean | 0..1 | attr | Disables the E2EStateMachine (only E2E check functionality is performed) |
| e2eProfile  Compatibility  Props | E2EProfileCompatibility  Props | 0..1 | ref | Reference to additional settings for the E2E state machine. |
| maxDelta Counter | PositiveInteger | 0..1 | attr | Maximum allowed difference between two counter values of two consecutively received valid messages. For example, if the receiver gets data with counter 1 and Max DeltaCounter is 3, then at the next reception the receiver can accept Counters with values 2, 3 or 4. |
| maxErrorState  Init | PositiveInteger | 0..1 | attr | Maximal number of checks in which ProfileStatus equal to E2E\_P\_ERROR was determined, within the last Window Size checks, for the state E2E\_SM\_INIT.  The minimum value is 0. |
| maxErrorState Invalid | PositiveInteger | 0..1 | attr | Maximal number of checks in which ProfileStatus equal to E2E\_P\_ERROR was determined, within the last Window Size checks, for the state E2E\_SM\_INVALID.  The minimum value is 0. |
| maxErrorState Valid | PositiveInteger | 0..1 | attr | Maximal number of checks in which ProfileStatus equal to E2E\_P\_ERROR was determined, within the last Window Size checks, for the state E2E\_SM\_VALID.  The minimum value is 0. |
| minOkStateInit | PositiveInteger | 0..1 | attr | Minimal number of checks in which ProfileStatus equal to E2E\_P\_OK was determined, within the last WindowSize checks, for the state E2E\_SM\_INIT.  The minimum value is 1. |
| minOkState  Invalid | PositiveInteger | 0..1 | attr | Minimal number of checks in which ProfileStatus equal to E2E\_P\_OK was determined, within the last WindowSize checks, for the state E2E\_SM\_INVALID.  The minimum value is 1. |
| minOkState Valid | PositiveInteger | 0..1 | attr | Minimal number of checks in which ProfileStatus equal to E2E\_P\_OK was determined, within the last WindowSize checks, for the state E2E\_SM\_VALID.  The minimum value is 1. |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **EndToEndTransformationComSpecProps** | | |  |
| windowSizeInit | PositiveInteger | 0..1 | attr | Size of the monitoring window of state Init for the E2E state machine. |
| windowSize  Invalid | PositiveInteger | 0..1 | attr | Size of the monitoring window of state Invalid for the E2E state machine. |
| windowSize Valid | PositiveInteger | 0..1 | attr | Size of the monitoring window of state Valid for the E2E state machine. |

**Table 3.117: EndToEndTransformationComSpecProps**

**[TPS\_MANI\_01327]**{DRAFT} **Value of EndToEndTransformationComSpecProps.disableEndToEndCheck vs. value of EndToEndTransformationComSpecProps.disableEndToEndStateMachine** dIf the value of attribute

EndToEndTransformationComSpecProps.disableEndToEndCheck is set to True, then the value of attribute EndToEndTransformationComSpecProps.

disableEndToEndStateMachine shall be ignored.c*(RS\_MANI\_00028)*

**3.15.5.1 Port Prototypes typed by Service Interfaces**

**3.15.5.1.1 Receiver ComSpec**

The ReceiverComSpec needs an attribute that indicates whether the enclosing AdaptiveApplicationSwComponentType has an intention to actually access the referenced dataElement. This attribute represents a security feature related to identity and access management [14].

Specifically, this aspect is typically summarized as a capability of the software, i.e. the

AdaptiveApplicationSwComponentType expresses expresses its capability with respect to the specific dataElement. The term "capability" is an integral part of the jargon in the domain of identity and access management.

However, outside the identity and access management domain, this terminology is sometimes hard to motivate. What could be motivated is that the AdaptiveApplicationSwComponentType expresses its *intent* to actually access the dataElement.

From that perspective, the process of adding an event to a ServiceInterface adds the capability to use the dataElement. But whether the software that uses the ServiceInterface actually intends to access the dataElement can be expressed by an attribute in the ReceiverComSpec named receiverIntent.

*RPortComSpec*

*ReceiverComSpec*

[0..1]

+

receiverIntent: ReceiverIntentEnum

QueuedReceiverComSpec

VariableDataPrototype

*PortPrototype*

*AbstractRequiredPortPrototype*

*Describable*

*TransformationComSpecProps*

«enumeration»

ReceiverIntentEnum

willReceive

wontReceive

EndToEndTransformationComSpecProps

clearFromValidToInvalid: Boolean

[0..1]

+

+

disableEndToEndCheck: Boolean

[0..1]

disableEndToEndStateMachine: Boolean

[0..1]

+

+

maxDeltaCounter: PositiveInteger

[0..1]

maxErrorStateInit: PositiveInteger

+

[0..1]

+

maxErrorStateInvalid: PositiveInteger

[0..1]

maxErrorStateValid: PositiveInteger

[0..1]

+

+

minOkStateInit: PositiveInteger

[0..1]

+

minOkStateInvalid: PositiveInteger

[0..1]

+

minOkStateValid: PositiveInteger

[0..1]

[0..1]

+

windowSizeInit: PositiveInteger

+

windowSizeInvalid: PositiveInteger

[0..1]

+

windowSizeValid: PositiveInteger

[0..1]

*DataPrototype*

*AutosarDataPrototype*

UserDefinedTransformationComSpecProps

ReceptionComSpecProps

[0..1]

dataUpdatePeriod: TimeValue

+

timeout: TimeValue

[0..1]

+

+

transformationComSpecProps

0..\*

+

requiredComSpec

0..\*

+

receptionProps

0..1

dataElement

+

0..1

**Figure 3.69: Modeling of the ReceiverComSpec on the *AUTOSAR adaptive platform***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***ReceiverComSpec*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Communication | | | |
| ***Note*** | Receiver-specific communication attributes (RPortPrototype typed by ServiceInterface) that are relevant for events and field notifiers. | | | |
| ***Base*** | *ARObject*, *RPortComSpec* | | | |
| ***Subclasses*** | NonqueuedReceiverComSpec, QueuedReceiverComSpec | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| dataElement | AutosarDataPrototype | 0..1 | ref | Data element these attributes belong to. |
| receiverIntent | ReceiverIntentEnum | 0..1 | attr | This attribute represents the expressed intent of the receiver. The receiver may decide to claim that existing resources of a ServiceInterface are expressly not used by this specific receiver. The conceptual background of this claim may be driven by security, safety, etc.  **Tags:**atp.Status=draft |
| receptionProps | ReceptionComSpec  Props | 0..1 | aggr | "This aggregation represents the definition transmission props in the context of the enclosing ReceiverComSpec. |
| transformation  ComSpecProps | TransformationCom  SpecProps | \* | aggr | This references the TransformationComSpecProps which define port-specific configuration for data transformation. |

**Table 3.118: ReceiverComSpec**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **QueuedReceiverComSpec** | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Communication | | | |
| ***Note*** | Communication attributes specific to queued receiving. | | | |
| ***Base*** | *ARObject*, *RPortComSpec*, *ReceiverComSpec* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.119: QueuedReceiverComSpec**

**[TPS\_MANI\_01106]**{DRAFT} **Specification of intentions for the receiver of events or field notifiers** dThe attribute ReceiverComSpec.receiverIntent can be used to specify whether the software actually intends to access the referenced events or field notifier or whether it explicitly states that it is not interested in the value.c*(RS\_MANI\_00034)*

|  |  |
| --- | --- |
| ***Enumeration*** | **ReceiverIntentEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec |
| ***Note*** | This meta-class represents the intent to specify how a given ServiceInterface is used from the perspective of a given event receiver.  **Tags:**atp.Status=draft |
| ***Literal*** | ***Description*** |
| willReceive | The receiver will receive the event or field notifier.  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |
| wontReceive | The receiver won’t receive the event or field notifier.  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |

**Table 3.120: ReceiverIntentEnum**

**[TPS\_MANI\_03132]**{DRAFT} **Semantics of E2E attributes in ReceiverComSpec** dThe EndToEndTransformationComSpecProps shall be used for the specification of RPortPrototype-specific configuration options related to end-to-end protection of events or field notifiers.c*(RS\_MANI\_00028)*

**3.15.5.1.2 Sender ComSpec**

The SenderComSpec is modeled in the same way as described in the Software Component Template [1].

*PPortComSpec*

*SenderComSpec*

senderIntent: SenderIntentEnum

+

[0..1]

«atpVariation»

[0..1]

usesEndToEndProtection: Boolean

+

*ValueSpecification*

shortLabel: Identifier

+

[0..1]

VariableDataPrototype

*PortPrototype*

*AbstractProvidedPortPrototype*

QueuedSenderComSpec

«enumeration»

SenderIntentEnum

willSend

wontSend

*DataPrototype*

*AutosarDataPrototype*

FieldSenderComSpec

TransmissionComSpecProps

+

[0..1]

dataUpdatePeriod: TimeValue

[0..1]

minimumSendInterval: TimeValue

+

[0..1]

transmissionMode: TransmissionModeDefinitionEnum

+

+

transmissionProps

0..1

+

dataElement

0..1

+

initValue

1

providedComSpec

+

0..\*

.

**Figure 3.70: Modeling of the SenderComSpec on the *AUTOSAR adaptive platform***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***SenderComSpec*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Communication | | | |
| ***Note*** | Communication attributes for a sender port (PPortPrototype typed by ServiceInterface) that are relevant for events and field notifiers. | | | |
| ***Base*** | *ARObject*, *PPortComSpec* | | | |
| ***Subclasses*** | FieldSenderComSpec, NonqueuedSenderComSpec, QueuedSenderComSpec | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| dataElement | AutosarDataPrototype | 0..1 | ref | Data element these quality of service attributes apply to. |
| senderIntent | SenderIntentEnum | 0..1 | attr | This attribute represents the expressed intent of the sender. The sender may decide to claim that existing resources of a ServiceInterface are expressly not used by this specific sender. The conceptual background of this claim may be driven by security, safety, etc.  **Tags:**atp.Status=draft |
| transmission Props | TransmissionComSpec  Props | 0..1 | aggr | This aggregation represents the definition transmission props in the context of the enclosing SenderComSpec. |
| usesEndToEnd  Protection | Boolean | 0..1 | attr | This indicates whether the corresponding dataElement shall be transmitted using end-to-end protection.  **Stereotypes:** atpVariation  **Tags:**vh.latestBindingTime=preCompileTime |

**Table 3.121: SenderComSpec**

**[TPS\_MANI\_03210]**{DRAFT} **Specification of event specific communication attributes** dThe meta-class QueuedSenderComSpec can be used to specify communication attributes that are relevant for an event on the sender side.c*(RS\_MANI\_00002)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **QueuedSenderComSpec** | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Communication | | | |
| ***Note*** | Communication attributes specific to distribution of events (PPortPrototype, SenderReceiverInterface and dataElement carries an "event"). | | | |
| ***Base*** | *ARObject*, *PPortComSpec*, *SenderComSpec* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.122: QueuedSenderComSpec**

**[TPS\_MANI\_03211]**{DRAFT} **Specification of field specific communication attributes** dThe meta-class FieldSenderComSpec can be used to specify communication attributes that are relevant for a field on the sender side.c*(RS\_MANI\_00002)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **FieldSenderComSpec** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec | | | |
| ***Note*** | Port specific communication attributes for a Field that is defined in a ServiceInterface.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *PPortComSpec*, *SenderComSpec* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| initValue | ValueSpecification | 1 | aggr | Initial value for a Field that is set before the Service Interface is offered.  **Tags:**atp.Status=draft |

**Table 3.123: FieldSenderComSpec**

**[TPS\_MANI\_03212]**{DRAFT} **Specification of initial value for a field** dThe attribute FieldSenderComSpec.initValue can be used to specify an initial Value for a field.c*(RS\_MANI\_00002)*

A field has a valid value at any time as described in subsection 3.4.4. ara::com ensures that a service implementation providing a field has a field value before the field becomes visible to potential consumers.

This is explained in more detail in [15] where it is defined that the initial field value shall be set at least once via Update() by the application code before OfferService() gets called.

Custom-code (e.g. component model above ara::com) may use the defined initValue to call Field.Update(initValue).

**[TPS\_MANI\_01107]**{DRAFT} **Specification of intentions for the sender of events or field notifiers** dThe attribute SenderComSpec.senderIntent can be used to specify whether the software actually intends to send the referenced events or field notifier.c*(RS\_MANI\_00034)*

|  |  |
| --- | --- |
| ***Enumeration*** | **SenderIntentEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec |
| ***Note*** | This meta-class represents the intent to specify how a given ServiceInterface is used from the perspective of a given event sender.  **Tags:**atp.Status=draft |
| ***Literal*** | ***Description*** |
| willSend | The sender will send the event or field notifier.  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |
| wontSend | The sender won’t send the event or field notifier.  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |

**Table 3.124: SenderIntentEnum**

**3.15.5.1.3 Client ComSpec**

The ClientComSpec undergoes extensions for the *AUTOSAR adaptive platform*, namely the ability to refer to the getter and setter method of a field and the definition of intentions.

*RPortComSpec*

ClientComSpec

[0..1]

clientIntent: ClientIntentEnum

+

+

[0..1]

endToEndCallResponseTimeout: TimeValue

*AtpStructureElement*

*Identifiable*

ClientServerOperation

+

[0..1]

fireAndForget: Boolean

*PortPrototype*

*AbstractRequiredPortPrototype*

*Describable*

*TransformationComSpecProps*

«enumeration»

ClientIntentEnum

willCall

wontCall

*AutosarDataPrototype*

Field

+

hasGetter: Boolean

hasNotifier: Boolean

+

hasSetter: Boolean

+

EndToEndTransformationComSpecProps

clearFromValidToInvalid: Boolean

+

[0..1]

+

disableEndToEndCheck: Boolean

[0..1]

[0..1]

+

disableEndToEndStateMachine: Boolean

+

maxDeltaCounter: PositiveInteger

[0..1]

+

maxErrorStateInit: PositiveInteger

[0..1]

+

maxErrorStateInvalid: PositiveInteger

[0..1]

+

maxErrorStateValid: PositiveInteger

[0..1]

minOkStateInit: PositiveInteger

[0..1]

+

[0..1]

+

minOkStateInvalid: PositiveInteger

[0..1]

minOkStateValid: PositiveInteger

+

[0..1]

windowSizeInit: PositiveInteger

+

+

[0..1]

windowSizeInvalid: PositiveInteger

[0..1]

windowSizeValid: PositiveInteger

+

UserDefinedTransformationComSpecProps

+

getter

0..1

transformationComSpecProps

+

0..\*

setter

+

0..1

+

requiredComSpec

0..\*

+

operation

0..1

**Figure 3.71: Modeling of the ClientComSpec on the *AUTOSAR adaptive platform***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ClientComSpec** | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Communication | | | |
| ***Note*** | Client-specific communication attributes (RPortPrototype typed by ServiceInterface) that are relevant for methods and field getters and setters. | | | |
| ***Base*** | *ARObject*, *RPortComSpec* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| clientIntent | ClientIntentEnum | 0..1 | attr | This attribute represents the expressed intent of the client. The client may decide to claim that existing resources of a ServiceInterface are expressly not used by this specific client. The conceptual background of this claim may be driven by security, safety, etc.  **Tags:**atp.Status=draft |
| endToEndCall  Response  Timeout | TimeValue | 0..1 | attr | This attribute defines the maximum time interval in which the application shall expect the servers’s response (time between the sending of the call invocation until the arrival of the server’s response). |
| getter | Field | 0..1 | ref | The existence of this reference indicates that the Client ComSpec refers to the getter of a Field.  **Tags:**atp.Status=draft |
| operation | ClientServerOperation | 0..1 | ref | This represents the corresponding ClientServerOperation. |
| setter | Field | 0..1 | ref | The existence of this reference indicates that the Client ComSpec refers to the setter of a Field.  **Tags:**atp.Status=draft |
| transformation  ComSpecProps | TransformationCom  SpecProps | \* | aggr | This references the TransformationComSpecProps which define port-specific configuration for data transformation. |

**Table 3.125: ClientComSpec**

**[TPS\_MANI\_01108]**{DRAFT} **Specification of intentions for the caller of a methods or field setter/getter** dThe attribute ClientComSpec.clientIntent can be used to specify whether the software actually intends to call the referenced methods or getter/setter of a referenced field.c*(RS\_MANI\_00034)*

|  |  |
| --- | --- |
| ***Enumeration*** | **ClientIntentEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec |
| ***Note*** | This meta-class represents the intent to specify how a given ServiceInterface is used from the perspective of a given client.  **Tags:**atp.Status=draft |
| ***Literal*** | ***Description*** |
| willCall | The client will call this method.  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |
| wontCall | The client won’t call this method.  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |

**Table 3.126: ClientIntentEnum**

**[TPS\_MANI\_01324]**{DRAFT} **Semantics of E2E attributes in ClientComSpec** dThe EndToEndTransformationComSpecProps shall be used for the specification of RPortPrototype-specific configuration options related to end-to-end protection of methods.c*(RS\_MANI\_00028)*

**3.15.5.1.4 Server ComSpec**

The ServerComSpec undergoes extensions for the AUTOSAR adaptive platform, namely the ability to refer to the getter and settermethod of a field and the definition of intentions.

**[TPS\_MANI\_01325]**{DRAFT} **Semantics of E2E attributes in ServerComSpec** dThe EndToEndTransformationComSpecProps shall be used for the specification of PPortPrototype-specific configuration options related to end-to-end protection of methods.c*(RS\_MANI\_00028)*

ServerComSpec

[0..1]

queueLength: PositiveInteger

+

*PPortComSpec*

*PortPrototype*

*AbstractProvidedPortPrototype*

*AutosarDataPrototype*

Field

+

hasGetter: Boolean

+

hasNotifier: Boolean

+

hasSetter: Boolean

*Describable*

*TransformationComSpecProps*

EndToEndTransformationComSpecProps

+

clearFromValidToInvalid: Boolean

[0..1]

disableEndToEndCheck: Boolean

[0..1]

+

+

disableEndToEndStateMachine: Boolean

[0..1]

+

maxDeltaCounter: PositiveInteger

[0..1]

+

maxErrorStateInit: PositiveInteger

[0..1]

maxErrorStateInvalid: PositiveInteger

[0..1]

+

+

maxErrorStateValid: PositiveInteger

[0..1]

+

minOkStateInit: PositiveInteger

[0..1]

minOkStateInvalid: PositiveInteger

+

[0..1]

[0..1]

minOkStateValid: PositiveInteger

+

windowSizeInit: PositiveInteger

+

[0..1]

[0..1]

windowSizeInvalid: PositiveInteger

+

[0..1]

+

windowSizeValid: PositiveInteger

UserDefinedTransformationComSpecProps

+

transformationComSpecProps

0..\*

+

providedComSpec

0..\*

getter

+

0..1

setter

+

0..1

**Figure 3.72: Modeling of the ServerComSpec on the *AUTOSAR adaptive platform***

|  |  |
| --- | --- |
| ***Class*** | **ServerComSpec** |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Communication |
| ***Note*** | Server-specific communication attributes (PPortPrototype typed by ServiceInterface) that are relevant for methods and field getters and setters. |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ServerComSpec** | |  |  |
| ***Base*** | *ARObject*, *PPortComSpec* | |  |  |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| getter | Field | 0..1 | ref | The existence of this reference indicates that the Server ComSpec refers to the getter of a Field.  **Tags:**atp.Status=draft |
| operation | ClientServerOperation | 0..1 | ref | Operation these communication attributes apply to. |
| queueLength | PositiveInteger | 0..1 | attr | Length of call queue on the server side. |
| setter | Field | 0..1 | ref | The existence of this reference indicates that the Server ComSpec refers to the setter of a Field.  **Tags:**atp.Status=draft |
| transformation  ComSpecProps | TransformationCom  SpecProps | \* | aggr | This references the TransformationComSpecProps which define port-specific configuration for data transformation. |

**Table 3.127: ServerComSpec**

**3.15.5.1.5 Communication behavior to be implemented by the Application Software**

In order to support periodic data transmission and reception (the main reason for this is to be able to perform periodic invocation of the EndToEnd transformation), the application software needs to implement the period in its program flow. But there may also be other reasons why the application shall take care of the periodicity.

The period of the invocation of transmission and reception APIs can be defined using the TransmissionComSpecProps and ReceptionComSpecProps available at the SenderComSpec and ReceiverComSpec.

As the TransmissionComSpecProps and ReceptionComSpecProps define what the expected communication behavior is, the values can also be utilized by communication (network) measurement tools to verify whether the application code actually implements the attributes properly.

The attribute ReceptionComSpecProps.dataUpdatePeriod defines the time period in which the receiving application shall call the reception API to check for new data.

The attribute ReceptionComSpecProps.timeout defines the time after which the application shall assume that the to be received data reception has timed out.

The attribute TransmissionComSpecProps.dataUpdatePeriod defines the time period in which the sending application shall call the send API.

The attributes TransmissionComSpecProps.minimumSendInterval and TransmissionComSpecProps.transmissionMode define values which influence the transmission behavior, implemented by the application code.

The attribute End2EndEventProtectionProps.dataUpdatePeriod also defines an expected period to be implemented by the application software for EndToEnd protection.

More specifically, the attribute End2EndEventProtectionProps.dataUpdatePeriod represents a network perspective, especially when no application software is defined yet and thus no SenderComSpec and ReceiverComSpec are available.

**3.15.5.2 Port Prototypes typed by Persistency Data Interfaces**

**[TPS\_MANI\_01314]**{DRAFT} **Further qualification of properties of PortPrototypes typed by PersistencyKeyValueStorageInterfaces** dFor PortPrototypes typed by PersistencyKeyValueStorageInterfaces it is possible to define further qualifying attributes for the required side.

For this purpose meta-class PersistencyDataRequiredComSpec is provided.c

*(RS\_MANI\_00027)*

PersistencyDataRequiredComSpec

*RPortComSpec*

*PortPrototype*

*AbstractRequiredPortPrototype*

*AutosarDataPrototype*

*PersistencyInterfaceElement*

PersistencyDataElement

*ValueSpecification*

+

shortLabel: Identifier

[0..1]

+

dataElement

1

+

requiredComSpec

0..\*

+

initValue

0..1

**Figure 3.73: Modeling of ComSpec for persistency**

**[TPS\_MANI\_01160]**{DRAFT} **Definition of initial value for PersistencyDataElement** dThe definition of an initial value for a PersistencyDataElement can be done on the level of a PortPrototype by means of PersistencyDataRequiredComSpec.initValuec*(RS\_MANI\_00027)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PersistencyDataRequiredComSpec** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec | | | |
| ***Note*** | This meta-class represents the ability to define port-specific attributes for supporting use cases of data persistency on the required side.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *RPortComSpec* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| dataElement | PersistencyData  Element | 1 | ref | This refrence represents the PersistencyDataElement for which the PersistencyDataRequiredComSpec applies.  **Tags:**atp.Status=draft |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PersistencyDataRequiredComSpec** | | |  |
| initValue | ValueSpecification | 0..1 | aggr | This aggregation represents the definition of an initial value for the PersistencyDataElement referenced by the enclosing PersistencyDataRequiredComSpec  **Tags:**atp.Status=draft |

**Table 3.128: PersistencyDataRequiredComSpec**

**[constr\_10081]**{DRAFT} **Existence of initial values in the definition of PersistencyDataRequiredComSpec** dFor each PersistencyDataRequiredComSpec, if the the value of attribute dataElement.updateStrategy is set to the value delete, then attribute PersistencyDataRequiredComSpec.initValue shall not

exist.c*()*

## 3.16 Executable

**[TPS\_MANI\_01010]**{DRAFT} **Root element for a hierarchical softwarecomponent** dExecutable aggregates meta-class RootSwComponentPrototype in the role rootSwComponentPrototype to provide a root element for an arbitrarily nested hierarchy of software-components represented by the reference RootSwComponentPrototype.applicationType.c*(RS\_MANI\_00001, RS\_MANI\_00004)*

Please note that the aggregation of RootSwComponentPrototype by Executable is the basis for the applicability of an instanceRef reference into the hierarchy of software-components that represent the functionality of the Executable.

This modeling approach is similar to the modeling of a System on the *AUTOSAR classic platform*.

**[TPS\_MANI\_01279]**{DRAFT} **Semantics of Executable.reportingBehavior** dAttribute Executable.reportingBehavior shall be used to control the reporting of the execution state of the enclosing Executable to the Execution Management. If the attribute does not exist, the Executable shall report its execution state to the Execution Management.c*(RS\_MANI\_00023)*

**[constr\_1605]**{DRAFT} **Standardized values of attribute Executable.category** dThe following values for attribute Executable.category are standardized by AUTOSAR:

* PLATFORM\_LEVEL: the Executable represents software on the platform level (i.e. conceptually located *on the level of* the middleware).
* APPLICATION\_LEVEL: the Executable represents software on the application level (i.e. conceptually located *above* the middleware).

### c()

|  |  |  |
| --- | --- | --- |
| «enumeration»  LoggingBehaviorEnum |  | «enumeration»  ExecutionStateReportingBehaviorEnum |
| usesLogging doesNotUseLogging | reportsExecutionState doesNotReportExecutionState |

*ARElement*

*AtpClassifier*

Executable

+

buildType: BuildTypeEnum

[0..1]

+

loggingBehavior: LoggingBehaviorEnum

[0..1]

[0..1]

+

minimumTimerGranularity: TimeValue

[0..1]

reportingBehavior: ExecutionStateReportingBehaviorEnum

+

[0..1]

version: StrongRevisionLabelString

+

*AtpPrototype*

*Identifiable*

RootSwComponentPrototype

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*SwComponentType*

«isOfType»

+

applicationType

1

{

}

redefines atpType

+

rootSwComponentPrototype

0..1

**Figure 3.74: Modeling of the Executable**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Executable** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure | | | |
| ***Note*** | This meta-class represents an executable program.  **Tags:**  atp.Status=draft  atp.recommendedPackage=Executables | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| buildType | BuildTypeEnum | 0..1 | attr | This attribute describes the buildType of a module and/or platform implementation. **Tags:**atp.Status=draft |
| loggingBehavior | LoggingBehaviorEnum | 0..1 | attr | This attribute indicates the intended logging behavior of the enclosing Executable. **Tags:**atp.Status=draft |
| minimumTimer Granularity | TimeValue | 0..1 | attr | This attribute describes the minimum timer resolution (TimeValue of one tick) that is required by the Executable.  **Tags:**atp.Status=draft |
| reporting Behavior | ExecutionState  ReportingBehavior  Enum | 0..1 | attr | this attribute controls the execution state reporting behavior of the enclosing Executable.  **Tags:**atp.Status=draft |
| rootSw Component  Prototype | RootSwComponent  Prototype | 0..1 | aggr | This represents the root SwCompositionPrototype of the Executable. This aggregation is required (in contrast to a direct reference of a SwComponentType) in order to support the definition of instanceRefs in Executable context.  **Tags:**atp.Status=draft |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Executable** |  |  |  |
| version | StrongRevisionLabel  String | 0..1 | attr | Version of the executable.  **Tags:**atp.Status=draft |

**Table 3.129: Executable**

|  |  |
| --- | --- |
| ***Enumeration*** | **BuildTypeEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation |
| ***Note*** | This enumeration defines the possible buildTypes a software module may be implemented.  **Tags:**atp.Status=draft |
| ***Literal*** | ***Description*** |
| buildTypeDebug | Used for debugging.  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |
| buildTypeRelease | Used for releasing.  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |

**Table 3.130: BuildTypeEnum**

|  |  |
| --- | --- |
| ***Enumeration*** | **LoggingBehaviorEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure |
| ***Note*** | This enumeration provides options for controlling of whether an Executable uses logging.  **Tags:**atp.Status=draft |
| ***Literal*** | ***Description*** |
| doesNotUse Logging | The Executable indicates its intention to not use logging.  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |
| usesLogging | The Executable indicates its intention to use logging  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |

**Table 3.131: LoggingBehaviorEnum**

|  |  |
| --- | --- |
| ***Enumeration*** | **ExecutionStateReportingBehaviorEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure |
| ***Note*** | This enumeration provides options for controlling of how an Executable reports its execution state to the Execution Management  **Tags:**atp.Status=draft |
| ***Literal*** | ***Description*** |
| doesNotReport ExecutionState | The Executable shall not report its execution state to the Execution Management.  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |

5

4

|  |  |
| --- | --- |
| ***Enumeration*** | **ExecutionStateReportingBehaviorEnum** |
| reportsExecution State | The Executable shall report its execution state to the Execution Management.  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |

**Table 3.132: ExecutionStateReportingBehaviorEnum**

**[TPS\_MANI\_01271]**{DRAFT} **Semantics of Executable.loggingBehavior** dAttribute Executable.loggingBehavior shall be used to indicate whether the enclosing Executable uses logging.

If the attribute does not exist, the Executable indicates that it does not use logging.c *(RS\_MANI\_00023, RS\_MANI\_00037)*

**[TPS\_MANI\_03056]**{DRAFT} **Optionality of Executable.rootSwComponentPrototype** dThe aggregation Executable.rootSwComponentPrototype has been made optional in order to support the implementation of *platform modules* that do not utilize any service oriented communication and don’t require any further formalization.c *(RS\_MANI\_00023)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **RootSwComponentPrototype** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure | | | |
| ***Note*** | The RootSwCompositionPrototype represents the top-level-composition of software components within an Executable.  The contained SwComponentPrototypes are fully specified by their SwComponentTypes (including Port Prototypes, PortInterfaces, VariableDataPrototypes, etc.).  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| applicationType | SwComponentType | 1 | tref | This SwComponentType acts as the Type of the RootSw ComponentPrototype.  **Stereotypes:** isOfType **Tags:**atp.Status=draft |

**Table 3.133: RootSwComponentPrototype**

**[constr\_1492]**{DRAFT} **SwComponentType referenced in the role Executable. rootSwComponentPrototype.applicationType**dAny SwComponentType referenced in the role Executable.rootSwComponentPrototype.applicationType, or used to type a SwComponentPrototype nested inside the SwComponentType referenced in the role Executable.rootSwComponentPrototype.applicationType shall **only** be either a CompositionSwComponentType or an AdaptiveApplicationSwComponentType.c*()*

The example depicted in Figure 3.75 exemplifies the statement of [constr\_1492]. The example shows a component hierarchy that consists of SwComponentPrototypes that are excursively typed by either a CompositionSwComponentType or an AdaptiveApplicationSwComponentType.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***SwComponentType*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Components | | | |
| ***Note*** | Base class for AUTOSAR software components. | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Subclasses*** | AdaptiveApplicationSwComponentType, *AtomicSwComponentType*, CompositionSwComponentType, ParameterSwComponentType | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| port | PortPrototype | \* | aggr | The PortPrototypes through which this SwComponent Type can communicate.  The aggregation of PortPrototype is subject to variability with the purpose to support the conditional existence of PortPrototypes.  **Stereotypes:** atpSplitable; atpVariation **Tags:**  atp.Splitkey=port.shortName, port.variationPoint.short Label  vh.latestBindingTime=preCompileTime |
| portGroup | PortGroup | \* | aggr | A port group being part of this component.  **Stereotypes:** atpVariation  **Tags:**vh.latestBindingTime=preCompileTime |
| swComponent  Documentation | SwComponent  Documentation | 0..1 | aggr | This adds a documentation to the SwComponentType.  **Stereotypes:** atpSplitable; atpVariation **Tags:**  atp.Splitkey=swComponentDocumentation, sw ComponentDocumentation.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=-10 |

**Table 3.134: SwComponentType**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CompositionSwComponentType** | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Composition | | | |
| ***Note*** | A CompositionSwComponentType aggregates SwComponentPrototypes (that in turn are typed by Sw ComponentTypes) as well as SwConnectors for primarily connecting SwComponentPrototypes among each others and towards the surface of the CompositionSwComponentType. By this means, hierarchical structures of software-components can be created.  **Tags:**atp.recommendedPackage=SwComponentTypes | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| component | SwComponent  Prototype | \* | aggr | The instantiated components that are part of this composition.  **Stereotypes:** atpSplitable; atpVariation **Tags:**  atp.Splitkey=component.shortName, component.variation Point.shortLabel  vh.latestBindingTime=postBuild |
| connector | SwConnector | \* | aggr | SwConnectors have the principal ability to establish a connection among PortPrototypes. They can have many roles in the context of a CompositionSwComponentType. Details are refined by subclasses.  5 |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CompositionSwComponentType** | |  |  |
|  |  |  |  | 4  The aggregation of SwConnectors is subject to variability with the purpose to support variant data flow.  The aggregation is marked as atpSplitable in order to allow the extension of the ECU extract with AssemblySw Connectors between ApplicationSwComponentTypes and ServiceSwComponentTypes during the ECU integration.  **Stereotypes:** atpSplitable; atpVariation **Tags:**  atp.Splitkey=connector.shortName, connector.variation Point.shortLabel  vh.latestBindingTime=postBuild |
| constantValue Mapping | ConstantSpecification  MappingSet | \* | ref | Reference to the ConstantSpecificationMapping to be applied for initValues of PPortComSpecs and RPortCom Spec.  **Stereotypes:** atpSplitable  **Tags:**atp.Splitkey=constantValueMapping |
| dataType Mapping | DataTypeMappingSet | \* | ref | Reference to the DataTypeMapping to be applied for the used ApplicationDataTypes in ServiceInterfaces.  **Stereotypes:** atpSplitable  **Tags:**atp.Splitkey=dataTypeMapping |

**Table 3.135: CompositionSwComponentType**

While the left part of Figure 3.75 resembles the modeling in the meta-model, the right part uses a simplified notation to give an idea how the nested definition of softwarecomponents could look like.

An obvious consequence of [constr\_1492] is that no software-component that could be used on the *AUTOSAR classic platform* is allowed on the *AUTOSAR adaptive platform*, i.e. in the context of an Executable.rootSwComponentPrototype.applicationType.

Software-components on the *AUTOSAR adaptive platform* are mainly defined by their interaction with the outside world by means of PortPrototypes typed by ServiceInterfaces. The definition of an internal behavior, with a minor exception, is not foreseen.

This lack of internal structure, in combination with decisions made regarding the scope of the generation of header files, leads to a situation where the implementation of a software component in source code is (in comparison to the situation on the *AUTOSAR classic platform*) way less subject to a strict separation.

In other words, there is no real motivation to implement software-components separately from each other. It would be possible, although not encouraged, to implement all software-components of a given executable program directly within the Main() function of the program.

Executable

CompositionSwComponentType

RootSwComponent

Prototype

rootSwComponentPrototype

ComponentPrototype

1

(

CompositionSwComponentType

)

ComponentPrototype

2

(

CompositionSwComponentType

)

ComponentPrototype

3

(

AdaptiveApplicationSwComponentType

)

ComponentPrototype

4

(

AdaptiveApplicationSwComponentType

)

ComponentPrototype

5

(

AdaptiveApplicationSwComponentType

)

applicationType

P

R

R

R

R

R

P

**Figure 3.75: Example of the possible structure of an Executable**

## 3.17 Optional Members in complex Data Structures

**3.17.1 Background**

The *AUTOSAR adaptive platform* supports the usage of a TLV[[7]](#footnote-7) data encoding on the SOME/IP transport layer. TLV is typically used where at least a part of the transmitted data is only *optionally* existing and filled with meaningful values.

In other words: an optional part of a data structure may exist and carry meaningful values in one instance of data transmission and be completely missing in another instance of the data transmission.

The receiving software needs to be able to identify whether the optional part exists and read its value accordingly.

The receiving software also needs to be able to still execute meaningfully if the optional part of such a data structure does not exist in the specific communication instance.

Consequently, it is necessary to be able to precisely identify the parts of a data structure that may become optional for specific instances of data transmission.

In terms of the AUTOSAR meta-model, the identification could - in principle - be attached at various levels of abstraction:

**AutosarDataType** In this case the optionality that is primarily only needed for communication purposes would still be existing in all other usages of data types. AUTOSAR still sees use cases for implementing this option, especially in the context of the *AUTOSAR classic platform*.

Admittedly, the definition of different optionality configurations for the same data type may lead to the existence of a bunch of structurally identical data types that only vary in terms of optionality. The existence of variation points may help to mitigate this effect, though.

**ServiceInterface** In this case the optionality is defined where it is actually required. However, different optionality could - in principle - be defined for DataPrototypes typed by the same AutosarDataType.

This would lead to an increased effort for the definition of C++ data types in the context of the same ServiceInterface. Additional constraints have been identified in the context of the *AUTOSAR classic platform* that finally render this option as not viable.

**ComSpec** In this case the definition of optionality would even be more specific in comparison to the definition of optionality on the level of ServiceInterfaces.

On top of that, the task to define optionality in the vast majority of cases is done by an OEM, whereas the model definition on the level of ComSpec requires the existence of SwComponentTypes and this definition is in many cases in the domain of a supplier.

As a result of this consideration, AUTOSAR has opted for implementation the concept of defining the optionality on the level of the AutosarDataType.

**3.17.2 Definition of Optionality**

As mentioned before, the concrete definition of optionality on the level of an Autosar-

DataType is done by the indication of individual elements of the composite AutosarDataType.

More specifically, the definition of optionality needs to be supported for subclasses of

AutosarDataType, namely on the level of ApplicationDataType as well as on the level of CppImplementationDataType.

In other words, if ApplicationDataTypes with optional elements are used to define a ServiceInterface then it is still necessary to convey the optionality down to the level of data type definition that directly affects the language binding of the AUTOSAR model.

Figure 3.76 shows the modeling of optionality on the level of ApplicationDataType.

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | «enumeration»  ArraySizeSemanticsEnum | | fixedSize variableSize | |  |
| |  | | --- | | «enumeration»  ArraySizeHandlingEnum | | allIndicesSameArraySize allIndicesDifferentArraySize inheritedFromArrayElementTypeSize | |

**Figure 3.76: Modeling of optionality on the level of ApplicationDataType**

**[TPS\_MANI\_01184]**{DRAFT} **Definition of optional elements on the level of ApplicationDataType** dThe modeling approach for the definition of optional elements on the level of ApplicationDataType is to set the attribute ApplicationRecordElement.isOptional to the value True.

If the attribute is not set or set to the value False then the respective Application-

RecordElement **shall be considered mandatory**.c*(RS\_MANI\_00030)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ApplicationRecordDataType** | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes | | | |
| ***Note*** | An application data type which can be decomposed into prototypes of other application data types.  **Tags:**atp.recommendedPackage=ApplicationDataTypes | | | |
| ***Base*** | *ARElement*, *ARObject*, *ApplicationCompositeDataType*, *ApplicationDataType*, *AtpBlueprint*, *Atp*  *Blueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| element  (ordered) | ApplicationRecord  Element | \* | aggr | Specifies an element of a record.  The aggregation of ApplicationRecordElement is subject to variability with the purpose to support the conditional existence of elements inside a ApplicationrecordData Type.  **Stereotypes:** atpVariation  **Tags:**vh.latestBindingTime=preCompileTime |

**Table 3.136: ApplicationRecordDataType**

|  |  |
| --- | --- |
| ***Class*** | **ApplicationRecordElement** |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes |
| ***Note*** | Describes the properties of one particular element of an application record data type. |
| ***Base*** | *ARObject*, *ApplicationCompositeElementDataPrototype*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable* |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ApplicationRecordElement** | |  |  |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| isOptional | Boolean | 0..1 | attr | This attribute represents the ability to declare the enclosing ApplicationRecordElement as optional. This means the that, at runtime, the ApplicationRecord Element may or may not have a valid value and shall therefore be ignored.  The underlying runtime software provides means to set the ApplicationRecordElement as not valid at the sending end of a communication and determine its validity at the receiving end. |

**Table 3.137: ApplicationRecordElement**

On top of that, it is still possible to use CppImplementationDataType directly for the definition of a ServiceInterface.

**[TPS\_MANI\_01185]**{DRAFT} **Definition of optional elements on the level of CppImplementationDataType** dThe modeling approach for the definition of optional elements on the level of CppImplementationDataType is to set the attribute CppImplementationDataTypeElement.isOptional to the value True.

If the attribute is not set or set to the value False then the respective CppImplementationDataTypeElement **shall be considered mandatory**.c*(RS\_MANI\_00030)*

The attribute NotAvailableValueSpecification.defaultPattern has no meaning for the initialization of DataPrototypes on the *AUTOSAR adaptive platform*. This aspect is covered by [TPS\_MANI\_01333]:

**[TPS\_MANI\_01333]**{DRAFT} **Attribute NotAvailableValueSpecification. defaultPattern is not applicable** dThe attribute NotAvailableValueSpecification.defaultPattern (if defined) shall be ignored by the adaptive platform.

The rationale for ignoring the defaultPattern is that the optional data is technically not accessible from application code in case it has not been received.c*(RS\_MANI\_00030)*

## 3.18 Serialization Properties

In Adaptive AUTOSAR, the serialization code is generated out of the service description and is compiled and executed in the application context.

The meta-class TransformationPropsToServiceInterfaceElementMapping defines the serialization for a ServiceInterface element and provides the necessary serialization settings with the TransformationProps element.

The existence of a TransformationPropsToServiceInterfaceElementMapping demands the existence of serialization code that is linked with the application component object file to an application binary.

The serialization of SOME/IP is based on the ServiceInterface specification. If an AutosarDataPrototype that is used within a ServiceInterface is composite like a structure, union or array then SOME/IP supports the configuration of length fields that will be put in front of the serialized data.

AUTOSAR supports the configuration of such serialization settings on two different levels:

* Modeling on ServiceInterface element level that is valid for all available occurrences of a DataPrototype in the ServiceInterface element. This case is described in detail in chapter 3.18.1.
* Fine granular modeling on the level of DataPrototypes described in this chapter. This case is described in detail in chapter 3.18.2.

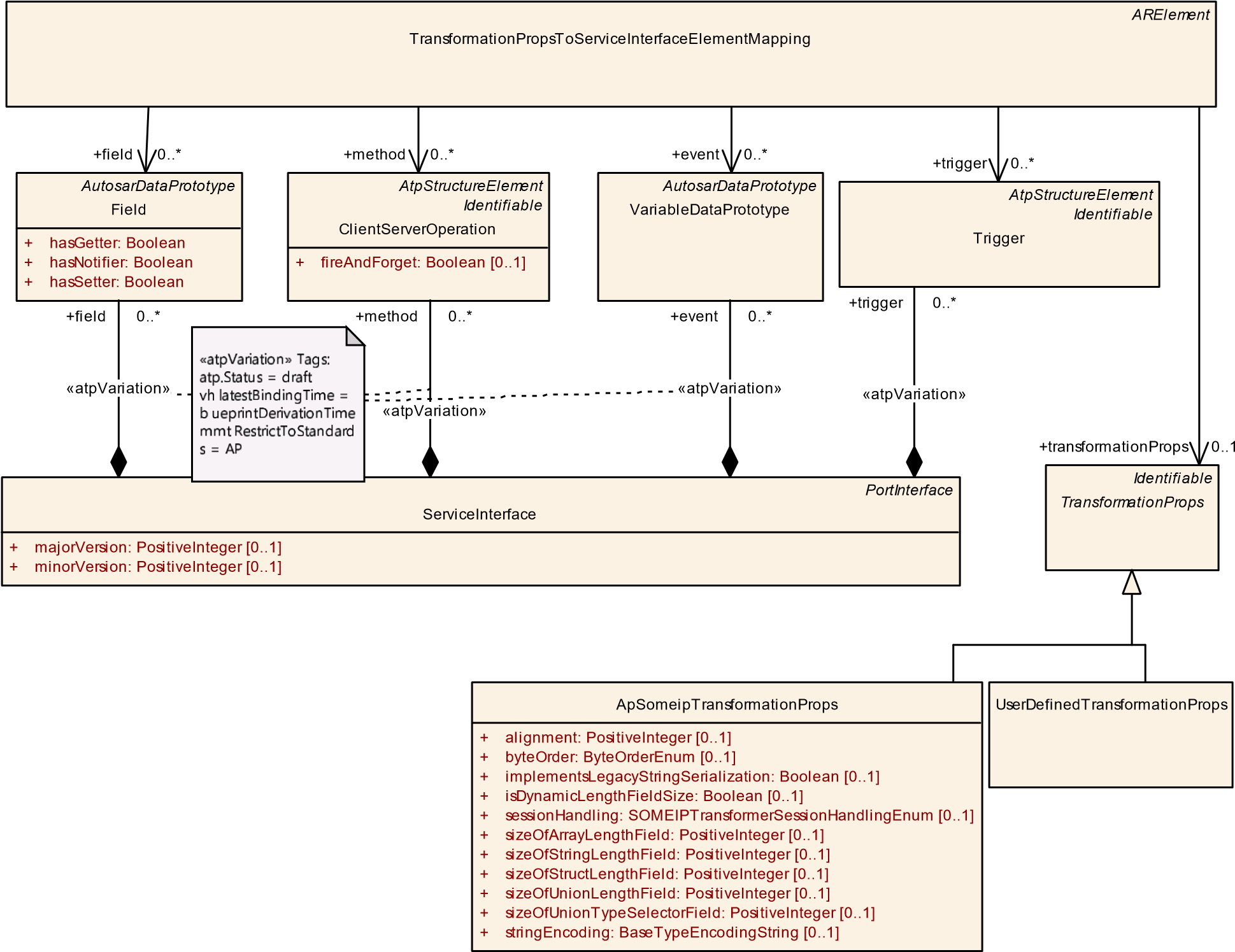
**3.18.1 Default Values for Serialization Properties**

**[TPS\_MANI\_03101]**{DRAFT} **SOME/IP serialization** dThe ApSomeipTransformationProps meta-class that is referenced by the TransformationPropsToServiceInterfaceElementMapping in the role transformationProps provides the ability to define a SOME/IP serialization settings for ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, trigger, method or field.c*(RS\_MANI\_00008, RS\_MANI\_00025)*

**[constr\_3395]**{DRAFT} **TransformationPropsToServiceInterfaceElementMapping is restricted to one single ServiceInterface** dAll ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, trigger, method or field shall be aggregated by the same ServiceInterface in the role event, trigger, method or field.c*()*

**[TPS\_MANI\_03288]**{DRAFT} **ApSomeipTransformationProps for triggers** dIf ApSomeipTransformationProps are assigned to a trigger then only the attribute sessionHandling is relevant for the SOME/IP transformation. All other attributes of the ApSomeipTransformationProps are irrelevant for triggers and will be ignored by the SOME/IP transformation.c*(RS\_MANI\_00008, RS\_MANI\_00025)*

**[TPS\_MANI\_03103]**{DRAFT} **Default size for all array and map length fields** dThe attribute sizeOfArrayLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available variable size arrays (vectors), fixed size arrays and associative\_maps defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field.c*(RS\_MANI\_00008, RS\_MANI\_00025)*



**Figure 3.77: Association of serialization properties with a ServiceInterface**

**[TPS\_MANI\_03104]**{DRAFT} **Default size for all structure length fields** dThe attribute sizeOfStructLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available structures defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field.c*(RS\_MANI\_00008, RS\_MANI\_00025)*

**[TPS\_MANI\_03117]**{DRAFT} **Default size for all string length fields** dThe attribute sizeOfStringLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available strings defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field.c*(RS\_MANI\_00008, RS\_MANI\_00025)*

**[TPS\_MANI\_03105]**{DRAFT} **Default size for all union length fields** dThe attribute sizeOfUnionLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available unions defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field.c*(RS\_MANI\_00008, RS\_MANI\_00025)*

**[TPS\_MANI\_03106]**{DRAFT} **Default size for all union type selector fields** dThe attribute sizeOfUnionTypeSelectorField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a type field generated by SOME/IP in front of all available unions defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field.c*(RS\_MANI\_00008, RS\_MANI\_00025)*

**[TPS\_MANI\_03107]**{DRAFT} **Default alignment for all dynamic DataPrototypes** dThe attribute alignment of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the padding for alignment purposes that will be added by SOME/IP after the serialized data of all variable data length data elements defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field.c*(RS\_MANI\_00008, RS\_MANI\_00025)*

**[TPS\_MANI\_03108]**{DRAFT} **Default Byte Order for all DataPrototypes** dThe attribute byteOrder of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the Byte Order in the serialized data stream resulting from ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field.c *(RS\_MANI\_00008, RS\_MANI\_00025)*

**[constr\_1614]**{DRAFT} **Existence of attribute TransformationPropsToServiceInterfaceElementMapping.transformationProps.sessionHandling** dThe attribute ApSomeipTransformationProps.sessionHandling shall only exist if the TransformationPropsToServiceInterfaceElementMapping that refers to the respective ApSomeipTransformationProps in the role transformationProps does **not** refer to a ClientServerOperation in the role method.c

### ()

**[TPS\_MANI\_01210]**{DRAFT} **Default encoding for all DataPrototypes typed by CppImplementationDataType of category STRING** dThe attribute stringEncoding of a ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the string encoding in the serialized data stream resulting from ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field.c *(RS\_MANI\_00008, RS\_MANI\_00025)*

**[constr\_1675]**{DRAFT} **Existence of attribute ApSomeipTransformationProps.stringEncoding** dThe attribute TransformationPropsToServiceInterfaceElementMapping.transformationProps.stringEncoding shall only exist for a event, method or field (referenced by the same TransformationPropsToServiceInterfaceElementMapping) that consists of or contains a DataPrototype typed by a CppImplementationDataType of categorySTRING.c*()*

Please note that more details about ApSomeipTransformationProps can be found in chapter 3.18.2.

**[constr\_1678]**{DRAFT} **Allowed values for attribute ApSomeipTransformationProps.stringEncoding** dImposed by technical restrictions in the definition of the SOME/IP message format [7], only two possible values of attribute ApSomeipTransformationProps.stringEncoding are allowed:

* UTF-8: UCS Transformation Format 8
* UTF-16: Character encoding for Unicode *code points* based on 16 bit *code units* [16]

### c()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ApSomeipTransformationProps** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::SerializationProperties | | | |
| ***Note*** | SOME/IP serialization properties.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TransformationProps* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| alignment | PositiveInteger | 0..1 | attr | Defines the padding for alignment purposes that will be added by the SOME/IP transformer after the serialized data of the variable data length data element. The alignment shall be specified in Bits.  **Tags:**atp.Status=draft |
| byteOrder | ByteOrderEnum | 0..1 | attr | Specifies the byte order of data in the serialized data stream.  **Tags:**atp.Status=draft |
| implements LegacyString  Serialization | Boolean | 0..1 | attr | This attribute indicates that Strings in the SOME/IP message shall NOT be serialized according to the SOME/ IP specification for Strings.  If this attribute is set to true, BOM and null-termination shall NOT be added in the serialization for Strings in the payload.  If this attribute is set to false (or not set) BOM and null-termination shall be added in the serialization for Strings in the payload according to the SOME/IP specification for Strings.  NOTE! This attribute is not future safe, and will be removed in an upcoming AUTOSAR release!  **Tags:**atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ApSomeipTransformationProps** | | | |
| isDynamic  LengthFieldSize | Boolean | 0..1 | attr | This attribute represents the ability to control the setting of the wire type for TLV encoding.  If the attribute is set to True then wire type 5-7 shall be used.  If the attribute does not exist or is set to False then wire type 4 shall be used.  **Tags:**atp.Status=draft |
| session  Handling | SOMEIPTransformer  SessionHandlingEnum | 0..1 | attr | Defines whether the SOME/IP transformer shall use session handling for Sender/Receiver communication.  **Tags:**atp.Status=draft |
| sizeOfArray  LengthField | PositiveInteger | 0..1 | attr | Configures the SOME/IP serialization for the referenced dataPrototype in case of a variable size Array (Vector), fixed-size Array or an Associative\_Map. It describes the size of the length field (in Bytes) that will be put in front of the Array or Associative\_Map in the SOME/IP message.  **Tags:**atp.Status=draft |
| sizeOfString LengthField | PositiveInteger | 0..1 | attr | Configures the SOME/IP serialization for the referenced dataPrototype in case of a String. It describes the size of the length field (in Bytes) that will be put in front of the String in the SOME/IP message.  **Tags:**atp.Status=draft |
| sizeOfStruct LengthField | PositiveInteger | 0..1 | attr | Configures the SOME/IP serialization for the referenced dataPrototype in case of an Struct. It describes the size of the length field (in Bytes) that will be put in front of the Struct in the SOME/IP message.  **Tags:**atp.Status=draft |
| sizeOfUnion LengthField | PositiveInteger | 0..1 | attr | Configures the SOME/IP serialization for the referenced dataPrototype in case of a Union. It describes the size of the length field (in Bytes) that will be put in front of the Union in the SOME/IP message.  **Tags:**atp.Status=draft |
| sizeOfUnion TypeSelector  Field | PositiveInteger | 0..1 | attr | Configures the SOME/IP serialization for the referenced dataPrototype in case of a Union. It describes the size of the type selector field (in Bytes) that will be put in front of the Union in the SOME/IP message.  **Tags:**atp.Status=draft |
| stringEncoding | BaseTypeEncoding  String | 0..1 | attr | Configures the encoding for SOME/IP serialization for the referenced dataPrototype in case of an String.  **Tags:**atp.Status=draft |

**Table 3.138: ApSomeipTransformationProps**

**[TPS\_MANI\_03102]**{DRAFT} **UserDefined serialization** dThe UserDefined-

TransformationProps meta-class that is referenced by the Transformation-

PropsToServiceInterfaceElementMapping in the role transformationProps provides the ability to define a User defined serialization for ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, method or field.c*(RS\_MANI\_00014, RS\_MANI\_00025)*

Please note that UserDefinedTransformationProps is derived from meta-class

Identifiable and therefore has the ability to describe special data (sdg) by which it is possible to define custom structural extensions of an AUTOSAR model in a generic way. For more information about special data please refer to [6].

