Document Title	Specification of Manifest
Document Category	Template specification (TPS)
Document UID	713
Document Status	published
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	R26-11

Document Change History

Date	Release	Changed by	Description
2026-11-26	R26-11	AUTOSAR Release Management	• Improve support for SOVD • Improve support for Crypto • Improve interaction of functional clusters • minor corrections / clarifications / editorial changes
2025-11-27	R25-11	AUTOSAR Release Management	• Improve support for SOVD • Add support for suspend-to-RAM • Rework support for cryptography • minor corrections / clarifications / editorial changes
2024-11-27	R24-11	AUTOSAR Release Management	• Improve configuration of diagnostic transport layers • Improve configuration of IEEE 1722-related transport • Offset Time Base Feature Removal • minor corrections / clarifications / editorial changes
2023-11-23	R23-11	AUTOSAR Release Management	• Improve diagnostic configuration • Improve firewall configuration • minor corrections / clarifications / editorial changes
2022-11-24	R22-11	AUTOSAR Release Management	• Support for MACsec • Support for firewall • Modeled State Management • Support for Service-oriented Vehicle Diagnostic • minor corrections / clarifications / editorial changes



△

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 26
1.1 Modeling Approach . . . 27
1.1.1 General Considerations . . . 27
1.1.2 Modeling of Diagnostic Configuration . . . 28
1.2 The Term Service . . . 29
1.3 Document Conventions . . . 29
1.4 About Tables . . . 31
2 Big Picture of Manifest Definition 32
2.1 Design vs. Target Configuration . . . 32
2.1.1 Preface . . . 32
2.1.2 Overview . . . 32
2.1.3 Relation between Design and Target Configuration Models . . . 33
2.1.4 Structure of the document . . . 34
2.2 About Manifest . . . 34
2.3 Serialization Format . . . 34
2.4 Scope . . . 35
2.5 Manifests described in this Document . . . 35
3 Application Design 37
3.1 Overview . . . 37
3.2 Application Structure . . . 37
3.2.1 Software Component . . . 37
3.2.2 Executable . . . 39
3.2.2.1 Overview . . . 39
3.2.2.2 Category . . . 41
3.2.2.3 Root Software Component Prototype . . . 42
3.2.2.4 Reporting Behavior . . . 45
3.2.2.5 Implementation Props . . . 46
3.2.2.6 Logging . . . 47
3.2.3 TraceSwitchConfig . . . 47
3.2.4 Interaction Endpoint for Application . . . 49
3.2.4.1 Service-oriented Communication . . . 49
3.2.4.2 Port Prototype Props . . . 50
3.2.5 Port Prototype ComSpec . . . 52
3.2.5.1 Port Prototypes typed by Service Interfaces . . . 54
3.2.5.2 Port Prototypes typed by Persistency Data Interfaces . . . 65
3.2.6 Process Design . . . 68
3.3 Data Type . . . 69
3.3.1 Overview . . . 69
3.3.2 ApplicationDataType . . . 70
3.3.2.1 String Data Type . . . 70

3.3.2.2 Associative Map Data Type . . . 73
3.3.2.3 Attributes of SwDataDefProps . . . 77
3.3.3 CppImplementationDataType . . . 79
3.3.3.1 Overview . . . 79
3.3.3.2 Attributes of SwDataDefProps . . . 93
3.3.3.3 Primitive Data Types . . . 97
3.3.3.4 String Data Type . . . 97
3.3.3.5 Array Data Type . . . 98
3.3.3.6 Vector Data Type . . . 99
3.3.3.7 Struct Data Type . . . 101
3.3.3.8 Enumeration Data Type . . . 101
3.3.3.9 Map Data Type . . . 102
3.3.3.10 Variant Data Type . . . 103
3.3.3.11 Bitfield Data Type . . . 104
3.3.4 Compatibility of ApplicationDataType and CppImplementation-DataType . . . 105
3.3.5 Optional Members in complex Data Structures . . . 108
3.3.5.1 Background . . . 108
3.3.5.2 Definition of Optionality . . . 110
3.3.6 Usage of constant Values . . . 112
3.4 Serialization Of Data . . . 112
3.4.1 Default Values for Serialization Properties . . . 113
3.4.2 Individual Definition of Serialization Properties . . . 120
3.4.3 Assignment of TLV properties . . . 126
3.4.3.1 Assignment of TLV Data IDs . . . 126
3.4.3.2 Assignment of Wire Type Selection . . . 133
3.5 Identity and Access Management . . . 133
3.5.1 Com Grant Design . . . 134
3.5.2 Remote access control . . . 140
3.5.2.1 Remote Subject in the Case of TLS . . . 142
3.5.2.2 Remote Subject in the Case of IPsec . . . 144
3.5.2.3 Remote Subject in the Case of IP Communication . . . 146
3.5.2.4 Remote Subject in the Case of SecOC Communication . . . 147
3.6 Intrusion Detection Design . . . 147
3.7 Relation to Crypto System Design . . . 149
4 Application Port Interface Design 152
4.1 Overview . . . 152
4.2 Service Interface . . . 152
4.2.1 Overview . . . 152
4.2.2 Event . . . 155
4.2.3 Trigger . . . 155
4.2.4 Field . . . 156
4.2.5 Method . . . 158

4.2.5.1 Fire and Forget Method . . . 159
4.2.6 Versioning of ServiceInterfaces . . . 160
4.2.6.1 Versioning driven by transport layer . . . 161
4.2.7 Namespace . . . 162
4.2.8 Error Handling . . . 164
4.2.9 Service Interface Data Type Mapping . . . 169
4.3 Service Interface Mapping . . . 172
4.4 Service Interface Element Mapping . . . 177
4.4.1 Overview . . . 177
4.4.2 Service Interface Event Mapping . . . 180
4.4.3 Service Interface Trigger Mapping . . . 181
4.4.4 Service Interface Field Mapping . . . 182
4.4.5 Service Interface Method Mapping . . . 183
4.5 State Management Interface . . . 185
4.5.1 Overview . . . 185
4.5.2 State Management Interfaces for Notifications . . . 187
4.5.3 State Management Interfaces for Triggers . . . 188
4.5.4 State Management Interfaces for Errors . . . 189
4.6 Execution Management Interface . . . 190
4.6.1 State Client Interface . . . 190
4.7 Persistency Interface . . . 191
4.7.1 Overview . . . 191
4.7.1.1 The big Picture . . . 191
4.7.1.2 Modeling of Persistency Interface . . . 192
4.7.1.3 Redundancy Handling . . . 193
4.7.1.4 Update Handling . . . 198
4.7.2 Persistency Key Value Storage Interface . . . 199
4.7.3 Persistency File Storage Interface . . . 204
4.8 Time Synchronization Interface . . . 206
4.9 Platform Health Management Interface . . . 209
4.9.1 Overview . . . 209
4.9.2 Supervised Entities and Checkpoints . . . 209
4.9.3 Recovery notification to State Management . . . 212
4.10 Diagnostic Interface . . . 213
4.10.1 Overview . . . 213
4.10.2 Diagnostic Interfaces for UDS . . . 215
4.10.2.1 Diagnostic Routine Interface . . . 215
4.10.2.2 Interface to Data Identifier and Element of Data Identifier . . . 218
4.10.2.3 Security Level Interface . . . 221
4.10.2.4 Service Validation Interface . . . 222
4.10.2.5 Generic UDS Interface . . . 222
4.10.2.6 Diagnostic Interfaces for Upload and Download . . . 223
4.10.2.7 Interface to support managing the EcuReset . . . 224

4.10.2.8 Diagnostic Authentication Interface . . . 225
4.10.2.9 Diagnostic External Authentication Interface . . . 226
4.10.2.10 Diagnostic Communication Control Interface . . . 227
4.10.2.11 Diagnostic Request File Transfer Interface . . . 228
4.10.2.12 Diagnostic Input/Output Control Interface . . . 228
4.10.3 Diagnostic Interfaces for Fault Memory . . . 230
4.10.3.1 Interface for providing Data to Extended Data Record . . . 230
4.10.3.2 Interface to diagnostic Events . . . 231
4.10.3.3 Interface to diagnostic Condition . . . 232
4.10.3.4 Interface for multiple diagnostic Resources . . . 233
4.10.3.5 Indicator Interface . . . 235
4.10.3.6 Operation Cycle Interface . . . 236
4.10.4 Diagnostic Interfaces for SOVD . . . 237
4.10.4.1 Overview . . . 237
4.10.4.2 Port Interface for SOVD Client Authorization . . . 237
4.10.4.3 Port Interface for SOVD Proximity Challenge . . . 238
4.10.4.4 Port Interface for SOVD Bulk Data . . . 238
4.10.4.5 Port Interface for SOVD Service Validation . . . 239
4.10.4.6 Port Interface for SOVD Update . . . 239
4.10.4.7 Port Interface for SOVD Configuration . . . 240
4.10.4.8 Port Interface for SOVD Operation . . . 240
4.10.4.9 Port Interface for SOVD Content Access . . . 243
4.10.5 Diagnostic Interfaces for DoIP . . . 245
4.11 Crypto Interfaces . . . 248
4.11.1 Interaction with Crypto Software . . . 248
4.11.2 Crypto Key Slot Interface . . . 249
4.11.3 Crypto Certificate Interface . . . 252
4.11.4 Crypto Certificate Group Interface . . . 253
4.11.5 Crypto Provider Interface . . . 253
4.12 Raw Data Stream Interface . . . 254
4.13 Intrusion Detection Interface . . . 257
4.13.1 Security Event Report Interface . . . 258
4.13.2 Intrusion Detection Timestamp Provider Interface . . . 258
4.13.3 Intrusion Detection Context Provider Interface . . . 259
4.13.4 Intrusion Detection Reporting Mode Provider Interface . . . 259
4.13.5 Intrusion Detection Qualified Events Receiver Interface . . . 259
4.14 Log And Trace Interface . . . 260
4.15 Firewall Interface . . . 262
4.16 Network Management Interface . . . 263
4.17 Suspend to RAM Interface . . . 263

5 Diagnostic Design 266
5.1 Extensions to the Diagnostic Common Props . . . 266
5.1.1 Diagnostic SOVD Props . . . 269
5.1.2 Diagnostic Address . . . 270
5.1.3 Diagnostic Authentication Configuration . . . 271
5.2 Diagnostic Mapping . . . 272
5.2.1 Overview . . . 272
5.2.1.1 Mapping of multiple Diagnostic Resources to a PortPrototype 276
5.2.2 Fault-Memory-related Diagnostic Mapping . . . 277
5.2.2.1 Diagnostic Monitor to Port Mapping . . . 277
5.2.2.2 Diagnostic Event to Port Mapping . . . 282
5.2.2.3 Multiple Diagnostic Event to Port Mapping . . . 284
5.2.2.4 Diagnostic Operation Cycle to Port Mapping . . . 285
5.2.2.5 Diagnostic Enable Condition to Port Mapping . . . 287
5.2.2.6 Diagnostic Clear Condition to Port Mapping . . . 289
5.2.2.7 Multiple Diagnostic Condition to Port Mapping . . . 291
5.2.2.8 Diagnostic Indicator to Port Mapping . . . 293
5.2.2.9 Diagnostic Memory Destination to Port Mapping . . . 294
5.2.2.10 Diagnostic Extended Data Record Port Mapping . . . 298
5.2.3 UDS-related Diagnostic Mapping . . . 300
5.2.3.1 Diagnostic Security to Port Mapping . . . 300
5.2.3.2 Diagnostic Data Identifier to Port Mapping . . . 302
5.2.3.3 Diagnostic Generic UDS Service Handler to Port Mapping . . . 307
5.2.3.4 Diagnostic Service Validation Mapping . . . 314
5.2.3.5 Diagnostic Authentication Mapping . . . 315
5.2.3.6 Diagnostic External Authentication Mapping . . . 318
5.2.3.7 Diagnostic Input/Output Control Mapping . . . 319
5.2.4 SOVD-related Diagnostic Mapping . . . 321
5.2.4.1 Diagnostic SOVD Proximity Challenge Mapping . . . 321
5.2.4.2 Diagnostic SOVD Authorization Port Mapping . . . 323
5.2.4.3 Diagnostic SOVD Bulk Data Port Mapping . . . 325
5.2.4.4 Diagnostic SOVD Update Port Mapping . . . 328
5.2.4.5 Diagnostic SOVD Service Validation Port Mapping . . . 330
5.2.4.6 Diagnostic SOVD Configuration Port Mapping . . . 332
5.2.4.7 Diagnostic SOVD Data Port Mapping . . . 334
5.2.4.8 Diagnostic SOVD Operation Port Mapping . . . 337
5.2.4.9 Diagnostic SOVD Configuration to SOVD Content Mapping . . . 339
5.2.5 DoIP-related Diagnostic Mapping . . . 340
5.2.5.1 Diagnostic DoIP Mapping . . . 340
5.2.5.2 Diagnostic DoIP Group Identification Mapping . . . 341
5.2.5.3 Diagnostic DoIP Activation Line Mapping . . . 343
5.2.5.4 Diagnostic DoIP Entity Identification Mapping . . . 345
5.2.5.5 Diagnostic DoIP Power Mode Mapping . . . 347

5.2.5.6 Diagnostic DoIP Custom Payload Mapping . . . 349
5.3 Diagnostic Clear Condition . . . 354
5.4 Security Access . . . 356
5.5 Diagnostic Provided Data Mapping . . . 356
5.6 Usage of Diagnostic Environmental Data Element Condition . . . 358
5.7 SOVD Configuration Elements . . . 362
5.7.1 SOVD Service Instance . . . 362
5.7.1.1 Overview . . . 362
5.7.1.2 SOVD Log . . . 363
5.7.1.3 SOVD Update . . . 364
5.7.1.4 SOVD Fault Memory Access . . . 365
5.7.1.5 Diagnostic SOVD Data . . . 367
5.7.1.6 SOVD Operation . . . 370
5.7.2 SOVD Method . . . 373
5.7.3 SOVD Lock . . . 376
5.7.4 SOVD Content . . . 377
5.7.4.1 Overview . . . 377
5.7.4.2 SOVD Composite Content . . . 380
5.7.4.3 SOVD Primitive Content . . . 383
5.7.5 Environmental Data Condition to SOVD Content Element . . . 385
5.7.6 SOVD Snapshots . . . 386
5.8 Diagnostic Miscellanea . . . 387
6 Machine Design 389
6.1 Overview . . . 389
6.2 Processor . . . 390
6.2.1 Overview . . . 390
6.2.2 Core Affinity . . . 391
6.3 Functional Cluster Design . . . 392
6.3.1 Ids Functional Cluster Design . . . 393
6.3.2 Sovd Functional Cluster Design . . . 395
6.3.2.1 SovdServerDesign . . . 397
6.3.2.2 SovdGatewayDesign . . . 398
6.3.3 DoIp Functional Cluster Design . . . 400
6.4 Ethernet Endpoint Configuration for Platform Modules . . . 409
6.5 Network Design aspects of Machine Design . . . 411
6.6 Service Discovery Configuration . . . 415
6.6.1 SOME/IP Service Discovery Configuration . . . 415
6.7 TcpIp stack configuration properties . . . 417
7 System Design 418
7.1 Overview . . . 418
7.2 Specification of Communication System Structure . . . 420
7.2.1 Network connection . . . 421

7.2.1.1 Support of 10BASE-T1S Network Topologies . . . 425
7.2.1.2 Support of Ethernet tunneling through CAN XL . . . 426
7.2.1.3 IP configuration properties . . . 428
7.2.1.4 TCP and UDP configuration properties . . . 434
7.2.1.5 ICMP configuration properties . . . 436
7.2.2 Securing Communication with IPsec . . . 437
7.2.3 MACsec . . . 445
7.2.4 Partial Network . . . 453
7.3 Specification of Application Software System Structure . . . 455
7.4 Modeling of Service-Oriented Communication between Classic and Adaptive Platform . . . 456
7.4.1 Method Mapping . . . 458
7.4.2 Event Mapping . . . 459
7.4.3 FieldMapping . . . 460
7.4.4 Fire And Forget Method Mapping . . . 462
7.5 Crypto System Design . . . 464
7.6 Process Design to Machine Design Mapping . . . 468
7.7 Ethernet Synchronized Time Design . . . 469
7.8 SOVD System Design . . . 473
8 Sub-System Design 478
8.1 Overview . . . 478
8.2 Software Cluster Design . . . 478
8.3 Relevance of Software Cluster Design for Diagnostics . . . 484
8.4 Software Cluster Dependency . . . 484
8.4.1 Overview . . . 484
8.4.2 Modeling of a Software Cluster Dependency . . . 485
8.5 Provided and required Services of Software Cluster Design . . . 489
8.6 Mapping of Services to Executables . . . 492
9 Signal-based Communication 496
9.1 Overview . . . 496
9.2 Signal-based Prerequisites . . . 501
9.2.1 Message Header for IP based transport . . . 502
9.2.2 Message Identification for IEEE1722 ACF based transport . . . 505
9.3 Signal-based Deployment . . . 507
9.4 Signal/Service Translation Mapping . . . 508
9.4.1 Handling of ISignalGroups . . . 509
9.4.2 Direction of Signal/Service Translation Mapping . . . 510
9.4.3 SignalBasedEvent Mapping . . . 512
9.4.4 SignalBasedTrigger Mapping . . . 515
9.4.5 SignalBasedField Mapping . . . 516
9.4.6 SignalBased Fire and Forget Method Mapping . . . 518
9.5 Service Discovery and Static Communication Configuration . . . 520

9.5.1 Service discovery control . . . 522
9.5.2 Service control right after translation start . . . 524
9.5.3 Service control due to availability of related service instance . . . 525
9.5.3.1 Signal-service-translation . . . 525
9.5.3.2 Service-signal-translation . . . 525
9.5.4 Static Communication Configuration . . . 525
9.5.4.1 Static Communication Configuration for the Server . . . 526
9.5.4.2 Static Communication Configuration for the Client . . . 526
9.6 Translation Behavior . . . 527
9.6.1 `ServiceInterface` Representation of Translation Data . . . 527
9.6.1.1 Optional Elements in the Case `signal-service-translation` . . . 527
9.6.1.2 Optional Elements in the Case `service-signal-translation` . . . 528
9.6.2 Event and update-bit Semantics . . . 528
9.6.3 Data Filtering . . . 531
9.6.4 Translation from one Source . . . 533
9.6.5 Translation from several Sources . . . 534
9.6.6 Signal-Service-Translation to several Targets . . . 535
9.7 Translation pass-through Composition . . . 535
9.8 Expected Features of Classic Platform . . . 536
9.8.1 Processing Order . . . 536
9.8.2 Reception Data Filter . . . 536
9.8.3 Reception of Invalid Signal . . . 537
9.8.4 Update Bit Handling . . . 537
9.8.5 Transfer Properties and Transmission Modes for Service-Signal-Translation . . . 538
9.8.6 Deadline Monitoring . . . 539
9.8.7 Signal and IPdu Transmission . . . 539
9.8.8 IPdu Multiplexing . . . 540
9.9 End-to-End considerations . . . 541
9.9.1 Safety . . . 542
9.9.1.1 Signal-service-translation . . . 544
9.9.1.2 Service-signal-translation . . . 545
9.9.2 Security . . . 545
9.9.2.1 Restrictions on Classic Platform SecOC . . . 547
9.9.2.2 Link between Classic Platform SecOC and `CryptoKeySlot` 547
10 Execution Manifest 550
10.1 Overview . . . 550
10.2 Process . . . 550
10.2.1 Process relates to Executable . . . 552
10.2.2 Process States . . . 553
10.3 Startup Configuration . . . 554

10.3.1 State-dependent Startup Configuration . . . 554
10.3.2 Process Attributes . . . 559
10.3.3 Scheduling . . . 561
10.3.4 Process Arguments . . . 562
10.3.5 Association with Resource Group . . . 562
10.3.6 Execution Dependency . . . 563
10.3.7 Assignment of Processes to Function Group states . . . 567
10.3.8 Resource Consumption Boundaries . . . 568
10.3.9 Error and Termination Behavior . . . 569
10.4 Function Groups . . . 569
10.4.1 Semantics of Function Group . . . 569
10.5 Reporting of Security Events . . . 572
11 Machine Manifest 573
11.1 Machine . . . 573
11.2 Environment Variable . . . 575
11.3 Process To Machine Mapping . . . 576
11.3.1 General Modeling Approach . . . 576
11.3.2 Core Affinity . . . 579
11.3.3 Default Start-up and Termination Timeout . . . 580
12 Platform Module Development 582
12.1 OS Module Configuration . . . 584
12.2 Persistency Target-Configuration . . . 586
12.2.1 Overview . . . 586
12.2.1.1 Redundancy Handling . . . 591
12.2.1.2 Update Handling . . . 592
12.2.1.3 Size Handling . . . 593
12.2.1.4 Security Handling . . . 594
12.2.1.5 Preference Rules . . . 594
12.2.2 Target-Configuration of Persistent Key-Value Storage . . . 595
12.2.3 Target-Configuration of File Storage . . . 599
12.3 Platform Health Management Target-Configuration . . . 604
12.3.1 Overview . . . 604
12.3.2 Relation between design and Target-Configuration . . . 607
12.3.3 Supervision Target-Configuration . . . 608
12.3.3.1 Global supervision . . . 611
12.3.3.2 Supervision Mode . . . 613
12.3.3.3 Supervision Mode condition . . . 615
12.3.3.4 `NoSupervision` definition . . . 619
12.3.3.5 `AliveSupervision` definition . . . 620
12.3.3.6 `CheckpointTransition` definition . . . 623
12.3.3.7 `LogicalSupervision` definition . . . 624
12.3.3.8 `DeadlineSupervision` definition . . . 625

12.3.3.9 NoCheckpointSupervision definition . . . 626
12.3.4 Supervision Recovery Notification Target-Configuration . . . 627
12.3.5 Recovery Notification . . . 628
12.4 Time Synchronization Target-Configuration . . . 630
12.4.1 Overview . . . 630
12.4.2 Time Synchronization Functional Cluster Configuration . . . 631
12.4.3 Time Base . . . 632
12.4.3.1 Synchronized Time Base . . . 632
12.4.3.2 Persistent Time Base Value Storage . . . 635
12.4.4 Time Base to Port Prototype Mapping . . . 636
12.5 DoIP Configuration . . . 638
12.6 Log and Trace Module Configuration . . . 639
12.6.1 Log and Trace Target-Configuration . . . 640
12.6.2 Relation between Design and Target-Configuration . . . 645
12.7 Network Management configuration . . . 648
12.7.1 UDP Network Management . . . 648
12.7.2 Relation between Application Software and Network Handles . . . 655
12.7.3 Relation between Function Group states and Network Handles . . . 656
12.7.3.1 UdpNm configuration constraints . . . 660
12.7.4 CAN XL Network Management . . . 661
12.8 Update and Configuration Management . . . 664
12.8.1 Overview . . . 664
12.8.2 Retry Strategy . . . 666
12.8.3 UCM Master Module Instantiation . . . 667
12.8.4 UCM Subordinate Module Instantiation . . . 668
12.9 CM configuration . . . 670
12.9.1 Com Grant Target-Configuration . . . 672
12.9.1.1 Com Field Grant Target-Configuration . . . 672
12.9.1.2 Com Method Grant Target-Configuration . . . 674
12.9.1.3 Com Event Grant Target Configuration . . . 675
12.9.1.4 Com Trigger Grant Target-Configuration . . . 676
12.9.1.5 Com Offer Service Grant Target-Configuration . . . 678
12.9.2 Remote Access Control . . . 679
12.10 Crypto Target-Configuration . . . 681
12.10.1 Crypto Provider . . . 682
12.10.2 Crypto Key Slot . . . 684
12.10.2.1 Crypto Key Slot to Port Prototype Mapping . . . 686
12.10.2.2 Crypto Key Slot to Client Port Prototype Mapping . . . 688
12.10.3 Crypto Certificate . . . 690
12.10.4 Crypto Certificate Group . . . 692
12.10.5 Relation to Crypto System Design . . . 695
12.11 IdsM Target-Configuration . . . 695
12.11.1 IdsM Instantiation . . . 695

12.11.2 Time Stamps for Security Events via Time-Base Resource 699
12.11.3 Mapping to Timestamp Provider 699
12.11.4 Mapping to Context Provider 701
12.11.5 Mapping to Reporting Mode Provider 703
12.11.6 Mapping to Qualified Event Receiver 704
12.11.7 Target-Configuration for Security Events 706
12.12 Service-oriented Vehicle Diagnostics 708
12.12.1 SOVD Server 709
12.12.2 SOVD Gateway 709
12.13 Firewall Target-Configuration 710
12.13.1 Firewall Rules on Datalink Layer 716
12.13.2 Firewall Rules on Network Layer 717
12.13.3 Firewall Rules on Transport Layer 720
12.13.4 Firewall Rules via Byte Pattern 721
12.13.5 Firewall Deep Packet Inspection of SOME/IP SD and SOME/IP Messages 722
12.13.6 Firewall Deep Packet Inspection of DoIP Messages 723
12.13.7 Firewall Deep Packet Inspection of DDS Messages 724
12.14 State Management Target-Configuration 726
12.14.1 Overview 726
12.14.2 Requests to State Management 728
12.14.2.1 Trigger Requests to State Management 730
12.14.2.2 Error Requests to State Management 731
12.14.3 Notifications from State Management 732
12.14.4 Rules 733
12.14.4.1 Overview 733
12.14.4.2 Trigger Rules 737
12.14.4.3 Error Rules 738
12.14.5 Action Items 739
12.14.5.1 State Machine Action Item 741
12.14.5.2 Sync Action Item 744
12.14.5.3 Function Group Action Item 745
12.14.5.4 Network Management Action Item 747
12.14.5.5 Sleep Action Item 749
12.14.5.6 Suspend to RAM Action Item 750
12.14.6 State Client 752
12.14.7 Error Handling inside State Management 754
12.14.7.1 Request Function Group State 756
12.15 Suspend to RAM 756
12.15.1 Suspend to RAM Module Instantiation 757
12.15.2 Suspend to RAM Module Satellite Mapping 758
12.15.3 Suspend to RAM Module Hub Mapping 760
12.16 Generic Diagnostic TransportInstantiation 762

13 Service Instance Manifest 764
13.1 ApplicationEndpoint . . . 764
13.1.1 VLAN Priority . . . 766
13.2 Service Interface Deployment . . . 767
13.2.1 SOME/IP Service Interface Deployment . . . 773
13.2.2 DDS Service Interface Deployment . . . 784
13.2.3 User Defined Service Interface Deployment . . . 789
13.3 Service Instance Target-Configuration . . . 793
13.3.1 Handling of Client ID . . . 800
13.3.2 SOME/IP Service Instance Deployment . . . 802
13.3.2.1 Provided Service Instance . . . 802
13.3.2.2 Required Service Instance . . . 826
13.3.2.3 Provided Service Instance with static remote peers . . . 837
13.3.2.4 Required Service Instance with static remote peers . . . 839
13.3.3 DDS Service Instance Deployment . . . 840
13.3.3.1 Provided DDS Service Instance . . . 842
13.3.3.2 Required DDS Service Instance . . . 846
13.3.3.3 DDS Service Instance to Machine mapping . . . 848
13.3.4 User Defined Service Instance Deployment . . . 849
13.3.5 Service Deployment Relations . . . 850
13.4 End-to-End Protection . . . 851
13.5 Secure Communication . . . 860
13.5.1 Secure Communication over TLS . . . 864
13.5.2 Secure Communication over SecOC . . . 874
13.5.3 Secure Communication over DDS . . . 877
14 Raw Data Stream Manifest 882
14.1 Raw Data Stream Deployment . . . 882
14.2 Raw Data Stream Mapping . . . 882
14.3 Raw Data Streams over Ethernet using IP based Protocol . . . 883
14.3.1 Ethernet Raw Data Stream Client Mapping . . . 888
14.3.2 Ethernet Raw Data Stream Server Mapping . . . 889
14.4 Raw Data Streams over Ethernet using MAC Layer Protocol . . . 891
14.4.1 Raw Data Streams over Ethernet using IEEE1722 Protocol . . . 892
14.4.2 Raw Data Streams for Control Messages over Ethernet using IEEE1722 ACF Protocol . . . 895
15 Cross-Functional Cluster interaction 899
15.1 Com Certificate To Crypto Certificate Mapping . . . 901
15.2 Com Key To Crypto Key Slot Mapping . . . 902
15.3 Com SecOc To Crypto Key Slot Mapping . . . 904
15.4 Persistency Deployment To Crypto Key Slot Mapping . . . 905
15.5 Persistency Deployment Element To Crypto Key Slot Mapping . . . 907
15.6 Persistency Deployment To Dlt Log Sink Mapping . . . 908

15.7 Functional Cluster To Dlt Log Sink Mapping . . . 909
15.8 Ucm To Time Base Resource Mapping . . . 909
15.9 Functional Cluster Interacts With Persistency Deployment Mapping . . . 910
15.10 Adaptive Module Instantiation to Security Event Definition Mapping . . . 912
15.11 State Management interacts with Network Management and vice versa 913
15.11.1 State Management interacts with Network Management . . . 913
15.11.2 Network Management interacts with State Management . . . 915
15.12 Functional Cluster reports Production Error . . . 916
15.13 Idsm needs Crypto Key for Signature Generation . . . 918

16 Software Distribution 920
16.1 Overview . . . 920
16.2 Uploadable Package Element . . . 921
16.3 Software Cluster . . . 924
16.3.1 Software Cluster General Modeling . . . 924
16.3.2 Relevance of Software Cluster for Diagnostics . . . 930
16.3.2.1 Diagnostic Props . . . 930
16.3.2.2 Relation to the Diagnostic Contribution Set . . . 931
16.3.2.3 Diagnostic Service Validation . . . 932
16.3.3 Contract between Software Cluster and Software Cluster Design . . . 934
16.3.4 References between Software Clusters . . . 934
16.3.5 Software Cluster Artifact Checksum . . . 936
16.3.6 Software Cluster Artifact Locator . . . 937
16.4 Software Package . . . 939
16.5 Vehicle Package . . . 944
16.5.1 Overview . . . 944
16.5.2 VehicleRolloutStep . . . 948
16.5.3 UcmStep . . . 950
16.5.4 SoftwarePackageStep . . . 951

17 Interoperability between Classic Platform and Adaptive Platform 954
17.1 Usage of major Version in the SOME/IP network binding . . . 954

A Reference Material 956
A.1 Terms and Abbreviations . . . 956
A.2 Imposition Times of Constraints . . . 958

B Examples 959
B.1 Data Type . . . 959
B.1.1 Definition of Map Data Type . . . 959
B.1.2 Definition of String Data Type . . . 960
B.1.3 Definition of String Data Type . . . 960
B.1.4 Definition of Vector Data Type . . . 961
B.1.5 Definition of Struct Data Type . . . 963
B.1.6 Definition of Enumeration Data Type . . . 963

B.1.7 Definition of Map Data Type . . . 964
B.1.8 Definition of Variant Data Type . . . 964
B.1.9 Scenarios to define a Vector . . . 965
B.2 Service-oriented Communication . . . 966
B.2.1 Radar and Camera ServiceInterface example . . . 966
B.2.2 Service Interface Mapping . . . 970
B.2.2.1 Service Interface Element Mapping . . . 970
B.2.2.2 Service Interface Event Mapping . . . 971
B.2.2.3 Service Interface Field Mapping . . . 972
B.2.2.4 Service Interface Method Mapping . . . 972
B.2.3 Service Instance Deployment . . . 972
B.2.3.1 Service Instance Deployment by Service Interface Mapping . . . 972
B.2.3.2 Service Instance Deployment by Service Interface Element Mapping . . . 974
B.2.4 Service Instance Mapping . . . 977
B.2.4.1 SOME/IP Service Instance Mapping . . . 977
B.2.4.2 User-Defined Service Instance Mapping . . . 977
B.2.5 Definition of VLAN Priority . . . 978
B.2.6 Static Configuration of remote Peers . . . 979
B.2.6.1 Provided Service Instance with remote Peers . . . 979
B.2.6.2 Required Service Instance with remote Peers . . . 980
B.3 Execution Management . . . 981
B.3.1 Definition of Startup Configuration . . . 981
B.4 Persistency . . . 984
B.4.1 Definition of Persistent Data . . . 984
B.4.2 Definition of Persistent File . . . 985
B.5 Interaction with the PHM . . . 986
B.5.1 Relation between Instance Specifier and Instance Ref . . . 986
B.5.2 Phm Application Design example . . . 986
B.5.3 Phm configuration example . . . 987
B.6 State Management . . . 988
B.6.1 Consideration of User Perspective . . . 988
B.6.2 Scenario for Trigger . . . 989
B.6.3 Scenario for Error . . . 993
B.7 Crypto . . . 996
B.7.1 Relation between a Crypto Certificate and a Crypto Key Slot . . . 996
B.8 Machine Manifest . . . 996
B.8.1 Distributed Definition of Environment Variables . . . 996
B.9 Ethernet Raw Data Stream Configuration . . . 997
B.9.1 Preamble . . . 997
B.9.2 Fully formalized Model Example . . . 999
B.9.3 Only Client is formalized . . . 1000
B.9.4 Only Server is formalized . . . 1001

B.10 Signal/Service Translation . . . 1002
B.10.1 Signal/Service Mapping . . . 1002
B.10.1.1 Map Event to Signal . . . 1002
B.10.1.2 Map Field to Signal . . . 1003
B.10.2 Signal/Service Translation Behavior . . . 1004
B.10.2.1 Optional Structure Element in the context of Signal/Service Translation . . . 1004
B.10.2.2 Optional Structure Element in the context of Service/Signal Translation . . . 1004
B.10.2.3 Translation from different Sources . . . 1005
B.10.2.4 Translation to different Targets . . . 1006
B.11 Software Cluster Design . . . 1007
B.11.1 Dependencies among Software Cluster Designs . . . 1007
B.11.2 Usage of SoftwarePackageStep . . . 1008
B.11.2.1 Usage of transfer and process . . . 1008
B.12 Miscellaneous Examples . . . 1010
B.12.1 Relation of ApApplicationError and ApApplicationErrorDomain . . . 1010
C Custom Model Extension 1011
C.1 Overview . . . 1011
C.2 Custom Attribute Definition . . . 1013
C.2.1 Custom Primitive Attribute Definition . . . 1013
C.2.2 Custom Complex Attribute Definition . . . 1015
C.3 Custom Foreign Reference Definition . . . 1017
C.4 Custom Subclass Configuration . . . 1019
C.5 Custom Constraints . . . 1021
C.6 Definition of Reference from SdgClass to SdgClass . . . 1021
D General Modeling 1025
D.1 Reference to a DataPrototype in a PortInterface . . . 1025
D.1.1 Reference to the inside of an `ApplicationDataType` . . . 1025
D.1.2 Reference to the inside of a `CppImplementationDataType` . . . 1027
D.2 Reference to a AutosarDataPrototype in an Executable . . . 1029
D.3 Reference to a Trigger in an Executable . . . 1033
D.4 Reference to a PortPrototype in an Executable . . . 1033
D.5 Modeling of a Method in an Executable . . . 1039
D.6 Modeling of Diagnostic-related InstanceRefs . . . 1041
D.7 Modeling of PHM-related InstanceRefs . . . 1043
D.8 Modeling of Time-related InstanceRefs . . . 1045
D.9 Modeling of Persistency-related InstanceRefs . . . 1045
D.10 Modeling of SoftwareClusterDesign-related InstanceRefs . . . 1046
D.11 Modeling of State-Management-related InstanceRefs . . . 1048
D.12 Modeling of Intrusion-Detection-related InstanceRefs . . . 1051
D.13 Modeling of Crypto-related InstanceRefs . . . 1052

D.14 Firewall InstanceRef 1052
E Mentioned Class Tables 1054
F Change History of AUTOSAR Traceable Items 1139
F.1 Traceable item history of this document according to AUTOSAR Release R17-03 (original version) 1139
F.1.1 Created Constraints in R17-03 1139
F.1.2 Created Specification Items in R17-03 1141
F.2 Traceable item history of this document according to AUTOSAR Release R17-10 1146
F.2.1 Added Specification Items in R17-10 1146
F.2.2 Changed Specification Items in R17-10 1150
F.2.3 Deleted Specification Items in R17-10 1151
F.2.4 Added Constraints in R17-10 1151
F.2.5 Changed Constraints in R17-10 1153
F.2.6 Deleted Constraints in R17-10 1153
F.3 Traceable item history of this document according to AUTOSAR Release R18-03 1154
F.3.1 Added Specification Items in R18-03 1154
F.3.2 Changed Specification Items in R18-03 1157
F.3.3 Deleted Specification Items in R18-03 1159
F.3.4 Added Constraints in R18-03 1159
F.3.5 Changed Constraints in R18-03 1162
F.3.6 Deleted Constraints in R18-03 1162
F.4 Traceable item history of this document according to AUTOSAR Release R18-10 1163
F.4.1 Added Specification Items in R18-10 1163
F.4.2 Changed Specification Items in R18-10 1165
F.4.3 Deleted Specification Items in R18-10 1167
F.4.4 Added Constraints in R18-10 1169
F.4.5 Changed Constraints in R18-10 1171
F.4.6 Deleted Constraints in R18-10 1172
F.5 Traceable item history of this document according to AUTOSAR Release R19-03 1173
F.5.1 Added Specification Items in R19-03 1173
F.5.2 Changed Specification Items in R19-03 1176
F.5.3 Deleted Specification Items in R19-03 1177
F.5.4 Added Constraints in R19-03 1177
F.5.5 Changed Constraints in R19-03 1178
F.5.6 Deleted Constraints in R19-03 1179
F.6 Traceable item history of this document according to AUTOSAR Release R19-11 1180
F.6.1 Added Specification Items in R19-11 1180
F.6.2 Changed Specification Items in R19-11 1183

F.6.3 Deleted Specification Items in R19-11 . . . 1184
F.6.4 Added Constraints in R19-11 . . . 1184
F.6.5 Changed Constraints in R19-11 . . . 1186
F.6.6 Deleted Constraints in R19-11 . . . 1186
F.7 Traceable item history of this document according to AUTOSAR Release R20-11 . . . 1187
F.7.1 Added Specification Items in R20-11 . . . 1187
F.7.2 Changed Specification Items in R20-11 . . . 1190
F.7.3 Deleted Specification Items in R20-11 . . . 1192
F.7.4 Added Constraints in R20-11 . . . 1194
F.7.5 Changed Constraints in R20-11 . . . 1196
F.7.6 Deleted Constraints in R20-11 . . . 1197
F.8 Traceable item history of this document according to AUTOSAR Release R21-11 . . . 1198
F.8.1 Added Specification Items in R21-11 . . . 1198
F.8.2 Changed Specification Items in R21-11 . . . 1200
F.8.3 Deleted Specification Items in R21-11 . . . 1203
F.8.4 Added Constraints in R21-11 . . . 1204
F.8.5 Changed Constraints in R21-11 . . . 1208
F.8.6 Deleted Constraints in R21-11 . . . 1209
F.9 Traceable item history of this document according to AUTOSAR Release R22-11 . . . 1210
F.9.1 Added Specification Items in R22-11 . . . 1210
F.9.2 Changed Specification Items in R22-11 . . . 1213
F.9.3 Deleted Specification Items in R22-11 . . . 1214
F.9.4 Added Constraints in R22-11 . . . 1215
F.9.5 Changed Constraints in R22-11 . . . 1223
F.9.6 Deleted Constraints in R22-11 . . . 1223
F.9.7 Added Advisories in R22-11 . . . 1224
F.9.8 Changed Advisories in R22-11 . . . 1225
F.9.9 Deleted Advisories in R22-11 . . . 1225
F.10 Traceable item history of this document according to AUTOSAR Release R23-11 . . . 1225
F.10.1 Added Specification Items in R23-11 . . . 1225
F.10.2 Changed Specification Items in R23-11 . . . 1227
F.10.3 Deleted Specification Items in R23-11 . . . 1228
F.10.4 Added Constraints in R23-11 . . . 1228
F.10.5 Changed Constraints in R23-11 . . . 1232
F.10.6 Deleted Constraints in R23-11 . . . 1233
F.10.7 Added Advisories in R23-11 . . . 1234
F.10.8 Changed Advisories in R23-11 . . . 1234
F.10.9 Deleted Advisories in R23-11 . . . 1234
F.11 Traceable item history of this document according to AUTOSAR Release R24-11 . . . 1234

F.11.1 Added Specification Items in R24-11 . . . 1234
F.11.2 Changed Specification Items in R24-11 . . . 1236
F.11.3 Deleted Specification Items in R24-11 . . . 1237
F.11.4 Added Constraints in R24-11 . . . 1238
F.11.5 Changed Constraints in R24-11 . . . 1241
F.11.6 Deleted Constraints in R24-11 . . . 1243
F.11.7 Added Advisories in R24-11 . . . 1243
F.11.8 Changed Advisories in R24-11 . . . 1244
F.11.9 Deleted Advisories in R24-11 . . . 1244
F.12 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 1244
F.12.1 Added Specification Items in R25-11 . . . 1244
F.12.2 Changed Specification Items in R25-11 . . . 1246
F.12.3 Deleted Specification Items in R25-11 . . . 1247
F.12.4 Added Constraints in R25-11 . . . 1248
F.12.5 Changed Constraints in R25-11 . . . 1251
F.12.6 Deleted Constraints in R25-11 . . . 1254
F.12.7 Added Advisories in R25-11 . . . 1255
F.12.8 Changed Advisories in R25-11 . . . 1255
F.12.9 Deleted Advisories in R25-11 . . . 1255
F.13 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 1255
F.13.1 Added Specification Items in R26-11 . . . 1255
F.13.2 Changed Specification Items in R26-11 . . . 1256
F.13.3 Deleted Specification Items in R26-11 . . . 1257
F.13.4 Added Constraints in R26-11 . . . 1258
F.13.5 Changed Constraints in R26-11 . . . 1260
F.13.6 Deleted Constraints in R26-11 . . . 1260
F.13.7 Added Advisories in R26-11 . . . 1261
F.13.8 Changed Advisories in R26-11 . . . 1261
F.13.9 Deleted Advisories in R26-11 . . . 1261

G Non-Splitable Elements in the Scope of this Document 1262

H Variation Points in the Scope of this Document 1265

I Upstream Mapping 1266
I.1 Introduction . . . 1266
I.2 CryptoInstantiation . . . 1267
I.3 Persistency . . . 1269

References

[1] Adaptive Platform Machine Configuration
AUTOSAR_AP_TPS_MachineConfiguration

[2] Software Component Template
AUTOSAR_CP_TPS_SoftwareComponentTemplate

[3] Diagnostic Extract Template
AUTOSAR_CP_TPS_DiagnosticExtractTemplate

[4] System Template
AUTOSAR_CP_TPS_SystemTemplate

[5] Layered Software Architecture
AUTOSAR_CP_EXP_LayeredSoftwareArchitecture

[6] Reference Model for Service Oriented Architecture 1.0
https://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

[7] Standardization Template
AUTOSAR_FO_TPS_StandardizationTemplate

[8] Generic Structure Template
AUTOSAR_FO_TPS_GenericStructureTemplate

[9] Explanation of Identity and Access Management
AUTOSAR_AP_EXP_IdentityAndAccessManagement

[10] Explanation of ara::com API
AUTOSAR_AP_EXP_ARAComAPI

[11] SOME/IP Protocol Specification
AUTOSAR_FO_PRS_SOMEIPProtocol

[12] ISO 10646:2012 – Information technology – Universal Coded Character Set (UCS)
https://www.iso.org

[13] Specification of Execution Management
AUTOSAR_AP_SWS_ExecutionManagement

[14] Specification of Platform Health Management
AUTOSAR_AP_SWS_PlatformHealthManagement

[15] Specification of Persistency
AUTOSAR_AP_SWS_Persistency

[16] 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

[17] Specification of Time Synchronization

AUTOSAR_AP_SWS_TimeSynchronization

[18] Specification of Cryptography
AUTOSAR_AP_SWS_Cryptography

[19] IEEE Standard 1722-2016 - IEEE Standard for a Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks

[20] Specification of Raw Data Stream
AUTOSAR_AP_SWS_RawDataStream

[21] ISO 14229-1 – Unified diagnostic services (UDS) – Part 1: Specification and requirements (Release 2006-12)
https://www.iso.org

[22] ISO 17978-3:2025 – Road vehicles — Service-oriented vehicle diagnostics (SOVD) – Part 3: Application programming interface (API)
http://www.iso.org

[23] ISO 13400 – Road vehicles – Diagnostic communication over Internet Protocol (DoIP)
https://www.iso.org

[24] Specification of Diagnostics
AUTOSAR_AP_SWS_Diagnostics

[25] ASAM Service Oriented Vehicle Diagnostics V1.0.0
http://www.asam.net
ASAM_SOVD_BS_V1-0-0_PR.pdf

[26] Multicast DNS
https://www.ietf.org/rfc/rfc6762.txt

[27] DNS-Based Service Discovery
https://www.rfc-editor.org/rfc/rfc6763

[28] mDNS IPv4 Parameters
https://www.iana.org/assignments/multicast-addresses/multicast-addresses.txt

[29] mDNS Port Parameters
https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt

[30] mDNS IPv6Parameters
https://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.txt

[31] REST: Architectural Styles and the Design of Network-based Software Architectures

[32] Specification of Communication Management
AUTOSAR_AP_SWS_CommunicationManagement

[33] Specification of Secure Onboard Communication
AUTOSAR_CP_SWS_SecureOnboardCommunication

[34] Security Extract Template
AUTOSAR_FO_TPS_SecurityExtractTemplate

[35] Technical Report on Operating System Tracing Interface
AUTOSAR_AP_TR_OperatingSystemTracingInterface

[36] Log and Trace Protocol Specification
AUTOSAR_FO_PRS_LogAndTraceProtocol

[37] Specification of Network Management
AUTOSAR_AP_SWS_NetworkManagement

[38] Specification of Update and Configuration Management
AUTOSAR_AP_SWS_UpdateAndConfigurationManagement

[39] Specification of Intrusion Detection System Manager for Adaptive Platform
AUTOSAR_AP_SWS_IntrusionDetectionSystemManager

[40] Procedures for the Management of the Service Name and Transport Protocol Port Number Registry
https://www.rfc-editor.org/rfc/rfc6335.txt

[41] SOME/IP Service Discovery Protocol Specification
AUTOSAR_FO_PRS_SOMEIPServiceDiscoveryProtocol

[42] Data Distribution Service (DDS), Version 1.4
http://www.omg.org/spec/DDS/1.4

[43] RPC over DDS, Version 1.0
https://www.omg.org/spec/DDS-RPC/1.0

[44] DDS Consolidated XML Syntax, Version 1.0
https://www.omg.org/spec/DDS-XML

[45] Specification of SW-C End-to-End Communication Protection Library
AUTOSAR_CP_SWS_E2ELibrary

[46] DDS Security, Version 1.1
https://www.omg.org/spec/DDS-SECURITY/1.1

[47] Glossary
AUTOSAR_FO_TR_Glossary

1 Introduction

This document specifies the so-called *Manifest* on the *AUTOSAR adaptive platform*. The overall modeling approach is described in Chapter 1.1. The definition of the term *service* is given in Chapter 1.2.