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **TransformationPropsToServiceInterfaceElementMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure | | | |
| ***Note*** | This meta-class represents the ability to associate a ServiceInterface element with TransformationProps. The referenced elements of the Service Interface will be serialized according to the settings defined in the TransformationProps.  **Tags:** atp.Status=draft  atp.recommendedPackage=TransformationPropsToServiceInterfaceElementMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| event | VariableDataPrototype | \* | ref | This represents the reference to one or several events of one ServiceInterface.  **Tags:**atp.Status=draft |
| field | Field | \* | ref | This represents the reference to one or several fields of one ServiceInterface.  **Tags:**atp.Status=draft |
| method | ClientServerOperation | \* | ref | This represents the reference to one or several methods of one ServiceInterface. **Tags:**atp.Status=draft |
| tlvDataId Definition | TlvDataIdDefinitionSet | \* | ref | This reference identifies the TlvDataIdDefinitions relevant for the enclosing TransformationPropsToServiceInterface Mapping.  **Tags:**atp.Status=draft |
| transformation Props | TransformationProps | 0..1 | ref | This represents the reference to the applicable Serialization properties.  **Tags:**atp.Status=draft |
| trigger | Trigger | \* | ref | This represents the reference to one or several triggers of one ServiceInterface.  **Tags:**atp.Status=draft |

**Table 3.139: TransformationPropsToServiceInterfaceElementMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **UserDefinedTransformationProps** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Transformer | | | |
| ***Note*** | The class UserDefinedTransformationProps specifies specific configuration properties of a user defined serializer. | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TransformationProps* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.140: UserDefinedTransformationProps**

**3.18.2 Individual Definition of Serialization Properties**

**[TPS\_MANI\_03109]**{DRAFT} **TransformationProps on the level of DataPrototypes overwrites TransformationProps settings on the level of a ServiceInterface** dThe fine granular modeling of TransformationProps on the level of DataPrototypes overwrites the TransformationProps settings defined on the level of a ServiceInterface described with the TransformationPropsToServiceInterfaceElementMapping.c*(RS\_MANI\_00025)*

**[constr\_3361]**{DRAFT} **Selective definition of serialization settings** dIf a SomeipDataPrototypeTransformationProps is defined for a composite DataPrototype of an element of a ServiceInterface (method, field, event) and if the reference someipTransformationProps exists then SomeipDataPrototypeTransformationProps that define the reference someipTransformationProps shall be defined for all other composite DataPrototypes of the ServiceInterface element as well.c*()*

«enumeration»

SOMEIPTransformerSessionHandlingEnum

sessionHandlingActive

sessionHandlingInactive

ApSomeipTransformationProps

[0..1]

alignment: PositiveInteger

+

+

byteOrder: ByteOrderEnum

[0..1]

+

implementsLegacyStringSerialization: Boolean

[0..1]

[0..1]

isDynamicLengthFieldSize: Boolean

+

[0..1]

sessionHandling: SOMEIPTransformerSessionHandlingEnum

+

sizeOfArrayLengthField: PositiveInteger

+

[0..1]

[0..1]

sizeOfStringLengthField: PositiveInteger

+

sizeOfStructLengthField: PositiveInteger

+

[0..1]

sizeOfUnionLengthField: PositiveInteger

+

[0..1]

+

sizeOfUnionTypeSelectorField: PositiveInteger

[0..1]

+

stringEncoding: BaseTypeEncodingString

[0..1]

*ARElement*

SomeipDataPrototypeTransformationProps

*ARElement*

TransformationPropsSet

*Identifiable*

*TransformationProps*

«atpVariation»

SwDataDefProps

UserDefinedTransformationProps

DataPrototypeInServiceInterfaceRef

transformationProps

+

0..\*

+

dataPrototype

0..\*

+

someipTransformationProps

0..1

+

networkRepresentation

0..1

**Figure 3.78: Overview about SOME/IP Serialization Properties**

|  |  |
| --- | --- |
| ***Class*** | **TransformationPropsSet** |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Transformer |
| ***Note*** | Collection of TransformationProps.  **Tags:**atp.recommendedPackage=TransformationPropsSets |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **TransformationPropsSet** | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| transformation Props | TransformationProps | \* | aggr | Transformer specific configuration properties. |

**Table 3.141: TransformationPropsSet**

|  |  |
| --- | --- |
| ***Enumeration*** | **SOMEIPTransformerSessionHandlingEnum** |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Transformer |
| ***Note*** | Enables or disable session handling for SOME/IP transformer |
| ***Literal*** | ***Description*** |
| sessionHandling Active | The SOME/IP Transformer shall use session handling  **Tags:**atp.EnumerationLiteralIndex=0 |
| sessionHandling Inactive | The SOME/IP Transformer doesn’t use session handling  **Tags:**atp.EnumerationLiteralIndex=1 |

**Table 3.142: SOMEIPTransformerSessionHandlingEnum**

**[TPS\_MANI\_03070]**{DRAFT} **Size of a length field for a chosen array or map** dThe attribute sizeOfArrayLengthField of ApSomeipTransformationProps defines the size of a length field generated by SOME/IP in front of a variable size array (vector), fixed size array or associative\_map for which the SomeipDataPrototypeTransformationProps is defined, i.e. the variable size array (vector), fixed size array or associative\_map that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.c*(RS\_MANI\_00008, RS\_MANI\_00024)*

**[constr\_3353]**{DRAFT} **Restriction in usage of ApSomeipTransformationProps.sizeOfArrayLengthField** dThe value of the attribute sizeOfArrayLengthField shall be either 0, 1, 2 or 4.c*()*

**[constr\_3447]**{DRAFT} **ApSomeipTransformationProps.sizeOfArrayLengthField that equals 0** dThe sizeOfArrayLengthField value of 0 is only allowed to be used if a fixed size array for which the SomeipDataPrototypeTransformationProps is defined is referenced within the aggregated DataPrototypeInServiceInterfaceRef.c*()*

The setting of sizeOfArrayLengthField for fixed size arrays supports a backward compatible extension of such arrays with additional array elements.

**[TPS\_MANI\_03071]**{DRAFT} **Size of a length field for a chosen structure** dThe attribute sizeOfStructLengthField of ApSomeipTransformationProps defines the size of a length field generated by SOME/IP in front of a structure for which the

SomeipDataPrototypeTransformationProps is defined, i.e. the structure that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.c *(RS\_MANI\_00008, RS\_MANI\_00024)*

**[constr\_3354]**{DRAFT} **Restriction in usage of ApSomeipTransformationProps.sizeOfStructLengthField** dThe value of the attribute sizeOfStructLengthField shall be either 0, 1, 2 or 4.c*()*

**[TPS\_MANI\_03116]**{DRAFT} **Size of a length field for a chosen string** dThe attribute sizeOfStringLengthField of ApSomeipTransformationProps defines the size of a length field generated by SOME/IP in front of a String for which the SomeipDataPrototypeTransformationProps is defined, i.e. the String that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.c *(RS\_MANI\_00008, RS\_MANI\_00024)*

**[constr\_3372]**{DRAFT} **Restriction in usage of ApSomeipTransformationProps.sizeOfStringLengthField** dThe value of the attribute sizeOfStringLengthField shall be either 0, 1, 2 or 4.c*()*

**[TPS\_MANI\_03217]**{DRAFT} **On-the-wire encoding for a chosen string** dThe attribute stringEncoding of ApSomeipTransformationProps defines the on-thewire encoding of a String for which the SomeipDataPrototypeTransformationProps is defined, i.e. the String that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.c*(RS\_MANI\_00008, RS\_MANI\_00024)*

**[TPS\_MANI\_03072]**{DRAFT} **Size of a length field for a chosen union** dThe attribute sizeOfUnionLengthField of ApSomeipTransformationProps defines the size of a length field generated by SOME/IP in front of a union for which the

SomeipDataPrototypeTransformationProps is defined, i.e. the union that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.c*(RS\_MANI\_00008, RS\_MANI\_00024)*

**[constr\_3355]**{DRAFT} **Restriction in usage of ApSomeipTransformation-**

**Props.sizeOfUnionLengthField** dThe value of the attribute sizeOfUnionLengthField shall be either 0, 1, 2 or 4.c*()*

**[TPS\_MANI\_03073]**{DRAFT} **Alignment of a dynamic DataPrototype** dThe attribute alignment of ApSomeipTransformationProps defines the padding for alignment purposes that will be added by SOME/IP after the serialized data of the variable data length data element for which the SomeipDataPrototypeTransformationProps is defined, i.e. the variable data length DataPrototype that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.c*(RS\_MANI\_00008, RS\_MANI\_00024)*

**[constr\_3356]**{DRAFT} **Restriction in usage of ApSomeipTransformationProps.alignment** dThe value of the attribute alignment shall be either 8, 16, 32, 64, 128, or 256.c*()*

**[TPS\_MANI\_03074]**{DRAFT} **Size of a type selector field for a chosen union** dThe attribute sizeOfUnionTypeSelectorField of ApSomeipTransformationProps defines the size of a type selector field generated by SOME/IP in front of a union for which the SomeipDataPrototypeTransformationProps is defined, i.e. the union that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.c*(RS\_MANI\_00008, RS\_MANI\_00024)*

**[constr\_3357]**{DRAFT} **Restriction in usage of ApSomeipTransformationProps.sizeOfUnionTypeSelectorField** dThe value of the attribute sizeOfUnionTypeSelectorField shall be either 1, 2 or 4.c*()*

**[TPS\_MANI\_03235]**{DRAFT} **Usage of ApSomeipTransformationProps.sessionHandling** dThe sessionHandling attribute defined in an ApSomeipTransformationProps that is referenced by SomeipDataPrototypeTransformationProps is not relevant for the DataPrototypes that are referenced in the SomeipDataPrototypeTransformationProps.c*(RS\_MANI\_00008, RS\_MANI\_00024)*

**[TPS\_MANI\_03278]**{DRAFT} **Usage of ApSomeipTransformationProps.byteOrder** dThe byteOrder attribute defined in an ApSomeipTransformationProps that is referenced by SomeipDataPrototypeTransformationProps shall be ignored for the dataPrototypes for which the SomeipDataPrototypeTransformationProps apply.c*(RS\_MANI\_00008, RS\_MANI\_00024)*

The byteOrder attribute defines the byte order of the complete payload in the SOME/IP message and therefore the configuration via TransformationPropsToServiceInterfaceElementMapping is the only valid option to define the byte order for a ServiceInterface element. Please note that according to SOME/IP, the header is encoded in network byte order (Big Endian).

The sessionHandling attribute is used for the activation/deactivation of the SessionHandling for Events/Notifiers and therefore the usage via TransformationPropsToServiceInterfaceElementMapping is the only valid configuration option.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SomeipDataPrototypeTransformationProps** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::SerializationProperties | | | |
| ***Note*** | This meta-class represents the ability to define data transformation props specifically for a SOME/IP serialization for a given DataPrototype.  **Tags:**  atp.Status=draft  atp.recommendedPackage=SomeipDataPrototypeTransformationPropss | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| dataPrototype | DataPrototypeInService  InterfaceRef | \* | aggr | Collection of DataPrototypes for which the settings in SomeipDataPrototypeTransformationProps are valid. For reuse reasons the SomeipDataPrototypeTransformation Props is able to aggregate several DataPrototypes.  **Tags:**atp.Status=draft |
| network  Representation | SwDataDefProps | 0..1 | aggr | Optional specification of the actual network representation for the referenced primitive DataPrototype. If a network representation is provided then the baseType available in the SwDataDefProps shall be used as input for the serialization/deserialization. If the network  Representation is not provided then the baseType of the AbstractImplementationDataType shall be used for the serialization/deserialization.  **Tags:**atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SomeipDataPrototypeTransformationProps** | | | |
| someip Transformation  Props | ApSomeip  TransformationProps | 0..1 | ref | This reference represents the ability to define data transformation props specifically for a SOME/IP serialization.  **Tags:**atp.Status=draft |

**Table 3.143: SomeipDataPrototypeTransformationProps**

The modeling of the reference to a DataPrototype in the context of a PortInterface that is typed by an ApplicationDataType or by a CppImplementationDataType is depicted in Figure 10.17.

**[TPS\_MANI\_01136]**{DRAFT} **AutosarDataPrototype is the target of the DataPrototypeInServiceInterfaceRef** dIf the target of an DataPrototypeInServiceInterfaceRef is an AutosarDataPrototype the role DataPrototypeInServiceInterfaceRef.dataPrototype shall be used to describe the reference **independently** of whether the AutosarDataPrototype is typed by an ApplicationDataType or a CppImplementationDataType and even **independently** of whether the AutosarDataType of the AutosarDataPrototype represents a composite data type.c*(RS\_MANI\_00008, RS\_MANI\_00024)*

**[TPS\_MANI\_01137]**{DRAFT} **Applicable use cases for DataPrototypeInServiceInterfaceRef** dTable 3.144 contains a comprehensive list of use cases for the usage of DataPrototypeInServiceInterfaceRef.c*(RS\_MANI\_00008, RS\_MANI\_00024)*

|  |  |
| --- | --- |
| **Use case** | **Role** |
| AutosarDataPrototype typed by an Application-  DataType | dataPrototype |
| DataPrototype in AutosarDataPrototype typed by an ApplicationCompositeDataType | dataPrototype |
| AutosarDataPrototype typed by a CppImplementationDataType | dataPrototype |
| DataPrototype in AutosarDataPrototype typed by a CppImplementationDataType | elementInImplDatatype |

**Table 3.144: Possible use cases for the usage of DataPrototypeInServiceInterfaceRef**

From a careful observation of Table 3.144 it should be clear that there is no valid use case to simultaneously use the two roles dataPrototype and elementInImplDatatype in the context of the same DataPrototypeInServiceInterfaceRef.

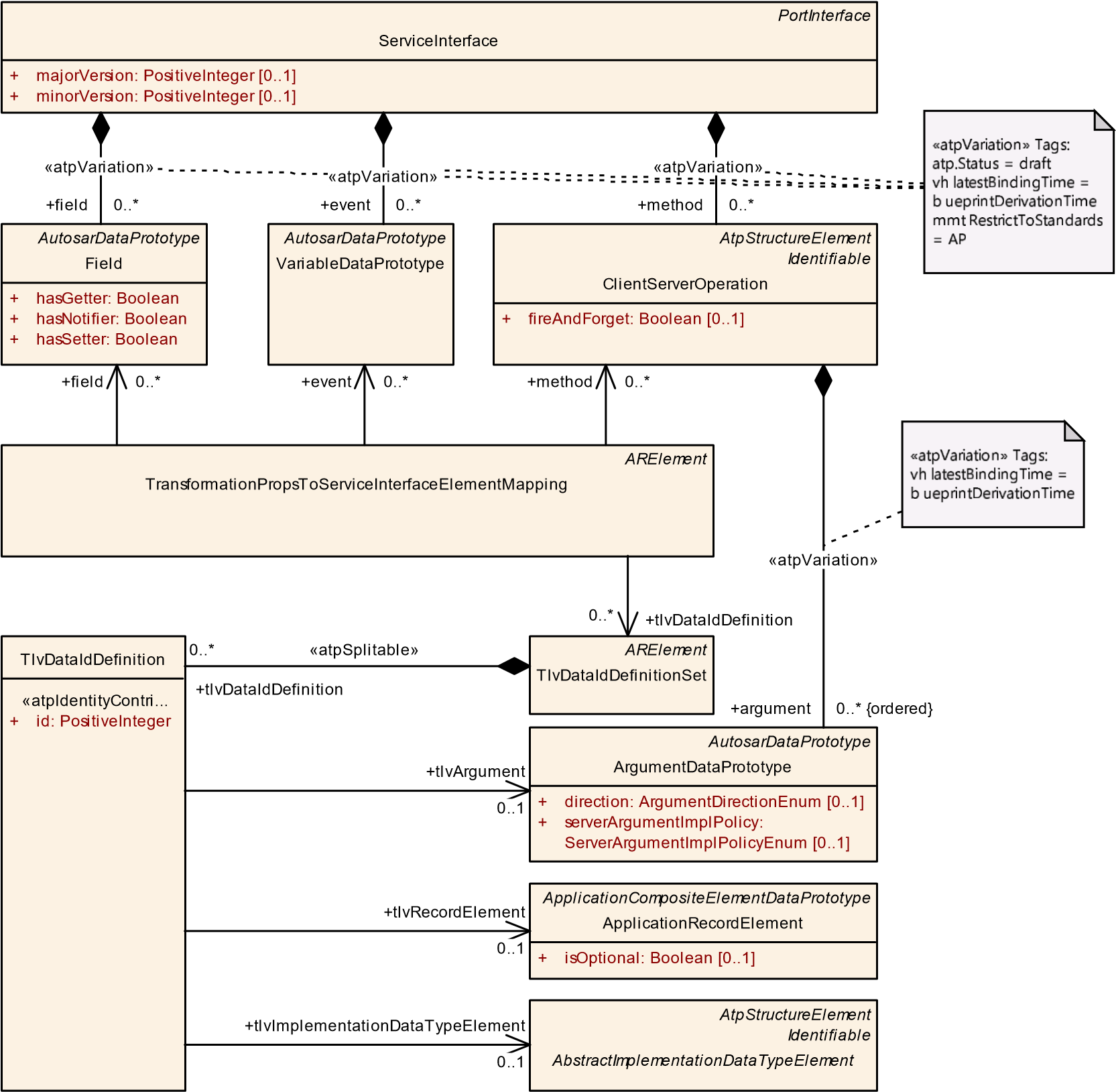
**[constr\_1551]**{DRAFT} **Existence of DataPrototypeInServiceInterfaceRef. dataPrototype vs. DataPrototypeInServiceInterfaceRef.elementInImplDatatype** dFor every given DataPrototypeInServiceInterfaceRef, either the aggregation DataPrototypeInServiceInterfaceRef.dataPrototype or DataPrototypeInServiceInterfaceRef.elementInImplDatatype shall exist.c*()*

The usage of the SomeipDataPrototypeTransformationProps.networkRepresentation is explained in more detail in the System Template [17] in [TPS\_SYST\_02136] and [TPS\_SYST\_02137].

**3.18.3 Assignment of TLV properties**

**3.18.3.1 Assignment of TLV Data IDs**

**[TPS\_MANI\_01097]**{DRAFT} **Assignment of TLV data ids** dThe assignment of TLV data ids is done in the context of the specification of TransformationPropsToServiceInterfaceElementMapping, namely by means of the attribute TransformationPropsToServiceInterfaceElementMapping. tlvDataIdDefinition.id.c*(RS\_MANI\_00030)*



**Figure 3.79: Modeling of the TLV data id**

This approach takes benefit from the fact that the TlvDataIdDefinition is able to create references to relevant model elements.

The assignment of the TLV data id is therefore done by creating such a reference and assigning a TLV data id to it by means of the attribute TlvDataIdDefinition.id.

Please note that the assignment of TLV data ids is compulsory for an entire data structure that has at least one optional member. In a nutshell, this conclusion (that is also backed by [PRS\_SOMEIP\_00230], see [7]) is the motivation for the existence of [constr\_1594], and [constr\_1595].

Please note further that the assignment of TLV data ids is not restricted to data structures with optional members. There is also a use case to support sending the elements of a specific data structure in arbitrary order even if none of the elements is considered optional.

**[TPS\_MANI\_01270]**{DRAFT} **Reference from TransformationPropsToServiceInterfaceElementMapping to TlvDataIdDefinitionSet** dThe reference from TransformationPropsToServiceInterfaceElementMapping to TlvDataIdDefinitionSet means that it is in the hand of the creator of a model to decide whether a global scope should be assumed or whether the definition needs to be customized for a specific case.c*(RS\_MANI\_00030)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **TlvDataIdDefinitionSet** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Transformer | | | |
| ***Note*** | This meta-class acts as a container of TlvDataIdDefinitions to be used in a given context  **Tags:**atp.recommendedPackage=TlvDataDefinitionSets | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| tlvDataId Definition | TlvDataIdDefinition | \* | aggr | This aggregation represents the collection of TlVDataTid Definitions aggregated by the TlvDataIdDefinitionSet  **Stereotypes:** atpSplitable  **Tags:**atp.Splitkey=tlvDataIdDefinition.id |

**Table 3.145: TlvDataIdDefinitionSet**

**[constr\_1594]**{DRAFT} **Consistent assignment of TLV data ids to Application-**

**RecordDataType** dFor every ApplicationRecordDataType where direct members set the attribute ApplicationRecordElement.isOptional to the value True references to **all direct members** of this ApplicationRecordDataType shall be created on the basis of the definition of TlvDataIdDefinition.c*()*

**[constr\_1595]**{DRAFT} **Consistent assignment of TLV data ids to CppImplementationDataType or CppImplementationDataTypeElement**dFor every CppImplementationDataType of categorySTRUCTURE where direct members set the attribute CppImplementationDataTypeElement.isOptional to the value True references to **all direct members** of this CppImplementationDataType shall be created on the basis of the definition of TlvDataIdDefinition.c*()*

The definition of a TlvDataIdDefinition that refers to an eligible model element is not limited to scenarios where optional elements are defined. It is also possible to define TlvDataIdDefinition for arbitrary methods or data structures.

A typical use case could be to prepare the argument list or sub-elements for future extensions. However, if one argument or sub-element is referenced then it is necessary to define references from TlvDataIdDefinitions to all other arguments or subelements as well.

**[constr\_1593]**{DRAFT} **Completeness of the existence of a set of TlvDataId-**

**Definition.tlvArguments** dIf the reference TlvDataIdDefinition.tlvArgument exists for one argument of a given ClientServerOperation then further TlvDataIdDefinition.tlvArgument shall exist **for all** arguments of the given ClientServerOperation and all affected TlvDataIdDefinition shall be aggregated by the same TransformationPropsToServiceInterfaceElementMapping.c*()*

Although it would be possible to apply an optimization in the definition of the TLV configuration such that the TLV configuration could be defined direction-specific[[8]](#footnote-8), AUTOSAR defines that such a mixed TLV configuration shall not be used.

**[constr\_1603]**{DRAFT} **Completeness of the existence of a set of TlvDataIdDefinition.tlvRecordElements** dIf the reference TlvDataIdDefinition.tlvRecordElement exists for one element of a given ApplicationRecordDataType then further TlvDataIdDefinition.tlvRecordElement shall exist **for all** elements of the given ApplicationRecordDataType and all affected TlvDataIdDefinition shall be aggregated by the same TransformationPropsToServiceInterfaceElementMapping.c*()*

**[constr\_1604]**{DRAFT} **Completeness of the existence of a set of Tlv-**

**DataIdDefinition.tlvImplementationDataTypeElements** dIf the reference TlvDataIdDefinition.tlvImplementationDataTypeElement exists for one subElement of a given CppImplementationDataType or CppImplementationDataTypeElement then further TlvDataIdDefinition.tlvImplementationDataTypeElement shall exist **for all** subElements of the given CppImplementationDataType or CppImplementationDataTypeElement and all affected TlvDataIdDefinition shall be aggregated by the same TransformationPropsToServiceInterfaceElementMapping.c*()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **TlvDataIdDefinition** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Transformer | | | |
| ***Note*** | This meta-class represents the ability to define the tlvDataId. | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| id | PositiveInteger | 1 | attr | This attribute represents the definition of the value of the TlvDataId  **Stereotypes:** atpIdentityContributor |
| tlvArgument | ArgumentDataPrototype | 0..1 | ref | This reference assigns a tlvDataId to a given argument of a ClientServerOperation. |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **TlvDataIdDefinition** |  |  |  |
| tlv  Implementation  DataType  Element | AbstractImplementation  DataTypeElement | 0..1 | ref | This reference associates the definition of a TLV data id with a given AbstractImplementationDataTypeElement. |
| tlvRecord Element | ApplicationRecord  Element | 0..1 | ref | This reference associates the definition of a TLV data id with a given ApplicationRecordElement. |

**Table 3.146: TlvDataIdDefinition**

The definition of a TlvDataIdDefinition.id has the purpose to provide means to unambiguously identify the argument or sub-element. For this purpose, the value of the id needs to be unique in the respective context.

**[constr\_1596]**{DRAFT} **Scope of the uniqueness of the value of TlvDataId-**

**Definition.id for references to ArgumentDataPrototype** dFor all Tlv-

DataIdDefinition that are referencing ArgumentDataPrototypes of a given

ClientServerOperation in the role tlvArgument, the attribute TlvDataIdDefinition.id **shall exist and have a unique value per communication direction**, i.e. in the context of the collection of all

* arguments where attribute direction is set to either in or inout
* arguments where attribute direction is set to either out or inout
* arguments where attribute direction is set to inout (if the method **only** has arguments where attribute direction is set to inout) of the respective enclosing ClientServerOperation.c*()*

Rationale for the existence of [constr\_1596]: arguments where attribute direction is set to either in or inout are never sent in the same SOME/IP message as arguments where attribute direction is set to either out or inout.

**[constr\_1597]**{DRAFT} **Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to ApplicationRecordElement**dFor all TlvDataIdDefinition that are referencing ApplicationRecordElements of a given ApplicationDataType in the role tlvRecordElement the attribute TlvDataIdDefinition.id **shall exist and have a unique value** in the context of respective enclosing ApplicationRecordDataType.c*()*

**[constr\_1598]**{DRAFT} **Scope of the uniqueness of the value of TlvDataId-**

**Definition.id for references to CppImplementationDataTypeElement** dFor all TlvDataIdDefinition that are referencing CppImplementationDataTypeElements of a given CppImplementationDataType/CppImplementationDataTypeElement in the role tlvImplementationDataTypeElement the attribute TlvDataIdDefinition.id **shall exist and have a unique value** in the context of respective enclosing CppImplementationDataType or CppImplementationDataTypeElement.c*()*

Obviously, it is necessary to avoid ambiguity with respect to the definition of TLV data ids. Each model element that can be assigned such an id shall only be assigned one id.

**[constr\_1599]**{DRAFT}**TlvDataIdDefinition referencing ArgumentDataPrototype** dEach ArgumentDataPrototype shall be referenced **at most once** in the role tlvArgument in the context of the same TransformationPropsToServiceInterfaceElementMapping.c*()*

**[constr\_1600]**{DRAFT} **TlvDataIdDefinition referencing ApplicationRecordElement**dEach ApplicationRecordElement shall be referenced **at most once** in the role tlvRecordElement in the context of the same TransformationPropsToServiceInterfaceElementMapping.c*()*

**[constr\_1601]**{DRAFT} **TlvDataIdDefinition referencing CppImplementationDataTypeElement** dEach CppImplementationDataTypeElement shall be referenced **at most once** in the role tlvImplementationDataTypeElement in the context of the same TransformationPropsToServiceInterfaceElementMapping.c*()*

**[constr\_1748]**{DRAFT} **Existence of references TlvDataIdDefinition.tlvArgument, TlvDataIdDefinition.tlvRecordElement, and TlvDataIdDefinition.tlvImplementationDataTypeElement** dFor each TlvDataIdDefinition, only one out of the following references shall exist:

* reference to an ArgumentDataPrototype in the role tlvArgument
* reference to an ApplicationRecordElement in the role tlvRecordElement
* reference to an AbstractImplementationDataTypeElement in the role tlvImplementationDataTypeElement.

### c()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***AbstractImplementationDataTypeElement*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes | | | |
| ***Note*** | This meta-class represents the ability to act as an abstract base class for specific derived meta-classes that support the modeling of ImplementationDataTypes for a particular language binding. | | | |
| ***Base*** | *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*,  *Referrable* | | | |
| ***Subclasses*** | CppImplementationDataTypeElement, ImplementationDataTypeElement | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.147: AbstractImplementationDataTypeElement**

**[constr\_1628]**{DRAFT} **Definition of static length field sizes in case of TLV usage** dIf the aggregation tlvDataIdDefinition exists for a given Transformation-

PropsToServiceInterfaceElementMapping then attributes

* sizeOfArrayLengthField,
* sizeOfStringLengthField,
* sizeOfStructLengthField, and
* sizeOfUnionLengthField

shall have a value greater than 0.c*()*

Rationale for the existence of [constr\_1628]: The TLV serialization requires the usage of length fields:

* If wire type 4 is used (for more details, please refer to [TPS\_MANI\_01186]) then the length field size shall be statically configured.
* If wire types 5-7 are used (see [TPS\_MANI\_01186]) then the static configuration of the length field size shall also be present since not all length fields are preceded by a tag, e.g. structures contained in an array or the top-level structure contained in a SOME/IP event.

Without demanding the existence of length fields in such a case the result of a serialization could be ambiguous, i.e. make it impossible for the de-serializer to figure out the data layout[[9]](#footnote-9).

**[constr\_1629]**{DRAFT} **Identical sizes of length fields in case of TLV usage** dIf the aggregation tlvDataIdDefinition exists for a given TransformationPropsToServiceInterfaceElementMapping then attributes

* sizeOfArrayLengthField,
* sizeOfStringLengthField,
* sizeOfStructLengthField, and
* sizeOfUnionLengthField

shall have an identical value.c*()*

Rationale for the existence of [constr\_1629]: if wire type 4 is used (for more details, please refer to [TPS\_MANI\_01186]) and if the receiver encounters a member of a structure or an argument with an unknown tag the de-serializer cannot determine the actual data type of the member of the structure or argument.

**[constr\_1630]**{DRAFT} **No definition of length field sizes on DataPrototype level in case of TLV usage** dIf the reference in the role tlvDataIdDefinition exists for a given TransformationPropsToServiceInterfaceElementMapping

then attributes

* sizeOfArrayLengthField,
* sizeOfStringLengthField,
* sizeOfStructLengthField, and
* sizeOfUnionLengthField

shall not be individually defined on the level of a DataPrototype (i.e. by means of the reference SomeipDataPrototypeTransformationProps.someipTransformationProps) but only on the level of a ServiceInterface (i.e. by means of the reference TransformationPropsToServiceInterfaceElementMapping.transformationProps).c*()*

Rationale for the existence of [constr\_1630]: if wire type 4 is used (for more details, please refer to [TPS\_MANI\_01186]) and if the receiver encounters a member or argument with an unknown tag the de-serializer needs to know the size of the length field.

The most reliable way to achieve this is to demand the definition of the size of the length field on the level of the ServiceInterface.

**3.18.3.2 Assignment of Wire Type Selection**

The TLV encoding supports the definition of a so-called wire type that controls how the information about the length of length fields shall be interpreted.

The meaning of specific settings of the wire type is defined in [7, PRS SOME/IP Protocol].

**[TPS\_MANI\_01186]**{DRAFT} **Definition of the applicable wire type** dAttribute

ApSomeipTransformationProps.isDynamicLengthFieldSize shall be used to define the applicable wire type.

If the value of attribute ApSomeipTransformationProps.isDynamicLengthFieldSize is set to True then wire type 5-7 shall be used.

If the value of attribute ApSomeipTransformationProps.isDynamicLength-

FieldSize does not exist or is set to False then wire type 4 shall be used.c

*(RS\_MANI\_00030)*

## 3.19 Process Design

Within the definition of e.g. a diagnostic mapping, the assignment to the Process is typically done in a methodological step[[10]](#footnote-10) that happens when all the diagnostic mapping[[11]](#footnote-11) is already complete.

Therefore, it would be good to implement a proxy for an actual Process that can stand in as the target of the relation to a Process at design time. This semantics is realized by meta-class ProcessDesign.

**[TPS\_MANI\_01228]**{DRAFT} **Semantics of meta-class ProcessDesign** dMetaclass ProcessDesign shall be used whenever a design-time representation is required for a Process that is designed in a **later** step in the workflow as part of the deployment specification.c*(RS\_MANI\_00004)*

The integrator would have to take care that an actual Process refers to the corresponding ProcessDesign such that by means of this reference an AUTOSAR software tool is able to figure out the relation between a diagnostic mapping and a process, provided that each ProcessDesign is **only** referenced by a single Process.

**[constr\_1550]**{DRAFT} **Reference from Process to ProcessDesign**dEach ProcessDesign shall only be referenced from a single Process.c*()*

Note that the reference from the Process to the ProcessDesign acknowledges the fact that the Process is typically created later in time[[12]](#footnote-12).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ProcessDesign** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ProcessDesign | | | |
| ***Note*** | This meta-class has the ability to stand in for a Process at the time when the Process does not yet exist. But its future existence already needs to be considered during design phase and for that a dedicated model element is required..  **Tags:**  atp.Status=draft  atp.recommendedPackage=ProcessDesigns | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| deterministic  ClientResource  Needs | DeterministicClient  ResourceNeeds | \* | aggr | This aggregation represents the collection of applicable resource needs for the design of deterministic clients.  **Tags:**atp.Status=draft |
| executable | Executable | 0..1 | ref | Reference to executable that is executed in the process.  **Tags:**atp.Status=draft |

**Table 3.148: ProcessDesign**

Conceivably, the association of diagnostic mappings with Meta-class ProcessDesign may still happen as a finalizing last step of the activity to create the diagnostic mappings. To accommodate for this potential modeling, the reference from a diagnostic mapping to ProcessDesign has been decorated by stereotype atpSplitable.

For more information concerning the semantics of this stereotype please refer to the specification of the AUTOSAR Generic Structure Template [6].

**[constr\_1693]**{DRAFT} **Relation of Executable, ProcessDesign, and Process** dAny Executable that is referenced by a ProcessDesign shall also be referenced by every Process that references the ProcessDesign.c*()*

*AtpStructureElement*

*FibexElement*

MachineDesign

+

accessControl: AccessControlEnum

[0..1]

+

pncPrepareSleepTimer: TimeValue

[0..1]

+

pnResetTimer: TimeValue

[0..1]

*ARElement*

ProcessDesign

*ARElement*

ProcessDesignToMachineDesignMapping

machineDesign

+

0..1

+

processDesign

1

.

**Figure 3.80: Modeling of the ProcessDesignToMachineDesignMapping**

**[TPS\_MANI\_01229]**{DRAFT} **Pre-allocation of a given ProcessDesign on a specific MachineDesign** dIt is also possible to pre-allocate a given ProcessDesign on a specific MachineDesign. For this purpose meta-class ProcessDesignToMachineDesignMapping exists.c*(RS\_MANI\_00004)*

The semantics of meta-class MachineDesign is explained in section 5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ProcessDesignToMachineDesignMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign | | | |
| ***Note*** | This element is used in the design phase to predefine a mapping of a process to a machine. Such a mapping may be overruled in the deployment phase.  **Tags:**  atp.Status=draft  atp.recommendedPackage=ProcessDesignToMachineDesignMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| machineDesign | MachineDesign | 0..1 | ref | This reference identifies the MachineDesign in the context of the ProcessDesignToMachineDesignMapping.  **Tags:**atp.Status=draft |
| processDesign | ProcessDesign | 1 | ref | This reference identifies the ProcessDesign in the context of the ProcessDesignToMachineDesignMapping.  **Tags:**atp.Status=draft |

**Table 3.149: ProcessDesignToMachineDesignMapping**

Please note that an intended ProcessDesignToMachineDesignMapping may not be possible for utilization of the target machine and therefore a different ProcessToMachineMapping may be created in the deployment phase.

**3.19.1 Deterministic Client Resource**

Meta-class ProcessDesign can also be used to add support for the so-called Deterministic Client.

Please note that an explanation of the specific meaning of the term Deterministic Client is out of the scope of this document. A detailed explanation can be found in the SWS Execution Management [18].

To formalize the support for the Deterministic Client, meta-class DeterministicClientResourceNeeds is aggregated at ProcessDesign.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DeterministicClientResourceNeeds** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ProcessDesign | | | |
| ***Note*** | This meta-class specifies process and cycle specific computing resource needs of DeterministicClient library functions.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| hardware Platform | String | 0..1 | attr | This attribute represents a textual identification of the target platform.  **Tags:**atp.Status=draft |
| initResource | DeterministicClient  Resource | 0..1 | aggr | This represents the computing resource needs of a  DeterministicClient::WaitForNextActivation kInit cycle.  **Tags:**atp.Status=draft |
| runResource | DeterministicClient  Resource | 0..1 | aggr | This represents the computing resource needs of a  DeterministicClient::WaitForNextActivation kRun cycle.  **Tags:**atp.Status=draft |

**Table 3.150: DeterministicClientResourceNeeds**

*ARElement*

ProcessDesign

*Identifiable*

DeterministicClientResourceNeeds

+

hardwarePlatform: String

[0..1]

DeterministicClientResource

[0..1]

numberOfInstructions: NormalizedInstruction

+

+

sequentialInstructionsBegin: NormalizedInstruction

[0..1]

+

sequentialInstructionsEnd: NormalizedInstruction

[0..1]

[0..1]

+

speedup: Float

+

deterministicClientResourceNeeds

0..\*

runResource

+

0..1

+

initResource

0..1

**Figure 3.81: Modeling of the DeterministicClientResourceNeeds**

**[TPS\_MANI\_01199]**{DRAFT} **Semantics of DeterministicClientResourceNeeds** dMeta-class DeterministicClientResourceNeeds aggregates DeterministicClientResource in two roles in order to be able to specify resource needs in two different contexts of the execution of a Deterministic Client.c*(RS\_MANI\_00050)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DeterministicClientResource** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ProcessDesign | | | |
| ***Note*** | This meta-class specifies computing resource needs of DeterministicClient library functions.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| numberOf  Instructions | NormalizedInstruction | 0..1 | attr | This attribute represents the normalized runtime consumption on the target system within one  DeterministicClient::WaitForNextActivation cycle, assuming the "worst-case" runtime where the workers would be executed sequentially.  **Tags:**atp.Status=draft |
| sequential Instructions  Begin | NormalizedInstruction | 0..1 | attr | Normalized sequential runtime at the beginning of the  DeterministicClient::WaitForNextActivation cycle (which mostly cannot be parallelized), before the main usage of the worker pool starts. **Tags:**atp.Status=draft |
| sequential  InstructionsEnd | NormalizedInstruction | 0..1 | attr | WaitForNextActivation cycle (which mostly cannot be parallelized), after the main usage of the worker pool has ended.  **Tags:**atp.Status=draft |
| speedup | Float | 0..1 | attr | This attribute defines how much faster the calculations within one DeterministicClient::WaitForNextActivation cycle can be finished if numberOfWorkers are physically available, i.e. if enough cores were available on the machine to perform parallel execution of all workers (sequential runtime / parallelized runtime).  **Tags:**atp.Status=draft |

**Table 3.151: DeterministicClientResource**

**[TPS\_MANI\_01200]**{DRAFT} **Semantics of meta-class DeterministicClientResource** dMeta-class DeterministicClientResource defines several attributes that provide information about the nature of the execution of worker threads. The values of these attributes are given a dimensionless NormalizedInstruction.

Nevertheless, the values of the attributes

* numberOfInstructions
* sequentialInstructionsBegin
* sequentialInstructionsEnd

are only valid for a specific hardware platform. The purpose of using NormalizedInstruction is to align resource usage of different Processes (possibly from different vendors) at integration time.c*(RS\_MANI\_00050)*

|  |  |
| --- | --- |
| ***Primitive*** | **NormalizedInstruction** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ProcessDesign |

5

4

|  |  |
| --- | --- |
| ***Primitive*** | **NormalizedInstruction** |
| ***Note*** | This meta-class is used to describe runtime budget needs on the target system within Deterministic Client::WaitForNextActivation cycles. NormalizedInstructions does not reflect the actual number of code instructions, but allows the description of comparative resource needs. NormalizedInstructions is used for configuration of computing resources at integration time. NormalizedInstruction = runtime in sec \* clock frequency in Hz  **Tags:**  atp.Status=draft  xml.xsd.customType=NORMALIZED-INSTRUCTION xml.xsd.pattern=[1-9][0-9]\* xml.xsd.type=string |

**Table 3.152: NormalizedInstruction**

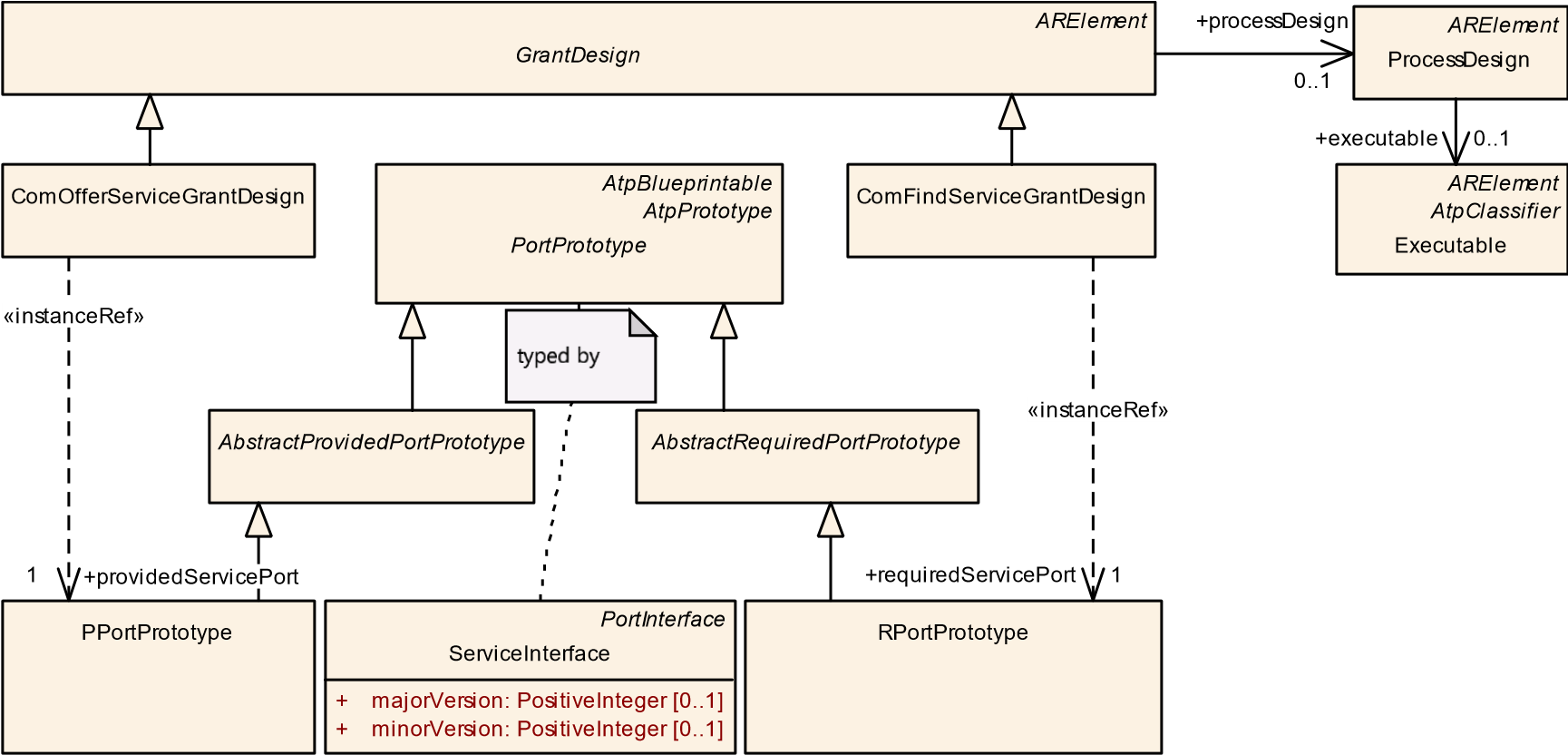
## 3.20 Grant Design

The definition of intents (for example: ClientComSpec.clientIntent) as described in chapter 3.15.5.1.3 is used to express the intention of the software designer to use (or refrain from using) specific APIs in the application software.

The definition of intents represents one aspect of the identity and Access Management (IAM). Another aspect of the IAM configuration is the definition of the actual permissions granted by the platform software.

The modeling of such grants is done on two levels:

* the definition of GrantDesign allows for the pre-specification of grants already on the design level. The modeling of GrantDesign is described in this chapter.
* the definition of Grant allows for the actual and final specification of grants from the perspective of the platform software. The modeling of Grant is described in chapter 9.9.



**Figure 3.82: Modeling of grant designs for service discovery**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***GrantDesign*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign | | | |
| ***Note*** | This meta-class serves as an abstract base class for the description of grants on design level.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Subclasses*** | ComFindServiceGrantDesign, *ComGrantDesign*, ComOfferServiceGrantDesign, RawDataStreamGrant  Design | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| processDesign | ProcessDesign | 0..1 | ref | This reference identifies the corresponding Process Design that gives context to the GrantDesing.  **Tags:**atp.Status=draft |

**Table 3.153: GrantDesign**

Abstract meta-class GrantDesign acts as the base class for the definition of grants on the design level.

Grants are specific for a given Process. In other words, two Processes created from the same Executable may be assigned different sets of grants. This specific relation shall also be available on the design level.

**[TPS\_MANI\_01231]**{DRAFT} **GrantDesign references ProcessDesign** dMetaclass GrantDesign references ProcessDesign as a means to design the set of Grants for the given Process.c*(RS\_MANI\_00060)*

**3.20.1 Com Grant Design**

Subclasses of GrantDesign are created to cover specific aspects of grants for communication on the *AUTOSAR adaptive Platform*.

**[TPS\_MANI\_01232]**{DRAFT} **Semantics of meta-class ComOfferServiceGrantDesign**dThe existence of a ComOfferServiceGrantDesign that references a specific AbstractProvidedPortPrototype in the role providedServicePort indicates that the design foresees that the referenced AbstractProvidedPortPrototype shall be granted rights to offer the respective service.c*(RS\_MANI\_00060)*

Please note that there is no explicitly modeled intent that corresponds to the existence of the ComOfferServiceGrantDesign. The understanding is that the mere existence of an AbstractProvidedPortPrototype typed by a ServiceInterface indicates the intent to offer a service.

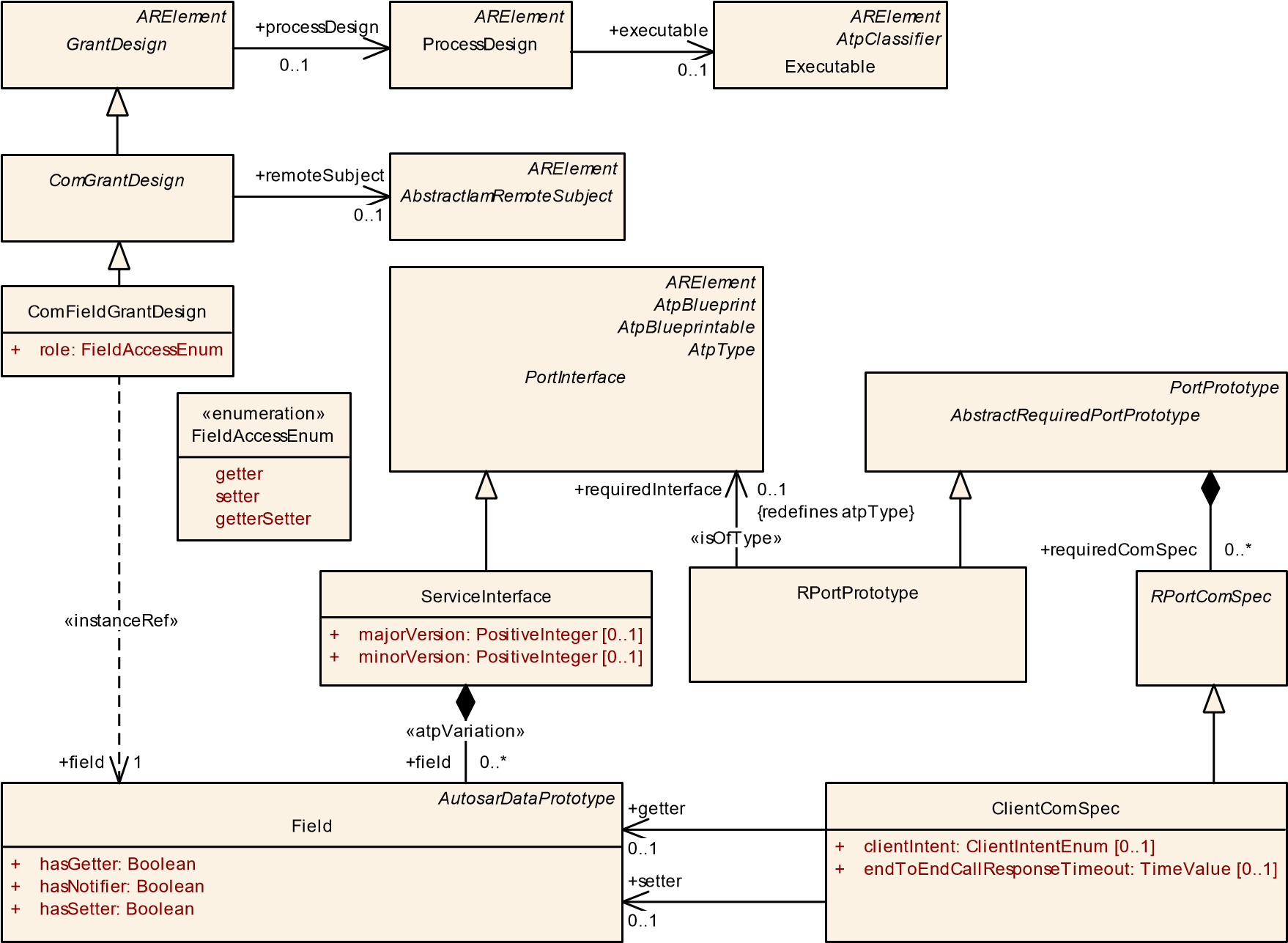
**[TPS\_MANI\_01233]**{DRAFT} **Semantics of meta-class ComFindServiceGrantDesign**dThe existence of a ComFindServiceGrantDesign that references a specific AbstractRequiredPortPrototype in the role requiredServicePort indicates that the design foresees that the referenced AbstractRequiredPortPrototype shall be granted rights to find the respective service.c*(RS\_MANI\_00060)*

Please note that there is no explicitly modeled intent that corresponds to the existence of the ComFindServiceGrantDesign.