In this specification, the term *Manifest* means a formal specification of configuration content. A more detailed description of the term and its implications for the *AUTOSAR adaptive platform* is provided in Chapter 2.

Over time, the manifest-related content in this document will be replaced step by step by a target configuration model based on the APMC[1] approach. Further details about the modeling approach implemented by APMC are provided in [1, TPS Adaptive Platform Machine Configuration].

Despite its name, this document also describes design elements necessary for developing software for the *AUTOSAR adaptive platform*.

The design-related modeling focuses mainly on developing application software for the *AUTOSAR adaptive platform* and on the connection between applications and diagnostics. It is described in detail in Chapter 3, Chapter 4 and Chapter 5.

On the *AUTOSAR adaptive platform*, these chapters complement the [2, TPS Software Component Template] and the [3, TPS Diagnostic Extract Template] on the *AUTOSAR classic platform*. Consequently, certain parts of those documents also apply to the *AUTOSAR adaptive platform*.

This relationship is explicitly marked in the definitions of constraint imposition times in those documents.

Chapter 6 describes Machine Design, which corresponds to some sections of the [4, TPS System Template] on the *AUTOSAR classic platform*.

Chapter 7, in particular, describes the overall view of the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform* communicating through service-oriented communication.

Chapter 8 supports the design-related sections of this document by describing how installable subsystems are represented at the design level.

Chapter 9 concludes the design-related part by explaining how signal-based communication can be transformed into service-oriented communication and vice versa to enable communication between ECUs on the *AUTOSAR classic platform*.

Chapter 10 begins the target-configuration part of the document. It is the target-configuration counterpart to Chapter 3 and describes the content of the so-called *execution manifest*.

Chapter 11 describes the options for configuring a machine by means of a *manifest*.

[1] Adaptive Platform Machine Configuration

Chapter 12 contains subsections that explain the manifest content for platform module functionality.

Chapter 13 describes in detail how service-oriented communication shall be configured at the *manifest* level.

Chapter 14 describes the target-configuration modeling for raw data stream communication.

Chapter 15 explains the modeling of interactions among functional clusters in the AUTOSAR stack.

Chapter 16 describes the concept of an uploadable software package and its configuration.

Finally, Chapter 17 makes some remarks about the interaction of the *AUTOSAR adaptive platform* with software on the *AUTOSAR classic platform*.

1.1 Modeling Approach

1.1.1 General Considerations

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.

  Consequently, even when specific model elements have clear counterparts on the other platform, different terminology could be used to avoid confusing users of the two 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 [2, TPS Software Component Template], as much as possible.

  Consequently, identical meta-classes are used for similar purposes on both platforms. Where applicable, some affected meta-classes are extended in a platform-

specific manner, and constraints clarify the platform-specific use of those 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 `CompositionSwComponentType`s that in turn aggregate `SwComponentPrototype`s typed by e.g. (in the case of the *AUTOSAR adaptive platform*) `AdaptiveApplicationSwComponentType`s.

Reusing existing model elements to define the meta-model for the *AUTOSAR adaptive platform* has the side effect that their descriptions may refer to technical details that apply only to the *AUTOSAR classic platform*.

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

These references shall be interpreted in context. Readers are expected to abstract from those details and identify the aspects of the model elements that are relevant to the *AUTOSAR adaptive platform*.

1.1.2 Modeling of Diagnostic Configuration

The configuration of the “external behavior” of the AUTOSAR diagnostic stack is done by means of the so-called `Diagnostic Extract`. The `Diagnostic Extract` in general extends to both the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform*.

In particular, some parts of the `Diagnostic Extract` apply only for the *AUTOSAR classic platform*, some parts are shared between the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform*, and some parts apply only for the *AUTOSAR adaptive platform*.

The parts that are specific to the *AUTOSAR classic platform* and the parts that apply to both the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform* are documented in the deliverable [3, TPS Diagnostic Extract Template].

For technical reasons, the parts that apply to the *AUTOSAR adaptive platform* cannot be included in the document [3, TPS Diagnostic Extract Template] and are consequently documented in this document, “TPS Manifest Specification”.

The parts about the configuration of Diagnostic Management, as described in this document, fit seamlessly into the same modeling framework that is used on the *AUTOSAR classic platform*. This means that the specific parts for the *AUTOSAR adaptive platform* utilize the shared parts that are described in the [3, TPS Diagnostic Extract Template].

Readers focusing on the `Diagnostic Extract` for application on the *AUTOSAR classic platform* may refer to the [3, TPS Diagnostic Extract Template].

To understand how the `Diagnostic Extract` works on the *AUTOSAR adaptive platform*, readers need to consult the diagnostics-related chapters of the "TPS Manifest Specification" and the relevant parts of the [3, TPS Diagnostic Extract Template].

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 term *service* has the following meanings within AUTOSAR:

- The term *service* is used in the [5, EXP Layered Software Architecture] to denote the highest layer of the AUTOSAR software architecture that interacts with the application. In this context, model elements such as `ServiceSwComponentType`, `SwcServiceDependency`, `ServiceNeeds`, and `PortInterface.isService` have been created on the *AUTOSAR classic platform*.
- The term *service* is used for information related to or required during vehicle servicing. In this context, *service-only diagnostic trouble codes* (DTCs) 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 with the meaning defined by the **service-oriented architecture** [6, SOA]. This meaning is most closely related to the use of the term *service* on the *AUTOSAR adaptive platform*.

1.3 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. `PortPrototype`s. 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 ⌈ character and terminated by the ⌋ 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 “imposition 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 imposition times of constraints are defined and the imposition 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 [7, Standardization Template].

Please note: By intent, TPS documents (and their traceable items) do not trace up to an RS (requirement item).

1.4 About Tables

This document contains a number of tables that describe the applicability or multiplicity of model elements. Empty cells indicate “not applicable”. Model elements corresponding to an empty cell shall not exist in the context described by the respective table.

If the existence or multiplicity of a model element is to be ignored, the table cell contains “d/c” (“don’t care”). If the model element exists, it shall be ignored during downstream processing of the model.

2 Big Picture of Manifest Definition

2.1 Design vs. Target Configuration

2.1.1 Preface

This document underwent a change in terminology. Previously, the term “Deployment” was used to identify modeling or workflow steps that relate to the creation of model elements that are mostly local to a given compute node. These elements are therefore usually created during integration.

In other words, the purpose of all these “deployment-related” steps and modeling elements is to create a configuration of the target on which the code is executed at run-time.

In this context, the usage of the term “deployment” is considered too broad and lacks focus on the real task. In contrast, the usage of the term “target-configuration” is considered the better and more precise alternative. To provide a service to the reader, the text in this document has been updated to reflect the new terminology wherever reasonable.

Model elements, on the other hand, have not been renamed to achieve conformance with the updated terminology. The existing M2-based target-configuration model that is described in this document is to be replaced by an M1-based APMC model anyway, and therefore a costly update of the M2 model is not considered worthwhile.

2.1.2 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 *target configuration* aspect.

Model elements discussed in this document are related either to *design* or to *target configuration*; there is no overlap between the two groups. However, *design* model elements may be required in a concrete model for *target configuration* model elements to make sense.

As mentioned before, some *target configuration* model elements may need one or more references to *design* model elements, and the *design* model elements need to be uploaded to the target platform to complete the semantics of the *target configuration* model.

A *design* model element does not automatically become a *target configuration* model element merely because it is required to be uploaded to a target platform.

2.1.3 Relation between Design and Target Configuration Models

In many cases, the part of the meta-model related to *target configuration* reflects 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 *target configuration* may impact a concrete development project. The following scenarios for the example of *E2E properties* might occur:

Example 2.1

An OEM delivers the description of `AdaptivePlatformServiceInstance`s including the definition of *E2E properties*.

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

Example 2.2

Software exists that has defined *E2E properties* by means of `ComSpec`s. 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 `AdaptivePlatformServiceInstance`s may then have to respect the existing modeling on the software side.

Example 2.3

It could also happen that existing definitions are partly overwritten by experienced engineers.

Some 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 of such a relation is the definition of `ProcessDesign` and `Process`. In these cases, it is easier to draw the line between design and target configuration aspects of the model.

There are cases where the mere existence of a reference into the design model already represents 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 an 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 in which a specific `PhmCheckpoint.checkpointId` is used is identified by means of a reference stereotyped as ≪`atpUriDef`≫.

Because the target of the reference does not necessarily exist 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 12.9).

2.1.4 Structure of the document

The structure of the document maps to the division between *design* and *target configuration* such that the *design* aspect is mostly described in Chapter 3, Chapter 4, Chapter 5, Chapter 7, Chapter 8, and Chapter 9.

In contrast, Chapter 10, Chapter 11, Chapter 12, Chapter 13, Chapter 15, and Chapter 16 focus on *target configuration*-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] Definition of the term `Manifest` ⌈A `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.⌋

It is important to stress that the usage of a `Manifest` is 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] Serialization format of the `Manifest` in AUTOSAR ⌈The standardized serialization format of `Manifest` content in AUTOSAR is ARXML.

Consequently, `Manifest` model content can be validated against the AUTOSAR XML Schema.⌋

An important consequence of [TPS_MANI_01020] is that there is no limitation to just one “manifest file” (also called “the manifest”).

Content may be distributed among several physical files according to the rules given in the specification of the [8, TPS Generic Structure Template].

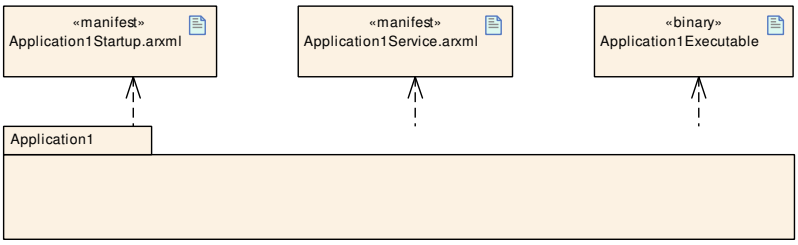


Figure 2.1: Example usage of several manifest files within one software delivery

[TPS_MANI_01021] Serialization format of `Manifest` content on a machine ⌈The 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` shall be **fully preserved**.⌋

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 has to support the full complexity of the `Manifest` meta-model.

The meta-model foresees references from manifest-related meta-classes to design-related meta-classes.

These references are created for clarity, but the referenced content is not required to be resolvable.

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 also be 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*.

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 target configuration aspect would be handled in this context.

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

This aspect is the main motivation for subdividing the definition of the term `Manifest` into five partitions:

Execution Manifest This kind of `Manifest` is used to specify the target configuration-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.

More information on this topic is given in Chapter 10.

Service Instance Manifest This kind of `Manifest` is used to specify how service-oriented 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.

More information on this topic is given in Chapter 13.

Machine Manifest This kind of `Manifest` is intended to describe target configuration-related 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*.

More information on this topic is given in Chapter 11 and Chapter 12.

Raw Data Stream Manifest This kind of `Manifest` describes in Chapter 14 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*.

More information on this topic is given in Chapter 16.

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*.

There are also some extensions of existing modeling used on the *AUTOSAR classic platform* specifically for use on the *AUTOSAR adaptive platform*, e.g. the introduction of new values of the attribute `category`.

In particular, this chapter focuses on the following aspects:

- Definition of application structure, consisting of `SwComponentType` for the *AUTOSAR adaptive platform* (Chapter 3.2.1), `Executable` as the design-level representation of the smallest executable unit (Chapter 3.2.2), as well as the description of how to front-load the configuration of a `Process` by means of `ProcessDesign`, see Chapter 3.2.6.
- Definition of data types specifically for the *AUTOSAR adaptive platform* (Chapter 3.3), including the handling of optional elements in data structures, see Chapter 3.3.5.
- Configuration of transformation properties (Chapter 3.4).
- Description of the design-level IAM configuration by means of the `GrantDesign`, see Chapter 3.5.
- Description of the support for intrusion detection (Chapter 3.6).
- Description of the design of cryptographic solutions (Chapter 3.7).

The definition of a number of relevant `PortInterface`s, among them `ServiceInterface` as the pivotal element for service-oriented communication (Chapter 4.2), is given in a separate top-level chapter, Chapter 4.

3.2 Application Structure

3.2.1 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 Chapter 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*.

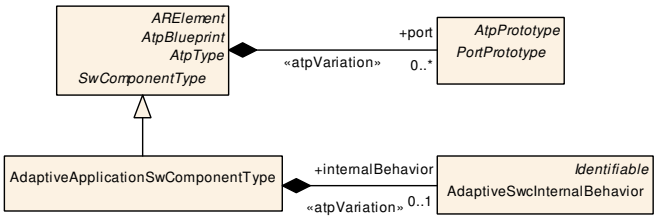


Figure 3.1: Modeling of meta-class `AdaptiveApplicationSwComponentType`

The existence of the `AdaptiveApplicationSwComponentType` provides a convenient means (see [constr_1492]) to lock out most kinds of software components defined for the *AUTOSAR classic platform* from usage on the *AUTOSAR adaptive platform*.

Clarifying 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, has not been clarified so far, and no restriction is in place that prohibits `AdaptiveApplicationSwComponentType` from appearing in the context of a `System`.

Later versions of this specification may add the missing regulation.

Class	**AdaptiveApplicationSwComponentType**			
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.recommendedPackage=AdaptiveApplicationSwComponentTypes This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Aggregated by	ARPackage.element			
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:** atpVariation **Tags:** atp.Splitkey=internalBehavior, internalBehavior.short Name, internalBehavior.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table 3.1: AdaptiveApplicationSwComponentType

Class	**AdaptiveSwcInternalBehavior**
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. This Class is only used by the AUTOSAR Adaptive Platform.



△

Class	AdaptiveSwcInternalBehavior			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Aggregated by	AdaptiveApplicationSwComponentType.internalBehavior			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
service Dependency	SwcService Dependency	*	aggr	This represents the collection of SwcService Dependencys owned by AdaptiveInternalBehavior. **Tags:** atp.Splitkey=serviceDependency, service Dependency.shortName

Table 3.2: AdaptiveSwcInternalBehavior

3.2.2 Executable

3.2.2.1 Overview

An `Executable` represents an executable program in the AUTOSAR meta-model. An explanation of the semantics of the attributes of `Executable` can be found in the following sub-sections.

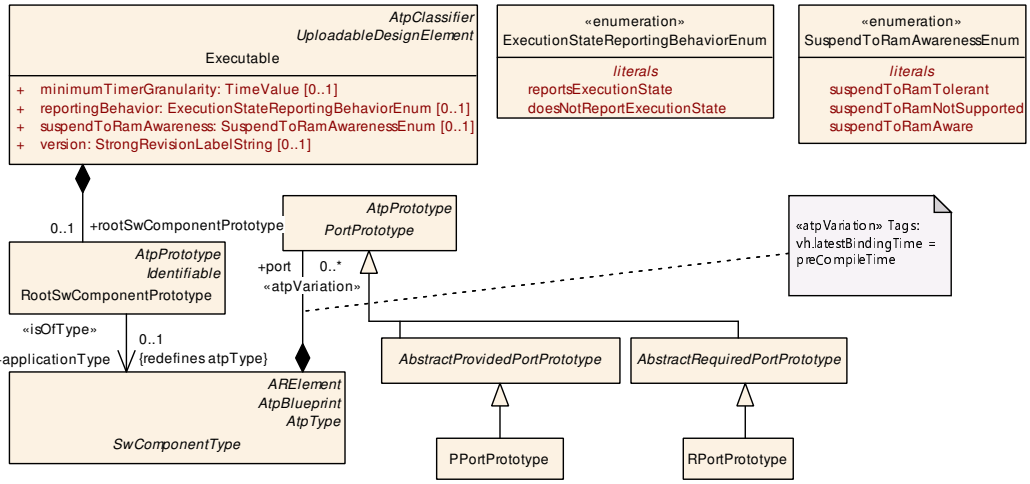


Figure 3.2: Modeling of the `Executable`

Class	Executable			
Note	This meta-class represents an executable program. **Tags:** atp.recommendedPackage=Executables This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, AtpClassifier, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadableDesignElement, UploadablePackageElement			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**Executable**			
implementation Props	Executable ImplementationProps	*	aggr	This aggregation contains the collection of implementation-specific properties necessary to properly build the enclosing Executable. **Tags:** atp.Splitkey=implementationProps
minimumTimer Granularity	TimeValue	0..1	attr	This attribute describes the minimum timer resolution (TimeValue of one tick) that is required by the Executable.
reporting Behavior	ExecutionState ReportingBehavior Enum	0..1	attr	this attribute controls the execution state reporting behavior of the enclosing Executable.
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.Splitkey=rootSwComponentPrototype, rootSw ComponentPrototype.shortName
suspendToRam Awareness	SuspendToRam AwarenessEnum	0..1	attr	This attribute describes the type of awareness of the enclosing Executable to suspend-to-RAM functionality. **Tags:** atp.Status=candidate
version	StrongRevisionLabel String	0..1	attr	Version of the executable.

Table 3.3: Executable

[TPS_MANI_01513] Semantics of `Executable.suspendToRamAwareness`

Status: DRAFT

⌈The awareness for potential suspensions of a running process to RAM is subdivided into three categories:

- The `Executable` is always able to be suspended to RAM without the need of any advanced notification. This case is addressed by the enumeration value `SuspendToRamAwarenessEnum.suspendToRamTolerant` filled into attribute `Executable.suspendToRamAwareness`.

  In this case, a `Process` running the `Executable` will **not** be notified of the `suspend-to-RAM` cycle. The `Process` running the `Executable` will stay in state “Running” throughout the `suspend-to-RAM` cycle.

  No configuration is needed to support this variant of `suspend-to-RAM` awareness.

- The `Executable` is able to respond to a request to be suspended to RAM and it needs to do some preparations until it is ready for `suspend-to-RAM`. This case is addressed by the enumeration value `SuspendToRamAwareness-Enum.suspendToRamAware` filled into attribute `Executable.suspendTo-RamAwareness`.

  This variant of `suspend-to-RAM` awareness requires explicit configuration of the interaction with state management, as explained in Chapter 4.17 and Chapter 12.15.

Specifically, a Process running the Executable will be notified of the suspend-to-RAM cycle and can take actions. The Process running the Executable will stay in state “Running” throughout the suspend-to-RAM cycle.

- The Executable is not supporting suspend-to-RAM at all. This case is addressed by the enumeration value SuspendToRamAwarenessEnum.suspendToRamNotSupported filled into attribute Executable.suspendToRamAwareness.

  No support means that a Process running the Executable needs to be terminated throughout the suspend-to-RAM cycle. This requires a switch to a Function Group state in which the respective Process is not active.

  No additional configuration is needed to support this variant of suspend-to-RAM awareness.

⌋

Enumeration	**SuspendToRamAwarenessEnum**
Note	This enumeration provides values that describe the awareness of an Executable to suspend-to-RAM functionality. **Tags:** atp.Status=candidate This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	Executable.suspendToRamAwareness
Literal	***Description***
suspendToRam Aware	The Executable is notified of suspend-to-RAM activity and can prepare accordingly. **Tags:** atp.EnumerationLiteralIndex=2 atp.Status=candidate
suspendToRamNot Supported	The Executable does not support suspend-to-RAM. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=candidate
suspendToRam Tolerant	The Executable is not notified of suspend-to-RAM activity and seamlessly handles suspend-to-RAM activities. **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=candidate

Table 3.4: SuspendToRamAwarenessEnum

3.2.2.2 Category

[constr_1605] Standardized values of attribute Executable.category

Imposition time: IT_BefAraApiGen

⌈The 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).

⌋

3.2.2.3 Root Software Component Prototype

[TPS_MANI_01010] Root element for a hierarchical software-component ⌈Executable 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.⌋

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*.

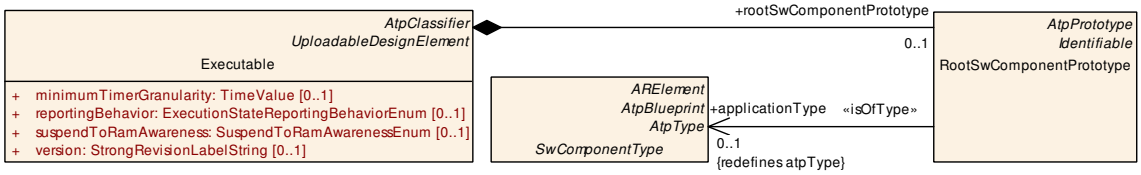


Figure 3.3: Modeling of the RootSwComponentPrototype in the context of Executable

[TPS_MANI_03056] Optionality of Executable.rootSwComponentPrototype ⌈The 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 do not require any further formalization.⌋

[constr_1492] SwComponentType referenced in the role Executable.rootSwComponentPrototype.applicationType

Imposition time: IT_BefAraApiGen

⌈Any 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.⌋

The existence of the reference in the role RootSwComponentPrototype.applicationType is governed by [constr_2633].

The example depicted in Figure 3.4 exemplifies the statement of [constr_1492]. The example shows a component hierarchy that consists of SwComponentPrototypes that

are recursively typed by either a `CompositionSwComponentType` or an `AdaptiveApplicationSwComponentType`.

Class	**RootSwComponentPrototype**			
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.). This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Executable.rootSwComponentPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
applicationType	SwComponentType	0..1	tref	This SwComponentType acts as the Type of the RootSw ComponentPrototype. **Stereotypes:** isOfType **Tags:** atp.Splitkey=applicationType

Table 3.5: RootSwComponentPrototype

Class	***SwComponentType*** (abstract)			
Note	Base class for AUTOSAR software components.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	AdaptiveApplicationSwComponentType, *AtomicSwComponentType*, CompositionSwComponentType, ParameterSwComponentType			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
port	PortPrototype	*	aggr	The `PortPrototype`s through which this `SwComponentType` can communicate. The aggregation of `PortPrototype` is subject to variability with the purpose to support the conditional existence of `PortPrototype`s. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=port, port.shortName, port.variation Point.shortLabel vh.latestBindingTime=preCompileTime
portGroup	PortGroup	*	aggr	A port group being part of this component. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=portGroup, portGroup.shortName, port Group.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
swComponent Documentation	SwComponent Documentation	0..1	aggr	This adds a documentation to the `SwComponentType`. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=swComponentDocumentation, sw ComponentDocumentation.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=-10

Table 3.6: SwComponentType

Class	**CompositionSwComponentType**			
Note	A `CompositionSwComponentType` aggregates `SwComponentPrototype`s (that in turn are typed by `SwComponentType`)s as well as `SwConnector`s for primarily connecting `SwComponentPrototype`s among each others and towards the surface of the `CompositionSwComponentType`. By this means, a hierarchical structures of software-components can be created. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
component	SwComponent Prototype	*	aggr	The instantiated components that are part of this composition. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=component, component.shortName, component.variationPoint.shortLabel vh.latestBindingTime=postBuild
connector	SwConnector	*	aggr	`SwConnector`s have the principal ability to establish a connection among `PortPrototype`s. They can have many roles in the context of a `CompositionSwComponentType`. Details are refined by subclasses. The aggregation of `SwConnector`s is subject to variability with the purpose to support variant data flow. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=connector, connector.shortName, connector.variationPoint.shortLabel vh.latestBindingTime=postBuild
constantValue Mapping	ConstantSpecification MappingSet	*	ref	Reference to the `ConstantSpecificationMapping` to be applied for initValues of `PPortComSpec`s and `RPortComSpec`. **Tags:** atp.Splitkey=constantValueMapping
dataType Mapping	DataTypeMappingSet	*	ref	Reference to the `DataTypeMappingSet` to be applied for the used `ApplicationDataType`s in `ServiceInterface`s. **Tags:** atp.Splitkey=dataTypeMapping
physical Dimension Mapping	PhysicalDimension MappingSet	0..1	ref	This reference identifies the `PhysicalDimensionMappingSet` that is applicable in the context of the enclosing `CompositionSwComponentType`. The `PhysicalDimensionMapping`s contained in the `PhysicalDimensionMappingSet` shall be taken into account for the assessment of the compatibility of `PhysicalDimension`s in the context of creation of a `PortInterfaceMapping` in the scope of the `CompositionSwComponentType`. **Tags:** atp.Splitkey=physicalDimensionMapping

Table 3.7: CompositionSwComponentType

While the left part of Figure 3.4 resembles the modeling in the meta-model, the right part uses a simplified notation to give an idea of how the nested definition of software components might look.

A 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 `PortPrototype`s typed by `ServiceInterface`s. 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*) far less subject to strict separation.

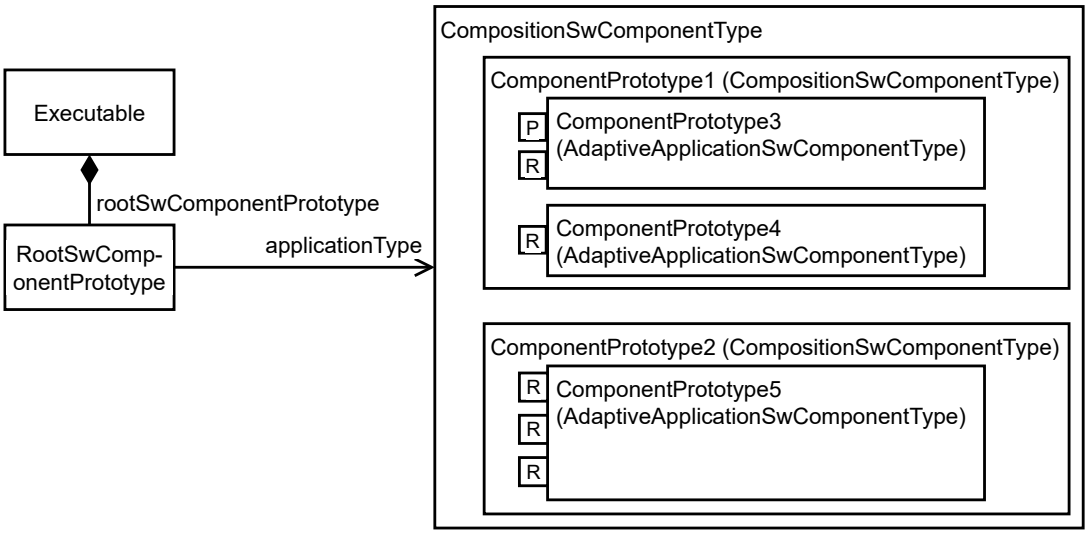


Figure 3.4: Example of the possible structure of an `Executable`

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.

3.2.2.4 Reporting Behavior

[TPS_MANI_01279] Semantics of `Executable.reportingBehavior` ⌈Attribute `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.⌋

Enumeration	**ExecutionStateReportingBehaviorEnum**
Note	This enumeration provides options for controlling of how an Executable reports its execution state to the Execution Management This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	Executable.reportingBehavior

▽

△

Enumeration	**ExecutionStateReportingBehaviorEnum**
Literal	***Description***
doesNotReport ExecutionState	The Executable shall not report its execution state to the Execution Management. **Tags:** atp.EnumerationLiteralIndex=1
reportsExecution State	The Executable shall report its execution state to the Execution Management. **Tags:** atp.EnumerationLiteralIndex=0

Table 3.8: ExecutionStateReportingBehaviorEnum

3.2.2.5 Implementation Props

[TPS_MANI_01370] Semantics of `ExecutableImplementationProps` ⌈If an `Executable` is built for a specific purpose inside the *AUTOSAR adaptive platform*, i.e. the implementation of a functional cluster, then it may be necessary to add configurations that need to be considered at compile time, e.g. whether the implementation of the logging functional cluster needs access to resources provided by the synchronized time base functional cluster.

The configuration of such implementation-specific properties can be done by means of the `Executable.implementationProps`, specifically by a sub-class of `ExecutableImplementationProps`.⌋

Class	***ExecutableImplementationProps*** (abstract)			
Note	This abstract class has the ability to act as a base class for classes that detail out the implementation-specific properties of an Executable. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Describable*			
Subclasses	ExecutableLoggingImplementationProps			
Aggregated by	Executable.implementationProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.9: ExecutableImplementationProps

As depicted in Figure 3.5, one example of the use of `ExecutableImplementationProps` is the definition of `ExecutableLoggingImplementationProps` (see Chapter 3.2.2.6).

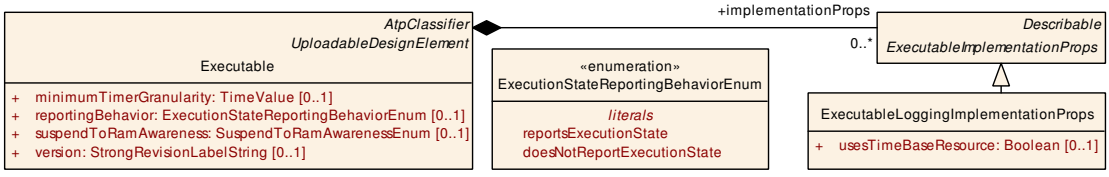


Figure 3.5: Modeling of the `ExecutableImplementationProps`

3.2.2.6 Logging

[TPS_MANI_01383] Semantics of `ExecutableLoggingImplementationProps` ⌈The implementation properties of an `Executable` that is implementing (a part of) the logging functional cluster can be configured using the `ExecutableLoggingImplementationProps`.⌋

[constr_10113] Restriction for the existence of `ExecutableLoggingImplementationProps`

Imposition time: `IT_BefAraApiGen`

⌈The aggregation of `ExecutableLoggingImplementationProps` in the role `Executable.implementationProps` is only allowed for an `Executable` where attribute `category` is set to the value `PLATFORM_LEVEL`.⌋

Class	**ExecutableLoggingImplementationProps**			
Note	This meta-class contains configuration relevant for the implementation of an Executable used in the context of the LogAndTraceInstantiation. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Describable*, *ExecutableImplementationProps*			
Aggregated by	Executable.implementationProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
usesTimeBase Resource	Boolean	0..1	attr	This attribute indicates that the implementation of the enclosing Executable is required to access resources provided by the synchronized time base functional cluster.

Table 3.10: ExecutableLoggingImplementationProps

3.2.3 TraceSwitchConfig

[TPS_MANI_03336] Semantics of `TraceSwitchConfig` ⌈The `TraceSwitchConfig` element maps a `DltMessage` that is referenced in the role `traceMessage` in the context of an `Executable` to a trace switch type defined by the `TraceSwitchConfig.traceSwitch` attribute that in turn decides how the `DltMessage` is routed.⌋

The `TraceSwitchConfig` is used by the compile-time configuration of ARTI trace. It is not required for normal logging. If no `TraceSwitchConfig` exists for a dedicated `DltMessage`, the `DltMessage` is forwarded to the logger.

[TPS_MANI_03337] The definition of the `namespace` of a `TraceSwitchConfig` may follow a hierarchical pattern ⌈The `namespace` of a `TraceSwitchConfig` 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.⌋

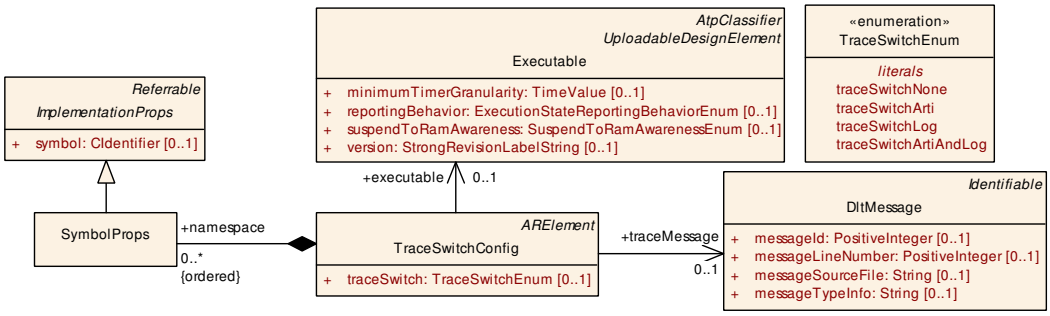


Figure 3.6: Modeling of the `TraceSwitchConfig`

Class	**TraceSwitchConfig**			
Note	This class maps a DltMessage to a trace switch type. **Tags:** atp.recommendedPackage=TraceSwitchConfigs This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
executable	Executable	0..1	ref	Executable that defines the context of the trace switch. **Tags:** atp.Splitkey=executable
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 TraceSwitchConfig.
traceMessage	DltMessage	0..1	ref	Reference to the DltMessage that has to be routed in the trace switch. **Tags:** atp.Splitkey=traceMessage
traceSwitch	TraceSwitchEnum	0..1	attr	Defines how the message is routed, either to • ARTI trace, • Logger (default), • ARTI trace and logger • None

Table 3.11: TraceSwitchConfig

[TPS_MANI_03338] Ordered definition of `TraceSwitchConfig.namespace` ⌈In a hierarchical definition of `TraceSwitchConfig.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.⌋

[constr_9392] Existence of reference `TraceSwitchConfig.traceMessage`

Imposition time: `IT_BefAraApiGen`

⌈The reference in the role `TraceSwitchConfig.traceMessage` shall exist.⌋

[constr_9393] Existence of attribute `TraceSwitchConfig.traceSwitch`

Imposition time: `IT_BefAraApiGen`

⌈The attribute `TraceSwitchConfig.traceSwitch` shall exist.⌋

[constr_9394] Existence of reference `TraceSwitchConfig.executable`

Imposition time: `IT_BefAraApiGen`

⌈The reference in the role `TraceSwitchConfig.executable` shall exist.⌋

Enumeration	**TraceSwitchEnum**
Note	This enumeration provides possible trace switches for DltMessages. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	TraceSwitchConfig.traceSwitch
Literal	***Description***
traceSwitchArti	The message will be routed to ARTI-trace with minimal runtime influence. **Tags:** atp.EnumerationLiteralIndex=1
traceSwitchArtiAndLog	The message will be routed to ARTI-trace and to the logger. **Tags:** atp.EnumerationLiteralIndex=3
traceSwitchLog	The message will be routed to the logger that further processes the message to different sinks (see LogMode) with different verbosities at runtime. **Tags:** atp.EnumerationLiteralIndex=2
traceSwitchNone	The message will not be routed. No runtime resources will be used. **Tags:** atp.EnumerationLiteralIndex=0

Table 3.12: TraceSwitchEnum

3.2.4 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 is needed as an anchor point for adding configuration attributes that make sense in this context but not 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 `PortPrototype`s 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.

3.2.4.1 Service-oriented Communication

The service-oriented communication by means of `PortPrototype`s 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].

[TPS_MANI_01039] Representation of provided service ⌈A **provided service** shall be modeled by means of a `PPortPrototype` that is typed by a `ServiceInterface`.⌋

[constr_1473] No support for PRPortPrototype

Imposition time: IT_BefAraApiGen

⌈A ServiceInterface shall not be referenced by a PRPortPrototype in the role providedRequiredInterface.⌋

[TPS_MANI_01040] Representation of required service ⌈A **required service** shall be modeled by means of an RPortPrototype that is typed by a ServiceInterface.⌋

For more background regarding the rationale of [constr_1473], refer to [2, TPS Software Component Template].

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, means for configuring a given PortPrototype are still required 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 Chapter 3.2.4.2.
- The usage of elements of the corresponding ServiceInterface may need to be configured for a given PortPrototype. This aspect is discussed in Chapter 3.2.5.

3.2.4.2 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 here.

In particular, PortPrototype aggregates the abstract meta-class PortPrototypeProps, which in turn starts an inheritance tree of derived meta-classes that have the ability to qualify sub-classes of PortPrototype accordingly.

[constr_3359] RPortPrototypeProps are related only to RPortPrototypes

Imposition time: IT_BefAraApiGen

⌈The RPortPrototypeProps shall be aggregated only by a RPortPrototype in the role portPrototypeProps.⌋

[constr_3642] Restriction of aggregation of `PortPrototypeProps` to the Adaptive Platform

Imposition time: `IT_BefAraApiGen`

⌈The aggregation of `PortPrototypeProps` is only supported in the context of a `SwComponentType` that is (transitively) referenced by `rootSwComponentPrototype`.⌋

One example for this approach is the definition of the meta-class `RPortPrototypeProps`, sketched in Figure 3.7.

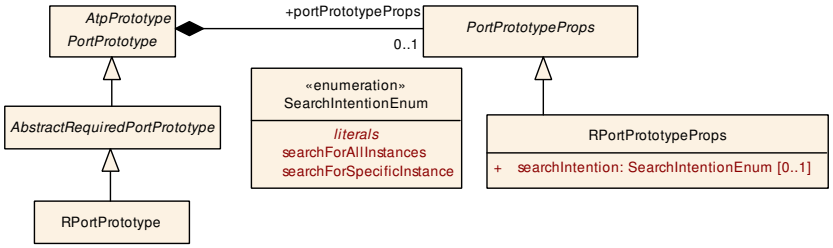


Figure 3.7: Modeling of the `RPortPrototypeProps` for `RPortPrototype`

Class	***PortPrototypeProps*** (abstract)			
Note	This meta-class represents the ability to define a further qualification of semantics of sub-classes of Port Prototype. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Subclasses	RPortPrototypeProps			
Aggregated by	*PortPrototype*.portPrototypeProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.13: PortPrototypeProps

Class	**RPortPrototypeProps**			
Note	PortPrototypeProps for a RPort. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *PortPrototypeProps*			
Aggregated by	*PortPrototype*.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.

Table 3.14: RPortPrototypeProps

[TPS_MANI_01057] Semantics of `RPortPrototypeProps.searchIntention`
⌈The 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.⌋

Enumeration	**SearchIntentionEnum**
Note	This meta-class allows for the definition of a dedicated search intention from the application's point of view. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	RPortPrototypeProps.searchIntention
Literal	***Description***
searchForAll Instances	This value represents the intention to search for all instances of the given service. **Tags:** atp.EnumerationLiteralIndex=0
searchForSpecific Instance	This value represents the intention to search for a specific instance of the given service. **Tags:** atp.EnumerationLiteralIndex=1

Table 3.15: SearchIntentionEnum

3.2.5 Port Prototype ComSpec

[TPS_MANI_01053] Usage of `ComSpec`s on the *AUTOSAR adaptive platform* ⌈The aspect of further qualification of elements of the `ServiceInterface` used to type a given `PortPrototype` is implemented by means of `ComSpec`s, i.e. specific sub-classes of the abstract meta-classes `RPortComSpec` and `PPortComSpec`.

However, the support for `ComSpec`s on the *AUTOSAR adaptive platform* only covers a **limited selection** of attributes of a specific `ComSpec`.⌋

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`.

[TPS_MANI_01327] Value of attribute `EndToEndTransformationComSpecProps.disableEndToEndCheck` vs. value of `EndToEndTransformationComSpecProps.disableEndToEndStateMachine` ⌈If the value of attribute `EndToEndTransformationComSpecProps.disableEndToEndCheck` is set to `true`, then the value of attribute `EndToEndTransformationComSpecProps.disableEndToEndStateMachine` shall be ignored.⌋

Class	***TransformationComSpecProps*** (abstract)			
Note	`TransformationComSpecProps` holds all the attributes for transformers that are port specific.			
Base	*ARObject*, *Describable*			
Subclasses	EndToEndTransformationComSpecProps, UserDefinedTransformationComSpecProps			
Aggregated by	ClientComSpec.transformationComSpecProps, *ReceiverComSpec*.transformationComSpecProps, ServerComSpec.transformationComSpecProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.16: TransformationComSpecProps

Class	**UserDefinedTransformationComSpecProps**			
Note	The `UserDefinedTransformationComSpecProps` is used to specify port specific configuration properties for custom transformers.			
Base	*ARObject*, *Describable*, *TransformationComSpecProps*			
Aggregated by	ClientComSpec.transformationComSpecProps, *ReceiverComSpec*.transformationComSpecProps, ServerComSpec.transformationComSpecProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.17: UserDefinedTransformationComSpecProps

Class	**EndToEndTransformationComSpecProps**			
Note	The class EndToEndTransformationIComSpecProps specifies port specific configuration properties for EndToEnd transformer attributes.			
Base	*ARObject*, *Describable*, *TransformationComSpecProps*			
Aggregated by	ClientComSpec.transformationComSpecProps, *ReceiverComSpec*.transformationComSpecProps, ServerComSpec.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. **Tags:** atp.Splitkey=e2eProfileCompatibilityProps
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.



△

Class	**EndToEndTransformationComSpecProps**			
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.
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 3.18: EndToEndTransformationComSpecProps

3.2.5.1 Port Prototypes typed by Service Interfaces

3.2.5.1.1 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 [9, EXP Identity and Access Management].

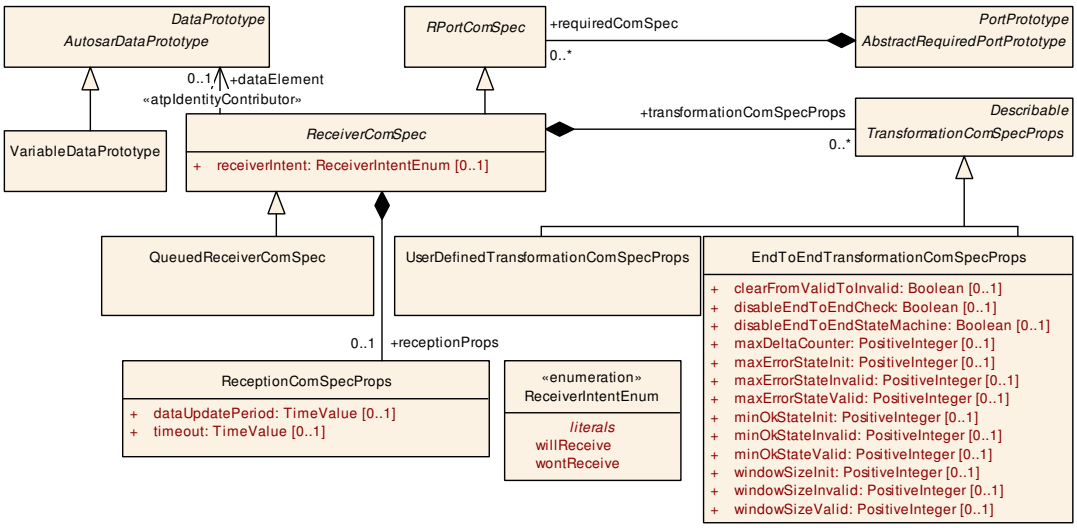


Figure 3.8: Modeling of the `ReceiverComSpec` on the *AUTOSAR adaptive platform*

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.