The understanding is that the mere existence of an AbstractRequiredPortPrototype typed by a ServiceInterface indicates the intent to find a service.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ComOfferServiceGrantDesign** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant | | | |
| ***Note*** | This meta-class represents the ability to define a Grant for offering a service.  **Tags:**  atp.Status=draft  atp.recommendedPackage=GrantDesigns | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| providedService Port | PPortPrototype | 1 | iref | This instanceRef identifies the PPortPrototype on which the service shall be offered.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**PPortPrototypeIn  ExecutableInstanceRef |

**Table 3.154: ComOfferServiceGrantDesign**



**Figure 3.83: Modeling of grant designs for field**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ComFindServiceGrantDesign** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant | | | |
| ***Note*** | This meta-class represents the ability to define a Grant for finding a service.  **Tags:**  atp.Status=draft  atp.recommendedPackage=GrantDesigns | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| requiredService Port | RPortPrototype | 1 | iref | This instanceRef identifies the RPortPrototype on which the service shall be found.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**RPortPrototypeIn  ExecutableInstanceRef |

**Table 3.155: ComFindServiceGrantDesign**

**[TPS\_MANI\_01234]**{DRAFT} **Semantics of ComFieldGrantDesign** dThe existence of a ComFieldGrantDesign that references a specific Field in the role field indicates that the design foresees that the application software shall be granted rights to access the respective Field. The nature of the access, i.e. get vs. set is specified by means of the attribute role.c*(RS\_MANI\_00060)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ComFieldGrantDesign** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant | | | |
| ***Note*** | This meta-class represents the ability to define a Grant for a ServiceInterface.field.  **Tags:**  atp.Status=draft  atp.recommendedPackage=GrantDesigns | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *ComGrantDesign*, *GrantDesign*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| field | Field | 1 | iref | Reference to the affected Field in the context of an Executable.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**FieldInExecutableInstance Ref |
| role | FieldAccessEnum | 1 | attr | This attribute provides the ability to further specify the access to the ServiceInterface.field from a design perspective.  **Tags:**atp.Status=draft |

**Table 3.156: ComFieldGrantDesign**

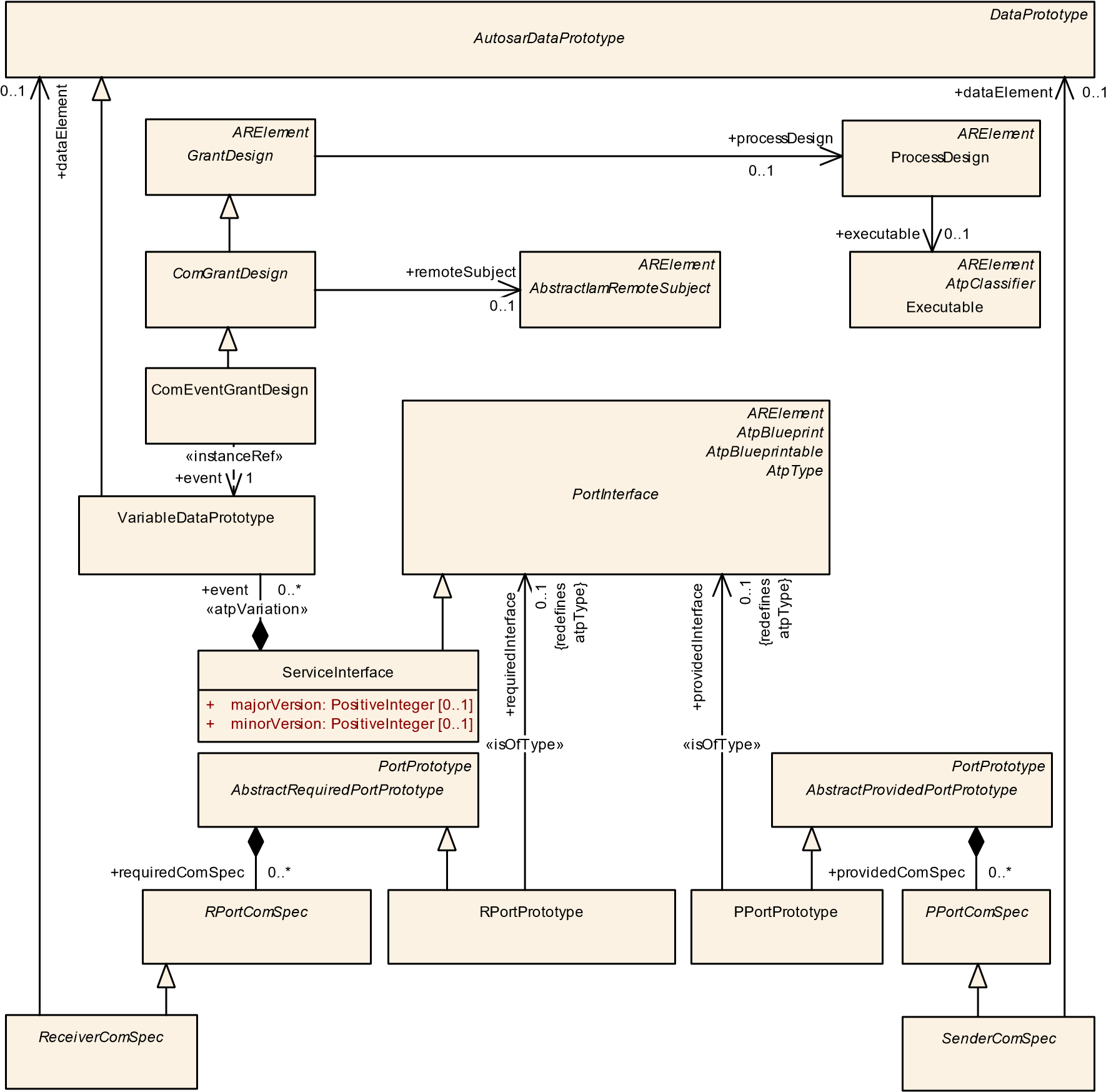
|  |  |
| --- | --- |
| ***Enumeration*** | **FieldAccessEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant |
| ***Note*** | This meta-class provides values that qualify access to a field.  **Tags:**atp.Status=draft |
| ***Literal*** | ***Description*** |

5

4

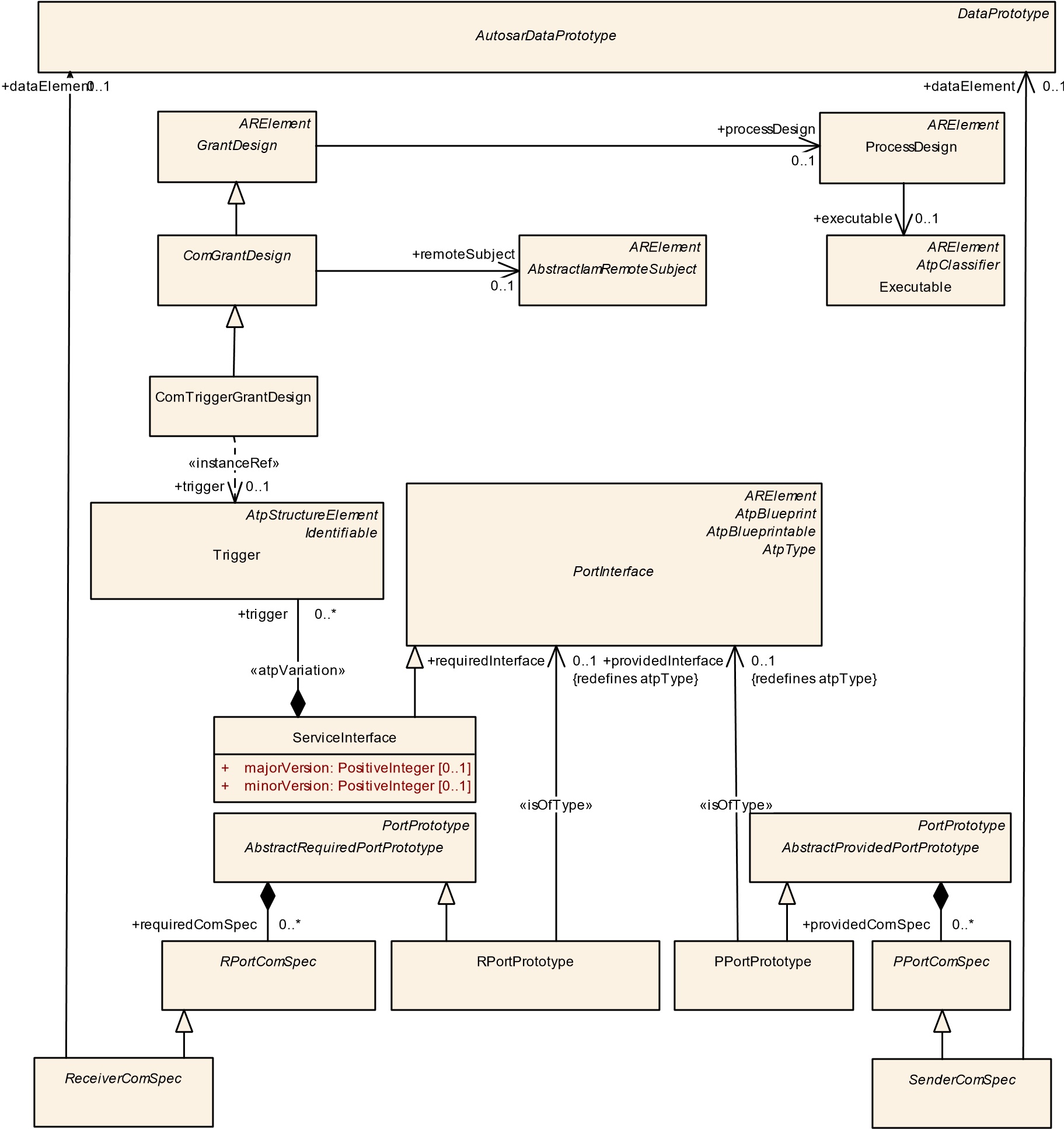
|  |  |
| --- | --- |
| ***Enumeration*** | **FieldAccessEnum** |
| getter | Access to the getter of the Field.  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |
| getterSetter | Access to getter and setter of the field  **Tags:**  atp.EnumerationLiteralIndex=2 atp.Status=draft |
| setter | Access to the setter of the Field.  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |

**Table 3.157: FieldAccessEnum**



**Figure 3.84: Modeling of grant designs for event**

**[TPS\_MANI\_01235]**{DRAFT} **Semantics of ComEventGrantDesign** dThe existence of a ComEventGrantDesign that references a specific VariableDataPrototype that is aggregated in the role event by the enclosing ServiceInterface indicates that the design foresees that the application software shall be granted rights to access the respective event.c*(RS\_MANI\_00060)*



**Figure 3.85: Modeling of grant designs for trigger**

|  |  |
| --- | --- |
| ***Class*** | **ComEventGrantDesign** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ComEventGrantDesign** | | | |
| ***Note*** | This meta-class represents the ability to define a Grant for a ServiceInterface.event.  **Tags:**  atp.Status=draft  atp.recommendedPackage=GrantDesigns | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *ComGrantDesign*, *GrantDesign*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| event | VariableDataPrototype | 1 | iref | This reference represents the affected event.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**EventInExecutable InstanceRef |

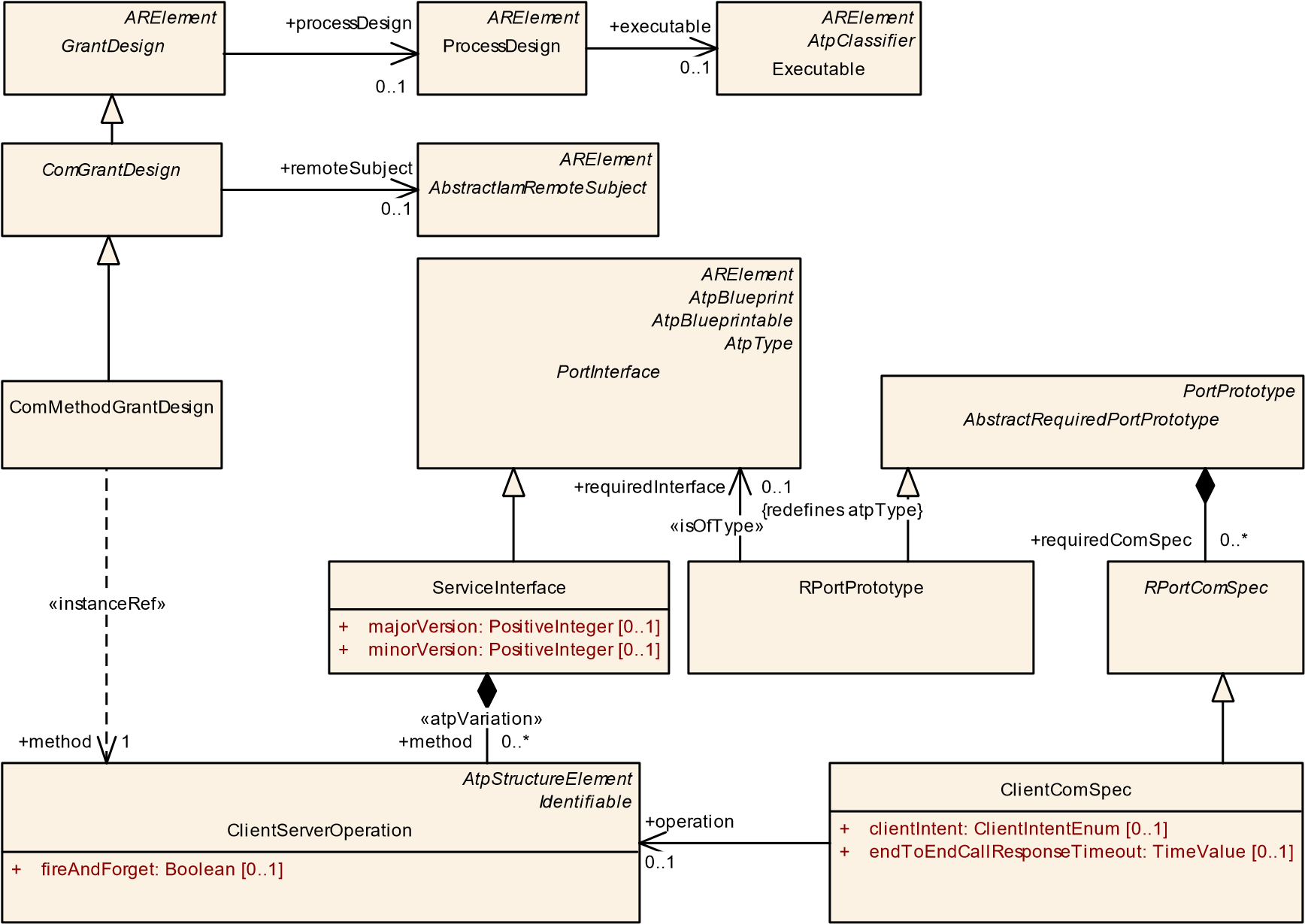
**Table 3.158: ComEventGrantDesign**

**[TPS\_MANI\_03290]**{DRAFT} **Semantics of ComTriggerGrantDesign** dThe existence of a ComTriggerGrantDesign that references a specific Trigger that is aggregated in the role trigger by the enclosing ServiceInterface indicates that the design foresees that the application software shall be granted rights to access the respective trigger.c*(RS\_MANI\_00060)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ComTriggerGrantDesign** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant | | | |
| ***Note*** | This meta-class represents the ability to define a Grant for a ServiceInterface.trigger.  **Tags:**  atp.Status=draft  atp.recommendedPackage=GrantDesigns | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *ComGrantDesign*, *GrantDesign*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| trigger | Trigger | 1 | iref | This reference represents the affected trigger.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**TriggerInExecutable InstanceRef |

**Table 3.159: ComTriggerGrantDesign**

**[TPS\_MANI\_01236]**{DRAFT} **Semantics of ComMethodGrantDesign** dThe existence of a ComMethodGrantDesign that references a specific ClientServerOperation that is aggregated in the role method by the enclosing ServiceInterface indicates that the design foresees that the application software shall be granted rights to call the respective method.c*(RS\_MANI\_00060)*



**Figure 3.86: Modeling of grant designs for method**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ComMethodGrantDesign** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant | | | |
| ***Note*** | This meta-class represents the ability to define a Grant for a ServiceInterface.method.  **Tags:**  atp.Status=draft  atp.recommendedPackage=GrantDesigns | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *ComGrantDesign*, *GrantDesign*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| method | ClientServerOperation | 1 | iref | This reference identifies the corresponding method.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**RequiredMethodIn ExecutableInstanceRef |

**Table 3.160: ComMethodGrantDesign**

**3.20.2 Grant Design for Raw Streaming Data**

The usage of a raw data stream is subject to restrictions imposed by the IAM. Therefore, meta-class RawDataStreamGrantDesign exists to support this use case.

**[TPS\_MANI\_01284]**{DRAFT} **Granularity of meta-class RawDataStreamGrantDesign** dThe granularity of the RawDataStreamGrantDesign is the entire AbstractRawDataStreamInterface. It is not expected that a definition of an IAM policy makes sense on a smaller level, i.e. on the level of ClientServerOperation aggregated by a AbstractRawDataStreamInterface.c*(RS\_MANI\_00060)*

*ARElement*

ProcessDesign

*ARElement*

*GrantDesign*

RawDataStreamGrantDesign

*AbstractRawDataStreamInterface*

RawDataStreamClientInterface

+

processDesign

0..1

+

rawDataStream

0..1

**Figure 3.87: Modeling of the RawDataStreamGrantDesign**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **RawDataStreamGrantDesign** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::RawDataStreamGrant | | | |
| ***Note*** | This meta-class represents the ability to define the IAM configuration for a RawDataStream on design level.  **Tags:**  atp.Status=draft  atp.recommendedPackage=GrantDesigns | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| rawDataStream | RawDataStreamClient  Interface | 0..1 | ref | This reference identifies the applicable RawDataStream Interface.  **Tags:**atp.Status=draft |

**Table 3.161: RawDataStreamGrantDesign**

**3.20.3 Remote access control**

The definition of the deployment for the *Identity and Access Manager* and the definition of grants relies on the local enforcement of identity and access permissions. In other words it is possible for example to define that a particular method of a ServiceInterface is allowed to be called by a local Process on the local Machine. But it is not possible to restrict the remote Machines that are allowed to call this method.

The fact that the Machine on which the service is running has no mean to make additional checks on the incoming requests enables processing of wrongly issued requests by a healthy remote Machine as well as escalation of privileges by an attacker via issuing arbitrary request towards services from a compromised remote Machine.

Most of the times it is not possible for a Machine to recognize that its communicating peer is compromised because the attacker has access to all the resources of that Machine and can run in stealth mode. An effective way to minimize the damage of a compromised remote Machine is to enforce additional checks on the incoming requests at the receiver side ensuring that remote Machine cannot go beyond what they could request in a healthy state.

|  |  |  |
| --- | --- | --- |
| Network information -> SubjectId |  | SubjectId x Object -> {allow, deny} |

Subject

Authentication

Authorization

Object

**Figure 3.88: Access policy enforcement based on the Subject ID from the network binding**

The access control process aims at enforcing policies on the relation between a “Subject” and an “Object”. In the example where an remote Machine makes calls to a service interface, the remote Machine is the Subject and the method of a ServiceInterface is the Object.

The access control process comprises of the two main operations, namely, Authentication and Authorization, which are mostly independent. During the authentication process the identity of the subject is verified and an authentic identifier is resolved. Authentication is an essential part of the chain to ensure that different subjects cannot impersonate each other.

In the next step, during Authorization, the identity of the Subject is checked upon the rules and policies defined for the accessing the Object to verify if the Subject’s request is legitimate. These policies shall be defined by the system or the resource owner.

The authentication of the remote subject is based on the network binding. When a secure channel is established, the remote peer has already gone through an authentication protocol. Therefore, the identity information can be forwarded to the IAM to apply the corresponding defined policies that are defined for the requests coming from that channel as depicted in Figure 3.88.

The remote subject is modeled as a specialization of AbstractIamRemoteSubject.

The different specializations will be presented in the following sections.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***AbstractIamRemoteSubject*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM | | | |
| ***Note*** | This abstract meta-class defines the proxy information about the remote node.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Subclasses*** | IPSecIamRemoteSubject, IpIamRemoteSubject, TlsIamRemoteSubject | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 3.162: AbstractIamRemoteSubject**

With the modeling of ComGrantDesigns the permissions that are granted by the platform software are defined. As an option a ComGrantDesign is able to reference the AbstractIamRemoteSubject in the role remoteSubject. **[TPS\_MANI\_03238]**{DRAFT} **Definition of ComMethodGrantDesign.remoteSubject**dIf the ComMethodGrantDesign references one or several AbstractIamRemoteSubjects in the role remoteSubject then the design foresees that only the defined remoteSubjects shall be granted rights to access the ClientServerOperation that is referenced in the role method by the same ComMethodGrantDesign.c*(RS\_MANI\_00060)*

**[TPS\_MANI\_03239]**{DRAFT} **Definition of ComEventGrantDesign.remoteSubject** dIf the ComEventGrantDesign references one or several AbstractIamRemoteSubjects in the role remoteSubject then the design foresees that only the defined remoteSubjects shall be granted rights to access the VariableDataPrototype that is referenced in the role event by the same ComEventGrantDesign.c *(RS\_MANI\_00060)*

**[TPS\_MANI\_03251]**{DRAFT} **Definition of ComFieldGrantDesign.remoteSubject** dIf the ComFieldGrantDesign references one or several AbstractIamRemoteSubjects in the role remoteSubject then the design foresees that only the defined remoteSubjects shall be granted rights to access the Field that is referenced in the role field by the same ComFieldGrantDesign.c*(RS\_MANI\_00060)*

**3.20.3.1 Remote subject in case of TLS**

This chapter defines how a AbstractIamRemoteSubject is modeled in case of a TLS-based secure channel.

**[TPS\_MANI\_03240]**{DRAFT} **Modeling of a remote peer in case of TLS-based secure channel** dIn case of TLS-based secure channel the remote peer is modeled as

TlsIamRemoteSubject that is identified either by

* a CryptoServiceCertificate that is referenced by the TlsIamRemoteSubject in the role acceptedRemoteCertificate,
* a Pre-shared Key that is referenced by the TlsIamRemoteSubject via

TlsCryptoCipherSuite in the role acceptedCryptoCipherSuiteWithPsk.

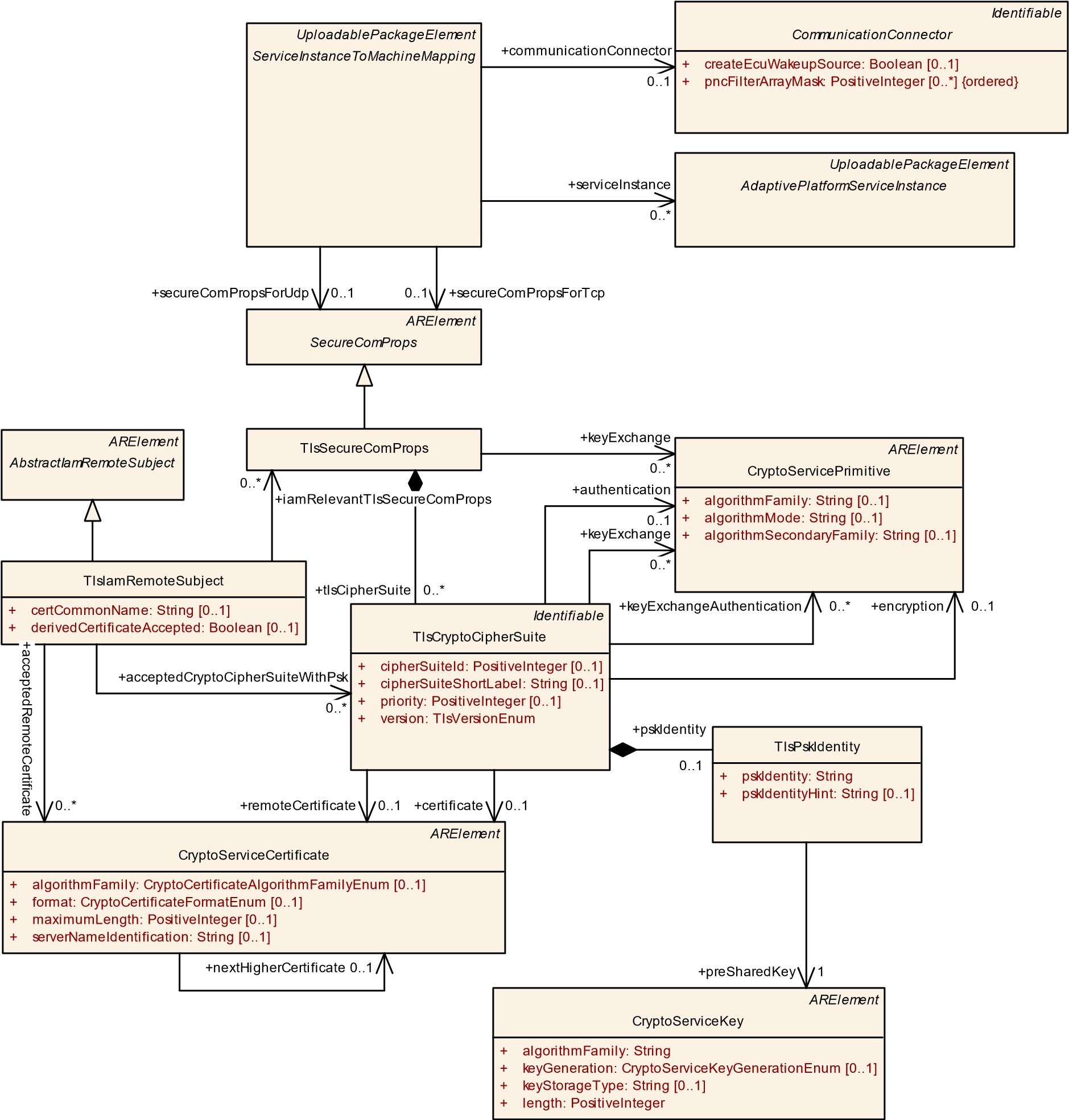
c*(RS\_MANI\_00036, RS\_MANI\_00060)*

Please note that the security of a pre-shared key as authentication in TLS protocol depends on the number of entities sharing the same key. If multiple Machines are using the same shared key, one cannot reliably distinguish between those Machines because any of them can impersonate the others.

It can only be ensured that no other Machine without the knowledge of the pre-shared key can established a secure channel.

The TlsIamRemoteSubject may be identified by using certificates in two ways. First, it is possible to directly specify the certificates that the TlsIamRemoteSubject may to use by referencing them and setting derivedCertificateAccepted to false.

This approach requires the presence of the remote certificate on the local Machine. Secondly, by setting derivedCertificateAccepted to true it is possible to specify the Common Name (as given in the X509 Certificate) of the TlsIamRemoteSubject.



**Figure 3.89: Proxy information about the remote node in case of TLS**

In that case, the acceptedRemoteCertificates define the set of allowed root certificates for the certificate presented by the TlsIamRemoteSubject.

The reason for the upper multiplicity is that the OEM may have multiple suppliers for a Machine and it shall be allowed to define that in such a case all these Machines are allowed to connect even though they have different certificate chains. **[TPS\_MANI\_03241]**{DRAFT} **Modeling of relevant TlsSecureComProps for**

**TlsIamRemoteSubject**dWith the TlsIamRemoteSubject.iamRelevantTlsSecureComProps reference it is possible to define all TlsSecureComProps that the

TlsIamRemoteSubject supports to establish a secure channel.c*(RS\_MANI\_00036,*

*RS\_MANI\_00060)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **TlsIamRemoteSubject** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM | | | |
| ***Note*** | This meta-class defines the proxy information about the remote node in case of TLS.  **Tags:**  atp.Status=draft  atp.recommendedPackage=IamRemoteSubjects | | | |
| ***Base*** | *ARElement*, *ARObject*, *AbstractIamRemoteSubject*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| acceptedCrypto CipherSuiteWith  Psk | TlsCryptoCipherSuite | \* | ref | This reference is used to identify a remote node by means of the preshared Key.  **Tags:**atp.Status=draft |
| accepted Remote  Certificate | CryptoService  Certificate | \* | ref | This reference is used to identify a remote node by means of the certificate. **Tags:**atp.Status=draft |
| certCommon Name | String | 0..1 | attr | This attribute defines the common name (CN) of the certificate of the remote peer.  **Tags:**atp.Status=draft |
| derived Certificate  Accepted | Boolean | 0..1 | attr | This attribute defines whether a derivedCertificate is accepted (true) or not (false).  **Tags:**atp.Status=draft |
| iamRelevantTls  SecureCom  Props | TlsSecureComProps | \* | ref | This reference defines the local TlsSecureComProps that are relevant for IAM.  **Tags:**atp.Status=draft |

**Table 3.163: TlsIamRemoteSubject**

**3.20.3.2 Remote subject in case of IPsec**

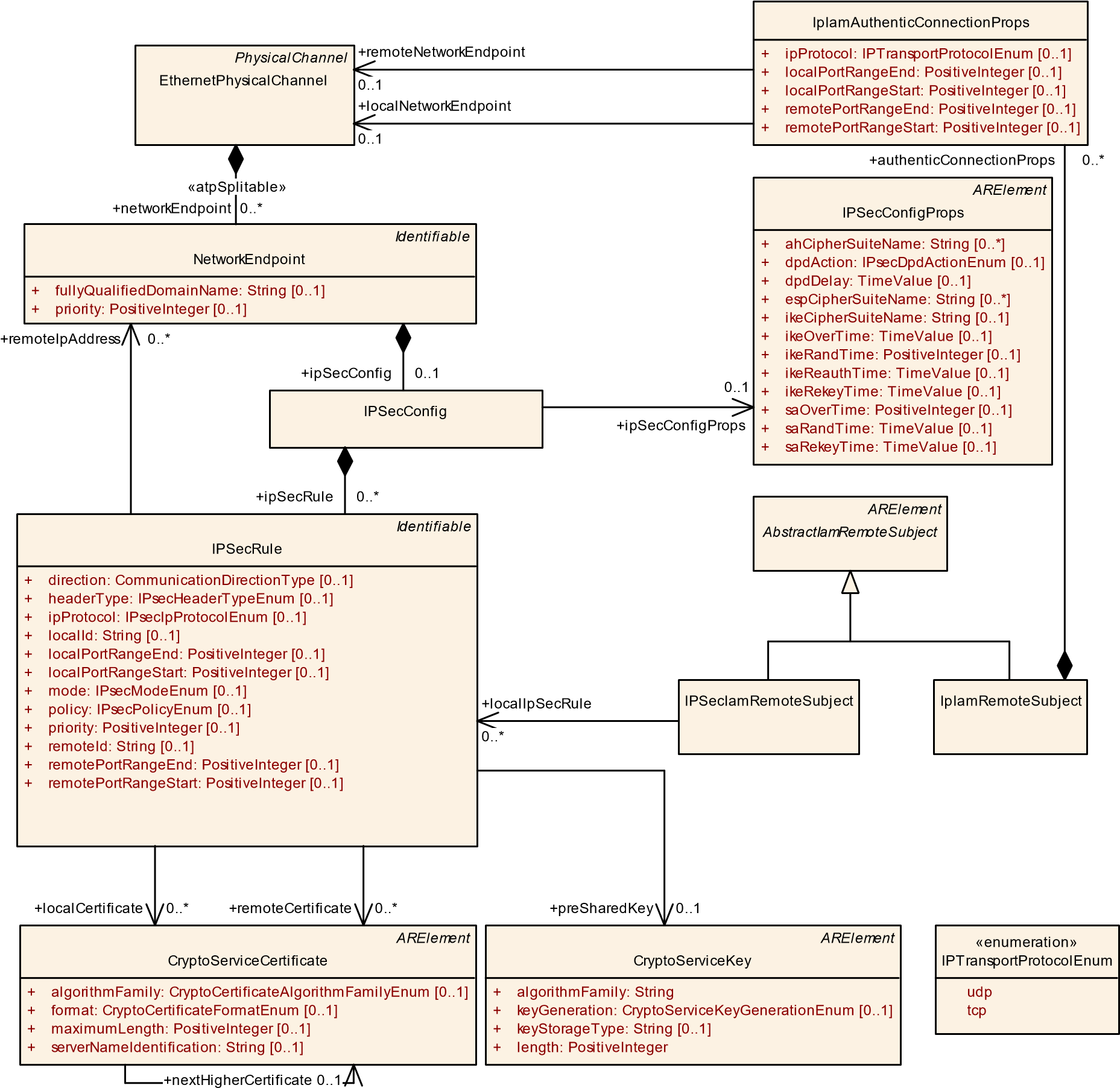
This chapter defines how a AbstractIamRemoteSubject is modeled in case of a IPsec-based secure channel.

**[TPS\_MANI\_03242]**{DRAFT} **Modeling of a remote peer in case of IPsec-based secure channel** dIn case of IPsec-based secure channel the remote peer is modeled as IPSecIamRemoteSubject that is identified by IPSecRules that are referenced by localIpSecRule. The IPSecRules define all secure connections that the remote peer is allowed to establish.c*(RS\_MANI\_00036, RS\_MANI\_00060)*

Please note that the local IP Address of the remote peer is defined by the NetworkEndpoint that aggregates the IPSecRules.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **IPSecIamRemoteSubject** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM | | | |
| ***Note*** | This meta-class defines the proxy information about the remote node in case of IPsec.  **Tags:**  atp.Status=draft  atp.recommendedPackage=IamRemoteSubjects | | | |
| ***Base*** | *ARElement*, *ARObject*, *AbstractIamRemoteSubject*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| localIpSecRule | IPSecRule | \* | ref | This reference is used to describe theRemoteSubjects local IPSecRules.  **Tags:**atp.Status=draft |

**Table 3.164: IPSecIamRemoteSubject**



**Figure 3.90: Proxy information about the remote node in case of IPsec**

**3.20.3.3 Remote subject in case of IP communication**

Please note that it is possible to define a AbstractIamRemoteSubject that is based on the general IP communication. In this case no details about how the communication is secured are given and actually securing the communication (e.g., cryptographically, via hardware mechanism, or appropriate network and switch design) is not part of the model. A IpIamRemoteSubject is identified by a combination of a local and a remote IP address, local and remote port ranges, and a transport protocol.

**[TPS\_MANI\_03244]**{DRAFT} **Modeling of a remote peer in case of a general IP communication** dIn case of a general IP communication the remote peer is modeled as IpIamRemoteSubject that is identified by the NetworkEndpoint that is referenced by the localNetworkEndpoint reference. The defined remote peer is allowed to establish IP connections to the remoteNetworkEndpoint over the ipProtocol and the defined local port range and remote port range.c*(RS\_MANI\_00060)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **IpIamRemoteSubject** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM | | | |
| ***Note*** | This meta-class defines the proxy information about the remote node in case of general IP communication.  **Tags:**  atp.Status=draft  atp.recommendedPackage=IamRemoteSubjects | | | |
| ***Base*** | *ARElement*, *ARObject*, *AbstractIamRemoteSubject*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| authentic  Connection  Props | IpIamAuthentic  ConnectionProps | \* | aggr | Definition of IP rules assigned to the IpIamRemote Subject.  **Tags:**atp.Status=draft |

**Table 3.165: IpIamRemoteSubject**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **IpIamAuthenticConnectionProps** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM | | | |
| ***Note*** | This meta-class defines a set of properties for IP connections in the context of IAM configuration.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| ipProtocol | IPTransportProtocol  Enum | 0..1 | attr | This attribute defines the relevant IP protocol.  **Tags:**atp.Status=draft |
| localNetwork Endpoint | EthernetPhysical  Channel | 0..1 | ref | This reference defines an authentic local Network Endpoint in terms of IAM configuration.  **Tags:**atp.Status=draft |
| localPortRange End | PositiveInteger | 0..1 | attr | This attribute restricts the traffic monitoring and defines an end value for the local port range.  **Tags:**atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **IpIamAuthenticConnectionProps** | |  |  |
| localPortRange Start | PositiveInteger | 0..1 | attr | This attribute restricts the traffic monitoring and defines a start value for the local port range.  **Tags:**atp.Status=draft |
| remoteNetwork Endpoint | EthernetPhysical  Channel | 0..1 | ref | This reference defines an authentic remote Network Endpoint in terms of IAM configuration.  **Tags:**atp.Status=draft |
| remotePort RangeEnd | PositiveInteger | 0..1 | attr | This attribute restricts the traffic monitoring and defines an end value for the remote port range.  **Tags:**atp.Status=draft |
| remotePort RangeStart | PositiveInteger | 0..1 | attr | This attribute restricts the traffic monitoring and defines a start value for the remote port range.  **Tags:**atp.Status=draft |

**Table 3.166: IpIamAuthenticConnectionProps**

**3.20.3.4 Remote subject in case of SecOC communication**

The identity information in the case of SecOC depends on the group of Machines that are sharing the same cryptographic key.

In other words, if a valid SecOC message is received with a given key it is given that only remote Machines that “know the key” were able to send the message. The key is associated with a DataId and defines the “object” in the access control model. If a message received for a given DataID cannot be validated, then it will be dropped. Therefore, the access control between the remote subject and local object is taking place.

To summarize, the modeling of a Remote subject in case of SecOC cannot provide additional benefit neither by increasing the granularity of the subject identification nor providing new enforcement of rules on the object.

# 4 Diagnostic Design

## 4.1 Diagnostic Mapping

**4.1.1 Overview**

The configuration of diagnostics on the *AUTOSAR adaptive platform* will typically be done by creating a Diagnostic Extract by means of the Diagnostic Extract Template [19] that is also used on the *AUTOSAR classic platform*.

Therefore, concepts within the Diagnostic Extract should be similarly applicable to models on both platforms uniformly.

It can even be safely expected that a given Diagnostic Extract can be divided into parts that apply for ECUs build on top of the *AUTOSAR classic platform* and parts that apply to ECUs built on top of the *AUTOSAR adaptive platform* that all belong to the same vehicle.

In terms of applicability to this document, the part of the Diagnostic Extract that is relevant in this context is the mapping between the definition of information related to diagnostic protocol content and the application software.

In order to exemplify the approach, the diagram depicted in Figure 4.1 describes a very simplistic situation where two different PPortPrototypes typed by possibly two different DiagnosticDataIdentifierInterface exposed by an AdaptiveApplicationSwComponentType is accessed by the AUTOSAR Adaptive Diagnostic Management on the *AUTOSAR adaptive platform* with the purpose of accessing an entire DID.

P



R

Application Software

Component

Diagnostic Manager

DiagnosticDataIdentifierInterface

P

DiagnosticDataPortMapping

DiagnosticDataIdentifier

Data

Element

Data

Element

Data

Element

Data

Element

**Figure 4.1: Example data exchange for diagnostic purpose**

In particular, a subclass of DiagnosticSwMapping (in this specific case: DiagnosticDataPortMapping) formalizes the “connection” between both ends of the communication.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***DiagnosticSwMapping*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping::ServiceMapping | | | |
| ***Note*** | This represents the ability to define a mapping between a diagnostic information (at this point there is no way to become more specific about the semantics) to a software-component. | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Subclasses*** | DiagnosticAuthenticationPortMapping, DiagnosticClearConditionPortMapping, DiagnosticDataPort  Mapping, DiagnosticEnableConditionPortMapping, DiagnosticEventPortMapping, DiagnosticExternal  AuthenticationPortMapping, DiagnosticFimFunctionMapping, DiagnosticIndicatorPortMapping,  DiagnosticMemoryDestinationPortMapping, DiagnosticMonitorPortMapping, DiagnosticOperationCycle  PortMapping, DiagnosticSecurityLevelPortMapping, DiagnosticServiceGenericMapping, Diagnostic ServiceSwMapping, DiagnosticServiceValidationMapping | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 4.1: DiagnosticSwMapping**

A further kind of mapping that is necessary to enable diagnostics on the *AUTOSAR adaptive platform* comes with slightly more complexity.

In this case use-cases are implemented that may or may not involve several communication ends (in the form of PortPrototypes).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticDataIdentifier** | | | |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics | | | |
| ***Note*** | This meta-class represents the ability to model a diagnostic data identifier (DID) that is fully specified regarding the payload at configuration-time.  **Tags:**atp.recommendedPackage=DiagnosticDataIdentifiers | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticAbstractDataIdentifier*, *DiagnosticCommon*  *Element*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| dataElement | DiagnosticParameter | \* | aggr | This is the dataElement associated with the Diagnostic  DataIdentifier.  **Stereotypes:** atpSplitable; atpVariation **Tags:**  atp.Splitkey=dataElement.bitOffset, dataElement.variation Point.shortLabel  vh.latestBindingTime=postBuild |
| didSize | PositiveInteger | 0..1 | attr | This attribute indicates the size in bytes of the Diagnostic DataIdentifier. |
| representsVin | Boolean | 0..1 | attr | This attributes indicates whether the specific Diagnostic DataIdentifier represents the vehicle identification. |
| supportInfoByte | DiagnosticSupportInfo  Byte | 0..1 | aggr | This attribute represents the supported information associated with the DiagnosticDataIdentifier. |

**Table 4.2: DiagnosticDataIdentifier**

The response to this situation on the *AUTOSAR classic platform* has been the definition of the SwcServiceDependency that allows for associating several PortPrototypes in specific roles to a given use-case.

On the other hand, there is a clear difference between the design of PortInterfaces for diagnostic usage on the *AUTOSAR classic platform* (where generic ClientServerInterfaces, SenderReceiverInterfaces, and ModeSwitchInterfaces are used) and the *AUTOSAR adaptive platform* (where the PortInterfaces are specifically tailored to the respective diagnostic use case).

In other words, it specifically tailored PortInterfaces were used, the need for an “umbrella” that defines the scope of the modeling of a given service use case would be drastically reduced.

On top of that, the usage of the InstanceSpecifier as the element of identification is strictly bound to the PortPrototype rather than the SwcServiceDependency. In other words, the diagnostic management uses instanceRefs to identify the “endpoint” in the application software.

The instanceRef and the corresponding InstanceSpecifier are strongly related to each other. If one is known the other can be deduced. The consequence of this conclusion is that it would be possible to configure the diagnostic management in the presence of mappings to the application software model that itself does not have to be accessible at the time of creating the configuration of diagnostic management.

This means that an instanceRef originating from a subclass of DiagnosticSwMapping needs to have an additional decoration in the form of the stereotype atpUriDef.

Please note that the mapping targets[[13]](#footnote-13) within a set of diagnostic mappings may exist in several instances at run-time.

This kind of multiple instantiation is formalized by the existence of meta-class Process (which in turn is represented by meta-class ProcessDesign on design level), see chapter 3.19.

It is very typical that different instances of a piece of application software could require a different diagnostic mapping and the modeling needs to accommodate to this requirement, i.e. a relation between a diagnostic mapping and the ProcessDesign needs to be established.

**[constr\_10002]**{DRAFT} **Only one mapping per PortPrototype** dIf one instance of the following sub-classes of DiagnosticSwMapping refers to a PortPrototype then no other instance of DiagnosticSwMapping shall refer to the same PortPrototype:

* DiagnosticEventPortMapping that is associated with a RPortPrototype typed by a DiagnosticMonitorInterface or a DiagnosticEventInterface.
* DiagnosticOperationCyclePortMapping that is associated with a RPortPrototype typed by a DiagnosticOperationCycleInterface.
* DiagnosticEnableConditionPortMapping that is associated with a RPortPrototype typed by a DiagnosticConditionInterface.
* DiagnosticClearConditionPortMapping that is associated with a RPortPrototype typed by a DiagnosticConditionInterface.
* DiagnosticIndicatorPortMapping that is associated with a RPortPrototype typed by a DiagnosticIndicatorInterface.
* DiagnosticMemoryDestinationPortMapping that is associated with an RPortPrototype typed by a DiagnosticDTCInformationInterface.
* DiagnosticSecurityLevelPortMapping that is associated with an PPortPrototype typed by a DiagnosticSecurityLevelInterface.
* DiagnosticDataPortMapping that is associated with a PPortPrototype typed by a DiagnosticDataIdentifierInterface.
* DiagnosticSecurityLevelPortMapping that is associated with a PPortPrototype typed by a DiagnosticSecurityLevelInterface.
* DiagnosticServiceValidationMapping that is associated with a PPortPrototype typed by a DiagnosticServiceValidationInterface.

### c()

The rationale for the existence of [constr\_10002] is that the respective PortPrototype has a clearly defined functionality. For example, it can only provide the content of one DID, but it cannot provide the content of an arbitrary number of DIDs.

For such a case, the DiagnosticServiceGenericMapping (see section 4.1.12) shall be applied.

Please note the [constr\_10002] does not apply to the DiagnosticServiceGenericMapping, i.e. a PortPrototype that is not subject to [constr\_10002] can be referenced by multiple DiagnosticServiceGenericMapping.

In other words, the ability for several DiagnosticServiceGenericMappings to refer to the same PortPrototype is what makes the DiagnosticServiceGenericMapping *generic*.

In addition, [constr\_10002] is intentionally not applied for PPortPrototypes typed by DiagnosticDataElementInterface because the operations executed in such context are stateless and can therefore be called from several clients without restrictions.

**4.1.2 Diagnostic Monitor to Port Mapping**

**[TPS\_MANI\_01351]**{DRAFT} **Reporting the status of a DiagnosticEvent on the *AUTOSAR adaptive platform*** dFor the purpose of reporting the status of a diagnostic event on the *AUTOSAR adaptive platform*, the relation between a DiagnosticEvent and one RPortPrototypes is created by using the DiagnosticMonitorPortMapping that refers to

* a DiagnosticEvent in the role diagnosticEvent,
* a ProcessDesign in the role process, and
* an RPortPrototype typed by a DiagnosticMonitorInterface in the role rPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype will use at run-time.

c*(RS\_MANI\_00005, RS\_MANI\_00061)*

The use case that supports the reporting of a diagnostic event is depicted in Figure 4.2.

**[constr\_10047]**{DRAFT} **Restriction for the applicability of DiagnosticMonitorPortMapping** dIf an RPortPrototype is referenced by a DiagnosticMonitorPortMapping, then the RPortPrototype shall be typed by a DiagnosticMonitorInterface.c*()*

**[constr\_10048]**{DRAFT} **Existence of reference from DiagnosticMonitorPortMapping to DiagnosticEvent** dEach DiagnosticEvent shall only be referenced by exactly one DiagnosticMonitorPortMapping.c*()*

DiagnosticEvent

*AbstractExecutionContext*

Process

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

*ARElement*

ProcessDesign

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

*AbstractRequiredPortPrototype*

RPortPrototype

DiagnosticMonitorInterface

DiagnosticMonitorPortMapping

+

diagnosticEvent

0..1

«instanceRef,atpUriDef»

+

rPortPrototypeInExecutable

0..1

«isOfType»

requiredInterface

+

0..1

}

{

redefines atpType

+

design

0..1

«atpSplitable»

+

process

0..1

**Figure 4.2: Modeling of DiagnosticMonitorPortMapping for reporting the status of diagnostic events on the *AUTOSAR adaptive platform***

|  |  |
| --- | --- |
| ***Class*** | **DiagnosticMonitorPortMapping** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticMonitorPortMapping** | | | |
| ***Note*** | Defines to which SWC service port the Diagnostic Monitor is mapped.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*,  *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| diagnosticEvent | DiagnosticEvent | 0..1 | ref | Reference to the DiagnosticEvent that is assigned to SWC service ports.  **Tags:**atp.Status=draft |
| process | ProcessDesign | 0..1 | ref | Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=process atp.Status=draft |
| rPortPrototype InExecutable | RPortPrototype | 0..1 | iref | This aggregation allows for the usage of the Diagnostic MonitorPortMapping on the AUTOSAR adaptive platform.  **Stereotypes:** atpUriDef  **Tags:**atp.Status=draft  **InstanceRef implemented by:**RPortPrototypeIn  ExecutableInstanceRef |

**Table 4.3: DiagnosticMonitorPortMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticEvent** | | | |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticEvent | | | |
| ***Note*** | This element is used to configure DiagnosticEvents.  **Tags:**atp.recommendedPackage=DiagnosticEvents | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| associated Event  Identification | PositiveInteger | 0..1 | attr | This attribute represents the identification number that is associated with the enclosing DiagnosticEvent and allows to identify it when placed into a snapshot record or extended data record storage.  This value can be reported as internal data element in snapshot records or extended data records. |
| clearEvent Allowed  Behavior | DiagnosticClearEvent  AllowedBehaviorEnum | 0..1 | attr | This attribute defines the resulting UDS status byte for the related event, which shall not be cleared according to the ClearEventAllowed callback |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticEvent** | | | |
| confirmation Threshold | PositiveInteger | 0..1 | attr | This attribute defines the number of operation cycles with a failed result before a confirmed DTC is set to 1. The semantic of this attribute is a by "1" increased value compared to the confirmation threshold of the "trip counter" mentioned in ISO 14229-1 in figure D.4. A value of "1" defines the immediate confirmation of the DTC along with the first reported failed. This is also sometimes called "zero trip DTC". A value of "2" defines a DTC confirmation in the operation cycle after the first occurred failed. A value of "2" is typically used in the US for OBD DTC confirmation.  **Stereotypes:** atpVariation  **Tags:**vh.latestBindingTime=preCompileTime |
| connected Indicator | DiagnosticConnected  Indicator | \* | aggr | Event specific description of Indicators.  **Stereotypes:** atpSplitable; atpVariation **Tags:**  atp.Splitkey=connectedIndicator.shortName, connected Indicator.variationPoint.shortLabel vh.latestBindingTime=postBuild |
| eventClear Allowed | DiagnosticEventClear  AllowedEnum | 0..1 | attr | This attribute defines whether the Dem has access to a "ClearEventAllowed" callback. |
| prestorage  FreezeFrame | Boolean | 0..1 | attr | This attribute describes whether the Prestorage of Freeze Frames is supported by the assigned event or not.  True: Prestorage of FreezeFrames is supported  False: Prestorage of FreezeFrames is not supported |
| prestored Freezeframe  StoredInNvm | Boolean | 0..1 | attr | If the Event uses a prestored freeze-frame (using the operations PrestoreFreezeFrame and ClearPrestored FreezeFrame of the service interface DiagnosticMonitor) this attribute indicates if the Event requires the data to be stored in non-volatile memory. TRUE = Dem shall store the prestored data in non-volatile memory, FALSE = Data can be lost at shutdown (not stored in Nvm) |
| recoverableIn  SameOperation  Cycle | Boolean | 0..1 | attr | If the attribute is set to true then reporting PASSED will reset the indication of a failed test in the current operation cycle. If the attribute is set to false then reporting  PASSED will be ignored and not lead to a reset of the indication of a failed test. |

**Table 4.4: DiagnosticEvent**

**4.1.3 Diagnostic Event to Port Mapping**

**[TPS\_MANI\_01048]**{DRAFT} **Retrieving the status of a DiagnosticEvent to PortPrototype(s) on the *AUTOSAR adaptive platform*** dFor the purpose of reporting the status of a diagnostic event on the *AUTOSAR adaptive platform*, the relation between a DiagnosticEvent and one RPortPrototypes is created by using the

DiagnosticEventPortMapping that refers to

* a DiagnosticEvent in the role diagnosticEvent,
* a ProcessDesign in the role process, and
* an RPortPrototype typed by a DiagnosticEventInterface in the role rPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype will use at run-time.

c*(RS\_MANI\_00005, RS\_MANI\_00061)*

**[constr\_10049]**{DRAFT} **Restriction for the applicability of DiagnosticEvent-**

**PortMapping** dIf an RPortPrototype is referenced by a DiagnosticEventPortMapping, then the RPortPrototype shall be typed by a DiagnosticEventInterface.c*()*

The use case that supports the retrieval of information about a diagnostic event is depicted in Figure 4.3.

DiagnosticEventPortMapping

DiagnosticEvent

*AbstractExecutionContext*

Process

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

*ARElement*

ProcessDesign

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

*AbstractRequiredPortPrototype*

RPortPrototype

DiagnosticEventInterface

+

design

0..1

«isOfType»

requiredInterface

+

0..1

}

redefines atpType

{

«instanceRef,atpUriDef»

+

rPortPrototypeInExecutable

0..1

+

diagnosticEvent

0..1

«atpSplitable»

+

process

0..1

**Figure 4.3: Modeling of DiagnosticEventPortMapping to retrieve information about a diagnostic event on the *AUTOSAR adaptive platform***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticEventPortMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping | | | |
| ***Note*** | Defines to which SWC service ports with DiagnosticEventInfoNeeds the DiagnosticEvent is mapped.  **Tags:**atp.recommendedPackage=DiagnosticMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*,  *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| diagnosticEvent | DiagnosticEvent | 0..1 | ref | Reference to the DiagnosticEvent that is assigned to SWC service ports with DiagnosticEventInfoNeeds. |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticEventPortMapping** | | | |
| process | ProcessDesign | 0..1 | ref | Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=process atp.Status=draft |
| rPortPrototype InExecutable | RPortPrototype | 0..1 | iref | This aggregation allows for the usage of the Diagnostic EventPortMapping on the AUTOSAR adaptive platform.  **Stereotypes:** atpUriDef  **Tags:**atp.Status=draft  **InstanceRef implemented by:**RPortPrototypeIn  ExecutableInstanceRef |

**Table 4.5: DiagnosticEventPortMapping**

**4.1.4 Diagnostic Operation Cycle to Port Mapping**

**[TPS\_MANI\_01049]**{DRAFT} **Mapping of DiagnosticOperationCycle to PortPrototype(s) on the *AUTOSAR adaptive platform*** dOn the *AUTOSAR adaptive platform*, the relation between a DiagnosticOperationCycle and one RPortPrototype is created by using the DiagnosticOperationCyclePortMapping that refers to

* a DiagnosticOperationCycle in the role operationCycle,
* a ProcessDesign in the role process, and
* an RPortPrototype typed by a DiagnosticOperationCycleInterface in the role rPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype will use at run-time.

c*(RS\_MANI\_00005, RS\_MANI\_00061)*

**[constr\_10050]**{DRAFT} **Restriction for the applicability of DiagnosticOperationCyclePortMapping**dIf an RPortPrototype is referenced by a DiagnosticOperationCyclePortMapping, then the RPortPrototype shall be typed by a DiagnosticOperationCycleInterface.c*()*

**[constr\_10051]**{DRAFT} **Existence of reference from DiagnosticOperationCyclePortMapping to DiagnosticOperationCycle** dEach DiagnosticOperationCycle shall only be referenced by exactly one DiagnosticOperationCyclePortMappingc*()*

DiagnosticOperationCyclePortMapping

DiagnosticOperationCycle

*AbstractExecutionContext*

Process

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

*ARElement*

ProcessDesign

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticOperationCycleInterface

*AbstractRequiredPortPrototype*

RPortPrototype

«atpSplitable»

+

process

0..1

«isOfType»

requiredInterface

+

0..1

}

redefines atpType

{

+

operationCycle

0..1

«instanceRef,atpUriDef»

+

rPortPrototypeInExecutable

0..1

design

+

0..1

**Figure 4.4: Modeling of DiagnosticOperationCyclePortMapping for the usage on the *AUTOSAR adaptive platform***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticOperationCycle** | | | |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticOperationCycle | | | |
| ***Note*** | Definition of an operation cycle that is the base of the event qualifying and for Dem scheduling.  **Tags:**atp.recommendedPackage=DiagnosticOperationCycles | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| cycleStatus Storage | Boolean | 0..1 | attr | Defines if the operation cycle state is available over the power cycle (stored non-volatile) or not.   * true: the operation cycle state is stored non-volatile * false: the operation cycle state is only stored volatile   This attribute is only relevant for the AUTOSAR adaptive platform. It no longer has a meaning on the AUTOSAR classic platform. |
| type | DiagnosticOperation  CycleTypeEnum | 0..1 | attr | Operation cycles types for the Dem. |

**Table 4.6: DiagnosticOperationCycle**

|  |  |
| --- | --- |
| ***Class*** | **DiagnosticOperationCyclePortMapping** |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping |
| ***Note*** | Defines to which SWC service ports with DiagnosticOperationCycleNeeds the DiagnosticOperationCycle is mapped.  **Tags:**atp.recommendedPackage=DiagnosticMappings |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticOperationCyclePortMapping** | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*,  *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| operationCycle | DiagnosticOperation  Cycle | 0..1 | ref | Reference to the DiagnosticOperationCycle that is assigned to SWC service ports with DiagnosticOperation CycleNeeds. |
| process | ProcessDesign | 0..1 | ref | Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=process atp.Status=draft |
| rPortPrototype InExecutable | RPortPrototype | 0..1 | iref | This aggregation allows for the usage of the Diagnostic OperationCyclePortMapping on the AUTOSAR adaptive platform.  **Stereotypes:** atpUriDef  **Tags:**atp.Status=draft  **InstanceRef implemented by:**RPortPrototypeIn  ExecutableInstanceRef |
| swcFlatService Dependency | SwcService  Dependency | 0..1 | ref | Reference to a SwcServiceDependencyType that links ServiceNeeds to SWC service ports. |
| swcService  DependencyIn  System | SwcService  Dependency | 0..1 | iref | Instance reference to a SwcServiceDependency that links ServiceNeeds to SWC service ports.  **InstanceRef implemented by:**SwcServiceDependency InSystemInstanceRef |

**Table 4.7: DiagnosticOperationCyclePortMapping**

**4.1.5 Diagnostic Enable Condition to Port Mapping**

**[TPS\_MANI\_01050]**{DRAFT} **Mapping of DiagnosticEnableCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*** dOn the *AUTOSAR adaptive platform*, the relation between a DiagnosticEnableCondition and one RPortPrototypeis created by using the DiagnosticEnableCondition-

PortMapping that refers to

* a DiagnosticEnableCondition in the role enableCondition,
* a ProcessDesign in the role process, and
* an RPortPrototype typed by a DiagnosticConditionInterface in the role rPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype will use at run-time.

c*(RS\_MANI\_00005, RS\_MANI\_00061)*

**[constr\_10052]**{DRAFT} **Restriction for the applicability of DiagnosticEnable-**

**ConditionPortMapping**dIf an RPortPrototype is referenced by a DiagnosticEnableConditionPortMapping, then the RPortPrototype shall be typed by a DiagnosticConditionInterface.c*()*

**[constr\_10053]**{DRAFT} **Existence of reference from DiagnosticEnableConditionPortMapping to DiagnosticEnableCondition** dEach DiagnosticEnableCondition shall only be referenced by at most one DiagnosticEnableConditionPortMappingc*()*

DiagnosticEnableConditionPortMapping

DiagnosticEnableCondition

*AbstractExecutionContext*

Process

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

*DiagnosticCondition*

*ARElement*

ProcessDesign

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticConditionInterface

*AbstractRequiredPortPrototype*

RPortPrototype

+

design

0..1

«isOfType»

requiredInterface

+

0..1

}

{

redefines atpType

+

enableCondition

0..1

«instanceRef,atpUriDef»

+

rPortPrototypeInExecutable

0..1

«atpSplitable»

process

+

0..1

**Figure 4.5: Modeling of DiagnosticEnableConditionPortMapping for the usage on the *AUTOSAR adaptive platform***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticEnableCondition** | | | |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticCondition | | | |
| ***Note*** | Specification of an enable condition.  **Tags:**atp.recommendedPackage=DiagnosticConditions | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticCondition*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 4.8: DiagnosticEnableCondition**

|  |  |
| --- | --- |
| ***Class*** | **DiagnosticEnableConditionPortMapping** |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticEnableConditionPortMapping** | | | |
| ***Note*** | Defines to which SWC service ports with DiagnosticEnableConditionNeeds the DiagnosticEnable Condition is mapped.  **Tags:**atp.recommendedPackage=DiagnosticMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*,  *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| enableCondition | DiagnosticEnable  Condition | 0..1 | ref | Reference to the EnableCondition which is mapped to a  SWC service port with DiagnosticEnableConditionNeeds. |
| process | ProcessDesign | 0..1 | ref | Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=process atp.Status=draft |
| rPortPrototype InExecutable | RPortPrototype | 0..1 | iref | This aggregation allows for the usage of the Diagnostic EnableConditionPortMapping on the AUTOSAR adaptive platform.  **Stereotypes:** atpUriDef  **Tags:**atp.Status=draft  **InstanceRef implemented by:**RPortPrototypeIn  ExecutableInstanceRef |
| swcFlatService Dependency | SwcService  Dependency | 0..1 | ref | Reference to a SwcServiceDependencyType that links ServiceNeeds to SWC service ports. This reference can be used in early stages of the development in order to identify the SwcServiceDependency without a full System Context. |
| swcService  DependencyIn  System | SwcService  Dependency | 0..1 | iref | Instance reference to a SwcServiceDependency that links ServiceNeeds to SWC service ports.  **InstanceRef implemented by:**SwcServiceDependency InSystemInstanceRef |

**Table 4.9: DiagnosticEnableConditionPortMapping**

**4.1.6 Diagnostic Clear Condition to Port Mapping**

**[TPS\_MANI\_01259]**{DRAFT} **Mapping of DiagnosticClearCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*** dOn the *AUTOSAR adaptive platform*, the relation between a DiagnosticClearCondition and one RPortPrototypeis created by using the DiagnosticClearConditionPortMapping that refers to

* a DiagnosticClearCondition in the role clearCondition,
* a ProcessDesign in the role process, and
* an RPortPrototype typed by a DiagnosticConditionInterface in the role rPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype will use at run-time.

c*(RS\_MANI\_00005, RS\_MANI\_00061)*

**[constr\_10054]**{DRAFT} **Restriction for the applicability of Diagnostic-**

**ClearConditionPortMapping**dIf an RPortPrototype is referenced by a DiagnosticClearConditionPortMapping, then the RPortPrototype shall be typed by a DiagnosticConditionInterface.c*()*

**[constr\_10055]**{DRAFT} **Existence of reference from DiagnosticClearConditionPortMapping to DiagnosticClearCondition** dEach DiagnosticClearCondition shall only be referenced by at most one DiagnosticClearConditionPortMappingc*()*

*AbstractExecutionContext*

Process

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

*DiagnosticCondition*

*ARElement*

ProcessDesign

DiagnosticClearConditionPortMapping

DiagnosticClearCondition

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticConditionInterface

*AbstractRequiredPortPrototype*

RPortPrototype

design

+

0..1

«instanceRef,atpUriDef»

+

rPortPrototypeInExecutable

0..1

clearCondition

+

1

«isOfType»

+

requiredInterface

0..1

}

{

redefines atpType

«atpSplitable»

+

process

0..1

**Figure 4.6: Modeling of DiagnosticClearConditionPortMapping for the usage on the *AUTOSAR adaptive platform***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticClearConditionPortMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***Note*** | Defines to which SWC service ports with DiagnosticsClearConditionNeeds the DiagnosticClearCondition is mapped.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*,  *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| clearCondition | DiagnosticClear  Condition | 1 | ref | Reference to the ClearCondition which is mapped to a SWC service port with DiagnosticClearConditionNeeds.  **Tags:**atp.Status=draft |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticClearConditionPortMapping** | | | |
| process | ProcessDesign | 0..1 | ref | Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=process atp.Status=draft |
| rPortPrototype InExecutable | RPortPrototype | 0..1 | iref | This aggregation allows for the usage of the Diagnostic ClearConditionMapping on the AUTOSAR adaptive platform.  **Stereotypes:** atpUriDef  **Tags:**atp.Status=draft  **InstanceRef implemented by:**RPortPrototypeIn  ExecutableInstanceRef |

**Table 4.10: DiagnosticClearConditionPortMapping**

**4.1.7 Diagnostic Indicator to Port Mapping**

**[TPS\_MANI\_01260]**{DRAFT} **Mapping of DiagnosticIndicator to PortPrototype(s) on the *AUTOSAR adaptive platform*** dOn the *AUTOSAR adaptive platform*, the relation between a DiagnosticIndicator and one RPortPrototypeis created by using the DiagnosticIndicatorPortMapping that refers to

* a DiagnosticIndicator in the role indicator,
* a ProcessDesign in the role process, and
* an RPortPrototype typed by a DiagnosticIndicatorInterface in the role rPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype will use at run-time.

c*(RS\_MANI\_00005, RS\_MANI\_00061)*

**[constr\_10056]**{DRAFT} **Restriction for the applicability of DiagnosticIndicatorPortMapping** dIf an RPortPrototype is referenced by a DiagnosticIndicatorPortMapping, then the RPortPrototype shall be typed by a DiagnosticIndicatorInterface.c*()*

DiagnosticIndicator

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

DiagnosticIndicatorPortMapping

*AbstractExecutionContext*

Process

*ARElement*

ProcessDesign

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticIndicatorInterface

*AbstractRequiredPortPrototype*

RPortPrototype

«instanceRef,atpUriDef»

+

rPortPrototypeInExecutable

0..1

+

indicator

1

+

design

0..1

«isOfType»

requiredInterface

+

0..1

}

{

redefines atpType

«atpSplitable»

process

+

0..1

**Figure 4.7: Modeling of DiagnosticIndicatorPortMapping for the usage on the**

***AUTOSAR adaptive platform***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticIndicatorPortMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***Note*** | Defines to which SWC service ports with DiagnosticsIndicatorNeeds the DiagnosticIndicator is mapped.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*,  *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| indicator | DiagnosticIndicator | 1 | ref | Reference to the DiagnosticIndicator which is mapped to a SWC service port with DiagnosticIndicatorNeeds.  **Tags:**atp.Status=draft |
| process | ProcessDesign | 0..1 | ref | Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=process atp.Status=draft |
| rPortPrototype InExecutable | RPortPrototype | 0..1 | iref | This aggregation allows for the usage of the Diagnostic IndicatorMapping on the AUTOSAR adaptive platform.  **Stereotypes:** atpUriDef  **Tags:**atp.Status=draft  **InstanceRef implemented by:**RPortPrototypeIn  ExecutableInstanceRef |

**Table 4.11: DiagnosticIndicatorPortMapping**

**4.1.8 Diagnostic Memory Destination to Port Mapping**

**[TPS\_MANI\_01261]**{DRAFT} **Mapping of DiagnosticMemoryDestination to PortPrototype(s) on the *AUTOSAR adaptive platform*** dOn the *AUTOSAR adaptive platform*, the relation between a DiagnosticMemoryDestination and one RPortPrototypeis created by using the DiagnosticMemoryDestination-

PortMapping that refers to

* a DiagnosticMemoryDestination in the role memoryDestination,
* a ProcessDesign in the role process, and
* an RPortPrototype typed by a DiagnosticDTCInformationInterface in the role rPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype will use at run-time.

c*(RS\_MANI\_00005, RS\_MANI\_00061)*

**[constr\_10057]**{DRAFT} **Restriction for the applicability of DiagnosticMemory-**

**DestinationPortMapping**dIf an RPortPrototype is referenced by a DiagnosticMemoryDestinationPortMapping, then the RPortPrototype shall be typed by a DiagnosticDTCInformationInterface.c*()*

*DiagnosticMemoryDestination*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticMapping*

*DiagnosticSwMapping*

DiagnosticMemoryDestinationPortMapping

*AbstractExecutionContext*

Process

*ARElement*

ProcessDesign

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticDTCInformationInterface

*AbstractRequiredPortPrototype*

RPortPrototype

«instanceRef,atpUriDef»

+

rPortPrototypeInExecutable

0..1

«atpSplitable»

+

process

0..1

«isOfType»

+

requiredInterface

0..1

{

redefines atpType

}

+

design

0..1

+

memoryDestination

1

**Figure 4.8: Modeling of DiagnosticMemoryDestinationPortMapping for the usage on the *AUTOSAR adaptive platform***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticMemoryDestinationPortMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***Note*** | Defines to which SWC service ports with DiagnosticsEventInfoNeeds the DiagnosticMemoryDestination is mapped.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*,  *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| memory  Destination | DiagnosticMemory  Destination | 1 | ref | Reference to the MemoryDestination which is mapped to a SWC service port with DiagnosticEventInfoNeeds.  **Tags:**atp.Status=draft |
| process | ProcessDesign | 0..1 | ref | Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=process atp.Status=draft |
| rPortPrototype InExecutable | RPortPrototype | 0..1 | iref | This aggregation allows for the usage of the Diagnostic MemoryDestinationMapping on the AUTOSAR adaptive platform.  **Stereotypes:** atpUriDef  **Tags:**atp.Status=draft  **InstanceRef implemented by:**RPortPrototypeIn  ExecutableInstanceRef |

**Table 4.12: DiagnosticMemoryDestinationPortMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***DiagnosticMemoryDestination*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode | | | |
| ***Note*** | This abstract meta-class represents a possible memory destination for a diagnostic event. | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable* | | | |
| ***Subclasses*** | DiagnosticMemoryDestinationPrimary, DiagnosticMemoryDestinationUserDefined | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| agingRequires TestedCycle | Boolean | 0..1 | attr | Defines whether the aging cycle counter is processed every aging cycles or else only tested aging cycle are considered.  If the attribute is set to TRUE: only tested aging cycle are considered for aging cycle counter.  If the attribute is set to FALSE: aging cycle counter is processed every aging cycle. |
| clearDtc  Limitation | DiagnosticClearDtc  LimitationEnum | 0..1 | attr | Defines the scope of the DEM\_ClearDTC Api. |
| dtcStatus  AvailabilityMask | PositiveInteger | 0..1 | attr | Mask for the supported DTC status bits by the Dem. |
| event  Displacement  Strategy | DiagnosticEvent  DisplacementStrategy  Enum | 0..1 | attr | This attribute defines, whether support for event displacement is enabled or not, and which displacement strategy is followed. |
| maxNumberOf  EventEntries | PositiveInteger | 0..1 | attr | This attribute fixes the maximum number of event entries in the fault memory. |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***DiagnosticMemoryDestination*** (abstract) | | |  |
| memoryEntry  StorageTrigger | DiagnosticMemoryEntry  StorageTriggerEnum | 0..1 | attr | Describes the trigger to allocate an event memory entry. |
| statusBit  HandlingTest  FailedSinceLast  Clear | DiagnosticStatusBit  HandlingTestFailed  SinceLastClearEnum | 0..1 | attr | This attribute defines, whether the aging and displacement mechanism shall be applied to the "Test FailedSinceLastClear" status bits. |
| statusBit StorageTest  Failed | Boolean | 0..1 | attr | This parameter is used to activate/deactivate the permanent storage of the "TestFailed" status bits.  true: storage activated false: storage deactivated |
| typeOfFreeze FrameRecord  Numeration | DiagnosticTypeOf  FreezeFrameRecord  NumerationEnum | 0..1 | attr | This attribute defines the type of assigning freeze frame record numbers for event-specific freeze frame records. |

**Table 4.13: DiagnosticMemoryDestination**

**4.1.9 Diagnostic Security to Port Mapping**

**[TPS\_MANI\_01262]**{DRAFT} **Mapping of DiagnosticSecurityLevel to PortPrototype(s) on the *AUTOSAR adaptive platform*** dOn the *AUTOSAR adaptive platform*, the relation between a DiagnosticSecurityLevel and one PPortPrototypeis created by using the DiagnosticSecurityLevelPortMapping that refers to

* a DiagnosticSecurityLevel in the role securityLevel,
* a ProcessDesign in the role process, and
* an PPortPrototype typed by a DiagnosticSecurityLevelInterface in the role pPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype on the application layer will use at run-time.

c*(RS\_MANI\_00005, RS\_MANI\_00061)*

**[constr\_10058]**{DRAFT} **Restriction for the applicability of DiagnosticSecurityLevelPortMapping**dIf a PPortPrototype is referenced by a DiagnosticSecurityLevelPortMapping, then the PPortPrototype shall be typed by a DiagnosticSecurityLevelInterface.c*()*

**[constr\_10059]**{DRAFT} **Existence of reference from DiagnosticSecurityLevelPortMapping to DiagnosticSecurityLevel** dEach DiagnosticSecurityLevel shall only be referenced by exactly one DiagnosticSecurityLevelPortMapping.c*()*

DiagnosticSecurityLevelPortMapping

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

DiagnosticSecurityLevel

*AbstractExecutionContext*

Process

*ARElement*

ProcessDesign

*AbstractProvidedPortPrototype*

PPortPrototype

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticSecurityLevelInterface

+

securityLevel

0..1

«instanceRef,atpUriDef»

+

pPortPrototypeInExecutable

0..1

design

+

0..1

«isOfType»

providedInterface

+

0..1

}

{

redefines atpType

«atpSplitable»

+

process

0..1

**Figure 4.9: Modeling of DiagnosticSecurityLevelPortMapping for the usage on the**

***AUTOSAR adaptive platform***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticSecurityLevelPortMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***Note*** | Defines to which SWC service ports with DiagnosticsCommunicationSecurityNeeds the Diagnostic SecurityLevel is mapped.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*,  *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| pPortPrototype InExecutable | PPortPrototype | 0..1 | iref | This aggregation allows for the usage of the Diagnostic SecurityLevelMapping on the AUTOSAR adaptive platform.  **Stereotypes:** atpUriDef  **Tags:**atp.Status=draft  **InstanceRef implemented by:**PPortPrototypeIn  ExecutableInstanceRef |
| process | ProcessDesign | 0..1 | ref | Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=process atp.Status=draft |
| securityLevel | DiagnosticSecurityLevel | 0..1 | ref | Reference to the SecurityLevelwhich is mapped to a SWC service port with DiagnosticCommunicationSecurity Needs.  **Tags:**atp.Status=draft |

**Table 4.14: DiagnosticSecurityLevelPortMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticSecurityLevel** | | | |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::Dcm | | | |
| ***Note*** | This meta-class represents the ability to define a security level considered for diagnostic purposes.  **Tags:**atp.recommendedPackage=DiagnosticSecurityLevels | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| accessData RecordSize | PositiveInteger | 0..1 | attr | This represents the size of the AccessDataRecord used in GetSeed. Unit:byte. |
| keySize | PositiveInteger | 0..1 | attr | This represents the size of the security key. Unit: byte. |
| numFailed  SecurityAccess | PositiveInteger | 0..1 | attr | This represents the number of failed security accesses after which the delay time is activated. |
| securityDelay Time | TimeValue | 0..1 | attr | This represents the delay time after a failed security access. Unit: second. |
| seedSize | PositiveInteger | 0..1 | attr | This represents the size of the security seed. Unit: byte. |

**Table 4.15: DiagnosticSecurityLevel**

**4.1.10 Diagnostic Data Identifier to Port Mapping**

The DM on the *AUTOSAR adaptive platform* has the ability to access entire DiagnosticDataIdentifiers at once. For supporting this ability, a dedicated mapping class named DiagnosticDataPortMapping is introduced.

**[TPS\_MANI\_01263]**{DRAFT} **Mapping of DiagnosticDataIdentifier or DiagnosticDataElement to PortPrototype(s) on the *AUTOSAR adaptive platform*** dOn the *AUTOSAR adaptive platform*, the relation between a DiagnosticDataIdentifier resp. DiagnosticDataElement and one PPortPrototype is created by using the DiagnosticDataPortMapping that refers to either

* a DiagnosticDataIdentifier in the role diagnosticDataIdentifier or
* a DiagnosticDataElement in the role diagnosticDataElement. c*(RS\_MANI\_00005, RS\_MANI\_00061)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticDataElement** | | | |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics | | | |
| ***Note*** | This meta-class represents the ability to describe a concrete piece of data to be taken into account for diagnostic purposes. | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| arraySize Semantics | ArraySizeSemantics  Enum | 0..1 | attr | This attribute controls the meaning of the value of the array size. |
| maxNumberOf Elements | PositiveInteger | 0..1 | attr | The existence of this attribute turns the data instance into an array of data. The attribute determines the size of the array in terms of how many elements the array can take. |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticDataElement** |  |  |  |
| scalingInfoSize | PositiveInteger | 0..1 | attr | Size in bytes of scaling information for the DiagnosticData  Element if used with DiagnosticReadScalingDataBy  Identifier |
| swDataDef Props | SwDataDefProps | 0..1 | aggr | This property allows to specify data definition properties in order to support the definition of e.g. computation formulae and data constraints. |

**Table 4.16: DiagnosticDataElement**

As depicted in Figure 4.10, DiagnosticDataPortMapping has the ability to handle access to **either** an entire DID **or** to just an element of a DID.

Therefore, the existence of [constr\_10003] is required to enforce that just one of the references is actually used for any given DiagnosticDataPortMapping.

**[constr\_10003]**{DRAFT} **Restriction for the existence of DiagnosticDataPortMapping.diagnosticDataIdentifier vs. DiagnosticDataPortMapping.diagnosticDataElement** dFor each DiagnosticDataPortMapping, **either** the reference in the role diagnosticDataIdentifier **or** diagnosticDataElement shall exist.c*()*

**[constr\_10060]**{DRAFT}**PortInterface of PPortPrototype referenced by DiagnosticDataPortMapping** dAny particular PPortPrototype that is referenced in the role DiagnosticDataPortMapping.pPortPrototypeInExecutable shall be typed by either of

* DiagnosticDataIdentifierInterface
* DiagnosticDataElementInterface
* DiagnosticDataIdentifierGenericInterface c*()*

**[constr\_10061]**{DRAFT} **Mapping to DiagnosticDataIdentifierInterface,**

**DiagnosticDataElementInterface, or DiagnosticDataIdentifierGenericInterface**dAll PPortPrototypes typed by either

* DiagnosticDataIdentifierInterface
* DiagnosticDataElementInterface
* DiagnosticDataIdentifierGenericInterface

shall **only** be referenced by a DiagnosticDataPortMapping. No other subclass of DiagnosticSwMapping is eligible for this purpose.c*()*

**[TPS\_MANI\_01347]**{DRAFT} **Definition of a DiagnosticDataElement used in the context of a DID obtained from a PPortPrototype typed by a DiagnosticDataElementInterface** dIf a DiagnosticDataElement is aggregated by a DiagnosticParameter that in turn is aggregated by a DiagnosticDataIdentifier and the DiagnosticDataElement is also referenced by a DiagnosticDataPortMapping then the referenced DiagnosticDataElement (by way of SwDataDefProps) shall refer to a SwBaseType with attribute baseTypeSize set to either

* 8,
* 16, or
* 32

and attribute baseTypeEncoding set to either

* NONE (for 8, 16, or 32 bit) or
* 2C (for 8, 16, or 32 bit) or
* IEEE754 (for 32 bit).

In this case it is only possible to define the DiagnosticDataElement as a scalar or an array (see [TPS\_DEXT\_01001], [TPS\_DEXT\_01002]). The ability to define a

Variable-Size Array shall only be used for the last element of the DID.c*(RS\_MANI\_00061)*

*DiagnosticMapping*

*DiagnosticSwMapping*

*ARElement*

*DiagnosticCommonElement*

DiagnosticDataPortMapping

DiagnosticDataIdentifier

*AbstractExecutionContext*

Process

*ARElement*

ProcessDesign

*DiagnosticAbstractDataIdentifier*

*AbstractProvidedPortPrototype*

PPortPrototype

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticDataIdentifierInterface

*DiagnosticAbstractDataIdentifierInterface*

*Identifiable*

DiagnosticDataElement

DiagnosticDataElementInterface

«instanceRef,atpUriDef»

pPortPrototypeInExecutable

+

0..1

+

design

0..1

diagnosticDataElement

+

0..1

«atpSplitable»

+

process

«isOfType»

providedInterface

+

0..1

{

}

redefines atpType

+

diagnosticDataIdentifier

0..1

**Figure 4.10: Modeling of DiagnosticDataPortMapping for the usage on the *AUTOSAR adaptive platform***

**[TPS\_MANI\_01348]**{DRAFT} **Definition of a DiagnosticDataElement used in the context of a DID obtained from a PPortPrototype typed by a DiagnosticDataIdentifierInterface**dIf a DiagnosticDataIdentifier is referenced by a DiagnosticDataPortMapping then the aggregated DiagnosticDataIdentifier.dataElement.dataElement (by way of SwDataDefProps) shall refer to a

SwBaseType with attribute baseTypeSize set to either

* 8,
* 16, or
* 32

and attribute baseTypeEncoding set to either

* NONE (for 8, 16, or 32 bit) or
* 2C (for 8, 16, or 32 bit) or
* IEEE754 (for 32 bit).

In this case it is only possible to define the DiagnosticDataElement as a scalar or an array (see [TPS\_DEXT\_01001], [TPS\_DEXT\_01002]). The ability to define a

Variable-Size Array shall only be used for the last element of the DID.c*(RS\_-*

*MANI\_00061)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticDataPortMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***Note*** | This meta-class provides the ability to define a diagnostic access to an entire DID.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticServiceMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*,  *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| diagnosticData Element | DiagnosticDataElement | 0..1 | ref | This reference represents the applicable DiagnosticData Element.  **Tags:**atp.Status=draft |
| diagnosticData Identifier | DiagnosticDataIdentifier | 0..1 | ref | This reference represents the applicable DiagnosticData  Identfiier.  **Tags:**atp.Status=draft |
| pPortPrototype InExecutable | PPortPrototype | 0..1 | iref | This reference identifies the applicable PPortPrototype from which that data is obtained. The reference has the ability to point into the component hierarchy (under possible consideration of the rootSoftwareComposition).  **Stereotypes:** atpUriDef  **Tags:**atp.Status=draft  **InstanceRef implemented by:**PPortPrototypeIn  ExecutableInstanceRef |
| process | ProcessDesign | 1 | ref | Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=process atp.Status=draft |

**Table 4.17: DiagnosticDataPortMapping**

**4.1.11 Diagnostic Generic UDS Service Handler to Port Mapping**

It is possible to associate a collection of UDS services to a given PPortPrototype with the intention that the PPortPrototype can handle the associated services.

By creating a dedicated association between generic UDS handlers and the services they can take it is possible to use multiple generic UDS handlers and let each take only the associated services.

Technically, a possible alternative to the documented modeling of generic UDS handling would be to avoid the mapping at all and foresee the existence of a catch-all generic UDS handler.

This, to a large extent, contradicts the idea of having modular software installations on the basis of the definition of SoftwareClusters (see section 14.2).

**4.1.12 Diagnostic Generic Mapping**

**[TPS\_MANI\_01326]**{DRAFT} **Generic Mapping to a DiagnosticServiceInstance on the AUTOSAR Adaptive Platform** dOn the AUTOSAR adaptive platform, the relation between a DiagnosticServiceInstance and one PPortPrototype for select diagnostic services is created by using the meta-class DiagnosticServiceGenericMapping that refers to

* a DiagnosticServiceInstance in the role diagnosticServiceInstance,
* the PPortPrototype in the role pPortPrototypeInExecutable, and
* a ProcessDesign in the role process.

c*(RS\_MANI\_00005, RS\_MANI\_00061)*

**[constr\_10062]**{DRAFT}**DiagnosticServiceInstances that can be mapped by a DiagnosticServiceGenericMapping** dDiagnosticServiceGenericMapping shall only be used for the following list of DiagnosticServiceInstances:

* DiagnosticEcuReset
* DiagnosticComControl
* DiagnosticRoutineControl
* DiagnosticCustomServiceInstance
* DiagnosticRequestUpload
* DiagnosticRequestDownload c*()*

*DiagnosticServiceInstance*

*DiagnosticSwMapping*

DiagnosticServiceGenericMapping

*ARElement*

ProcessDesign

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*AbstractExecutionContext*

Process

*AbstractProvidedPortPrototype*

PPortPrototype

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

+

design

0..1

«instanceRef,atpUriDef»

pPortPrototypeInExecutable

+

0..1

«isOfType»

+

providedInterface

0..1

{

redefines atpType

}

diagnosticServiceInstance

+

0..1

«atpSplitable»

+

process

0..1

**Figure 4.11: Modeling of the diagnostic generic mapping**

Please note that services DiagnosticDataTransfer and DiagnosticTransferExit are mapped dynamically according to the UDS request sequence for upload and download.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticEcuReset** | | | |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EcuReset | | | |
| ***Note*** | This represents an instance of the "ECU Reset" diagnostic service.  **Tags:**atp.recommendedPackage=DiagnosticEcuResets | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| customSub  Function  Number | PositiveInteger | 0..1 | attr | This attribute shall be used to define a custom sub-function number if none of the standardized values of category shall be used. |
| ecuResetClass | DiagnosticEcuReset  Class | 0..1 | ref | This reference substantiates that abstract reference in the role serviceClass for this specific concrete class.  Thereby, the reference represents the ability to access shared attributes among all DiagnosticEcuReset in the given context. |

**Table 4.18: DiagnosticEcuReset**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticComControl** | | | |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommunicationControl | | | |
| ***Note*** | This represents an instance of the "Communication Control" diagnostic service.  **Tags:**atp.recommendedPackage=DiagnosticCommunicationControls | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticComControl** |  |  |  |
| comControl Class | DiagnosticComControl  Class | 0..1 | ref | This reference substantiates that abstract reference in the role serviceClass for this specific concrete class.  Thereby, the reference represents the ability to access shared attributes among all DiagnosticComControl in the given context. |
| customSub  Function  Number | PositiveInteger | 0..1 | attr | This attribute shall be used to define a custom sub-function number if none of the standardized values of category shall be used. |

**Table 4.19: DiagnosticComControl**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticRoutineControl** | | | |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::RoutineControl | | | |
| ***Note*** | This represents an instance of the "Routine Control" diagnostic service.  **Tags:**atp.recommendedPackage=DiagnosticRoutineControls | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| routine | DiagnosticRoutine | 1 | ref | This refers to the applicable DiagnosticRoutine. |
| routineControl Class | DiagnosticRoutine  ControlClass | 0..1 | ref | This reference substantiates that abstract reference in the role serviceClass for this specific concrete class.  Thereby, the reference represents the ability to access shared attributes among all DiagnosticRoutineControl in the given context. |

**Table 4.20: DiagnosticRoutineControl**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticCustomServiceInstance** | | | |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CustomServiceInstance | | | |
| ***Note*** | This meta-class has the ability to define an instance of a custom diagnostic service.  **Tags:**atp.recommendedPackage=DiagnosticCustomInstances | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| customService Class | DiagnosticCustom  ServiceClass | 0..1 | ref | Reference to the corresponding DiagnosticCustom ServiceClass. |

**Table 4.21: DiagnosticCustomServiceInstance**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticRequestUpload** | | | |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::MemoryByAddress | | | |
| ***Note*** | This represents an instance of the "Request Upload" diagnostic service.  **Tags:**atp.recommendedPackage=DiagnosticMemoryByAdresss | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMemory*  *AddressableRangeAccess*, *DiagnosticMemoryByAddress*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticRequestUpload** | |  |  |
| requestUpload Class | DiagnosticRequest  UploadClass | 0..1 | ref | This reference substantiates that abstract reference in the role serviceClass for this specific concrete class.  Thereby, the reference represents the ability to access shared attributes among all DiagnosticRequestUpload in the given context. |

**Table 4.22: DiagnosticRequestUpload**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticRequestDownload** | | | |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::MemoryByAddress | | | |
| ***Note*** | This represents an instance of the "Request Download" diagnostic service.  **Tags:**atp.recommendedPackage=DiagnosticMemoryByAdresss | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMemory*  *AddressableRangeAccess*, *DiagnosticMemoryByAddress*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| request  DownloadClass | DiagnosticRequest  DownloadClass | 0..1 | ref | This reference substantiates that abstract reference in the role serviceClass for this specific concrete class.  Thereby, the reference represents the ability to access shared attributes among all DiagnosticRequestDownload in the given context. |

**Table 4.23: DiagnosticRequestDownload**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticServiceGenericMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***Note*** | This meta-class represents the ability to implement a generic generic mapping for select diagnostics services on the adaptive platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticServiceMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*,  *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| diagnostic  ServiceInstance | DiagnosticService  Instance | 0..1 | ref | Reference to the ServiceInstance mapped to a SWC service port.  **Tags:**atp.Status=draft |
| pPortPrototype InExecutable | PPortPrototype | 0..1 | iref | This aggregation allows for the usage of the Diagnostic ServiceGenericMapping on the AUTOSAR adaptive platform.  **Stereotypes:** atpUriDef  **Tags:**atp.Status=draft  **InstanceRef implemented by:**PPortPrototypeIn  ExecutableInstanceRef |
| process | ProcessDesign | 0..1 | ref | Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=process atp.Status=draft |

**Table 4.24: DiagnosticServiceGenericMapping**

**4.1.13 Diagnostic Service Validation Mapping**

**[TPS\_MANI\_01352]**{DRAFT} **Definition of DiagnosticServiceValidation-**

**Mapping** dOn the *AUTOSAR adaptive platform*, meta-class DiagnosticServiceValidationMapping exists to indicate that the referenced PPortPrototype is used for the execution of a manufacturer-specific or supplier-specific check prior to the execution of a diagnostic service.

The number of PPortPrototypes that implement such a checking routine is arbitrary and the order in which the individual checks are executed is subject to a decision made on deployment level.

This is the main reason why the DiagnosticServiceValidationMapping (beside the reference to the ProcessDesign) has no reference into another design object in the context of the DEXT.c*(RS\_MANI\_00005, RS\_MANI\_00061)*

DiagnosticServiceValidationMapping

*AbstractProvidedPortPrototype*

PPortPrototype

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

*AbstractExecutionContext*

Process

functionClusterAffiliation: String

+

[0..1]

[0..1]

numberOfRestartAttempts: PositiveInteger

+

+

preMapping: Boolean

[0..1]

*ARElement*

ProcessDesign

*DiagnosticMapping*

*DiagnosticSwMapping*

DiagnosticServiceValidationInterface

«isOfType»

providedInterface

+

0..1

}

redefines atpType

{

«atpSplitable»

+

process

0..1

«instanceRef,atpUriDef»

+

pPortPrototypeInExecutable

0..1

design

+

0..1

**Figure 4.12: Modeling of the diagnostic validation mapping**

**[constr\_10063]**{DRAFT} **Possible values for DiagnosticServiceValidationMapping.category**dThe value of attribute DiagnosticServiceValidationMapping.category is restricted to the following values:

**MANUFACTURER\_VALIDATION** The enclosing DiagnosticServiceValidationMapping represents a validation defined by the manufacturer.

**SUPPLIER\_VALIDATION** The enclosing DiagnosticServiceValidationMapping represents a validation defined by the supplier.

### c()

**[constr\_10064]**{DRAFT} **Existence of DiagnosticServiceValidationMapping.pPortPrototypeInExecutable**dA PPortPrototype referenced in the role

DiagnosticServiceValidationMapping.pPortPrototypeInExecutable shall be typed by a DiagnosticServiceValidationInterface.c*()*

Please note that the final decision about the order in which the validations are executed is decided in the deployment phase, see section 14.2.2.4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticServiceValidationMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***Note*** | This meta-class provides the ability to specify manufacturer/supplier checks to be executed before diagnostic services can be processed.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*,  *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| pPortPrototype InExecutable | PPortPrototype | 0..1 | iref | This mapping identifies a PortPrototype typed by a DiagnosticValidationInterface in which a manufacturer/ supplier-specific check is executed.  **Stereotypes:** atpUriDef  **Tags:**atp.Status=draft  **InstanceRef implemented by:**PPortPrototypeIn  ExecutableInstanceRef |
| process | ProcessDesign | 0..1 | ref | Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=process atp.Status=draft |

**Table 4.25: DiagnosticServiceValidationMapping**

**4.1.14 Diagnostic Authentication Mapping**

**[TPS\_MANI\_01360]**{DRAFT} **Creation of two diagnostic mappings the fulfill different roles in the context of authenticating a diagnostic client** dThe support for the diagnostic service Authentication (0x29) on the *AUTOSAR adaptive platform* requires the creation of two diagnostic mappings the fulfill different roles in the context of authenticating a diagnostic client:

* The meta-class DiagnosticAuthenticationPortMapping (described in this section) is used to configure the forwarding of the request for authentication from the DM to some application software that acts an authentication manager.
* The meta-class DiagnosticExternalAuthenticationPortMapping is used to configure the transmission of the response of the authentication manager software back to the DM. c*(RS\_MANI\_00005, RS\_MANI\_00061)*

Please note that meta-class DiagnosticExternalAuthenticationPortMapping is described in section 4.1.15.

The two different roles mentioned in [TPS\_MANI\_01360] for the authentication of a diagnostic client are depicted in Figure 4.13.



R

Application Software

Component

Diagnostic Manager

DiagnosticAuthenti

-

cationInterface

P



P

R

DiagnosticExternal

-

AuthenticationInterface

Authentication Request

Authentication Confirm/Deny

DiagnosticAuthenti

-

cationMapping

DiagnosticExternalAuthenti

-

cationMapping

**Figure 4.13: Example data exchange for diagnostic authentication**

**[TPS\_MANI\_01361]**{DRAFT} **Support the authentication request of a diagnostic client** dFor the purpose of supporting the authentication request of a diagnostic client on the *AUTOSAR adaptive platform*, the relation between a DiagnosticAuthentication and one RPortPrototype is created by using the DiagnosticAuthenticationPortMapping that refers to

* a DiagnosticAuthentication in the role diagnosticAuthentication,
* a ProcessDesign in the role process, and
* a PPortPrototype typed by a DiagnosticAuthenticationInterface in the role pPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype will use at run-time.

c*(RS\_MANI\_00005, RS\_MANI\_00061)*

**[constr\_10092]**{DRAFT} **Restriction for the applicability of DiagnosticAuthenticationPortMapping** dIf a PPortPrototype is referenced by a DiagnosticAuthenticationPortMapping, then the PPortPrototype shall be typed by a DiagnosticAuthenticationInterfacec*()*

**[constr\_10093]**{DRAFT} **Existence of reference from DiagnosticAuthenticationPortMapping to DiagnosticAuthentication** dEach DiagnosticAuthentication shall only be referenced by exactly one DiagnosticAuthenticationPortMapping.c*()*

*AbstractExecutionContext*

Process

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

*ARElement*

ProcessDesign

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticExternalAuthenticationInterface

DiagnosticAuthenticationPortMapping

*DiagnosticAuthentication*

*DiagnosticServiceInstance*

*AbstractProvidedPortPrototype*

PPortPrototype

+

diagnosticAuthentication

0..1

«instanceRef,atpUriDef»

+

pPortPrototypeInExecutable

0..1

«isOfType»

providedInterface

+

0..1

}

redefines atpType

{

«atpSplitable»

+

process

0..1

+

design

0..1

**Figure 4.14: Modeling of DiagnosticAuthenticationPortMapping to forward the authentication request of a diagnostic client to an authentication manager**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticAuthenticationPortMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***Note*** | This mapping class identifies the PortPrototype in the application software that handles the client authentication.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortMapppings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*,  *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| diagnostic  Authentication | Diagnostic  Authentication | 0..1 | ref | Reference to the DiagnosticAuthentication that is assigned to a SWC service port.  **Tags:**atp.Status=draft |
| pPortPrototype InExecutable | PPortPrototype | 0..1 | iref | This aggregation allows for the usage of the Diagnostic AuthenticationPortMapping on the AUTOSAR adaptive platform.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**PPortPrototypeIn  ExecutableInstanceRef |
| process | ProcessDesign | 0..1 | ref | Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=process atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticAuthenticationPortMapping** | | |  |
| rPortPrototype InExecutable | RPortPrototype | 0..1 | iref | This aggregation allows for the usage of the Diagnostic AuthenticationPortMapping on the AUTOSAR adaptive platform.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**RPortPrototypeIn  ExecutableInstanceRef |

**Table 4.26: DiagnosticAuthenticationPortMapping**

**4.1.15 Diagnostic External Authentication Mapping**

**[TPS\_MANI\_01362]**{DRAFT} **Support the responding to an authentication request of a diagnostic client** dFor the purpose of supporting the transmission of a response to the authentication request of a diagnostic client on the *AUTOSAR adaptive platform*, the relation between a DiagnosticAuthentication and one RPortPrototype is created by using the DiagnosticExternalAuthenticationPortMapping that refers to

* a DiagnosticAuthentication in the role diagnosticAuthentication,
* a ProcessDesign in the role process, and
* an RPortPrototype typed by a DiagnosticExternalAuthenticationInterface in the role rPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype will use at run-time.

c*(RS\_MANI\_00005, RS\_MANI\_00061)*

**[constr\_10094]**{DRAFT} **Restriction for the applicability of DiagnosticExternalAuthenticationPortMapping**dIf an RPortPrototype is referenced by a DiagnosticExternalAuthenticationPortMapping, then the RPortPrototype shall be typed by a DiagnosticExternalAuthenticationInterfacec*()*

**[constr\_10095]**{DRAFT} **Existence of reference from DiagnosticExternalAuthenticationPortMapping to DiagnosticAuthentication** dEach DiagnosticAuthentication shall only be referenced by exactly one DiagnosticExternalAuthenticationPortMapping.c*()*

*AbstractExecutionContext*

Process

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

*ARElement*

ProcessDesign

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

*AbstractRequiredPortPrototype*

RPortPrototype

DiagnosticExternalAuthenticationPortMapping

DiagnosticExternalAuthenticationInterface

*DiagnosticAuthentication*

*DiagnosticServiceInstance*

+

diagnosticAuthentication

0..1

design

+

0..1

«instanceRef,atpUriDef»

+

rPortPrototypeInExecutable

0..1

«isOfType»

requiredInterface

+

0..1

redefines atpType

{

}

«atpSplitable»

+

process

0..1

**Figure 4.15: Modeling of DiagnosticExternalAuthenticationPortMapping to transmit the response to an authentication request back to the DM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticExternalAuthenticationPortMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***Note*** | This mapping class identifies the PortPrototype in the application software that handles the external authentication.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticPortMapppings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*,  *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| diagnostic  Authentication | Diagnostic  Authentication | 0..1 | ref | Reference to the DiagnosticAuthentication that is assigned to a SWC service port.  **Tags:**atp.Status=draft |
| process | ProcessDesign | 0..1 | ref | Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=process atp.Status=draft |
| rPortPrototype InExecutable | RPortPrototype | 0..1 | iref | This aggregation allows for the usage of the Diagnostic ClientAuthenticationPortMapping on the AUTOSAR adaptive platform.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**RPortPrototypeIn  ExecutableInstanceRef |

**Table 4.27: DiagnosticExternalAuthenticationPortMapping**

## 4.2 Diagnostic Clear Condition

On the *AUTOSAR adaptive platform*, a new model element similar DiagnosticEnableCondition is introduced: DiagnosticClearCondition.

*DiagnosticTroubleCode*

DiagnosticTroubleCodeUds

*DiagnosticCommonElement*

DiagnosticEvent

*DiagnosticMapping*

DiagnosticEventToTroubleCodeUdsMapping

*DiagnosticConditionGroup*

DiagnosticClearConditionGroup

*DiagnosticMapping*

DiagnosticTroubleCodeUdsToClearConditionGroupMapping

*DiagnosticCondition*

DiagnosticClearCondition

+

troubleCodeUds

0..1

«atpSplitable»

+

clearCondition

0..\*

+

clearConditionGroup

0..1

diagnosticEvent

+

0..1

+

troubleCodeUds

0..1

**Figure 4.16: Modeling of the diagnostic clear condition**

In contrast to DiagnosticEnableCondition, DiagnosticClearCondition is not mapped to a DiagnosticEvent but (via the aggregation by DiagnosticClearConditionGroup) to a DiagnosticTroubleCodeUds.

For this purpose, meta-class DiagnosticTroubleCodeUdsToClearConditionGroupMapping has been defined.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticClearCondition** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition | | | |
| ***Note*** | This meta-class describes a clear condition for diagnostic purposes.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticConditions | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticCondition*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 4.28: DiagnosticClearCondition**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticClearConditionGroup** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition | | | |
| ***Note*** | Clear condition group which includes one or several clear conditions.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticClearConditionGroups | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticConditionGroup*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| clearCondition | DiagnosticClear  Condition | \* | ref | This aggregation represents the collection of Diagnostic ClearConditions that belong to the DiagnosticClear ConditionGroup.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=clearCondition atp.Status=draft |

**Table 4.29: DiagnosticClearConditionGroup**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticTroubleCodeUdsToClearConditionGroupMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition | | | |
| ***Note*** | This meta-class provides the ability to map a DiagnosticClearConditionGroup to a collection of Diagnostic TroubleCodeUds.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DiagnosticMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| clearCondition Group | DiagnosticClear  ConditionGroup | 0..1 | ref | Thi reference identifies the applicable DiagnosticClear ConditionGroup.  **Tags:**atp.Status=draft |
| troubleCodeUds | DiagnosticTroubleCode  Uds | 0..1 | ref | This reference identifies the DiagnosticTroubleCodeUds that are relevant for the mapping.  **Tags:**atp.Status=draft |

**Table 4.30: DiagnosticTroubleCodeUdsToClearConditionGroupMapping**

**[constr\_1658]**{DRAFT} **Number of DiagnosticTroubleCodeUdsToClearConditionGroupMapping elements per DiagnosticTroubleCodeUds** dThe mapping element DiagnosticTroubleCodeUdsToClearConditionGroupMapping shall be created no more than once per DiagnosticTroubleCodeUds.

If several DiagnosticTroubleCodeUdsToClearConditionGroupMapping elements referring to the same DiagnosticTroubleCodeUds are defined, then the Clear Condition Group mapping shall be regarded as defective.c*()*

## 4.3 Security Access

the implementation of the diagnostics manager on the adaptive platform requires a refined modeling of meta-class DiagnosticSecurityAccessClass.

A new attribute named sharedTimer is introduced that controls whether a single timer is used for all security access levels or whether the individual levels utilize separate timers respectively.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticSecurityAccessClass** | | | |
| ***Package*** | M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::SecurityAccess | | | |
| ***Note*** | This meta-class contains attributes shared by all instances of the "Security Access" diagnostic service.  **Tags:**atp.recommendedPackage=DiagnosticSecurityAccesss | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticSecurityAccessClass** | |  |  |
| sharedTimer | Boolean | 0..1 | attr | Switch between separate or single shared timer instance and timer value.   * True: use shared timer instance and timer value for all security access levels combined. * False: use separate timer instance and timer values for each security level.   **Tags:**atp.Status=draft |

**Table 4.31: DiagnosticSecurityAccessClass**

*DiagnosticCommonElement*

*DiagnosticServiceInstance*

DiagnosticSecurityAccess

[0..1]

requestSeedId: PositiveInteger

+

[0..1]

securityDelayTimeOnBoot: TimeValue

+

*DiagnosticCommonElement*

DiagnosticSecurityLevel

accessDataRecordSize: PositiveInteger

+

[0..1]

+

[0..1]

keySize: PositiveInteger

[0..1]

numFailedSecurityAccess: PositiveInteger

+

[0..1]

securityDelayTime: TimeValue

+

+

seedSize: PositiveInteger

[0..1]

*DiagnosticCommonElement*

DiagnosticAccessPermission

*DiagnosticCommonElement*

*DiagnosticServiceClass*

DiagnosticSecurityAccessClass

+

sharedTimer: Boolean

[0..1]

*DiagnosticCommonElement*

DiagnosticAuthRole

+

bitPosition: PositiveInteger

[0..1]

+

[0..1]

isDefault: Boolean

authenticationRole

+

0..\*

+

securityLevel

0..\*

+

accessPermission

0..1

«atpAbstract»

+

serviceClass

0..1

«atpSplitable»

securityLevel

+

0..1

+

securityAccessClass

0..1

}

redefines serviceClass

{

**Figure 4.17: Refined modeling of the diagnostic security access**

## 4.4 DiagnosticProvidedDataMapping

**[TPS\_MANI\_01230]**{DRAFT} **Semantics of DiagnosticProvidedDataMapping** dThe meta-class DiagnosticProvidedDataMapping does not seem to fulfill the condition for representing a mapping class because it only has one reference to a DiagnosticDataElement in the role dataElement.

However, the specific nature of this mapping is that the second element (the DiagnosticProvidedDataMapping.dataProvider) that is supposed to take place in the mapping cannot precisely be modeled as a single meta-class.

Therefore, there is no better way than to model the DiagnosticProvidedDataMapping.dataProvider by a NameToken.c*(RS\_MANI\_00005)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DiagnosticProvidedDataMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticProvidedDataMapping | | | |
| ***Note*** | This represents the ability to define the nature of a data access for a DiagnosticDataElement based on a data provider that cannot be modeled explicitly.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DataMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| dataElement | DiagnosticDataElement | 0..1 | ref | This represents the DiagnosticDataElement for which the access is further qualified by the DiagnosticProvidedData Mapping.dataProvider.  **Tags:**atp.Status=draft |
| dataProvider | NameToken | 1 | attr | This represents the ability to further specify the data provider.  **Tags:**atp.Status=draft |

**Table 4.32: DiagnosticProvidedDataMapping**

Please note that the list of standardized values of attribute DiagnosticProvidedDataMapping.dataProvider is defined in the SWS Diagnostics [20].

*DiagnosticMapping*

DiagnosticProvidedDataMapping

+

dataProvider: NameToken

*Identifiable*

DiagnosticDataElement

+

arraySizeSemantics: ArraySizeSemanticsEnum

[0..1]

+

maxNumberOfElements: PositiveInteger

[0..1]

+

scalingInfoSize: PositiveInteger

[0..1]

+

dataElement

0..1

**Figure 4.18: Modeling of DiagnosticProvidedDataMapping**

## 4.5 Diagnostic Miscellanea

**[TPS\_MANI\_01358]**{DRAFT} **Restriction for the configuration of diagnostic debouncing** dThe restriction for the configuration of the diagnostic debouncing, as formulated in [TPS\_DEXT\_03004], shall **not** apply on *the AUTOSAR adaptive platform*.c *(RS\_MANI\_00005)*

# 5 System Design

## 5.1 Overview

A typical vehicle will most likely be equipped with ECUs developed on the AUTOSAR classic platform and Machines developed on the AUTOSAR adaptive platform. The system design for the entire vehicle has therefore to cover both platform deployment types.

AUTOSAR supports System design descriptions with the possibility to describe Software Components of both AUTOSAR Platforms that will be used in a System and even allows to indicate the service oriented communication between them if possible.

Especially when it comes to the description of the communication behavior of AUTOSAR classic and adaptive ECUs in a harmonized way, the notion of a System Design becomes a special focus point.

All the system design aspects have in common is that they have to cope with both AUTOSAR classic ECUs and adaptive Machines. The basic design aspects of such inter-disciplinary systems have to be already available in the AUTOSAR classic modeling approach otherwise they would not be available to both platforms.

Thus, it is straight forward to take the existing meta-class System as the starting point for the modeling of such mixed systems.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **System** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate | | | |
| ***Note*** | The top level element of the System Description.  **Tags:**atp.recommendedPackage=Systems | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| fibexElement | FibexElement | \* | ref | Reference to ASAM FIBEX elements specifying Communication and Topology.  All Fibex Elements used within a System Description shall be referenced from the System Element.  atpVariation: In order to describe a product-line, all Fibex Elements can be optional.  **Stereotypes:** atpVariation  **Tags:**vh.latestBindingTime=postBuild |
| interpolation Routine  MappingSet | InterpolationRoutine  MappingSet | \* | ref | This reference identifies the InterpolationRoutineMapping Sets that are relevant in the context of the enclosing System. |
| mapping | SystemMapping | \* | aggr | Aggregation of all mapping aspects relevant in the System Description.  **Stereotypes:** atpSplitable; atpVariation **Tags:**  atp.Splitkey=mapping.shortName, mapping.variation Point.shortLabel  vh.latestBindingTime=postBuild |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **System** |  |  |  |
| pncVector Length | PositiveInteger | 0..1 | attr | Length of the partial networking request release information vector (in bytes). |
| pncVectorOffset | PositiveInteger | 0..1 | attr | Absolute offset (with respect to the NM-PDU) of the partial networking request release information vector that is defined in bytes as an index starting with 0. |
| rootSoftware Composition | RootSwComposition  Prototype | 0..1 | aggr | Aggregation of the root software composition, containing all software components in the System in a hierarchical structure. This element is not required when the System description is used for a network-only use-case.  atpVariation: The RootSwCompositionPrototype can vary.  **Stereotypes:** atpSplitable; atpVariation **Tags:**  atp.Splitkey=rootSoftwareComposition.shortName, root SoftwareComposition.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime |
| systemVersion | RevisionLabelString | 1 | attr | Version number of the System Description. |

**Table 5.1: System**

There are use cases to exchange parts of such a SYSTEM\_DESIGN\_DESCRIPTION in different methodology stages and between different development parties, therefore further system categories are supported by AUTOSAR.

A common approach is for example that the OEM provides a basis for designing an ECU, which is later advanced by the supplier. Therefore Classic AUTOSAR supports

System categories like ECU\_EXTRACT or ECU\_SYSTEM\_DESCRIPTION that have only a single ECU in scope. Classic Platform System.categorys are defined in [TPS\_SYST\_01003].

The AUTOSAR adaptive platform is using the same approach. If an OEM wants to provide design artifacts that are relevant for the configuration of a single Machine all unnecessary information is stripped from the System with categorySYSTEM\_DESIGN\_DESCRIPTION and a definition of the subsystem is provided.

**[TPS\_MANI\_01274]**{DRAFT} **Standardized System Category Definitions for the**

**Adaptive Platform** dThe standardized System category definitions are defined in Table 5.2c*()*

|  |  |
| --- | --- |
| Category | Meaning |
| SYSTEM\_DESIGN\_DESCRIPTION | The System description that contains design artifacts that are relevant for the Adaptive Platform and Classic Platform. |
| MACHINE\_DESIGN\_EXTRACT | The System description that contains design artifacts that are relevant for a single Adaptive Platform Machine. |

**Table 5.2: Standardized System Category definitions**

**[constr\_3421]**{DRAFT} **Fibex elements applicable for a System of categoryMA-**

**CHINE\_DESIGN\_EXTRACT** dA System with the categoryMACHINE\_DESIGN\_EXTRACT is allowed to reference the following fibexElements:

* CommunicationCluster
* MachineDesign
* GlobalTimeDomain
* NmConfig
* SystemMapping that is allowed to contain only a PncMapping

c*()*

## 5.2 Specification of Communication System Structure

When the communication interaction is designed for a vehicle system the focus is put on the network and the connected ECUs. Whether a specific ECU connected to the network is implemented using AUTOSAR classic or AUTOSAR adaptive does not influence the major communication design.

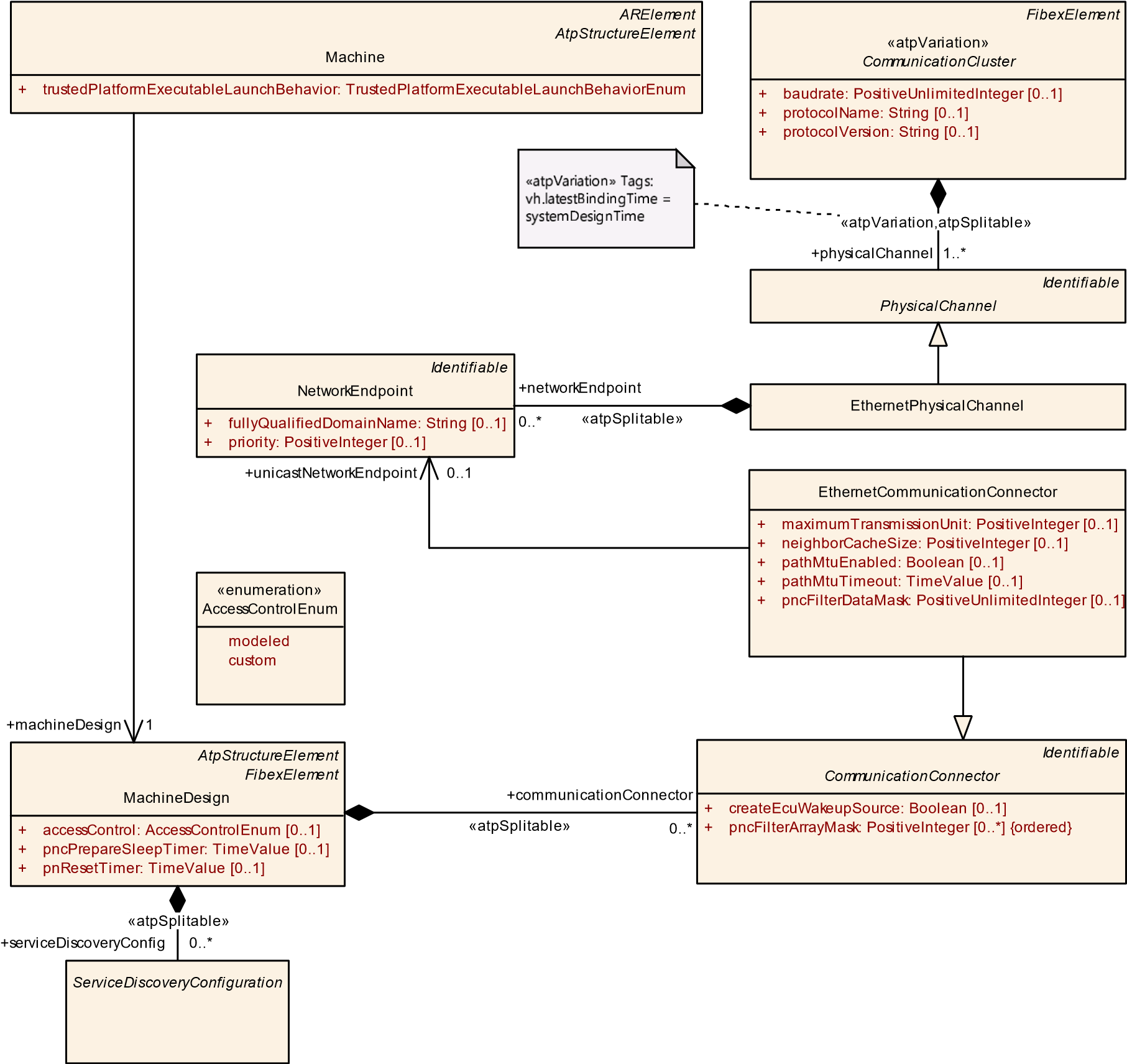
But of course, it is essential from a car manufacturer point of view whether a specific ECU will be implemented using AUTOSAR classic or adaptive. Thus, already on system design level there is a need to specify the AUTOSAR Platform kind which shall be used to implement an ECU.

In AUTOSAR classic the element EcuInstance is used to define one ECU in the system design.

In AUTOSAR adaptive the element Machine is an entity which already represents a specific ECU Implementation with dedicated configurations for e.g. Processors, ProcessorCores.

The Machine is a model entity which is not in the focus of communication designers and should not be used during system design.

Therefore, the MachineDesign has been introduced in order to allow the communication system designer to define a placeholder for an adaptive ECU in the scope of the System (the MachineDesign corresponds to the EcuInstance of AUTOSAR classic).



**Figure 5.1: MachineDesign**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **MachineDesign** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign | | | |
| ***Note*** | This meta-class represents the ability to define requirements on a Machine in the context of designing a system.  **Tags:**  atp.Status=draft  atp.recommendedPackage=MachineDesigns | | | |
| ***Base*** | *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| accessControl | AccessControlEnum | 0..1 | attr | This attribute defines how the access restriction to the Service Instance is defined.  **Tags:**atp.Status=draft |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **MachineDesign** | | | |
| communication Connector | Communication  Connector | \* | aggr | This aggregation defines the network connection of the machine.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=communicationConnector.shortName, communicationConnector.variationPoint.shortLabel atp.Status=draft |
| ethIpProps | EthIpProps | 0..1 | ref | Maschine specific IP attributes.  **Tags:**atp.Status=draft |
| pncPrepare SleepTimer | TimeValue | 0..1 | attr | Time in seconds the PNC state machine shall wait in PNC\_PREPARE\_SLEEP.  **Tags:**atp.Status=draft |
| pnResetTimer | TimeValue | 0..1 | attr | Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests.  **Tags:**atp.Status=draft |
| service Discovery  Config | ServiceDiscovery  Configuration | \* | aggr | Set of service discovery configuration settings that are defined on the machine for individual Communication Connectors.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=serviceDiscoveryConfig atp.Status=draft |
| tcpIpIcmpProps | EthTcpIpIcmpProps | 0..1 | ref | Machine specific ICMP (Internet Control Message Protocol) attributes  **Tags:**atp.Status=draft |
| tcpIpProps | EthTcpIpProps | 0..1 | ref | Machine specific TcpIp Stack attributes.  **Tags:**atp.Status=draft |

**Table 5.3: MachineDesign**

**[TPS\_MANI\_03209]**{DRAFT} **The meaning of MachineDesign.accessControl** dThe MachineDesign.accessControl defines whether the access control is defined by AUTOSAR means in the Application Design with receiverIntent (see [TPS\_MANI\_01106]) and senderIntent (see [TPS\_MANI\_01107]) or by a custom lists that are created by a non-AUTOSAR process.c*(RS\_MANI\_00034)*

|  |  |
| --- | --- |
| ***Enumeration*** | **AccessControlEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment |
| ***Note*** | This enumeration describes the options for the definition of access restriction to resources.  **Tags:**atp.Status=draft |
| ***Literal*** | ***Description*** |
| custom | The access restriction to the resource is defined by a non-AUTOSAR process.  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |

5

**Table 5.4: AccessControlEnum**

**5.2.1 Network connection**

One of the most prominent information defined in the context of the MachineDesign is the network connectivity. Since the *AUTOSAR adaptive platform* focuses on the usage of Ethernet for communication, this boils down to the specification of IP addresses.