Class	**ReceiverComSpec** (abstract)			
Note	Receiver-specific communication attributes (RPortPrototype typed by ServiceInterface) that are relevant for events and field notifiers.			
Base	*ARObject*, *RPortComSpec*			
Subclasses	NonqueuedReceiverComSpec, QueuedReceiverComSpec			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dataElement	AutosarDataPrototype	0..1	ref	Data element these attributes belong to. **Stereotypes:** atpIdentityContributor
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. This Attribute is only used by the AUTOSAR Adaptive Platform.
receptionProps	ReceptionComSpec Props	0..1	aggr	"This aggregation represents the definition transmission props in the context of the enclosing ReceiverComSpec. **Tags:** atp.Splitkey=receptionProps
transformation ComSpecProps	TransformationCom SpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation. **Tags:** atp.Splitkey=transformationComSpecProps

Table 3.19: ReceiverComSpec

Class	**QueuedReceiverComSpec**			
Note	Communication attributes specific to queued receiving.			
Base	*ARObject*, *RPortComSpec*, *ReceiverComSpec*			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.20: QueuedReceiverComSpec

[TPS_MANI_01106] Specification of intentions for the receiver of `events` or `field` notifiers

Status: DRAFT

⌈The attribute `ReceiverComSpec.receiverIntent` can be used to specify whether the software actually intends to access the referenced `event`s or `field` notifier or whether it explicitly states that it is not interested in the value.⌋

Enumeration	**ReceiverIntentEnum**
Note	This meta-class represents the intent to specify how a given ServiceInterface is used from the perspective of a given event receiver. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	*ReceiverComSpec*.receiverIntent
Literal	***Description***
willReceive	The receiver will receive the event or field notifier. **Tags:** atp.EnumerationLiteralIndex=0
wontReceive	The receiver won't receive the event or field notifier. **Tags:** atp.EnumerationLiteralIndex=1

Table 3.21: ReceiverIntentEnum

[TPS_MANI_03132] Semantics of E2E attributes in `ReceiverComSpec` ⌈The `EndToEndTransformationComSpecProps` shall be used for the specification of `RPortPrototype`-specific configuration options related to end-to-end protection of events or field notifiers.⌋

3.2.5.1.2 Sender ComSpec

The `SenderComSpec` is modeled in the same way as described in the Software Component Template [2, TPS Software Component Template].

[TPS_MANI_03210] Specification of `event` specific communication attributes ⌈The meta-class `QueuedSenderComSpec` can be used to specify communication attributes that are relevant for an `event` on the sender side.⌋

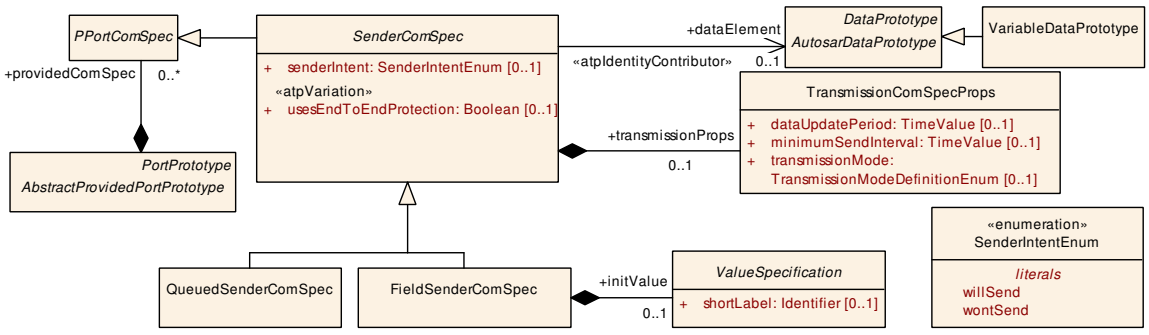


Figure 3.9: Modeling of the `SenderComSpec` on the *AUTOSAR adaptive platform*

[TPS_MANI_03211] Specification of `field` specific communication attributes ⌈The meta-class `FieldSenderComSpec` can be used to specify communication attributes that are relevant for a `field` on the sender side.⌋

Class	***SenderComSpec*** (abstract)			
Note	Communication attributes for a sender port (`PPortPrototype` typed by `ServiceInterface`) that are relevant for `event`s and `field` notifiers.			
Base	*ARObject*, *PPortComSpec*			
Subclasses	FieldSenderComSpec, NonqueuedSenderComSpec, QueuedSenderComSpec			
Aggregated by	*AbstractProvidedPortPrototype*.providedComSpec, PortPrototypeBlueprint.providedComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	AutosarDataPrototype	0..1	ref	Data element these quality of service attributes apply to. **Stereotypes:** atpIdentityContributor
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. This Attribute is only used by the AUTOSAR Adaptive Platform.
transmissionProps	TransmissionComSpecProps	0..1	aggr	This aggregation represents the definition transmission props in the context of the enclosing SenderComSpec. **Tags:** atp.Splitkey=transmissionProps
usesEndToEndProtection	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 3.22: SenderComSpec

Class	**QueuedSenderComSpec**			
Note	Communication attributes specific to distribution of events (`PPortPrototype`, `SenderReceiverInterface` and `dataElement` carries an "event").			
Base	*ARObject*, *PPortComSpec*, *SenderComSpec*			
Aggregated by	*AbstractProvidedPortPrototype*.providedComSpec, PortPrototypeBlueprint.providedComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.23: QueuedSenderComSpec

Class	**FieldSenderComSpec**			
Note	Port specific communication attributes for a Field that is defined in a ServiceInterface. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *PPortComSpec*, *SenderComSpec*			
Aggregated by	*AbstractProvidedPortPrototype*.providedComSpec, PortPrototypeBlueprint.providedComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initValue	ValueSpecification	0..1	aggr	Initial value for a Field that is set before the Service Interface is offered. **Tags:** atp.Splitkey=initValue

Table 3.24: FieldSenderComSpec

[constr_10152] Multiplicity of reference in the role `FieldSenderComSpec.initValue`

Imposition time: `IT_BefAraApiGen`

⌈For each `FieldSenderComSpec`, the reference in the role `initValue` shall exist.⌋

[TPS_MANI_03212] Specification of initial value for a `field` ⌈The attribute `FieldSenderComSpec.initValue` can be used to specify an initial Value for a `field`.⌋

A `field` has a valid value at any time as described in Chapter 4.2.4. The functional cluster 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 the [10, EXP ARA COM API] 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] Specification of intentions for the sender of `event`s or `field` notifiers

Status: DRAFT

⌈The attribute `SenderComSpec.senderIntent` can be used to specify whether the software actually intends to send the referenced `event`s or `field` notifier.⌋

Enumeration	**SenderIntentEnum**
Note	This meta-class represents the intent to specify how a given ServiceInterface is used from the perspective of a given event sender. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	*SenderComSpec*.senderIntent
Literal	***Description***
willSend	The sender will send the event or field notifier. **Tags:** atp.EnumerationLiteralIndex=0
wontSend	The sender won't send the event or field notifier. **Tags:** atp.EnumerationLiteralIndex=1

Table 3.25: SenderIntentEnum

3.2.5.1.3 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.

[TPS_MANI_01108] Specification of intentions for the caller of a `method`s or `field` setter/getter

Status: DRAFT

⌈The attribute `ClientComSpec.clientIntent` can be used to specify whether the software actually intends to call the referenced `method`s or getter/setter of a referenced `field`.⌋

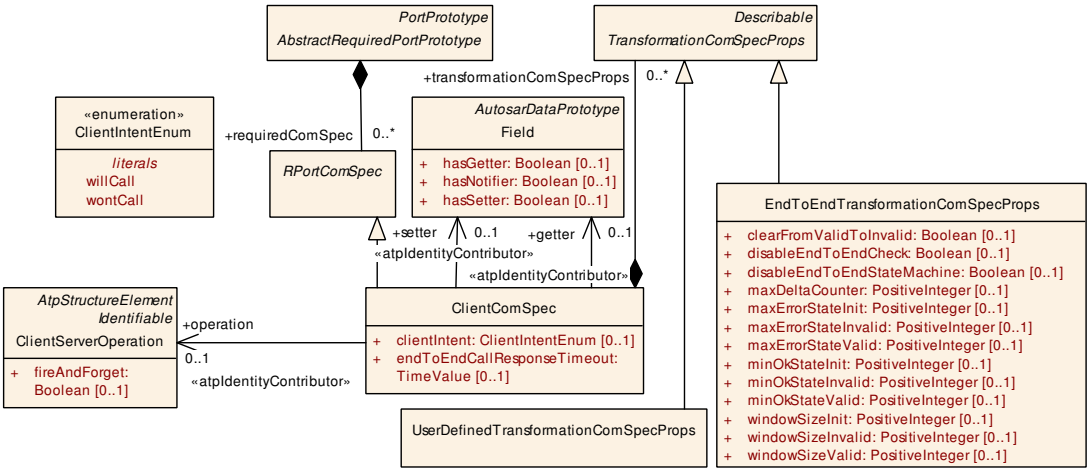


Figure 3.10: Modeling of the `ClientComSpec` on the *AUTOSAR adaptive platform*

[TPS_MANI_01324] Semantics of E2E attributes in `ClientComSpec` ⌈The `EndToEndTransformationComSpecProps` shall be used for the specification of `RPortPrototype`-specific configuration options related to end-to-end protection of `method`s.⌋

Class	**ClientComSpec**			
Note	Client-specific communication attributes (`RPortPrototype` typed by `ServiceInterface`) that are relevant for `method` and `field` getters and setters.			
Base	*ARObject*, *RPortComSpec*			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
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. This Attribute is only used by the AUTOSAR Adaptive Platform.
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).

△

Class	**ClientComSpec**			
getter	Field	0..1	ref	The existence of this reference indicates that the Client ComSpec refers to the getter of a Field. **Stereotypes:** atpIdentityContributor This Attribute is only used by the AUTOSAR Adaptive Platform.
operation	ClientServerOperation	0..1	ref	This represents the corresponding ClientServerOperation. **Stereotypes:** atpIdentityContributor
setter	Field	0..1	ref	The existence of this reference indicates that the Client ComSpec refers to the setter of a Field. **Stereotypes:** atpIdentityContributor This Attribute is only used by the AUTOSAR Adaptive Platform.
transformation ComSpecProps	TransformationCom SpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation. **Tags:** atp.Splitkey=transformationComSpecProps

Table 3.26: ClientComSpec

Enumeration	**ClientIntentEnum**
Note	This meta-class represents the intent to specify how a given ServiceInterface is used from the perspective of a given client. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	ClientComSpec.clientIntent
Literal	***Description***
willCall	The client will call this method. **Tags:** atp.EnumerationLiteralIndex=0
wontCall	The client won't call this method. **Tags:** atp.EnumerationLiteralIndex=1

Table 3.27: ClientIntentEnum

3.2.5.1.4 Server ComSpec

The `ServerComSpec` 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.

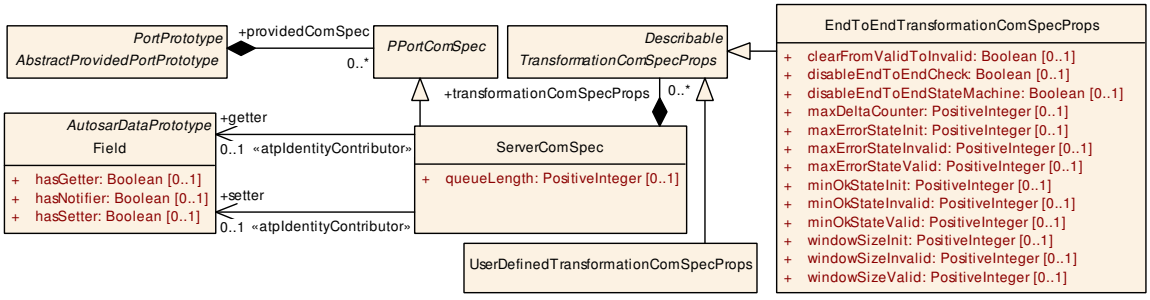


Figure 3.11: Modeling of the `ServerComSpec` on the *AUTOSAR adaptive platform*

[TPS_MANI_01325] Semantics of E2E attributes in `ServerComSpec` ⌈The `EndToEndTransformationComSpecProps` shall be used for the specification of `PPortPrototype`-specific configuration options related to end-to-end protection of `method`s.⌋

Class	**ServerComSpec**			
Note	Server-specific communication attributes (`PPortPrototype` typed by `ServiceInterface`) that are relevant for `method`s and `field` getters and setters.			
Base	*ARObject*, *PPortComSpec*			
Aggregated by	*AbstractProvidedPortPrototype*.providedComSpec, PortPrototypeBlueprint.providedComSpec			
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. **Stereotypes:** atpIdentityContributor This Attribute is only used by the AUTOSAR Adaptive Platform.
operation	ClientServerOperation	0..1	ref	Operation these communication attributes apply to. **Stereotypes:** atpIdentityContributor
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. **Stereotypes:** atpIdentityContributor This Attribute is only used by the AUTOSAR Adaptive Platform.
transformation ComSpecProps	TransformationCom SpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation. **Tags:** atp.Splitkey=transformationComSpecProps

Table 3.28: ServerComSpec

3.2.5.1.5 Communication behavior to be implemented by the Application Software

3.2.5.1.5.1 Transmission ComSpec Props

To support periodic data transmission and reception (the main reason is to be able to perform periodic invocation of the EndToEnd transformation), the application software needs to implement the period in its program flow. Furthermore, there may 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 used 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.

3.2.5.1.5.2 Transmission triggered by Element of composite Data Type

If the SenderComSpec.dataElement is typed by a composite data type, then the "on change" transmissions (TransmissionModeDefinitionEnum.onChange and TransmissionModeDefinitionEnum.cyclicAndOnChange) AUTOSAR provides the opportunity to formally define which parts of the dataElement shall actually trigger the "on change" transmission.

[constr_10535] Restriction for the attribute SenderComSpec.transmissionProps.onChangeDataPrototype

Status: DRAFT
Imposition time: IT_BefAraApiGen

⌈If the SenderComSpec.dataElement refers to an AutosarDataPrototype aggregated by a ServiceInterface in the role event or field, then a DataPrototypeReference aggregated in the role SenderComSpec.transmissionProps.onChangeDataPrototype shall only exist as

- a DataPrototypeInPortInterfaceRef that aggregates a DataPrototypeInServiceInterfaceInstanceRef in the role dataPrototypeInServiceInterface or
- an ImplementationDataTypeElementInPortInterfaceRef.

⌋

The reference to the "root" AutosarDataPrototype is already specified by the role SenderComSpec.dataElement. Therefore, the reference in the role Data-

PrototypeInServiceInterfaceInstanceRef.rootDataPrototype shall not be defined again to avoid contradictory identification of the “root”.

This is the rationale for [constr_10536] and [constr_10537], the latter covering the possible case of ImplementationDataTypeElementInPortInterfaceRef.rootDataPrototype.

[constr_10536] Existence of TransmissionComSpecProps.onChangeDataPrototype.dataPrototypeInServiceInterface.rootDataPrototype

Status: DRAFT

Imposition time: IT_BefAraApiGen

⌈If all of the following conditions apply:

- the SenderComSpec.dataElement refers to an AutosarDataPrototype aggregated by a ServiceInterface in either the role event or field and
- the aggregation in the role TransmissionComSpecProps.onChangeDataPrototype.dataPrototypeInServiceInterface exists,

then the aggregation in the role TransmissionComSpecProps.onChangeDataPrototype.dataPrototypeInServiceInterface.rootDataPrototype shall **not** exist.⌋

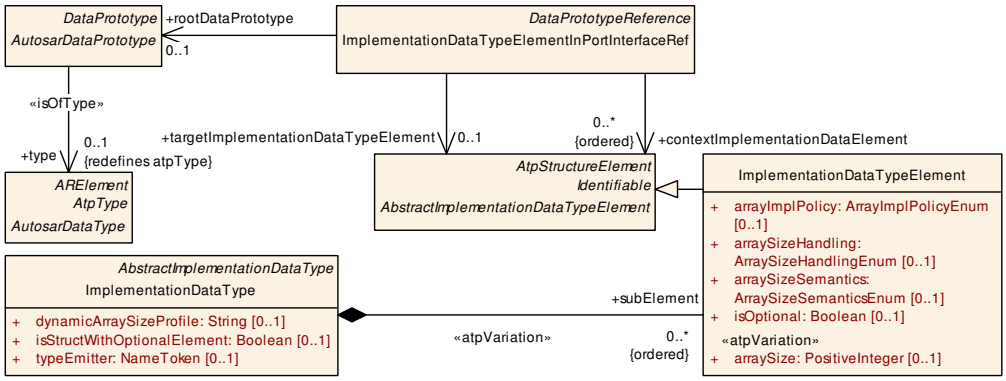


Figure 3.12: Modeling of ImplementationDataTypeElementInPortInterfaceRef

[constr_10537] Existence of TransmissionComSpecProps.onChangeDataPrototype.rootDataPrototype

Imposition time: IT_BefAraApiGen

⌈If all of the following conditions apply:

- the SenderComSpec.dataElement refers to an AutosarDataPrototype aggregated by a ServiceInterface in either the role event or field and
- the aggregation in the role TransmissionComSpecProps.onChangeDataPrototype exists,

then the aggregation in the role `TransmissionComSpecProps.onChangeDataPrototype.rootDataPrototype` shall **not** exist.⌋

Class	**DataPrototypeInPortInterfaceRef**			
Note	This class represents a RootDataPrototype that is typed by an ApplicationDataType or Implementation DataType or a DataTypeElement that is aggregated within a composite application data type (record or array).			
Base	*ARObject*, *DataPrototypeReference*			
Aggregated by	DataPrototypeTransformationProps.dataPrototypeInPortInterfaceRef, SignalServiceTranslationElement Props.element, TransmissionComSpecProps.onChangeDataPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataPrototypeIn ServiceInterface	DataPrototype	0..1	iref	This element defines a reference to a DataPrototype in the context of a ServiceInterface. **Tags:** atp.Splitkey=dataPrototypeInServiceInterface, data PrototypeInServiceInterface.rootDataPrototype, data PrototypeInServiceInterface.contextDataPrototype, data PrototypeInServiceInterface.targetDataPrototype atp.Status=draft **InstanceRef implemented by:** DataPrototypeInService InterfaceInstanceRef This Attribute is only used by the AUTOSAR Adaptive Platform.

Table 3.29: DataPrototypeInPortInterfaceRef

Class	**ImplementationDataTypeElementInPortInterfaceRef**			
Note	This meta-class represents the ability to refer to the internal structure of an AutosarDataPrototype which is typed by an ImplementationDatatype in the context of a PortInterface. In other words, this meta-class shall not be used to model a reference to the AutosarDataPrototype as a target itself, even if the AutosarDataPrototype is typed by an ImplementationDataType and even if that ImplementationDataType represents a composite data type.			
Base	*ARObject*, *DataPrototypeReference*			
Aggregated by	DataPrototypeTransformationProps.dataPrototypeInPortInterfaceRef, SignalServiceTranslationElement Props.element, TransmissionComSpecProps.onChangeDataPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
context Implementation DataElement (ordered)	AbstractImplementation DataTypeElement	*	ref	This is a context in case there are subelements with explicit types. The reference has to be ordered to properly reflect the nested structure. **Tags:** xml.sequenceOffset=20
rootData Prototype	AutosarDataPrototype	0..1	ref	This refers to the AutosarDataPrototype which is typed by the ImplementationDatatype. The targetDataPrototype and all defined contextDataPrototypes can be found within this rootDataPrototype. **Tags:** atp.Splitkey=rootDataPrototype xml.sequenceOffset=10
target Implementation DataType Element	AbstractImplementation DataTypeElement	0..1	ref	This is a target ImplementationDataTypeElement in case that the rootDataPrototype is composite and the target is a subElement of the rootDataPrototype. **Tags:** atp.Splitkey=targetImplementationDataTypeElement xml.sequenceOffset=30

Table 3.30: ImplementationDataTypeElementInPortInterfaceRef

3.2.5.2 Port Prototypes typed by Persistency Data Interfaces

[TPS_MANI_01314] Further qualification of properties of RPortPrototypes typed by PersistencyKeyValueStorageInterfaces ⌈For RPortPrototypes typed by PersistencyKeyValueStorageInterfaces it is possible to define further qualifying attributes for the required side.

For this purpose meta-class PersistencyDataRequiredComSpec is provided.⌋

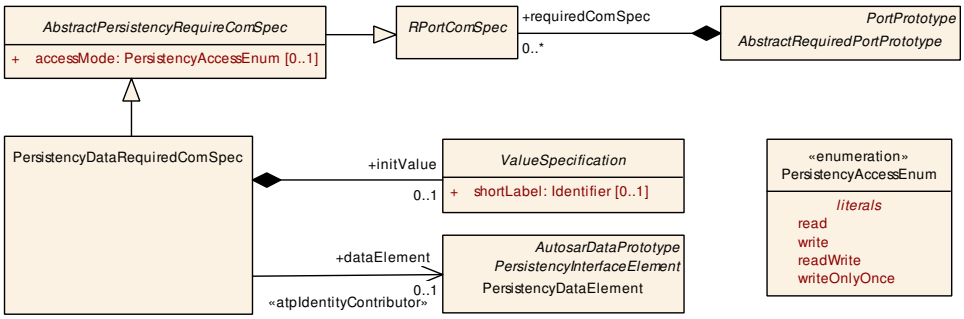


Figure 3.13: Modeling of ComSpec for persistency

[TPS_MANI_01160] Definition of initial value for PersistencyDataElement ⌈The definition of an initial value for a PersistencyDataElement can be done on the level of a PortPrototype by means of PersistencyDataRequiredComSpec.initValue⌋

Class	***AbstractPersistencyRequireComSpec*** (abstract)			
Note	This meta-class acts as a bas-class of persistency-related ComSpec classes. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *RPortComSpec*			
Subclasses	PersistencyDataRequiredComSpec, PersistencyFileRequiredComSpec			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
accessMode	PersistencyAccessEnum	0..1	attr	This attribute controls how persistent data can be accessed.

Table 3.31: AbstractPersistencyRequireComSpec

Class	**PersistencyDataRequiredComSpec**			
Note	This meta-class represents the ability to define port-specific attributes for supporting use cases of data persistency on the required side. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractPersistencyRequireComSpec*, *RPortComSpec*			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	PersistencyDataElement	0..1	ref	This refrence represents the PersistencyDataElement for which the PersistencyDataRequiredComSpec applies. **Stereotypes:** atpIdentityContributor

▽

△

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.Splitkey=initValue

Table 3.32: PersistencyDataRequiredComSpec

[constr_10654] Existence of `PersistencyFileRequiredComSpec`

Imposition time: `IT_BefAraApiGen`

⌈In the context of any given `RPortPrototype` typed by a `PersistencyFileStorageInterface`, **at most** one `PersistencyFileRequiredComSpec` may exist.⌋

[constr_10081] Existence of initial values in the definition of `PersistencyDataRequiredComSpec`

Imposition time: `IT_BefAraApiGen`

⌈For each `PersistencyDataRequiredComSpec`, if the value of attribute `dataElement.updateStrategy` is set to the value `delete`, then attribute `PersistencyDataRequiredComSpec.initValue` shall not exist.⌋

[TPS_MANI_01456] Definition of `PersistencyDataRequiredComSpec.accessMode` ⌈The value of attribute `PersistencyDataRequiredComSpec.accessMode` applies for the entire `RPortPrototype`, as opposed to single `dataElement`s.

To indicate this specific semantics, the `accessMode` needs to be specified in a `PersistencyDataRequiredComSpec` that, at the same time, does **not** refer to a `dataElement`.⌋

As a consequence of [TPS_MANI_01456], [constr_10576] is defined.

[constr_10576] Existence of primitive attribute `PersistencyDataRequiredComSpec.accessMode`

Imposition time: `IT_BefAraApiGen`

⌈Each `PersistencyDataRequiredComSpec` that defines the reference in the role `dataElement` shall **not** define the attribute `accessMode` and vice versa.⌋

3.2.5.2.1 Port Prototypes typed by Persistency File Interfaces

[TPS_MANI_01457] Further qualification of properties of `RPortPrototype`s typed by `PersistencyFileStorageInterface`s ⌈For `RPortPrototype`s typed by `PersistencyFileStorageInterface`s it is possible to define the access mode (modeled as attribute `accessMode`) for the required side.

For this purpose meta-class `PersistencyFileRequiredComSpec` is provided.⌋

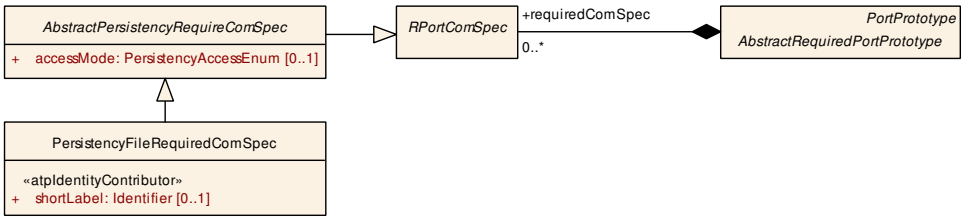


Figure 3.14: Modeling of `ComSpec` for file persistency

Class	**PersistencyFileRequiredComSpec**			
Note	This meta-class represents the ability to define port-specific attributes for supporting use cases of file persistency on the required side. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractPersistencyRequireComSpec*, *RPortComSpec*			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
shortLabel	Identifier	0..1	attr	The shortLabel does not carry a semantics and is only needed to support the merging of partial models. **Stereotypes:** atpIdentityContributor

Table 3.33: PersistencyFileRequiredComSpec

Enumeration	**PersistencyAccessEnum**
Note	This enumeration provides standardized ways of how persistent data can be accessed.
Aggregated by	*AbstractPersistencyRequireComSpec*.accessMode
Literal	***Description***
read	Access to persistent data is read-only. **Tags:** atp.EnumerationLiteralIndex=0 This EnumerationLiteral is only used by the AUTOSAR Adaptive Platform.
readWrite	Persistent data can both be written and read. **Tags:** atp.EnumerationLiteralIndex=2 This EnumerationLiteral is only used by the AUTOSAR Adaptive Platform.
write	Access to persistent data is write-only. **Tags:** atp.EnumerationLiteralIndex=1 This EnumerationLiteral is only used by the AUTOSAR Adaptive Platform.
writeOnlyOnce	The elements of this storage are prevented from being changed after first initialization. **Tags:** atp.EnumerationLiteralIndex=3 This EnumerationLiteral is only used by the AUTOSAR Adaptive Platform.

Table 3.34: PersistencyAccessEnum

3.2.6 Process Design

Within the definition of e.g. a diagnostic mapping, the assignment to the Process is typically done in a methodological step[1] that happens when all the diagnostic mappings[2] are already complete.

Therefore, a proxy for an actual Process is required 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] Semantics of meta-class ProcessDesign ⌈Meta-class 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 target-configuration specification.⌋

The integrator needs to ensure that an actual Process refers to the corresponding ProcessDesign so 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] Reference from Process to ProcessDesign

Imposition time: IT_Mani

⌈Each ProcessDesign shall only be referenced from a single Process.⌋

The reference from the Process to the ProcessDesign reflects that the Process is typically created later in time[3].

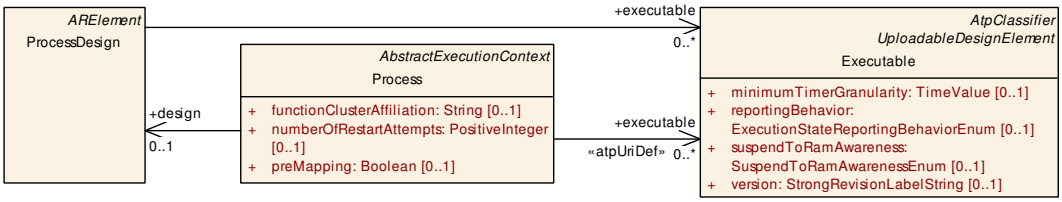


Figure 3.15: Modeling of the ProcessDesign

The association of diagnostic mappings with ProcessDesign may still happen as a final step of the activity to create the diagnostic mappings.

For more information on the semantics of this stereotype, refer to the specification of the AUTOSAR Generic Structure Template [8, TPS Generic Structure Template].

[1] i.e. during the creation of the execution manifest

[2] From the methodological point of view, the creation of the diagnostic mapping is typically considered a design-time activity.

[3] In other words, if references are needed between design-related and target-configuration-related meta-classes then the direction of these references shall always point from target-configuration to design.

Class	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.recommendedPackage=ProcessDesigns This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
executable	Executable	*	ref	Reference to executable that is executed in the process. **Tags:** atp.Splitkey=executable

Table 3.35: ProcessDesign

[constr_3731] Upper multiplicity of reference in the role `ProcessDesign.executable`

Imposition time: `IT_SubSysDes`

⌈In the context of `ProcessDesign`, the reference in the role `executable` shall exist at most once.⌋

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 [2, TPS Software Component Template].

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 use 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 be used directly 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)			
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 ApplicationDataTypes only.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*ApplicationCompositeDataType*, ApplicationPrimitiveDataType			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.36: 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 refer to Chapter 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.

The meta-classes used to define an ApplicationPrimitiveDataType of category STRING are summarized in Figure 3.16.

Thanks to the usage of programming languages with richer data types than plain C, the implementation of an ApplicationPrimitiveDataType of category STRING on the *AUTOSAR adaptive platform* is predefined for a given *language binding*.

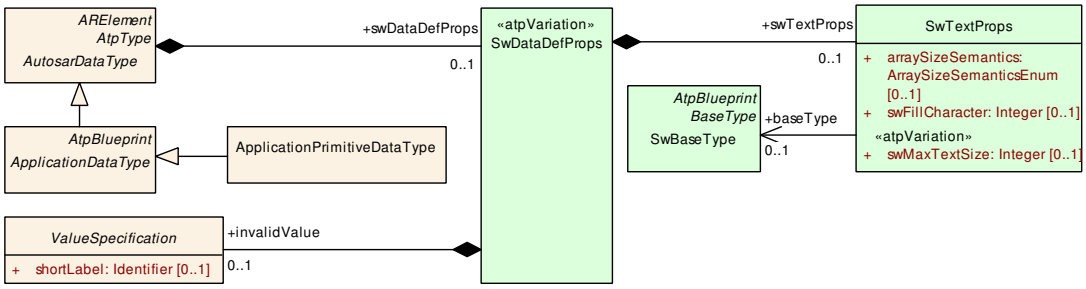


Figure 3.16: Specification of textual strings

[TPS_MANI_01047] Existence of a `SwRecordLayout` for an `ApplicationPrimitiveDataType` of `category STRING` ⌈For the usage of an `ApplicationPrimitiveDataType` of `category STRING` on the *AUTOSAR adaptive platform*, the existence of `ApplicationPrimitiveDataType.swDataDefProps.swRecordLayout` shall be ignored.⌋

[TPS_MANI_01047] intentionally does not forbid the existence of `SwRecordLayout` because the same `ApplicationPrimitiveDataType` of `category STRING` could rightfully be used **on both** the *AUTOSAR adaptive platform* and the *AUTOSAR classic platform*.

Class	**ApplicationPrimitiveDataType**			
Note	A primitive data type defines a set of allowed values. **Tags:** atp.recommendedPackage=ApplicationDataTypes			
Base	*ARElement*, *ARObject*, *ApplicationDataType*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.37: ApplicationPrimitiveDataType

Class	**SwTextProps**			
Note	This meta-class expresses particular properties applicable to strings in variables or calibration parameters.			
Base	*ARObject*			
Aggregated by	SwDataDefProps.swTextProps			
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 `ImplementationDataType`. 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.

△

Class	**SwTextProps**			
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 `ApplicationDataType`. **Tags:** atp.Splitkey=baseType xml.sequenceOffset=30
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 `arraySizeSemantics`. **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 3.38: SwTextProps

Enumeration	**ArraySizeSemanticsEnum**
Note	This enumeration provides options for how the number of elements in an array object shall be interpreted.
Aggregated by	ApplicationArrayElement.arraySizeSemantics, DiagnosticDataElement.arraySizeSemantics, ImplementationDataTypeElement.arraySizeSemantics, SwTextProps.arraySizeSemantics
Literal	***Description***
fixedSize	This means that the array object will always have a fixed number of elements. **Tags:** atp.EnumerationLiteralIndex=0
variableSize	This implies that the actual number of elements in the array object might vary at run-time. The configured size represents the maximum number of elements in the array object. **Tags:** atp.EnumerationLiteralIndex=1

Table 3.39: ArraySizeSemanticsEnum

Class	**SwBaseType**			
Note	This meta-class represents a base type used within ECU software. **Tags:** atp.recommendedPackage=BaseTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *BaseType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.40: SwBaseType

3.3.2.2 Associative Map Data Type

[TPS_MANI_01027] Semantics of `ApplicationAssocMapDataType` ⌈An `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`).⌋

Figure 3.17 depicts an example of the structure of an `ApplicationAssocMapDataType`.

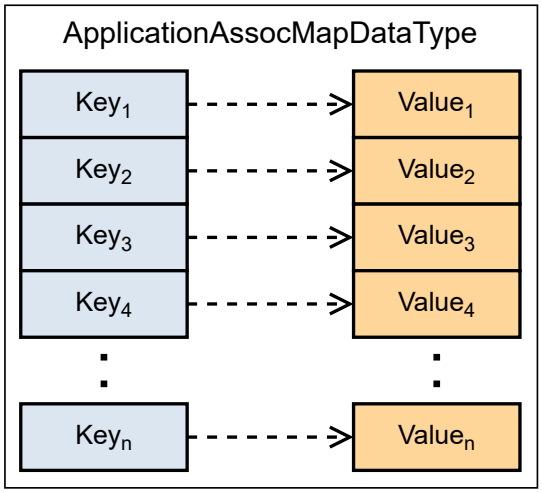


Figure 3.17: Example `ApplicationAssocMapDataType` on the *AUTOSAR adaptive platform*

[constr_3349] Usage of `ApplicationAssocMapDataType` is limited

Imposition time: `IT_BefAraApiGen`

⌈The usage of an `ApplicationAssocMapDataType` is limited to the context of `AdaptiveApplicationSwComponentType`s and `CompositionSwComponentType`s defined in the context of an `Executable`, i.e. such a data type shall not be used on the *AUTOSAR classic platform*.⌋

[constr_3349] is a formal approach to express that an `ApplicationAssocMapDataType` shall only be used on the *AUTOSAR adaptive platform*.

[TPS_MANI_01016] Category of `ApplicationAssocMapDataType` ⌈The value `ApplicationAssocMapDataType.category` shall be set to `ASSOCIATIVE_MAP`.⌋

As can be deduced from looking at Figure 3.17, the concept of an ApplicationDataType of category MAP shall not be confused with an ApplicationAssocMapDataType[4].

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.

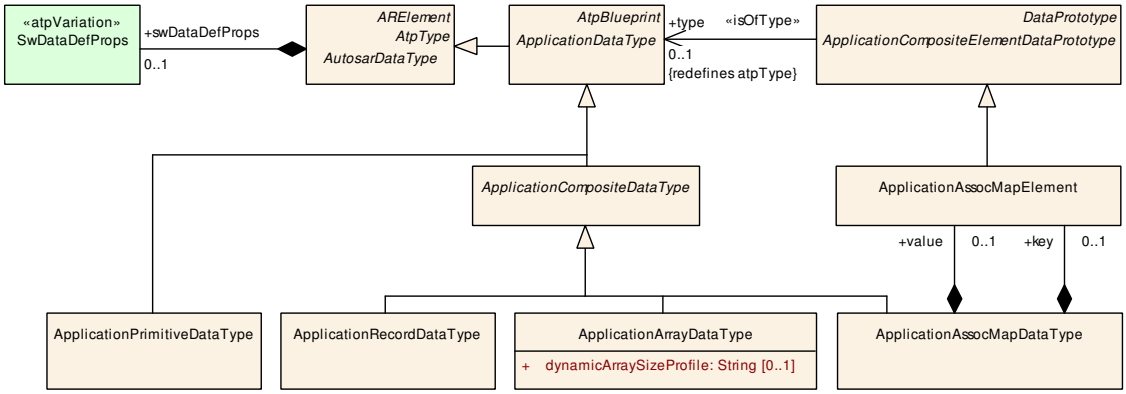


Figure 3.18: Formal model of ApplicationAssocMapDataType

On the other hand, it is not a problem at all if content on the value side contains duplicates, e.g. two unique keys are associated with values that have 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 by the fact that data types for both key and value need to be defined.

There is no assumption on how the structure of an implementation of an associative map may look. 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 use the container ara::core::Map (where the implementation is opaque to the client programmer).

Class	**ApplicationAssocMapDataType**
Note	An application data type which is a map and consists of a key and a value **Tags:** atp.recommendedPackage=ApplicationDataTypes This Class is only used by the AUTOSAR Adaptive Platform.

▽

[4]On the other hand, both concepts of a “map” are justified in their respective “community” and choosing to name one of these differently in order to reduce overall potential confusion would probably not be appropriate

△

Class	ApplicationAssocMapDataType			
Base	ARElement, ARObject, ApplicationCompositeDataType, ApplicationDataType, AtpBlueprint, Atp Classifier, AtpType, AutosarDataType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
key	ApplicationAssocMap Element	0..1	aggr	Key element of the map that is used to uniquely identify the value of the map. **Tags:** atp.Splitkey=key, key.shortName
value	ApplicationAssocMap Element	0..1	aggr	Value element of the map that stores the content associated to a key. **Tags:** atp.Splitkey=value, value.shortName

Table 3.41: ApplicationAssocMapDataType

[constr_10124] Multiplicity of attribute ApplicationAssocMapDataType.key

Imposition time: IT_BefAraApiGen

⌈For each ApplicationAssocMapDataType, the attribute key shall exist.⌋

[constr_10125] Multiplicity of attribute ApplicationAssocMapDataType.value

Imposition time: IT_BefAraApiGen

⌈For each ApplicationAssocMapDataType, the attribute value shall exist.⌋

Class	ApplicationAssocMapElement			
Note	Describes the properties of the elements of an application map data type. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARObject, ApplicationCompositeElementDataPrototype, AtpFeature, AtpPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable			
Aggregated by	ApplicationAssocMapDataType.key, ApplicationAssocMapDataType.value, *AtpClassifier*.atpFeature			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.42: ApplicationAssocMapElement

The existence of the reference in the role ApplicationAssocMapElement.type is governed by [constr_2633].

An example of the definition of an ApplicationAssocMapDataType can be found in Chapter B.1.1 in the appendix.

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.

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].

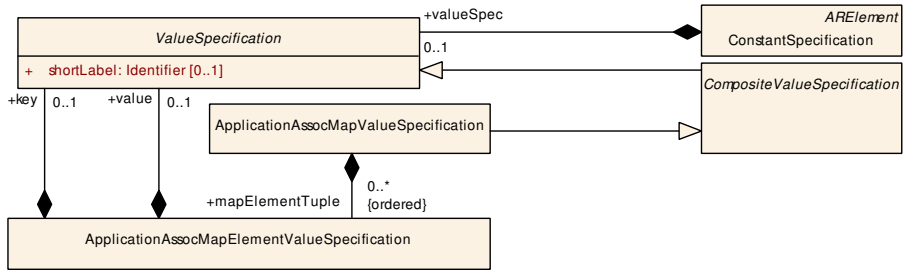


Figure 3.19: Formal model of the initialization of an `ApplicationAssocMapDataType`

[constr_1488] Initialization of a `DataPrototype` typed by an `ApplicationAssocMapDataType`

Imposition time: `IT_BefAraApiGen`

⌈A `DataPrototype` typed by an `ApplicationAssocMapDataType` shall only be initialized by an `ApplicationAssocMapValueSpecification`.⌋

As already mentioned, there is a semantic requirement that the *key* elements of an *associative map* need to be 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 for the initialization of an `ApplicationAssocMapDataType`, see [constr_1489].

[constr_1489] Uniqueness of `ApplicationAssocMapValueSpecification.mapElementTuple.key`

Imposition time: `IT_BefAraApiGen`

⌈The value of all `mapElementTuple.key` elements in the context of a given `ApplicationAssocMapValueSpecification` shall be unique.⌋

Class	**ApplicationAssocMapValueSpecification**			
Note	This meta-class represents the ability to define the initialization of an ApplicationAssocMapDataType. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *CompositeValueSpecification*, *ValueSpecification*			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValue Specification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, CompositeRuleBasedValueSpecification.argument, Constant Specification.valueSpec, CryptoServiceKey.developmentValue, DiagnosticEnvDataCondition.compare Value, DiagnosticEnvDataElementCondition.compareValue, DiagnosticEnvSovdDataCondition.compare Value, FieldSenderComSpec.initValue, ISignal.initValue, ISignal.timeoutSubstitutionValue, Nonqueued ReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, Nonqueued SenderComSpec.initValue, NvProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInit Value, NvRequireComSpec.initValue, ParameterDataPrototype.initValue, ParameterProvideComSpec.init Value, ParameterRequireComSpec.initValue, PersistencyDataRequiredComSpec.initValue, Persistency KeyValuePair.initValue, PortDefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, Record ValueSpecification.field, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalid Value, VariableDataPrototype.initValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	ApplicationAssocMapValueSpecification			
mapElement Tuple (ordered)	ApplicationAssocMap ElementValue Specification	*	aggr	This aggregation represents the initial values for the elements of the ApplicationAssocMapValueSpecification.

Table 3.43: ApplicationAssocMapValueSpecification

Class	ApplicationAssocMapElementValueSpecification			
Note	This meta-class represents the ability to define the initialization of the elements of an ApplicationAssoc MapDataType. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	ApplicationAssocMapValueSpecification.mapElementTuple			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
key	ValueSpecification	0..1	aggr	This aggregation represents the initialization of the key part of an AssociativeElementValueSpecification.
value	ValueSpecification	0..1	aggr	This aggregation represents the initialization of the value part of an AssociativeElementValueSpecification.

Table 3.44: ApplicationAssocMapElementValueSpecification

[constr_10126] Multiplicity of attribute ApplicationAssocMapElementValue-Specification.key

Imposition time: IT_BefAraApiGen

⌈For each ApplicationAssocMapElementValueSpecification, the attribute key shall exist.⌋

[constr_10127] Multiplicity of attribute ApplicationAssocMapElementValue-Specification.value

Imposition time: IT_BefAraApiGen

⌈For each ApplicationAssocMapElementValueSpecification, the attribute value shall exist.⌋

3.3.2.3 Attributes of SwDataDefProps

This section clarifies which attributes of meta-class SwDataDefProps used in the context of ApplicationDataTypes are relevant for the *AUTOSAR adaptive platform*.

A consequence of [constr_1478] is that it only lists the values of category that are exclusive to the *AUTOSAR adaptive platform*. For all other values of category that are also supported on the *AUTOSAR classic platform*, refer to a similar table contained in the specification of the [2, TPS Software Component Template].

Legend for [constr_1478]

In the columns headlined *Root Element* an “x” denotes that the attribute is applicable for the respective meta-class in the table header. Empty cells indicate that the attribute is not applicable for the respective *Root Element* at all.

A multiplicity definition in one of the columns headlined *Attribute Existence per Category* indicates that the attribute is applicable to the value of `category`. Empty cells indicate that the combination of attribute and value of `category` shall not be used.

[constr_1478] `SwDataDefProps` applicable to `ApplicationDataType`s exclusive to the *AUTOSAR adaptive platform*

Imposition time: `IT_BefAraApiGen`

⌈

Attributes of SwDataDefProps	Root Elem.		Attribute Existence per Category
	ApplicationAssocMapDataType	ApplicationAssocMapElement	ASSOCIATIVE_MAP
additionalNativeTypeQualifier			
annotation	x	x	*
baseType			
compuMethod			
dataConstr			
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			

⌋

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[5] 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 parlance 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 proper support for the capabilities of the C++ language.

While it would technically be feasible to extend the semantics of `ImplementationDataType` to support a C++ language binding, this would significantly water down the clarity and expressiveness of `ImplementationDataType`[6].

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.

[5] As explained in [2, TPS Software Component Template]

[6] 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.

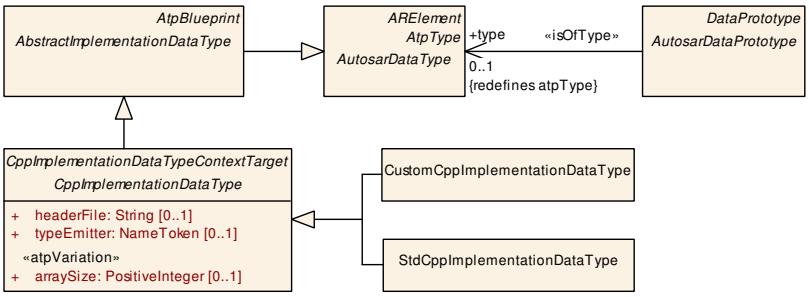


Figure 3.20: Modeling of meta-class `CppImplementationDataType`

[TPS_MANI_01166] Semantics of `CppImplementationDataType` ⌈The abstract meta-class `CppImplementationDataType` supports the modeling of data types specifically tailored towards support for a C++ language binding.⌋

[TPS_MANI_03197] Semantics of `StdCppImplementationDataType` ⌈Meta-class `StdCppImplementationDataType` supports the modeling of data types that will be mapped to C++ Standard Library features in the C++ language binding.⌋

Structures (`category = STRUCTURE`) and type aliases (`category = TYPE_REFERENCE`) are also modeled as `StdCppImplementationDataType`s for simplification reasons.

[TPS_MANI_03198] Semantics of `CustomCppImplementationDataType` ⌈Meta-class `CustomCppImplementationDataType` supports the modeling of data types that will be mapped to a custom implementation in the C++ language binding that is declared in the `headerFile`.⌋

The `category` values for a `CustomCppImplementationDataType` are restricted by [constr_1578].

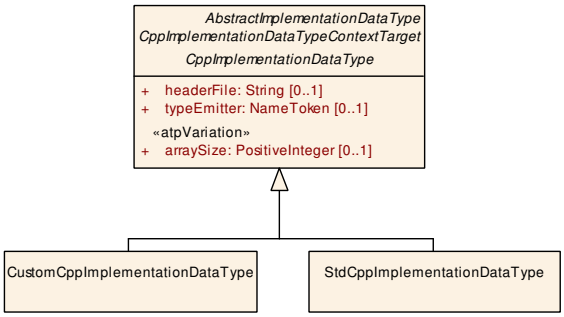


Figure 3.21: Specializations of `CppImplementationDataType`

This means that the modeling of primitive data types and strings is only possible with `StdCppImplementationDataType`s. 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.

[constr_1571] `CppImplementationDataType` is limited

Imposition time: `IT_BefAraApiGen`

⌈The usage of a `CppImplementationDataType` is limited to the context of `AdaptiveApplicationSwComponentType`s and `CompositionSwComponentType`s defined in the context of an `Executable`.⌋

[TPS_MANI_01167] **`AbstractImplementationDataType`** ⌈Meta-class `CppImplementationDataType` inherits from abstract base class `AbstractImplementationDataType` to become a valid target for specific references from other meta-classes that want to refer to "`ImplementationDataType` in general".⌋

Class	***CppImplementationDataType*** (abstract)			
Note	This meta-class represents the way to specify a reusable data type definition taken as a the basis for a C++ language binding This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	CustomCppImplementationDataType, StdCppImplementationDataType			
Aggregated by	ARPackage.element			
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:** vh.latestBindingTime=preCompileTime
headerFile	String	0..1	attr	Configuration of the Header File with the custom class declaration.
namespace (ordered)	SymbolProps	*	aggr	This aggregation allows for the definition an own namespace for the enclosing CppImplementationDataType.
subElement (ordered)	CppImplementationDataTypeElement	*	aggr	This represents the collection of sub-elements of the enclosing CppImplementationDataType
templateArgument (ordered)	CppTemplateArgument	*	aggr	This aggregation allows for the specification of properties of template arguments
typeEmitter	NameToken	0..1	attr	This attribute can be taken to control how the respective CppImplementationDataType is contributed to the language binding.
typeReference	CppImplementationDataType	0..1	ref	This reference shall be defined to define a type reference (a.k.a. typedef). **Tags:** atp.Splitkey=typeReference

Table 3.45: CppImplementationDataType

Class	**StdCppImplementationDataType**
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.recommendedPackage=CppImplementationDataTypes This Class is only used by the AUTOSAR Adaptive Platform.

▽

△

Class	StdCppImplementationDataType			
Base	ARElement, ARObject, AbstractImplementationDataType, AtpBlueprint, AtpClassifier, AtpType, AutosarDataType, CollectableElement, CppImplementationDataType, CppImplementationDataTypeContextTarget, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 3.46: StdCppImplementationDataType

Class	CustomCppImplementationDataType			
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.recommendedPackage=CppImplementationDataTypes This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, AbstractImplementationDataType, AtpBlueprint, AtpClassifier, AtpType, AutosarDataType, CollectableElement, CppImplementationDataType, CppImplementationDataTypeContextTarget, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 3.47: CustomCppImplementationDataType

Class	**AbstractImplementationDataType** (abstract)			
Note	This meta-class represents an abstract base class for different flavors of ImplementationDataType.			
Base	ARElement, ARObject, AtpBlueprint, AtpClassifier, AtpType, AutosarDataType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Subclasses	CppImplementationDataType, ImplementationDataType			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 3.48: 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 permitted for development on the *AUTOSAR adaptive platform* and therefore needs to be represented in the modeling approach.

[TPS_MANI_01168] Specification of a namespace for a CppImplementationDataType ⌈The ability to define a namespace for a CppImplementationDataType is expressed by means of the aggregation of SymbolProps at CppImplementationDataType in the role namespace.⌋

[constr_3443] Specification of a namespace for a StdCppImplementationDataType

Imposition time: IT_BefAraApiGen

⌈The 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 StdCppImplementationDataType.⌋

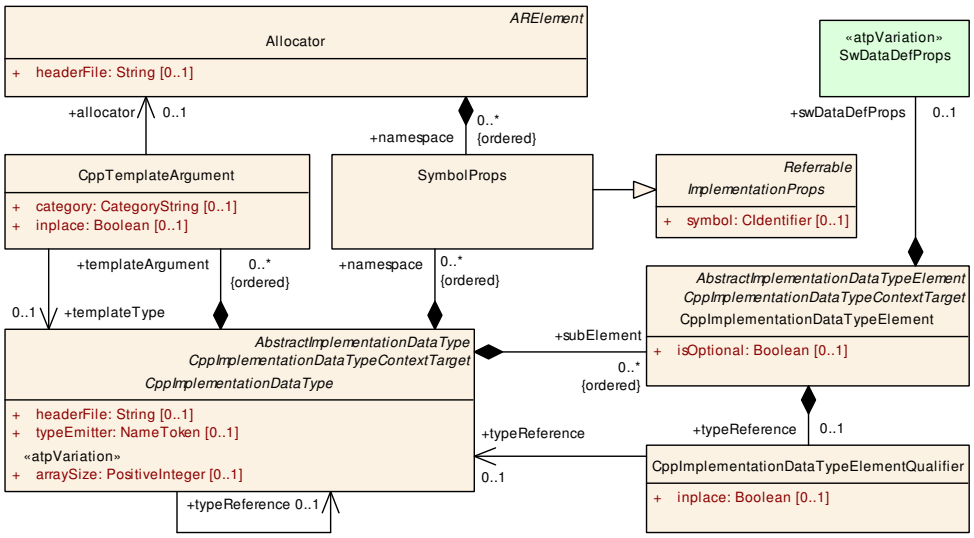


Figure 3.22: CppImplementationDataType overview

[TPS_MANI_01309] Semantics of attribute CppImplementationDataType.headerFile ⌈The 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.

⌋

[constr_1743] Attribute CppImplementationDataType.headerFile vs. CppImplementationDataType.typeEmitter

Imposition time: IT_BefAraApiGen

⌈The two attributes CppImplementationDataType.headerFile and CppImplementationDataType.typeEmitter shall always be used mutually exclusively.

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

[TPS_MANI_01176] Standardized value for attribute CppImplementationDataType.typeEmitter ⌈The 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.

⌋

[TPS_MANI_01177] Semantics of attribute CppImplementationDataType.typeEmitter ⌈The 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.

⌋

In the context of [TPS_MANI_01177], [TPS_MANI_01309] and [constr_1743] apply.

[TPS_MANI_01212] Usage of attribute typeEmitter in the context of a CustomCppImplementationDataType ⌈Attribute 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.⌋

[TPS_MANI_01169] Support for template data types ⌈Meta-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.⌋

[TPS_MANI_01174] Semantics of reference in the role CppTemplateArgument.templateType ⌈Attribute CppTemplateArgument.templateType specifies the data type to be filled in the respective position of the template in the language binding.⌋

[TPS_MANI_01175] Semantics of reference in the role CppTemplateArgument.allocator ⌈Attribute CppTemplateArgument.allocator specifies the behavior of an allocator class to be filled in the respective position of the template in the language binding.⌋

[constr_1576] Existence of CppTemplateArgument.templateType vs. CppTemplateArgument.allocator

Imposition time: IT_BefAraApiGen

⌈For any given CppTemplateArgument, **at most one of** the references

- CppTemplateArgument.templateType or
- CppTemplateArgument.allocator

shall exist.⌋

[TPS_MANI_01201] Standardized values for attribute CppTemplateArgument.category ⌈AUTOSAR reserves the following values for attribute CppTemplateArgument.category:

ASSOC_MAP_KEY : the specific CppTemplateArgument represents the *key* data-type of an associative map.

ASSOC_MAP_VALUE : the specific CppTemplateArgument represents the *value* data-type of an associative map.

⌋

Class	**CppTemplateArgument**			
Note	This meta-class has the ability to define properties for template arguments. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	*CppImplementationDataType*.templateArgument			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allocator	Allocator	0..1	ref	This reference identifies the applicable allocator.
category	CategoryString	0..1	attr	This attribute shall be used to contribute further clarification regarding the semantics of the enclosing Cpp TemplateArgument.

▽

△

Class	**CppTemplateArgument**			
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).
templateType	CppImplementation DataType	0..1	ref	This reference identifies the data type of the specific template argument required for the language binding.

Table 3.49: CppTemplateArgument

[TPS_MANI_01171] Modeling of structured data types ⌈Meta-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.⌋

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]).

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 existence of attributes of CppImplementationDataType depending on the usage of data categories eligible for CppImplementationDataType according to [constr_1578] is documented in [constr_10417].

[constr_10417] only includes the attributes of CppImplementationDataType that depend on the usage of category. All attributes of CppImplementationDataType that are not mentioned in [constr_10417] are applicable for all values of category.

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++.

On the *AUTOSAR adaptive platform*, it is not allowed to reference an ImplementationDataType from within SwDataDefProps. This aspect is expressed in a general form by [constr_1579].

As a consequence of [constr_1579], type-references have to be done differently on the *AUTOSAR adaptive platform*. For this purpose dedicated references are available.

[constr_1578] Applicable data categories

Imposition time: IT_BefAraApiGen

⌈

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.

⌋

Class	**CppImplementationDataTypeElement**			
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. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractImplementationDataTypeElement*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *CppImplementationDataType*.subElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

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.
swDataDef Props	SwDataDefProps	0..1	aggr	This aggregation allows for the definition of qualifying properties of the enclosing CppImplementationDataType Element. **Tags:** atp.Splitkey=swDataDefProps
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.Splitkey=typeReference

Table 3.50: CppImplementationDataTypeElement

[TPS_MANI_01172] Description of type references in the scope of meta-class CppImplementationDataType ⌈The reference CppImplementation-DataType.typeReference can be used to create a type reference from the enclosing CppImplementationDataType to another CppImplementationDataType.⌋

[TPS_MANI_01173] Description of type references in the scope of meta-class CppImplementationDataTypeElement ⌈The attribute CppImplementation-DataTypeElement.typeReference can be used to create a reference to the CppImplementationDataType that shall apply for the enclosing Cpp-ImplementationDataTypeElement.⌋

[TPS_MANI_03196] Semantics of CppImplementationDataTypeElement-Qualifier.inplace attribute ⌈The CppImplementationDataTypeElement-Qualifier.inplace attribute defines whether the data type of the Cpp-ImplementationDataTypeElement 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.type-Reference.typeReference shall be used in the C++ language binding.
- if CppImplementationDataTypeElement.typeReference.inplace is set to true then only the **properties** of the CppImplementationData-Type referenced in the role CppImplementationDataTypeElement.type-Reference.typeReference shall be used in the C++ language binding and the shortName is ignored.

⌋

[constr_10417] Existence of attributes of CppImplementationDataTypes depending on the category

Imposition time: IT_BefAraApiGen

⌈

Attributes of CppImplementationDataType	Attribute Existence per Category							
	VALUE	TYPE_REFERENCE	STRUCTURE	VARIANT	ARRAY	VECTOR	ASSOCIATIVE_MAP	STRING
subElement			1..*					
templateArgument				1..*	1	1..*	2..*	0..1
typeReference		1						
arraySize					1	0..1[7]		

⌋

The CppImplementationDataTypeElement.typeReference is realized as an Association Class that allows adding the inplace attribute to the typeReference.

Class	**CppImplementationDataTypeElementQualifier**			
Note	This element qualifies the typeReference of the CppImplementationDataTypeElement to the Cpp ImplementationDataType. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	CppImplementationDataTypeElement.typeReference			
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.
typeReference	CppImplementation DataType	0..1	ref	This reference defines a type reference. **Tags:** atp.Splitkey=typeReference

Table 3.51: CppImplementationDataTypeElementQualifier

[constr_10128] Multiplicity of attribute CppImplementationDataTypeElementQualifier.typeReference

Imposition time: IT_BefAraApiGen

⌈For each CppImplementationDataTypeElementQualifier, the attribute typeReference shall exist⌋

Figure B.7 shows an example of a Structure where the typeReference of one subElement is classified as inplace.

[7] This depends on the existence of an allocator, see [TPS_MANI_03186].

[constr_1659] Restriction for the usage of CppImplementationDataTypeElementQualifier.inplace

Imposition time: IT_BefAraApiGen

⌈The 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 category TEXTTABLE

⌋

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 category VALUE
- CustomCppImplementationDataType
- CppImplementationDataType of category TYPE_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] Semantics of CppTemplateArgument.inplace attribute
⌈The 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.

⌋

[constr_1660] Restriction for the usage of CppTemplateArgument.inplace

Imposition time: IT_BefAraApiGen

⌈The 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

⌋

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 category VALUE
- CustomCppImplementationDataType
- CppImplementationDataType of category TYPE_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.

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] Combination of CppImplementationDataTypeElement.isOptional and CppImplementationDataTypeElementQualifier.inplace

Imposition time: IT_BefAraApiGen

⌈If a CppImplementationDataTypeElement is typed by a CppImplementationDataType of category STRUCTURE then the combination of attribute CppImplementationDataTypeElement.isOptional set to true and CppImplementationDataTypeElement.typeReference.inplace set to true is not allowed.⌋

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] CppTemplateArgument.templateType reference to StdCppImplementationDataType of category STRUCTURE and the inplace flag

Imposition time: IT_BefAraApiGen

⌈CppTemplateArgument.templateType that points to a StdCppImplementationDataType of category STRUCTURE shall have the inplace attribute set to **false**.⌋

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.

Class	**Allocator**			
Note	This meta-class represents the ability to specify an optional custom C++ allocator for a C++ type which may dynamically grow beyond it's initial allocated size during it's lifetime. Any storage principles are defined in the implementation of the allocator itself, which should implement the ISO C++ std::allocator_ traits interface. **Tags:** atp.recommendedPackage=Allocators This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
headerFile	String	0..1	attr	Configuration of the Header File with the custom class declaration
namespace (ordered)	SymbolProps	*	aggr	This aggregation allows for the definition of a namespace of an Allocator.

Table 3.52: Allocator

[constr_3446] `CppTemplateArgument` with `allocator` reference and the `inplace` flag

Imposition time: `IT_BefAraApiGen`

⌈A `CppTemplateArgument` that points with an `allocator` reference to an `Allocator` shall not have the `inplace` flag set to a value.⌋

ISO C++ permits the usage of an `Allocator` only for those `StdCppImplementationDataType`s which may dynamically grow in size. Thus, it is necessary to restrict the other `StdCppImplementationDataType`s. This restriction ([constr_6905]) is deliberately not imposed on `CustomCppImplementationDataType`s because of their custom nature.

[constr_6905] `CppTemplateArgument.allocator`

Imposition time: `IT_BefAraApiGen`

⌈The reference `CppTemplateArgument.allocator` shall only exist if the `StdCppImplementationDataType.category` is set to either of the values:

- `STRING`
- `VECTOR`
- `ASSOCIATIVE_MAP`

⌋

3.3.3.2 Attributes of SwDataDefProps

In contrast to the situation on the *AUTOSAR classic platform* (see [constr_1009]), the usage of `SwDataDefProps` plays a significantly less prominent role in the definition of `CppImplementationDataType` on the *AUTOSAR adaptive platform*.

This aspect is clarified by [constr_1579].

A consequence of [constr_1578] is that [constr_1579] 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*, refer to a similar table contained in the specification of the [2, TPS Software Component Template].

It is especially notable that on the *AUTOSAR adaptive platform* the usage of the reference in the role `SwDataDefProps.implementationDataType` is **not** supported.

A dedicated reference in one of the following roles

- `CppImplementationDataType.typeReference`
- `CppImplementationDataTypeElement.typeReference`

shall be used instead.

[constr_1579] SwDataDefProps applicable to CppImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*

Imposition time: IT_BefAraApiGen

⌈

Attributes of SwDataDefProps	Root Element		Attribute Existence per Category							
	CppImplementationDataType	CppImplementationDataTypeElement	VALUE	TYPE_REFERENCE	STRUCTURE	VARIANT	ARRAY	VECTOR	ASSOCIATIVE_MAP	STRING
additionalNativeTypeQualifier										
annotation	x	x	*	*	*	*	*	*	*	*
baseType										
compuMethod	x	x		0..1						
dataConstr.dataConstrRule.physConstrs	x	x		d/c			d/c	d/c		
dataConstr.dataConstrRule.internalConstrs	x	x		0..1			0..1	0..1		
displayFormat	x	x	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1
implementationDataType										
invalidValue	x	x		0..1						0..1
stepSize										
swAddrMethod										
swAlignment										
swBitRepresentation										
swCalibrationAccess										
swCalprmAxisSet										
swComparisonVariable										
swDataDependency										
swHostVariable										
swImplPolicy										
swIntendedResolution										
swInterpolationMethod										
swIsVirtual										
swPointerTargetProps										
swPointerTargetProps.swDataDefProps										
swRecordLayout										
swRefreshTiming	x		0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1
swTextProps										
swValueBlockSize										
unit										

▽

△

Attributes of SwDataDefProps	Root Element		Attribute Existence per Category							
	CppImplementationDataType	CppImplementationDataTypeElement	VALUE	TYPE_REFERENCE	STRUCTURE	VARIANT	ARRAY	VECTOR	ASSOCIATIVE_MAP	STRING
valueAxisDataType										

⌋

Legend for [constr_1579]

In the columns headlined *Root Element* an “x” denotes that the attribute is applicable for the respective meta-class in the table header. Empty cells indicate that the attribute is not applicable for the respective *Root Element* at all.

A multiplicity definition in one of the columns headlined *Attribute Existence per Category* indicates that the attribute is applicable to the value of `category`. Empty cells indicate that the combination of attribute and value of `category` shall not be used.

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] Applicability of attribute `invalidValue` on `CppImplementationDataType` of `category` `TYPE_REFERENCE`

Imposition time: `IT_BefAraApiGen`

⌈If a `CppImplementationDataType` of `category` `TYPE_REFERENCE` has an `invalidValue` defined, then the referenced `CppImplementationDataType` (via `typeReference`) shall eventually be of `category`.⌋

The following rationale exists for the support of `invalidValue` for specific `category`s of data types:

- The usage of category VALUE 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 category TYPE_REFERENCE represents the main-stream use case for the definition of an invalidValue.

[TPS_MANI_01450] Definition of invalidValue for DataPrototype typed by a CppImplementationDataType of category STRING ⌈The attribute invalidValue in the context of a DataPrototype that is typed by a CppImplementationDataType of category STRING shall be (after possible redirections via ConstantReference are resolved) assigned to a TextValueSpecification.⌋

[TPS_MANI_01451] Definition of invalidValue for DataPrototype typed by a CppImplementationDataType of category TYPE_REFERENCE ⌈The attribute invalidValue in the context of a DataPrototype that is typed by a CppImplementationDataType of category TYPE_REFERENCE shall be (after possible redirections via ConstantReference are resolved) assigned to by either

- TextValueSpecification if the DataPrototype represents an enumeration according to [TPS_MANI_03187] or
- NumericalValueSpecification.

⌋

For CppImplementationDataType, there are (see [constr_1579]) only two values of attribute CompuMethod.category allowed for the definition of a CppImplementationDataType.swDataDefProps.compuMethod:

- TEXTTABLE ([TPS_MANI_03187]) and
- BITFIELD_TEXTTABLE ([TPS_MANI_03202]).

This restriction motivates the existence of [constr_10566].

[constr_10566] Allowed values of attribute AutosarDataType.swDataDefProps.compuMethod.category

Imposition time: IT_BefAraApiGen

⌈The only allowed values for attribute CppImplementationDataType[8].swDataDefProps.compuMethod.category are

- TEXTTABLE
- BITFIELD_TEXTTABLE

⌋

3.3.3.3 Primitive Data Types

[TPS_MANI_03192] CppImplementationDataType of category VALUE ⌈The primitive data types like Boolean, fixed-width integer data types and floating-point data types are described as CppImplementationDataTypes of category VALUE.⌋

[TPS_MANI_01411] Initialization of a DataPrototype typed by a StdCppImplementationDataType of category VALUE that represents a Boolean ⌈A DataPrototype typed by a StdCppImplementationDataType of category VALUE that represents a Boolean shall be initialized by a NumericalValueSpecification that contains one of the following values:

- 0
- 1
- true
- false

⌋

[TPS_MANI_03193] CppImplementationDataType where attribute category is set to TYPE_REFERENCE ⌈The definition of a CppImplementationDataType of category TYPE_REFERENCE creates an alias for another CppImplementationDataType that is referenced by the typeReference.⌋

3.3.3.4 String Data Type

[TPS_MANI_03178] StdCppImplementationDataType of category STRING ⌈A StdCppImplementationDataType of category STRING represents a container data type for a sequence of characters.⌋

[8] [constr_1579] applies

[TPS_MANI_03179] C++ language binding of StdCppImplementationDataTypes of category STRING ⌈A CppImplementationDataType of category STRING shall be implemented as ara::core::String.⌋

The formulation of [TPS_MANI_03179] leaves room for potential later extensions towards the support for other storage formats.

Since C++ strings are not containerized types, they do not receive a containerized typename in the template<> argument list on code level. It is therefore necessary in this case to restrict the referenced templateType in CppTemplateArgument.

[constr_6815] Existence of CppTemplateArgument.templateType for CppImplementationDataType of category STRING

Imposition time: IT_BefAraApiGen

⌈In a CppImplementationDataType of category STRING, the reference templateType shall not exist.⌋

[TPS_MANI_03188] Usage of an Allocator for a StdCppImplementationDataType of category STRING ⌈A StdCppImplementationDataType of category STRING is allowed to aggregate a CppTemplateArgument that refers to an Allocator with the allocator reference.⌋

An example of the definition of a string data type can be found in Chapter B.1.2.

3.3.3.5 Array Data Type

[TPS_MANI_03169] CppImplementationDataType with fixed size array semantics ⌈A CppImplementationDataType of category ARRAY represents a container data type that encapsulates fixed-size arrays.⌋

[TPS_MANI_03170] CppImplementationDataType of category ARRAY ⌈For a C++ binding, a CppImplementationDataType of category ARRAY 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).

⌋

[TPS_MANI_03171] Value type of a CppImplementationDataType of category ARRAY ⌈The type of elements contained in a CppImplementationDataType of category ARRAY is defined by the aggregated templateArgument and the corresponding templateType that defines the data type of the CppTemplateArgument.⌋

[constr_3433] Aggregation of templateArguments for an ARRAY

Imposition time: IT_BefAraApiGen

⌈CppImplementationDataType of category ARRAY that boils down to ara::core::Array shall aggregate exactly one templateArgument that defines the type of elements contained in the CppImplementationDataType of category ARRAY.⌋

[TPS_MANI_03172] Size of a CppImplementationDataType of category ARRAY
⌈The primitive attribute arraySize of a CppImplementationDataType of category ARRAY shall be used to define the size of the array.⌋

[TPS_MANI_03173] Definition of a multidimensional Array ⌈A multidimensional CppImplementationDataType of category ARRAY contains nested CppImplementationDataTypes of category ARRAY.

The CppImplementationDataType of category ARRAY that represents the outer array will refer to a CppImplementationDataType of category ARRAY 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 simply nesting more CppImplementationDataTypes of category ARRAY.

The array element itself is specified by the innermost CppImplementationDataType with category different from ARRAY.⌋

Examples of the definition of CppImplementationDataType of category ARRAY can be found in Chapter B.1.3.

3.3.3.6 Vector Data Type

[TPS_MANI_03174] CppImplementationDataType with variable size array semantics ⌈A CppImplementationDataType of category VECTOR represents a container data type that encapsulates variable-size arrays.⌋

[TPS_MANI_03175] CppImplementationDataType of category VECTOR ⌈For a C++ binding, a CppImplementationDataType of category VECTOR 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).

⌋

[TPS_MANI_03176] Value type of a CppImplementationDataType of category VECTOR ⌈The type of elements contained in a CppImplementationDataType of category VECTOR is defined by the aggregated templateArgument and the corresponding templateType that defines the data type of the CppTemplateArgument.⌋

[constr_3434] Aggregation of templateArguments for a VECTOR

Imposition time: IT_BefAraApiGen

⌈A CppImplementationDataType of category VECTOR that boils down to ara::core::Vector shall aggregate at least one and at most two templateArgument where the first one shall define the type of elements contained in the CppImplementationDataType of category VECTOR with the templateType reference.⌋

The semantics of the second templateArgument is described in [constr_10416].

[constr_10416] Aggregation of optional templateArgument that defines an Allocator for a VECTOR

Imposition time: IT_BefAraApiGen

⌈If a CppImplementationDataType of category VECTOR that boils down to ara::core::Vector aggregates (in addition to the templateArgument mentioned in [constr_3434]) one additional templateArgument, then this additional templateArgument shall reference the Allocator by means of the allocator reference.⌋

[TPS_MANI_03186] Usage of arraySize in the case of a Vector ⌈If the CppImplementationDataType of category VECTOR 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.⌋

[TPS_MANI_03177] Definition of a multidimensional Vector ⌈A multidimensional CppImplementationDataType of category VECTOR contains nested CppImplementationDataTypes of category VECTOR.

The CppImplementationDataType of category VECTOR that represents the outer vector will refer to a CppImplementationDataType of category VECTOR 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 simply nesting more CppImplementationDataTypes of category VECTOR.

The vector element itself is specified by the innermost CppImplementationDataType with category different from VECTOR.⌋

The meta-model supports the creation of a reference to a specific element (identified by means of the index) of a CppImplementationDataType of category VECTOR.

However, this may lead to a problem at run-time if the specific element does not exist at that 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 category VECTOR becomes the target of a reference in the model.

Examples of the definition of CppImplementationDataType of category VECTOR can be found in Chapter B.1.4.

3.3.3.7 Struct Data Type

[TPS_MANI_03180] Definition of Structures ⌈A StdCppImplementationDataType of category STRUCTURE represents a data type for holding an ordered collection of variables of arbitrary data types.⌋

[TPS_MANI_03181] Definition of members in StdCppImplementationDataType of category STRUCTURE ⌈Members in a StdCppImplementationDataType of category STRUCTURE are defined by ordered CppImplementationDataTypeElements that are aggregated in the role subElement by the enclosing StdCppImplementationDataType of category STRUCTURE.

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

The type of each member of the structure is defined by the typeReference to a CppImplementationDataType.⌋

The inplace flag that is able to classify a CppImplementationDataTypeElement.typeReference is documented in [TPS_MANI_03196].

Examples of the definition of CppImplementationDataType of category STRUCTURE can be found in Chapter B.1.5.

3.3.3.8 Enumeration Data Type

[TPS_MANI_03187] Definition of enumeration types ⌈In 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 category TYPE_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 semantic definition via an attached CompuMethod with category TEXTTABLE rather than part of the structural description.⌋

The rules for the usage of a CompuMethod with category TEXTTABLE are the same as in the AUTOSAR Classic Platform and are described in the [2, TPS Software Component Template].

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.

An example of the definition of an enumeration can be found in Chapter B.1.6.

3.3.3.9 Map Data Type

[TPS_MANI_03183] **CppImplementationDataType of category ASSOCIATIVE_MAP** ⌈A CppImplementationDataType of category ASSOCIATIVE_MAP represents a container that contains key-value pairs with unique keys.⌋

[TPS_MANI_03184] **CppImplementationDataType of category ASSOCIATIVE_MAP** ⌈For a C++ binding, a CppImplementationDataType of category ASSOCIATIVE_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).

⌋

[TPS_MANI_03185] **Structure of a CppImplementationDataType of category ASSOCIATIVE_MAP** ⌈A CppImplementationDataType of category ASSOCIATIVE_MAP that boils down to a ara::core::Map shall aggregate the following CppTemplateArguments:

- one CppTemplateArgument shall have the category ASSOC_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 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.

⌋

Examples of the definition of CppImplementationDataType of category ASSOCIATIVE_MAP can be found in Chapter B.1.7.

3.3.3.10 Variant Data Type

[TPS_MANI_03189] Definition of CppImplementationDataType of category VARIANT ⌈A CppImplementationDataType of category VARIANT represents a type-safe union.⌋

[TPS_MANI_03190] CppImplementationDataType of category VARIANT ⌈For 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).

⌋

[TPS_MANI_03191] Definition of type alternatives stored in a VARIANT ⌈A 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.⌋

[constr_3429] No allocator usage for CppImplementationDataTypes of category VARIANT

Imposition time: IT_BefAraApiGen

⌈CppImplementationDataType of category VARIANT is not allowed to aggregate a templateArgument that points to an Allocator in the role allocator.⌋

[TPS_MANI_01393] Initialization of a data object typed by a CppImplementationDataType of category VARIANT ⌈The initialization of a data object that is typed by a CppImplementationDataType of category VARIANT is done by using a RecordValueSpecification with two fields:

- The first `field` identifies the index of the element inside the variant data type. The index of the first element shall be 1 and increasing from there.
- The second `field` identifies the "payload" of the initialization and consists of a `ValueSpecification` that matches the data type of the element in the variant data type that is to be initialized.

⌋

The convention to let the index be 1-based is in place to achieve the same initialization semantics as on the *AUTOSAR classic platform*.

This means that – in principle – the same `RecordValueSpecification` can be used to initialize a data object typed by a variant data type (*AUTOSAR adaptive platform*) and a corresponding data object typed by a `Wrapped Union Data Type` on the *AUTOSAR classic platform*.

The conversion of the index value to a numerical value used in an expression of the language binding of the variant data type is out of scope of the TPS Manifest Specification.

In other words, if the actual initialization uses a 0-based index value, the value provided in the model has to be decremented by 1.

[constr_10399] Allowed interval of the "index" `field` according to the initialization rule for data object typed by a `CppImplementationDataType` of `category VARIANT`

Imposition time: `IT_BefAraApiGen`

⌈The allowed value range of the "index" `field` of a `RecordValueSpecification` according to [TPS_MANI_01393] goes from 1 to the number of `templateArgument`s owned by the `CppImplementationDataType` of `category VARIANT`.⌋

Examples of the definition of `CppImplementationDataType` of `category VARIANT` can be found in Chapter B.1.8.

3.3.3.11 Bitfield Data Type

[TPS_MANI_03202] Definition of bitfield types ⌈In the AUTOSAR meta-model, 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 `category VALUE`.

A `CompuMethod` of `category BITFIELD_TEXTTABLE` is used to assign a special meaning to each bit of the primitive `StdCppImplementationDataType`.⌋

CompuScales with a mask inside of the CompuMethod of category BITFIELD_TEXTTABLE define 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 category BITFIELD_TEXTTABLE are the same as in the AUTOSAR Classic Platform and are described in the [2, TPS Software Component Template].

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 the basis for the serialization of the payload in the network binding.

[TPS_MANI_03223] Existence of CppImplementationDataType ⌈The 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.⌋

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 category TYPE_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 category VALUE this shall include the possibility of mapping to a CppImplementationDataType of category TYPE_REFERENCE which refers to another CppImplementationDataType of category VALUE.

2. **[constr_5033] Compatibility of data types with category VALUE**

   Imposition time: IT_BefAraApiGen

   ⌈An ApplicationDataType of category VALUE can only be mapped to a CppImplementationDataType which also has category VALUE.⌋

   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.

3. **[constr_5034] Compatibility of data types with category BOOLEAN**

   Imposition time: IT_BefAraApiGen

   ⌈An ApplicationDataType of category BOOLEAN can only be mapped to a CppImplementationDataType of category VALUE.⌋

4. **[constr_5035] Compatibility of data types with category STRING**

   Imposition time: IT_BefAraApiGen

   ⌈A CppImplementationDataType where attribute category is set to the value STRING can only be mapped to an ApplicationDataType

   - where attribute category is set to the value STRING **and**
   - where attribute swDataDefProps.swTextProps.baseType.baseTypeDefinition.baseTypeEncoding is set to the value UTF-8.

   ⌋

   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.

5. **[constr_5036] Compatibility of data types with category ARRAY**

   Imposition time: IT_BefAraApiGen

   ⌈An ApplicationDataType of category ARRAY can only be mapped to

   - a CppImplementationDataType of category ARRAY **or**
   - a CppImplementationDataType of category VECTOR.

   ⌋

   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 to the corresponding application data and vice versa.

6. **[constr_5037] Compatibility of data types with `category ARRAY` with `variableSize`**

   Imposition time: IT_BefAraApiGen

   ⌈An ApplicationDataType of category ARRAY that includes one ApplicationArrayElement with arraySizeSemantics set to variableSize in one of the defined dimensions shall be mapped to

   - a CppImplementationDataType of category VECTOR

   ⌋

7. **[constr_5038] Compatibility of data types with `category ARRAY` with `fixedSize`**

   Imposition time: IT_BefAraApiGen

   ⌈An ApplicationDataType of category ARRAY that includes only ApplicationArrayElements with arraySizeSemantics set to fixedSize in all defined dimensions shall be mapped to

   - a CppImplementationDataType of category ARRAY

   ⌋

8. **[constr_5039] Compatibility of data types with `category STRUCTURE`**

   Imposition time: IT_BefAraApiGen

   ⌈An ApplicationDataType of category STRUCTURE can only be mapped to a CppImplementationDataType of category STRUCTURE.⌋

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

9. **[constr_5040] Compatibility of `ApplicationRecordDataType` and `CppImplementationDataType` that both represent an `Optional Element Structure`**

   Imposition time: IT_BefAraApiGen

   ⌈An ApplicationRecordDataType that represents an Optional Element Structure can only be mapped to a CppImplementationDataType of category STRUCTURE that represents an Optional Element Structure if corresponding pairs of elements have the same value of the attribute isOptional.⌋

10. **[constr_5041] Compatibility of data types with `category ASSOCIATIVE_MAP`**

    Imposition time: IT_BefAraApiGen

    ⌈An ApplicationDataType of category ASSOCIATIVE_MAP can only be mapped to a CppImplementationDataType of category ASSOCIATIVE_MAP.⌋

11. **[constr_5042] No data type mapping for `CppImplementationDataType` of `category VARIANT`**

    Imposition time: `IT_BefAraApiGen`

    ⌈An `ApplicationDataType` shall never be mapped to a `CppImplementationDataType` of `category VARIANT`.⌋

12. **[constr_5043] Forbidden mappings to `CppImplementationDataType`**

    Imposition time: `IT_BefAraApiGen`

    ⌈An `ApplicationDataType` of `category` `COM_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`.⌋

The categories listed in [constr_5043] are not supported because there is no use case for the usage in the 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 meta-class `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] `DataTypeMap` for composite data types

Imposition time: `IT_BefAraApiGen`

⌈In 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 `DataTypeMappingSet`s that are referenced in a `PortInterfaceToDataTypeMapping` that also references the mentioned `ServiceInterface`.⌋

3.3.5 Optional Members in complex Data Structures

3.3.5.1 Background

The *AUTOSAR adaptive platform* supports the usage of a `TLV`[9] data encoding on the SOME/IP transport layer. `TLV` is typically used where at least a part of the transmitted data optionally exists and is filled with meaningful values.

[9] This abbreviation stands for tag-length-value

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 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 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 number 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 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 implementing the concept of defining the optionality on the level of the AutosarDataType.

3.3.5.2 Definition of Optionality

As mentioned before, the concrete definition of optionality on the level of an `AutosarDataType` is done by indicating 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 `ApplicationDataType`s 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 3.23 shows the modeling of optionality on the level of `ApplicationDataType`.

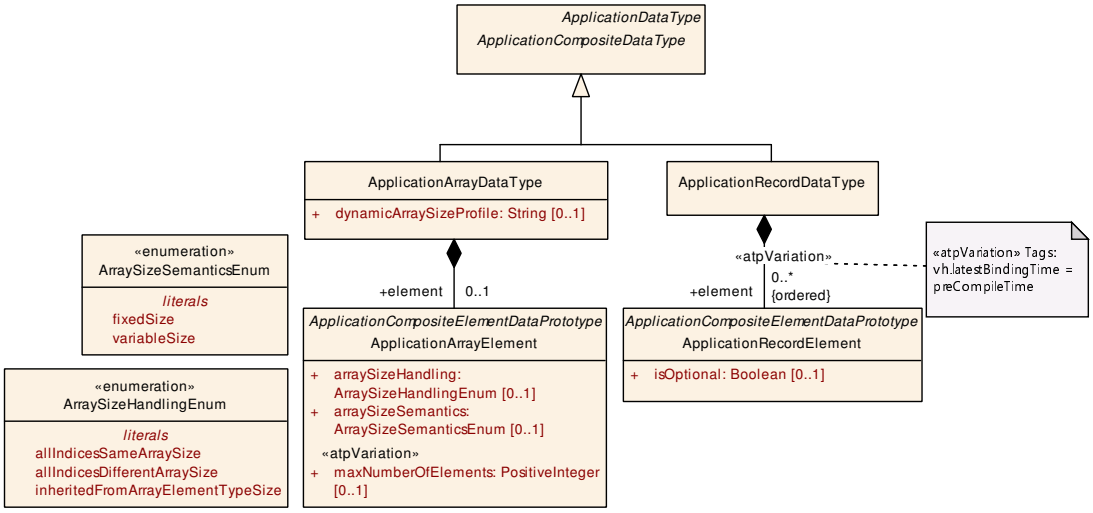


Figure 3.23: Modeling of optionality on the level of `ApplicationDataType`

[TPS_MANI_01184] Definition of optional elements on the level of `ApplicationDataType` ⌈The 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 `ApplicationRecordElement` **shall be considered mandatory**.⌋

Class	**ApplicationRecordDataType**
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*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

Class	ApplicationRecordDataType			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
element (ordered)	ApplicationRecordElement	*	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 3.53: ApplicationRecordDataType

Class	ApplicationRecordElement			
Note	Describes the properties of one particular element of an application record data type.			
Base	*ARObject*, *ApplicationCompositeElementDataPrototype*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ApplicationRecordDataType.element, *AtpClassifier*.atpFeature			
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 ApplicationRecordElement 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 3.54: ApplicationRecordElement

On top of that, it is still possible to use CppImplementationDataType directly for the definition of a ServiceInterface.

[TPS_MANI_01185] Definition of optional elements on the level of CppImplementationDataType ⌈The 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**.⌋

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] Attribute NotAvailableValueSpecification.defaultPattern is not applicable ⌈The 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 if it has not been received.⌋

3.3.6 Usage of constant Values

In some cases, configuration of the *AUTOSAR adaptive Platform* requires the usage of constant values. The AUTOSAR meta-model provides a wide range of possibilities for the definition of constant values, as documented in the [2, TPS Software Component Template].

A part of this range of possibilities, as sketched in Figure 3.24, is available to the *AUTOSAR adaptive Platform*.

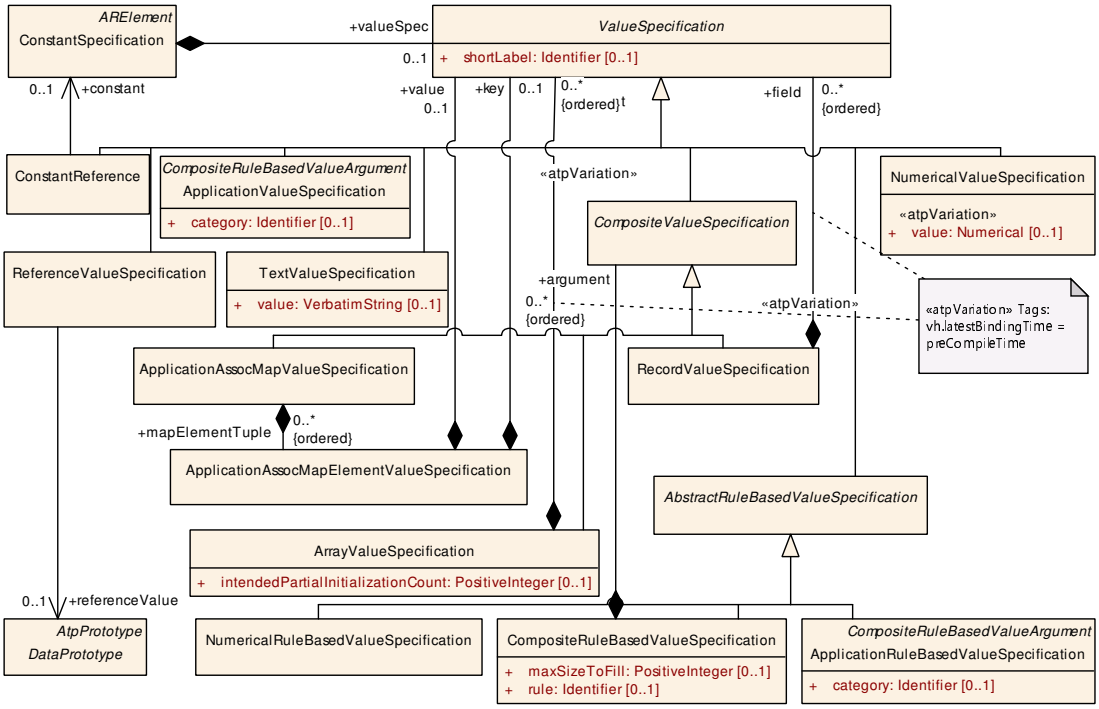


Figure 3.24: Overview about constant value definitions usable on the *AUTOSAR adaptive Platform*

3.4 Serialization Of Data

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 `TransformationPropsToServiceInterfaceElement-Mapping` 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 3.4.1.
- Fine granular modeling on the level of DataPrototypes described in this chapter. This case is described in detail in Chapter 3.4.2.

[advisory_03004] Definition of SOME/IP serialization properties via ApSomeipTransformationProps

Imposition time: IT_BefAraApiGen

⌈For each ServiceInterface element (i.e. an event, field, method) of a ServiceInterface that is referenced by the SomeipServiceInterfaceDeployment either a TransformationPropsToServiceInterfaceElementMapping should be defined or SomeipDataPrototypeTransformationProps should exist for the DataPrototype of the ServiceInterface element and if the DataPrototype of the ServiceInterface element is composite for each included DataPrototype of the ServiceInterface element (see [constr_3361] for more details).⌋

In other words, the input for the SOME/IP serialization of a ServiceInterface element should be present in the design phase so that the serialization code can be created.

3.4.1 Default Values for Serialization Properties

[TPS_MANI_03101] SOME/IP serialization ⌈The ApSomeipTransformationProps meta-class that is referenced by the TransformationPropsToServiceInterfaceElementMapping in the role transformationProps provides the ability to define SOME/IP serialization settings for ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, trigger, methodCall, methodReturn, or field.⌋

[constr_3395] TransformationPropsToServiceInterfaceElementMapping is restricted to one single ServiceInterface

Imposition time: IT_BefAraApiGen

⌈All ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, trigger, methodCall, methodReturn, or field shall be aggregated by the same ServiceInterface in the role event, trigger, method or field.⌋

[constr_9370] `ApSomeipTransformationProps` shall be defined at most once for a `ServiceInterface` element

Imposition time: `IT_BefAraApiGen`

⌈For each `ServiceInterface` element (i.e. an `event`, `field`, `methodCall` or `methodReturn`) at most one `TransformationPropsToServiceInterfaceElementMapping` shall be defined that references `ApSomeipTransformationProps` in the role `transformationProps`.⌋

[constr_9370] ensures that SOME/IP serialization rules for a `ServiceInterface` element are clearly defined and no contradicting rules exist.