Specifically, the basic definition of the connectivity of a MachineDesign is created by aggregating the abstract base-class CommunicationConnector in the role communicationConnector. The specific subclass of CommunicationConnector that is used in this context is the EthernetCommunicationConnector.

|  |  |
| --- | --- |
| ***Enumeration*** | **AccessControlEnum** |
| modeled | The access restriction to the resource is modeled in the AUTOSAR Application Design model or the AUTOSAR Deployment model.  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |

The EthernetCommunicationConnector is used to connect the MachineDesign with a VLAN that is represented in AUTOSAR by a EthernetPhysicalChannel that is part of an EthernetCluster.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***PhysicalChannel*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology | | | |
| ***Note*** | This element represents a physical connection (in case of CAN, FlexRay, LIN) or a logical connection (VLAN in case of Ethernet) between communicating devices. | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Subclasses*** | *AbstractCanPhysicalChannel*, EthernetPhysicalChannel, FlexrayPhysicalChannel, UserDefinedPhysical  Channel | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 5.5: PhysicalChannel**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | <<atpVariation>> **EthernetCluster** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | Ethernet-specific cluster attributes.  **Tags:**atp.recommendedPackage=CommunicationClusters | | | |
| ***Base*** | *ARObject*, *CollectableElement*, *CommunicationCluster*, *FibexElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | <<atpVariation>> **EthernetCluster** | | | |
| couplingPort Connection | CouplingPort  Connection | \* | aggr | Specification of connections between CouplingElements and EcuInstances.  Note: This atpSplitable property has no atp.Splitkey due to atpVariation (PropertySetPattern).  **Stereotypes:** atpSplitable; atpVariation  **Tags:**vh.latestBindingTime=postBuild |
| couplingPort  StartupActive  Time | TimeValue | 0..1 | attr | The attribute specifies the time in second a coupling port is switched on to enable the host ECU (ECU that maintains an Ethernet switch) to listen to the network for potential network management requests. |
| couplingPort  SwitchoffDelay | TimeValue | 0..1 | attr | Switch off delay for CouplingPorts in seconds. It denotes the delay of switching off couplingPorts after the request to switch off a couplingPort was issued. (e.g. switch off of Ethernet switch ports). |
| macMulticast Group | MacMulticastGroup | \* | aggr | MacMulticastGroup that is defined for the Subnet (EthernetCluster). |

**Table 5.6: EthernetCluster**

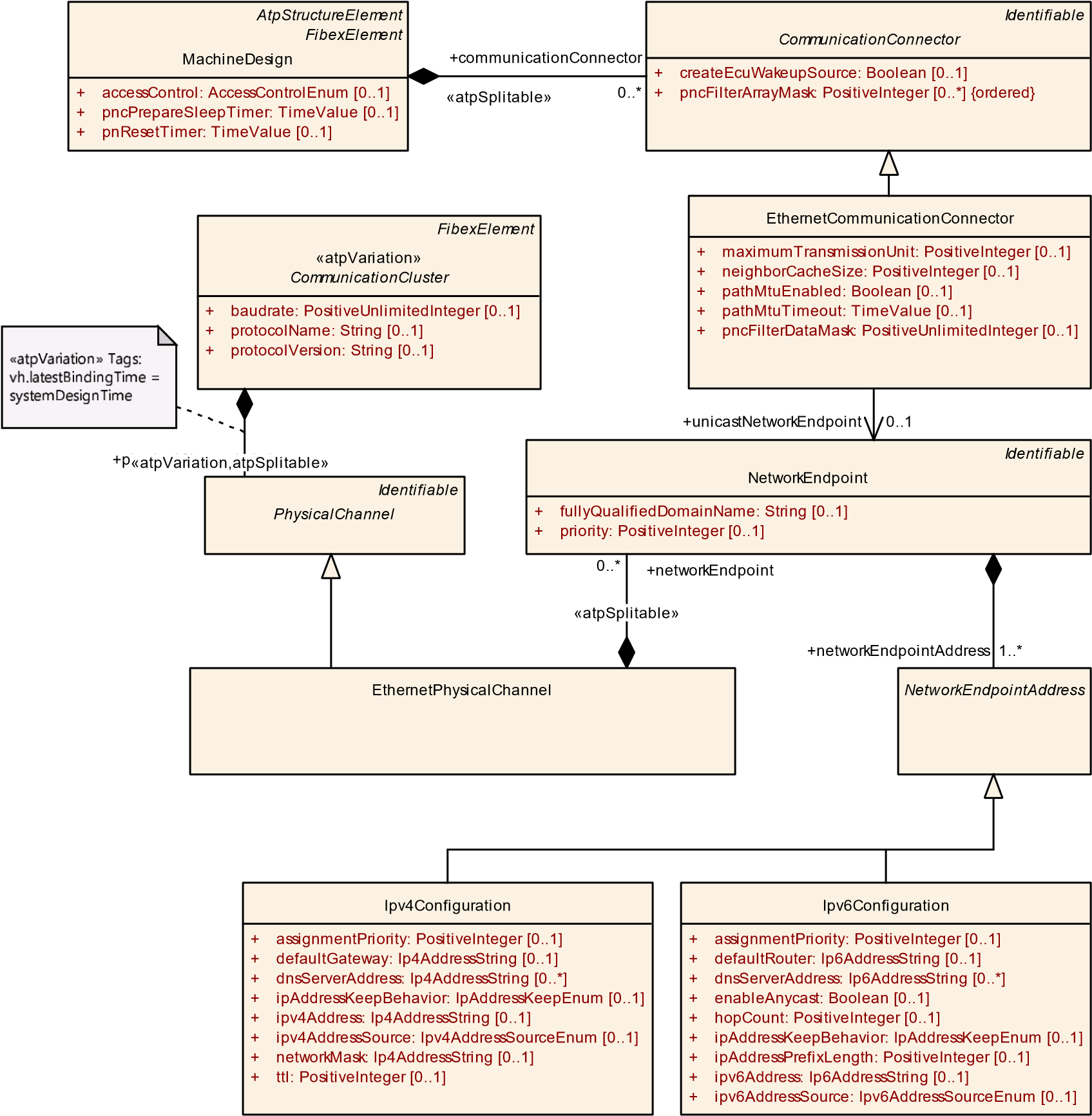
**[constr\_3320]**{DRAFT} **Aggregation of CommunicationConnector by MachineDesign** dMeta-Class MachineDesign shall only aggregate EthernetCommunicationConnectors in the role communicationConnector. No other subclass of CommunicationConnector shall appear in this aggregation.c*()*

The canonical way to specify an IP address is the modeling of a NetworkEndpoint, referenced from an EthernetCommunicationConnector that is aggregated by MachineDesign in the role communicationConnector.

In addition to the IP address, the NetworkEndpoint may have a *Fully Qualified Domain Name* and a priority.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **NetworkEndpoint** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | The network endpoint defines the network addressing (e.g. IP-Address or MAC multicast address). | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| fullyQualified  DomainName | String | 0..1 | attr | Defines the fully qualified domain name (FQDN) e.g.  some.example.host. |
| ipSecConfig | IPSecConfig | 0..1 | aggr | Optional IPSec configuration that provides security services for IP packets. |
| network Endpoint Address | NetworkEndpoint  Address | 1..\* | aggr | Definition of a Network Address.  **Tags:**xml.name  Plural=NETWORK-ENDPOINT-ADDRESSES |
| priority | PositiveInteger | 0..1 | attr | Defines the frame priority where values from 0 (best effort) to 7 (highest) are allowed. |

**Table 5.7: NetworkEndpoint**



**Figure 5.2: Network connection of a MachineDesign**

More precisely, the particular IP address is configured by means of the aggregation of

Ipv4Configuration or Ipv6Configuration in the role networkEndpointAddress.

The NetworkEndpoint is aggregated by the EthernetPhysicalChannel that in turn is aggregated by the EthernetCluster.

**[TPS\_MANI\_03052]**{DRAFT} **Static IPv4 configuration** dIf the value of attribute ipv4AddressSource of meta-class Ipv4Configuration is set to Ipv4AddressSourceEnum.fixed then the ipv4Address defines the static IPv4

Address.c*(RS\_MANI\_00018)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Ipv4Configuration** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | Internet Protocol version 4 (IPv4) configuration. | | | |
| ***Base*** | *ARObject*, *NetworkEndpointAddress* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| assignment  Priority | PositiveInteger | 0..1 | attr | Priority of assignment (1 is highest). If a new address from an assignment method with a higher priority is available, it overwrites the IP address previously assigned by an assignment method with a lower priority. |
| defaultGateway | Ip4AddressString | 0..1 | attr | IP address of the default gateway. |
| dnsServer Address | Ip4AddressString | \* | attr | IP addresses of preconfigured DNS servers.  **Tags:**xml.namePlural=DNS-SERVER-ADDRESSES |
| ipAddressKeep Behavior | IpAddressKeepEnum | 0..1 | attr | Defines the lifetime of a dynamically fetched IP address. |
| ipv4Address | Ip4AddressString | 0..1 | attr | IPv4 Address. Notation: 255.255.255.255. The IP Address shall be declared in case the ipv4AddressSource is FIXED and thus no auto-configuration mechanism is used. |
| ipv4Address Source | Ipv4AddressSource  Enum | 0..1 | attr | Defines how the node obtains its IP address. |
| networkMask | Ip4AddressString | 0..1 | attr | Network mask. Notation 255.255.255.255 |
| ttl | PositiveInteger | 0..1 | attr | Lifespan of data (0..255). The purpose of the TimeToLive field is to avoid a situation in which an undeliverable datagram keeps circulating on a system. |

**Table 5.8: Ipv4Configuration**

|  |  |
| --- | --- |
| ***Enumeration*** | **Ipv4AddressSourceEnum** |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology |
| ***Note*** | Defines how the node obtains its IPv4-Address. |
| ***Literal*** | ***Description*** |
| autoIp | AutoIP is used to dynamically assign IP addresses at device startup.  **Tags:**atp.EnumerationLiteralIndex=0 |
| autoIp\_doip | Linklocal IPv4 Address Assignment using DoIP Parameters  **Tags:**atp.EnumerationLiteralIndex=2 |
| dhcpv4 | DHCP is a service for the automatic IP configuration of a client.  **Tags:**atp.EnumerationLiteralIndex=3 |
| fixed | The IP Address shall be declared manually.  **Tags:**atp.EnumerationLiteralIndex=4 |

**Table 5.9: Ipv4AddressSourceEnum**

**[TPS\_MANI\_03053]**{DRAFT} **Static IPv6 configuration** dIf the value of attribute ipv6AddressSource of meta-class Ipv6Configuration is set to Ipv6AddressSourceEnum.fixed then the ipv6Address defines the static IPv6

Address.c*(RS\_MANI\_00018)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Ipv6Configuration** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | Internet Protocol version 6 (IPv6) configuration. | | | |
| ***Base*** | *ARObject*, *NetworkEndpointAddress* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| assignment  Priority | PositiveInteger | 0..1 | attr | Priority of assignment (1 is highest). If a new address from an assignment method with a higher priority is available, it overwrites the IP address previously assigned by an assignment method with a lower priority. |
| defaultRouter | Ip6AddressString | 0..1 | attr | IP address of the default router. |
| dnsServer Address | Ip6AddressString | \* | attr | IP addresses of pre configured DNS servers.  **Tags:**xml.namePlural=DNS-SERVER-ADDRESSES |
| enableAnycast | Boolean | 0..1 | attr | This attribute is used to enable anycast addressing (i.e. to one of multiple receivers). |
| hopCount | PositiveInteger | 0..1 | attr | The distance between two hosts. The hop count n means that n gateways separate the source host from the destination host (Range 0..255) |
| ipAddressKeep Behavior | IpAddressKeepEnum | 0..1 | attr | Defines the lifetime of a dynamically fetched IP address. |
| ipAddressPrefix Length | PositiveInteger | 0..1 | attr | IPv6 prefix length defines the part of the IPv6 address that is the network prefix. |
| ipv6Address | Ip6AddressString | 0..1 | attr | IPv6 Address. Notation: FFFF:...:FFFF. The IP Address shall be declared in case the ipv6AddressSource is FIXED and thus no auto-configuration mechanism is used. |
| ipv6Address Source | Ipv6AddressSource  Enum | 0..1 | attr | Defines how the node obtains its IP address. |

**Table 5.10: Ipv6Configuration**

|  |  |
| --- | --- |
| ***Enumeration*** | **Ipv6AddressSourceEnum** |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology |
| ***Note*** | Defines how the node obtains its IPv6-Address. |
| ***Literal*** | ***Description*** |
| dhcpv6 | DHCP is a service for the automatic IP configuration of a client.  **Tags:**atp.EnumerationLiteralIndex=0 |
| fixed | The IP Address shall be declared manually.  **Tags:**atp.EnumerationLiteralIndex=1 |
| linkLocal | LinkLocal is intended only for communications within the segment of a local network (a link) or a point-to-point connection that a host is connected to.  **Tags:**atp.EnumerationLiteralIndex=2 |
| linkLocal\_doip | Linklocal IPv6 Address Assignment using DoIP Parameters  **Tags:**atp.EnumerationLiteralIndex=3 |
| router  Advertisement | IPv6 Stateless Autoconfiguration.  **Tags:**atp.EnumerationLiteralIndex=4 |

**Table 5.11: Ipv6AddressSourceEnum**

|  |  |
| --- | --- |
| ***Enumeration*** | **IpAddressKeepEnum** |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology |
| ***Note*** | Defines the behavior after a dynamic IP address has been assigned. |
| ***Literal*** | ***Description*** |
| forget | After a dynamic IP address has been assigned just use it for this session.  **Tags:**atp.EnumerationLiteralIndex=0 |
| storePersistently | After a dynamic IP address has been assigned store the address persistently.  **Tags:**atp.EnumerationLiteralIndex=1 |

**Table 5.12: IpAddressKeepEnum**

Please note that there is also the possibility to describe a MacMulticastConfiguration as NetworkEndpointAddress in addition to Ipv4Configuration and Ipv6Configuration in a system topology description. This may be useful for description of endpoints that are not based on IP, e.g. for streaming protocols like AVB (Audio Video Bridging). But please note that there is no foundation software or ara::com support for such MacMulticastConfigurationNetworkEndpoints in Adaptive Autosar. For SOME/IP communication such NetworkEndpoints are excluded by [constr\_3288].

**5.2.1.1 Support of 10BASE-T1S Network Topologies**

Please note that 10BASE-T1S network topology description is supported in the System

Design with a CouplingPortConnection that points with the nodePort reference to CouplingPorts that represent the 10Base-T1S PHYs connected to the network.

More details about the modeling of 10BASE-T1S networks can be found in the System Template [17]. Since the same modeling approach is used in the Classic Platform and the Adaptive Platform the detailed description of the meta-model available in the System Template is not repeated in this specification.

**5.2.2 TcpIp stack configuration properties**

The MachineDesign references the following elements and allows to set Machine specific TcpIp stack configuration options in the System Design:

* EthIpProps - used to configure IPv4 and IPv6
* EthTcpIpProps - used to configure TCP and UDP
* EthTcpIpIcmpProps - used to configure ICMP

Please note that the System Template [17] defines constraints for the usage of EthIpProps, EthTcpIpProps, EthTcpIpIcmpProps. These constraints are also valid if the MachineDesign references these elements.

**5.2.2.1 IP configuration properties**

Ipv4Props

Ipv6Props

*ARElement*

EthIpProps

Ipv4ArpProps

[0..1]

tcpIpArpNumGratuitousArpOnStartup: PositiveInteger

+

tcpIpArpPacketQueueEnabled: Boolean

[0..1]

+

tcpIpArpRequestTimeout: TimeValue

+

[0..1]

tcpIpArpTableEntryTimeout: TimeValue

[0..1]

+

Ipv4AutoIpProps

tcpIpAutoIpInitTimeout: TimeValue

+

[0..1]

Ipv4FragmentationProps

[0..1]

tcpIpIpFragmentationRxEnabled: Boolean

+

tcpIpIpNumFragments: PositiveInteger

[0..1]

+

tcpIpIpNumReassDgrams: PositiveInteger

+

[0..1]

+

tcpIpIpReassTimeout: TimeValue

[0..1]

Dhcpv6Props

+

tcpIpDhcpV6CnfDelayMax: TimeValue

[0..1]

+

tcpIpDhcpV6CnfDelayMin: TimeValue

[0..1]

[0..1]

tcpIpDhcpV6InfDelayMax: TimeValue

+

+

tcpIpDhcpV6InfDelayMin: TimeValue

[0..1]

tcpIpDhcpV6SolDelayMax: TimeValue

[0..1]

+

+

[0..1]

tcpIpDhcpV6SolDelayMin: TimeValue

Ipv6FragmentationProps

tcpIpIpReassemblyBufferCount: PositiveInteger

+

[0..1]

tcpIpIpReassemblyBufferSize: PositiveInteger

[0..1]

+

[0..1]

+

tcpIpIpReassemblySegmentCount: PositiveInteger

+

tcpIpIpReassemblyTimeout: TimeValue

[0..1]

+

tcpIpIpTxFragmentBufferCount: PositiveInteger

[0..1]

tcpIpIpTxFragmentBufferSize: PositiveInteger

+

[0..1]

Ipv6NdpProps

tcpIpNdpDefaultReachableTime: TimeValue

[0..1]

+

tcpIpNdpDefaultRetransTimer: TimeValue

+

+

[0..1]

tcpIpNdpDefaultRouterListSize: PositiveInteger

[0..1]

tcpIpNdpDefensiveProcessing: Boolean

+

tcpIpNdpDelayFirstProbeTime: PositiveInteger

[0..1]

+

[0..1]

tcpIpNdpDestinationCacheSize: PositiveInteger

+

tcpIpNdpDynamicHopLimitEnabled: Boolean

[0..1]

+

[0..1]

tcpIpNdpDynamicMtuEnabled: Boolean

+

tcpIpNdpDynamicReachableTimeEnabled: Boolean

+

[0..1]

[0..1]

+

tcpIpNdpDynamicRetransTimeEnabled: Boolean

+

tcpIpNdpMaxRandomFactor: PositiveInteger

[0..1]

tcpIpNdpMaxRtrSolicitationDelay: TimeValue

+

[0..1]

+

tcpIpNdpMaxRtrSolicitations: PositiveInteger

[0..1]

+

tcpIpNdpMinRandomFactor: PositiveInteger

[0..1]

+

tcpIpNdpNeighborUnreachabilityDetectionEnabled: Boolean

[0..1]

tcpIpNdpNumMulticastSolicitations: PositiveInteger

[0..1]

+

tcpIpNdpNumUnicastSolicitations: PositiveInteger

[0..1]

+

+

tcpIpNdpPacketQueueEnabled: Boolean

[0..1]

+

tcpIpNdpPrefixListSize: PositiveInteger

[0..1]

+

tcpIpNdpRandomReachableTimeEnabled: Boolean

[0..1]

[0..1]

tcpIpNdpRndRtrSolicitationDelayEnabled: Boolean

+

[0..1]

tcpIpNdpRtrSolicitationInterval: TimeValue

+

+

[0..1]

tcpIpNdpSlaacDadNumberOfTransmissions: PositiveInteger

tcpIpNdpSlaacDadRetransmissionDelay: TimeValue

[0..1]

+

[0..1]

tcpIpNdpSlaacDelayEnabled: Boolean

+

[0..1]

tcpIpNdpSlaacOptimisticDadEnabled: Boolean

+

*AtpStructureElement*

*FibexElement*

MachineDesign

+

accessControl: AccessControlEnum

[0..1]

[0..1]

+

pncPrepareSleepTimer: TimeValue

pnResetTimer: TimeValue

[0..1]

+

+

fragmentationProps

0..1

ipv4Props

+

0..1

ethIpProps

+

0..1

+

dhcpProps

0..1

+

fragmentationProps

0..1

+

ipv6Props

0..1

+

ndpProps

0..1

+

arpProps

0..1

autoIpProps

+

0..1

**Figure 5.3: Machine specific IP configuration options**

|  |  |
| --- | --- |
| ***Class*** | **EthIpProps** |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **EthIpProps** | | | |
| ***Note*** | This meta-class is used to configure the Machine specific IP attributes.  **Tags:**atp.recommendedPackage=EthIpProps | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| ipv4Props | Ipv4Props | 0..1 | aggr | Configuration options for IPv4. |
| ipv6Props | Ipv6Props | 0..1 | aggr | Configuration options for IPv6. |

**Table 5.13: EthIpProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Ipv4Props** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | This meta-class specifies the configuration options for IPv4. | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| arpProps | Ipv4ArpProps | 0..1 | aggr | Configuration properties for the ARP (Address Resolution Protocol). |
| autoIpProps | Ipv4AutoIpProps | 0..1 | aggr | Configuration options for Auto-IP (automatic private IP addressing). |
| fragmentation Props | Ipv4Fragmentation  Props | 0..1 | aggr | Configuration options for IPv4 packet fragmentation/ reassembly. |

**Table 5.14: Ipv4Props**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Ipv4ArpProps** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | Specifies the configuration options for the ARP (Address Resolution Protocol). | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| tcpIpArpNum  GratuitousArp OnStartup | PositiveInteger | 0..1 | attr | This attribute specifies the number of gratuitous ARP replies which shall be sent on assignment of a new IP address. |
| tcpIpArpPacket QueueEnabled | Boolean | 0..1 | attr | This attribute enables (TRUE) or disables (FALSE) support of the ARP Packet Queue according to IETF RFC 1122, section 2.3.2.2. |
| tcpIpArp Request  Timeout | TimeValue | 0..1 | attr | This attribute specifies a timeout in seconds for the validity of ARP requests. After the transmission of an ARP request the TcpIp shall skip the transmission of any further ARP requests to the same destination within a duration of tcpIpArpRequestTimeout seconds. (IETF RFC 1122, section 2.3.2.1). |
| tcpIpArpTable EntryTimeout | TimeValue | 0..1 | attr | This attribute specifies the timeout in seconds after which an unused ARP entry is removed. |

**Table 5.15: Ipv4ArpProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Ipv4AutoIpProps** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | Specifies the configuration options for Auto-IP (automatic private IP addressing). | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| tcpIpAutoIpInit Timeout | TimeValue | 0..1 | attr | This attribute specifies the time in seconds Auto-IP waits at startup, before beginning with ARP probing. This delay is used to give DHCP time to acquire a lease in case a DHCP server is present. |

**Table 5.16: Ipv4AutoIpProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Ipv4FragmentationProps** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | Specifies the configuration options for IPv4 packet fragmentation/reassembly. | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| tcpIpIp  Fragmentation  RxEnabled | Boolean | 0..1 | attr | Enables (TRUE) or disables (FALSE) support for reassembling of incoming datagrams that are fragmented according to IETF RFC 815 (IP Datagram Reassembly  Algorithms). |
| tcpIpIpNum Fragments | PositiveInteger | 0..1 | attr | Specifies the maximum number of IP fragments per datagram. |
| tcpIpIpNum  ReassDgrams | PositiveInteger | 0..1 | attr | Specifies the maximum number of fragmented IP datagrams that can be reassembled in parallel. |
| tcpIpIpReass Timeout | TimeValue | 0..1 | attr | Specifies the timeout in [s] after which an incomplete datagram gets discarded. |

**Table 5.17: Ipv4FragmentationProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Ipv6Props** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | This meta-class specifies the configuration options for IPv6. | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| dhcpProps | Dhcpv6Props | 0..1 | aggr | Configuration properties for DHCPv6. |
| fragmentation Props | Ipv6Fragmentation  Props | 0..1 | aggr | Configuration properties for IPv6 packet fragmentation/ reassembly. |
| ndpProps | Ipv6NdpProps | 0..1 | aggr | Configuration properties for the Neighbor Discovery Protocol for IPv6. |

**Table 5.18: Ipv6Props**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Ipv6FragmentationProps** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | This meta-class specifies the configuration options for IPv6 packet fragmentation/reassembly. | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Ipv6FragmentationProps** | | | |
| tcpIpIp  Reassembly  BufferCount | PositiveInteger | 0..1 | attr | Number of buffers that can be used for fragment reassembly. In case of a reassembly error or if not all fragments are received in time this buffer will be blocked until the specified "Fragment Reassembly Timeout" has been exceeded.  A value of 0 disables fragment reassembly. |
| tcpIpIp  Reassembly  BufferSize | PositiveInteger | 0..1 | attr | Size of each fragment tx buffer in bytes. |
| tcpIpIp  Reassembly  SegmentCount | PositiveInteger | 0..1 | attr | Specifies the maximum number of consecutive data segments that can be managed in each reassembly buffer. If all fragments are received in order, only one segment will be needed.  To deal with fragments received out of order this value should be configured bigger than 1. |
| tcpIpIp Reassembly  Timeout | TimeValue | 0..1 | attr | Specifies the timeout in seconds after which an incomplete datagram gets discarded. |
| tcpIpIpTx  FragmentBuffer  Count | PositiveInteger | 0..1 | attr | These buffers will be used if the IpV6 receives packets from the upper layer that do not fit into the MTU and thus must be fragmented.  A value of 0 disables tx fragmentation. |
| tcpIpIpTx  FragmentBuffer  Size | PositiveInteger | 0..1 | attr | Size of each fragment tx buffer in bytes. |

**Table 5.19: Ipv6FragmentationProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Dhcpv6Props** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | This meta-class specifies the configuration options for DHCPv6. | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| tcpIpDhcp  V6CnfDelayMax | TimeValue | 0..1 | attr | Maximum delay in seconds before sending the first Confirm message. If this value is bigger than the previous minimum delay value a random delay will be chosen from the interval. |
| tcpIpDhcp  V6CnfDelayMin | TimeValue | 0..1 | attr | Minimum delay in seconds before the first Confirm message will be sent. |
| tcpIpDhcpV6Inf DelayMax | TimeValue | 0..1 | attr | Maximum delay in seconds before sending the first Information Request message. If this value is bigger than the previous minimum delay value a random delay will be chosen from the interval. |
| tcpIpDhcpV6Inf DelayMin | TimeValue | 0..1 | attr | Minimum delay (s) before the first Information Request message will be sent. |
| tcpIpDhcpV6Sol DelayMax | TimeValue | 0..1 | attr | Maximum delay in seconds before sending the first Solicit message. If this value is bigger than the previous minimum delay value a random delay will be chosen from the interval. |
| tcpIpDhcpV6Sol DelayMin | TimeValue | 0..1 | attr | Minimum delay (s) before the first Solicit message will be sent. |

**Table 5.20: Dhcpv6Props**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Ipv6NdpProps** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | This meta-class specifies the configuration options for the Neighbor Discovery Protocol for IPv6. | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| tcpIpNdpDefault ReachableTime | TimeValue | 0..1 | attr | Configuration of the ReachableTime (s) specified in [RFC4861 6.3.2. Host Variables]. |
| tcpIpNdpDefault RetransTimer | TimeValue | 1 | attr | Configures the default value (s) for the RetransTimer variable specified in [RFC4861 6.3.2. Host Variables]. |
| tcpIpNdpDefault RouterListSize | PositiveInteger | 0..1 | attr | Maximum number of default router entries. |
| tcpIpNdp Defensive  Processing | Boolean | 0..1 | attr | If enabled the NDP shall only process Neighbor Advertisements which are received in reaction to a previously transmitted Neighbor Solicitation as well as skipping updates to the Neighbor Cache based on received Neighbor Solicitations. If disabled all Neighbor Advertisements and Solicitations shall be processed as specified in RFC4861. |
| tcpIpNdpDelay FirstProbeTime | PositiveInteger | 0..1 | attr | Delay before sending the first NUD probe in (s). |
| tcpIpNdp Destination  CacheSize | PositiveInteger | 0..1 | attr | Maximum number of entries in the destination cache. |
| tcpIpNdp  DynamicHop  LimitEnabled | Boolean | 0..1 | attr | If enabled the default hop limit may be reconfigured based on received Router Advertisements. |
| tcpIpNdp  DynamicMtu  Enabled | Boolean | 0..1 | attr | Allow dynamic reconfiguration of link MTU via Router Advertisements. |
| tcpIpNdp  Dynamic  ReachableTime  Enabled | Boolean | 0..1 | attr | If enabled the default Reachable Time value may be reconfigured based on received Router Advertisements. |
| tcpIpNdp Dynamic  RetransTime  Enabled | Boolean | 0..1 | attr | If enabled the default Retransmit Timer value may be reconfigured based on received Router Advertisements. |
| tcpIpNdpMax  RandomFactor | PositiveInteger | 0..1 | attr | Maximum random factor used for randomization |
| tcpIpNdpMaxRtr  Solicitation  Delay | TimeValue | 0..1 | attr | Maximum delay before the first Router Solicitation will be sent after interface initialization in (s). |
| tcpIpNdpMaxRtr  Solicitations | PositiveInteger | 0..1 | attr | Maximum number of Router Solicitations that will be sent before the first Router Advertisement has been received. |
| tcpIpNdpMin  RandomFactor | PositiveInteger | 0..1 | attr | Minimum random factor used for randomization |
| tcpIpNdp  Neighbor  Unreachability  Detection Enabled | Boolean | 0..1 | attr | Neighbor Unreachability Detection is used to remove unused entries from the neighbor cache. This feature is a basic feature of NDP and should be turned on. |
| tcpIpNdpNum  Multicast  Solicitations | PositiveInteger | 0..1 | attr | Maximum number of multicast solicitations that will be sent when performing address resolution. |
| tcpIpNdpNum  Unicast  Solicitations | PositiveInteger | 0..1 | attr | Maximum number of unicast solicitations that will be sent when performig Neighbor Unreachability Detection. |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Ipv6NdpProps** | | | |
| tcpIpNdpPacket QueueEnabled | Boolean | 0..1 | attr | Enables (TRUE) or disables (FALSE) support of a NDP  Packet Queue according to IETF RFC 4861, section  7.2.2. |
| tcpIpNdpPrefix ListSize | PositiveInteger | 0..1 | attr | Maximum number of entries in the on-link prefix list. |
| tcpIpNdp  Random  ReachableTime  Enabled | Boolean | 0..1 | attr | If enabled the value of ReachableTime will be multiplied with a random value between MIN\_RANDOM\_FACTOR and MAX\_RANDOM\_FACTOR in order to prevent multiple nodes from transmitting at exactly the same time. |
| tcpIpNdpRndRtr  Solicitation  DelayEnabled | Boolean | 0..1 | attr | If enabled the first router solicitation will be delayed randomly from [0...MAX\_RTR\_SOLICITATION\_DELAY]. Otherwise the first router solicitation will be sent after exactly MAX\_RTR\_SOLICITATION\_DELAY milliseconds. |
| tcpIpNdpRtr  Solicitation  Interval | TimeValue | 0..1 | attr | Interval between consecutive Router Solicitations in (s). |
| tcpIpNdpSlaac DadNumberOf  Transmissions | PositiveInteger | 0..1 | attr | Number of Neighbor Solicitations that have to be unanswered in order to set an autoconfigurated address to PREFERRED (usable) state. |
| tcpIpNdpSlaac  Dad  Retransmission  Delay | TimeValue | 0..1 | attr | Sets the maximum value for the address configuration delay (s). |
| tcpIpNdpSlaac DelayEnabled | Boolean | 0..1 | attr | If enabled transmission of the first DAD Neighbor Solicitation will be delayed by a random value from [0...MAX\_DAD\_DELAY]. |
| tcpIpNdpSlaac  OptimisticDad  Enabled | Boolean | 0..1 | attr | Enable Optimistic Duplicate Address Detection (DAD) according to RFC4429. |

**Table 5.21: Ipv6NdpProps**

**5.2.2.2 TCP and UDP configuration properties**

*ARElement*

EthTcpIpProps

TcpProps

[0..1]

tcpCongestionAvoidanceEnabled: Boolean

+

[0..1]

tcpDelayedAckTimeout: TimeValue

+

[0..1]

tcpFastRecoveryEnabled: Boolean

+

[0..1]

tcpFastRetransmitEnabled: Boolean

+

tcpFinWait2Timeout: TimeValue

+

[0..1]

[0..1]

tcpKeepAliveEnabled: Boolean

+

tcpKeepAliveInterval: TimeValue

+

[0..1]

+

tcpKeepAliveProbesMax: PositiveInteger

[0..1]

+

tcpKeepAliveTime: TimeValue

[0..1]

+

tcpMaxRtx: PositiveInteger

[0..1]

+

tcpMsl: TimeValue

[0..1]

+

tcpNagleEnabled: Boolean

[0..1]

+

[0..1]

tcpReceiveWindowMax: PositiveInteger

[0..1]

tcpRetransmissionTimeout: TimeValue

+

+

[0..1]

tcpSlowStartEnabled: Boolean

tcpSynMaxRtx: PositiveInteger

+

[0..1]

tcpSynReceivedTimeout: TimeValue

+

[0..1]

[0..1]

+

tcpTtl: PositiveInteger

UdpProps

+

udpTtl: PositiveInteger

[0..1]

TcpIpIcmpv6Props

[0..1]

tcpIpIcmpV6EchoReplyAvoidFragmentation: Boolean

+

[0..1]

+

tcpIpIcmpV6EchoReplyEnabled: Boolean

+

tcpIpIcmpV6HopLimit: PositiveInteger

[0..1]

+

tcpIpIcmpV6MsgDestinationUnreachableEnabled: Boolean

[0..1]

[0..1]

+

tcpIpIcmpV6MsgParameterProblemEnabled: Boolean

TcpIpIcmpv4Props

+

tcpIpIcmpV4EchoReplyEnabled: Boolean

[0..1]

tcpIpIcmpV4Ttl: PositiveInteger

[0..1]

+

*ARElement*

EthTcpIpIcmpProps

*AtpStructureElement*

*FibexElement*

MachineDesign

+

accessControl: AccessControlEnum

[0..1]

+

pncPrepareSleepTimer: TimeValue

[0..1]

pnResetTimer: TimeValue

[0..1]

+

+

tcpIpProps

0..1

udpProps

+

0..1

tcpProps

+

0..1

tcpIpIcmpProps

+

0..1

+

icmpV6Props

0..1

+

icmpV4Props

0..1

**Figure 5.4: Machine specific TCP/UDP and ICMP configuration options**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **EthTcpIpProps** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | This meta-class is used to configure the Machine specific TcpIp Stack attributes.  **Tags:**atp.recommendedPackage=EthTcpIpProps | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| tcpProps | TcpProps | 0..1 | aggr | TCP configuration properties |
| udpProps | UdpProps | 0..1 | aggr | UDP configuration properties |

**Table 5.22: EthTcpIpProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **UdpProps** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | This meta-class specifies the configuration options for UDP (User Datagram Protocol). | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| udpTtl | PositiveInteger | 0..1 | attr | Default Time-to-live value of outgoing UDP packets. |

**Table 5.23: UdpProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **TcpProps** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | This meta-class specifies the configuration options for TCP (Transmission Control Protocol). | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| tcpCongestion  Avoidance  Enabled | Boolean | 0..1 | attr | Enables (TRUE) or disables (FALSE) support of TCP congestion avoidance algorithm according to IETF RFC 5681. |
| tcpDelayedAck Timeout | TimeValue | 0..1 | attr | The maximal time an acknowledgement is delayed for transmission in seconds. |
| tcpFast Recovery  Enabled | Boolean | 0..1 | attr | Enables (TRUE) or disables (FALSE) support of TCP Fast Recovery according to IETF RFC 5681. |
| tcpFast Retransmit  Enabled | Boolean | 0..1 | attr | Enables (TRUE) or disables (FALSE) support of TCP Fast Retransmission according to IETF RFC 5681. |
| tcpFin  Wait2Timeout | TimeValue | 0..1 | attr | Timeout in [s] to receive a FIN from the remote node  (after this node has initiated connection termination), i.e. maximum time waiting in FINWAIT-2 for a connection termination request from the remote TCP. |
| tcpKeepAlive Enabled | Boolean | 0..1 | attr | Enables (TRUE) or disables (FALSE) TCP Keep Alive Probes according to IETF RFC 1122 chapter 4.2.3.6. |
| tcpKeepAlive Interval | TimeValue | 0..1 | attr | Specifies the interval in seconds between subsequent keepalive probes. |
| tcpKeepAlive ProbesMax | PositiveInteger | 0..1 | attr | Maximum number of times that a TCP Keep Alive is retransmitted before the connection is closed. |
| tcpKeepAlive Time | TimeValue | 0..1 | attr | Specifies the time in [s] between the last data packet sent (simple ACKs are not considered data) and the first keepalive probe. |
| tcpMaxRtx | PositiveInteger | 0..1 | attr | Maximum number of times that a TCP segment is retransmitted before the TCP connection is closed. This parameter is only valid if tcpRetransmissionTimeout is configured. Note: This parameter also applies for FIN retransmissions. |
| tcpMsl | TimeValue | 0..1 | attr | Maximum segment lifetime in [s]. |
| tcpNagle Enabled | Boolean | 0..1 | attr | Enables (TRUE) or disables (FALSE) support of Nagle’s algorithm according to IETF RFC 1122 (chapter 4.2.3.4 When to Send Data). If enabled the Nagle’s algorithm is activated per default for all TCP sockets, but can be deactivated per Socket (with the attribute TcpTp.nagle Algorithm). |
| tcpReceive  WindowMax | PositiveInteger | 0..1 | attr | Default value of maximum receive window in bytes. |
| tcp  Retransmission  Timeout | TimeValue | 0..1 | attr | Timeout in [s] before an unacknowledged TCP segment is sent again. If the timeout is disabled, no TCP segments shall be retransmitted. |
| tcpSlowStart Enabled | Boolean | 0..1 | attr | Enables (TRUE) or disables (FALSE) support of TCP slow start algorithm according to IETF RFC 5681. |
| tcpSynMaxRtx | PositiveInteger | 0..1 | attr | Maximum number of times that a TCP SYN is retransmitted. |
| tcpSynReceived Timeout | TimeValue | 0..1 | attr | Timeout in [s] to complete a remotely initiated TCP connection establishment, i.e. maximum time waiting in SYN-RECEIVED for a confirming connection request acknowledgement after having both received and sent a connection request. |
| tcpTtl | PositiveInteger | 0..1 | attr | Default Time-to-live value of outgoing TCP packets. |

**Table 5.24: TcpProps**

**5.2.2.3 ICMP configuration properties**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **EthTcpIpIcmpProps** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | This meta-class is used to configure the Machine specific ICMP (Internet Control Message Protocol) attributes  **Tags:**atp.recommendedPackage=EthTcpIcmpProps | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| icmpV4Props | TcpIpIcmpv4Props | 0..1 | aggr | ICMPv4 configuration properties |
| icmpV6Props | TcpIpIcmpv6Props | 0..1 | aggr | ICMPv6 configuration properties |

**Table 5.25: EthTcpIpIcmpProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **TcpIpIcmpv4Props** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | This meta-class specifies the configuration options for ICMPv4 (Internet Control Message Protocol). | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| tcpIpIcmp V4EchoReply  Enabled | Boolean | 0..1 | attr | This attribute enables or disables transmission of ICMP echo reply message in case of a ICMP echo reception. |
| tcpIpIcmpV4Ttl | PositiveInteger | 0..1 | attr | This attribute is only relevant in case that ICMP (Internet Control Message Protocol) is used. It specifies the default Time-to-live value of outgoing ICMP packets. |

**Table 5.26: TcpIpIcmpv4Props**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **TcpIpIcmpv6Props** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***Note*** | This meta-class specifies the configuration options for ICMPv6 (Internet Control Message Protocol). | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| tcpIpIcmp  V6EchoReply  Avoid  Fragmentation | Boolean | 0..1 | attr | This attribute defines whether the echo reply is only transmitted in case that the incoming ICMPv6 Echo Request (Pings) fits the MTU of the respective interface, i.e. can be transmitted without IPv6 fragmentation. |
| tcpIpIcmp V6EchoReply  Enabled | Boolean | 0..1 | attr | This attribute enables or disables transmission of ICMP echo reply message in case of a ICMP echo reception. |
| tcpIpIcmp  V6HopLimit | PositiveInteger | 0..1 | attr | Default Hop-Limit value of outgoing ICMPv6 packets. |
| tcpIpIcmp  V6Msg  Destination  Unreachable  Enabled | Boolean | 0..1 | attr | This attribute Enables/Disables the transmission of Destination Unreachable Messages. |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **TcpIpIcmpv6Props** | |  |  |
| tcpIpIcmp  V6Msg  Parameter  Problem  Enabled | Boolean | 0..1 | attr | If enabled an ICMPv6 parameter problem message will be sent if a received packet has been dropped due to unknown options or headers that are found in the packet. |

**Table 5.27: TcpIpIcmpv6Props**

**5.2.3 Securing Communication with IPsec**

IPsec is a protocol suite that provides cryptographic protection for IP datagrams in IPv4 and IPv6 network packets.

IPsec uses a security association to specify security properties that are shared between the communicating parties. The security association defines a relationship between two or more parties and determines which security services will be used to communicate securely. In other words the security association serves as a “contract” between the different devices.

A single security association protects data in one communication direction. Two security associations shall be present to secure traffic in both directions. Each security association can provide encryption, data integrity and data authentication.

In addition, the senders and receivers of IP datagrams can determine the required protection for an IP packet according to IPsec security policies. These are rules that define how datagrams are processed that are received by a device. For example, security policies are used to decide if a particular packet needs to be dropped or needs to be processed by IPsec.

**[TPS\_MANI\_03203]**{DRAFT} **Configuration of IPsec** dThe IPSecConfig metaclass that is aggregated by a NetworkEndpoint in the role ipSecConfig provides the ability to define IPsec settings that are necessary to configure IPsec security associations and IPsec security policies.c*(RS\_MANI\_00036)*

**[TPS\_MANI\_03204]**{DRAFT} **Definition of IPSecRules** dThe IPSecConfig metaclass may contain one or several IPSecRules. Each IPSecRule defines the network connection that is monitored by IPsec by defining the local endpoint and the remote endpoint. Each endpoint is defined by the IP Address and the Tcp/Udp Port. The communication direction for which the IPSecRule is valid is defined by the direction attribute.c*(RS\_MANI\_00036)*

*PhysicalChannel*

EthernetPhysicalChannel

*Identifiable*

NetworkEndpoint

+

fullyQualifiedDomainName: String

[0..1]

+

priority: PositiveInteger

[0..1]

*NetworkEndpointAddress*

Ipv6Configuration

+

[0..1]

assignmentPriority: PositiveInteger

+

defaultRouter: Ip6AddressString

[0..1]

+

[0..\*]

dnsServerAddress: Ip6AddressString

[0..1]

enableAnycast: Boolean

+

+

hopCount: PositiveInteger

[0..1]

+

ipAddressKeepBehavior: IpAddressKeepEnum

[0..1]

[0..1]

+

ipAddressPrefixLength: PositiveInteger

+

ipv6Address: Ip6AddressString

[0..1]

+

ipv6AddressSource: Ipv6AddressSourceEnum

[0..1]

Ipv4Configuration

assignmentPriority: PositiveInteger

+

[0..1]

[0..1]

defaultGateway: Ip4AddressString

+

dnsServerAddress: Ip4AddressString

[0..\*]

+

ipAddressKeepBehavior: IpAddressKeepEnum

+

[0..1]

ipv4Address: Ip4AddressString

[0..1]

+

[0..1]

ipv4AddressSource: Ipv4AddressSourceEnum

+

+

networkMask: Ip4AddressString

[0..1]

[0..1]

ttl: PositiveInteger

+

*ARElement*

IPSecConfigProps

+

ahCipherSuiteName: String

[0..\*]

[0..1]

+

dpdAction: IPsecDpdActionEnum

[0..1]

+

dpdDelay: TimeValue

[0..\*]

+

espCipherSuiteName: String

+

ikeCipherSuiteName: String

[0..1]

+

ikeOverTime: TimeValue

[0..1]

+

ikeRandTime: PositiveInteger

[0..1]

ikeReauthTime: TimeValue

[0..1]

+

+

ikeRekeyTime: TimeValue

[0..1]

+

saOverTime: PositiveInteger

[0..1]

+

saRandTime: TimeValue

[0..1]

+

saRekeyTime: TimeValue

[0..1]

*Identifiable*

IPSecRule

+

direction: CommunicationDirectionType

[0..1]

[0..1]

+

headerType: IPsecHeaderTypeEnum

ipProtocol: IPsecIpProtocolEnum

+

[0..1]

[0..1]

+

localId: String

+

localPortRangeEnd: PositiveInteger

[0..1]

+

localPortRangeStart: PositiveInteger

[0..1]

+

mode: IPsecModeEnum

[0..1]

+

policy: IPsecPolicyEnum

[0..1]

priority: PositiveInteger

[0..1]

+

+

remoteId: String

[0..1]

remotePortRangeEnd: PositiveInteger

+

[0..1]

[0..1]

remotePortRangeStart: PositiveInteger

+

IPSecConfig

*ARElement*

CryptoServiceCertificate

[0..1]

algorithmFamily: CryptoCertificateAlgorithmFamilyEnum

+

+

format: CryptoCertificateFormatEnum

[0..1]

[0..1]

+

maximumLength: PositiveInteger

[0..1]

serverNameIdentification: String

+

*ARElement*

CryptoServiceKey

algorithmFamily: String

+

[0..1]

keyGeneration: CryptoServiceKeyGenerationEnum

+

keyStorageType: String

[0..1]

+

length: PositiveInteger

+

networkEndpointAddress

+

1..\*

ipSecConfig

+

0..1

+

localCertificate

0..\*

0..1

+

nextHigherCertificate

ipSecRule

+

0..\*

preSharedKey

+

0..1

+

remoteCertificate

0..\*

+

remoteIpAddress

0..\*

+

networkEndpoint

0..\*

«atpSplitable»

ipSecConfigProps

+

0..1

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | «enumeration»  IPsecIpProtocolEnum | | udp tcp any icmp | | |  | | --- | | «enumeration»  CommunicationDirectionType | | in out | | |  | | --- | | «enumerati... IPsecPolicyEnum | | ipsec passthrough drop  reject | | |  | | --- | | «enumeration»  IPsecModeEnum | | tunnel transport | |
| |  | | --- | | «enumeration»  IPsecHeaderTypeEnum | | ah esp none | |
| |  | | --- | | «enumeration»  IPsecDpdActionEnum | | clear trap  restart | |

**Figure 5.5: IPsec configuration model**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **IPSecConfig** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::SecureCommunication | | | |
| ***Note*** | IPsec is a protocol that is designed to provide "end-to-end" cryptographically-based security for IP network connections. | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| ipSecConfig Props | IPSecConfigProps | 0..1 | ref | Global IPsec configuration settings that are valid for all IPSecRules that are defined on the NetworkEndpoint. |
| ipSecRule | IPSecRule | \* | aggr | IPSec rules and filters that are defined in the IPSecConfig for a specific NetworkEndpoint. |

**Table 5.28: IPSecConfig**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **IPSecRule** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::SecureCommunication | | | |
| ***Note*** | This element defines an IPsec rule that describes communication traffic that is monitored, protected and filtered. | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| direction | Communication  DirectionType | 0..1 | attr | This attribute defines the direction in which the traffic is monitored. If this attribute is not set a bidirectional traffic monitoring is assumed. |
| headerType | IPsecHeaderTypeEnum | 0..1 | attr | Header type specifying the IPsec security mechanism. |
| ipProtocol | IPsecIpProtocolEnum | 0..1 | attr | This attribute defines the relevant IP protocol used in the Security Policy Database (SPD) entry. |
| localCertificate | CryptoService  Certificate | \* | ref | This reference identifies the applicable certificate used for a local authentication. |
| localId | String | 0..1 | attr | This attribute defines how the local participant should be identified for authentication. |
| localPortRange End | PositiveInteger | 0..1 | attr | This attribute restricts the traffic monitoring and defines an end value for the local port range.  If this attribute is not set then this rule shall be effective for all local ports.  Please note that port ranges are currently not supported in the AUTOSAR AP’s operating system backend. If AP systems are involved, each IPsec rule may only contain a single port. |
| localPortRange Start | PositiveInteger | 0..1 | attr | This attribute restricts the traffic monitoring and defines a start value for the local port range.  If this attribute is not set then this rule shall be effective for all local ports.  Please note that port ranges are currently not supported in the AUTOSAR AP’s operating system backend. If AP systems are involved, each IPsec rule may only contain a single port. |
| mode | IPsecModeEnum | 0..1 | attr | This attribute defines the type of the connection. |
| policy | IPsecPolicyEnum | 0..1 | attr | An IPsec policy defines the rules that determine which type of IP traffic needs to be secured using IPsec and how that traffic is secured. |
| preSharedKey | CryptoServiceKey | 0..1 | ref | This reference identifies the applicable cryptograhic key used for authentication. |
| priority | PositiveInteger | 0..1 | attr | This attribute defines the priority of the IPSecRule (SPD entry). The processing of entries is based on priority, starting with the highest priority "0". |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **IPSecRule** |  | | |
| remote  Certificate | CryptoService  Certificate | \* | ref | This reference identifies the applicable certificate used for a remote authentication. |
| remoteId | String | 0..1 | attr | This attribute defines how the remote participant should be identified for authentication. |
| remoteIp Address | NetworkEndpoint | \* | ref | Definition of the remote NetworkEndpoint. With this reference the connection between the local Network Endpoint and the remote NetworkEndpoint is described on which the traffic is monitored. |
| remotePort RangeEnd | PositiveInteger | 0..1 | attr | This attribute restricts the traffic monitoring and defines an end value for the remote port range.  If this attribute is not set then this rule shall be effective for all local ports.  Please note that port ranges are currently not supported in the AUTOSAR AP’s operating system backend. If AP systems are involved, each IPsec rule may only contain a single port. |
| remotePort RangeStart | PositiveInteger | 0..1 | attr | This attribute restricts the traffic monitoring and defines a start value for the remote port range.  If this attribute is not set then this rule shall be effective for all local ports.  Please note that port ranges are currently not supported in the AUTOSAR AP’s operating system backend. If AP systems are involved, each IPsec rule may only contain a single port. |

**Table 5.29: IPSecRule**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **IPSecConfigProps** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::SecureCommunication | | | |
| ***Note*** | This element holds all the attributes for configuration of IPsec that are independent of specific IPsec rules.  **Tags:**atp.recommendedPackage=IPSecConfigProps | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| ahCipherSuite  Name | String | \* | attr | AH (Authentication Header) algorithm to be used for the connection, e.g. HMAC/SHA2-256 |
| dpdAction | IPsecDpdActionEnum | 0..1 | attr | This attribute defines what to do if the peer is considered dead.  If not configured "restart" shall be assumed. |
| dpdDelay | TimeValue | 0..1 | attr | This attribute describes the interval to check the liveness of a peer actively using IKEv2 INFORMATIONAL exchanges. Active DPD checking is only enforced if no IKE or ESP/AH packet has been received for the configured DPD delay.  In not configured the value "5 minutes" shall be assumed. |
| espCipherSuite  Name | String | \* | attr | ESP (Encapsulating Security Payload) algorithm that provides encryption and optional authentication for the connection, e.g. AES-128+SHA2-256. |
| ikeCipherSuite  Name | String | 0..1 | attr | IKE encryption/authentication algorithms to be used for the connection. |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **IPSecConfigProps** | | | |
| ikeOverTime | TimeValue | 0..1 | attr | This attribute describes the hard deadline when an SA becomes invalid in percentage.  Example: ikeOverTime of max(ikeReauthTime, ikeRekey Time).  Default: 10 % |
| ikeRandTime | PositiveInteger | 0..1 | attr | This attribute defines in percentage by how long before the expiration of ikeReauthTime and ikeRekeyTime will be rekeyed/reauthenticated.  Default: 10% |
| ikeReauthTime | TimeValue | 0..1 | attr | This attribute defines the absolute time after which an IKE SA will be reauthenticated.  0 means reauthentication is disabled. |
| ikeRekeyTime | TimeValue | 0..1 | attr | This attribute defines the absolute time after which an IKE SA will be rekeyed.  0 means rekey is disabled. |
| saOverTime | PositiveInteger | 0..1 | attr | This attribute describes the hard deadline when an IPsec SA becomes invalid in percentage.  Example: saOverTime \* saRekeyTime.  Default: 110% |
| saRandTime | TimeValue | 0..1 | attr | This attribute defines by how long before the expiration of saRekeyTime will be rekeyed. |
| saRekeyTime | TimeValue | 0..1 | attr | This attribute defines the absolute time after which an IPsec SA will be rekeyed.  0 means rekey is disabled. |

**Table 5.30: IPSecConfigProps**

|  |  |
| --- | --- |
| ***Enumeration*** | **IPsecIpProtocolEnum** |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::SecureCommunication |
| ***Note*** | Definition of supported TcpIp protocols that are supported in Security Policy Database (SPD) entries in IPSec configurations. |
| ***Literal*** | ***Description*** |
| any | ANY protocol  **Tags:**atp.EnumerationLiteralIndex=3 |
| icmp | Internet Control Message Protocol (ICMP)  **Tags:**atp.EnumerationLiteralIndex=2 |
| tcp | TCP Protocol  **Tags:**atp.EnumerationLiteralIndex=1 |
| udp | UDP Protocol  **Tags:**atp.EnumerationLiteralIndex=0 |

**Table 5.31: IPsecIpProtocolEnum**

**[constr\_5102]**{DRAFT} **Usage of remote port ranges in IPSecRule is not allowed** dIPSecRule.remotePortRangeStart and IPSecRule.remotePortRangeEnd shall always be set to the same value.c*()*

**[constr\_5103]**{DRAFT} **Usage of local port ranges in IPSecRule is not allowed** dIPSecRule.localPortRangeStart and IPSecRule.localPortRangeEnd shall always be set to the same value.c*()*

The reason for [constr\_5102] and [constr\_5103] is that port ranges are currently not supported by the AUTOSAR Adaptive Platform operating system backend and each IPSecRule is allowed to define only a single local Port and a single remote Port.

**[TPS\_MANI\_03232]**{DRAFT} **Definition of general IPsec configuration settings** dGeneral configuration properties that are independent of particular IPSecRules are collected in the IPSecConfigProps element that is referenced from the IPSecConfig in the role ipSecConfigProps.c*(RS\_MANI\_00036)*

**[TPS\_MANI\_03205]**{DRAFT} **IPsec policy** dThe IPSecRule.policy attribute defines how IP packets are handled that are going over the network connection defined by the IPSecRule. In detail, it defines whether the IP packet is processed by IPsec or not.c*(RS\_MANI\_00036)*

|  |  |
| --- | --- |
| ***Enumeration*** | **IPsecPolicyEnum** |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::SecureCommunication |
| ***Note*** | Defines the filter actions that are supported by IPsec. |
| ***Literal*** | ***Description*** |
| drop | Signifying that packets should be discarded  **Tags:**atp.EnumerationLiteralIndex=3 |
| ipsec | Signifying that packets should be protected.  **Tags:**atp.EnumerationLiteralIndex=1 |
| passthrough | Signifying that no IPsec processing should be done at all.  **Tags:**atp.EnumerationLiteralIndex=2 |
| reject | Signifying that packets should be discarded and a diagnostic ICMP returned.  **Tags:**atp.EnumerationLiteralIndex=4 |

**Table 5.32: IPsecPolicyEnum**

IPsec can be configured to operate in two different modes, Tunnel and Transport mode. With tunnel mode, the entire IP packet is protected by IPsec. IPsec wraps the original packet, encrypts it and adds a new IP header to it.

The tunnel mode is most commonly used between VPN gateways and the IP addresses of the newly added outer IP header are that of the VPN Gateways. In other words the traffic between the two VPN Gateways is protected and each gateway acts as a proxy for the hosts behind it.

The transport mode provides the protection of the Data Payload of the IP datagram with an AH or ESP header. The IP Header remains the same and IPsec inserts its header between the IP header and the upper level headers.