[TPS_MANI_01516] Different `transformationProps` for the `methodCall` and `methodReturn` where attribute `argument` is set to `direction inout` ⌈If a `ClientServerOperation` with at least one `argument` where attribute `direction` is set to `inout` is referenced by different `TransformationPropsToServiceInterfaceElementMapping`s in the role `methodCall` and `methodReturn`, then the `argument`(s) with `direction inout` are transported in both, the call and the return message, and may be serialized differently for the call and the return messages based on the configuration of the `transformationProps`.⌋

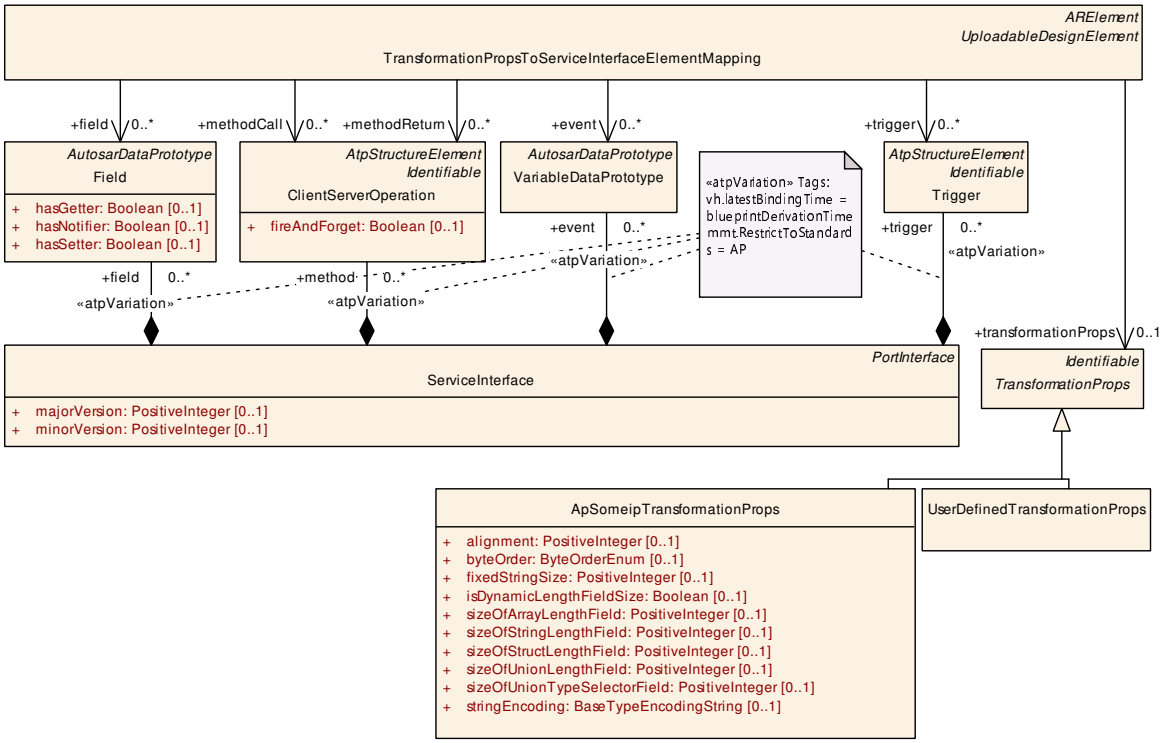


Figure 3.25: Association of serialization properties with a ServiceInterface

[TPS_MANI_03103] Default size for all array and map length fields ⌈The 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, methodCall, methodReturn, or field.⌋

[TPS_MANI_03104] Default size for all structure length fields ⌈The 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, methodCall, methodReturn, or field.⌋

[TPS_MANI_03117] Default size for all string length fields ⌈The 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, methodCall, methodReturn, or field.⌋

[TPS_MANI_03105] Default size for all union length fields ⌈The 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, methodCall, methodReturn, or field.⌋

[TPS_MANI_03106] Default size for all union type selector fields ⌈The 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, methodCall, methodReturn, or field.⌋

[TPS_MANI_03107] Default alignment for all dynamic DataPrototypes ⌈The 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, methodCall, methodReturn, or field.⌋

[TPS_MANI_03108] Default Byte Order for all DataPrototypes ⌈The 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, methodCall, methodReturn, or field.⌋

According to [11, PRS SOME/IP Protocol], **all SOME/IP** header **fields (see [PRS_SOMEIP_00368]) and length fields in the payload of SOME/IP messages (see [PRS_SOMEIP_00759])** are encoded in network byte order (**big endian**).

[TPS_MANI_01210] Default encoding for all DataPrototypes typed by CppImplementationDataType of category STRING ⌈The 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, methodCall, methodReturn, or field.⌋

[constr_1675] Existence of attribute ApSomeipTransformationProps.stringEncoding

Imposition time: IT_BefAraApiGen

⌈The attribute TransformationPropsToServiceInterfaceElementMapping.transformationProps.stringEncoding shall only exist for a event, methodCall, methodReturn, or field (referenced by the same TransformationPropsToServiceInterfaceElementMapping) that consists of or contains a DataPrototype typed by a CppImplementationDataType of category STRING.⌋

More details about ApSomeipTransformationProps can be found in Chapter 3.4.2.

[constr_1678] Allowed values for attribute ApSomeipTransformationProps.stringEncoding

Imposition time: IT_BefAraApiGen

⌈Imposed by technical restrictions in the definition of the SOME/IP message format [11, PRS SOME/IP Protocol], 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* [12, ISO 10646]

⌋

[constr_9460] Attribute `fixedStringSize` of `ApSomeipTransformationProps` not applicable for `TransformationPropsToServiceInterfaceElement-Mapping`

Imposition time: IT_BefAraApiGen

⌈Any TransformationPropsToServiceInterfaceElementMapping shall not reference ApSomeipTransformationProps in the role transformationProps in which the attribute fixedStringSize is set to a value.⌋

[constr_10743] Existence of `TransformationPropsToServiceInterface-ElementMapping.methodReturn`

Imposition time: IT_BefAraApiGen

⌈If the ClientServerOperation referenced by a TransformationPropsTo-ServiceInterfaceElementMapping in the role methodCall has attribute fire-AndForget set to true, then the TransformationPropsToServiceInterface-ElementMapping shall **not** define a reference in the role methodReturn.⌋

Class	**ApSomeipTransformationProps**			
Note	SOME/IP serialization properties. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TransformationProps*			
Aggregated by	TransformationPropsSet.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.
byteOrder	ByteOrderEnum	0..1	attr	Specifies the byte order of data in the serialized data stream.
fixedStringSize	PositiveInteger	0..1	attr	Specifies the length of a fixed size string in bytes in the serialized data stream. The length includes the BOM, optional length field, payload and null-termination characters.
isDynamic LengthFieldSize	Boolean	0..1	attr	This attribute represents the ability to control the setting of the wire type for TLV encoding.
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.
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.
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.

▽

△

Class	**ApSomeipTransformationProps**			
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.
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.
stringEncoding	BaseTypeEncoding String	0..1	attr	Configures the encoding for SOME/IP serialization for the referenced dataPrototype in case of an String.

Table 3.55: ApSomeipTransformationProps

Enumeration	**ByteOrderEnum**
Note	When more than one byte is stored in the memory the order of those bytes may differ depending on the architecture of the processing unit. If the least significant byte is stored at the lowest address, this architecture is called little endian and otherwise it is called big endian. ByteOrder is very important in case of communication between different PUs or ECUs.
Aggregated by	ApSomeipTransformationProps.byteOrder, BaseTypeDirectDefinition.byteOrder, DiagnosticCommon Props.defaultEndianness, ISignalToIPduMapping.packingByteOrder, MultiplexedIPdu.selectorField ByteOrder, PduToFrameMapping.packingByteOrder, SegmentPosition.segmentByteOrder, SOMEIPTransformationDescription.byteOrder, System.containerIPduHeaderByteOrder
Literal	***Description***
mostSignificantByte First	Most significant byte shall come at the lowest address (also known as BigEndian or as Motorola-Format) **Tags:** atp.EnumerationLiteralIndex=0
mostSignificantByte Last	Most significant byte shall come highest address (also known as LittleEndian or as Intel-Format) **Tags:** atp.EnumerationLiteralIndex=1
opaque	For opaque data endianness conversion has to be configured to Opaque. See AUTOSAR COM Specification for more details. **Tags:** atp.EnumerationLiteralIndex=2

Table 3.56: ByteOrderEnum

Primitive	**BaseTypeEncodingString**
Note	This is the string denotion of a BaseType encoding. It may be refined by specific use-cases. **Tags:** xml.xsd.customType=BASE-TYPE-ENCODING-STRING xml.xsd.type=string

Table 3.57: BaseTypeEncodingString

[TPS_MANI_03102] UserDefined serialization ⌈The `UserDefined-TransformationProps` meta-class that is referenced by the `Transformation-PropsToServiceInterfaceElementMapping` in the role `transformation-Props` provides the ability to define a User defined serialization for `Service-Interface` elements that are referenced by the `TransformationPropsTo-ServiceInterfaceElementMapping` in the role `event`, `methodCall`, `method-Return`, or `field`.⌋

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, refer to [8, TPS Generic Structure Template].

Class	**TransformationPropsToServiceInterfaceElementMapping**			
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.recommendedPackage=TransformationPropsToServiceInterfaceElementMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
event	VariableDataPrototype	*	ref	This represents the reference to one or several events of one ServiceInterface. **Tags:** atp.Splitkey=event
field	Field	*	ref	This represents the reference to one or several fields of one ServiceInterface. **Tags:** atp.Splitkey=field
methodCall	ClientServerOperation	*	ref	This represents the reference to one or several method calls of one ServiceInterface. **Tags:** atp.Splitkey=methodCall
methodReturn	ClientServerOperation	*	ref	This represents the reference to one or several method return of one ServiceInterface. **Tags:** atp.Splitkey=methodReturn
tlvDataId Definition	TlvDataIdDefinitionSet	*	ref	This reference identifies the TlvDataIdDefinitions relevant for the enclosing TransformationPropsToServiceInterface Mapping. **Tags:** atp.Splitkey=tlvDataIdDefinition
transformation Props	TransformationProps	0..1	ref	This represents the reference to the applicable Serialization properties. **Tags:** atp.Splitkey=transformationProps
trigger	Trigger	*	ref	This represents the reference to one or several triggers of one ServiceInterface. **Tags:** atp.Splitkey=trigger

Table 3.58: TransformationPropsToServiceInterfaceElementMapping

Class	**UserDefinedTransformationProps**			
Note	The class UserDefinedTransformationProps specifies specific configuration properties of a user defined serializer.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TransformationProps*			
Aggregated by	TransformationPropsSet.transformationProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.59: UserDefinedTransformationProps

3.4.2 Individual Definition of Serialization Properties

[TPS_MANI_03109] `TransformationProps` on the level of `DataPrototype`s overwrites `TransformationProps` settings on the level of a `ServiceInterface` ⌈The fine granular modeling of `TransformationProps` on the level of `DataPrototype`s overwrites the `TransformationProps` settings defined on the level of a `ServiceInterface` described with the `TransformationPropsToServiceInterfaceElementMapping`.⌋

[constr_3361] Selective definition of serialization settings

Imposition time: `IT_BefAraApiGen`

⌈If 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 `DataPrototype`s of the `ServiceInterface` element as well.⌋

Consider the following example to illustrate [constr_3361]: Assume five composite data types (e.g., `ApplicationCompositeDataType`s) `DT_0`, `DT_1`, `DT_2`, `DT_3`, and `DT_4` consisting of `DataPrototype`s (`el_0` to `el_10`) which in turn are also typed by composite data types, see Figure 3.26.

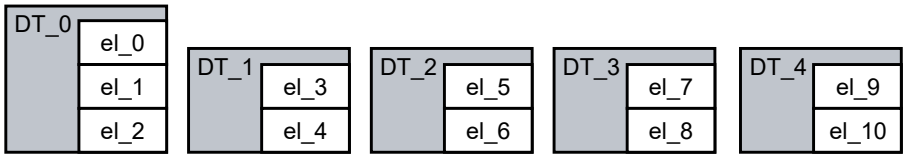


Figure 3.26: Definition of data types used in the example

Further consider a `ServiceInterface` `SI_0` consisting of two `event`s `EV_0` and `EV_1` typed by `DT_0` and `DT_1` respectively.

Additionally, `SI_0` provides two `method`s `M_0` and `M_1` where `M_0` has two `argument`s `ARG_0` and `ARG_1` typed by `DT_2` and `DT_3` and `M_1` has one `argument` `ARG_2` typed by `DT_4`. This yields the following hierarchy for `SI_0`, see Figure 3.27.

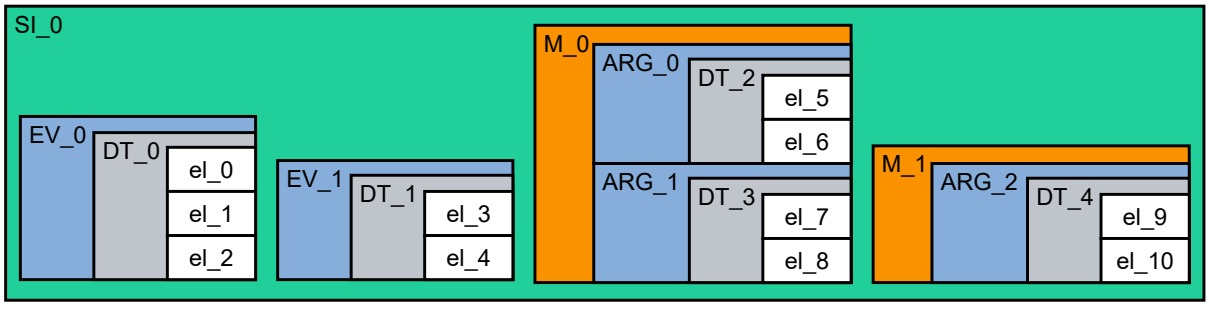


Figure 3.27: Definition of service interface used in the example

In this setup [constr_3361] would mandate that if a SomeipDataPrototype-TransformationProps were defined for el_2, then SomeipDataPrototype-TransformationProps have to be defined for el_0 and el_1 as well since el_0 and el_1 are composite DataPrototypes of the same ServiceInterface element EV_0 as el_2.

Conversely, [constr_3361], would not mandate that if a SomeipDataPrototype-TransformationProps were defined for el_2, then SomeipDataPrototype-TransformationProps have to be defined for el_3 and el_4 as well since el_3 and el_4 are composite DataPrototypes of a different ServiceInterface element (EV_1 instead of EV_0) than el_2.

Similarly for fields (since the setter and getter methods of a field are not explicitly modeled as ClientServerOperation with their input and output argument) and thus there is only the Field itself (which derives from AutosarDataPrototype) to reason about.

As far as methods are concerned, [constr_3361] would mandate the following: If in the above example a SomeipDataPrototypeTransformationProps were defined for el_5, then SomeipDataPrototypeTransformationProps have to be defined for el_6, el_7, and el_8 as well since el_6, el_7, and el_8 are composite Data-Prototypes of the same ServiceInterface element M_0 as el_5.

Conversely, [constr_3361] would not mandate that if a SomeipDataPrototype-TransformationProps were defined for el_5, then SomeipDataPrototype-TransformationProps have to be defined for el_9 and el_10 as well since el_9 and el_10 are composite DataPrototypes of a different ServiceInterface element M_1 (instead of M_0) than el_5.

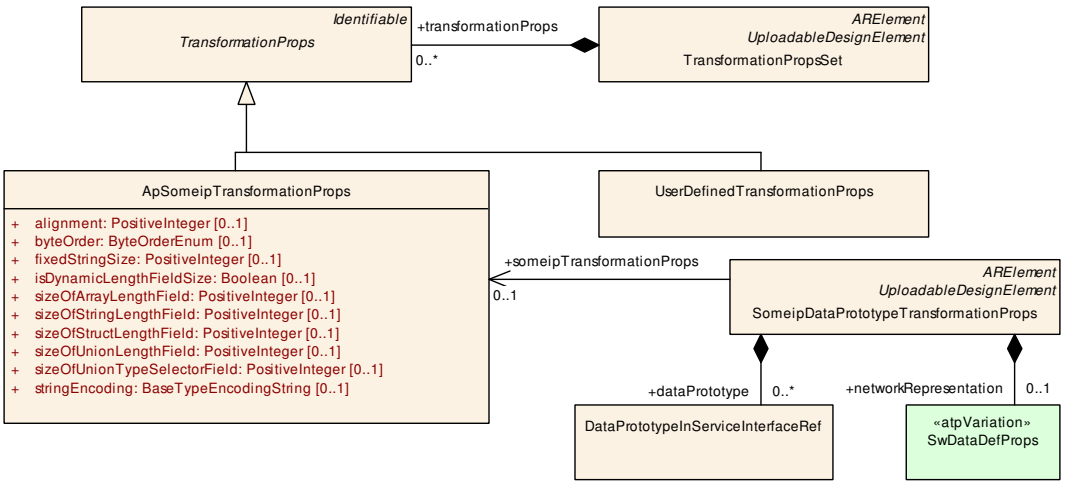


Figure 3.28: Overview about SOME/IP Serialization Properties

[TPS_MANI_03070] Size of a length field for a chosen array or map ⌈The 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 SomeipDataPrototype-

TransformationProps is defined, i.e. the variable size array (vector), fixed size array or associative_map that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.⌋

Class	TransformationPropsSet			
Note	Collection of TransformationProps. **Tags:** atp.recommendedPackage=TransformationPropsSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
transformation Props	TransformationProps	*	aggr	Transformer specific configuration properties. **Tags:** atp.Splitkey=transformationProps, transformation Props.shortName

Table 3.60: TransformationPropsSet

[constr_3353] Restriction in usage of ApSomeipTransformationProps.sizeOfArrayLengthField

Imposition time: IT_BefAraApiGen

⌈The value of the attribute sizeOfArrayLengthField shall be either 0, 1, 2 or 4.⌋

[constr_3447] ApSomeipTransformationProps.sizeOfArrayLengthField that equals 0

Imposition time: IT_BefAraApiGen

⌈The 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.⌋

The setting of sizeOfArrayLengthField for fixed size arrays supports a backward compatible extension of such arrays with additional array elements.

[TPS_MANI_03071] Size of a length field for a chosen structure ⌈The 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.⌋

[constr_3354] Restriction in usage of ApSomeipTransformationProps.sizeOfStructLengthField

Imposition time: IT_BefAraApiGen

⌈The value of the attribute sizeOfStructLengthField shall be either 0, 1, 2 or 4.⌋

[TPS_MANI_03116] Size of a length field for a chosen string ⌈The attribute sizeOfStringLengthField of ApSomeipTransformationProps defines the size of a length field generated by SOME/IP in front of a String for which the Someip-

DataPrototypeTransformationProps is defined, i.e. the String that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.⌋

[constr_3372] Restriction in usage of ApSomeipTransformationProps.sizeOfStringLengthField

Imposition time: IT_BefAraApiGen

⌈The value of the attribute sizeOfStringLengthField shall be either 0, 1, 2 or 4.⌋

[TPS_MANI_03217] On-the-wire encoding for a chosen string ⌈The attribute stringEncoding of ApSomeipTransformationProps defines the on-the-wire encoding of a String for which the SomeipDataPrototypeTransformationProps is defined, i.e. the String that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.⌋

[TPS_MANI_03072] Size of a length field for a chosen union ⌈The 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.⌋

[TPS_MANI_01532] Length of a fixed size string ⌈The attribute fixedStringSize of ApSomeipTransformationProps defines the fixed length of a string in bytes for which the SomeipDataPrototypeTransformationProps is defined, i.e. the DataPrototype typed by a CppImplementationDataType of category STRING that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.

The fixedStringSize includes the BOM, optional length field, payload and null-termination characters.⌋

[constr_9461] fixedStringSize is only applicable for DataPrototypes of category STRING

Imposition time: IT_BefAraApiGen

⌈The SomeipDataPrototypeTransformationProps shall only reference an ApSomeipTransformationProps in which the attribute fixedStringSize is set to a value if the DataPrototype that is referenced by the aggregated DataPrototypeInServiceInterfaceRef is typed by a CppImplementationDataType of category STRING.⌋

[constr_3355] Restriction in usage of ApSomeipTransformationProps.sizeOfUnionLengthField

Imposition time: IT_BefAraApiGen

⌈The value of the attribute sizeOfUnionLengthField shall be either 0, 1, 2 or 4.⌋

[TPS_MANI_03073] Alignment of a dynamic DataPrototype ⌈The 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.⌋

[constr_3356] Restriction in usage of ApSomeipTransformationProps.alignment

Imposition time: IT_BefAraApiGen

⌈The value of the attribute alignment shall be either 8, 16, 32, 64, 128, or 256.⌋

[TPS_MANI_03074] Size of a type selector field for a chosen union ⌈The 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.⌋

[constr_3357] Restriction in usage of ApSomeipTransformationProps.sizeOfUnionTypeSelectorField

Imposition time: IT_BefAraApiGen

⌈The value of the attribute sizeOfUnionTypeSelectorField shall be either 1, 2 or 4.⌋

[TPS_MANI_03278] Usage of ApSomeipTransformationProps.byteOrder ⌈The byteOrder attribute defined in an ApSomeipTransformationProps that is referenced by SomeipDataPrototypeTransformationProps shall be ignored for the dataPrototypes for which the SomeipDataPrototypeTransformationProps apply.⌋

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.

According to [11, PRS SOME/IP Protocol], **all SOME/IP** header **fields (see [PRS_SOMEIP_00368]) and length fields in the payload of SOME/IP messages (see [PRS_SOMEIP_00759])** are encoded in network byte order (**big endian**).

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 13.18.

Class	SomeipDataPrototypeTransformationProps			
Note	This meta-class represents the ability to define data transformation props specifically for a SOME/IP serialization for a given DataPrototype. **Tags:** atp.recommendedPackage=SomeipDataPrototypeTransformationPropss This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
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.Splitkey=dataPrototype
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.Splitkey=networkRepresentation
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.Splitkey=someipTransformationProps

Table 3.61: SomeipDataPrototypeTransformationProps

[TPS_MANI_01136] **`AutosarDataPrototype` is the target of the `DataPrototypeInServiceInterfaceRef`** ⌈If 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.⌋

[TPS_MANI_01137] Possible use cases for the usage of `DataPrototypeInServiceInterfaceRef` ⌈

Use case	Role
`AutosarDataPrototype` typed by an `ApplicationDataType`	`dataPrototype`
`DataPrototype` in `AutosarDataPrototype` typed by an `ApplicationCompositeDataType`	`dataPrototype`
`AutosarDataPrototype` typed by a `CppImplementationDataType`	`dataPrototype`
`DataPrototype` in `AutosarDataPrototype` typed by a `CppImplementationDataType`	`elementInImplDatatype`

⌋

From a careful observation of [TPS_MANI_01137], it should be clear that there is no valid use case to simultaneously use the two roles `dataPrototype` and `elementIn-`

ImplDatatype in the context of the same DataPrototypeInServiceInterfaceRef.

[constr_1551] Existence of DataPrototypeInServiceInterfaceRef.dataPrototype vs. DataPrototypeInServiceInterfaceRef.elementInImplDatatype

Imposition time: IT_BefAraApiGen

⌈For every given DataPrototypeInServiceInterfaceRef, either the aggregation DataPrototypeInServiceInterfaceRef.dataPrototype or DataPrototypeInServiceInterfaceRef.elementInImplDatatype shall exist.⌋

The usage of the SomeipDataPrototypeTransformationProps.networkRepresentation is explained in more detail in the System Template [4, TPS System Template] in [TPS_SYST_02136] and [TPS_SYST_02137].

3.4.3 Assignment of TLV properties

3.4.3.1 Assignment of TLV Data IDs

[TPS_MANI_01097] Assignment of TLV data IDs ⌈The assignment of TLV data IDs is done in the context of the specification of TransformationPropsToServiceInterfaceElementMapping, namely by means of the attribute TransformationPropsToServiceInterfaceElementMapping.tlvDataIdDefinition.id.⌋

This approach benefits 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.

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 [11, PRS SOME/IP Protocol]) is the motivation for the existence of [constr_1594], and [constr_1595].

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] Reference from TransformationPropsToServiceInterfaceElementMapping to TlvDataIdDefinitionSet ⌈The reference from TransformationPropsToServiceInterfaceElementMapping to TlvDataIdDefinitionSet means that it is up to 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.⌋

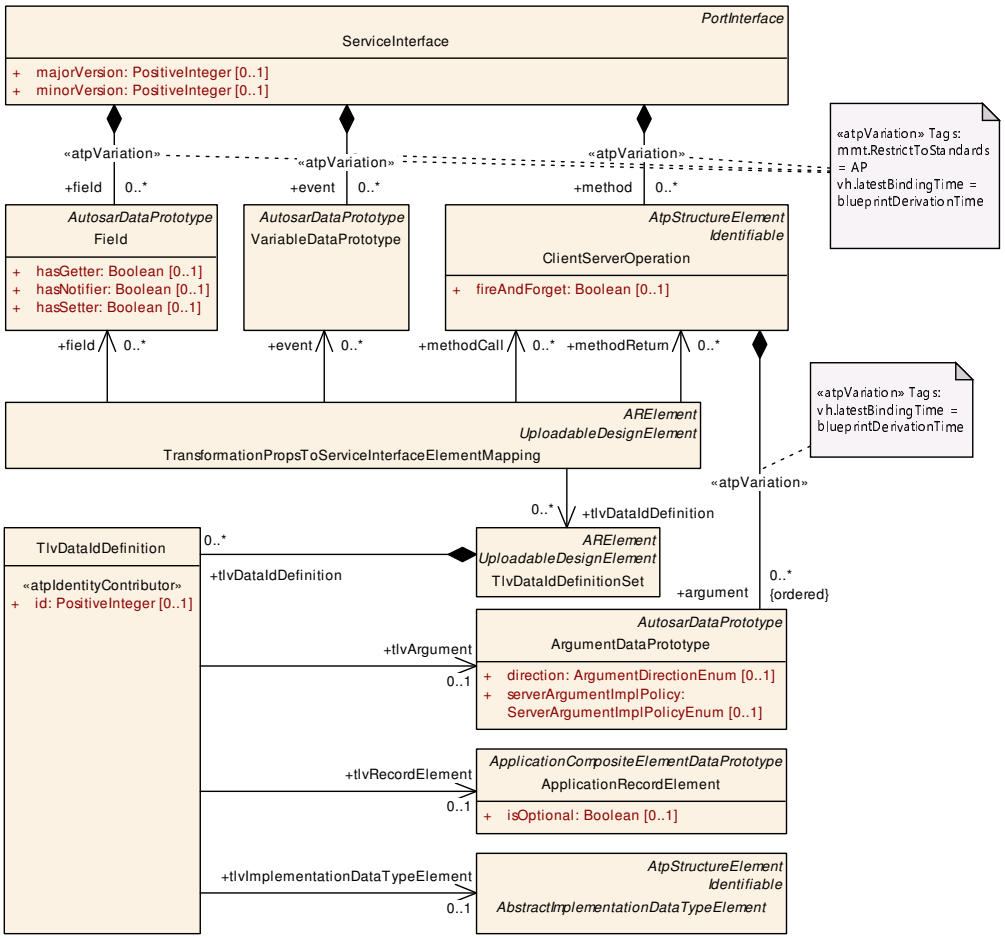


Figure 3.29: Modeling of the TLV data ID

[constr_1594] Consistent assignment of TLV data IDs to ApplicationRecordDataType

Imposition time: IT_BefAraApiGen

⌈For 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.⌋

[constr_1595] Consistent assignment of TLV data IDs to CppImplementationDataType or CppImplementationDataTypeElement

Imposition time: IT_BefAraApiGen

⌈For every CppImplementationDataType of category STRUCTURE 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.⌋

Class	TlvDataIdDefinitionSet			
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*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
tlvDataId Definition	TlvDataIdDefinition	*	aggr	This aggregation represents the collection of TlVDataTid Definitions aggregated by the TlvDataIdDefinitionSet **Tags:** atp.Splitkey=tlvDataIdDefinition, tlvDataId Definition.id

Table 3.62: TlvDataIdDefinitionSet

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 `TlvDataIdDefinition`s to all other arguments or sub-elements as well.

[constr_1593] Completeness of the existence of a set of `TlvDataIdDefinition.tlvArgument`s

Imposition time: `IT_BefAraApiGen`

⌈If the reference `TlvDataIdDefinition.tlvArgument` exists for one `argument` of a given `ClientServerOperation` then further `TlvDataIdDefinition.tlvArgument` shall exist **for all** `argument`s of the given `ClientServerOperation` and all affected `TlvDataIdDefinition` shall be aggregated by the same `TransformationPropsToServiceInterfaceElementMapping`.⌋

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[10], AUTOSAR defines that such a mixed TLV configuration shall not be used.

[constr_1603] Completeness of the existence of a set of `TlvDataIdDefinition.tlvRecordElement`s

Imposition time: `IT_BefAraApiGen`

⌈If the reference `TlvDataIdDefinition.tlvRecordElement` exists for one `element` of a given `ApplicationRecordDataType` then further `TlvDataIdDefinition.tlvRecordElement` shall exist **for all** `element`s of the given `ApplicationRecordDataType` and all affected `TlvDataIdDefinition` shall be aggregated by the same `TransformationPropsToServiceInterfaceElementMapping`.⌋

[10] For example, the usage of TLV encoding for `argument`s with attribute `direction` set to `in`, but **not** for `argument`s with attribute `direction` set to `out`.

[constr_1604] Completeness of the existence of a set of TlvDataIdDefinition.tlvImplementationDataTypeElements

Imposition time: IT_BefAraApiGen

⌈If 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 TlvDataIdDefinitionSet.⌋

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.

Class	**TlvDataIdDefinition**			
Note	This meta-class represents the ability to define the tlvDataId.			
Base	*ARObject*			
Aggregated by	TlvDataIdDefinitionSet.tlvDataIdDefinition			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
id	PositiveInteger	0..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. **Tags:** atp.Splitkey=tlvArgument
tlv Implementation DataType Element	AbstractImplementation DataTypeElement	0..1	ref	This reference associates the definition of a TLV data id with a given AbstractImplementationDataTypeElement. **Tags:** atp.Splitkey=tlvImplementationDataTypeElement
tlvRecord Element	ApplicationRecord Element	0..1	ref	This reference associates the definition of a TLV data id with a given ApplicationRecordElement. **Tags:** atp.Splitkey=tlvRecordElement

Table 3.63: TlvDataIdDefinition

[constr_1596] Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to ArgumentDataPrototype

Imposition time: IT_BefAraApiGen

⌈For all TlvDataIdDefinition 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.⌋

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] Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to ApplicationRecordElement

Imposition time: IT_BefAraApiGen

⌈For 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.⌋

[constr_1598] Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to CppImplementationDataTypeElement

Imposition time: IT_BefAraApiGen

⌈For 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.⌋

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] TlvDataIdDefinition referencing ArgumentDataPrototype

Imposition time: IT_BefAraApiGen

⌈Each ArgumentDataPrototype shall be referenced **at most once** in the role tlvArgument in the context of the same TransformationPropsToServiceInterfaceElementMapping.⌋

[constr_1600] TlvDataIdDefinition referencing ApplicationRecordElement

Imposition time: IT_BefAraApiGen

⌈Each ApplicationRecordElement shall be referenced **at most once** in the role tlvRecordElement in the context of the same TransformationPropsToServiceInterfaceElementMapping.⌋

[constr_1601] `TlvDataIdDefinition` referencing `CppImplementationDataTypeElement`

Imposition time: IT_BefAraApiGen

⌈Each CppImplementationDataTypeElement shall be referenced **at most once** in the role tlvImplementationDataTypeElement in the context of the same TransformationPropsToServiceInterfaceElementMapping.⌋

[constr_1748] Existence of references `TlvDataIdDefinition.tlvArgument`, `TlvDataIdDefinition.tlvRecordElement`, and `TlvDataIdDefinition.tlvImplementationDataTypeElement`

Imposition time: IT_BefAraApiGen

⌈For 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.

⌋

Class	***AbstractImplementationDataTypeElement*** (abstract)			
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			
Aggregated by	*AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.64: AbstractImplementationDataTypeElement

[constr_1628] Definition of static length field sizes in the case of TLV usage

Imposition time: IT_BefAraApiGen

⌈If the aggregation tlvDataIdDefinition exists for a given TransformationPropsToServiceInterfaceElementMapping then attributes

- sizeOfArrayLengthField,
- sizeOfStringLengthField,
- sizeOfStructLengthField, and
- sizeOfUnionLengthField

⌋

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, refer to [TPS_MANI_01186]) then the length field size shall be statically configured.
- If `wire type`s 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. it could become impossible for the de-serializer to figure out the data layout[11].

[constr_1629] Identical sizes of length fields in the case of TLV usage

Imposition time: `IT_BefAraApiGen`

⌈If the aggregation `tlvDataIdDefinition` exists for a given `TransformationPropsToServiceInterfaceElementMapping` then attributes

- `sizeOfArrayLengthField`,
- `sizeOfStringLengthField`,
- `sizeOfStructLengthField`, and
- `sizeOfUnionLengthField`

⌋

Rationale for the existence of [constr_1629]: if `wire type` 4 is used (for more details, 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] No definition of length field sizes on `DataPrototype` level in the case of TLV usage

Imposition time: `IT_BefAraApiGen`

⌈If the reference in the role `tlvDataIdDefinition` exists for a given `TransformationPropsToServiceInterfaceElementMapping` then attributes

- `sizeOfArrayLengthField`,
- `sizeOfStringLengthField`,
- `sizeOfStructLengthField`, and
- `sizeOfUnionLengthField`

[11] 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.

shall not be individually defined on the level of a DataPrototype (i.e. by means of the reference SomeipDataPrototypeTransformationProps.someip-TransformationProps) but only on the level of a ServiceInterface (i.e. by means of the reference TransformationPropsToServiceInterfaceElement-Mapping.transformationProps).⌋

Rationale for the existence of [constr_1630]: if wire type 4 is used (for more details, 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.

3.4.3.2 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 [11, PRS SOME/IP Protocol].

[TPS_MANI_01186] Definition of the applicable wire type ⌈Attribute ApSomeip-TransformationProps.isDynamicLengthFieldSize shall be used to define the applicable wire type.

If the value of attribute ApSomeipTransformationProps.isDynamicLength-FieldSize 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.⌋

3.5 Identity and Access Management

The definition of intents (for example: ClientComSpec.clientIntent) as described in Chapter 3.2.5.1.3 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 12.9.

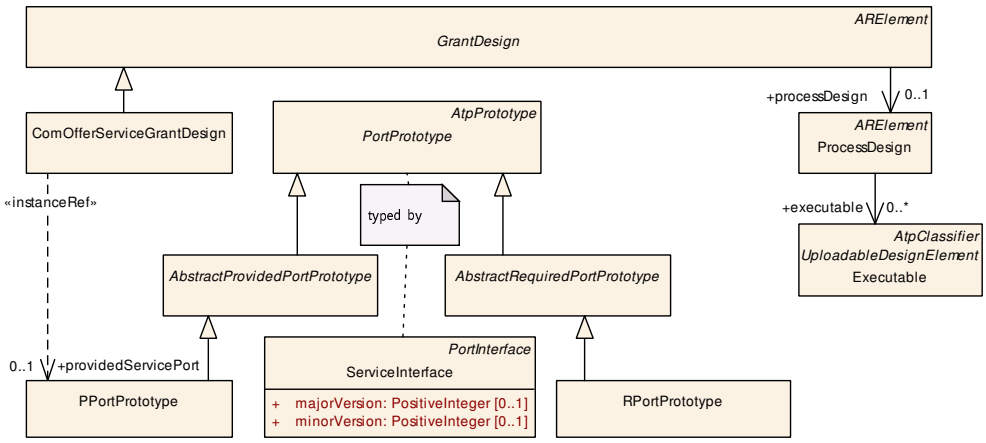


Figure 3.30: Modeling of grant designs for service discovery

Class	***GrantDesign*** (abstract)			
Note	This meta-class serves as an abstract base class for the description of grants on design level. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	*ComGrantDesign*, ComOfferServiceGrantDesign			
Aggregated by	ARPackage.element			
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.Splitkey=processDesign

Table 3.65: GrantDesign

Abstract meta-class GrantDesign acts as the base class for the definition of grants on the design level.

Grants are specific to 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] **GrantDesign references ProcessDesign** ⌈Meta-class GrantDesign references ProcessDesign as a means to design the set of Grants for the given Process.⌋

3.5.1 Com Grant Design

Subclasses of GrantDesign are created to cover specific aspects of grants for communication on the *AUTOSAR adaptive Platform*.

[TPS_MANI_01232] Semantics of meta-class `ComOfferServiceGrantDesign`
⌈The existence of a `ComOfferServiceGrantDesign` that references a specific `PPortPrototype` in the role `providedServicePort` indicates that the design foresees that the referenced `PPortPrototype` shall be granted rights to offer the respective service.⌋

There is no explicitly modeled intent that corresponds to the existence of the `ComOfferServiceGrantDesign`. The understanding is that

- the mere existence of a `PPortPrototype` typed by a `ServiceInterface` indicates the intent to offer a service.
- the mere existence of an `RPortPrototype` typed by a `ServiceInterface` indicates the intent to find a service.

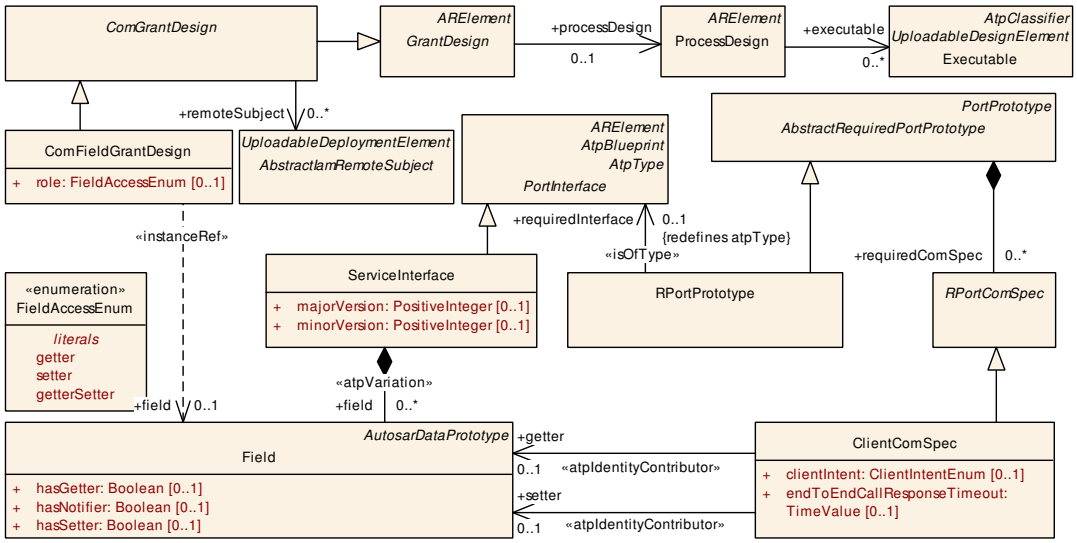


Figure 3.31: Modeling of grant designs for `field`

Class	**ComOfferServiceGrantDesign**			
Note	This meta-class represents the ability to define a Grant for offering a service. **Tags:** atp.recommendedPackage=GrantDesigns This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
providedServicePort	PPortPrototype	0..1	iref	This instanceRef identifies the PPortPrototype on which the service shall be offered. **Tags:** atp.Splitkey=providedServicePort, providedServicePort.contextRootSwComponentPrototype, providedServicePort.contextComponentPrototype, providedServicePort.targetPPortPrototype **InstanceRef implemented by:** PPortPrototypeInExecutableInstanceRef

Table 3.66: ComOfferServiceGrantDesign

[TPS_MANI_01234] Semantics of `ComFieldGrantDesign` ⌈The 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.⌋

[constr_10155] Multiplicity of reference in the role `ComOfferServiceGrantDesign.providedServicePort`

Imposition time: `IT_GraDes`

⌈For each `ComOfferServiceGrantDesign`, the reference in the role `providedServicePort` shall exist.⌋

Class	***ComGrantDesign*** (abstract)			
Note	This meta-class serves as an abstract base class for the description of com grants on design level. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	ComEventGrantDesign, ComFieldGrantDesign, ComMethodGrantDesign, ComTriggerGrantDesign			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
remoteSubject	AbstractIamRemote Subject	*	ref	This optional reference defines the remoteSubject that is allowed to access the defined Object via the Grant. **Tags:** atp.Splitkey=remoteSubject

Table 3.67: ComGrantDesign

[constr_3720] Upper multiplicity of reference in the role `ComGrantDesign.remoteSubject`

Imposition time: `IT_GraDes`

⌈In the context of `ComGrantDesign`, the reference in the role `remoteSubject` shall exist at most once.⌋

Class	**ComFieldGrantDesign**			
Note	This meta-class represents the ability to define a Grant for a ServiceInterface.field. **Tags:** atp.recommendedPackage=GrantDesigns This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrantDesign*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ComFieldGrantDesign**			
field	Field	0..1	iref	Reference to the affected Field in the context of an Executable. **Tags:** atp.Splitkey=field, field.contextRootSwComponent Prototype, field.contextSwComponentPrototype, field.contextRPortPrototype, field.targetField **InstanceRef implemented by:** FieldInExecutable InstanceRef
role	FieldAccessEnum	0..1	attr	This attribute provides the ability to further specify the access to the ServiceInterface.field from a design perspective.

Table 3.68: ComFieldGrantDesign

[constr_10157] Multiplicity of reference in the role `ComFieldGrantDesign.field`

Imposition time: `IT_GraDes`

⌈For each `ComFieldGrantDesign`, the reference in the role `field` shall exist.⌋

[constr_10158] Multiplicity of reference in the role `ComFieldGrantDesign.role`

Imposition time: `IT_GraDes`

⌈For each `ComFieldGrantDesign`, the reference in the role `role` shall exist.⌋

Enumeration	**FieldAccessEnum**
Note	This meta-class provides values that qualify access to a field. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	ComFieldGrant.role, ComFieldGrantDesign.role
Literal	***Description***
getter	Access to the getter of the Field. **Tags:** atp.EnumerationLiteralIndex=0
getterSetter	Access to getter and setter of the field **Tags:** atp.EnumerationLiteralIndex=2
setter	Access to the setter of the Field. **Tags:** atp.EnumerationLiteralIndex=1

Table 3.69: FieldAccessEnum

[TPS_MANI_01235] Semantics of `ComEventGrantDesign` ⌈The 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`.⌋

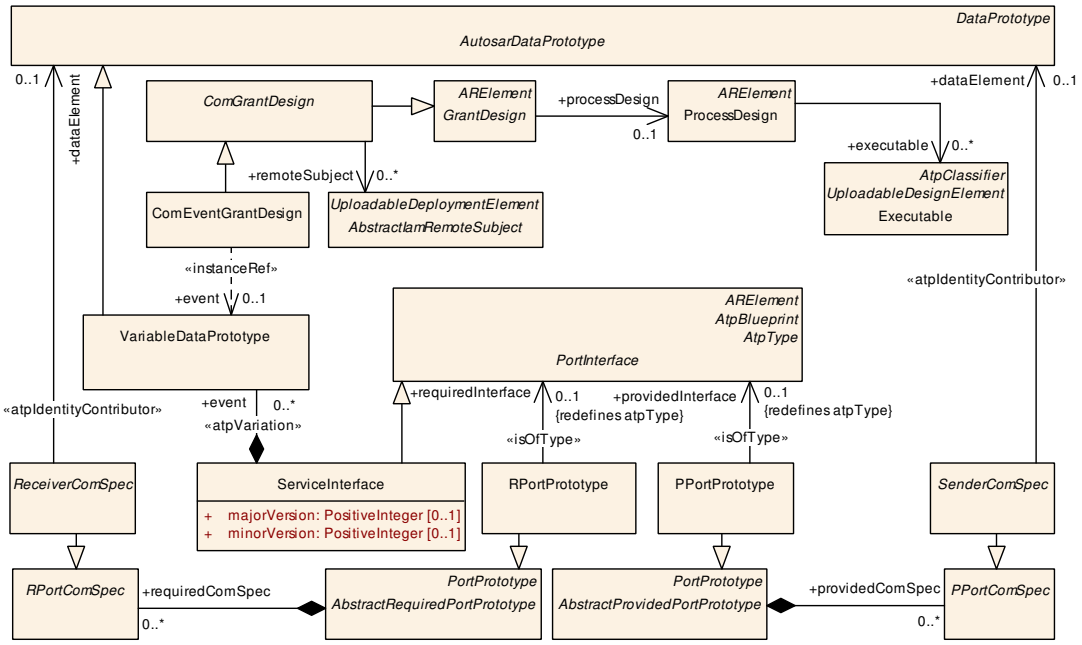


Figure 3.32: Modeling of grant designs for `event`

[TPS_MANI_03290] Semantics of `ComTriggerGrantDesign` ⌈The 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`.⌋

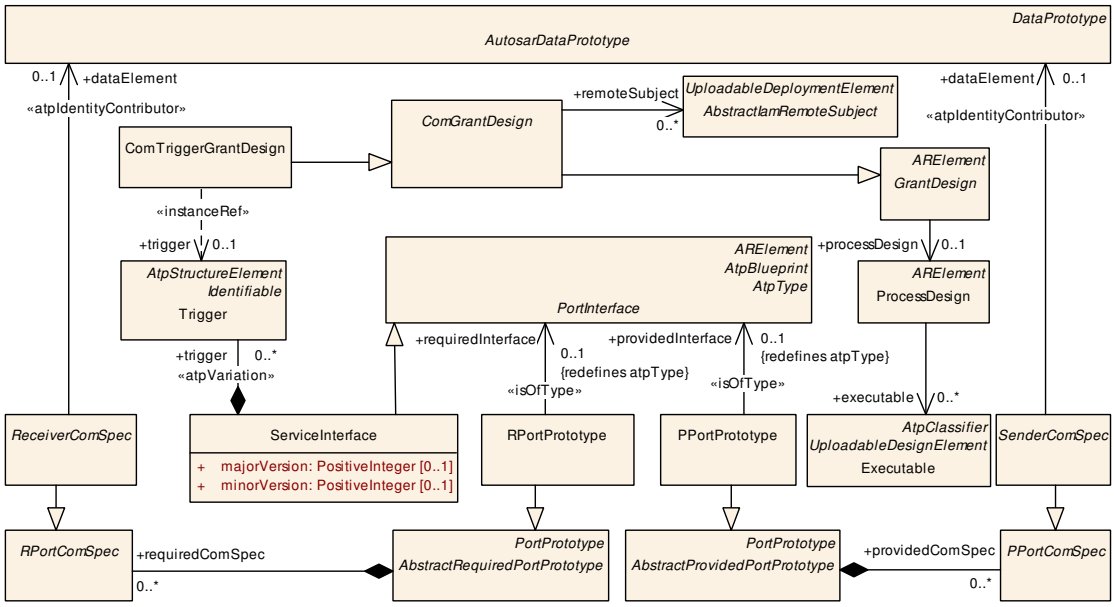


Figure 3.33: Modeling of grant designs for `trigger`

Class	ComEventGrantDesign			
Note	This meta-class represents the ability to define a Grant for a ServiceInterface.event. **Tags:** atp.recommendedPackage=GrantDesigns This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrantDesign*, *GrantDesign*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
event	VariableDataPrototype	0..1	iref	This reference represents the affected event. **Tags:** atp.Splitkey=event, event.contextRootSw ComponentPrototype, event.contextSwComponent Prototype, event.contextRPortPrototype, event.target Event **InstanceRef implemented by:** EventInExecutable InstanceRef

Table 3.70: ComEventGrantDesign

[constr_10159] Multiplicity of reference in the role `ComEventGrantDesign.event`

Imposition time: `IT_GraDes`

⌈For each `ComEventGrantDesign`, the reference in the role `event` shall exist.⌋

Class	ComTriggerGrantDesign			
Note	This meta-class represents the ability to define a Grant for a ServiceInterface.trigger. **Tags:** atp.recommendedPackage=GrantDesigns This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrantDesign*, *GrantDesign*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
trigger	Trigger	0..1	iref	This reference represents the affected trigger. **Tags:** atp.Splitkey=trigger, trigger.contextComponent Prototype, trigger.contextRootSwComponentPrototype, trigger.contextRPortPrototype, trigger.targetTrigger **InstanceRef implemented by:** TriggerInExecutable InstanceRef

Table 3.71: ComTriggerGrantDesign

[constr_10160] Multiplicity of reference in the role `ComTriggerGrantDesign.trigger`

Imposition time: `IT_GraDes`

⌈For each `ComTriggerGrantDesign`, the reference in the role `trigger` shall exist.⌋

[TPS_MANI_01236] Semantics of `ComMethodGrantDesign` ⌈The 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`.⌋

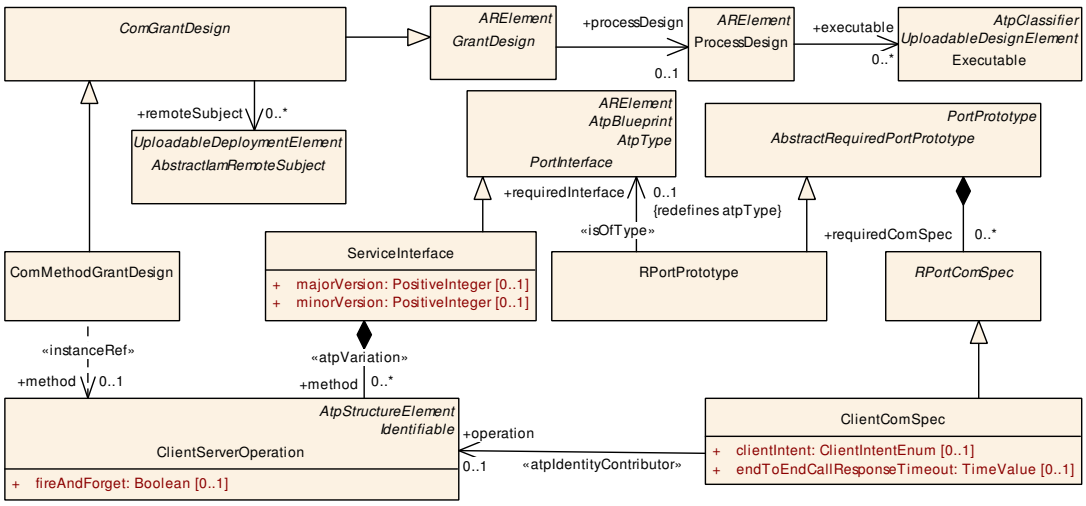


Figure 3.34: Modeling of grant designs for `method`

Class	**ComMethodGrantDesign**			
Note	This meta-class represents the ability to define a Grant for a ServiceInterface.method. **Tags:** atp.recommendedPackage=GrantDesigns This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrantDesign*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
method	ClientServerOperation	0..1	iref	This reference identifies the corresponding method. **Tags:** atp.Splitkey=method, method.contextRootSw ComponentPrototype, method.contextComponent Prototype, method.contextRPortPrototype, method.target Method **InstanceRef implemented by:** RequiredMethodIn ExecutableInstanceRef

Table 3.72: ComMethodGrantDesign

[constr_10161] Multiplicity of reference in the role `ComMethodGrantDesign.method`

Imposition time: `IT_GraDes`

⌈For each `ComMethodGrantDesign`, the reference in the role `method` shall exist.⌋

3.5.2 Remote access control

The definition of the target-configuration 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 `Machine`s that are allowed to call this `method`.

The fact that the Machine on which the service is running has no means to make additional checks on the incoming requests enables the processing of wrongly issued requests by a healthy remote Machine as well as the escalation of privileges by an attacker via issuing arbitrary request towards services from a compromised remote Machine.

In most cases 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 a remote Machine cannot go beyond what it could request in a healthy state.

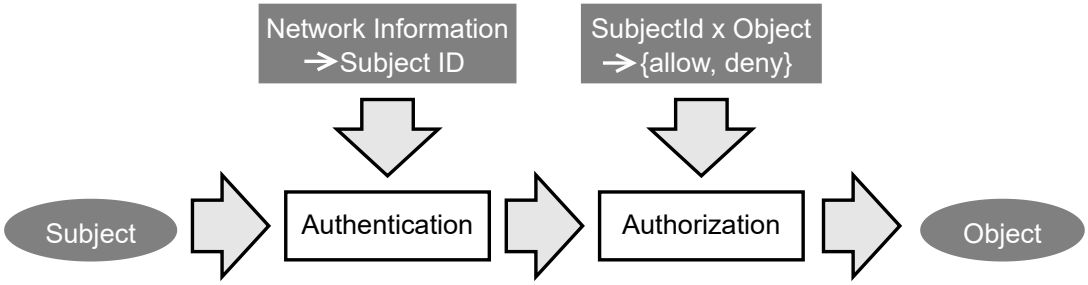


Figure 3.35: Access policy enforcement based on the Subject ID from the network binding

The access control process aims at enforcing policies on the relation between a "Subject" and an "Object". In the example where a 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 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 against the rules and policies defined for 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 policies defined for requests coming from that channel, as depicted in Figure 3.35.

The remote subject is modeled as a specialization of AbstractIamRemoteSubject. The different specializations will be presented in the following sections.

Class	**AbstractIamRemoteSubject** (abstract)			
Note	This abstract meta-class defines the proxy information about the remote node. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	IPSecIamRemoteSubject, IpIamRemoteSubject, TlsIamRemoteSubject			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.73: 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] Definition of ComMethodGrantDesign.remoteSubject ⌈If 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.⌋

[TPS_MANI_03239] Definition of ComEventGrantDesign.remoteSubject ⌈If 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.⌋

[TPS_MANI_03251] Definition of ComFieldGrantDesign.remoteSubject ⌈If 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.⌋

3.5.2.1 Remote Subject in the Case of TLS

This chapter defines how a AbstractIamRemoteSubject is modeled in the case of a TLS-based secure channel.

[TPS_MANI_03240] Modeling of a remote peer in the case of TLS-based secure channel ⌈In the case of a 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.

⌋

The security of a pre-shared key as authentication in the 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 establish 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 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.

In that case, the acceptedRemoteCertificates define the set of allowed root certificates for the certificate presented by the TlsIamRemoteSubject.

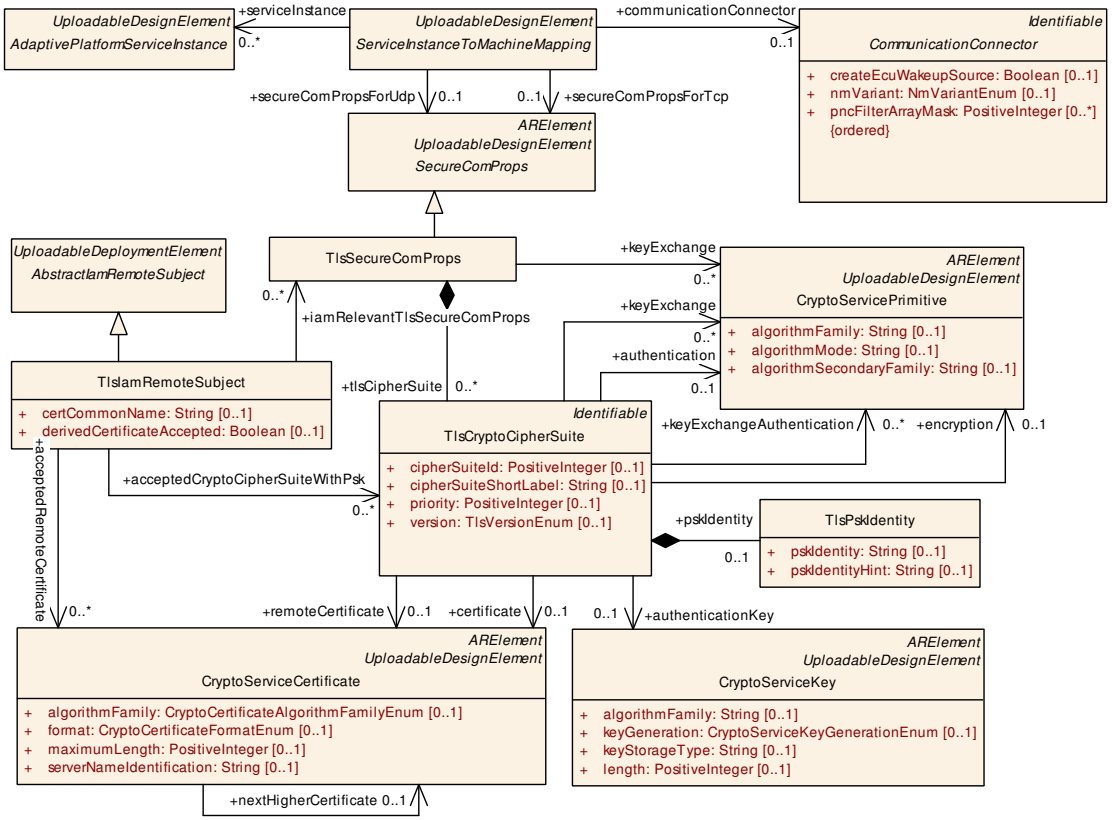


Figure 3.36: Proxy information about the remote node in the case of TLS

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] Modeling of relevant TlsSecureComProps for TlsIamRemoteSubject ⌈With the TlsIamRemoteSubject.iamRelevantTlsSecureComProps reference it is possible to define all TlsSecureComProps that the TlsIamRemoteSubject supports to establish a secure channel.⌋

Class	**TlsIamRemoteSubject**			
Note	This meta-class defines the proxy information about the remote node in case of TLS. **Tags:** atp.recommendedPackage=IamRemoteSubjects This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractIamRemoteSubject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
acceptedCryptoCipherSuiteWithPsk	TlsCryptoCipherSuite	*	ref	This reference is used to identify a remote node by means of the preshared Key. **Tags:** atp.Splitkey=acceptedCryptoCipherSuiteWithPsk
acceptedRemoteCertificate	CryptoServiceCertificate	*	ref	This reference is used to identify a remote node by means of the certificate. **Tags:** atp.Splitkey=acceptedRemoteCertificate
certCommonName	String	0..1	attr	This attribute defines the common name (CN) of the certificate of the remote peer.
derivedCertificateAccepted	Boolean	0..1	attr	This attribute defines whether a derivedCertificate is accepted (true) or not (false).
iamRelevantTlsSecureComProps	TlsSecureComProps	*	ref	This reference defines the local TlsSecureComProps that are relevant for IAM. **Tags:** atp.Splitkey=iamRelevantTlsSecureComProps

Table 3.74: TlsIamRemoteSubject

3.5.2.2 Remote Subject in the Case of IPsec

This chapter defines how a AbstractIamRemoteSubject is modeled in the case of an IPsec-based secure channel.

[TPS_MANI_03242] Modeling of a remote peer in the case of IPsec-based secure channel ⌈In the case of an 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.⌋

Class	**IPSeclamRemoteSubject**			
Note	This meta-class defines the proxy information about the remote node in case of IPsec. **Tags:** atp.recommendedPackage=IamRemoteSubjects This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractIamRemoteSubject*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackage Element*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
localIpSecRule	IPSecRule	*	ref	This reference is used to describe theRemoteSubjects local IPSecRules. **Tags:** atp.Splitkey=localIpSecRule

Table 3.75: IPSeclamRemoteSubject

The local IP address of the remote peer is defined by the `NetworkEndpoint` that aggregates the `IPSecRule`s.

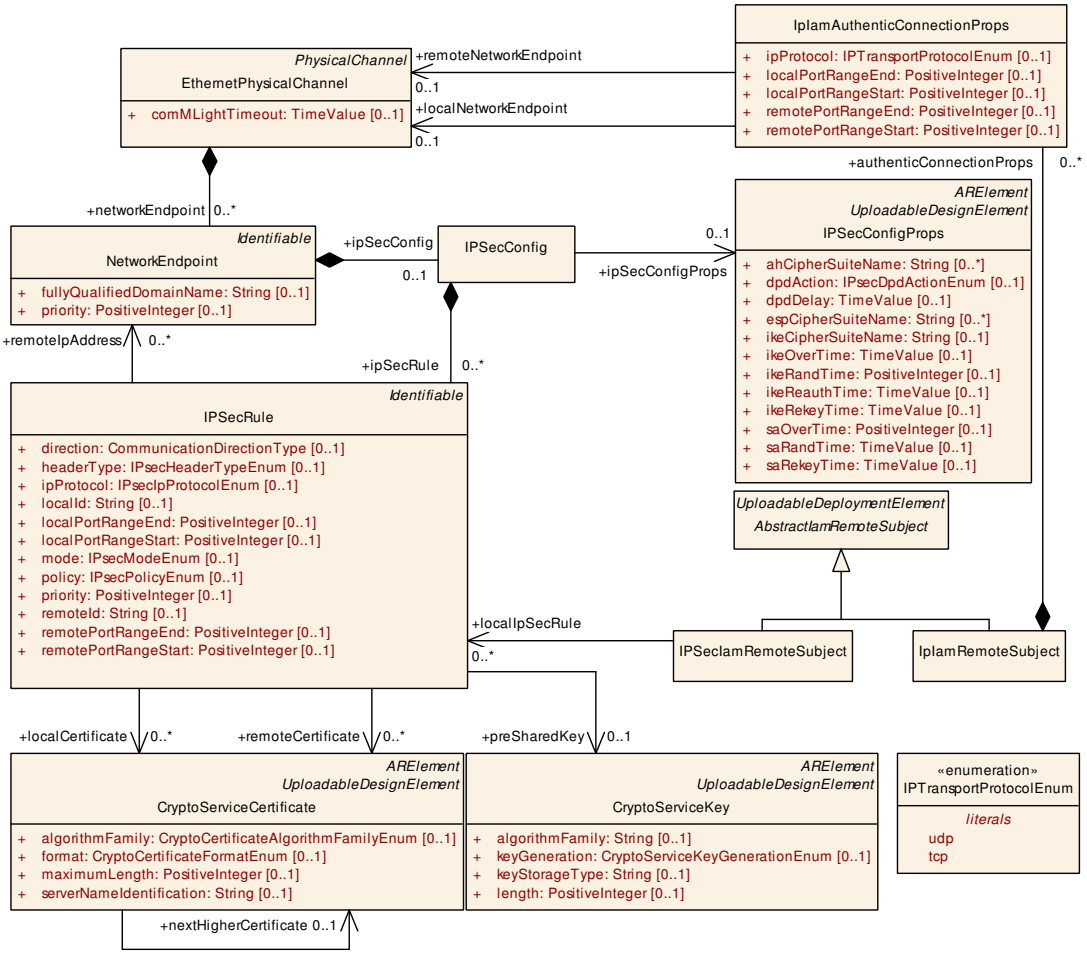


Figure 3.37: Proxy information about the remote node in the case of IPsec

3.5.2.3 Remote Subject in the Case of IP Communication

It is possible to define an `AbstractIamRemoteSubject` that is based on general IP communication. In this case, no details about how the communication is secured are given and actually securing the communication is not part of the model.

Trust can be added to the IP communication, e.g., by using a secure communication layer protocol below the network layer (e.g. MACsec), hardware mechanisms and/or by appropriate network and switch design.

An `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] Modeling of a remote peer in the case of general IP communication ⌈In the case of 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.⌋

Class	**IpIamRemoteSubject**			
Note	This meta-class defines the proxy information about the remote node in case of general IP communication. **Tags:** atp.recommendedPackage=IamRemoteSubjects This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractIamRemoteSubject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authentic ConnectionProps	IpIamAuthenticConnectionProps	*	aggr	Definition of IP rules assigned to the IpIamRemoteSubject. **Tags:** atp.Splitkey=authenticConnectionProps

Table 3.76: IpIamRemoteSubject

Enumeration	**IPTransportProtocolEnum**
Note	Transport Layer protocols in the Internet protocol suite. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	IpIamAuthenticConnectionProps.ipProtocol
Literal	***Description***
tcp	Transmission Control Protocol (TCP) **Tags:** atp.EnumerationLiteralIndex=1
udp	User Datagram Protocol (UDP). **Tags:** atp.EnumerationLiteralIndex=0

Table 3.77: IPTransportProtocolEnum

Class	**IplamAuthenticConnectionProps**			
Note	This meta-class defines a set of properties for IP connections in the context of IAM configuration. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	IplamRemoteSubject.authenticConnectionProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ipProtocol	IPTransportProtocol Enum	0..1	attr	This attribute defines the relevant IP protocol.
localNetwork Endpoint	EthernetPhysical Channel	0..1	ref	This reference defines an authentic local Network Endpoint in terms of IAM configuration. **Tags:** atp.Splitkey=localNetworkEndpoint
localPortRange End	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines an end value for the local port range.
localPortRange Start	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines a start value for the local port range.
remoteNetwork Endpoint	EthernetPhysical Channel	0..1	ref	This reference defines an authentic remote Network Endpoint in terms of IAM configuration. **Tags:** atp.Splitkey=remoteNetworkEndpoint
remotePort RangeEnd	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines an end value for the remote port range.
remotePort RangeStart	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines a start value for the remote port range.

Table 3.78: IplamAuthenticConnectionProps

3.5.2.4 Remote Subject in the Case of SecOC Communication

The identity information in the case of SecOC depends on the group of `Machine`s that are sharing the same cryptographic key.

In other words, if a valid SecOC message is received with a given key, only remote `Machine`s that "know the key" can have sent 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, it is dropped. Therefore, access control between the remote subject and the local object takes place.

To summarize, modeling a remote subject in the case of SecOC provides no additional benefit, neither by increasing the granularity of the subject identification nor by providing new enforcement of rules on the object.

3.6 Intrusion Detection Design

[TPS_MANI_01338] Semantics of `SecurityEventReportToSecurityEventDefinitionMapping` ⌈The modeling of the association between a specific security event and the corresponding `RPortPrototype` typed by a `SecurityEventReportInterface` is created by means of the `SecurityEventReportToSecurityEventDefinitionMapping`.⌋

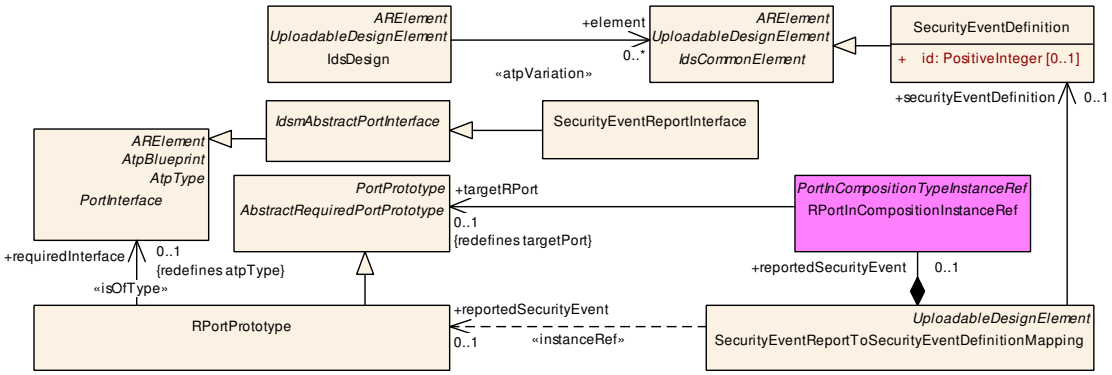


Figure 3.38: Specification of meta-class `SecurityEventReportToSecurityEventDefinitionMapping`

This meta-class maps the `RPortPrototype` to a `SecurityEventDefinition` that itself is part of the so-called `Security Extract`.

[TPS_MANI_01339] Existence of the `SecurityEventReportToSecurityEventDefinitionMapping` is motivated by the AUTOSAR methodology ⌈The existence of the `SecurityEventReportToSecurityEventDefinitionMapping` is motivated by the AUTOSAR methodology. At the point in time when a given `SecurityEventReportInterface` is defined, the corresponding `SecurityEventDefinition` might not yet be defined.

It is therefore 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 target-configuration of the application software.

The semantics of the security event is always either identical or at least comparable. Nevertheless, 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.⌋

Class	**SecurityEventReportToSecurityEventDefinitionMapping**			
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.recommendedPackage=SecurityEventReportToSecurityEventDefinitionMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**SecurityEventReportToSecurityEventDefinitionMapping**			
reported SecurityEvent	AbstractRequiredPort Prototype	0..1	iref	This identifies the mapped security event. **Tags:** atp.Splitkey=reportedSecurityEvent, reported SecurityEvent.contextComponent, reportedSecurity Event.targetRPort **InstanceRef implemented by:** RPortInComposition InstanceRef
securityEvent Definition	SecurityEventDefinition	0..1	ref	This reference identifies the definition of the security event. **Tags:** atp.Splitkey=securityEventDefinition

Table 3.79: SecurityEventReportToSecurityEventDefinitionMapping

3.7 Relation to Crypto System Design

It is possible to define the relation of crypto-related model elements on application design level to the crypto-related model on system design level (see Chapter 7.5). For this purpose, the classes `CryptoKeySlotUserDesignMapping` and `CryptoKeySlotUsageDesignMapping` exist.

A possible workflow is that the application-level design elements are created based on the existence of the corresponding crypto-related model on system design level.

[TPS_MANI_01468] Semantics of meta-class `CryptoKeySlotUserDesignMapping` ⌈Meta-class `CryptoKeySlotUserDesignMapping` creates a “tracing” relation between

- an `Executable` as the source end of the “tracing” relation and
- a `CryptoKeySlotUserDesign` as the target end of the “tracing” relation.

The semantics of the “tracing” relation is that the `Executable` traces to the `CryptoKeySlotUserDesign` as the model element from which the `Executable` was derived.⌋

Class	**CryptoKeySlotUserDesignMapping**			
Note	This meta-class represents the ability to map a CryptoKeySlotUserDesign (on the system design level) to a Executable (on the application design level). **Tags:** atp.recommendedPackage=CryptoKeySlotDesignMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**CryptoKeySlotUserDesignMapping**			
cryptoKeySlot UserDesign	CryptoKeySlotUser Design	0..1	ref	This reference identifies the applicable CryptoKeySlot UserDesign on system design level. **Tags:** atp.Splitkey=cryptoKeySlotUserDesign
processDesign	ProcessDesign	0..1	ref	This reference identifies the applicable ProcessDesign. **Tags:** atp.Splitkey=processDesign

Table 3.80: CryptoKeySlotUserDesignMapping

[advisory_01011] Existence of the reference in the role CryptoKeySlotUserDesignMapping.processDesign

Imposition time: IT_ProDes

⌈For each CryptoKeySlotUserDesignMapping, the reference in the role processDesign should exist.⌋

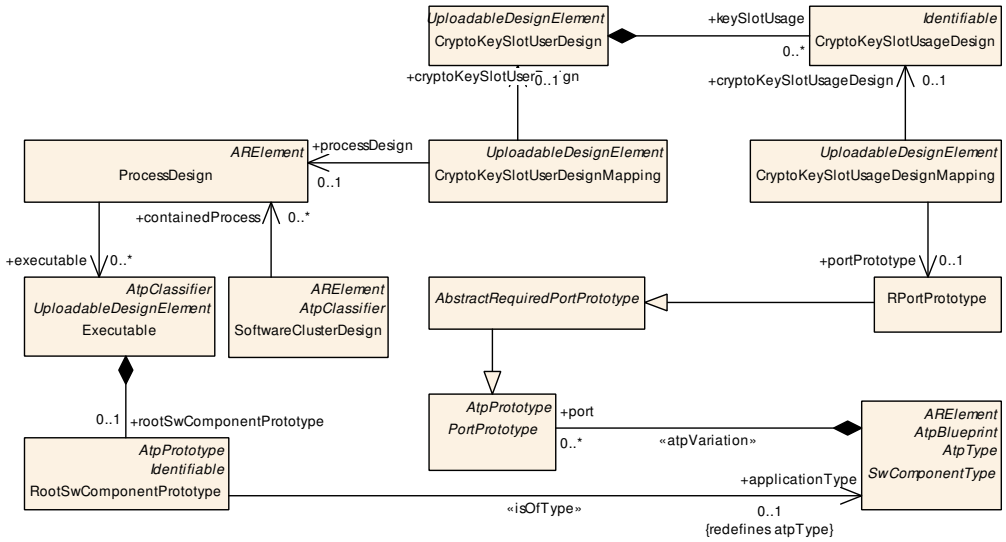


Figure 3.39: Modeling of meta-class CryptoKeySlotUserDesign and CryptoKeySlotUsageDesign

[advisory_01012] Existence of the reference in the role CryptoKeySlotUserDesignMapping.cryptoKeySlotUserDesign

Imposition time: IT_ProDes

⌈For each CryptoKeySlotUserDesignMapping, the reference in the role cryptoKeySlotUserDesign should exist.⌋

[TPS_MANI_01469] Semantics of meta-class CryptoKeySlotUsageDesignMapping

⌈Meta-class CryptoKeySlotUsageDesignMapping creates a "tracing" relation between

- an RPortPrototype as the source end of the "tracing" relation and
- a CryptoKeySlotUsageDesign as the target end of the "tracing" relation.

The semantics of the "tracing" relation is that the RPortPrototype traces to the CryptoKeySlotUsageDesign as the model element from which the RPort-Prototype was derived.⌋

The existence of the "tracing" semantics between RPortPrototype and Crypto-KeySlotUsageDesign, as described in [TPS_MANI_01469], is the reason why the reference in the role CryptoKeySlotUsageDesignMapping.portPrototype is modeled as a flat reference instead of an instanceRef, which would otherwise be considered the canonical solution.

Class	**CryptoKeySlotUsageDesignMapping**			
Note	This meta-class represents the ability to map a CryptoKeySlotUsageDesign (on the system design level) to a PortPrototype (on the application design level). **Tags:** atp.recommendedPackage=CryptoKeySlotDesignMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cryptoKeySlot UsageDesign	CryptoKeySlotUsage Design	0..1	ref	This reference identifies the applicable CryptoKeySlot UsageDesign on system design level. **Tags:** atp.Splitkey=cryptoKeySlotUsageDesign
portPrototype	RPortPrototype	0..1	ref	This reference identifies the applicable PortPrototype on application design level. **Tags:** atp.Splitkey=portPrototype

Table 3.81: CryptoKeySlotUsageDesignMapping

[advisory_01014] Existence of the reference in the role CryptoKeySlotUsage-DesignMapping.portPrototype

Imposition time: IT_ProDes

⌈For each CryptoKeySlotUsageDesignMapping, the reference in the role port-Prototype should exist.⌋

[advisory_01015] Existence of the reference in the role CryptoKeySlotUsage-DesignMapping.cryptoKeySlotUsageDesign

Imposition time: IT_ProDes

⌈For each CryptoKeySlotUsageDesignMapping, the reference in the role cryptoKeySlotUsageDesign should exist.⌋

4 Application Port Interface Design

4.1 Overview

This chapter documents all relevant PortInterfaces used by application software on the *AUTOSAR adaptive platform*.

The description of the ServiceInterface (Chapter 4.2) represents the most prominent use case for PortInterfaces on the *AUTOSAR adaptive platform*. This prominence is also expressed by the ability to map different ServiceInterfaces onto each other. The support for this mapping is described in Chapter 4.3 and Chapter 4.4.

The other subclasses of PortInterface described in this chapter are mostly about interaction with the functional clusters on the *AUTOSAR adaptive platform*.

4.2 Service Interface

4.2.1 Overview

[TPS_MANI_01001] Meaning of ServiceInterface ⌈Meta-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.⌋

The purpose of this modeling is to embrace the concept of [6, service-oriented communication] and better support this paradigm for communication on the *AUTOSAR adaptive platform*.

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 Chapter 4.2.8.

[constr_1483] Applicability of a ServiceInterface

Imposition time: IT_BefAraApiGen

⌈The 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.

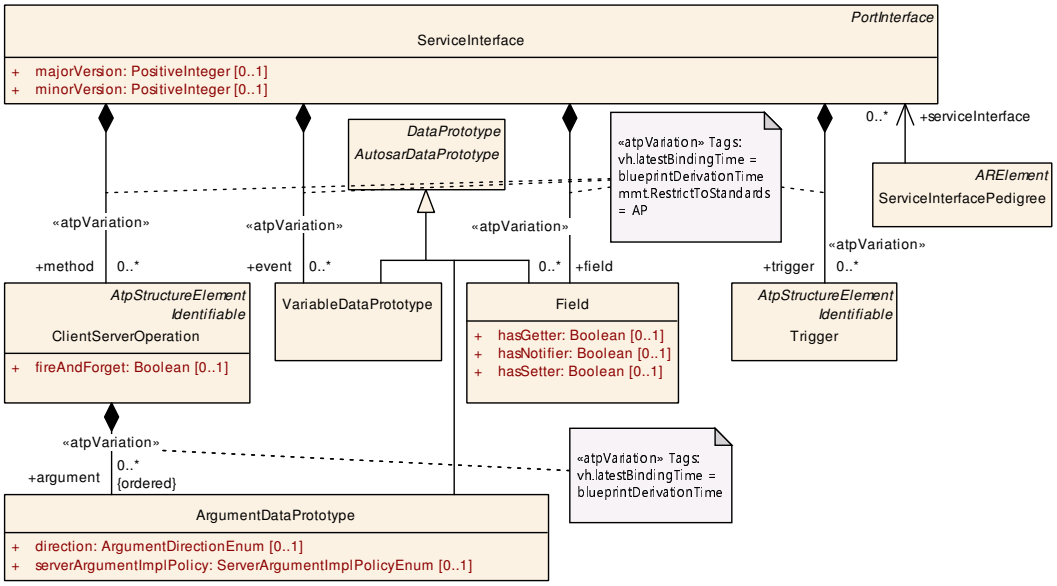


Figure 4.1: Modeling of the ServiceInterface

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, see [TPS_MANI_01032].

Class	**ServiceInterface**			
Note	This represents the ability to define a PortInterface that consists of a heterogeneous collection of methods, events and fields. **Tags:** atp.recommendedPackage=ServiceInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
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.Splitkey=event, event.shortName, event.variation Point.shortLabel 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.Splitkey=field, field.shortName, field.variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=40

△

Class	**ServiceInterface**			
majorVersion	PositiveInteger	0..1	attr	Major version of the service contract. **Tags:** xml.sequenceOffset=10
method	ClientServerOperation	*	aggr	This represents the collection of methods defined in the context of a ServiceInterface. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=method, method.shortName, method.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=50
minorVersion	PositiveInteger	0..1	attr	Minor version of the service contract. **Tags:** xml.sequenceOffset=20
trigger	Trigger	*	aggr	This represents the collection of triggers defined in the context of a ServiceInterface. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=trigger, trigger.shortName, trigger.variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=60

Table 4.1: 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] Atomic unit of service discovery ⌈As far as the application level is concerned, the atomic unit for **service discovery** on the *AUTOSAR adaptive platform* is the `ServiceInterface`.⌋

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.

4.2.2 Event

[TPS_MANI_01033] Semantics of `ServiceInterface.event` ⌈An `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).⌋

[constr_1494] Initial value for `event`

Imposition time: `IT_BefAraApiGen`

⌈A `ServiceInterface.event` shall **not** have an `initValue`.⌋

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**			
Note	A `VariableDataPrototype` represents a formalized generic piece of information that is typically mutable by the application software layer. `VariableDataPrototype` is used in various contexts and the specific context gives the otherwise generic `VariableDataPrototype` a dedicated semantics.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Aggregated by	ApplicationInterface.indication, *AtpClassifier*.atpFeature, BswInternalBehavior.arTypedPerInstance Memory, BswModuleDescription.providedData, BswModuleDescription.requiredData, BulkNvData Descriptor.bulkNvBlock, DiagnosticSovdAccessArgument.contentObject, *InternalBehavior*.staticMemory, NvBlockDescriptor.ramBlock, NvDataInterface.nvData, SenderReceiverInterface.dataElement, Service Interface.event, SwcInternalBehavior.arTypedPerInstanceMemory, SwcInternalBehavior.explicitInter RunnableVariable, SwcInternalBehavior.implicitInterRunnableVariable			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the VariableDataPrototype **Tags:** atp.Splitkey=initValue

Table 4.2: VariableDataPrototype

4.2.3 Trigger

[TPS_MANI_03291] Semantics of `ServiceInterface.trigger` ⌈A `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.⌋

The `trigger` is processed in the queued manner, i.e. the `trigger`s are stored in a queue and are processed in “first in first out” order.

Class	**Trigger**			
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*			
Aggregated by	*AtpClassifier*.atpFeature, BswModuleDescription.releasedTrigger, BswModuleDescription.requiredTrigger, ServiceInterface.trigger, TriggerInterface.trigger			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.3: Trigger

4.2.4 Field

[TPS_MANI_01034] Semantics of `ServiceInterface.field` ⌈A `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.⌋

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

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.

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. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ApplicationInterface.attribute, *AtpClassifier*.atpFeature, ServiceInterface.field			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
hasGetter	Boolean	0..1	attr	This attribute controls whether read access is foreseen to this field.
hasNotifier	Boolean	0..1	attr	This attribute controls whether a notification semantics is foreseen to this field.
hasSetter	Boolean	0..1	attr	This attribute controls whether write access is foreseen to this field.

Table 4.4: Field

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] Existence of attributes hasGetter, hasSetter, and hasNotifier

Imposition time: IT_BefAraApiGen

⌈For any given Field, all of the attributes

- hasGetter
- hasSetter
- hasNotifier

shall exist and at least one of the attributes shall be set to true.⌋

[constr_1673] allows a Field to 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.

[constr_10129] Multiplicity of attribute `Field.hasGetter`

Imposition time: `IT_BefAraApiGen`

⌈For each `Field`, the attribute `hasGetter` shall exist.⌋

[constr_10130] Multiplicity of attribute `Field.hasSetter`

Imposition time: `IT_BefAraApiGen`

⌈For each `Field`, the attribute `hasSetter` shall exist.⌋

[constr_10131] Multiplicity of attribute `Field.hasNotifier`

Imposition time: `IT_BefAraApiGen`

⌈For each `Field`, the attribute `hasNotifier` shall exist.⌋

4.2.5 Method

[TPS_MANI_01035] Semantics of `ServiceInterface.method` ⌈A `method` represents a function that is executed by and in the scope of a server on request of one or more client(s).⌋

Class	**ClientServerOperation**			
Note	An operation declared within the scope of a client/server interface.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ApplicationInterface.command, *AtpClassifier*.atpFeature, ClientServerInterface.operation, DiagnosticDataElementInterface.read, DiagnosticDataIdentifierInterface.read, DiagnosticDataIdentifierInterface.write, DiagnosticExtendedDataRecordInterface.provide, DiagnosticRoutineInterface.requestResult, DiagnosticRoutineInterface.start, DiagnosticRoutineInterface.stop, ServiceInterface.method			
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). This Attribute is only used by the AUTOSAR Adaptive Platform.
possibleApError	ApApplicationError	*	ref	This reference identifies AdaptivePlatformApplicationErrors as a possible error raised by the enclosing ClientServerOperation. **Tags:** atp.Splitkey=possibleApError This Attribute is only used by the AUTOSAR Adaptive Platform.

▽

△

Class	ClientServerOperation			
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.Splitkey=possibleApErrorSet This Attribute is only used by the AUTOSAR Adaptive Platform.

Table 4.5: ClientServerOperation

4.2.5.1 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.

Viewed 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] Semantics of attribute `method.fireAndForget` ⌈The activation of the “fire & forget” semantics of a given `method` is achieved by setting the value of attribute `method.fireAndForget` to value `true`.⌋

[TPS_MANI_03118] Semantics of `ServiceInterface.method` with `fireAndForget` set to true ⌈A `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.⌋

[constr_3374] `method` with attribute `fireAndForget` set to true shall not have any inout or out arguments

Imposition time: `IT_BefAraApiGen`

⌈A `method` that has the value of attribute `fireAndForget` set to `true` is not allowed to have any `argument`s with `direction` `inout` or `out`.⌋

[constr_3375] `method` with attribute `fireAndForget` set to true shall not reference an `ApApplicationError`

Imposition time: `IT_BefAraApiGen`

⌈A `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`.

⌋

[TPS_MANI_03119] Default value for the attribute `fireAndForget` of meta-class `ClientServerOperation` ⌈If the attribute `fireAndForget` is not defined then it shall be assumed that no “fire & forget” semantics is intended.⌋

4.2.6 Versioning of `ServiceInterface`s

Using multiple versions of the same `ServiceInterface` supports an independent life cycle of services and allows changing and enhancing `ServiceInterface`s without affecting 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.

If different versions of the same `ServiceInterface` are used in one `Executable`, 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] Semantic versioning of `ServiceInterface.majorVersion` and `ServiceInterface.minorVersion` ⌈Service 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.

⌋

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**			
Note	Collection of ServiceInterfaces that belong to the same versioning. **Tags:** atp.recommendedPackage=ServiceInterfacePedigrees This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
serviceInterface	ServiceInterface	*	ref	Reference to the ServiceInterfaces which belong to the same versioning. **Tags:** atp.Splitkey=serviceInterface

Table 4.6: 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.

4.2.6.1 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 a separate impact assessment on the compatibility needs to be performed whether the changed service interface has an incompatible representation on this transport layer.

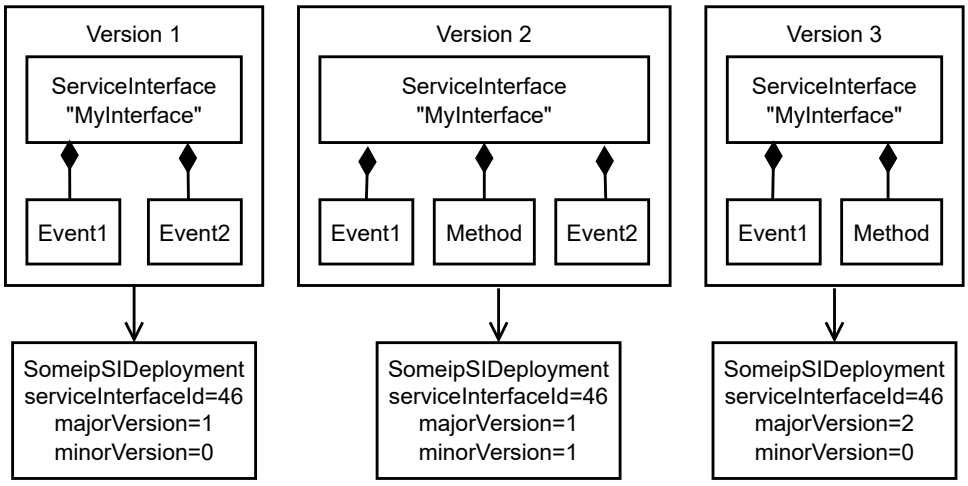


Figure 4.2: Example for different versions of the same ServiceInterface

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 13.2.

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.

The compatibility rules for SOME/IP are described in the [11, PRS SOME/IP Protocol].

4.2.7 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 4.3.

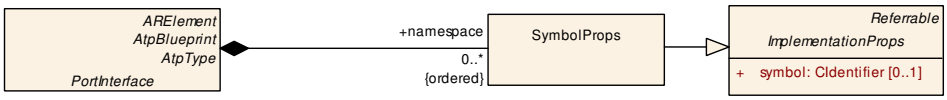


Figure 4.3: Specification of namespaces in PortInterfaces

[TPS_MANI_01004] Semantics of ServiceInterface.namespace ⌈The aggregation ServiceInterface.namespace shall be used to define the namespace to be used for the source code that corresponds to the given ServiceInterface.⌋

[TPS_MANI_01005] The definition of the namespace of a ServiceInterface may follow a hierarchical pattern ⌈The 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.⌋

```
<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>
```

Listing 4.1: Example for the definition of a namespace for a given ServiceInterface

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[1].

[1]This means that the definition of a namespace a::b is semantically different from the definition of a namespace b::a.

[TPS_MANI_01006] Ordered definition of `ServiceInterface.namespace` ⌈In 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.⌋

Class	***PortInterface*** (abstract)			
Note	Abstract base class for an interface that is either provided or required by a port of a software component.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractRawDataStreamInterface*, *AbstractSuspendToRamInterface*, *AbstractSynchronizedTimeBase Interface*, ClientServerInterface, *CryptoInterface*, *DataInterface*, *DiagnosticPortInterface*, FirewallState SwitchInterface, *IdsmAbstractPortInterface*, LogAndTraceInterface, ModeSwitchInterface, Network ManagementPortInterface, *PersistencyInterface*, *PlatformHealthManagementInterface*, ServiceInterface, *StateManagementPortInterface*, TriggerInterface			
Aggregated by	ARPackage.element			
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. This Attribute is only used by the AUTOSAR Adaptive Platform.