The IPsec transport mode can be used when securing traffic between two hosts or between a host and a VPN gateway.

**[TPS\_MANI\_03233]**{DRAFT} **IPsec mode** dThe IPSecRule.mode attribute defines whether the IP packet is processed in the transport or tunnel mode.c*(RS\_MANI\_00036)*

Please note that AUTOSAR currently supports only the transportmode as configuration option.

|  |  |
| --- | --- |
| ***Enumeration*** | **IPsecModeEnum** |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::SecureCommunication |
| ***Note*** | This enumeration describes the supported IPSec modes. |
| ***Literal*** | ***Description*** |
| transport | Signifying that the IPSec transport mode is used. With the transport mode the original IP header is retained and only the IP payload and ESP trailer is encrypted.  **Tags:**atp.EnumerationLiteralIndex=1 |
| tunnel | Signifying that the IPSec tunnel mode is used. With tunnel mode, the entire original IP packet is protected by IPSec. This means IPSec wraps the original packet, encrypts it, adds a new IP header and sends it to the other side.  **Tags:**atp.EnumerationLiteralIndex=0 |

**Table 5.33: IPsecModeEnum**

IPsec uses two protocols:

* AH - Authentication Header
* ESP - Encapsulating Security Payload

The AH protocol provides a mechanism for authentication only and authenticates the entire IP packet, including the outer IP header.

The ESP protocol provides data confidentiality (encryption) and/or authentication (data integrity, data origin authentication, and replay protection).

When ESP is used in transport mode, the IP payload is encrypted and the original IP header is moved to the front of the message. The ESP header is inserted after the IP header and is signed together with the IP payload. The original IP header remains unprotected.

When ESP is used in tunnel mode a new IP Header is created and the ESP header is added in front of the original IP Packet. The entire original IP packet is encrypted and signed in this mode.

**[TPS\_MANI\_03206]**{DRAFT} **IPsec AH and ESP protocol configuration** dIn the IPSecRule it is possible to define the IPsec protocol that shall be used to protect IP packets that are going over the defined network connection. The attribute headerType defines whether AH, ESP or neither one is used.c*(RS\_MANI\_00036)*

|  |  |
| --- | --- |
| ***Enumeration*** | **IPsecHeaderTypeEnum** |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::SecureCommunication |
| ***Note*** | IPsec Header Type options |
| ***Literal*** | ***Description*** |
| ah | Authentication Header (AH)  **Tags:**atp.EnumerationLiteralIndex=0 |

5

4

|  |  |
| --- | --- |
| ***Enumeration*** | **IPsecHeaderTypeEnum** |
| esp | Encapsulating Security Payloads (ESP)  **Tags:**atp.EnumerationLiteralIndex=1 |
| none | No header  **Tags:**atp.EnumerationLiteralIndex=2 |

**Table 5.34: IPsecHeaderTypeEnum**

**[TPS\_MANI\_03234]**{DRAFT} **IPsec AH and ESP CipherSuites** dThe attributes ah-

CipherSuiteName and espCipherSuiteName define the supported AH and ESP algorithms.c*(RS\_MANI\_00036)*

The naming convention for ahCipherSuiteName, espCipherSuiteName and IPSecConfigProps.ikeCipherSuiteName shall follow the naming convention for cryptographic primitives that is defined in [13].

**[TPS\_MANI\_03207]**{DRAFT} **IPsec Internet Key Exchange protocol configuration** dIn the IPSecRule it is possible to define how IKE protocol authenticates the remote party and how the local party authenticates itself to the remote party. In other words both sides use the same method. The usage of the IPSecRule.preSharedKey reference defines that the pre-shared key is used. The usage of the IPSecRule.local-

Certificate and IPSecRule.remoteCertificate defines that Digital Signature Authentication is used.c*(RS\_MANI\_00036)*

Please note that the supported IKE CipherSuites are configured with the IPSec-

ConfigProps.ikeCipherSuiteName. The IPSecConfigProps contains additional IKE specific configuration settings.

|  |  |
| --- | --- |
| ***Enumeration*** | **IPsecDpdActionEnum** |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::SecureCommunication |
| ***Note*** | Potential Dead Peer Detection (Dpd) Actions |
| ***Literal*** | ***Description*** |
| clear | Deletes the SA.  **Tags:**atp.EnumerationLiteralIndex=0 |
| restart | Immediately tries to establish the connection.  **Tags:**atp.EnumerationLiteralIndex=2 |
| trap | tries to establish the connection after traffic is sent to the peer.  **Tags:**atp.EnumerationLiteralIndex=1 |

**Table 5.35: IPsecDpdActionEnum**

**[TPS\_MANI\_03208]**{DRAFT} **Protection of AdaptivePlatformServiceInstance by IPsec** dTo describe the protection of an AdaptivePlatformServiceInstance by IPsec the AdaptivePlatformServiceInstance needs to be mapped by a ServiceInstanceToMachineMapping to an EthernetCommunicationConnector that points with the unicastNetworkEndpoint to a NetworkEndpoint that aggregates the IPSecConfig that in turn describes IPsec

Security Associations.c*(RS\_MANI\_00036)*

Please note that IP Multicast protection by IPsec is not supported. It is by intention not possible to model the IPsec protection of IP Multicast communication since the IP Multicast address is defined in the SomeipProvidedEventGroup by the two attributes ipv4MulticastIpAddress and ipv6MulticastIpAddress. The NetworkEndpoint element is used for description of IP Unicast Endpoints only. This means that only the IP Unicast communication of an AdaptivePlatformServiceInstance that is described according to [TPS\_MANI\_03208] will be protected by IPsec.

**5.2.4 Service Discovery Configuration**

Service Discovery messages are exchanged between network nodes to announce and to discover available service instances. This chapter describes the configuration that is necessary to exchange service discovery messages for supported middleware transport layers.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***ServiceDiscoveryConfiguration*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest | | | |
| ***Note*** | Service Discovery configuration settings for the middleware transport layer.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject* | | | |
| ***Subclasses*** | SomeipServiceDiscovery | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 5.36: ServiceDiscoveryConfiguration**

**5.2.4.1 SOME/IP Service Discovery Configuration**

**[TPS\_MANI\_03064]**{DRAFT} **SOME/IP Service Discovery message exchange configuration** dProvidedServiceInstances are announced in SOME/IP by the server with multicast addressing on a VLAN to a specifically designated IP multicast address (SomeipServiceDiscovery.multicastSdIpAddress) at a specific UDP port number (SomeipServiceDiscovery.someipServiceDiscoveryPort).c*(RS\_MANI\_00019)*

**[constr\_5045]**{DRAFT} **Only one SomeipServiceDiscovery configuration per VLAN is allowed** dOnly a single NetworkEndpoint on an EthernetPhysicalChannel (VLAN) is allowed to be referenced by a SomeipServiceDiscovery element in the role multicastSdIpAddress.c*()*

The SomeipServiceDiscovery is able to reference SecureComProps to define and to configure a security protocol that will provide communication security for Service Discovery messages.

For Service Discovery messages that will be transmitted to a designated multicast IP address the protection is defined by the SecureComProps that is referenced in the role multicastSecureComProps. For unicast Service Discovery messages different credentials may be used for the different ECU pairs.

Therefore, a list of SecureComProps is aggregated in the role unicastSecureComProps.

*PhysicalChannel*

EthernetPhysicalChannel

*Identifiable*

NetworkEndpoint

fullyQualifiedDomainName: String

[0..1]

+

[0..1]

priority: PositiveInteger

+

SomeipServiceDiscovery

+

someipServiceDiscoveryPort: PositiveInteger

*ServiceDiscoveryConfiguration*

*ARElement*

*AtpStructureElement*

Machine

trustedPlatformExecutableLaunchBehavior

+

:

TrustedPlatformExecutableLaunchBehaviorEnum

*ARElement*

*SecureComProps*

*AtpStructureElement*

*FibexElement*

MachineDesign

accessControl: AccessControlEnum

+

[0..1]

+

[0..1]

pncPrepareSleepTimer: TimeValue

+

pnResetTimer: TimeValue

[0..1]

«atpSplitable»

+

serviceDiscoveryConfig

0..\*

+

networkEndpoint

0..\*

«atpSplitable»

+

multicastSecureComProps

0..1

machineDesign

+

1

+

multicastSdIpAddress

0..1

+

unicastSecureComProps

0..\*

**Figure 5.6: SOME/IP Service Discovery Configuration**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SomeipServiceDiscovery** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment | | | |
| ***Note*** | This meta-class represents a specialization of the generic service discovery for the SOME/IP case.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *ServiceDiscoveryConfiguration* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| multicastSdIp Address | NetworkEndpoint | 0..1 | ref | This reference identifies the multicast IP address used for service discovery.  **Tags:**atp.Status=draft |
| multicastSecure ComProps | SecureComProps | 0..1 | ref | Reference to a communication security protocol and its configuration settings that will provide communication security for Service Discovery messages that are transmitted using multicast, e.g. FindService message.  **Tags:**atp.Status=draft |
| someipService DiscoveryPort | PositiveInteger | 1 | attr | This attribute represents the port number reserved for service discovery.  **Tags:**atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SomeipServiceDiscover** | **y** |  |  |
| unicastSecure ComProps | SecureComProps | \* | ref | Reference to a communication security protocol and its configuration settings that will provide communication security for Service Discovery messages that are transmitted using unicast, e.g. OfferService as answer to a FindService message.  .  **Tags:**atp.Status=draft |

**Table 5.37: SomeipServiceDiscovery**

**5.2.5 Partial Network**

AUTOSAR supports power saving during vehicle operation time with the partial networking mechanism. This mechanism allows shutting down and starting up the bus communication interfaces of groups of ECUs (Partial Network Cluster) during normal bus communication.

On the VFB Level Partial Networks are represented by Virtual Function Clusters and are described with PortGroups. The Virtual Function Cluster groups the communication necessary to realize one or more vehicle functions that can become activated/deactivated during normal vehicle operation. The Virtual Function Clusters are mapped onto Partial Network Clusters.

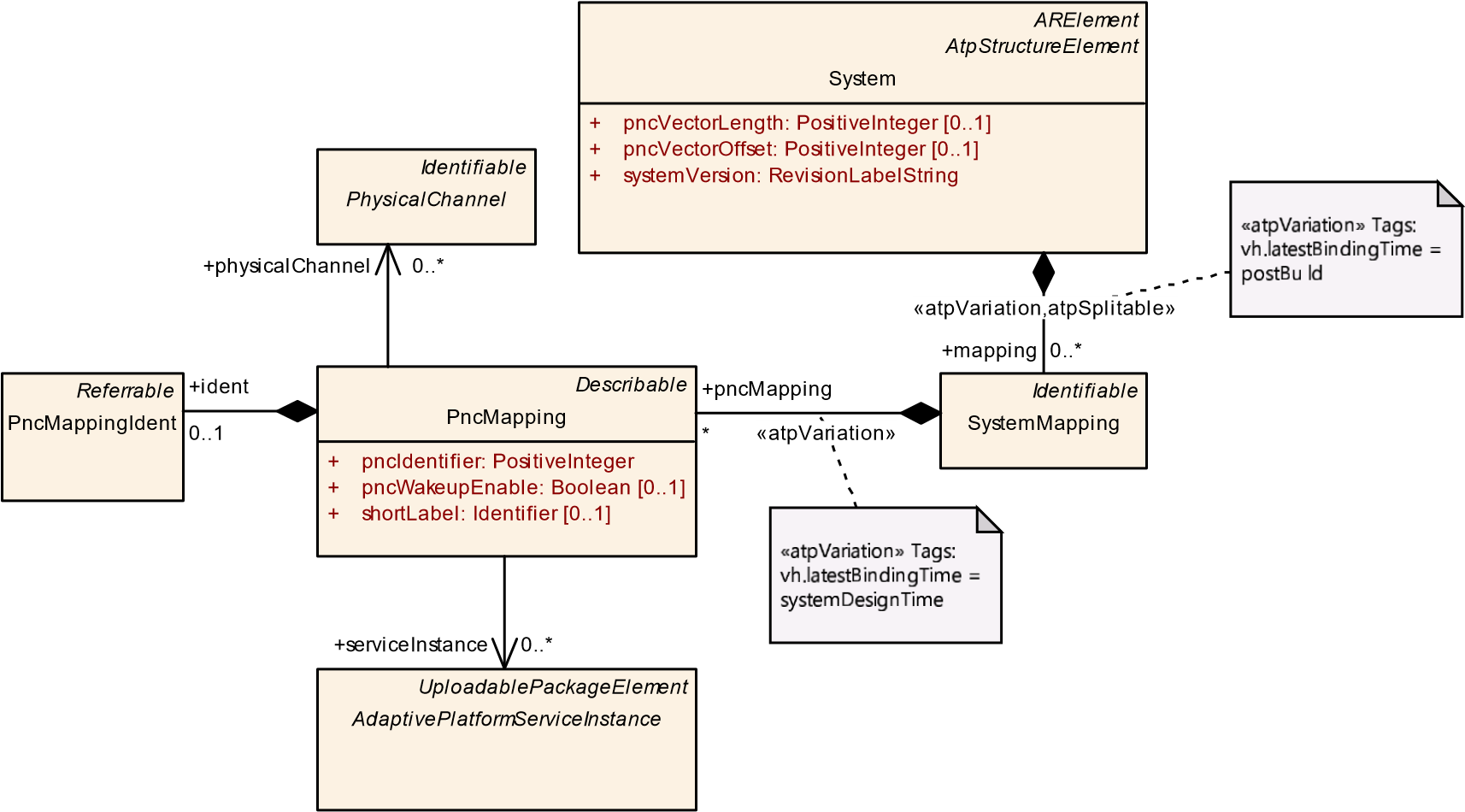
**[TPS\_MANI\_03224]**{DRAFT} **Modeling of a Partial Network Cluster** dA Partial Network Cluster is modeled with the PncMapping element and is identified by the pncIdentifier. The PncMapping defines the collection of AdaptivePlatformServiceInstances that are participating in the partial network with the PncMapping. serviceInstance reference.c*(RS\_MANI\_00062)*

**[TPS\_MANI\_03225]**{DRAFT} **References to VLANs in PncMapping**dThere are two ways for a PncMapping to relate to a VLAN or an untagged channel:

* via the reference from PncMapping to PhysicalChannel in the role physicalChannel,
* via a ServiceInstanceToMachineMapping that maps the AdaptivePlatformServiceInstance and that also refers an EthernetCommunicationConnector that in turn is connected to an EthernetPhysicalChannel via the unicastNetworkEndpoint.

The two options are not mutually exclusive, they can exist at the same time.c*(RS\_-*

*MANI\_00062)*



**Figure 5.7: PncMapping with collection of ServiceInstances that are participating in the**

**Partial Network Cluster**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SystemMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate | | | |
| ***Note*** | The system mapping aggregates all mapping aspects that are relevant in the System Description. | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| pncMapping | PncMapping | \* | aggr | Mappings between Virtual Function Clusters and Partial Network Clusters.  **Stereotypes:** atpVariation  **Tags:**vh.latestBindingTime=systemDesignTime |

**Table 5.38: SystemMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PncMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate::PncMapping | | | |
| ***Note*** | Describes a mapping between one or several Virtual Function Clusters onto Partial Network Clusters. A Virtual Function Cluster is realized by a PortGroup. A Partial Network Cluster is realized by one or more ServiceInstances. | | | |
| ***Base*** | *ARObject*, *Describable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| ident | PncMappingIdent | 0..1 | aggr | This adds the ability to become referrable to PncMapping. |
| physical  Channel | PhysicalChannel | \* | ref | This reference maps the partial network to a communication channel. |
| pncConsumed  Provided  ServiceInstance  Group | ConsumedProvided  ServiceInstanceGroup | \* | ref | ConsumedProvidedServiceInstanceGroup used in a Partial Network Cluster. This reference is optional, since this could be used for starting and stopping Consumed ProvidedServiceInstanceGroup according the requested partial network, but is not necessarily needed.  **Stereotypes:** atpVariation  **Tags:**vh.latestBindingTime=postBuild |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PncMapping** |  |  |  |
| pncIdentifier | PositiveInteger | 1 | attr | Identifer of the Partial Network Cluster. This number represents the absolute bit position of this Partial Network Cluster in the NM Pdu. |
| pncWakeup Enable | Boolean | 0..1 | attr | If this parameter is available and set to true then this PNC will be woken up as soon as a channel wakeup occurs on a channel where this PNC is assigned to. This is ensured by adding this PNC to the corresponding channel wakeup sources during upstream mapping. |
| serviceInstance | AdaptivePlatform  ServiceInstance | \* | ref | Reference to ServiceInstances that are participating in a Partial Network Cluster.  **Tags:**atp.Status=draft |
| shortLabel | Identifier | 0..1 | attr | This attribute specifies an identifying shortName for the PncMapping. It shall be unique in the System scope. |
| vfc | PortGroup | \* | iref | Virtual Function Cluster to be mapped onto a Partial Network Cluster. This reference is optional in case that the System Description doesn’t use a complete Software Component Description (VFB View). This supports the inclusion of legacy systems.  **InstanceRef implemented by:**PortGroupInSystem InstanceRef |

**Table 5.39: PncMapping**

## 5.3 Specification of Application Software System Structure

The root element of a System Design model is the System element that is already known from the AUTOSAR classic platform. The System aggregates the RootSwCompositionPrototype that represents the top-level-composition of all software components that are available in a given system.

**[TPS\_MANI\_03110]**{DRAFT} **Allowed components in system description with category SYSTEM\_DESIGN\_DESCRIPTION.** dSwComponentPrototypes nested inside the CompositionSwComponentType that is referenced by the RootSwCompositionPrototype of a System with category SYSTEM\_DESIGN\_DESCRIPTION are allowed to be of any SwComponentType that is supported by Classic or by Adaptive AUTOSAR.c*(RS\_MANI\_00026)*

|  |  |
| --- | --- |
| ***Class*** | **RootSwCompositionPrototype** |
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **RootSwCompositionPrototype** | | | |
| ***Note*** | The RootSwCompositionPrototype represents the top-level-composition of software components within a given System.  According to the use case of the System, this may for example be a more or less complete VFB description, the software of a System Extract or the software of a flat ECU Extract with only atomic SWCs.  Therefore the RootSwComposition will only occasionally contain all atomic software components that are used in a complete VFB System. The OEM is primarily interested in the required functionality and the interfaces defining the integration of the Software Component into the System. The internal structure of such a component contains often substantial intellectual property of a supplier. Therefore a top-level software composition will often contain empty compositions which represent subsystems.  The contained SwComponentPrototypes are fully specified by their SwComponentTypes (including Port Prototypes, PortInterfaces, VariableDataPrototypes, SwcInternalBehavior etc.), and their ports are interconnected using SwConnectorPrototypes. | | | |
| ***Base*** | *ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| software  Composition | CompositionSw  ComponentType | 1 | tref | We assume that there is exactly one top-level composition that includes all Component instances of the system.  **Stereotypes:** isOfType |

**Table 5.40: RootSwCompositionPrototype**

If a Software Component communicates over the service oriented communication and provides or requires a ServiceInterface the opposite communication end is not always known upfront. In the System with categorySYSTEM\_DESIGN\_DESCRIPTION a System Designer may want to indicate the service oriented communication between endpoints if it is already known at the System Design time.

AA1

P

AA2

R

AA3

R

IF\_A

IF\_A

IF\_A

searchForSpecificInstance

searchForAllInstances

**Figure 5.8: Example for Assembly connectors in System Design model**

**[TPS\_MANI\_03114]**{DRAFT} **Usage of AssemblySwConnectors in the System Design model** dIn the System with categorySYSTEM\_DESIGN\_DESCRIPTION it is allowed to indicate the service oriented communication between two communication endpoints by AssemblySwConnectors if the required RPortPrototype is searching for a specific service instance, i.e. if the RPortPrototypeProps.searchIntention is set to searchForSpecificInstance.

If the searchIntention is set to searchForAllInstances, the AssemblySwConnector shall not be used to connect this RPortPrototype.c*(RS\_MANI\_00026)*

**5.4 Modeling of service oriented communication between Classic**

## and Adaptive platform

AUTOSAR classic platform does not support ServiceInterfaces yet but provides the possibility to communicate in a service oriented way over SOME/IP. To mimic a ServiceInterface in the classic platform any combination of ClientServerInterfaces, SenderReceiverInterfaces or TriggerInterfaces may be used to describe a service to which later a SOME/IP Service ID is assigned.

To simplify the description of the service oriented communication between Classic and Adaptive Software components in a System design model the InterfaceMapping was introduced that allows to map elements of PortInterfaces of the Classic Platform to a single ServiceInterface of the Adaptive Platform.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **InterfaceMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign | | | |
| ***Note*** | This meta-class collects the mappings of elements of a single ServiceInterface to PortInterface elements of the AUTOSAR Classic Platform.  **Tags:**  atp.Status=draft  atp.recommendedPackage=InterfaceMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| eventMapping | EventMapping | \* | aggr | Mapping of a VariableDataPrototype in a SenderReceiver Interface to an Event in a ServiceInterface.  **Tags:**atp.Status=draft |
| fieldMapping | FieldMapping | \* | aggr | Mapping of a Field in a ServiceInterface to ClientServer Operations that represent the getter and setter methods and to a VariableDataPrototype that represents the notifier in the Field.  **Tags:**atp.Status=draft |
| fireAndForget Mapping | FireAndForgetMapping | \* | aggr | Mapping of a Fire&Forget Method that is located in a  ServiceInterface to a VariableDataPrototype in a Sender ReceiverInterface or to a Trigger in a TriggerInterface.  **Tags:**atp.Status=draft |
| methodMapping | MethodMapping | \* | aggr | Mapping of a ClientServerOperation in a ClientServer Interface to a Method in a ServiceInterface.  **Tags:**atp.Status=draft |

**Table 5.41: InterfaceMapping**

**[constr\_3370]**{DRAFT} **InterfaceMapping shall map all elements of a single ServiceInterface** dThe mappings that are included in an InterfaceMapping shall map all elements of a single ServiceInterface (i.e. fields, events, methods) to PortInterface elements of the classic platform.c*()*

*Identifiable*

FieldMapping

*Identifiable*

EventMapping

*Identifiable*

MethodMapping

*Identifiable*

FireAndForgetMapping

*ARElement*

InterfaceMapping

+

eventMapping

0..\*

+

fieldMapping

0..\*

+

methodMapping

0..\*

+

fireAndForgetMapping

0..\*

**Figure 5.9: InterfaceMapping Overview**

Figure 5.10 shows a possible System Design modeling approach where an a piece of application software is communicating in a service oriented way over SOME/IP with classic Software Components. SWC\_1 requires a ClientServerInterface IF\_Y with a ClientServerOperation and a SenderReceiverInterface IF\_X with a VariableDataPrototype. SWC\_2 requires a SenderReceiverInterface IF\_X with a VariableDataPrototype.

AA1

P

SWC\_1

R

SWC\_2

R

IF\_A

P

R

IF\_Y

R

IF\_X

IF\_X

IF\_Y

IF\_A

R

IF\_X

Interface

Mapping

Network / SOME/IP Protocol

**Figure 5.10: Example for a modeling of Service Oriented communication between application software on the *AUTOSAR adaptive platform* and Software Components of the Classic Platform**

The two PortInterfaces IF\_X and IF\_Y are mapped to a single ServiceInterface IF\_A using an InterfaceMapping.

On the other side the application software AA1 provides the ServiceInterface IF\_A.

Note that this is a mapping on PortInterface level. If each PortInterface is only used once in a network the actual communication can be directly derived out of the InterfaceMapping. If PortInterfaces are used several times on a network there is the need to take the network configuration into account in order to be able to emulate how the service discovery will behave on the network. From this information the actual communication relations on software level can be deduced.

**5.4.1 MethodMapping**

**[TPS\_MANI\_03111]**{DRAFT} **Mapping between method and operationlocated in a ClientServerInterface**dThe mapping between a method located in a ServiceInterface and a operation located in a ClientServerInterface is provided by the class MethodMapping.c*(RS\_MANI\_00026)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **MethodMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign | | | |
| ***Note*** | Mapping of a ClientServerOperation that is located in a ClientServerInterface to a Method that is located in a ServiceInterface.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| clientServer Operation | ClientServerOperation | 0..1 | ref | Reference to a ClientSeverOperation that is located in a ClientSeverInterface.  **Tags:**atp.Status=draft |
| method | ClientServerOperation | 0..1 | ref | Reference to a Method that is located in a Service  Interface.  **Tags:**atp.Status=draft |

**Table 5.42: MethodMapping**

|  |  |  |
| --- | --- | --- |
| |  | | --- | | *PortInterface*  ClientServerInterface | |  |

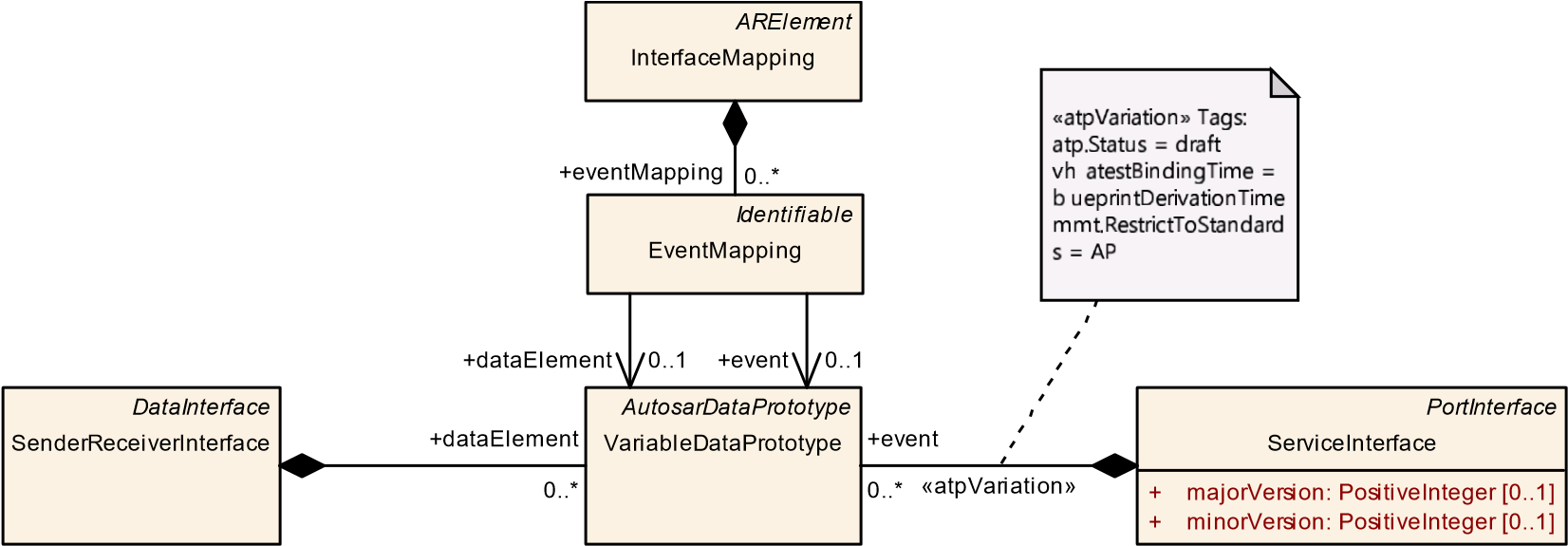
**Figure 5.11: Mapping of a Method to a ClientServerOperation**

**5.4.2 EventMapping**

**[TPS\_MANI\_03112]**{DRAFT} **Mapping between an event and a dataElement** dThe mapping between an event located in a ServiceInterface and a dataElement located in a SenderReceiverInterface is provided by the class EventMapping.c*(RS\_MANI\_00026)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **EventMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign | | | |
| ***Note*** | Mapping of a VariableDataPrototype that is located in a SenderReceiverInterface to an Event that is located in a ServiceInterface.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| dataElement | VariableDataPrototype | 0..1 | ref | Reference to a VariableDataPrototype that is located in a SenderReceiverInterface.  **Tags:**atp.Status=draft |
| event | VariableDataPrototype | 0..1 | ref | Reference to an Event that is located in a Service  Interface.  **Tags:**atp.Status=draft |

**Table 5.43: EventMapping**



**Figure 5.12: Mapping between an event and a dataElement**

**5.4.3 FieldMapping**

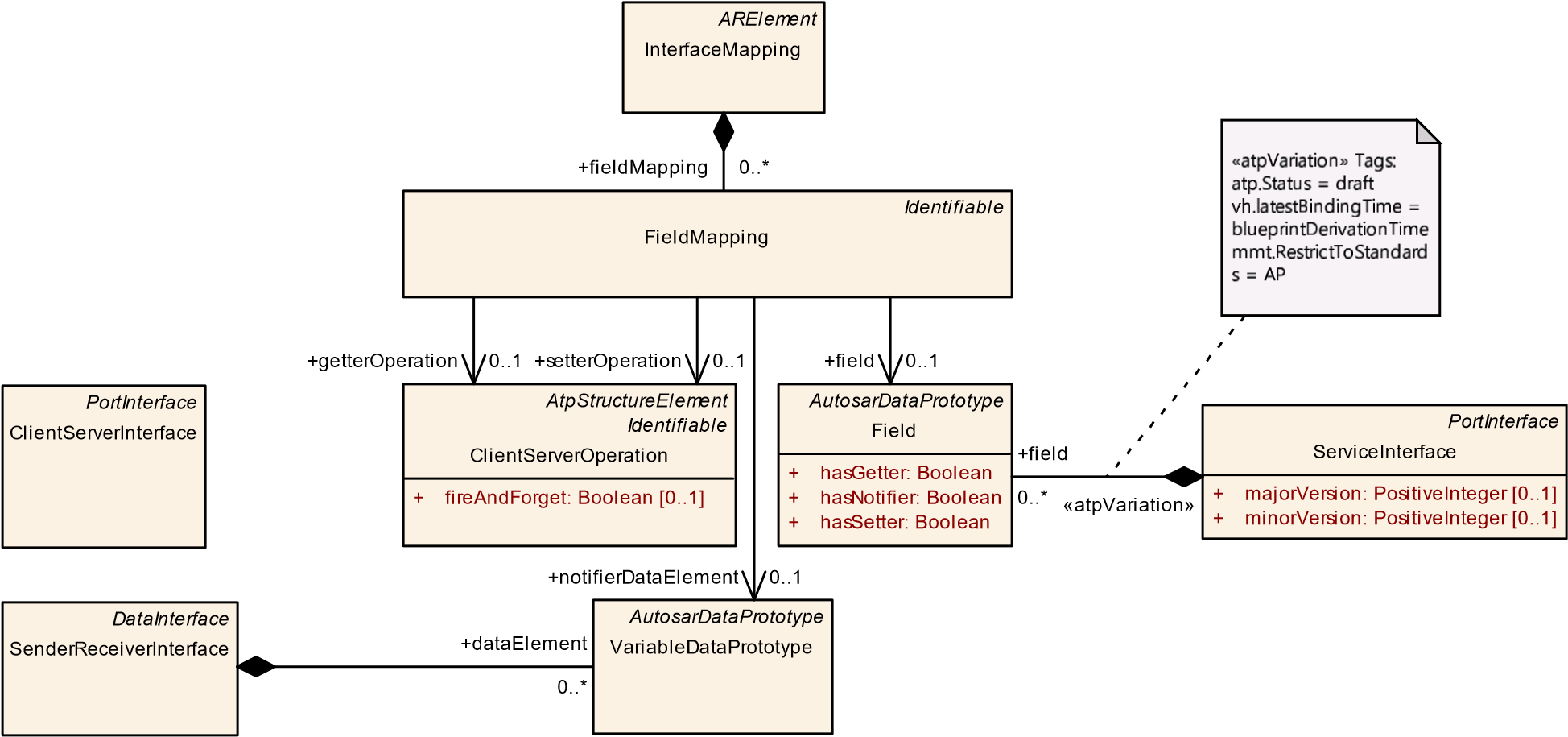
**[TPS\_MANI\_03113]**{DRAFT} **Mapping between a field and elements of Classic Platform PortInterfaces** dThe mapping between a field located in a ServiceInterface and elements of Classic Platform PortInterfaces is provided by the class FieldMapping. The field notifier in the classic platform is represented by a dataElement that is located in a SenderReceiverInterface. The getter and setter methods in the classic platform are represented by operations that are located in a ClientServerInterface.c*(RS\_MANI\_00026)*

**[constr\_3367]**{DRAFT} **FieldMapping.notifierDataElement reference** dThe

FieldMapping shall only contain the notifierDataElement reference if the hasNotifier attribute in the referenced field is set to true.c*()*

**[constr\_3368]**{DRAFT} **FieldMapping.getterOperation reference** dThe FieldMapping shall only contain the getterOperation reference if the hasGetter attribute in the referenced field is set to true.c*()*

**[constr\_3369]**{DRAFT} **FieldMapping.setterOperation reference** dThe FieldMapping shall only contain the setterOperation reference if the hasSetter attribute in the referenced field is set to true.c*()*



**Figure 5.13: Mapping between a field and elements of Classic Platform PortInterfaces**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **FieldMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign | | | |
| ***Note*** | Mapping of a Field that is located in a ServiceInterface to ClientServerOperations that represent the getter and setter methods and to a VariableDataPrototype that represents the notifier in the Field.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| field | Field | 0..1 | ref | Reference to a field that is located in a ServiceInterface.  **Tags:**atp.Status=draft |
| getterOperation | ClientServerOperation | 0..1 | ref | Reference to a ClientServerOperation that represents the getter Method in the Field.  **Tags:**atp.Status=draft |
| notifierData Element | VariableDataPrototype | 0..1 | ref | Reference to a VariableDataPrototype that represents the notifier in the Field.  **Tags:**atp.Status=draft |
| setterOperation | ClientServerOperation | 0..1 | ref | Reference to a ClientServerOperation that represents the setter Method in the Field.  **Tags:**atp.Status=draft |

**Table 5.44: FieldMapping**

**5.4.4 FireAndForgetMapping**

In a fire and forget Message Exchange Pattern the consumer sends a message to a provider with no expectation of a response as described in chapter 3.4.5.1.

In Adaptive AUTOSAR the fire and forget method is described with a method where the value of attribute method.fireAndForget is set to true as defined by [TPS\_MANI\_01064].

In classic AUTOSAR a fire and forget method can not be described with a ClientServerOperation since a client-server call always has a response. Therefore, a VariableDataPrototype is used if the fire and forget method contains input arguments.

If the fire and forget method contains several input arguments then the VariableDataPrototype needs to be of type Structure that hosts one element for each argument of the fire and forget method. It is important that the order of elements in the Structure is the same as the order of ArgumentDataPrototypes within the ClientServerOperation.

This representation ensures that the SOME/IP serialization results in the same byte stream as in the Adaptive Platform where all arguments which have the direction in are serialized according to the order of the ArgumentDataPrototypes within the ClientServerOperation.

If the fire and forget method is without any parameters a Trigger is used to describe such a method in classic AUTOSAR.

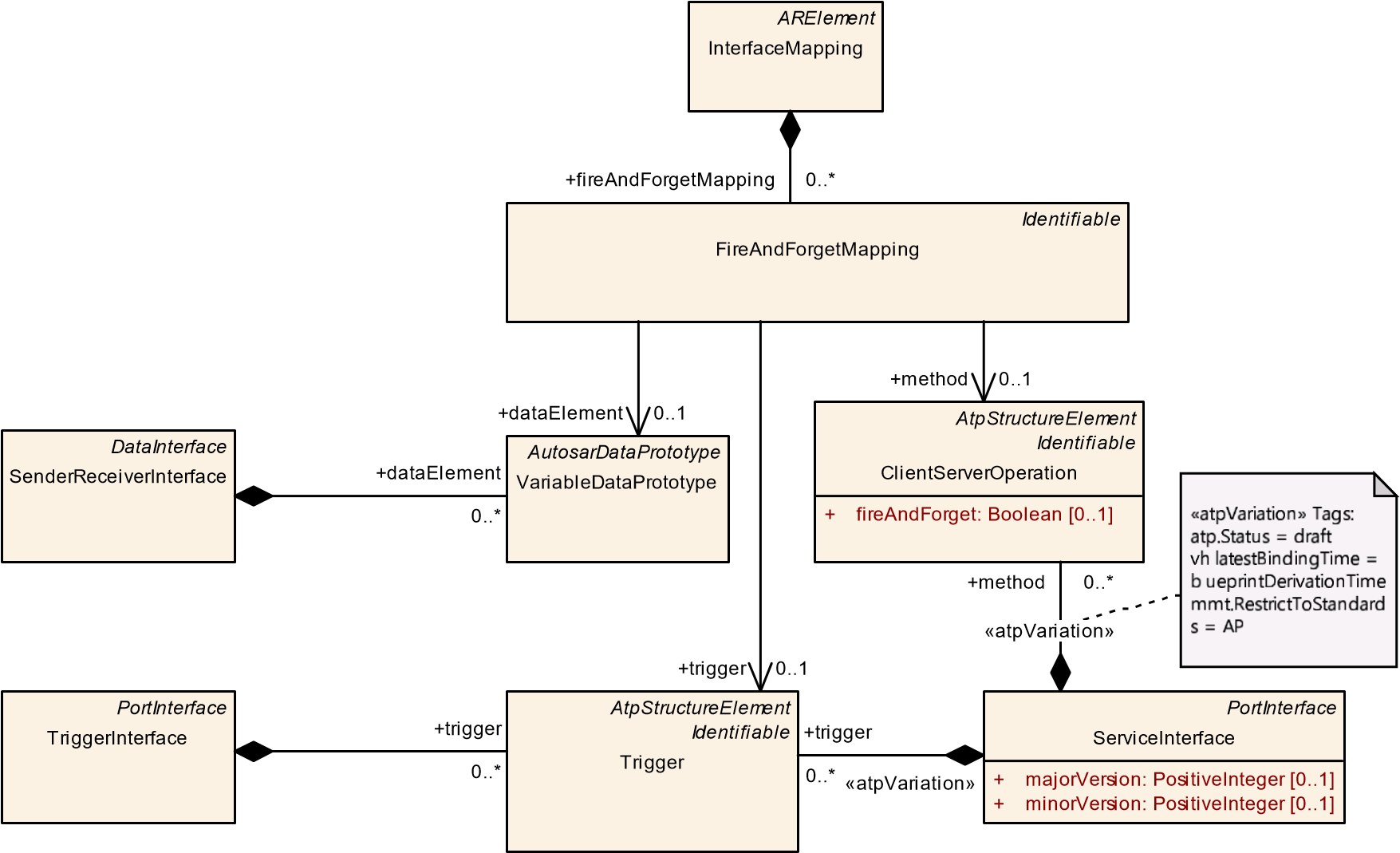
It is important that the SOME/IP MessageType is set to REQUEST\_NO\_RETURN if a fire and forget method is transmitted over SOME/IP.

**[TPS\_MANI\_03115]**{DRAFT} **Mapping between a fire and forget method and elements of Classic Platform PortInterfaces** dThe mapping between a method for which the value of attribute method.fireAndForget is set to true and elements of Classic Platform PortInterfaces is provided by the class FireAndForgetMapping.

If the fire and forget method is represented in the classic platform by a VariableDataPrototype then this dataElement is mapped to a method located in a ServiceInterface. If the fire and forget method is represented in the classic platform by a Trigger then this trigger is mapped to a method located in a ServiceInterface.c*(RS\_MANI\_00026)*

**[constr\_3371]**{DRAFT} **Mutually exclusive existence of FireAndForgetMapping.dataElement reference and FireAndForgetMapping.trigger reference** dA FireAndForgetMapping shall never reference a dataElement and a trigger at the same time.c*()*

**[constr\_3376]**{DRAFT} **FireAndForgetMapping shall reference only fire and forget methods** dA FireAndForgetMapping is only allowed to reference a ClientServerOperation in role method for which the value of attribute method. fireAndForget is set to true.c*()*



**Figure 5.14: Mapping between a fire and forget method and elements of Classic Platform**

**PortInterfaces**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **FireAndForgetMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign | | | |
| ***Note*** | Mapping of a Fire&Forget Method that is located in a ServiceInterface to a VariableDataPrototype in a SenderReceiverInterface or to a Trigger in a TriggerInterface.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| dataElement | VariableDataPrototype | 0..1 | ref | Reference to a VariableDataPrototype that is located in a  SenderReceiverInterface in case that the Fire&Forget Method is represented by this VariableDataPrototype.  **Tags:**atp.Status=draft |
| method | ClientServerOperation | 0..1 | ref | Reference to a Fire&Forget Method that is located in a ServiceInterface.  **Tags:**atp.Status=draft |
| trigger | Trigger | 0..1 | ref | Reference to a Trigger that is located in a TriggerInterface in case that the Fire&Forget Method is represented by this Trigger.  **Tags:**atp.Status=draft |

**Table 5.45: FireAndForgetMapping**

# 6 Sub-System Design

## 6.1 Overview

The nature of the *AUTOSAR adaptive platform* as a platform for deploying software units in the field implies that the software units that can be installed in the field need some design-support upfront.

More specifically, the software units that can be deployed in the field typically represent some sort of more or less self-contained driving function.

In other words, the design support for this purpose need to be tailored to facilitate the design of application-level software that communicates with other application level software.

It is assumed that one of the first steps in such a design is the definition of services that are provided and services that are required by the driving function under development.

Such a definition of required and provided services can be used as an input into the design of other such driving functions and, over time, a view of the communication on the level of driving functions is rendered.

It is further assumed that the communication view of the driving functions is mostly of interest for an OEM and the individual driving functions may be sub-contracted to tier-1 suppliers.

This means that for the tier-1 supplier the list of provided and required services of the driving function represents a technical contract against which the function shall be developed.

On design level, meta-class SoftwareClusterDesign is used for the formalization of software that might represent such a driving function. In other words, it is assumed that a workflow exists where the design of a certain functionality on the AUTOSAR adaptive platform starts with the creation of a SoftwareClusterDesign.

In this case, it is further assumed that the definition of the required and provided service instances for the respective functionality is a good starting point for the development.

Please note that SoftwareClusterDesign supports an arbitrary complexity of software and is therefore not bound to the design of, e.g. a single driving function.

## 6.2 Software Cluster Design

**Disclaimer: this chapter is out of date and will be updated in a later release.**

**[TPS\_MANI\_01112]**{DRAFT} **Semantics of SoftwareClusterDesign** dThe existence of a SoftwareClusterDesign represents the formalized response to requirements that have initially been formulated by an OEM and that may be enriched as the development of the software progresses.

Finally, the SoftwareClusterDesign shall be taken by the integration as a further input to the definition of the result of the integration step: the definition of the SoftwareCluster.c*(RS\_MANI\_00035)*

Just to be sure, the SoftwareClusterDesign is not intended to be uploaded to the target platform. It is just an early form of the final SoftwareCluster that indeed gets uploaded. The existence of the SoftwareClusterDesign is motivated from the methodological point of view.

**[constr\_1557]**{DRAFT} **Standardized values of SoftwareClusterDesign.category and SoftwareCluster.category** dThe AUTOSAR standard reserves the following values of attribute SoftwareClusterDesign.category and SoftwareCluster.category:

* ROOT\_SOFTWARE\_CLUSTER
* SUB\_SOFTWARE\_CLUSTER c*()*

**[TPS\_MANI\_01161]**{DRAFT} **Impact of values of category on the semantics of**

**SoftwareClusterDesign** dA SoftwareClusterDesign of category ROOT\_SOFTWARE\_CLUSTER may refer to other SoftwareClusterDesigns of category SUB\_SOFTWARE\_CLUSTER in the role subSoftwareCluster and thereby offer a way to further break down the creation of a SoftwareClusterDesign.c*(RS\_MANI\_00035)*

**[constr\_1558]**{DRAFT} **Existence of SoftwareClusterDesign.diagnosticAddress** dThe aggregation of SoftwareClusterDiagnosticAddress at SoftwareClusterDesign in the role diagnosticAddress shall only exist if the value of SoftwareClusterDesign.category is set to ROOT\_SOFTWARE\_CLUSTER.c*()*

**[constr\_1559]**{DRAFT} **Existence of SoftwareClusterDesign.subSoftwareCluster** dThe Reference from SoftwareClusterDesign to itself in the role subSoftwareCluster shall only exist if the value of SoftwareClusterDesign. category is set to ROOT\_SOFTWARE\_CLUSTER.c*()*

**[constr\_1560]**{DRAFT} **Usage of SoftwareClusterDesign.requiredARElement**dThe reference SoftwareClusterDesign.requiredARElement shall not be used to refer to another SoftwareClusterDesign or even SoftwareCluster.c*()*

Rationale for the existence of [constr\_1560]: dedicated references are defined for the purpose of referring to SoftwareClusterDesigns.

**[TPS\_MANI\_01211]**{DRAFT} **Specification of executable software within SoftwareClusterDesign**dOne of the most prominent contents of an uploadable software package is the reference to the executable software.

Within the definition of a SoftwareClusterDesign, this reference is implicitly given by means of the reference SoftwareCluster.containedProcess.

The target of SoftwareClusterDesign.containedProcess is a ProcessDesign that represents the design-level representation of an instance (formalized as Process) of the corresponding executable program (the software image), formalized as Executablec*(RS\_MANI\_00035)*

*AtpClassifier*

SoftwareClusterDesign

DiagnosticContributionSet

*UploadablePackageElement*

*PackageableElement*

*ARElement*

*SoftwareClusterDiagnosticAddress*

addressSemantics: SoftwareClusterDiagnosticAddressSemanticsEnum

+

SoftwareClusterDoipDiagnosticAddress

+

diagnosticAddress: PositiveInteger

[0..1]

«enumeration»

SoftwareClusterDiagnosticAddressSemanticsEnum

physicalAddress

functionalAddress

*PackageableElement*

*FibexElement*

*AtpStructureElement*

MachineDesign

+

[0..1]

accessControl: AccessControlEnum

pncPrepareSleepTimer: TimeValue

+

[0..1]

[0..1]

+

pnResetTimer: TimeValue

ProcessDesign

«atpSplitable»

diagnosticContribution

+

0..\*

«atpSplitable»

+

requiredARElement

0..\*

«atpSplitable»

diagnosticAddress

+

0..\*

«atpSplitable»

+

requiredPackageElement

0..\*

«atpUriDef»

intendedTargetMachine

+

0..1

0..\*

dependsOn

+

«atpSplitable»

+

subSoftwareCluster

0..\*

«atpSplitable»

requiredFibexElement

+

0..\*

«atpSplitable»

+

containedProcess

0..\*

**Figure 6.1: Modeling of SoftwareClusterDesign**

**[TPS\_MANI\_01113]**{DRAFT} **Semantics of SoftwareClusterDesign.diagnosticAddress** dThe existence of the attribute SoftwareClusterDesign.diagnosticAddress can be used to express information about the distribution of diagnostic addresses even in a very early stage of development, i.e. this is typically done by an OEM.

This includes the ability to specify multiple (i.e. several functional plus one physical) diagnostic addresses, thus the multiplicity of diagnosticAddress is set to 0..\*.c*(RS\_MANI\_00035)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SoftwareClusterDesign** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign | | | |
| ***Note*** | This meta-class represents the ability for the OEM to design the grouping of software uploadable to a specific target Machine.  **Tags:**  atp.Status=draft  atp.recommendedPackage=SoftwareClusterDesigns | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| contained Process | ProcessDesign | \* | ref | This reference represent the ProcessDesigns contained in the enclosing SoftwareCluster.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=containedProcess  atp.Status=draft |
| dependsOn | SoftwareClusterDesign | \* | ref | The owner SoftwareClusterDesign depends on the referenced SoftwareClusterDesign  **Tags:**atp.Status=draft |
| diagnostic Address | SoftwareCluster  DiagnosticAddress | \* | aggr | This aggregation is used to specify the diagnostic address.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=diagnosticAddress atp.Status=draft |
| diagnostic  Contribution | DiagnosticContribution  Set | \* | ref | This reference identifies the corresponding collection of DiagnosticContributionSet.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=diagnosticContribution atp.Status=draft |
| intendedTarget Machine | MachineDesign | 0..1 | ref | This reference can be taken to identify the Machine Design for which the final SoftwareCluster shall be developed.  **Stereotypes:** atpUriDef **Tags:**atp.Status=draft |
| required  ARElement | ARElement | \* | ref | This reference represents the collection of ARElements that are required for the completeness of the definition of the SoftwareCluster.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=requiredARElement atp.Status=draft |
| requiredFibex Element | FibexElement | \* | ref | This reference represents the collection of fibexElements that are required for the completeness of the definition of the SoftwareCluster.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=requiredFibexElement atp.Status=draft |

5 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SoftwareClusterDesign** |  |  |  |
| required Package  Element | UploadablePackage  Element | \* | ref | This reference points to uploadable elements that have been identified as relevant in the context of the enclosing SoftwareClusterDesign.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=requiredPackageElement atp.Status=draft |
| root  Composition | RootSwClusterDesign  ComponentPrototype | 0..1 | aggr | This aggregation represents the design of the software inside the SwClusterDesign terms of the communication endpoints.  **Tags:**atp.Status=draft |
| subSoftware Cluster | SoftwareClusterDesign | \* | ref | This reference is used to identify the sub-SoftwareCluster Designs of an "umbrella" SoftwareClusterDesign.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=subSoftwareCluster atp.Status=draft |

**Table 6.1: SoftwareClusterDesign**

**[TPS\_MANI\_01117]**{DRAFT} **Semantics of SoftwareClusterDesign.intendedTargetMachine** dThe specification of SoftwareClusterDesign.intendedTargetMachine allows for focusing the specification of an uploadable software package to a specific MachineDesign from early phases of a development project.c*(RS\_MANI\_00035)*

Please note that SoftwareCluster doesn’t have a dedicated reference to the target Machine.

This relation is expressed by means of a reference to Process that in turn can be mapped to a dedicated Machine by means of a ProcessToMachineMapping. In this context, [constr\_1536] applies.

**[TPS\_MANI\_01118]**{DRAFT} **Relation between SoftwareClusterDesign and DiagnosticContributionSet** dAn important aspect of the definition of a SoftwareClusterDesign is the question what diagnostic extract shall be associated with the SoftwareClusterDesign.

For this purpose, a reference from SoftwareClusterDesign to DiagnosticContributionSet in the role diagnosticContribution is provided.

In an early stage of the development process, it is intentionally made possible to reference multiple DiagnosticContributionSets in order to support the decentralized (e.g. partly done by OEM and partly done by supplier) configuration of the diagnostics stack.c*(RS\_MANI\_00035)*

**[TPS\_MANI\_01189]**{DRAFT} **Software Cluster and DiagnosticContribution-**

**Set.category** dA DiagnosticContributionSet used in the context of a

SoftwareCluster shall set the value of attribute category to DIAGNOSTICS\_SWCL\_EXTRACT.c*(RS\_MANI\_00035)*

**[constr\_1562]**{DRAFT} **Existence of SoftwareClusterDesign.diagnosticContribution** dThe existence of the reference SoftwareClusterDesign.diagnosticContribution is limited to SoftwareClusterDesigns where attribute category is set to the value ROOT\_SOFTWARE\_CLUSTER.c*()*

Rationale for the existence of [constr\_1562]: the definition of the diagnostic behavior is limited to the root level of a structure of SoftwareClusterDesigns in the same spirit that caused the existence of [constr\_1558].

Please mind the intentionally introduced difference between SoftwareCluster and SoftwareClusterDesign in terms of the relation to DiagnosticContributionSet.

In other words, the multiplicity of the references to DiagnosticContributionSet intentionally differ.

As already explained, the SoftwareClusterDesign shall support the decentralized configuration of the DiagnosticContributionSet while the SoftwareCluster requires the existence of a final (merged) DiagnosticContributionSet.

**[TPS\_MANI\_01119]**{DRAFT} **Reference to model elements from SoftwareClusterDesign**dSoftwareClusterDesign has the ability to define the following references to model elements relevant for the definition of an uploadable software package:

* references to meta-classes derived from UploadablePackageElement are formalized by way of SoftwareClusterDesign.requiredPackageElement.
* references to meta-classes derived from ARElement are formalized by way of SoftwareClusterDesign.requiredARElement.
* references to meta-classes derived from FibexElement are formalized by way of SoftwareClusterDesign.requiredFibexElement.

c*(RS\_MANI\_00035)*

Please note that the conversion of a SoftwareClusterDesign to a SoftwareCluster is not formalized by AUTOSAR. This step can be done by a tool at the discretion of the integrator.

In other words, in some cases it may be applicable to do this conversion relatively early in the development project while other projects may require to keep the SoftwareClusterDesign around for a longer period in time.

**[TPS\_MANI\_01310]**{DRAFT} **Semantics of SoftwareClusterDesign.dependsOn**dThe reference SoftwareClusterDesign.dependsOn can be used to prepare the definition of dependencies that exist between SoftwareClusters already on the design level.c*(RS\_MANI\_00035)*

In other words, the definition of SoftwareClusterDesign.dependsOn is certainly not required to build a consistent model of a SoftwareClusterDesign. The reference can only be used to “front-load” the formalization of dependencies that may later happen on the level of the design of SoftwareClusters.

## 6.3 Provided and required Services of Software Cluster Design

In order to support the definition of required and provided services early in the design of a SoftwareCluster[[14]](#footnote-14), AUTOSAR supports the definition of a RootSwClusterDesignComponentPrototype in the context of a given SoftwareClusterDesign.

The RootSwClusterDesignComponentPrototype itself refers to a SwComponentType that in turn exposes PortPrototypes to the outside world.

Note that for the specific case of the RootSwClusterDesignComponentPrototype it is expected that the referenced SwComponentType represents a CompositionSwComponentType without any further detailing. A detailing is obviously unnecessary because the only purpose is the exposure of PortPrototypes to which AdaptivePlatformServiceInstances can be mapped.

A dedicated mapping class, ServiceInstanceToSwClusterDesignPortPrototypeMapping, is defined to support the creation of the described relation between PortPrototype and AdaptivePlatformServiceInstance.

*ARElement*

*ServiceInstanceToSwClusterDesignPortPrototypeMapping*

*UploadablePackageElement*

*AdaptivePlatformServiceInstance*

*ProvidedApServiceInstance*

PPortPrototype

*PortPrototype*

*AbstractProvidedPortPrototype*

ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping

«instanceRef»

+

providedPortPrototype

0..1

+

providedServiceInstance

0..1

**Figure 6.2: Modeling of the ProvidedServiceInstanceToSwClusterDesignPPort-**

**PrototypeMapping**

**[TPS\_MANI\_01275]**{DRAFT} **Semantics of meta-class ServiceInstanceToSwClusterDesignPortPrototypeMapping** dThe software-component used to type the RootSwClusterDesignComponentPrototype typically exposes a set of PortPrototypes to the outside world.

These PortPrototypes could be used for the specification of required and provided service instances. For this purpose, meta-class ServiceInstanceToSwClusterDesignPortPrototypeMapping is used.c*(RS\_MANI\_00011)*

In Figure 6.4, the ServiceInstanceToSwClusterDesignPortPrototypeMapping is represented by a block labeled “mapping” with a circled 1. The block labeled “mapping” with a circled 2 represents the CompositionPortToExecutablePortMapping, as described in section 6.4.