Table 4.7: PortInterface

Class	**SymbolProps**			
Note	This meta-class represents the ability to contribute a part of a namespace.			
Base	*ARObject*, *ImplementationProps*, *Referrable*			
Aggregated by	Allocator.namespace, ApApplicationErrorDomain.namespace, *AtomicSwComponentType*.symbolProps, *CppImplementationDataType*.namespace, ImplementationDataType.symbolProps, *Port Interface*.namespace, SecurityEventDefinition.eventSymbolName, TraceSwitchConfig.namespace			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.8: SymbolProps

The Listing Listing 4.1 exemplifies the statement made by [TPS_MANI_01006], i.e. the resulting name space in e.g. C++ would look like sketched in Listing Listing 4.2.

```
namespace com {
    namespace myCompany {
        namespace software {

        }
    }
}
```

Listing 4.2: Resulting namespace for the example `ServiceInterface`

4.2.8 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] Semantics of ApApplicationError ⌈Meta-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.⌋

Class	**ApApplicationError**			
Note	This meta-class represents the ability to formally specify the semantics of an application error on the AUTOSAR adaptive platform **Tags:** atp.recommendedPackage=ApplicationErrors This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
errorCode	Integer	0..1	attr	This attribute has the ability to specify the error code value within the enclosing AdaptivePlatformApplication Error.
errorDomain	ApApplicationError Domain	0..1	ref	This reference represents the error domain of the Ap ApplicationError. **Tags:** atp.Splitkey=errorDomain

Table 4.9: ApApplicationError

[constr_10132] Multiplicity of attribute ApApplicationError.errorCode

Imposition time: IT_BefAraApiGen

⌈For each ApApplicationError, the attribute errorCode shall exist.⌋

[TPS_MANI_03674] Adaptive platform SOME/IP Service Instance interacting with a classic platform SOME/IP Service Instance ⌈If a ProvidedSomeipServiceInstance / RequiredSomeipServiceInstance is referenced by a SomeipServiceInstanceToMachineMapping and a ConsumedServiceInstance / ProvidedServiceInstance (on a classic platform EcuInstance) exists on the same VLAN with the following matching SOME/IP credentials:

For ProvidedSomeipServiceInstance:

- ProvidedSomeipServiceInstance.serviceInstanceId equals ConsumedServiceInstance.instanceIdentifier
- SomeipServiceInterfaceDeployment.serviceInterfaceId equals ConsumedServiceInstance.serviceIdentifier
- SomeipServiceVersion.majorVersion equals ConsumedServiceInstance.majorVersion

For RequiredSomeipServiceInstance:

- RequiredSomeipServiceInstance.requiredServiceInstanceId equals ProvidedServiceInstance.instanceIdentifier
- SomeipServiceInterfaceDeployment.serviceInterfaceId equals ProvidedServiceInstance.serviceIdentifier
- SomeipServiceVersion.majorVersion equals ProvidedServiceInstance.majorVersion

then the ProvidedSomeipServiceInstance / RequiredSomeipServiceInstance is considered to interact with the AUTOSAR classic platform.⌋

[constr_3772] Range of ApApplicationError.errorCode for interoperability with the AUTOSAR classic platform

Imposition time: IT_SysDes

⌈If a SOME/IP service instance interacts with a classic platform according to [TPS_MANI_03674], then the value of attribute ApApplicationError.errorCode used according to [TPS_MANI_01191] shall not exceed the closed interval [1..63].⌋

Rationale for [constr_3772]: This restriction originates from [SWS_Std_00011] which limits the usable bits of the Std_ReturnType (which is the return type of ClientServerOperations in AUTOSAR classic platform) for encoding errors to the lower 6 bits.

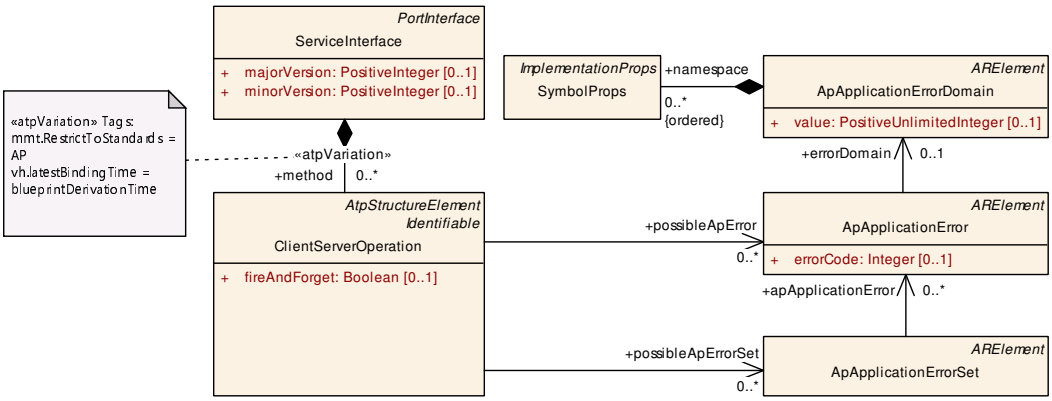


Figure 4.4: Modeling of ApApplicationError on the *AUTOSAR adaptive platform*

[constr_1625] Existence of reference ApApplicationError.errorDomain

Imposition time: IT_BefAraApiGen

⌈For each ApApplicationError, the reference errorDomain shall exist.

In other words, the association of an ApApplicationError with a corresponding ApApplicationErrorDomain is mandatory.⌋

[TPS_MANI_01198] Semantics of meta-class ApApplicationErrorSet
⌈Meta-class 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.⌋

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

It is also possible to mix the usage of ClientServerOperation.possibleApError and ClientServerOperation.possibleApErrorSet.

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.

Class	**ApApplicationErrorSet**			
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.recommendedPackage=ApplicationErrorSets This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
apApplication Error	ApApplicationError	*	ref	This reference represents the collection of ApApplication Error represented by the enclosing ApApplicationErrorSet **Tags:** atp.Splitkey=apApplicationError

Table 4.10: ApApplicationErrorSet

[TPS_MANI_01191] Modeling of possible errors ⌈A ClientServerOperation aggregated by a ServiceInterface in the role method may reference

- zero or more ApApplicationError(s) in the role possibleApError and/or
- zero or more ApApplicationErrorSet(s) in the role possibleApErrorSet

to formally specify the existence of possible errors raised by the ClientServerOperation.⌋

[TPS_MANI_01192] Semantics of ApApplicationErrorDomain ⌈Meta-class 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.⌋

[TPS_MANI_01863] Supported value range for attribute ApApplicationErrorDomain.value ⌈The supported value range of attribute ApApplicationErrorDomain.value is given by [constr_2637].⌋

[constr_2637] is defined in the document [8, TPS Generic Structure Template].

[constr_10574] Value of ApApplicationErrorDomain.value shall be unique in the context of a ClientServerOperation

Imposition time: IT_BefAraApiGen

⌈Within the context of one ClientServerOperation, no two ApApplicationErrorDomains referenced in the role ClientServerOperation.possibleApError.errorDomain or transitively referenced in the role ClientServerOperation.possibleApErrorSet.apApplicationError.errorDomain shall have the identical value of attribute ApApplicationErrorDomain.value.⌋

For more explanation of [constr_10574], see Chapter B.12.1.

[constr_10133] Multiplicity of attribute ApApplicationErrorDomain.value

Imposition time: IT_BefAraApiGen

⌈For each ApApplicationErrorDomain, the attribute value shall exist.⌋

[constr_1664] Unique ApApplicationError.shortName

Imposition time: IT_BefAraApiGen

⌈Within the collection of all ApApplicationErrors that reference a given ApApplicationErrorDomain in the role errorDomain, the attribute ApApplicationError.shortName shall have a unique value.⌋

[constr_1665] Unique ApApplicationError.errorCode

Imposition time: IT_BefAraApiGen

⌈Within the collection of all ApApplicationErrors that reference a given ApApplicationErrorDomain in the role errorDomain, the attribute ApApplicationError.errorCode shall have a unique value.⌋

Class	**ApApplicationErrorDomain**
Note	This meta-class represents the ability to define a global error domain for an ApApplicationError. **Tags:** atp.recommendedPackage=ApplicationErrors This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*
Aggregated by	ARPackage.element

▽

△

Class	**ApApplicationErrorDomain**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
namespace (ordered)	SymbolProps	*	aggr	This aggregation defines the namespace of the Ap ApplicationErrorDomain
value	PositiveUnlimitedInteger	0..1	attr	This attribute identifies the error category.

Table 4.11: ApApplicationErrorDomain

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 compile-time errors.

[constr_3773] Uniqueness of `ApApplicationError.errorCode` for interoperability with the AUTOSAR classic platform

Imposition time: `IT_BefAraApiGen`

⌈If a SOME/IP service instance interacts with a classic platform according to [TPS_MANI_03674], then the set of `ApApplicationError`s configured for a `ClientServerOperation` according to [TPS_MANI_01191] shall all have unique `ApApplicationError.errorCode` values defined.⌋

Rationale for [constr_3773]: For interoperability with the AUTOSAR classic platform the potential reference `ApApplicationError.errorDomain` is irrelevant because on the AUTOSAR classic platform the `ApApplicationErrorDomain` and the `ApApplicationErrorDomain.value` are not available. Thus [constr_3773] has to apply to guarantee unique `ApApplicationError.errorCode` value definitions in the context of a `ClientServerOperation`.

4.2.9 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 automated and will be executed by a code generator. This code generator needs an input from the model. The main input for this purpose is the definition of the `ServiceInterface` itself.

However, this is not sufficient. The designer of a `ServiceInterface` is free to use `ApplicationDataType`s 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[2].

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.

Since the definition of ServiceInterfaces is 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 introduction to 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.

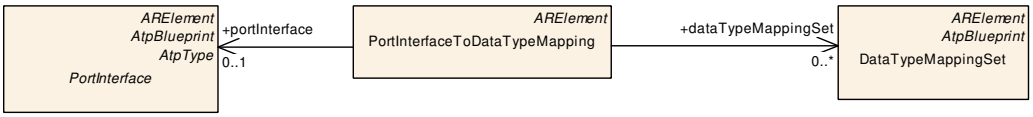


Figure 4.5: Modeling of PortInterfaceToDataTypeMapping

[constr_1507] PortInterfaceToDataTypeMapping is only applicable to PortInterfaces that contain modeled data types

Imposition time: IT_BefAraApiGen

⌈PortInterfaceToDataTypeMapping.portInterface shall only refer to **any** of the following sub-classes of PortInterface:

- ServiceInterface
- PersistencyKeyValueStorageInterface

[2]For more background regarding the definition and use of meta-class DataTypeMappingSet, see the [2, TPS Software ComponentTemplate].

- DiagnosticRoutineInterface
- DiagnosticDataIdentifierInterface
- DiagnosticDataElementInterface
- DiagnosticSovdContentInterface
- DiagnosticSovdContentElementInterface
- DiagnosticSovdOperationInterface

⌋

Class	**PortInterfaceToDataTypeMapping**			
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.recommendedPackage=PortInterfaceToDataTypeMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataType MappingSet	DataTypeMappingSet	*	ref	This represents the reference to the applicable data TypemappingSet **Tags:** atp.Splitkey=dataTypeMappingSet
portInterface	PortInterface	0..1	ref	This represents the reference to the applicable Port Interface **Tags:** atp.Splitkey=portInterface

Table 4.12: PortInterfaceToDataTypeMapping

[constr_10134] Multiplicity of reference in the role PortInterfaceToDataTypeMapping.dataTypeMappingSet

Imposition time: IT_BefAraApiGen

⌈For each PortInterfaceToDataTypeMapping, the reference in the role dataTypeMappingSet shall exist at least once.⌋

[constr_10135] Multiplicity of reference in the role PortInterfaceToDataTypeMapping.portInterface

Imposition time: IT_BefAraApiGen

⌈For each PortInterfaceToDataTypeMapping, the reference in the role portInterface shall exist.⌋

Class	**DataTypeMappingSet**			
Note	This class represents a list of mappings between `ApplicationDataType`s and `ImplementationDataType`s. In addition, it can contain mappings between `ImplementationDataType`s and `ModeDeclarationGroup`s. **Tags:** atp.recommendedPackage=DataTypeMappingSets			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataTypeMap	DataTypeMap	*	aggr	This is one particular association between an `ApplicationDataType` and its `AbstractImplementationDataType`. **Tags:** atp.Splitkey=dataTypeMap
modeRequest TypeMap	ModeRequestTypeMap	*	aggr	This is one particular association between an `ModeDeclarationGroup` and its `AbstractImplementationDataType`. **Tags:** atp.Splitkey=modeRequestTypeMap

Table 4.13: DataTypeMappingSet

Class	**DataTypeMap**			
Note	This class represents the relationship between `ApplicationDataType` and its implementing `AbstractImplementationDataType`.			
Base	*ARObject*			
Aggregated by	DataTypeMappingSet.dataTypeMap			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
applicationData Type	ApplicationDataType	0..1	ref	This is the corresponding `ApplicationDataType` **Tags:** atp.Splitkey=applicationDataType
implementation DataType	AbstractImplementation DataType	0..1	ref	This is the corresponding `AbstractImplementationDataType`. **Tags:** atp.Splitkey=implementationDataType

Table 4.14: DataTypeMap

4.3 Service Interface Mapping

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 supports different granularities 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.

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

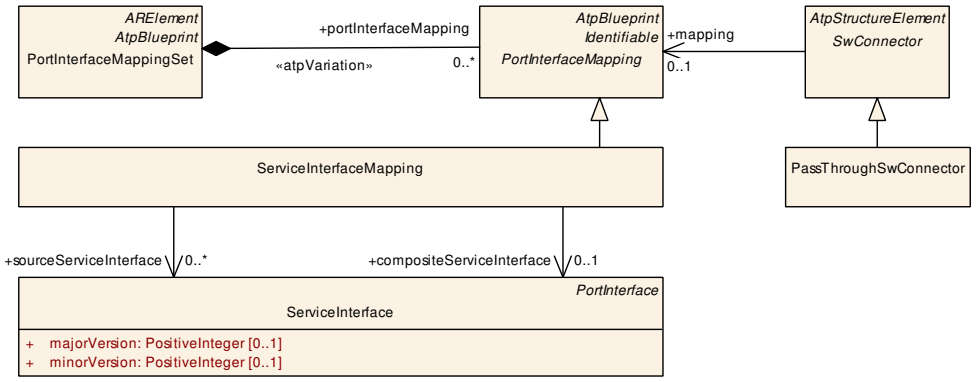


Figure 4.6: Modeling of the ServiceInterfaceMapping

[TPS_MANI_01002] Semantics of meta-class ServiceInterfaceMapping ⌈In 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.

⌋

[TPS_MANI_01032] Usage of ServiceInterfaceMapping ⌈It 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 "coarse-grained" 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 "coarse-grained" Port is described with the ServiceInterfaceMapping or rather ServiceInterfaceElementMapping.⌋

The modeling of a *facade* SwComponentType does not make any assumptions about the implementation and about the realization of such a *facade* functionality. The *fa-*

cade 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. timing of offering the coarse-grained service instance) are also project-specific and are not pre-defined by AUTOSAR.

Figure 4.7 summarizes the idea behind the creation of a *facade* software-component. The latter is able to "bundle" the communication of different `PortPrototype`s owned by potentially different `SwComponentType`s for external communication.

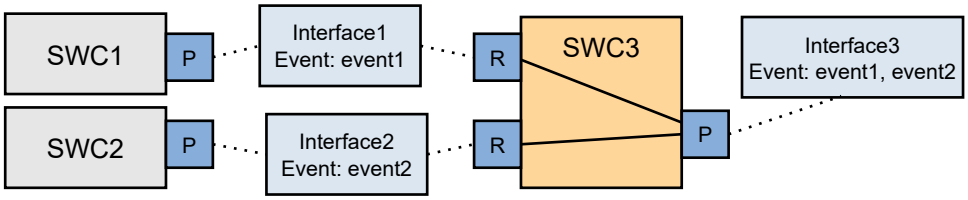


Figure 4.7: Concept of a facade software-component

In other words, elements `event1` owned by `SWC1` and `event2` owned by `SWC2` are combined into one `ServiceInterface` used to type one `PortPrototype` of the *facade* software-component.

From the outside, in terms of communication, `SWC3` acts as a facade that hides the "inner structure" created by `SWC1` and `SWC2`.

[constr_5056] Restriction of sub-class of `CompositionSwComponentType.connector`

Imposition time: `IT_BefAraApiGen`

⌈In the context of a `CompositionSwComponentType.connector` (transitively) referenced by a `Executable.rootSwComponentPrototype`, the only supported sub-class of `SwConnector` is `PassThroughSwConnector`.⌋

[constr_5057] `PassThroughSwConnector` and `ServiceInterfaceMapping`

Imposition time: `IT_BefAraApiGen`

⌈If a `PassThroughSwConnector` is defined between two Ports in a `CompositionSwComponentType` either:

- a `ServiceInterfaceMapping` between the `ServiceInterface`s of these two Ports shall be defined and the `PassThroughSwConnector` shall reference the relevant `ServiceInterfaceMapping` in the role `mapping` or
- `ServiceInterfaceElementMapping`s for elements of `ServiceInterface`s of the two Ports shall be defined and the `PassThroughSwConnector` shall reference the relevant `ServiceInterfaceElementMapping`s in the role `serviceInterfaceElementMapping`.

⌋

[TPS_MANI_01022] Concept behind `ServiceInterfaceMapping` ⌈The 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`.⌋

The regulation stated in [TPS_MANI_01022] is exemplified in Figure 4.8.

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 `ServiceInterface`s 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).

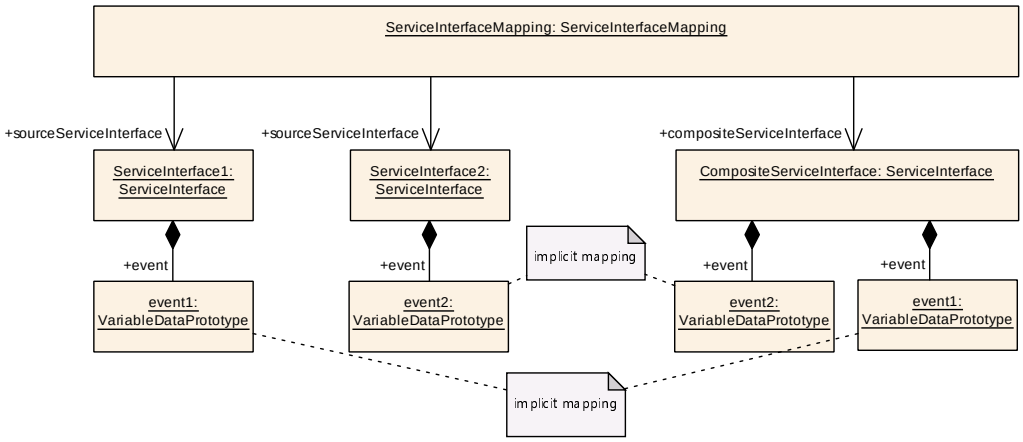


Figure 4.8: Example for the application of a `ServiceInterfaceMapping`

[TPS_MANI_01003] Limitation of the applicability of `ServiceInterfaceMapping` ⌈The applicability of the `ServiceInterfaceMapping` is limited to cases where the `shortName`s of the elements of the `compositeServiceInterface` are **unique** in the context of the `compositeServiceInterface`.⌋

Class	**ServiceInterfaceMapping**			
Note	Specifies one ServiceInterfaceMapping that allows to define that a ServiceInterface is composite of several other ServiceInterfaces. **Tags:** atp.recommendedPackage=ServiceInterfaceMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpBlueprint*, *Identifiable*, *MultilanguageReferrable*, *PortInterfaceMapping*, *Referrable*			
Aggregated by	PortInterfaceMappingSet.portInterfaceMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ServiceInterfaceMapping**			
composite ServiceInterface	ServiceInterface	0..1	ref	This represents the composite ServiceInterface. **Tags:** atp.Splitkey=compositeServiceInterface
sourceService Interface	ServiceInterface	*	ref	ServiceInterface that is mapped into the composite ServiceInterface. **Tags:** atp.Splitkey=sourceServiceInterface

Table 4.15: ServiceInterfaceMapping

[constr_10136] Multiplicity of reference in the role ServiceInterfaceMapping.compositeServiceInterface

Imposition time: IT_BefAraApiGen

⌈For each ServiceInterfaceMapping, the reference in the role compositeServiceInterface shall exist.⌋

[constr_10137] Multiplicity of reference in the role ServiceInterfaceMapping.sourceServiceInterface

Imposition time: IT_BefAraApiGen

⌈For each ServiceInterfaceMapping, the reference in the role sourceServiceInterface shall exist at least once.⌋

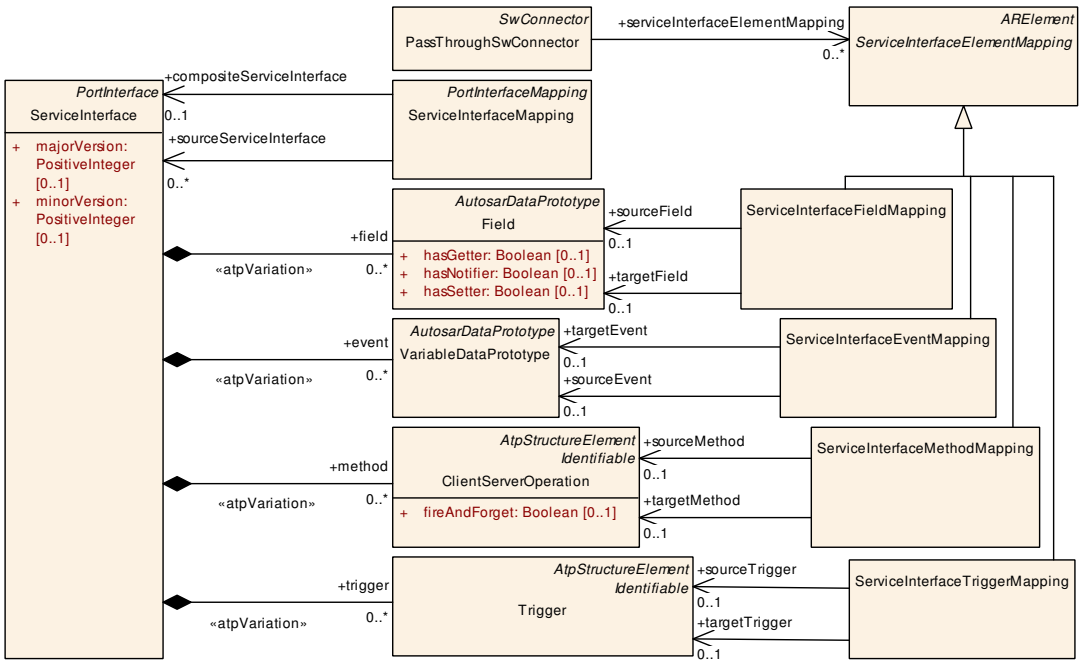


Figure 4.9: Overview of the modeling of the ServiceInterfaceMapping and ServiceInterfaceElementMapping

The ServiceInterfaceMapping is not an up-front association (by means of SwConnectors) between communication ends in the sense of Chapter 4.2.6.

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].

Class	**PassThroughSwConnector**			
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*			
Aggregated by	*AtpClassifier*.atpFeature, CompositionSwComponentType.connector			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
providedOuterPort	AbstractProvidedPortPrototype	0..1	ref	This represents the provided outer delegation Port Prototype of the PassThroughSwConnector. **Tags:** atp.Splitkey=providedOuterPort
requiredOuterPort	AbstractRequiredPortPrototype	0..1	ref	This represents the required outer delegation Port Prototype of the PassThroughSwConnector. **Tags:** atp.Splitkey=requiredOuterPort
serviceInterfaceElementMapping	ServiceInterfaceElementMapping	*	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.Splitkey=serviceInterfaceElementMapping This Attribute is only used by the AUTOSAR Adaptive Platform.

Table 4.16: PassThroughSwConnector

4.4 Service Interface Element Mapping

4.4.1 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.

An example of the definition of a ServiceInterfaceEventMapping can be found in Chapter B.2.2.1.

[constr_1482] Mapping of service interfaces vs. mapping of service interface elements

Imposition time: IT_BefAraApiGen

⌈In 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.

⌋

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.

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).

Class	***ServiceInterfaceElementMapping*** (abstract)			
Note	This abstract meta-class acts as base class for the mapping of specific elements of a ServiceInterface. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	ServiceInterfaceEventMapping, ServiceInterfaceFieldMapping, ServiceInterfaceMethodMapping, ServiceInterfaceTriggerMapping			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.17: ServiceInterfaceElementMapping

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

[TPS_MANI_03277] ServiceInterfaceElementMappings for a subset of elements of a single ServiceInterface ⌈If 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.⌋

With [TPS_MANI_03277] use cases are supported as shown in Figure 4.10 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.

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

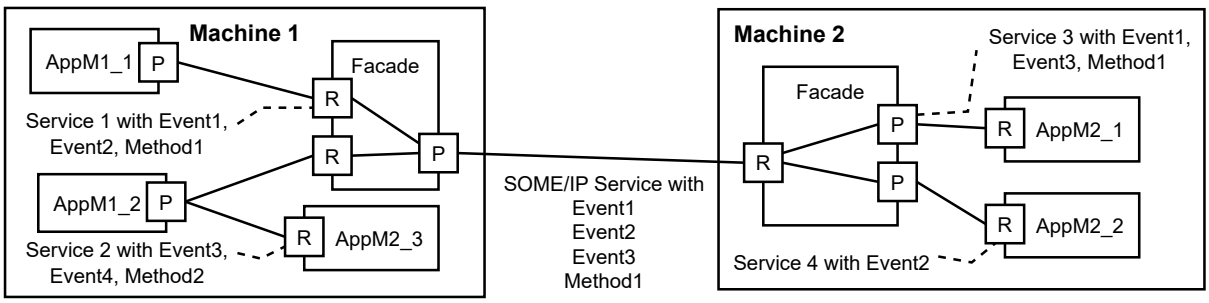


Figure 4.10: Example for ServiceInterfaceElementMapping for a subset of elements of a single ServiceInterface on the provider side

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.

Similar use cases are applicable on the consumer side as well as shown in Figure 4.11 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.

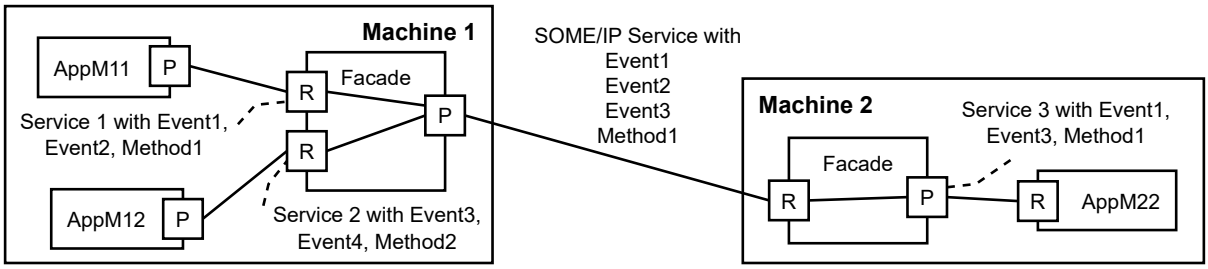


Figure 4.11: Example for `ServiceInterfaceElementMapping` for a subset of elements of a single `ServiceInterface` on the consumer side

4.4.2 Service Interface Event Mapping

[TPS_MANI_01024] Semantics of `ServiceInterfaceEventMapping` ⌈Meta-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`.⌋



Figure 4.12: Modeling of the `ServiceInterfaceEventMapping`

Class	**ServiceInterfaceEventMapping**			
Note	This meta-class allows to define a mapping between events of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:** atp.recommendedPackage=ServiceInterfaceElementMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInterfaceElementMapping*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ServiceInterfaceEventMapping**			
sourceEvent	VariableDataPrototype	0..1	ref	Reference to an event that is contained in the source ServiceInterface. **Tags:** atp.Splitkey=sourceEvent
targetEvent	VariableDataPrototype	0..1	ref	Reference to an event that is contained in the composite ServiceInterface. **Tags:** atp.Splitkey=targetEvent

Table 4.18: ServiceInterfaceEventMapping

[constr_10138] Multiplicity of reference in the role `ServiceInterfaceEventMapping.sourceEvent`

Imposition time: `IT_BefAraApiGen`

⌈For each `ServiceInterfaceEventMapping`, the reference in the role `sourceEvent` shall exist.⌋

[constr_10139] Multiplicity of reference in the role `ServiceInterfaceEventMapping.targetEvent`

Imposition time: `IT_BefAraApiGen`

⌈For each `ServiceInterfaceEventMapping`, the reference in the role `targetEvent` shall exist.⌋

The explicit mapping implemented by `ServiceInterfaceEventMapping` does **not** require equal `shortName`s 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 B.17.

The mapping of one `sourceEvent` to different `targetEvent`s 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 target-configuration. For more explanation, see an example of how the role-based mapping of elements of `ServiceInterface`s works in Figure B.24.

4.4.3 Service Interface Trigger Mapping

[TPS_MANI_03289] Semantics of `ServiceInterfaceTriggerMapping` ⌈Meta-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`.⌋

The explicit mapping implemented by `ServiceInterfaceTriggerMapping` does **not** require equal `shortName`s 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.

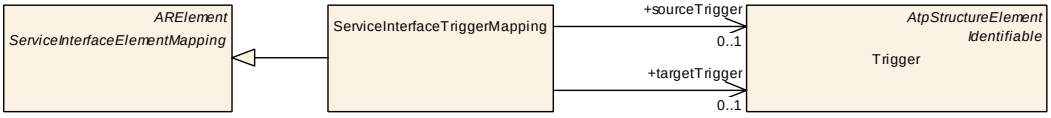


Figure 4.13: Modeling of the ServiceInterfaceTriggerMapping

Class	**ServiceInterfaceTriggerMapping**			
Note	This meta-class allows to define a mapping between triggers of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:** atp.recommendedPackage=ServiceInterfaceElementMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInterfaceElementMapping*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sourceTrigger	Trigger	0..1	ref	Reference to a trigger that is contained in the source ServiceInterface. **Tags:** atp.Splitkey=sourceTrigger
targetTrigger	Trigger	0..1	ref	Reference to a trigger that is contained in the target ServiceInterface. **Tags:** atp.Splitkey=targetTrigger

Table 4.19: ServiceInterfaceTriggerMapping

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 target-configuration.

4.4.4 Service Interface Field Mapping

[TPS_MANI_01025] Semantics of ServiceInterfaceFieldMapping ⌈Meta-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.⌋

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

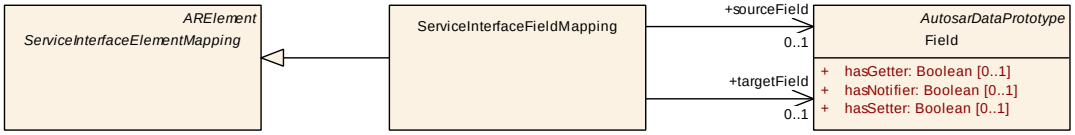


Figure 4.14: Modeling of the ServiceInterfaceFieldMapping

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 B.18.

Class	**ServiceInterfaceFieldMapping**			
Note	This meta-class allows to define a mapping between fields of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:** atp.recommendedPackage=ServiceInterfaceElementMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInterfaceElementMapping*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sourceField	Field	0..1	ref	Reference to a field that is contained in the source ServiceInterface. **Tags:** atp.Splitkey=sourceField
targetField	Field	0..1	ref	Reference to a field that is contained in the composite ServiceInterface. **Tags:** atp.Splitkey=targetField

Table 4.20: ServiceInterfaceFieldMapping

[constr_10140] Multiplicity of reference in the role ServiceInterfaceFieldMapping.sourceField

Imposition time: IT_BefAraApiGen

⌈For each ServiceInterfaceFieldMapping, the reference in the role sourceField shall exist.⌋

[constr_10141] Multiplicity of reference in the role ServiceInterfaceFieldMapping.targetField

Imposition time: IT_BefAraApiGen

⌈For each ServiceInterfaceFieldMapping, the reference in the role targetField shall exist.⌋

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 target-configuration. For more explanation, see an example of how the role-based mapping of elements of ServiceInterfaces works in Figure B.24.

4.4.5 Service Interface Method Mapping

[TPS_MANI_01026] Semantics of ServiceInterfaceMethodMapping ⌈Meta-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.⌋

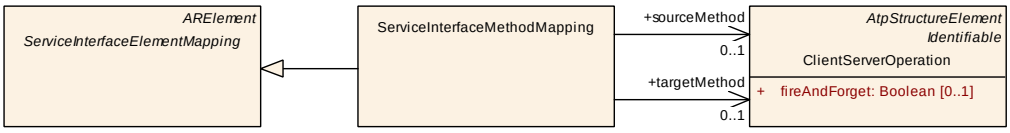


Figure 4.15: Modeling of the `ServiceInterfaceMethodMapping`

Class	**ServiceInterfaceMethodMapping**			
Note	This meta-class allows to define a mapping between methods of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:** atp.recommendedPackage=ServiceInterfaceElementMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInterfaceElementMapping*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sourceMethod	ClientServerOperation	0..1	ref	Reference to a method that is contained in the source ServiceInterface. **Tags:** atp.Splitkey=sourceMethod
targetMethod	ClientServerOperation	0..1	ref	Reference to a method that is contained in the composite ServiceInterface. **Tags:** atp.Splitkey=targetMethod

Table 4.21: ServiceInterfaceMethodMapping

[constr_10142] Multiplicity of reference in the role `ServiceInterfaceMethodMapping.sourceMethod`

Imposition time: `IT_BefAraApiGen`

⌈For each `ServiceInterfaceMethodMapping`, the reference in the role `sourceMethod` shall exist.⌋

[constr_10143] Multiplicity of reference in the role `ServiceInterfaceMethodMapping.targetMethod`

Imposition time: `IT_BefAraApiGen`

⌈For each `ServiceInterfaceMethodMapping`, the reference in the role `targetMethod` shall exist.⌋

The explicit mapping implemented by `ServiceInterfaceMethodMapping` does **not** require equal `shortName`s 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 B.19.

The mapping of one `sourceMethod` to different `targetMethod`s 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 target-configuration. For more explanation, see an example of how the role-based mapping of elements of `ServiceInterface`s works in Figure B.24.

4.5 State Management Interface

4.5.1 Overview

PortInterfaces defined in the context of State Management are intended to be used for the modeling of the State Management itself, i.e. the PortInterfaces described in this chapter shall be used to type PortPrototypes in the context of one or more Executable(s) that implement(s) the State Management.

There are two kinds of PortInterfaces used in the context of State Management:

- ServiceInterface, see Chapter 4.2
- sub-classes of StateManagementPortInterface, see Figure 4.16

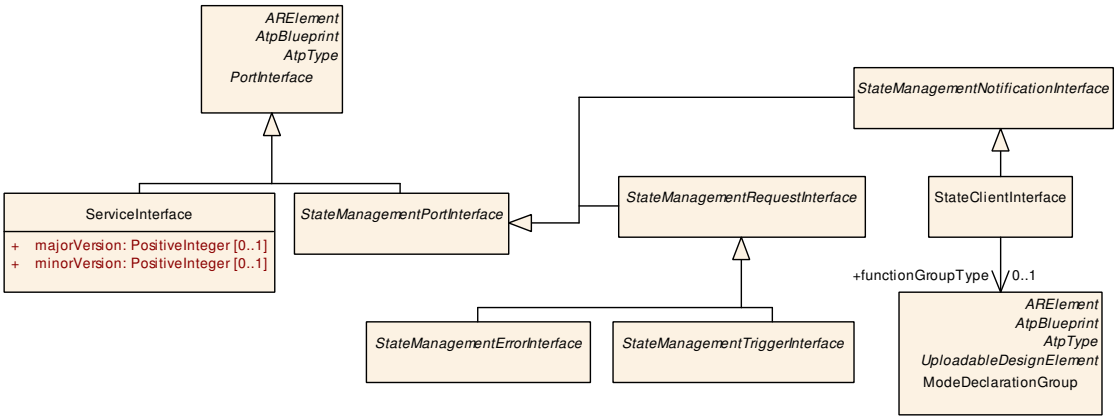


Figure 4.16: Big picture of State Management interface modeling

Furthermore, the subclasses of StateManagementPortInterface are subdivided in PortInterfaces for the processing of requests to State Management (represented by abstract class StateManagementRequestInterface) and the notification of state switches, modeled as abstract class StateManagementNotification-Interface.

The “big picture” of the modeling of PortInterfaces for State Management is depicted in Figure 4.16.

Class	***StateManagementPortInterface*** (abstract)			
Note	This abstract class acts as a base class for PortInterfaces that are used in the context of state management on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	*StateManagementNotificationInterface*, *StateManagementRequestInterface*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.22: StateManagementPortInterface

In terms of requests, State Management interacts with a variety of function cluster modules, and the interaction patterns as well as the underlying APIs are very diverse. Therefore, it is necessary (for the use cases that don't utilize `ServiceInterface`s) to define a collection of different `PortInterface`s separately for the two main request use cases:

- Processing of "triggers", see Chapter 4.5.3
- Processing of "errors", see Chapter 4.5.4

Class	***StateManagementNotificationInterface*** (abstract)			
Note	This meta-class acts as an abstract base class for state management notification interfaces. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *StateManagementPortInterface*			
Subclasses	StateClientInterface, StateManagementFunctionGroupSwitchNotificationInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.23: StateManagementNotificationInterface

Class	***StateManagementRequestInterface*** (abstract)			
Note	This meta-class acts as an abstract base class for state management request interfaces. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *StateManagementPortInterface*			
Subclasses	*StateManagementErrorInterface*, *StateManagementTriggerInterface*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.24: StateManagementRequestInterface

Consequently, further abstract base classes are introduced to organize the definition of `PortInterface`s for the different State Management use cases:

- `StateManagementTriggerInterface`
- `StateManagementErrorInterface`

Class	***StateManagementTriggerInterface*** (abstract)
Note	The usage of this meta-class for typing a PortPrototype indicates that the PortPrototype is used for the trigger provision in the context of state management on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.

▽

△

Class	***StateManagementTriggerInterface*** (abstract)			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *StateManagementPortInterface*, *StateManagementRequestInterface*			
Subclasses	StateManagementDiagTriggerInterface			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.25: StateManagementTriggerInterface

Class	***StateManagementErrorInterface*** (abstract)			
Note	The usage of this meta-class for typing a PortPrototype indicates that the PortPrototype is used for the error provision in the context of state management on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *StateManagementPortInterface*, *StateManagementRequestInterface*			
Subclasses	StateManagementEmErrorInterface, StateManagementPhmErrorInterface			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.26: StateManagementErrorInterface

4.5.2 State Management Interfaces for Notifications

As depicted by Figure 4.17, there are two kinds of notification that the State Management needs to support:

- Notification of `Function Group` state switch. This notification is typically directed at the execution management, it is formalized as `StateManagementFunctionGroupSwitchNotificationInterface`.
- Notification of the switch of an “internal” state, represented by a `field` owned by a `ServiceInterface`.

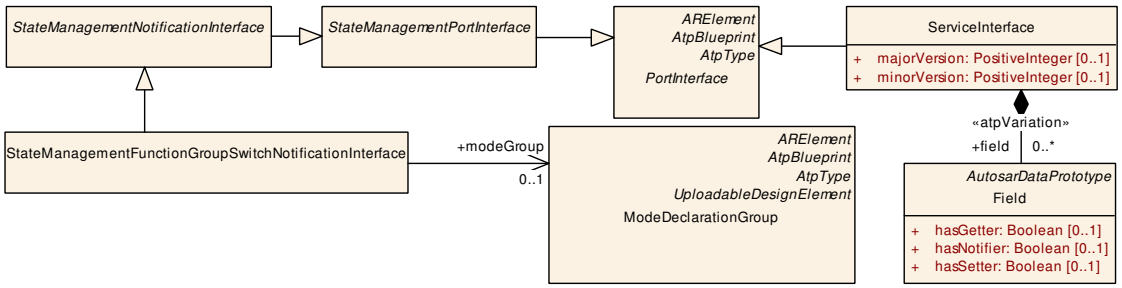


Figure 4.17: Modeling of `StateManagementPortInterface`s for state switch notifications

Class	StateManagementFunctionGroupSwitchNotificationInterface			
Note	The usage of this meta-class for typing a PortPrototype indicates that the PortPrototype is used for sending out a notification of a function group state change in the context of state management on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=StateManagementPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *StateManagementNotificationInterface*, *StateManagementPortInterface*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
modeGroup	ModeDeclarationGroup	0..1	ref	This reference identifies the ModeDeclarationGroup that defines the individual states that that can be switched to. **Tags:** atp.Splitkey=modeGroup atp.Status=draft

Table 4.27: StateManagementFunctionGroupSwitchNotificationInterface

[constr_10389] Existence of attribute StateManagementFunctionGroupSwitchNotificationInterface.modeGroup

Status: DRAFT
Imposition time: IT_Mani

⌈For each StateManagementFunctionGroupSwitchNotificationInterface, the aggregation in the role modeGroup shall exist.⌋

4.5.3 State Management Interfaces for Triggers

In the majority of cases, communication of triggers is done by means of ServiceInterface. The notable exception from this rule is the diagnostics management, which uses a dedicated subclass of StateManagementTriggerInterface: StateManagementDiagTriggerInterface.

Class	StateManagementDiagTriggerInterface			
Note	This meta-class indicates that the PortPrototype that references this class is used for accepting a state change trigger from the diagnostics management. **Tags:** atp.Status=draft atp.recommendedPackage=StateManagementPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *StateManagementPortInterface*, *StateManagementRequestInterface*, *StateManagementTriggerInterface*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.28: StateManagementDiagTriggerInterface

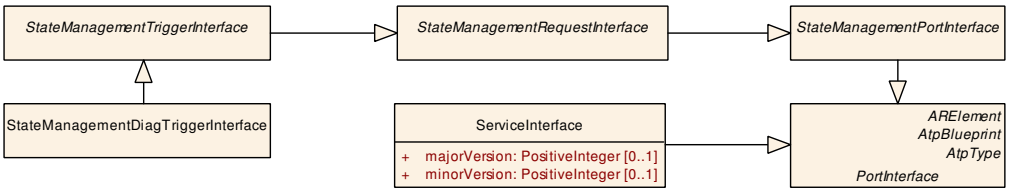


Figure 4.18: Modeling of State Management trigger interfaces

4.5.4 State Management Interfaces for Errors

In contrast to the communication of triggers, errors in the context of State Management are entirely handled by dedicated subclasses of `StateManagementErrorInterface`, as depicted in Figure 4.19.

As of the current state of interaction patterns with the State Management, two functional clusters report errors directly to State Management:

- For Execution Management (see [13, SWS Execution Management]), State Management exposes a `PortPrototype` typed by `StateManagementEmErrorInterface`.
- For Platform Health Management (see [14, SWS Platform Health Management]), State Management exposes a `PortPrototype` typed by `StateManagementPhmErrorInterface`.

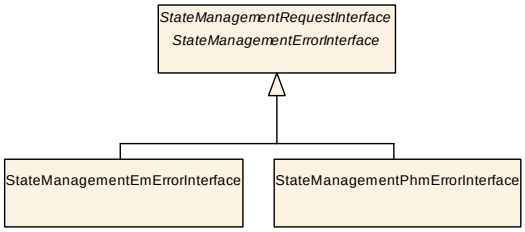


Figure 4.19: Modeling of State Management error interfaces

Class	**StateManagementEmErrorInterface**			
Note	This meta-class indicates that the PortPrototype that references this class is used for accepting a error submissions from the execution management. **Tags:** atp.Status=draft atp.recommendedPackage=StateManagementPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *StateManagementErrorInterface*, *StateManagementPortInterface*, *StateManagementRequestInterface*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.29: StateManagementEmErrorInterface

Class	StateManagementPhmErrorInterface			
Note	This meta-class indicates that the PortPrototype that references this class is used for accepting a error submissions from the platform health management. **Tags:** atp.Status=draft atp.recommendedPackage=StateManagementPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *StateManagementErrorInterface*, *StateManagementPortInterface*, *StateManagementRequestInterface*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.30: StateManagementPhmErrorInterface

4.6 Execution Management Interface

4.6.1 State Client Interface

[TPS_MANI_01473] Semantics of `StateClientInterface`

Status: DRAFT

⌈The `StateClientInterface` is used to type an `RPortPrototype` over which function group changes are communicated from the *State Management* to the *Execution Management*.⌋

The `StateClientInterface` works on type level, i.e. `ModeDeclarationGroup` and `ModeDeclaration`. The connection to a specific function group is created via the `FunctionGroupPortMapping` (see Chapter 12.14.5.3).

Class	StateClientInterface			
Note	This PortInterface is used communicate changes of a function group state towards the Execution Management. **Tags:** atp.Status=draft atp.recommendedPackage=StateManagementPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *StateManagementNotificationInterface*, *StateManagementPortInterface*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
functionGroupType	ModeDeclarationGroup	0..1	ref	This reference identifies the ModeDeclarationGroup that shall be used in the API of the StateClientInterface. **Tags:** atp.Splitkey=functionGroupType atp.Status=draft

Table 4.31: StateClientInterface

[constr_10616] Existence of the reference in the role `StateClientInterface.functionGroupType`

Status: DRAFT
Imposition time: `IT_Mani`

⌈For each `StateClientInterface`, the reference in the role `functionGroupType` shall exist.⌋

4.7 Persistency Interface

4.7.1 Overview

4.7.1.1 The big Picture

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

There are some similarities to the communication model in terms of the usage of `PortPrototype`s.

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 key-value storage an entire key-value storage is accessible on the level of `PortPrototype`).