*ARElement*

*ServiceInstanceToSwClusterDesignPortPrototypeMapping*

*UploadablePackageElement*

*AdaptivePlatformServiceInstance*

RequiredServiceInstanceToSwClusterDesignRPortPrototypeMapping

RPortPrototype

*PortPrototype*

*AbstractRequiredPortPrototype*

*RequiredApServiceInstance*

«instanceRef»

+

requiredPortPrototype

0..1

+

requiredServiceInstance

0..1

**Figure 6.3: Modeling of the RequiredServiceInstanceToSwClusterDesignRPort-**

**PrototypeMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **RootSwClusterDesignComponentPrototype** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution | | | |
| ***Note*** | This meta-class represents the ability to define the service endpoints in the scope of a SwClusterDesign.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| applicationType | SwComponentType | 1 | ref | This SwComponentType acts as the Type of the RootSw ClusterDesignComponentPrototype.  **Tags:**atp.Status=draft |

**Table 6.2: RootSwClusterDesignComponentPrototype**

SoftwareClusterDesign

SwComponentType (referenced by

RootSwClusterDesignComponentPrototype)

ProvidedServiceInstance 1

RequiredServiceInstance 45

Executable A

SwComponentType (referenced by

RootSwComponentPrototype)

Mapping

Mapping

Mapping

1

1

2

Executable B

SwComponentType (referenced by

RootSwComponentPrototype)

The position (left/right) of PortPrototypes does

not make any implications on their directions.

Mapping

2

ProcessDesign

Mapping

2

ProcessDesign

ProcessDesign

**Figure 6.4: Modeling of mappings in the context of SoftwareClusterDesign**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***ServiceInstanceToSwClusterDesignPortPrototypeMapping*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow | | | |
| ***Note*** | This abstract meta-class represents the ability to assign a transport-layer-dependent ServiceInstance to a PortPrototype in the context of the SoftwareClusterDesign. With this mapping it is possible to define the list of provided and required AdaptivePlatformServiceInstances in the scope of the SoftwareCluster Design.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Subclasses*** | ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping, RequiredServiceInstanceToSw ClusterDesignRPortPrototypeMapping | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 6.3: ServiceInstanceToSwClusterDesignPortPrototypeMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **RequiredServiceInstanceToSwClusterDesignRPortPrototypeMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow | | | |
| ***Note*** | This concrete meta-class represents the ability to assign a transport-layer-dependent RequiredService Instance to an RPortPrototype in the context of the SoftwareClusterDesign. With this mapping it is possible to define the list of provided and required AdaptivePlatformServiceInstances in the scope of the SoftwareClusterDesign.  **Tags:** atp.Status=draft  atp.recommendedPackage=ServiceInstanceToSwClusterDesignPortPrototypeMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable*  *Element*, *Referrable*, *ServiceInstanceToSwClusterDesignPortPrototypeMapping* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| requiredPort Prototype | RPortPrototype | 0..1 | iref | This reference identifies the applicable PortPrototype in the scope of the SwClusterDesign.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**RPortPrototypeInSoftware ClusterDesignInstanceRef |
| requiredService Instance | RequiredApService  Instance | 0..1 | ref | Reference to a RequiredServiceInstance mapped to a given RPortPrototype in the scope of the SwCluster Design.  **Tags:**atp.Status=draft |

**Table 6.4: RequiredServiceInstanceToSwClusterDesignRPortPrototypeMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow | | | |
| ***Note*** | This concrete meta-class represents the ability to assign a transport-layer-dependent ProvidedService Instance to a PPortPrototype in the context of the SoftwareClusterDesign. With this mapping it is possible to define the list of provided and required AdaptivePlatformServiceInstances in the scope of the Software ClusterDesign.  **Tags:**  atp.Status=draft  atp.recommendedPackage=ServiceInstanceToSwClusterDesignPortPrototypeMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable*  *Element*, *Referrable*, *ServiceInstanceToSwClusterDesignPortPrototypeMapping* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |

5

4

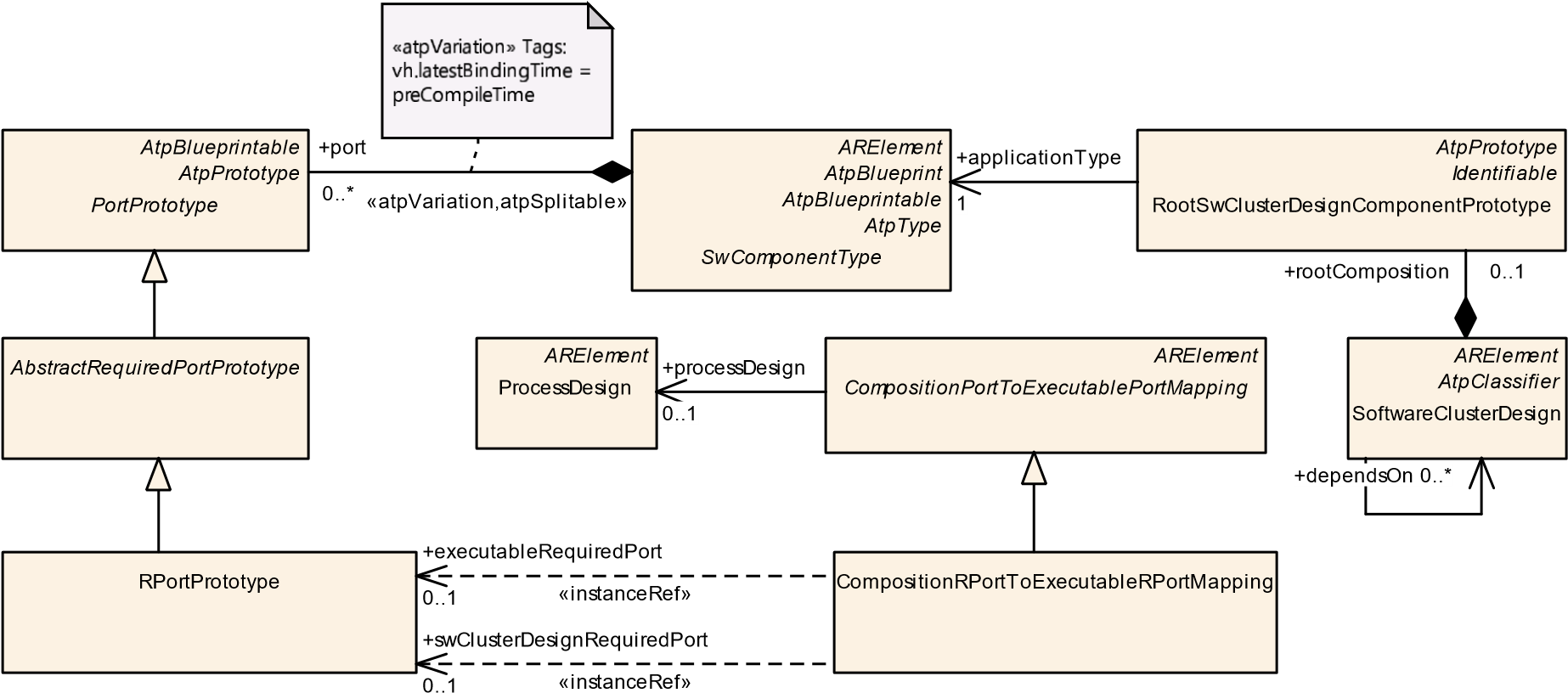
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping** | | | |
| providedPort Prototype | PPortPrototype | 0..1 | iref | This reference identifies the applicable PortPrototype in the scope of the SwClusterDesign.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**PPortPrototypeInSoftware ClusterDesignInstanceRef |
| providedService Instance | ProvidedApService  Instance | 0..1 | ref | Reference to a ProvidedServiceInstance mapped to a given PPortPrototype in the scope of the SwCluster Design.  **Tags:**atp.Status=draft |

**Table 6.5: ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping**

## 6.4 Mapping of Services to Executables

A typical next step in the design workflow could be to decide about the modeling of

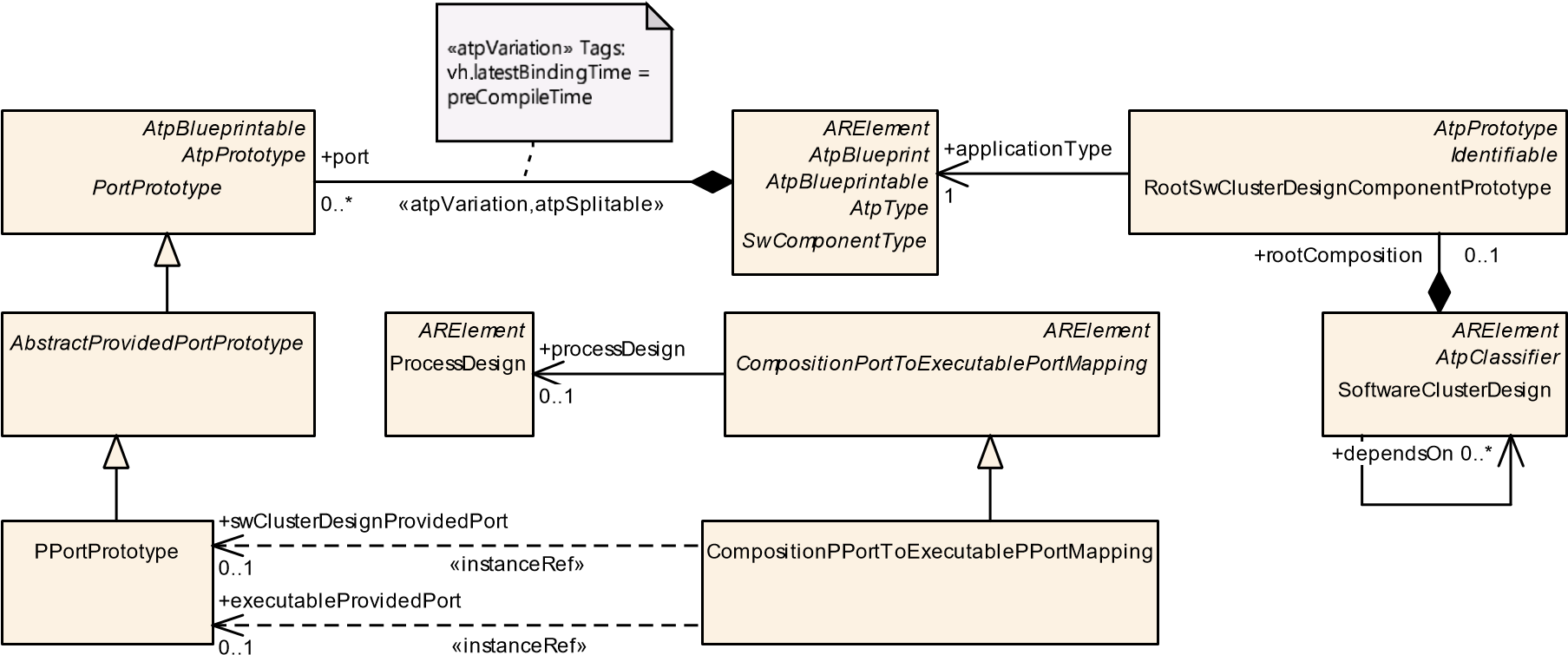
Executables inside the SoftwareClusterDesign. The PortPrototypes used in the modeling of an Executable actually implement the endpoints to which required and provided service instances shall be mapped.



**Figure 6.5: Modeling of the RootSwClusterDesignComponentPrototype and the CompositionRPortToExecutableRPortMapping**

**[TPS\_MANI\_01276]**{DRAFT} **Semantics of CompositionRPortToExecutableRPortMapping and CompositionPPortToExecutablePPortMapping** dIn the context of the creation of an SoftwareClusterDesign, it is not possible to already define the actual mapping of PortPrototypes to AdaptivePlatformServiceInstances.

To counter this issue, and as an additional guidance for the later creation of the actual ServiceInstanceToPortPrototypeMappings it is possible to create another mapping inside the scope of the SoftwareClusterDesign that maps the PortPrototypes defined in the context of the RootSwClusterDesignComponentPrototype to the refined PortPrototypes defined in the context of Executables.c *(RS\_MANI\_00011)*



**Figure 6.6: Modeling of the RootSwClusterDesignComponentPrototype and the CompositionPPortToExecutablePPortMapping**

This way, it is possible to retrace the design decisions on the level of the RootSwClusterDesignComponentPrototype one level deeper and provide a guidance for the creation of the ServiceInstanceToPortPrototypeMapping, as described in section 10.3.

**[TPS\_MANI\_01282]**{DRAFT} **Semantics of reference CompositionPortToExecutablePortMapping.processDesign**dThe reference CompositionPortToExecutablePortMapping.processDesign identifies the applicable ProcessDesign for the mapping. This reference therefore disambiguates the existence of multiple CompositionPortToExecutablePortMapping that refer to the exact same PortPrototype in the context of an Executable.c*(RS\_MANI\_00011)*

The statement made by [TPS\_MANI\_01282] is further explained in Figure 6.4. Two

CompositionPortToExecutablePortMapping refer to the same PortPrototype on the surface of the ExecutableB.

It is important to understand that each of these CompositionPortToExecutablePortMappings refer to a different ProcessDesign.

This means that, at run-time, the two CompositionPortToExecutablePortMappings apply to different instances of the ExecutableB launched as different Processes (that each, in turn, refer to one of the ProcessDesigns referenced by the ExecutableB).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***CompositionPortToExecutablePortMapping*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow | | | |
| ***Note*** | This abstract meta-class acts as a base class for the specification of a mapping between a PortPrototype owned by a RootSwClusterDesignComponentPrototype to a PortPrototype owned by a Component Prototype inside an Executable.rootSwComponentType.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Subclasses*** | CompositionPPortToExecutablePPortMapping, CompositionRPortToExecutableRPortMapping | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| processDesign | ProcessDesign | 0..1 | ref | This reference identifies the impacted ProcessDesign for this mapping. This allows for mapping multiple services to the same PortPrototype on an Executable by also referencing different ProcessDesigns.  **Tags:**atp.Status=draft |

**Table 6.6: CompositionPortToExecutablePortMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CompositionRPortToExecutableRPortMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow | | | |
| ***Note*** | This meta-class has the ability to associate an RPortPrototype defined in the context of a SwCluster Design to an RPortPrototype in the context of an Executable.  **Tags:**  atp.Status=draft  atp.recommendedPackage=CompositionPortToExecutablePortMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *CompositionPortToExecutablePortMapping*, *Identifiable*,  *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| executable  RequiredPort | RPortPrototype | 0..1 | iref | This reference identifies the applicable PortPrototype in the context on an Executable.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**RPortPrototypeIn  ExecutableInstanceRef |
| swCluster  DesignRequired  Port | RPortPrototype | 0..1 | iref | This reference identifies the applicable RPortPrototype in the context of the SwClusterDesign.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**RPortPrototypeInSoftware ClusterDesignInstanceRef |

**Table 6.7: CompositionRPortToExecutableRPortMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CompositionPPortToExecutablePPortMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow | | | |
| ***Note*** | This meta-class has the ability to associate a PPortPrototype defined in the context of a SwClusterDesign to a PPortPrototype in the context of an Executable.  **Tags:**  atp.Status=draft  atp.recommendedPackage=CompositionPortToExecutablePortMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *CompositionPortToExecutablePortMapping*, *Identifiable*,  *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **CompositionPPortToExecutablePPortMapping** | | | |
| executable  ProvidedPort | PPortPrototype | 0..1 | iref | This reference identifies the applicable PortPrototype in the context on an Executable.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**PPortPrototypeIn  ExecutableInstanceRef |
| swCluster  DesignProvided  Port | PPortPrototype | 0..1 | iref | This reference identifies the applicable PPortPrototype in the context of the SwClusterDesign.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**PPortPrototypeInSoftware ClusterDesignInstanceRef |

**Table 6.8: CompositionPPortToExecutablePPortMapping**

# 7 Machine Manifest

The Machine meta-class defines the entity on which one *Adaptive AUTOSAR Software Stack* is running with an operating system. The Machine may be physical or virtual.

Some aspects of the actual Machine are already available from the System Design (see chapter 5.2) at the MachineDesign. The information defined at the MachineDesign is available to the Machine as well since Machine has a reference to the MachineDesign in the role machineDesign (see figure 5.1).

The Machine is able to aggregate one or several Processors. And each Processor consists of one or several ProcessorCores.

Meta-class ProcessorCore provides attribute coreId that can be used e.g. in a bitmask to better control the utilization of processing resources.

**[constr\_1549]**{DRAFT} **Value of ProcessorCore.coreId** dThe value of ProcessorCore.coreId shall be unique in the context of the enclosing Processor.c*()* An overview of the Machine meta-class is sketched in Figure 7.1.

*ARElement*

*AtpStructureElement*

Machine

trustedPlatformExecutableLaunchBehavior: TrustedPlatformExecutableLaunchBehaviorEnum

+

*ServiceDiscoveryConfiguration*

*Identifiable*

*CommunicationConnector*

[0..1]

createEcuWakeupSource: Boolean

+

pncFilterArrayMask: PositiveInteger [0..\*] {ordered

+

}

*Identifiable*

*AdaptiveModuleInstantiation*

*Identifiable*

ProcessorCore

coreId: PositiveInteger

+

*Identifiable*

Processor

*AtpStructureElement*

*FibexElement*

MachineDesign

accessControl: AccessControlEnum

+

[0..1]

[0..1]

pncPrepareSleepTimer: TimeValue

+

[0..1]

pnResetTimer: TimeValue

+

TagWithOptionalValue

key: String

+

sequenceOffset: Integer

+

[0..1]

value: String

[0..1]

+

«enumeration»

TrustedPlatformExecutableLaunchBehaviorEnum

strictMode

monitorMode

noTrustedPlatformSupport

«atpSplitable»

+

moduleInstantiation

0..\*

«atpSplitable»

+

communicationConnector

0..\*

+

processor

1..\*

+

core

1..\*

«atpSplitable»

+

environmentVariable

0..\*

+

machineDesign

1

«atpSplitable»

+

serviceDiscoveryConfig

0..\*

**Figure 7.1: Overview about the content of the Machine configuration**

**[TPS\_MANI\_03035]**{DRAFT} **Content of the Machine configuration** dThe purpose of the Machine is to provide machine specific configuration settings.c*(RS\_MANI\_00020, RS\_MANI\_00021, RS\_MANI\_00022, RS\_MANI\_00023)*

**[TPS\_MANI\_01208]**{DRAFT} **Definition of environment variables in the scope of a Machine** dIt is possible to define environment variables in the scope of the entire Machine.

For this purpose the aggregation of TagWithOptionalValue in the role Machine. environmentVariable exists.

The name of the environment variable shall be specified by means of the attribute TagWithOptionalValue.key, the value can be modeled by means of TagWithOptionalValue.value.

This encloses the ability to define environment variables with empty values. For this purpose, the attribute TagWithOptionalValue.value shall simply be omitted.c *(RS\_MANI\_00022, RS\_MANI\_00023)*

Please note that the aggregation Machine.environmentVariable has been defined with the stereotype atpSplitable. The consequence of this modeling is that it is possible to contribute to the definition of environment variables from **different sources**.

As an example, assume two partial models (sketched in Listing 7.1 and 7.2) which both add a folder to the search path of a machine running on an adaptive platform ECU.

**<MACHINE>**

**<SHORT-NAME>**Machine**</SHORT-NAME>**

**<ENVIRONMENT-VARIABLES>**

**<TAG-WITH-OPTIONAL-VALUE>**

**<KEY>**PATH**</KEY>**

**<SEQUENCE-OFFSET>**10**</SEQUENCE-OFFSET>**

**<VALUE>**/usr/application-x/bin**</VALUE>**

**</TAG-WITH-OPTIONAL-VALUE>**

**</ENVIRONMENT-VARIABLES>**

**</MACHINE>**

**Listing 7.1: Example for the definition of environmentVariable (file 1)**

**<MACHINE>**

**<SHORT-NAME>**Machine**</SHORT-NAME>**

**<ENVIRONMENT-VARIABLES>**

**<TAG-WITH-OPTIONAL-VALUE>**

**<KEY>**PATH**</KEY>**

**<SEQUENCE-OFFSET>**20**</SEQUENCE-OFFSET>**

**<VALUE>**/usr/application-y/bin**</VALUE>**

**</TAG-WITH-OPTIONAL-VALUE>**

**</ENVIRONMENT-VARIABLES>**

**</MACHINE>**

**Listing 7.2: Example for the definition of environmentVariable (file 2)**

Merging the splitable elements from the partial models results in the following content. Please note that the merged model exists only internally in the AUTOSAR tool. The Listing 7.3 is therefore shown **only for illustration**.

**<MACHINE>**

**<SHORT-NAME>**Machine**</SHORT-NAME>**

**<ENVIRONMENT-VARIABLES>**

**<TAG-WITH-OPTIONAL-VALUE>**

**<KEY>**PATH**</KEY>**

**<SEQUENCE-OFFSET>**10**</SEQUENCE-OFFSET>**

**<VALUE>**/usr/application-x/bin**</VALUE>**

**</TAG-WITH-OPTIONAL-VALUE>**

**<TAG-WITH-OPTIONAL-VALUE>**

**<KEY>**PATH**</KEY>**

**<SEQUENCE-OFFSET>**20**</SEQUENCE-OFFSET>**

**<VALUE>**/usr/application-y/bin**</VALUE>**

**</TAG-WITH-OPTIONAL-VALUE>**

**</ENVIRONMENT-VARIABLES>**

**</MACHINE>**

**Listing 7.3: Example for the definition of environmentVariable (merged)**

The generator for the target machine configuration may use this information to create the following environment variable:

PATH=/usr/application-x/bin;/usr/application-y/bin;

**[TPS\_MANI\_01273]**{DRAFT} **Support for trusted Platform** dIf attribute Machine. trustedPlatformExecutableLaunchBehavior is set to a value that is different from noTrustedPlatformSupport then features of the "trusted platform" are activated, depending on the concrete value of Machine.trustedPlatformExecutableLaunchBehavior.c*(RS\_MANI\_00022)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Machine** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest | | | |
| ***Note*** | Machine that represents an Adaptive Autosar Software Stack.  **Tags:**  atp.Status=draft  atp.recommendedPackage=Machines | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| default Application  Timeout | EnterExitTimeout | 0..1 | aggr | This aggration defines a default timeout in the context of a given Machine with respect to the launching and termination of applications.  **Tags:**atp.Status=draft |
| environment Variable | TagWithOptionalValue | \* | aggr | This aggregation represents the collection of environment variables that shall be added to the environment defined on the level of the enclosing Machine.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=environmentVariable, environment Variable.variationPoint.shortLabel atp.Status=draft |
| machineDesign | MachineDesign | 1 | ref | Reference to the MachineDesign this Machine is implementing.  **Tags:**atp.Status=draft |
| module  Instantiation | AdaptiveModule  Instantiation | \* | aggr | Configuration of Adaptive Autosar module instances that are running on the machine.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=moduleInstantiation.shortName atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Machine** |  |  |  |
| processor | Processor | 1..\* | aggr | This represents the collection of processors owned by the enclosing machine.  **Tags:**atp.Status=draft |
| secure  Communication  Deployment | SecureCommunication  Deployment | \* | aggr | Deployment of secure communication protocol configuration settings to crypto module entities.  **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=secureCommunicationDeployment.short  Name  atp.Status=draft |
| trustedPlatform Executable  LaunchBehavior | TrustedPlatform  ExecutableLaunch  BehaviorEnum | 1 | attr | This attribute controls the behavior of how authentication affects the ability to launch for each Executable.  **Tags:**atp.Status=draft |

**Table 7.1: Machine**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Processor** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest | | | |
| ***Note*** | This represents a processor for the execution of an AUTOSAR adaptive platform  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| core | ProcessorCore | 1..\* | aggr | This represents the collection of cores owned by the enclosing processor.  **Tags:**atp.Status=draft |

**Table 7.2: Processor**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ProcessorCore** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest | | | |
| ***Note*** | This meta-class represents the ability to model a processor core for the execution of an AUTOSAR adaptive platform.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| coreId | PositiveInteger | 1 | attr | This attribute represents a numerical value assigned to the specific core. The value can be taken e.g. for use in a bitmask.  **Tags:**atp.Status=draft |

**Table 7.3: ProcessorCore**

|  |  |
| --- | --- |
| ***Enumeration*** | **TrustedPlatformExecutableLaunchBehaviorEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest |
| ***Note*** | This enumeration provides options for controlling the behavior of how authentication affects the ability to launch an Executable. **Tags:**atp.Status=draft |
| ***Literal*** | ***Description*** |

5

|  |  |
| --- | --- |
| ***Enumeration*** | **TrustedPlatformExecutableLaunchBehaviorEnum** |
| monitorMode | An Executable shall always launch, even if the corresponding authentication fails  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |
| noTrustedPlatform Support | This value shall be used if there is no TrustedPlatform support on the Machine  **Tags:**  atp.EnumerationLiteralIndex=2 atp.Status=draft |
| strictMode | An Executable shall not launch if the corresponding authentication fails.  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |

**Table 7.4: TrustedPlatformExecutableLaunchBehaviorEnum**

## 7.1 Process To Machine Mapping

**7.1.1 General Modeling Approach**

**[TPS\_MANI\_03147]**{DRAFT} **Mapping of a Process to a Machine**dThe meta-class ProcessToMachineMapping provides the ability to map a Process to a Machine.c *(RS\_MANI\_00006)*

**[constr\_1553]**{DRAFT} **Restriction for ProcessToMachineMapping** dThe following restrictions apply for the usage of ProcessToMachineMapping:

1. Each combination of Process and Machine shall only be referenced by one ProcessToMachineMapping in the role process or machine.
2. Each Process shall only be referenced by a single ProcessToMachineMapping in the role process.

### c()

Please note that [constr\_1553] does not imply that a given Machine shall only be referenced by a single ProcessToMachineMapping. It only says that one Process shall only be mapped once, to exactly one Machine.

**[constr\_5004]**{DRAFT} **Mapping of a Process to a Machine is mandatory in the Execution Manifest** dEach Process shall be mapped by a ProcessToMachineMapping to one Machine.c*()*

[constr\_5004] means that a formal description of the assignment of a Process to a Machine shall be provided in the Execution Manifest, even though the Manifest will be uploaded to the Machine in combination with other artifacts to which the Manifest applies. The formal ProcessToMachineMapping was introduced because it is useful in the processing of the model in many cases.

Please note that according to the AUTOSAR Methodology the Execution Manifest is created on the basis of an existing Machine Manifest and therefore the link to the Machine can always be created in the Execution Manifest.

*ARElement*

*AtpStructureElement*

Machine

trustedPlatformExecutableLaunchBehavior

+

:

TrustedPlatformExecutableLaunchBehaviorEnum

*AbstractExecutionContext*

Process

+

functionClusterAffiliation: String

[0..1]

numberOfRestartAttempts: PositiveInteger

[0..1]

+

[0..1]

preMapping: Boolean

+

*UploadablePackageElement*

ProcessToMachineMappingSet

*Identifiable*

ProcessToMachineMapping

+

[0..1]

persistencyCentralStorageURI: UriString

EnterExitTimeout

:

enterTimeoutValue

+

TimeValue [0..1]

+

:

exitTimeoutValue

TimeValue [0..1]

*Identifiable*

ProcessorCore

coreId: PositiveInteger

+

*Identifiable*

Processor

*ARElement*

ProcessDesignToMachineDesignMapping

processor

+

1..\*

+

core

1..\*

+

shallRunOn

0..\*

design

+

0..1

+

shallNotRunOn

0..\*

+

processToMachineMapping

0..\*

process

+

1

+

defaultApplicationTimeout

0..1

+

machine

0..1

**Figure 7.2: Mapping of a Process to a Machine**

**[constr\_10090]**{DRAFT} **Existence of ProcessToMachineMapping.persistencyCentralStorageURI** dAttribute ProcessToMachineMapping.persistencyCentralStorageURI shall exist if the Process referenced in the role ProcessToMachineMapping.process is also referenced by at least one PersistencyPortPrototypeToDeploymentMapping in the role process **at the time when the manifest is complete**.c*()*

If, in a later AUTOSAR release, usages of FunctionalClusterInteractsWithFunctionalClusterMapping were developed with the persistency functional cluster on the provider end, then the [constr\_10090] would have to be extended to also cover such cases.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ProcessToMachineMappingSet** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest | | | |
| ***Note*** | This meta-class acts as a bucket for collecting ProcessToMachineMappings.  **Tags:**  atp.Status=draft  atp.recommendedPackage=ProcessToMachineMappings | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable*  *Element*, *Referrable*, *UploadablePackageElement* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ProcessToMachineMappingSet** | |  |  |
| processTo Machine  Mapping | ProcessToMachine  Mapping | \* | aggr | This represents the collection of ProcessToMachine Mappings of the enclosing ProcessToMachineMapping Set.  **Tags:**atp.Status=draft |

**Table 7.5: ProcessToMachineMappingSet**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ProcessToMachineMapping** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest | | | |
| ***Note*** | This meta-class has the ability to associate a Process with a Machine. This relation involves the definition of further properties, e.g. timeouts.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| design | ProcessDesignTo  MachineDesignMapping | 0..1 | ref | This reference represents the identification of the design-time representation for the ProcessToMachine Mapping that owns the reference.  **Tags:**atp.Status=draft |
| machine | Machine | 0..1 | ref | This reference identifies the Machine in the context of the ProcessToMachineMapping.  **Tags:**atp.Status=draft |
| nonOsModule  Instantiation | NonOsModule  Instantiation | 0..1 | ref | This supports the optional case that the process represents a platform module.  **Tags:**atp.Status=draft |
| persistency  CentralStorage  URI | UriString | 0..1 | attr | This attribute identifies a central place for the mapped Process to store the list of available storages and version information.  **Tags:**atp.Status=draft |
| process | Process | 1 | ref | This reference identifies the Process in the context of the ProcessToMachineMapping.  **Tags:**atp.Status=draft |
| shallNotRunOn | ProcessorCore | \* | ref | This reference indicates a collection of cores onto which the mapped process shall not be executing.  **Tags:**atp.Status=draft |
| shallRunOn | ProcessorCore | \* | ref | This reference indicates a collection of cores onto which the mapped process shall be executing.  **Tags:**atp.Status=draft |

**Table 7.6: ProcessToMachineMapping**

**7.1.2 Core Affinity**

**[TPS\_MANI\_03148]**{DRAFT} **Description of Core affinity** dThe meta-class ProcessToMachineMapping provides the ability to restrict the assignment of processes to selected ProcessorCores with the two references shallRunOn and shallNotRunOn.c*(RS\_MANI\_00020)*

**[constr\_3393]**{DRAFT} **Usage of shallRunOn and shallNotRunOn references** dThe ProcessorCore that is referenced by a ProcessToMachineMapping in the role shallRunOn or shallNotRunOn shall be aggregated by the Machine that is referenced in the role machine by the same ProcessToMachineMapping.c*()*

**[constr\_1676]**{DRAFT} **Consistency of references shallRunOn and shall-**

**NotRunOn**dWithin the context of one ProcessToMachineMapping, all ProcessorCores referenced in the role shallRunOn or shallNotRunOn shall be aggregated by the same Processor.c*()*

If a model defines that a given Process shall run on a select set of ProcessorCores then there is hardly a use case to (in addition) also specify the opposite, i.e. that the Process shall not run on another set of ProcessorCores, and vice versa.

In other words, either there is a motivation to identify the ProcessorCores on which a Process is supposed to run or there is a motivation to do the exact opposite and specify the ProcessorCores where the Process is not supposed to run.

This conclusion provides the motivation for the existence of [constr\_1677].

**[constr\_1677]**{DRAFT} **Mutual exclusive existence of references shallRunOn and shallNotRunOn** dFor any given ProcessToMachineMapping, either the reference in the role shallRunOn or the reference in the role shallNotRunOn may

exist.c*()*

**7.1.3 Default Start-up and Termination Timeout**

**[TPS\_MANI\_03151]**{DRAFT} **Default value for termination timeout** dThe metaclass Machine provides the ability to define a default value for termination timeout of applications in the context of the Machine with the attribute exitTimeoutValue that is available in the EnterExitTimeout meta-class that is aggregated by the Machine in the role defaultApplicationTimeout.c*(RS\_MANI\_00007)*

**[constr\_3394]**{DRAFT} **Default value for start-up timeout on the Machine is not configurable** dThe attribute enterTimeoutValue that is available in the EnterExitTimeout is not allowed to be used if the EnterExitTimeout is aggregated by the Machine in the role defaultApplicationTimeout.c*()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **EnterExitTimeout** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest | | | |
| ***Note*** | This meta-class represents the ability to specify a pair of timeouts, one for entering, and one for exiting.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **EnterExitTimeout** |  |  |  |
| enterTimeout Value | TimeValue | 0..1 | attr | This attribute represents the value of the enter timeout in seconds.  **Tags:**atp.Status=draft |
| exitTimeout Value | TimeValue | 0..1 | attr | This attribute represents the value of the exit timeout in seconds.  **Tags:**atp.Status=draft |

**Table 7.7: EnterExitTimeout**

# 8 Execution Manifest

## 8.1 Overview

The purpose of the execution manifest is to provide information that is needed for the actual deployment of an application (formally modeled as an SwComponentType) onto the AUTOSAR adaptive platform.

One aspect of the deployment information is the provision of information that could in principle be provided as part of the application software code but which would make the application software code become very much bound to specific usage scenarios.

The general idea is to keep the application software code as independent as possible from the deployment scenario in order to increase the odds that the application software can be reused in different deployment scenarios.

In particular, the usage of PortPrototypes as a means to express communication with the “outside” of the application software allows for abstracting away the details (the concrete service instance identification) of the service configuration. As far as the model is concerned, the API between the application and the middleware is represented by the PortPrototype.

The application code does not use specific service instances but takes the PortPrototype as a symbolic replacement for this information. The specifics of this modeling aspect are described in section 10.

The top-level element of the Execution Manifest definition is the Process, in reference to the fact that the unit of deployment on the *AUTOSAR adaptive platform* is a binary that, at runtime, makes a POSIX process.

**[TPS\_MANI\_01308]**{DRAFT}**Process is not designed for re-usability** dMeta-class Process has **not** been created with the goal of reusing it on different Machines.

However, there is *some* potential for reusing configuration aspects in the definition of the Process.stateDependentStartupConfig.startupConfig.c*(RS\_MANI\_00006)*

|  |  |  |  |
| --- | --- | --- | --- |
| *AbstractExecutionContext*  Process  [0..1]  functionClusterAffiliation: String  +  +  [0..1]  numberOfRestartAttempts: PositiveInteger  [0..1]  preMapping: Boolean  +  *ARElement*  *AtpClassifier*  Executable  [0..1]  buildType: BuildTypeEnum  +  [0..1]  loggingBehavior: LoggingBehaviorEnum  +  minimumTimerGranularity: TimeValue  +  [0..1]  +  reportingBehavior: ExecutionStateReportingBehaviorEnum  [0..1]  +  version: StrongRevisionLabelString  [0..1]  «atpUriDef»  +  executable  0..1 | |  | | --- | | «enumeration»  LoggingBehaviorEnum | | usesLogging doesNotUseLogging | |
| |  | | --- | | «enumeration»  LogTraceDefaultLogLevelEnum | | fatal error warn info debug verbose off | |

**Figure 8.1: Relation of meta-classes Executable and Process**

**[TPS\_MANI\_01011]**{DRAFT} **Connection between application design and application deployment** dThe connection between the *application design* and the *application deployment* is implemented by means of a reference from meta-class Process to meta-class Executable in the role executable.

By modeling the reference in this direction it is possible to keep the design level independent of the deployment level and, at the same time, bind the deployment to a specific design.c*(RS\_MANI\_00006)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Process** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest | | | |
| ***Note*** | This meta-class provides information required to execute the referenced executable.  **Tags:**  atp.Status=draft  atp.recommendedPackage=Processes | | | |
| ***Base*** | *ARElement*, *ARObject*, *AbstractExecutionContext*, *AtpClassifier*, *CollectableElement*, *Identifiable*,  *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| design | ProcessDesign | 0..1 | ref | This reference represents the identification of the design-time representation for the Process that owns the reference.  **Tags:**atp.Status=draft |
| deterministic Client | DeterministicClient | 0..1 | ref | This reference adds further execution characteristics for deterministic clients.  **Tags:**atp.Status=draft |
| executable | Executable | 0..1 | ref | Reference to executable that is executed in the process.  **Stereotypes:** atpUriDef  **Tags:**atp.Status=draft |
| functionCluster Affiliation | String | 0..1 | attr | This attribute specifies which functional cluster the process is affiliated with. **Tags:**atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **Process** |  |  |  |
| numberOf  RestartAttempts | PositiveInteger | 0..1 | attr | This attribute defines how often a process shall be restarted if the start fails.  numberOfRestartAttempts = "0" OR Attribute not existing, start once  numberOfRestartAttempts = "1", start a second time  **Tags:**atp.Status=draft |
| preMapping | Boolean | 0..1 | attr | This attribute describes whether the executable is preloaded into the memory.  **Tags:**atp.Status=draft |
| processState Machine | ModeDeclarationGroup  Prototype | 0..1 | aggr | Set of Process States that are defined for the process.  **Tags:**atp.Status=draft |
| securityEvent | SecurityEventDefinition | \* | ref | The reference identifies the collection of SecurityEvents that can be reported by the enclosing SoftwareCluster.  **Stereotypes:** atpSplitable; atpUriDef **Tags:**  atp.Splitkey=securityEvent atp.Status=draft |
| stateDependent StartupConfig | StateDependentStartup  Config | \* | aggr | Applicable startup configurations.  **Tags:**atp.Status=draft |

**Table 8.1: Process**

**[TPS\_MANI\_01337]**{DRAFT} **Standardized values for attribute Process.functionClusterAffiliation**dThe following values of attribute Process.functionClusterAffiliation are standardized by AUTOSAR:

* **STATE\_MANAGEMENT**
* **PLATFORM\_HEALTH\_MANAGEMENT** c*(RS\_MANI\_00006)*

Please note that it is possible to use values other than from the standardized set in attribute Process.functionClusterAffiliation.

However, it is important that proprietary values of this attribute are formulated in a way that a potential clash with future standardized values can be avoided.

Clash-avoidance could be implemented by using a company-specific or project-specific prefix, infix, or suffix.

The preMapping approach of a Process is described in more detail in [SWS\_EM\_02109] in the SWS Execution Management [18].

|  |  |
| --- | --- |
| ***Class*** | ***AbstractExecutionContext*** (abstract) |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest |
| ***Note*** | This meta-class acts as a base class for entities that execute code on different levels, e.g. container, process, thread, fiber. **Tags:**atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***AbstractExecutionContext*** (abstract) | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*,  *PackageableElement*, *Referrable*, *UploadablePackageElement* | | | |
| ***Subclasses*** | Process | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 8.2: AbstractExecutionContext**

Please note that the meta-model, as depicted in Figure 8.1 supports the existence of two or more Processes that reference the same Executable.

Application1



ManifestOfInstance1.arxml



ManifestOfInstance2.arxml



Executable1

**Figure 8.2: Example deployment where one Executable is bundled with two ARXML files that each contain the description of one Process**

This is an indication that the specific Executable is supposed to be executed in several instances (i.e. in the form of POSIX processes) on the same platform. Such a situation is sketched in Figure 8.2

It is somehow likely that the startup conditions and startup parameters of different Processes may be different (in order to achieve a variation of the functionality of the Executable).

Therefore, it is necessary to allow for the definition of startup configurations on a perProcess-basis.

This aspect is described in section 8.2.

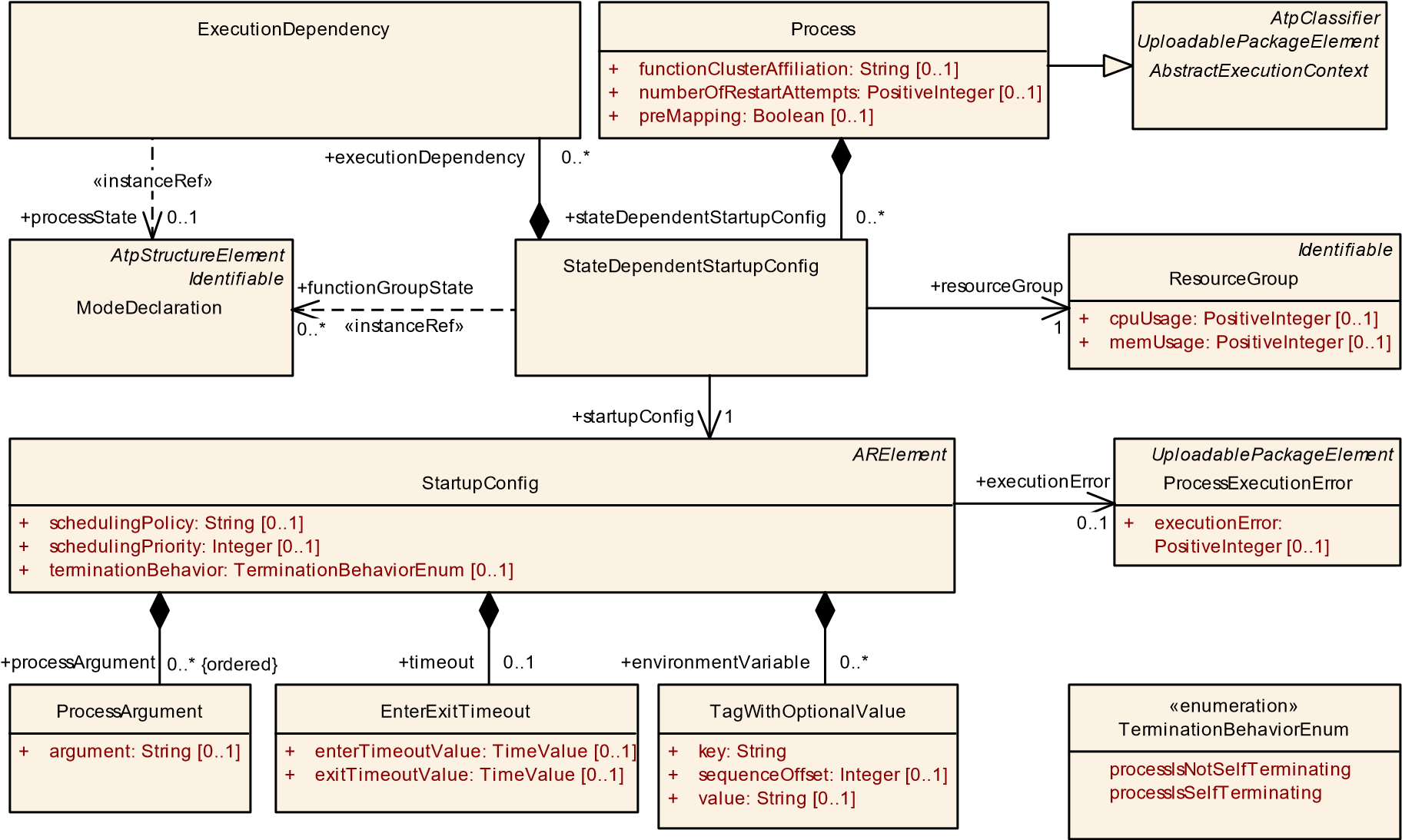
The supported process states that are defined in the Process.processStateMachine are described in more detail in [18].

## 8.2 Startup Configuration

The configuration of startup behavior is an essential part of the execution manifest.

**[TPS\_MANI\_01012]**{DRAFT} **Formal modeling of application startup behavior** dThe formal modeling of application startup behavior is implemented by means of the aggregation of meta-class StateDependentStartupConfig in the role Process.

stateDependentStartupConfig.c*(RS\_MANI\_00007)*



**Figure 8.3: Content of a Process**

**8.2.1 State-dependent Startup Configuration**

**[TPS\_MANI\_01013]**{DRAFT} **Semantics of meta-class StateDependentStartupConfig** dThe purpose of meta-class StateDependentStartupConfig is to qualify the startup configuration represented by meta-class StartupConfig for specific ModeDeclarations.

In other words, the intention is to express that the StartupConfig is applicable if the state machines that control the startup are in the states represented by the ModeDeclaration referenced in the role StateDependentStartupConfig.functionGroupState.c*(RS\_MANI\_00007)*

As a consequence of the reference from the StateDependentStartupConfig to ModeDeclaration the Execution Manifest is defined for a specific Machine to which the binary and the Manifest is deployed.

**[constr\_3423]**{DRAFT}**StateDependentStartupConfig of a Process shall reference a functionGroupState** dEach StateDependentStartupConfig of a Process shall reference at least one ModeDeclaration in the role functionGroupState.c*()*

However, the references to function group states within the context of one Process shall only refer to function group states **of the same function group**. This aspect is formalized by [constr\_1688].

**[constr\_1688]**{DRAFT} **StateDependentStartupConfig shall only refer to function group states of the same function group** dFor all StateDependentStartupConfigs aggregated in the role Process.stateDependentStartupConfig, references in the role functionGroupState to ModeDeclaration shall only refer to ModeDeclarations aggregated by the same ModeDeclarationGroup in the context of the same ModeDeclarationGroupPrototype (that represents the actual function group).c*()*

It is necessary to specify constraint [constr\_3396] to regulate the number of StateDependentStartupConfigs that refer to the same ModeDeclaration in the context of one Process because the resulting startup configuration would be ambiguous.

**[constr\_3396]**{DRAFT} **Number of Process.stateDependentStartupConfig that refer to the same functionGroupState** dWithin the context of a given Process, no two StateDependentStartupConfigs shall refer to the same ModeDeclaration in the role functionGroupState.c*()*

**[TPS\_MANI\_01046]**{DRAFT} **Semantics of StateDependentStartupConfig. functionGroupState**dThe ModeDeclarations referenced in the role StateDependentStartupConfig.functionGroupState shall be considered in a way such that the StateDependentStartupConfig applies if **any** of the referenced ModeDeclarations is active.

In other words, the ModeDeclarations are or-ed for the determination of whether a

StateDependentStartupConfig is applicable.c*(RS\_MANI\_00007)*

**[constr\_3424]**{DRAFT} **StateDependentStartupConfig shall never reference the functionGroupStateOff** dA StateDependentStartupConfig shall never reference the ModeDeclaration that has the shortName Off in the role functionGroupState. Please note that the OffModeDeclaration is a special state in a Function Group as defined by [TPS\_MANI\_03195].c*()*

**[constr\_1618]**{DRAFT} **Ability to shut down** dIn the context of one Machine, at least one Process shall have a stateDependentStartupConfig.functionGroupState that has the shortNameShutdown.c*()*

**[constr\_1619]**{DRAFT} **Ability to restart** dIn the context of one Machine, at least one Process shall have a stateDependentStartupConfig.functionGroupState that has the shortNameRestart.c*()*

**[TPS\_MANI\_01209]**{DRAFT} **Definition of environment variables in process scope** dIt is possible to define environment variables in the scope of any given Process.

For this purpose the aggregation of TagWithOptionalValue in the role StartupConfig.environmentVariable exists.

The name of the environment variable shall be specified by means of the attribute TagWithOptionalValue.key, the value can be modeled by means of TagWithOptionalValue.value.

This encloses the ability to define environment variables with empty values. For this purpose, the attribute TagWithOptionalValue.value shall simply be omitted.c *(RS\_MANI\_00007)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **StateDependentStartupConfig** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest | | | |
| ***Note*** | This meta-class defines the startup configuration for the process depending on a collection of machine states.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| execution  Dependency | ExecutionDependency | \* | aggr | This attribute defines that all processes that are referenced via the ExecutionDependency shall be launched and shall reach a certain ProcessState before the referencing process is started.  **Tags:**atp.Status=draft |
| functionGroup State | ModeDeclaration | \* | iref | This represent the applicable functionGroupMode.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**FunctionGroupStateIn  FunctionGroupSetInstanceRef |
| resource  Consumption | ResourceConsumption | 0..1 | aggr | This aggregation provides the ability to define resource consumption boundaries on a per-process-startup-config basis.  **Tags:**atp.Status=draft |
| resourceGroup | ResourceGroup | 1 | ref | Reference to an applicable resource group.  **Tags:**atp.Status=draft |
| startupConfig | StartupConfig | 1 | ref | Reference to a reusable startup configuration with startup parameters.  **Tags:**atp.Status=draft |

**Table 8.3: StateDependentStartupConfig**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **StartupConfig** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest | | | |
| ***Note*** | This meta-class represents a reusable startup configuration for processes..  **Tags:**  atp.Status=draft  atp.recommendedPackage=StartupConfigs | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| environment Variable | TagWithOptionalValue | \* | aggr | This aggregation represents the collection of environment variables that shall be added to the respective Process’s environment prior to launch.  **Tags:**atp.Status=draft |
| executionError | ProcessExecutionError | 0..1 | ref | this reference is used to identify the applicable execution error  **Tags:**atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **StartupConfig** |  |  |  |
| process Argument  (ordered) | ProcessArgument | \* | aggr | This aggregation represents the collection of command-line arguments applicable to the enclosing StartupConfig.  **Tags:**atp.Status=draft |
| scheduling Policy | String | 0..1 | attr | This attribute represents the ability to define the scheduling policy for the initial thread of the application.  **Tags:**atp.Status=draft |
| scheduling Priority | Integer | 0..1 | attr | This is the scheduling priority requested by the application itself.  **Tags:**atp.Status=draft |
| termination Behavior | TerminationBehavior  Enum | 0..1 | attr | This attribute defines the termination behavior of the Process.  **Tags:**atp.Status=draft |
| timeout | EnterExitTimeout | 0..1 | aggr | This aggregation can be used to specify the timeouts for launching and terminating the process depending on the StartupConfig.  **Tags:**atp.Status=draft |

**Table 8.4: StartupConfig**

**[TPS\_MANI\_01277]**{DRAFT} **Definition of a start-up timeout for a StartupConfig of a Process** dMeta-class StartupConfig provides the ability to define a startup timeout for a Process by means of the attribute enterTimeoutValue that is aggregated by meta-class EnterExitTimeout that is aggregated by the StartupConfig in the role timeout.c*(RS\_MANI\_00007)*

**[TPS\_MANI\_01278]**{DRAFT} **Definition of a termination timeout for a StartupConfig of a Process** dMeta-class StartupConfig provides the ability to define a termination timeout for a Process by means of the attribute exitTimeoutValue that is aggregated by meta-class EnterExitTimeout that is aggregated by the StartupConfig in the role timeout.c*(RS\_MANI\_00007)*

**8.2.2 Scheduling**

**[TPS\_MANI\_01061]**{DRAFT} **Requirements on scheduling** dThe attributes

StartupConfig.schedulingPolicy and StartupConfig.schedulingPriority make requirements on the scheduling of the main thread of a process that is created out of launching the corresponding Executable.c*(RS\_MANI\_00007)*

**[TPS\_MANI\_01328]**{DRAFT} **Standardized values for attribute StartupConfig. schedulingPolicy** dThe following values are standardized for attribute StartupConfig.schedulingPolicy:

* **SCHED\_RR**
* **SCHED\_FIFO**
* **SCHED\_OTHER**

c*(RS\_MANI\_00007)*

It is possible to use a custom, non-standardized value for the attribute StartupConfig.schedulingPolicy but this option comes with the obligation to use a value that is guaranteed to not clash with possible future extensions of the collection of standardized values.

**[TPS\_MANI\_01188]**{DRAFT} **Semantics of attribute schedulingPriority** dThe value of attribute StartupConfig.schedulingPriority shall be interpreted such that the higher values represent a higher scheduling priority.c*(RS\_MANI\_00007)*

**[constr\_1692]**{DRAFT} **Value of schedulingPriority** dThe value of attribute

StartupConfig.schedulingPriority shall be set to a positive integer value.c*()*

**8.2.3 Process Arguments**

Please find more information about the interpretation of ProcessArgument in the SWS Execution Manifest [18].

**[constr\_1769]**{DRAFT} **Existence of ProcessArgument.argument** dFor each ProcessArgument, attribute argument shall exist **at the time when manifest creation is finished**.c*()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ProcessArgument** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest | | | |
| ***Note*** | This meta-class has the ability to define command line arguments for processing by the Main function.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| argument | String | 0..1 | attr | This represents one command-line argument to be processed by the executable software.  **Tags:**atp.Status=draft |

**Table 8.5: ProcessArgument**

**8.2.4 Association with Resource Group**

Meta-class StateDependentStartupConfig also supports the specification of a relation to a resource group.

**[TPS\_MANI\_01017]**{DRAFT} **Relation of startup configuration to resource group** dThe modeling of a resource group is possible by means of meta-class ResourceGroup in the OsModuleInstantiation of the Machine and the assignment of a Process to a ResourceGroup is supported by the association from StateDependentStartupConfig to ResourceGroup in the role resourceGroup.c*(RS\_MANI\_00007)*

Process

[0..1]

functionClusterAffiliation: String

+

[0..1]

+

numberOfRestartAttempts: PositiveInteger

+

preMapping: Boolean

[0..1]

*Identifiable*

ResourceGroup

[0..1]

cpuUsage: PositiveInteger

+

[0..1]

memUsage: PositiveInteger

+

StateDependentStartupConfig

*AtpClassifier*

*UploadablePackageElement*

*AbstractExecutionContext*

+

stateDependentStartupConfig

0..\*

+

resourceGroup

1

**Figure 8.4: Modeling of how Process relates to ResourceGroup**

**[constr\_3413]**{DRAFT} **StateDependentStartupConfig of a Process is mapped to exactly one ResourceGroup** dEach StateDependentStartupConfig of a Process shall be assigned to exactly one ResourceGroup that is defined in the Machine Manifest.c*()*

**8.2.5 Execution Dependency**

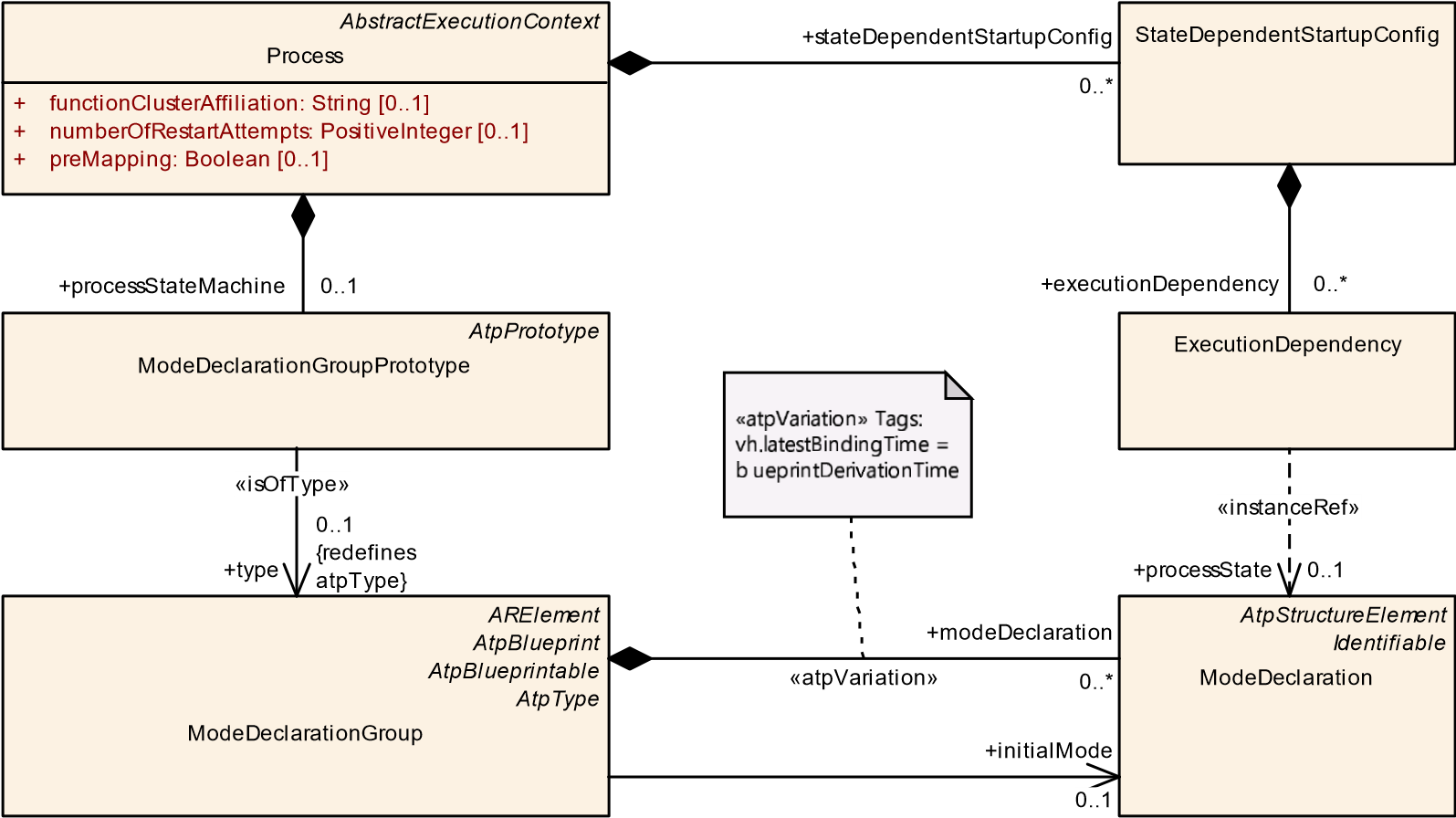
The modeling of an execution dependency makes two Processes become associated to each other by means of the definition of an ExecutionDependency.

But since the reference that defines the execution dependency is modeled as an instanceRef, the referenced Process needs to be extracted from the context references in the instanceRef.

Once the two Processes are identified it is necessary for the validity of the startup dependency that they refer to the identical function group.

**[TPS\_MANI\_01041]**{DRAFT} **Startup configuration supports the definition of a launch sequence dependency** dThe modeling of startup configuration also supports the definition of a launch sequence dependency, formalized by the meta-class ExecutionDependency that is aggregated by StateDependentStartupConfig in the role executionDependency.

The ExecutionDependency allows to define a dependency to a process that needs to be in a specific process state before the process that aggregates the ExecutionDependency via StateDependentStartupConfig is launched.c*(RS\_MANI\_00007)*



**Figure 8.5: Modeling of how Process relates to ModeDeclaration owned by another**

**Process**

**[constr\_1689]**{DRAFT} **Modeling of a startup dependency between different Processes** dThe existence of attribute Process.stateDependentStartupConfig.executionDependency is only valid if the owner of the stateDependentStartupConfig.executionDependency (in other words: the referencing Process) and the owner of the ModeDeclarationGroupPrototype referenced in the role contextModeDeclarationGroupPrototype within the reference stateDependentStartupConfig.executionDependency.processState (i.e. the referenced Process) refer to the identical function group state formalized as

ModeDeclaration.c*()*

Figure 8.6 provides an exemplary explanation of [constr\_1689]. In this example, Process “B” (the referencing Process as of [constr\_1689]) defines an executionDependency to Process “A”.

This executionDependency is only valid if both Process “A” and Process “B” aggregate a StateDependentStartupConfig that refers to the same function group state “MD” within function group “FG”.

Process “A” can be found by following the ExecutionDependency (specifically the contextModeDeclarationGroupPrototype) and the instanceRef that goes from the ExecutionDependency to the Process State “PS”.

The **owner** of “PS” is Process “B”, and if “B” refers to function group state “MD” within function group “FG” and if “A” refers to function group state“MD” within “FG” then the constraint [constr\_1689] is fulfilled.

Process "B"

Process "A"

StateDependentStartup

Config

ExecutionDependency

ProcessState "PS"

ModeDeclarationGroup

(

Prototype)

FunctionGroup "FG"

ModeDeclarationGroup

(

Prototype)

StateDependentStartup

Config

context

context

context

ModeDeclaration "MD"

ModeDeclarationGroup

type

target

target

**Figure 8.6: Explanation of dependencies from one Process to another**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ExecutionDependency** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest | | | |
| ***Note*** | This element defines a ProcessState in which a dependent process needs to be before the process that aggregates the ExecutionDependency element can be started.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| processState | ModeDeclaration | 0..1 | iref | This represent the applicable modeDeclaration that represents an ProcessState.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**ModeInProcessInstance Ref |

**Table 8.6: ExecutionDependency**

**[constr\_1606]**{DRAFT} **Processes with mutual ExecutionDependencys** dA

Process.stateDependentStartupConfig.executionDependency shall not refer to any ModeDeclaration owned by a second Process that in turn refers via stateDependentStartupConfig.executionDependency to any ModeDeclaration owned by the first Process.c*()*

**8.2.6 Assignment of Processes to Function Group states**

There are use cases where starting and terminating of individual groups of processes is necessary. This is supported in AUTOSAR by function groups that group processes together.

A function group may have a number of function group states, e.g. Running, Idle, Terminating. The StateDependentStartupConfig of a Process can be assigned to a function group state and the start-up of the Process will then depend on this assignment.

The modeling of a function group and its function group states is described in section 8.4 in more detail. The usage of Function Groups is described in more detail in [18].

**[TPS\_MANI\_03152]**{DRAFT} **Assignment of a StateDependentStartupConfig to a function group state** dThe StateDependentStartupConfig is assigned to a function group state with the functionGroupState reference.c*(RS\_MANI\_00041)*

**8.2.7 Resource Consumption Boundaries**

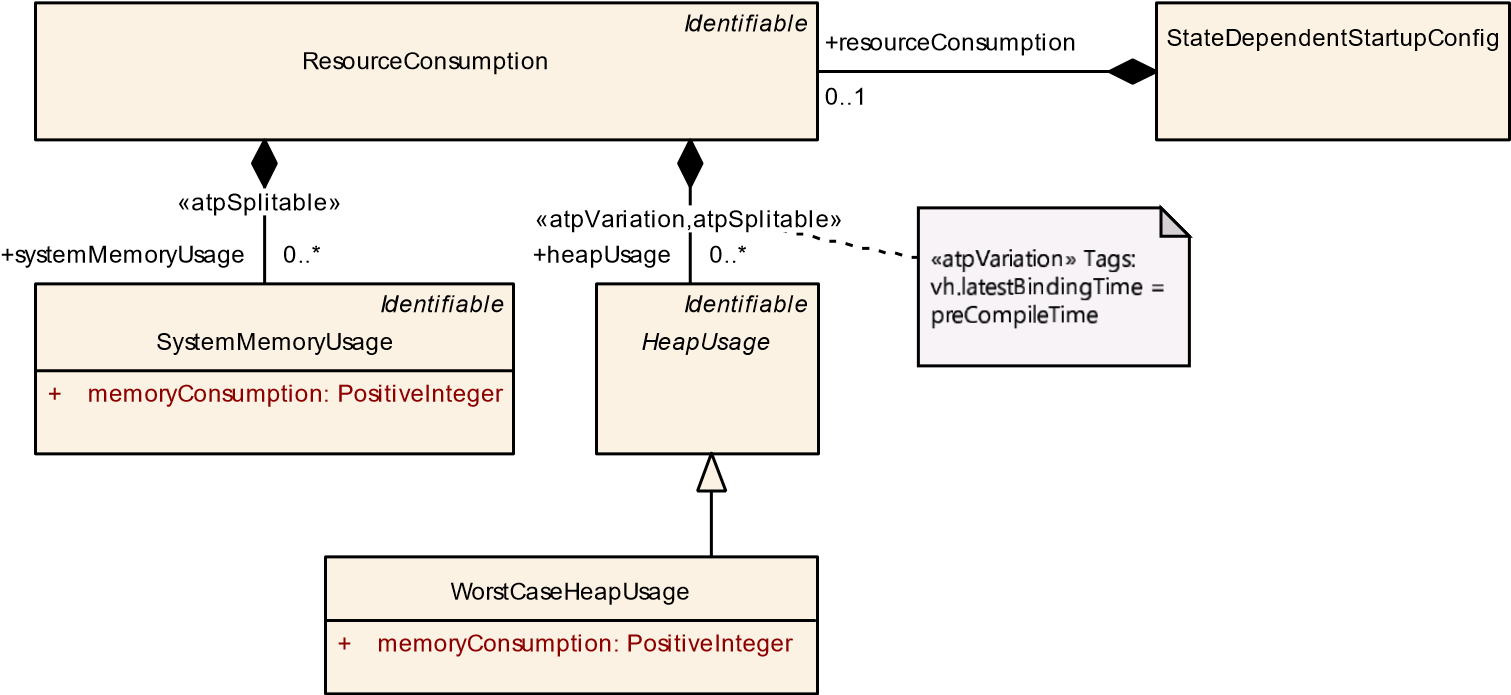
**[TPS\_MANI\_01269]**{DRAFT} **Specification of boundaries for resource consumption** dIt is possible to specify boundaries for resource consumption, specifically in terms of memory consumption for system memory and heap, of a given startup configuration of a Process:

* The formalization of heap usage is represented by meta-class HeapUsage, aggregated via meta-class ResourceConsumption at StateDependentStartupConfig. The actual value of the heap usage is computed out of the sum of all aggregated ResourceConsumption.heapUsage.
* The formalization of system (i.e. kernel-space) memory usage is represented by meta-class SystemMemoryUsage, aggregated via meta-class ResourceConsumption at StateDependentStartupConfig. The actual value of the system memory usage is computed out of the sum of all aggregated ResourceConsumption.systemMemoryUsage.

c*(RS\_MANI\_00020)*

Please note the difference between the ability of defining resource consumption boundaries for a single Process, as opposed to the ability to associate a Process with a ResourceGroup that has the ability to also define resource consumption boundaries, albeit on a more coarse-grained level.

In contrast to that, the StateDependentStartupConfig.resourceConsumption allows for a fine-grained definition that can even observe the differences in resource consumption with respect to different startup configurations.



**Figure 8.7: Modeling of resource consumption boundaries for a given Process**

**[constr\_1707]**{DRAFT} **Eligible subclasses of HeapUsage in the context of StateDependentStartupConfig.resourceConsumption** dThe definition of StateDependentStartupConfig.resourceConsumption.heapUsage shall only be done by means of the concrete sub-class WorstCaseHeapUsage.c*()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ResourceConsumption** | | | |
| ***Package*** | M2::AUTOSARTemplates::CommonStructure::ResourceConsumption | | | |
| ***Note*** | Description of consumed resources by one implementation of a software. | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| heapUsage | HeapUsage | \* | aggr | Collection of the heap memory allocated by this implementation.  **Stereotypes:** atpSplitable; atpVariation **Tags:**  atp.Splitkey=heapUsage.shortName, heap Usage.variationPoint.shortLabel vh.latestBindingTime=preCompileTime |
| systemMemory Usage | SystemMemoryUsage | \* | aggr | Collection of the system memory allocated by the owner.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=systemMemoryUsage.shortName atp.Status=draft |

**Table 8.7: ResourceConsumption**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***HeapUsage*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::CommonStructure::ResourceConsumption::HeapUsage | | | |
| ***Note*** | Describes the heap memory usage of a SW-Component. | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Subclasses*** | MeasuredHeapUsage, RoughEstimateHeapUsage, WorstCaseHeapUsage | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 8.8: HeapUsage**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **WorstCaseHeapUsage** | | | |
| ***Package*** | M2::AUTOSARTemplates::CommonStructure::ResourceConsumption::HeapUsage | | | |
| ***Note*** | Provides a formal worst case heap usage. | | | |
| ***Base*** | *ARObject*, *HeapUsage*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| memory  Consumption | PositiveInteger | 1 | attr | Worst case heap consumption. Unit: byte. |

**Table 8.9: WorstCaseHeapUsage**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SystemMemoryUsage** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest | | | |
| ***Note*** | Describes the system memory (i.e. kernel space) consumption.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| memory  Consumption | PositiveInteger | 1 | attr | Provides a formal worst case system usage. The unit is byte.  **Tags:**atp.Status=draft |

**Table 8.10: SystemMemoryUsage**

**8.2.8 Error and Termination Behavior**

**[TPS\_MANI\_01334]**{DRAFT} **Semantics of StartupConfig.terminationBehavior** dThe attribute StartupConfig.terminationBehavior defines the termination behavior of the Process in terms of whether (or not) the Process that references the enclosing StartupConfig in the role stateDependentStartupConfig. startupConfig is configured to self-terminate.c*(RS\_MANI\_00007)*

|  |  |
| --- | --- |
| ***Enumeration*** | **TerminationBehaviorEnum** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest |
| ***Note*** | This enumeration provides options for controlling of how a Process terminates.  **Tags:**atp.Status=draft |
| ***Literal*** | ***Description*** |
| processIsNotSelf Terminating | The Process terminates only on request from Execution Management.  **Tags:**  atp.EnumerationLiteralIndex=0 atp.Status=draft |
| processIsSelf Terminating | The Process is allowed to terminate without request from Execution Management.  **Tags:**  atp.EnumerationLiteralIndex=1 atp.Status=draft |

**Table 8.11: TerminationBehaviorEnum**

**[constr\_10007]**{DRAFT} **Existence of ProcessExecutionError.executionError**dFor each ProcessExecutionError, attribute executionError shall exist **at the time when manifest creation is finished**.c*()*

**[constr\_10008]**{DRAFT} **Value of ProcessExecutionError.executionError** dThe value of attribute ProcessExecutionError.executionError shall at least be set to 1 (or higher).c*()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ProcessExecutionError** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest | | | |
| ***Note*** | This meta-class has the ability to describe the value of a execution error along with a documentation of its semantics.  **Tags:**  atp.Status=draft  atp.recommendedPackage=ProcessExecutionErrors | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable*  *Element*, *Referrable*, *UploadablePackageElement* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| executionError | PositiveInteger | 0..1 | attr | This attribute defines the numeric value which Execution Management and Platform Health Management reports to State Management if the Process terminates unexpectedly or violates its supervision. It shall give further error information for error recovery.  **Tags:**atp.Status=draft |

**Table 8.12: ProcessExecutionError**

## 8.3 Deterministic Client

As already explained in section 3.19.1, there is a use case to support the concept of the so-called Deterministic Client on the *AUTOSAR adaptive platform*. The conceptual background of Deterministic Client is explained in the SWS Execution Management [18].

The support for this concept consists of two aspects. The *design aspect* has already been explained in section 3.19.1 while the *deployment aspect* is discussed in this chapter.

**[TPS\_MANI\_01203]**{DRAFT} **Semantics of DeterministicClient** dThe existence of reference Process.deterministicClient means that the enclosing Process implements the concept of a Deterministic Client.

Further information for the configuration of the Deterministic Client can be obtained from the ProcessDesign referenced in the role Process.design.c*(RS\_MANI\_00050)*

The details of the support for Deterministic Client are visualized in Figure 8.8.

*UploadablePackageElement*

DeterministicClient

+

[0..1]

cycleTimeValue: TimeValue

[0..1]

numberOfWorkers: PositiveInteger

+

*AbstractExecutionContext*

Process

[0..1]

functionClusterAffiliation: String

+

[0..1]

numberOfRestartAttempts: PositiveInteger

+

+

[0..1]

preMapping: Boolean

*ARElement*

ProcessDesign

*Identifiable*

DeterministicClientResourceNeeds

+

hardwarePlatform: String

[0..1]

DeterministicClientResource

[0..1]

numberOfInstructions: NormalizedInstruction

+

[0..1]

sequentialInstructionsBegin: NormalizedInstruction

+

[0..1]

sequentialInstructionsEnd: NormalizedInstruction

+

[0..1]

speedup: Float

+

+

runResource

0..1

design

+

0..1

+

initResource

0..1

+

deterministicClient

0..1

+

deterministicClientResourceNeeds

0..\*

**Figure 8.8: Modeling of support for Deterministic Client in the deployment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **DeterministicClient** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest | | | |
| ***Note*** | The meta-class DeterministicClient provides the ability to support the deterministic execution of one or more processes with specific configuration parameters for DeterministicClient library functions.  **Tags:**  atp.Status=draft  atp.recommendedPackage=DeterministicClients | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable*  *Element*, *Referrable*, *UploadablePackageElement* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| cycleTimeValue | TimeValue | 0..1 | attr | This attribute represents the cycle time for execution of a DeterministicClient activation cycle.  **Tags:**atp.Status=draft |
| numberOf Workers | PositiveInteger | 0..1 | attr | Number of independent workers that process data-sets. Size of the worker pool shall be decided based on availability of resources like processor cores or memory.  **Tags:**atp.Status=draft |

**Table 8.13: DeterministicClient**

## 8.4 Function Groups

**8.4.1 Semantics of Function Group**

Function groups with function group states individually control groups of functionally coherent Application processes. The Process state may depend on a mode that is defined in the function group in case that the StateDependentStartupConfig refers to the function group state with the functionGroupState reference.

The usage of Function Groups is described in more detail in [18]. **[TPS\_MANI\_03145]**{DRAFT} **Description of a function group** dBy defining a

ModeDeclarationGroupPrototype aggregated in the role FunctionGroupSet. functionGroup it is possible to define a function group that has a shortName and a set of Modes (States).

The ModeDeclarationGroupPrototype points to a reusable ModeDeclarationGroup in the role type that contains the different modes as ModeDeclarations and a designated initialMode.c*(RS\_MANI\_00041)*

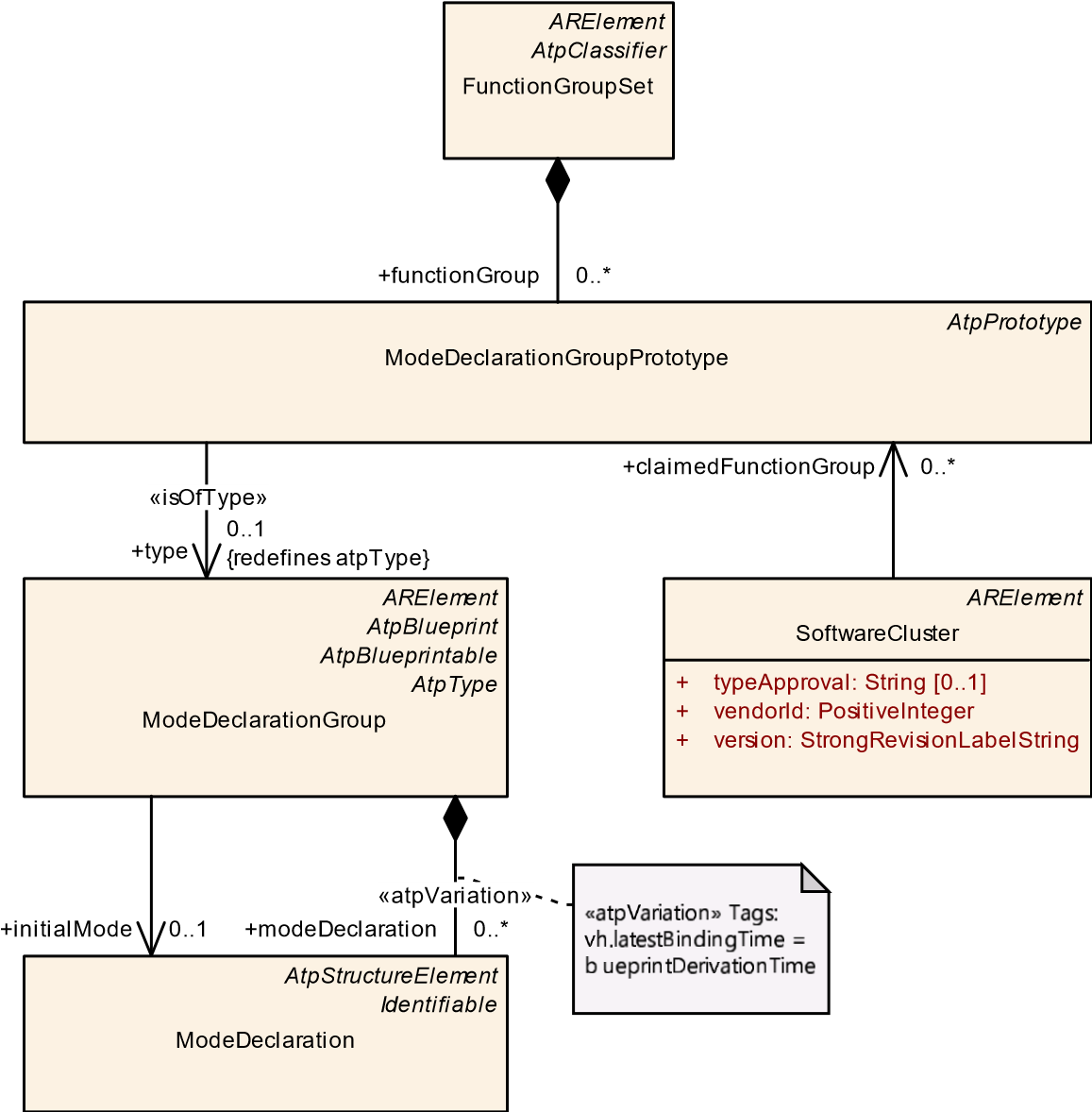
**[TPS\_MANI\_03194]**{DRAFT} **Function Group State** dA function group state is described by a ModeDeclaration within a ModeDeclarationGroup that is referenced in the role type by a ModeDeclarationGroupPrototype aggregated as functionGroup by a FunctionGroupSet. The function group state is identified by its shortName.c*(RS\_MANI\_00041)*

The modeling described in [TPS\_MANI\_03145] and [TPS\_MANI\_03194] is depicted in Figure 8.9.

**[TPS\_MANI\_03195]**{DRAFT} **Off state in Function Group** dEach functionGroup shall define a ModeDeclaration with the shortNameOff. This ModeDeclaration shall also be referenced in the role initialMode by ModeDeclarationGroup that types the respective functionGroup.c*(RS\_MANI\_00041)*

**[constr\_1786]**{DRAFT} **Restriction to use functionGroup in terms of SoftwareCluster**dEach functionGroup shall only be referenced in the role claimed-

FunctionGroup by **at most one** SoftwareCluster.c*()*



**Figure 8.9: Configuration of Function Groups**

**[constr\_1787]**{DRAFT} **Restricted use of function groups in the context of a**

**SoftwareCluster** dAll Processes referenced by a SoftwareCluster in the role containedProcess shall only aggregate StateDependentStartupConfigs where the reference functionGroupState refers to a ModeDeclarationGroupPrototype (as context) that is also referenced by the same SoftwareCluster in the role claimedFunctionGroup.c*()*

The description of SoftwareCluster can be found in section 14.

**[constr\_10023]**{DRAFT} **Mandatory content of any functionGroup** dAll ModeDeclarationGroupPrototypes aggregated by a FunctionGroupSet in the role functionGroup shall refer to a ModeDeclarationGroup that contains one ModeDeclaration with the shortName**Verify**.c*()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **FunctionGroupSet** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::General | | | |
| ***Note*** | This meta-class provides the ability to create arbitrary collections of function groups.  **Tags:**  atp.Status=draft  atp.recommendedPackage=FunctionGroupSets | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| functionGroup | ModeDeclarationGroup  Prototype | \* | aggr | This aggregation represents the collection of function groups.  **Tags:**atp.Status=draft |

**Table 8.14: FunctionGroupSet**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ModeDeclarationGroupPrototype** | | | |
| ***Package*** | M2::AUTOSARTemplates::CommonStructure::ModeDeclaration | | | |
| ***Note*** | The ModeDeclarationGroupPrototype specifies a set of Modes (ModeDeclarationGroup) which is provided or required in the given context. | | | |
| ***Base*** | *ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| type | ModeDeclarationGroup | 0..1 | tref | The "collection of ModeDeclarations" ( = ModeDeclaration  Group) supported by a component  **Stereotypes:** isOfType |

**Table 8.15: ModeDeclarationGroupPrototype**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ModeDeclarationGroup** | | | |
| ***Package*** | M2::AUTOSARTemplates::CommonStructure::ModeDeclaration | | | |
| ***Note*** | A collection of Mode Declarations. Also, the initial mode is explicitly identified.  **Tags:**atp.recommendedPackage=ModeDeclarationGroups | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*,  *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ModeDeclarationGroup** |  |  |  |
| initialMode | ModeDeclaration | 0..1 | ref | The initial mode of the ModeDeclarationGroup. This mode is active before any mode switches occurred. |
| mode  Declaration | ModeDeclaration | \* | aggr | The ModeDeclarations collected in this ModeDeclaration Group.  **Stereotypes:** atpVariation  **Tags:**vh.latestBindingTime=blueprintDerivationTime |
| modeTransition | ModeTransition | \* | aggr | This represents the avaliable ModeTransitions of the  ModeDeclarationGroup |

**Table 8.16: ModeDeclarationGroup**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ModeDeclaration** | | | |
| ***Package*** | M2::AUTOSARTemplates::CommonStructure::ModeDeclaration | | | |
| ***Note*** | Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model. | | | |
| ***Base*** | *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*,  *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 8.17: ModeDeclaration**

**8.4.2 Machine Function Group**

Please note that one functionGroup claimed by one SoftwareCluster of categoryPLATFORM\_CORE takes the role of a "machine function group".

This functionGroup is required to have a dedicated shortName and it also is required to define a certain minimal, but extensible set of ModeDeclarations that also have standardized shortNames.

**[TPS\_MANI\_01330]**{DRAFT} **Definition of machine function group** dExactly one functionGroup shall exist that has the shortName "MachineFG" and that is typed by a ModeDeclarationGroup that defines at least the following list of ModeDeclarations with the shortNames

* **Off**,
* **Verify**,
* **Startup**,
* **Shutdown**, and
* **Restart**. c*(RS\_MANI\_00041)* Please note that the startup of a Process may depend on Modes that are defined in the context of a SoftwareCluster of categoryPLATFORM\_CORE. The StateDependentStartupConfig is described in chapter 8.2.

**[constr\_1789]**{DRAFT} **Scope of machine function group** dThe functionGroup that represents the machine function group (see [TPS\_MANI\_01330]) shall only be referenced in the role claimedFunctionGroup by a SoftwareCluster of category PLATFORM\_CORE.c*()*

## 8.5 Reporting of Security Events

It is possible to report so-called security events (formalized by meta-class SecurityEventDefinition) from the context of a Process.

This approach works for application-level software as well as for functional clusters with the exception of the Execution Manager (because the Execution Manager is itself not modeled as a Process).

*IdsCommonElement*

SecurityEventDefinition

+

id: PositiveInteger

[0..1]

*AbstractExecutionContext*

Process

+

functionClusterAffiliation: String

[0..1]

+

numberOfRestartAttempts: PositiveInteger

[0..1]

+

preMapping: Boolean

[0..1]

«atpSplitable,atpUriDef»

+

securityEvent

0..\*

**Figure 8.10: Modeling of support for the reporting of SecurityEventDefinition on deployment level**

Please find more information about the semantics and usage of security events in the TPS Security Extract Template [21].

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **SecurityEventDefinition** | | | |
| ***Package*** | M2::AUTOSARTemplates::SecurityExtractTemplate | | | |
| ***Note*** | This meta-class defines a security-related event as part of the intrusion detection system.  **Tags:**  atp.Status=draft  atp.recommendedPackage=SecurityEventDefinitions | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *IdsCommonElement*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| eventSymbol Name | SymbolProps | 0..1 | aggr | This aggregation defines optionally an alternative Event Name for the SecurityEventDefinition in case there is a collision of shortNames.  **Stereotypes:** atpSplitable **Tags:**  atp.Splitkey=eventSymbolName.shortName atp.Status=draft |
| id | PositiveInteger | 0..1 | attr | This attribute represents the numerical identification of the defined security event. The identification shall be unique within the scope of the IDS.  **Tags:**atp.Status=draft |

**Table 8.18: SecurityEventDefinition**

# 9 Platform Module Development

The model of platform modules and their instantiation has two major use-cases:

* provide dedicated attributes to configure the platform modules
* define the potential start of the module’s executable as process.

The two use-cases are combined in one modeling approach: the Machine.module-

Instantiation, which collects sub-classes of AdaptiveModuleInstantiation. This modeling approach boils down to the variety of platform module models found in this chapter.

The OsModuleInstantiation defines several attributes to be configured for the Os, however the OsModuleInstantiation is the only AdaptiveModuleInstantiation where it is not possible to map it to a Process model element.

Of course there will be processes running the Os on the Machine anyway, however, these processes are not modeled.

Then there is the scenario where dedicated sub-classes of NonOsModuleInstantiation exist. Here the specific attributes are provided individually per sub-class, e.g.

NmInstantiation or LogAndTraceInstantiation.

Those NonOsModuleInstantiations are independent from the startup behavior implementation. If a stack implementation decides to implement a specific functional cluster in a dedicated Process, then the specific NonOsModuleInstantiation will also be part of a ProcessToMachineMapping.

But, if the stack implementation decides to implement a specific functional cluster as a library (or make the functionality part of another functional cluster), then the specific NonOsModuleInstantiation just defines the configuration values for that functionality and does not explicitly take part in a ProcessToMachineMapping.

Another scenario is a rather distributed nature of a functional cluster, where there is no need to provide centralized configuration means. This is applicable for example to PersistencyDeployment or PlatformHealthManagementContribution.

The functional behavior of the functional cluster is determined by the sum of several contributions. There is no single configuration entity provided.

Nevertheless, if a stack implementation decides to implement such a distributed functional cluster as a single Executable, the GenericModuleInstantiation can be used to define the startup behavior for a specific machine.

The configuration settings for individual Adaptive Autosar modules are covered by specializations of the abstract class AdaptiveModuleInstantiation.

*ARElement*

*AtpStructureElement*

Machine

trustedPlatformExecutableLaunchBehavior: TrustedPlatformExecutableLaunchBehaviorEnum

+

*Identifiable*

*AdaptiveModuleInstantiation*

OsModuleInstantiation

[0..1]

supportedTimerGranularity: TimeValue

+

«enumeration»

BuildTypeEnum

buildTypeDebug

buildTypeRelease

*Identifiable*

ResourceGroup

[0..1]

+

cpuUsage: PositiveInteger

+

memUsage: PositiveInteger

[0..1]

GenericModuleInstantiation

*AbstractExecutionContext*

Process

+

functionClusterAffiliation: String

[0..1]

+

numberOfRestartAttempts: PositiveInteger

[0..1]

preMapping: Boolean

[0..1]

+

*AtpPrototype*

*Identifiable*

RootSwComponentPrototype

*ARElement*

*AtpClassifier*

Executable

buildType: BuildTypeEnum

+

[0..1]

[0..1]

loggingBehavior: LoggingBehaviorEnum

+

[0..1]

minimumTimerGranularity: TimeValue

+

[0..1]

reportingBehavior: ExecutionStateReportingBehaviorEnum

+

version: StrongRevisionLabelString

[0..1]

+

*NonOsModuleInstantiation*

*Identifiable*

ProcessToMachineMapping

[0..1]

+

persistencyCentralStorageURI: UriString

«atpSplitable»

+

moduleInstantiation

0..\*

+

nonOsModuleInstantiation

0..1

process

+

1

+

machine

0..1

+

rootSwComponentPrototype

0..1

«atpUriDef»

+

executable

0..1

+

resourceGroup

0..\*

**Figure 9.1: Adaptive Autosar Module Configuration**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***AdaptiveModuleInstantiation*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation | | | |
| ***Note*** | This meta-class defines the abstract attributes for the configuration of an adaptive autosar module instance on a specific machine.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Subclasses*** | *NonOsModuleInstantiation*, OsModuleInstantiation | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 9.1: AdaptiveModuleInstantiation**

Each Adaptive Autosar module other than OS can be assigned to a Process with the ProcessToMachineMapping.

**[constr\_1490]**{DRAFT} **Allowed value for Executable.category if ProcessToMachineMapping references a NonOsModuleInstantiation** dIf a ProcessToMachineMapping references a NonOsModuleInstantiation, then the Process referenced in the role ProcessToMachineMapping.process shall only refer (in the role Process.executable) to an Executable where attribute Executable.category is set to PLATFORM\_LEVEL (see [constr\_1605]).c*()*

Please note that the model relation described in [constr\_1490] is sketched in Figure 9.1.

The meta-class GenericModuleInstantiation can be used to define configuration settings of generic modules and modules that are not standardized by AUTOSAR. Different modules are distinguishable by the category attribute.

Please note that both elements are Identifiable and therefore are able to describe special data (sdg), by which means it is possible to define generic custom settings that are not represented by the standard model. For more information, please refer to the AUTOSAR Generic Structure Template [6].

**[TPS\_MANI\_03096]**{DRAFT} **Machine-specific configuration settings for a generic module** dThe Machine-specific configuration settings for a generic module are collected in GenericModuleInstantiation where the value of attribute category value denotes the module.c*(RS\_MANI\_00023)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **GenericModuleInstantiation** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation | | | |
| ***Note*** | This meta-class defines the attributes for the generic module configuration on a specific machine. Different modules are distinguishable by the category attribute. This element can also be used to describe modules that are not standardized by AUTOSAR.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *NonOsModule Instantiation*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 9.2: GenericModuleInstantiation**

## 9.1 OS Module configuration

**[TPS\_MANI\_03098]**{DRAFT}**Machine-specific configuration settings for the OS module** dThe Machine-specific configuration settings for the OS module are collected in OsModuleInstantiation.c*(RS\_MANI\_00023)*

|  |  |
| --- | --- |
| ***Class*** | **OsModuleInstantiation** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation |
| ***Note*** | This meta-class defines the attributes for the OS configuration on a specific machine.  **Tags:**atp.Status=draft |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **OsModuleInstantiation** | | | |
| ***Base*** | *ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| resourceGroup | ResourceGroup | \* | aggr | This represents the collection of ResourceGroups owned by the enclosing OsModuleImplementation.  **Tags:**atp.Status=draft |
| supportedTimer Granularity | TimeValue | 0..1 | attr | This attribute describes the supported timer granularity (TimeValue of one tick).  **Tags:**atp.Status=draft |

**Table 9.3: OsModuleInstantiation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***NonOsModuleInstantiation*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation | | | |
| ***Note*** | This meta-class defines the abstract attributes for the configuration of an adaptive autosar module other than the OS module.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *AdaptiveModuleInstantiation*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Subclasses*** | CryptoModuleInstantiation, DoIpInstantiation, GenericModuleInstantiation, IamModuleInstantiation, *Ids*  *PlatformInstantiation*, LogAndTraceInstantiation, NmInstantiation, TimeSyncModuleInstantiation, Ucm  ModuleInstantiation | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 9.4: NonOsModuleInstantiation**

AUTOSAR supports the configuration of ResourceGroups in the OsModuleInstantiation of the Machine that correspond for example to cgroups (aka control groups) in Linux. ResourceGroups provide a mechanism to manage system resources by partitioning constraints like cpuUsage and memUsage into groups that limit the resource usage for a collection of processes (see also [TPS\_MANI\_01017]).

**[constr\_1661]**{DRAFT} **Multiplicity of OsModuleInstantiation.resourceGroup** dAny given OsModuleInstantiation shall always define at least one resourceGroup.c*()*

The rationale for [constr\_1661] is that the StateDependentStartupConfig requires a reference to a ResourceGroup.

More information about the semantics of meta-class ResourceGroup can be found in [SWS\_OSI\_02001].

|  |  |
| --- | --- |
| ***Class*** | **ResourceGroup** |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation |
| ***Note*** | This meta-class represents a resource group that limits the resource usage of a collection of processes.  **Tags:**atp.Status=draft |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **ResourceGroup** | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| cpuUsage | PositiveInteger | 0..1 | attr | CPU resource limit in percentage of the total CPU capacity on the machine. **Tags:**atp.Status=draft |
| memUsage | PositiveInteger | 0..1 | attr | Memory limit in bytes.  **Tags:**atp.Status=draft |

**Table 9.5: ResourceGroup**

## 9.2 Persistency Deployment

**9.2.1 Overview**

This chapter explains the part of the support for persistent storage in terms of mapping of concrete storage models to the corresponding parts of the application software.

**[TPS\_MANI\_01205]**{DRAFT} **Semantics of meta-class PersistencyDeployment** dAbstract meta-class PersistencyDeployment provides shared attributes to more specific specializations.c*(RS\_MANI\_00027)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***PersistencyDeployment*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***Note*** | This abstract meta-class serves as a base class for concrete classes representing different aspects of persistency.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable*  *Element*, *Referrable*, *UploadableExclusivePackageElement*, *UploadablePackageElement* | | | |
| ***Subclasses*** | PersistencyFileStorage, PersistencyKeyValueStorage | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| maximum  AllowedSize | PositiveUnlimitedInteger | 0..1 | attr | The value of this attribute represents the maximum size allowed at deployment time for the enclosing Persistency Deployment.  **Tags:**atp.Status=draft |
| minimum  SustainedSize | PositiveInteger | 0..1 | attr | The value of this attribute represents the minimum size guaranteed at deployment time for the enclosing PersistencyDeployment.  **Tags:**atp.Status=draft |
| redundancy Handling | PersistencyRedundancy  Handling | \* | aggr | This aggregation represents the chosen approaches to handle redundancy.  **Tags:**atp.Status=draft |
| updateStrategy | PersistencyCollection  LevelUpdateStrategy  Enum | 1 | attr | This attribute shall be used to specify the update strategy of the respective PersistencyDeployment as a whole.  **Tags:**atp.Status=draft |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***PersistencyDeployment*** (abstract) | |  |  |
| version | StrongRevisionLabel  String | 0..1 | attr | The attribute represents the version of the PersistencyFile Storage or PersistencyKeyValueStorage.  **Tags:**atp.Status=draft |

**Table 9.6: PersistencyDeployment**

*UploadableExclusivePackageElement*

*PersistencyDeployment*

+

[0..1]

maximumAllowedSize: PositiveUnlimitedInteger

minimumSustainedSize: PositiveInteger

[0..1]

+

+

updateStrategy: PersistencyCollectionLevelUpdateStrategyEnum

version: StrongRevisionLabelString

+

[0..1]

*PersistencyRedundancyHandling*

[0..1]

scope: PersistencyRedundancyHandlingScopeEnum

+

PersistencyRedundancyCrc

PersistencyRedundancyMOutOfN

+

m: PositiveInteger

n: PositiveInteger

+

«enumeration»

PersistencyRedundancyHandlingScopeEnum

persistencyRedundancyHandlingScopeElement

persistencyRedundancyHandlingScopeStorage

*PersistencyRedundancyChecksum*

algorithmFamily: String

+

length: PositiveInteger

+

PersistencyRedundancyHash

[0..1]

+

initializationVectorLength: PositiveInteger

redundancyHandling

+

0..\*

**Figure 9.2: Modeling of the abstract base class PersistencyDeployment**

**[constr\_10035]**{DRAFT} **Completeness of the PersistencyDeployment.version**dThe PersistencyDeployment.version shall contain all the following parts:

* Major version
* Minor version
* Patch version
* Additional labels for pre-release version and build metadata **at the time when the manifest is complete**.c*()*

The version is used to determine whether an update or rollback of persistent data is required. When an application opens the PersistencyFileStorage or PersistencyKeyValueStorage, the Persistency module will check the PersistencyDeployment.version in the manifest against the stored version information.

If the stored version is lower than the manifest version, the Persistency module will update persistent data after creating a backup of the data (see [SWS\_PER\_00386] for more details).

If the stored version is higher than the manifest version, a rollback of the persistent data from the backup is required (see [SWS\_PER\_00396] for more details).

**[TPS\_MANI\_01321]**{DRAFT} **Semantics of meta-class PersistencyDeploymentElement** dMeta-class PersistencyDeploymentElement represents an abstract base class for the modeling of different aspects of persistency on element level.c

*(RS\_MANI\_00027)*

*Identifiable*

*PersistencyDeploymentElement*

+

updateStrategy: PersistencyElementLevelUpdateStrategyEnum

[0..1]

PersistencyFile

+

contentUri: UriString

[0..1]

+

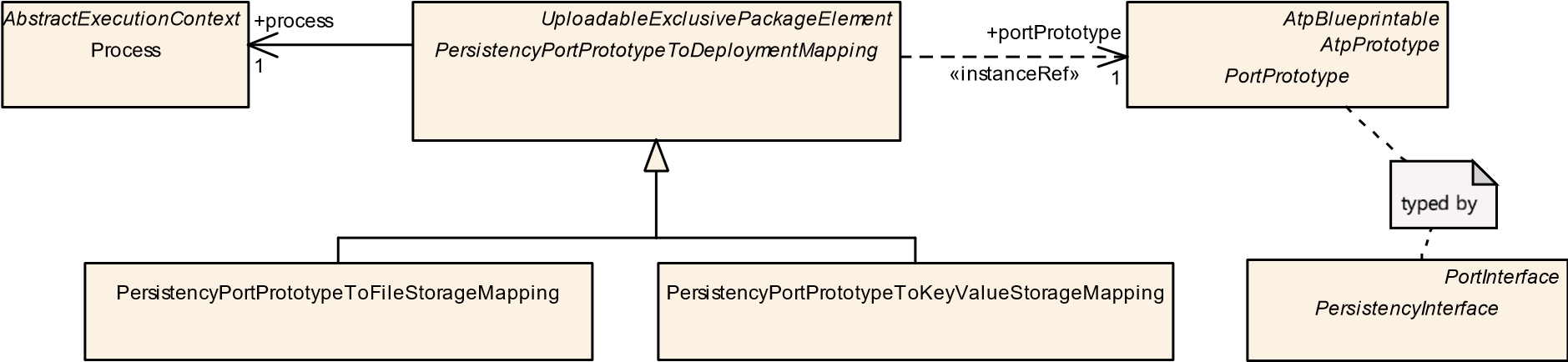
fileName: String

PersistencyKeyValuePair

**Figure 9.3: Modeling of the abstract base class PersistencyDeploymentElement**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***PersistencyDeploymentElement*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***Note*** | This abstract meta-class serves as a base class for concrete classes representing different aspects of elements of a PersistencyDeployment.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Subclasses*** | PersistencyFile, PersistencyKeyValuePair | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| updateStrategy | PersistencyElement  LevelUpdateStrategy  Enum | 0..1 | attr | This attribute can be used to specify the update strategy of the respective PersistencyDeploymentElement.  **Tags:**atp.Status=draft |

**Table 9.7: PersistencyDeploymentElement**



**Figure 9.4: Modeling of the abstract base class PersistencyPortPrototypeToDeploymentMapping**

**[TPS\_MANI\_01322]**{DRAFT} **Semantics of meta-class PersistencyPortPrototypeToDeploymentMapping** dMeta-class PersistencyPortPrototypeToDeploymentMapping represents an abstract base class for the modeling of the mapping of concrete persistency cases (key-value storage, file storage) to a PortPrototype and a Process.c*(RS\_MANI\_00027)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***PersistencyPortPrototypeToDeploymentMapping*** (abstract) | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***Note*** | This abstract bas class implements the shared functionality of all mapping between a PortPrototype, a Process, and a specific subclass of PersistencyDeployment.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable*  *Element*, *Referrable*, *UploadableExclusivePackageElement*, *UploadablePackageElement* | | | |
| ***Subclasses*** | PersistencyPortPrototypeToFileStorageMapping, PersistencyPortPrototypeToKeyValueStorageMapping | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| portPrototype | PortPrototype | 0..1 | iref | This reference represents the mapped PortPrototype.  **Tags:**atp.Status=draft  **InstanceRef implemented by:**PortPrototypeIn  ExecutableInstanceRef |
| process | Process | 1 | ref | This reference represents the process required as context for the mapping.  **Tags:**atp.Status=draft |

**Table 9.8: PersistencyPortPrototypeToDeploymentMapping**

**9.2.1.1 Redundancy Handling**

**[TPS\_MANI\_01206]**{DRAFT} **Modeling of redundancy in the context of PersistencyDeployment** dThe deployment level provides the ability to provide a more detailed definition of redundant behavior for both key-value storage and file storage.

This modeling is attached to the abstract base class PersistencyDeployment in order to let both aspects of persistency on the *AUTOSAR adaptive platform* benefit from the existence of meta-class PersistencyRedundancyHandling.c*(RS\_MANI\_00027)*

**[constr\_1710]**{DRAFT} **Consistency of values of attributes PersistencyInterface.redundancy and PersistencyRedundancyHandling.scope** dIf attribute PersistencyInterface.redundancy is set to value PersistencyRedundancyEnum.redundantPerElement then attribute PersistencyRedundancyHandling.scope shall be set to PersistencyRedundancyHandlingScopeEnum. persistencyRedundancyHandlingScopeElement for at least one PersistencyRedundancyHandling aggregated by the corresponding PersistencyDeploy-

ment.c*()*

**9.2.1.2 Update Handling**

**[TPS\_MANI\_01155]**{DRAFT} **PersistencyDeployment.updateStrategy overrides PersistencyInterface.updateStrategy**dThe value of attribute PersistencyDeployment.updateStrategy shall overrule the value of PersistencyInterface.updateStrategy for any combination of PersistencyInterface mapped to a PersistencyDeployment by means of a PersistencyPortPrototypeToDeploymentMapping.c*(RS\_MANI\_00027)*

**[TPS\_MANI\_01147]**{DRAFT} **Semantics of PersistencyDeployment.updateStrategy**dThe attribute PersistencyDeployment.updateStrategy shall be used to specify the strategy for updating the actual persistent elements.

This update strategy shall be applied to the PersistencyDeployment as a whole except for the explicitly modeled PersistencyDeploymentElements that define their own updateStrategy.c*(RS\_MANI\_00027)*

**[TPS\_MANI\_01157]**{DRAFT} **Semantics of updateStrategy on collection level** dThe semantics of attribute updateStrategy on collection level is specified in Table

9.9.c*(RS\_MANI\_00027)*

|  |  |  |
| --- | --- | --- |
| **updateStrategy** | **Use Case: Installation** | **Use Case: Update** |
| delete | irrelevant | delete all elements not contained in current manifest |
| keepExisting | irrelevant | keep all elements not contained in current manifest |

**Table 9.9: Semantics of updateStrategy on collection level**

**[TPS\_MANI\_01313]**{DRAFT} **Definition of updateStrategy on element level** dThe definition of the update strategy on element level is modeled by means of the abstract base class PersistencyDeploymentElement (and its attribute updateStrategy) from which the concrete sub-classes for persistency elements are derived.c *(RS\_MANI\_00027)*

**[TPS\_MANI\_01159]**{DRAFT} **Semantics of updateStrategy on element level** dThe semantics of attribute updateStrategy on element level is specified in Table

9.10.c*(RS\_MANI\_00027)*

|  |  |  |
| --- | --- | --- |
| **updateStrategy** | **Use Case: Installation** | **Use Case: Update** |
| delete | don’t create | remove |
| keepExisting | create | do nothing |
| overwrite | create | replace |

**Table 9.10: Semantics of updateStrategy on element level**

**[TPS\_MANI\_01148]**{DRAFT} **Semantics of PersistencyDeploymentElement. updateStrategy** dThe attribute PersistencyDeploymentElement.updateStrategy can be used to specify the strategy for updating the actual persistent element that corresponds to PersistencyDeploymentElement.c*(RS\_MANI\_00027)* **[TPS\_MANI\_01156]**{DRAFT}**PersistencyDeploymentElement.updateStrategy overrides PersistencyDeployment.updateStrategy** dThe value specified for PersistencyDeploymentElement.updateStrategy overrides the value of PersistencyDeployment.updateStrategy for this specific PersistencyDeploymentElement.c*(RS\_MANI\_00027)*

**[TPS\_MANI\_01182]**{DRAFT} **Value of PersistencyDeploymentElement.updateStrategy overrides PersistencyInterfaceElement.updateStrategy** dThe value of attribute PersistencyDeploymentElement.updateStrategy overrides the value of attribute PersistencyInterfaceElement.updateStrategyc *(RS\_MANI\_00027)*

This means that the integrator of the software gets the authority to either agree to the designer’s point of view or else overrule the designer’s decision based on superior knowledge regarding the integration strategy.

**9.2.1.3 Size Handling**

**[TPS\_MANI\_01196]**{DRAFT} **Semantics of PersistencyDeployment.minimumSustainedSize** dAttribute PersistencyDeployment.minimumSustainedSize can be used for the definition of a **minimum amount of storage** that the PersistencyDeployment will need to allocate from an integrator’s point of view.

It is the responsibility of the underlying platform to make sure that this minimum amount of storage is available at any time.c*(RS\_MANI\_00027)*

**[TPS\_MANI\_01197]**{DRAFT} **Semantics of PersistencyDeployment.maximumAllowedSize** dAttribute PersistencyDeployment.maximumAllowedSize can be used for the definition of the **maximum amount of storage** that the PersistencyDeployment may allocate at runtime from an integrator’s point of view.

The existence of PersistencyDeployment.maximumAllowedSize does not constitute a binding requirement to the platform that this amount of storage shall be available at any time.c*(RS\_MANI\_00027)*

For explanation, the amount of storage available shall be at least the sum of the values of minimumSustainedSize.

That said, it is consequently plausible that storage might be exceeded if more than the minimum amount of storage (let alone the maximum amount) is allocated by all the key-value storage at the same time.

**9.2.1.4 Security Handling**

The encryption and/or authentication of data stored in a Key-Value Storage or File Storage is described in the manifest by PersistencyDeploymentToCryptoKeySlotMapping or PersistencyDeploymentElementToCryptoKeySlotMapping that are described in more detail in chapter 13.4 and chapter

13.5.

If the PersistencyDeploymentToCryptoKeySlotMapping.keySlotUsage or

PersistencyDeploymentElementToCryptoKeySlotMapping.keySlotUsage is set to encryption, the Persistency cluster shall encrypt the data before storing it to the persistent memory or shall decrypt the data after reading it from persistent memory.

If the PersistencyDeploymentToCryptoKeySlotMapping.keySlotUsage or

PersistencyDeploymentElementToCryptoKeySlotMapping.keySlotUsage is set to verification, the Persistency cluster shall sign the data before storing it to the persistent memory or verify the signature of the data after reading it from persistent memory.

Please note that the PersistencyDeploymentToCryptoKeySlotMapping is able to define a verificationHash that shall by used by the PersistencyCluster to verify the data. The same is true for the PersistencyDeploymentElementToCryptoKeySlotMapping.verificationHash.

**9.2.2 Deployment of Persistent Key-Value Storage**

**[TPS\_MANI\_01079]**{DRAFT} **Semantics of PersistencyKeyValueStorage** dMeta-class PersistencyKeyValueStorage represents an actual key-value storage used for persistently storing data.c*(RS\_MANI\_00027)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PersistencyKeyValueStorage** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***Note*** | This meta-class represents the ability to model a key-value storage on deployment level.  **Tags:**  atp.Status=draft  atp.recommendedPackage=PersistencyKeyValueStorages | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*,  *PersistencyDeployment*, *Referrable*, *UploadableExclusivePackageElement*, *UploadablePackageElement* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| keyValuePair | PersistencyKeyValue  Pair | \* | aggr | This aggregation represents the key-value-pairs owned by the enclosing PersistencyKeyValueStorage.  **Tags:**atp.Status=draft |
| uri | UriString | 0..1 | attr | This attribute holds the storage location for the  PersistencyKeyValueStorage, e.g. file on the file system.  **Tags:**atp.Status=draft |

**Table 9.11: PersistencyKeyValueStorage**

**[TPS\_MANI\_01144]**{DRAFT} **Semantics of PersistencyKeyValuePair** dMetaclass PersistencyKeyValuePair represents an **entry** to a key-value storage (formalized by PersistencyKeyValueStorage) used for persistently storing data.c *(RS\_MANI\_00027)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | **PersistencyKeyValuePair** | | | |
| ***Package*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***Note*** | This meta-class represents the ability to formally model a key-value pair in the context of the deployment of persistency.  **Tags:**atp.Status=draft | | | |
| ***Base*** | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *PersistencyDeploymentElement*, *Referrable* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| initValue | ValueSpecification | 0..1 | aggr | This aggregation represents the ability to define an initial value for the value side of the key-value pair. Please note that it does not make sense to configure an initial value if the PersistencyDeploymentElement.updateStrategy is set to the value delete.  **Tags:**atp.Status=draft |
| valueDataType | AbstractImplementation  DataType | 1 | ref | This reference represents the data type applicable for the value of the key-value pair.  **Tags:**atp.Status=draft |

**Table 9.12: PersistencyKeyValuePair**

**[constr\_10083]**{DRAFT} **Existence of initial values for PersistencyKeyValuePair** dFor each PersistencyKeyValuePair, if the value of attribute updateStrategy is set to the value delete, then attribute PersistencyKeyValuePair. initValue shall not exist.c*()*

The modeling of PersistencyKeyValuePair aggregated in the role PersistencyKeyValueStorage.keyValuePair is optional. It would be possible to use persistency functionality regardless of the existence of keyValuePair.

However, the presence of keyValuePair gives more freedom and ways for the customization of behavior.

**[TPS\_MANI\_01078]**{DRAFT} **Semantics of PersistencyPortPrototype-**

**ToKeyValueStorageMapping** dMeta-class PersistencyPortPrototypeToKeyValueStorageMapping has the ability to map a specific PortPrototype referenced in the role portPrototype to a PersistencyKeyValueStorage referenced in the role keyValueStorage.

The mapping also comprises a reference to meta-class process in order to accommodate for the fact that identical combinations of keyValueStorage and portPrototype may or may not apply for a given Process that represents the enclosing Executable at runtime.c*(RS\_MANI\_00027)*

1. The description of the design elements may be moved to other model-related documents in the future. But for the time being, there is a coexistence of manifest-related and design-related model elements in this document. [↑](#footnote-ref-1)
2. On the other hand, both concepts of a “map” are justified in their respective “community” and choosing to name one of these very different in order so reduce overall potential confusion would probably not be applicable [↑](#footnote-ref-2)
3. As explained in [1] [↑](#footnote-ref-3)
4. And even if it were possible to extend ImplementationDataType towards a more or less clean support for C++ it may happen that further language bindings are added to the *AUTOSAR adaptive platform* for which further and further extensions of ImplementationDataType would be required. [↑](#footnote-ref-4)
5. This means that the definition of a namespace a::b is semantically different from the definition of a namespace b::a. [↑](#footnote-ref-5)
6. For more background regarding the definition and use of meta-class DataTypeMappingSet please refer to [1]. [↑](#footnote-ref-6)
7. This abbreviation stands for tag-length-value [↑](#footnote-ref-7)
8. For example, the usage of TLV encoding for arguments with attribute direction set to in, but **not** for arguments with attribute direction set to out. [↑](#footnote-ref-8)
9. If a structure consists only of optional elements, it would be hard to detect the case where an array element carries such a structure that happens to set all elements to non-available. [↑](#footnote-ref-9)
10. i.e. during the creation of the execution manifest [↑](#footnote-ref-10)
11. From the methodological point of view, the creation of the diagnostic mapping is typically considered a design-time activity. [↑](#footnote-ref-11)
12. In other words, if references are needed between design-related and deployment-related metaclasses then the direction of these references shall always point from deployment to design. [↑](#footnote-ref-12)
13. on the end of the application software [↑](#footnote-ref-13)
14. For more information, please refer to section 14.2. [↑](#footnote-ref-14)