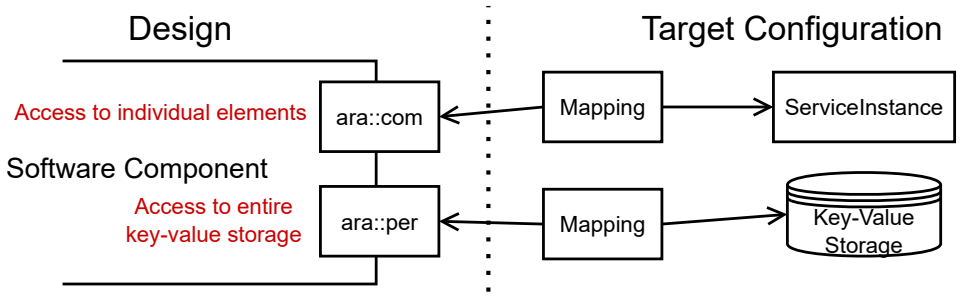


Figure 4.20: General approach for the modeling of persistency

The aspect of target-configuration for the configuration of persistent data is explained in Figure 4.20.

The AUTOSAR meta-model actually defines two separate meta-classes (for more details, see Figure 4.21) 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`).

4.7.1.2 Modeling of Persistency Interface

Abstract meta-class `PersistencyInterface` has been created as a means of categorization, i.e. it allows for easily referring to `PortInterface`s 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`.

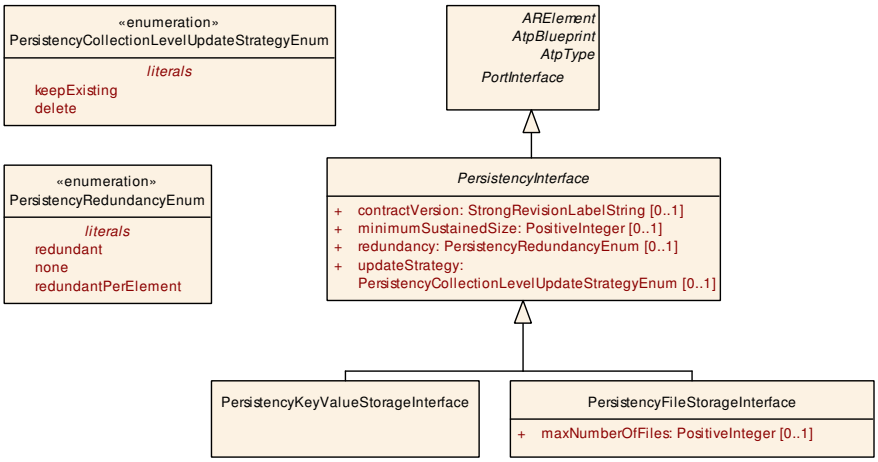


Figure 4.21: Specification of `PortInterface`s for persistency use cases

Class	***PersistencyInterfaceElement*** (abstract)			
Note	This meta-class provides the abstract ability to define an element of a PortInterface for the support of persistency use cases. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	PersistencyDataElement, PersistencyFileElement			
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 PersistencyInterfaceElement.

Table 4.32: PersistencyInterfaceElement

[constr_10427] Multiplicity of attribute `PersistencyInterfaceElement.updateStrategy`

Imposition time: `IT_Mani`

⌈For each `PersistencyInterfaceElement`, the attribute `updateStrategy` shall exist.⌋

[TPS_MANI_01194] Semantics of attribute `PersistencyInterface.minimumSustainedSize` ⌈Attribute `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.⌋

Class	***PersistencyInterface*** (abstract)			
Note	This meta-class provides the abstract ability to define a PortInterface for the support of persistency use cases. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	PersistencyFileStorageInterface, PersistencyKeyValueStorageInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contractVersion	StrongRevisionLabelString	0..1	attr	This attribute represents the contract version that is used to determine whether the Persistency configuration experienced structural changes and is also used for the check for data type compatibility.
minimumSustainedSize	PositiveInteger	0..1	attr	The value of this attribute represents the minimum size (unit: bytes) required at design time for the enclosing PersistencyInterface.
redundancy	PersistencyRedundancyEnum	0..1	attr	This attribute represents a requirement towards the redundancy of storage.
redundancyHandling	PersistencyRedundancyHandling	*	aggr	This aggregation represents the chosen approaches to handle redundancy for the various use cases implemented by subclasses **Tags:** atp.Splitkey=redundancyHandling
updateStrategy	PersistencyCollectionLevelUpdateStrategyEnum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyInterface as a whole.

Table 4.33: PersistencyInterface

[constr_10485] Existence of PersistencyInterface.contractVersion

Imposition time: IT_BefAraApiGen

⌈In the context of PersistencyInterface, the attribute in the role contractVersion shall exist.⌋

4.7.1.3 Redundancy Handling

[TPS_MANI_01204] Specification of redundancy of persistent data ⌈The 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.⌋

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.

[TPS_MANI_01319] Modeling of redundancy in the context of PersistencyInterface ⌈As 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.⌋

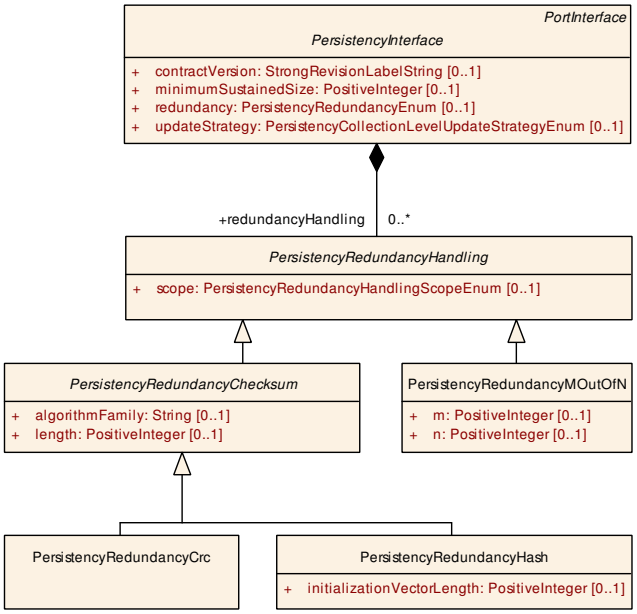


Figure 4.22: Specification of redundancy on the level of PersistencyInterface

Enumeration	**PersistencyRedundancyEnum**
Note	This meta-class provides a way to specify in which way redundancy shall be applied on collection level. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	*PersistencyInterface*.redundancy
Literal	***Description***
none	This value represents the requirement that redundancy measures are not applied on persistency storage level. **Tags:** atp.EnumerationLiteralIndex=1
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
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

Table 4.34: PersistencyRedundancyEnum

[TPS_MANI_01320] Definition of redundancy on interface level may be overruled in target-configuration ⌈The 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 target-configuration phase, it is explicitly foreseen that an integrator may overrule the design decision regarding persistency based on superior knowledge only available at target-configuration time.⌋

Class	***PersistencyRedundancyHandling*** (abstract)			
Note	This abstract base class represents a formal description of redundancy. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Subclasses	*PersistencyRedundancyChecksum*, PersistencyRedundancyMOutOfN			
Aggregated by	*PersistencyDeployment*.redundancyHandling, *PersistencyInterface*.redundancyHandling			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
scope	PersistencyRedundancy HandlingScopeEnum	0..1	attr	This attribute controls the scope in which the redundancy handling is applied.

Table 4.35: PersistencyRedundancyHandling

[constr_1746] Mutual exclusive existence of PersistencyInterface.redundancy and PersistencyInterface.redundancyHandling

Imposition time: IT_BefAraApiGen

⌈For each PersistencyInterface, either the attribute redundancy or the aggregation of PersistencyRedundancyHandling in the role redundancyHandling may exist.⌋

Enumeration	**PersistencyRedundancyHandlingScopeEnum**
Note	This meta-class provides values to control the scope of redundancy measures in the persistency target-configuration. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	*PersistencyRedundancyHandling*.scope
Literal	***Description***
persistency RedundancyHandlingScope Element	The redundancy handling shall be applied on element level (key-value pair and file). **Tags:** atp.EnumerationLiteralIndex=0
persistency RedundancyHandlingScope Storage	The redundancy handling shall be applied on storage (key-value storage and file storage) level. **Tags:** atp.EnumerationLiteralIndex=1

Table 4.36: PersistencyRedundancyHandlingScopeEnum

[TPS_MANI_01207] Standardized values of attribute PersistencyRedundancyChecksum.algorithmFamily

⌈The 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**

⌋

Class	***PersistencyRedundancyChecksum*** (abstract)			
Note	Abstract class that defines the common attributes for implementations of redundancy. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *PersistencyRedundancyHandling*			
Subclasses	PersistencyRedundancyCrc, PersistencyRedundancyHash			
Aggregated by	*PersistencyDeployment*.redundancyHandling, *PersistencyInterface*.redundancyHandling			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
algorithmFamily	String	0..1	attr	This attribute identifies the algorithm family that is used to execute the CRC/Hash.
length	PositiveInteger	0..1	attr	This attribute describes the length of the CRC/Hash in the unit bits.

Table 4.37: PersistencyRedundancyChecksum

[constr_10144] Multiplicity of reference in the role PersistencyRedundancyChecksum.algorithmFamily

Imposition time: IT_BefAraApiGen

⌈For each PersistencyRedundancyChecksum, the reference in the role algorithmFamily shall exist.⌋

[constr_10145] Multiplicity of reference in the role PersistencyRedundancyChecksum.length

Imposition time: IT_BefAraApiGen

⌈For each PersistencyRedundancyChecksum, the reference in the role length shall exist.⌋

[constr_1668] Allowed combinations of PersistencyRedundancyChecksum.length and algorithmFamily

Imposition time: IT_BefAraApiGen

⌈

	8	16	32	64
CRC_J1850	x			
CRC_CCITT_FALSE		x		
CRC_ETHERNET			x	
CRC_0x42F0E1EBA9EA3693				x
CRC_8H2F	x			
CRC_16ARC		x		
CRC_32P4			x	

⌋

Class	PersistencyRedundancyCrc			
Note	This meta-class formally describes the usage of a CRC for the implementation of redundancy. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *PersistencyRedundancyChecksum*, *PersistencyRedundancyHandling*			
Aggregated by	*PersistencyDeployment*.redundancyHandling, *PersistencyInterface*.redundancyHandling			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 4.38: PersistencyRedundancyCrc

Class	PersistencyRedundancyHash			
Note	This meta-class formally describes the usage of a Hash for the implementation of redundancy. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *PersistencyRedundancyChecksum*, *PersistencyRedundancyHandling*			
Aggregated by	*PersistencyDeployment*.redundancyHandling, *PersistencyInterface*.redundancyHandling			
Attribute	Type	Mult.	Kind	Note
initialization VectorLength	PositiveInteger	0..1	attr	Length of the initialization vector.

Table 4.39: PersistencyRedundancyHash

[constr_10046] Value of `PersistencyRedundancyMOutOfN.n`

Imposition time: `IT_BefAraApiGen`

⌈The value of attribute `PersistencyRedundancyMOutOfN.n` shall be set at least to 2 and at most to 255, i.e. the allowed interval is [2..255].⌋

[constr_1751] Value of `PersistencyRedundancyMOutOfN.m`

Imposition time: `IT_BefAraApiGen`

⌈The 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`].⌋

Class	PersistencyRedundancyMOutOfN			
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. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *PersistencyRedundancyHandling*			
Aggregated by	*PersistencyDeployment*.redundancyHandling, *PersistencyInterface*.redundancyHandling			
Attribute	Type	Mult.	Kind	Note
m	PositiveInteger	0..1	attr	This attribute represents the "M" coordinate in the "M out of N" scheme.
n	PositiveInteger	0..1	attr	This attribute represents the "N" coordinate in the "M out of N" scheme.

Table 4.40: PersistencyRedundancyMOutOfN

[constr_10146] Multiplicity of reference in the role PersistencyRedundancyMOutOfN.m

Imposition time: IT_BefAraApiGen

⌈For each PersistencyRedundancyMOutOfN, the reference in the role m shall exist.⌋

[constr_10147] Multiplicity of reference in the role PersistencyRedundancyMOutOfN.n

Imposition time: IT_BefAraApiGen

⌈For each PersistencyRedundancyMOutOfN, the reference in the role n shall exist.⌋

4.7.1.4 Update Handling

[TPS_MANI_01139] Semantics of PersistencyInterface.updateStrategy ⌈The 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.⌋

Enumeration	PersistencyCollectionLevelUpdateStrategyEnum
Note	This enumeration provides possible values for the update strategy on interface/storage level. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	*PersistencyDeployment*.updateStrategy, *PersistencyInterface*.updateStrategy
Literal	*Description*
delete	The update strategy is to delete all values on the level of the respective collection. **Tags:** atp.EnumerationLiteralIndex=1
keepExisting	The update strategy is to keep the existing values on the level of the respective collection. **Tags:** atp.EnumerationLiteralIndex=0

Table 4.41: PersistencyCollectionLevelUpdateStrategyEnum

[TPS_MANI_01140] Semantics of PersistencyInterfaceElement.updateStrategy ⌈The attribute PersistencyInterfaceElement.updateStrategy can be used to specify the strategy for updating the actual persistent element that corresponds to PersistencyInterfaceElement.⌋

The behavior of the software in terms of applying an update strategy is explained in detail in the [15, SWS Persistency].

Enumeration	**PersistencyElementLevelUpdateStrategyEnum**
Note	This enumeration provides possible values for the update strategy on element level. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	*PersistencyDeploymentElement*.updateStrategy, *PersistencyInterfaceElement*.updateStrategy
Literal	***Description***
delete	The update strategy is to delete the value of the respective data item. **Tags:** atp.EnumerationLiteralIndex=2
keepExisting	The update strategy is to keep the existing value of the respective data item. **Tags:** atp.EnumerationLiteralIndex=1
overwrite	The update strategy is to overwrite the respective data item. **Tags:** atp.EnumerationLiteralIndex=0

Table 4.42: PersistencyElementLevelUpdateStrategyEnum

4.7.2 Persistency Key Value Storage Interface

[TPS_MANI_01065] Purpose of `PersistencyKeyValueStorageInterface` ⌈The purpose of the `PersistencyKeyValueStorageInterface` is to support the persistent access to data in a key-value storage.⌋

Class	**PersistencyKeyValueStorageInterface**			
Note	This meta-class provides the ability to implement a `PortInterface` for supporting persistency use cases for data. **Tags:** atp.recommendedPackage=PersistencyKeyValueStorageInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyInterface*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
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.Splitkey=dataElement, dataElement.short Name
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.Splitkey=dataTypeForSerialization
dataType Mapping	PersistencyKeyValue DataTypeMapping	0..1	aggr	This aggregation provides a collection of replacement rules for data types used in the context of the enclosing PersistencyKeyValueStorageInterface. **Tags:** atp.Splitkey=dataTypeMapping

Table 4.43: PersistencyKeyValueStorageInterface

[TPS_MANI_01135] Semantics of `PersistencyKeyValueStorageInterface.dataTypeForSerialization` ⌈The 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.⌋

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.

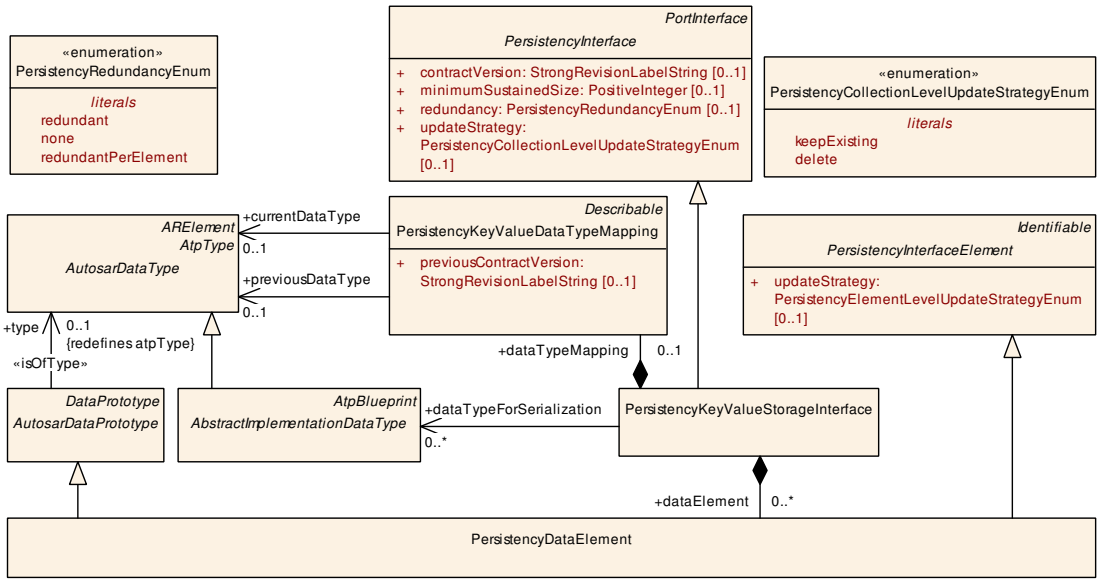


Figure 4.23: Modeling of the PersistencyKeyValueStorageInterface

[TPS_MANI_01138] Semantics of attribute PersistencyKeyValueStorageInterface.dataElement ⌈By 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.⌋

[TPS_MANI_01180] Collection of data types that requires serialization support ⌈The 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.

⌋

A PersistencyDataElement can be typed by either an ApplicationDataType or else a CppImplementationDataType.

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. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *PersistencyInterfaceElement*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, PersistencyKeyValueStorageInterface.dataElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.44: PersistencyDataElement

[constr_10425] Existence of initial values for PersistencyDataElement

Imposition time: IT_BefAraApiGen

⌈For each PersistencyDataElement, if the value of attribute PersistencyInterface.updateStrategy is set to either

- PersistencyElementLevelUpdateStrategyEnum.keepExisting or
- PersistencyElementLevelUpdateStrategyEnum.overwrite,

then a PersistencyDataRequiredComSpec shall be aggregated by a RPortPrototype that is typed by the PersistencyInterface that owns the specific PersistencyDataElement.

The PersistencyDataRequiredComSpec shall

- refer to the respective PersistencyDataElement and
- aggregate ValueSpecification in the role initValue.

⌋

[constr_10420] Restriction for the existence of initial values for PersistencyDataElement

Imposition time: IT_BefAraApiGen

⌈For each PersistencyDataElement, if the value of attribute updateStrategy is set to the value PersistencyElementLevelUpdateStrategyEnum.delete, then the specific PersistencyDataElement shall not be referenced by a PersistencyDataRequiredComSpec that is aggregated by a RPortPrototype that in turn is typed by the PersistencyInterface that owns the specific PersistencyDataElement.⌋

[TPS_MANI_01378] Semantics of meta-class PersistencyKeyValueDataTypeMapping

⌈Meta-class PersistencyKeyValueDataTypeMapping supports the porting of a previously used data type (represented by the role previousDataType)

used in an existing key-value pair to the currently used data type (represented by the role currentDataType) used in the context of the application software.

This way, application software that uses a given AutosarDataType can access a key-value-pair that was stored using a different (i.e. used in previous versions) data type as long as the mapping from the previous data type to the data type used currently in the key-value-pair is formally defined by means of the PersistencyKeyValueDataTypeMapping such that the application software can provide a suitable conversion. The conversion will typically happen in the update callback function registered by the application.⌋

Class	**PersistencyKeyValueDataTypeMapping**			
Note	This meta-class represents the ability to define a mapping between an existing data type in a key-value-storage stored by a previous version to a new data type used on application software level in the current version. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Describable*			
Aggregated by	PersistencyKeyValueStorageInterface.dataTypeMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
currentDataType	AutosarDataType	0..1	ref	This reference identifies the current data type for an existing key-value-pair in the context of the enclosing PersistencyKeyValueStorageInterface. **Tags:** atp.Splitkey=currentDataType
previousContractVersion	StrongRevisionLabelString	0..1	attr	This attribute identifies the contract version in which the previousDataType was used.
previousDataType	AutosarDataType	0..1	ref	This reference identifies the previous data type in a key-value-pair existing in the context of the enclosing PersistencyKeyValueStorageInterface. **Tags:** atp.Splitkey=previousDataType

Table 4.45: PersistencyKeyValueDataTypeMapping

[constr_10486] Existence of the attribute PersistencyKeyValueDataTypeMapping.previousContractVersion

Imposition time: IT_BefAraApiGen

⌈In the context of PersistencyKeyValueDataTypeMapping, the attribute in the role previousContractVersion shall exist.⌋

A PersistencyKeyValueDataTypeMapping needs to be complete, i.e. for each PersistencyKeyValueDataTypeMapping the two references shall exist in any case.

[constr_10377] Completeness of the modeling of PersistencyKeyValueDataTypeMapping

Imposition time: IT_BefAraApiGen

⌈For each PersistencyKeyValueDataTypeMapping, the references in the roles

- previousDataType
- currentDataType

⌋

It only makes sense to map types of the same data type level, which is ensured by the following two constraints:

[constr_10378] Meta-class PersistencyKeyValueDataTypeMapping references AbstractImplementationDataType in the role currentDataType

Imposition time: IT_BefAraApiGen

⌈Each PersistencyKeyValueDataTypeMapping that references to an AbstractImplementationDataType as part of the collection in the role currentDataType shall also refer to an AbstractImplementationDataType in the role previousDataType.⌋

[constr_10379] Meta-class PersistencyKeyValueDataTypeMapping references ApplicationDataType in the role currentDataType

Imposition time: IT_BefAraApiGen

⌈Each PersistencyKeyValueDataTypeMapping that references to an ApplicationDataType as part of the collection in the role currentDataType shall also refer to an ApplicationDataType in the role previousDataType.⌋

It does not make much sense if the modeling of PersistencyKeyValueDataTypeMapping.currentDataType is not even used in the context of the enclosing PersistencyKeyValueStorageInterface because this way a conversion to an unused data type is defined that would never be used in reality. This caveat is addressed by [advisory_01009].

[advisory_01009] PersistencyKeyValueDataTypeMapping.currentDataType shall refer to a data type used in the context of the PersistencyKeyValueStorageInterface

Imposition time: IT_BefAraApiGen

⌈All data types referenced in the role PersistencyKeyValueDataTypeMapping.currentDataType shall be in use in the context of the enclosing PersistencyKeyValueStorageInterface. This means any data type referenced in the role currentDataType shall also be referenced in one of the following roles:

- PersistencyKeyValueStorageInterface.dataElement.type
- PersistencyKeyValueStorageInterface.dataTypeForSerialization

⌋

The usage of references previousDataType and currentDataType does not make any assumptions with respect to the compatibility of a pair of previousDataType and currentDataType. It is up to the application software to sort out any potential compatibility issues.

4.7.3 Persistency File Storage Interface

[TPS_MANI_01067] Purpose of `PersistencyFileStorageInterface` ⌈The purpose of meta-class `PersistencyFileStorageInterface` is to support access to an abstract representation of file storage.⌋

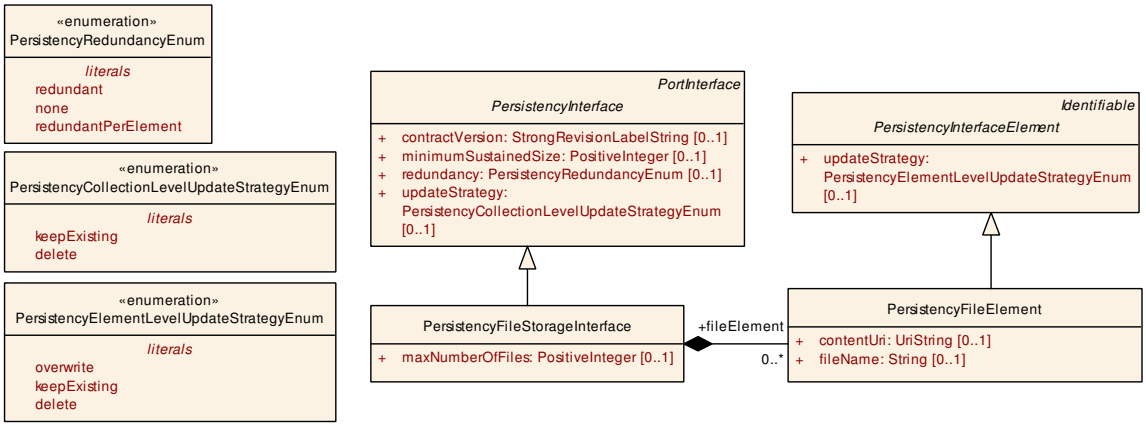


Figure 4.24: Modeling of the `PersistencyFileStorageInterface`

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] Semantics of `PersistencyFileStorageInterface.maxNumberOfFiles` ⌈Any `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.fileElement`s.⌋

The existence of the `PersistencyFileStorageInterface` does not violate the restrictions set by the POSIX subset PSE51 defined in [16, IEEE1003.13].

Class	**PersistencyFileStorageInterface**
Note	This meta-class provides the ability to implement a PortInterface for supporting persistency use cases for files. **Tags:** atp.recommendedPackage=PersistencyFileStorageInterfaces This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyInterface*, *PortInterface*, *Referrable*
Aggregated by	ARPackage.element

▽

△

Class	PersistencyFileStorageInterface			
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.Splitkey=fileElement, fileElement.shortName
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.

Table 4.46: 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] Semantics of PersistencyFileElement ⌈By 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.⌋

Class	PersistencyFileElement			
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. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PersistencyInterfaceElement*, *Referrable*			
Aggregated by	PersistencyFileStorageInterface.fileElement			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
contentUri	UriString	0..1	attr	This attribute represents the URI that identifies the initial content of the PersistencyFile.
fileName	String	0..1	attr	This attribute holds the filename part of the storage location, e.g. file on the file system.

Table 4.47: PersistencyFileElement

[constr_10148] Multiplicity of reference in the role PersistencyFileElement.contentUri

Imposition time: IT_BefAraApiGen

⌈For each PersistencyFileElement, the reference in the role contentUri shall exist if the value of attribute updateStrategy is set to either

- PersistencyElementLevelUpdateStrategyEnum.keepExisting or
- PersistencyElementLevelUpdateStrategyEnum.overwrite.

⌋

[constr_10149] Multiplicity of attribute in the role `PersistencyFileElement.fileName`

Imposition time: `IT_BefAraApiGen`

⌈For each `PersistencyFileElement`, the attribute in the role `fileName` shall exist.⌋

[constr_1581] Value of `fileElement.fileName`

Imposition time: `IT_BefAraApiGen`

⌈Within the scope of any given `PersistencyFileStorageInterface`, the value of all `fileElement.fileName` shall be unique.⌋

[constr_10080] Existence of initial values for `PersistencyFileElement`

Imposition time: `IT_BefAraApiGen`

⌈For each `PersistencyFileElement`, if the value of attribute `updateStrategy` is set to the value `delete`, then attribute `PersistencyFileElement.contentUri` shall not exist.⌋

4.8 Time Synchronization Interface

The Time Synchronization functional cluster within the Adaptive Platform is responsible for providing 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 `PortPrototype`s and `PortInterface`s 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 `PortPrototype`s and `PortInterface`s 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. [17, SWS Time Synchronization].

[TPS_MANI_03535] Definition of Time Synchronization interaction ⌈The meta-class `AbstractSynchronizedTimeBaseInterface` together with its subclasses are used to define the interaction of the application software with a Time Synchronization Time Base.⌋

For more information, see Figure 4.25.

Class	***AbstractSynchronizedTimeBaseInterface*** (abstract)			
Note	This meta-class provides the abstract ability to define a PortInterface for the interaction with Time Synchronization. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	SynchronizedTimeBaseConsumerInterface, SynchronizedTimeBaseProviderInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.48: AbstractSynchronizedTimeBaseInterface

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] Usage of `PortPrototype` for the interaction with Time Synchronization ⌈Depending 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.⌋

The application software may take the active or the passive role in the interaction with the 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] Time Synchronization interaction in a provider role ⌈The meta-class `SynchronizedTimeBaseProviderInterface` is used to indicate the intended interaction with a synchronized global Time Base in a *provider* role.⌋

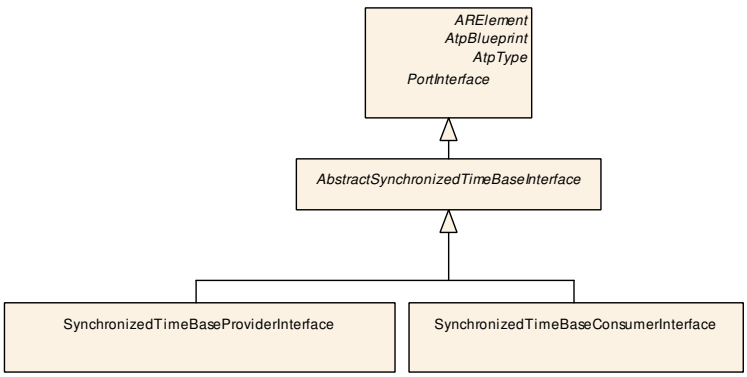


Figure 4.25: Modeling of Time Synch Interfaces

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] Time Synchronization interaction in a consumer role ⌈The meta-class `SynchronizedTimeBaseConsumerInterface` is used to indicate the intended interaction with a synchronized global Time Base in a *consumer* role.⌋

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.

Class	**SynchronizedTimeBaseProviderInterface**			
Note	This meta-class provides the ability to define a PortInterface for the interaction with a Time Synchronization Provider. **Tags:** atp.recommendedPackage=TimeSynchronizationInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractSynchronizedTimeBaseInterface*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.49: SynchronizedTimeBaseProviderInterface

Class	**SynchronizedTimeBaseConsumerInterface**			
Note	This meta-class provides the ability to define a PortInterface for the interaction with a Time Synchronization Consumer. **Tags:** atp.recommendedPackage=TimeSynchronizationInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractSynchronizedTimeBaseInterface*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.50: SynchronizedTimeBaseConsumerInterface

In the example in Figure 4.26 the interaction of one application with several time sync aspects is illustrated.

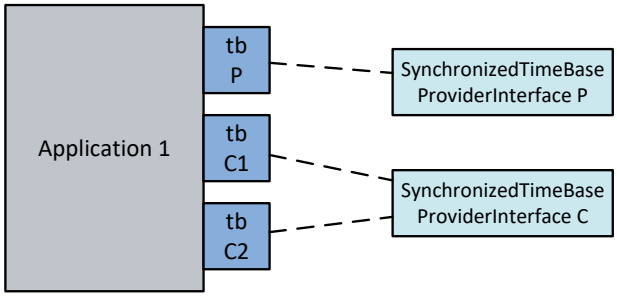


Figure 4.26: Example Application and Time Sync interaction

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 [17, SWS Time Synchronization]) is constructed using the InstanceSpecifier representing the PortPrototype name.

During application target-configuration, those PortPrototypes are mapped to actual Time-Base Resources in the Time-Sync Management (see Figure 12.23).

4.9 Platform Health Management Interface

4.9.1 Overview

The Platform Health Management functional cluster within the Adaptive Platform is responsible for supervising the execution of applications and triggering the State Management for the respective actions.

In order to interface with the Platform Health Management foundation software an application developer needs to declare which supervisions are 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 [14, SWS Platform Health Management].

4.9.2 Supervised Entities and Checkpoints

The interaction of supervision with the Platform Health Management is defined by PhmSupervisedEntityInterface and PhmCheckpoints.

[TPS_MANI_03500] Definition of Platform Health Management Supervision and Checkpoints ⌈The meta-class PhmSupervisedEntityInterface together with the aggregated PhmCheckpoint are used to define the interaction of one Supervised Entity with the Platform Health Management supervision.⌋

By defining an RPortPrototype which is typed by the PhmSupervisedEntityInterface the application indicates that it wants to report the checkpoints of this PhmSupervisedEntityInterface.

[constr_1727] Qualified combinations of PortPrototypes and PhmSupervisedEntityInterface on application software level

Imposition time: IT_BefAraApiGen

⌈Within the context of an Executable of category APPLICATION_LEVEL the usage of PhmSupervisedEntityInterface is **only** supported for an RPortPrototype.⌋

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.

An example for the relevance of the InstanceSpecifier is given in Chapter B.5.1.

The application code then calls the *ReportCheckpoint* API (defined in [14, SWS Platform Health Management]) of the *SupervisedEntity* (which has been constructed in the context of the respective RPortPrototype typed by the PhmSupervisedEntityInterface) in order to notify the Platform Health Management that a specific PhmCheckpoint has been reached in the program flow.

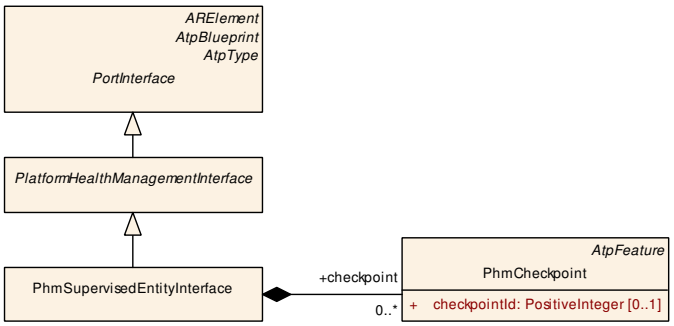


Figure 4.27: Modeling of Supervised Entities and Checkpoints

[constr_3530] Mandatory definition of checkpointId

Imposition time: IT_BefAraApiGen

⌈The checkpointId shall be defined for every PhmCheckpoint element.⌋

The checkpointId is used during the call to the *ReportCheckpoint* API as a representation of the PhmCheckpoint.

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 12.3.3.

The Platform Health Management functional cluster is only able to supervise Executables if these Executables are reporting their Execution State to the Execution Management. The reporting is defined using the attribute Executable.reportingBehavior.

If an Executable has Executable.reportingBehavior set to ExecutionStateReportingBehaviorEnum.doesNotReportExecutionState, then none of the Executable SwComponentPrototypes shall be involved in PHM supervision.

Class	**PhmSupervisedEntityInterface**			
Note	This meta-class provides the ability to implement a PortInterface for interaction with the Platform Health Management Supervised Entity. **Tags:** atp.recommendedPackage=PlatformHealthManagementInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PlatformHealthManagementInterface*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checkpoint	PhmCheckpoint	*	aggr	Defines the set of checkpoints which can be reported on this supervised entity. **Tags:** atp.Splitkey=checkpoint, checkpoint.shortName

Table 4.51: PhmSupervisedEntityInterface

Class	**PhmCheckpoint**			
Note	This meta-class provides the ability to implement a checkpoint for interaction with the Platform Health Management Supervised Entity. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpFeature*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, PhmSupervisedEntityInterface.checkpoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checkpointId	PositiveInteger	0..1	attr	Defines the numeric value which is used to indicate the reporting of this Checkpoint to the Phm.

Table 4.52: PhmCheckpoint

[constr_10151] Multiplicity of reference in the role PhmCheckpoint.checkpointId

Imposition time: IT_BefAraApiGen

⌈For each PhmCheckpoint, the reference in the role checkpointId shall exist.⌋

[constr_3795] No Platform Health Management supervision for non-reporting Executables

Imposition time: IT_BefAraApiGen

⌈If an Executable has Executable.reportingBehavior set to ExecutionStateReportingBehaviorEnum.doesNotReportExecutionState, then none of the SwComponentTypes that are used to type SwComponentPrototype used in the composition definition of the rootSwComponentPrototype shall define an RPortPrototype typed by a PhmSupervisedEntityInterface.⌋

4.9.3 Recovery notification to State Management

The Phm monitors the reporting of Supervised Entities and Checkpoints. If violations occur, 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 4.28 illustrates the reporting of Supervised Entities by Application 1 and 2.

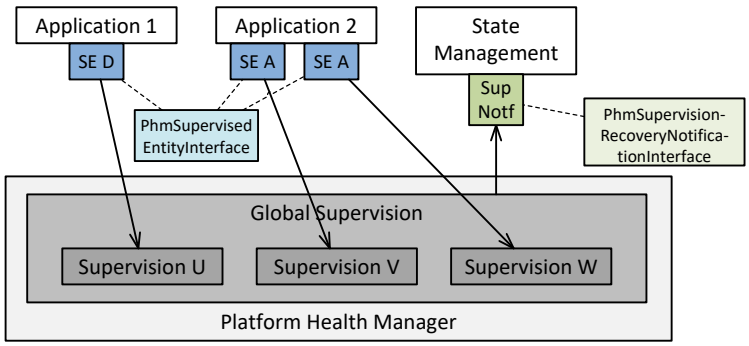


Figure 4.28: Example of a Phm monitoring and recovery setup

The Phm is configured to perform the supervision of these reported elements. If violations occur, the Phm is configured to notify the State Management application to deal with the situation.

[TPS_MANI_01280] Semantics of meta-class `PhmSupervisionRecoveryNotificationInterface` ⌈The recovery notification of a failed supervision by PHM 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`.⌋

The operation to be called by Phm in the context of [TPS_MANI_01280] is defined in the Platform Health Management specification document [14, SWS Platform Health Management].

As already mentioned, the State Management is intended to implement the recovery actions. This implies that the `PhmSupervisionRecoveryNotificationInterface` can only be used in combination with a `PPortPrototype`. This aspect is clarified by [constr_1729].

[constr_1729] `PhmSupervisionRecoveryNotificationInterface` only allowed for `PPortPrototype` on State Management software level

Imposition time: `IT_BefAraApiGen`

⌈Within the context of an `Executable` of `category APPLICATION_LEVEL`, the usage of `PhmSupervisionRecoveryNotificationInterface` is **only** supported for a `PPortPrototype`.⌋

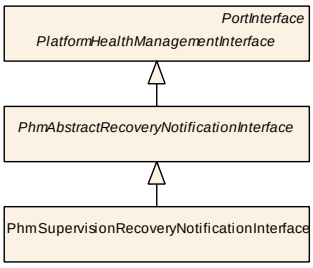


Figure 4.29: Modeling of the `PhmAbstractRecoveryNotificationInterface`

Class	**PhmSupervisionRecoveryNotificationInterface**			
Note	This meta-class represents a PortInterface that can be taken for implementing a PHM Supervision notification. **Tags:** atp.recommendedPackage=PlatformHealthManagementInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PhmAbstractRecoveryNotificationInterface*, *PlatformHealthManagementInterface*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.53: PhmSupervisionRecoveryNotificationInterface

Class	***PhmAbstractRecoveryNotificationInterface*** (abstract)			
Note	This abstract meta-class provides the abstract ability to define a PortInterface for the Recovery Notification by Platform Health Management. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PlatformHealthManagementInterface*, *PortInterface*, *Referrable*			
Subclasses	PhmSupervisionRecoveryNotificationInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.54: PhmAbstractRecoveryNotificationInterface

4.10 Diagnostic Interface

4.10.1 Overview

On the *AUTOSAR adaptive platform*, dedicated `PortInterface`s 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 `PortInterface`s and, by extension, the standardized `ara::diag` API **are only used on the application side** of this communication relation.

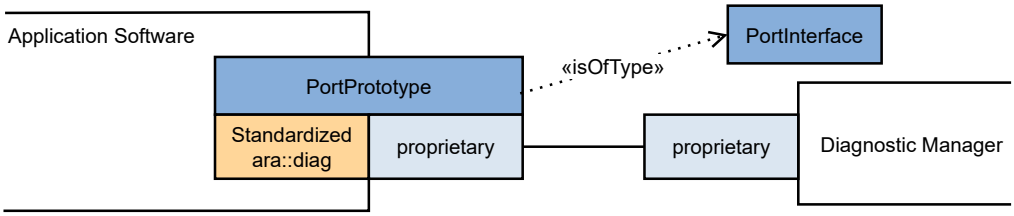


Figure 4.30: Standardized vs. proprietary parts in the implementation of `ara::diag`

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 4.30.



Figure 4.31: Modeling of `PortInterface`s for diagnostic purposes

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.

[TPS_MANI_01242] **`PortInterface`s used for communication with the AUTOSAR `Diagnostic Manager`** ⌈All `PortInterface`s used for this purpose are derived from the abstract meta-class `DiagnosticPortInterface`. A

DiagnosticPortInterface does not implement a service-oriented communication pattern; in particular, no explicit service discovery on the API level is involved.⌋

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 4.31 depicts all meta-classes that directly inherit from DiagnosticPortInterface.

Class	***DiagnosticPortInterface*** (abstract)			
Note	This meta-class serves as an abstract base-class for all diagnostics-related PortInterfaces. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	*DiagnosticAbstractDataIdentifierInterface*, *DiagnosticAbstractInputOutputControlInterface*, *DiagnosticAbstractRoutineInterface*, DiagnosticAuthenticationInterface, DiagnosticComControlInterface, DiagnosticConditionInterface, DiagnosticDTCInformationInterface, *DiagnosticDoIPAbstractCustomPayloadInterface*, DiagnosticDoIPActivationLineInterface, DiagnosticDoIPEntityIdentificationInterface, DiagnosticDoIPGroupIdentificationInterface, DiagnosticDoIPPowerModeInterface, DiagnosticDownloadInterface, DiagnosticEcuResetInterface, DiagnosticEventInterface, DiagnosticExtendedDataRecordInterface, DiagnosticExternalAuthenticationInterface, DiagnosticGenericUdsInterface, DiagnosticIndicatorInterface, DiagnosticMonitorInterface, *DiagnosticMultipleResourceInterface*, DiagnosticOperationCycleInterface, DiagnosticRequestFileTransferInterface, DiagnosticSecurityLevelInterface, DiagnosticServiceValidationInterface, *DiagnosticSovdPortInterface*, DiagnosticTransmitCertificateInterface, DiagnosticUploadInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.55: DiagnosticPortInterface

4.10.2 Diagnostic Interfaces for UDS

4.10.2.1 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.

The arguments to the diagnostic routine shall be modeled as the arguments of the respective ClientServerOperations aggregated in the roles start, stop, and requestResult.

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.

[constr_1696] ClientServerOperation aggregated by DiagnosticRoutineInterface

Imposition time: IT_BefAraApiGen

⌈Any ClientServerOperation aggregated by a DiagnosticRoutineInterface shall not define the following attributes:

- fireAndForget
- possibleApError
- possibleApErrorSet

⌋

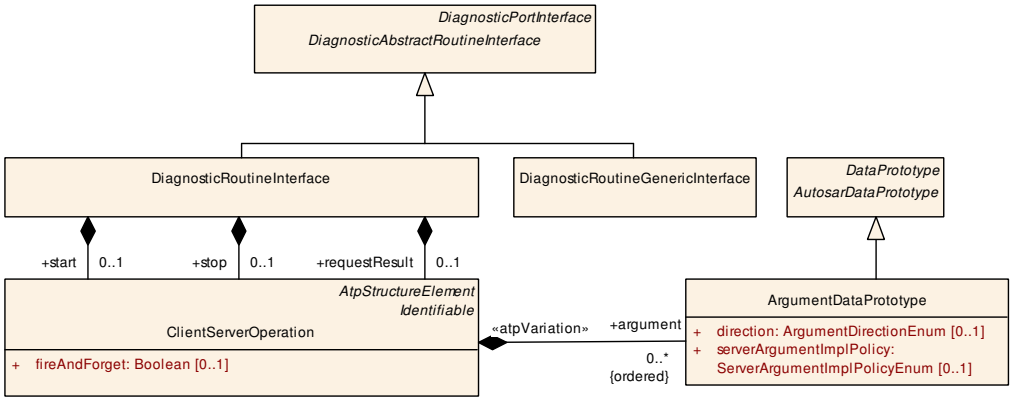


Figure 4.32: Modeling of DiagnosticRoutineInterface

This means that implicitly there are still up to three methods defined for the already mentioned roles of a diagnostic routine.

However, the methods in 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.

Class	**DiagnosticRoutineGenericInterface**
Note	This meta-class represents the ability to implement a generic Routine-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractRoutineInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*
Aggregated by	ARPackage.element

△

Class	**DiagnosticRoutineGenericInterface**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.56: DiagnosticRoutineGenericInterface

Class	**DiagnosticRoutineInterface**			
Note	This meta-class represents the ability to implement a routine-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractRoutineInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
requestResult	ClientServerOperation	0..1	aggr	This represents the request result method of the diagnostic routine. **Tags:** atp.Splitkey=requestResult, requestResult.shortName
start	ClientServerOperation	0..1	aggr	This represents the start method of the diagnostic routine. **Tags:** atp.Splitkey=start, start.shortName
stop	ClientServerOperation	0..1	aggr	This represents the stop method of the diagnostic routine. **Tags:** atp.Splitkey=stop, stop.shortName

Table 4.57: DiagnosticRoutineInterface

Meta-class `DiagnosticAbstractRoutineInterface` serves as the abstract base class to all routine-related `DiagnosticPortInterface`s on the *AUTOSAR adaptive platform*.

[constr_10031] Existence of `DiagnosticRoutineInterface.start`

Imposition time: `IT_BefAraApiGen`

⌈Attribute `DiagnosticRoutineInterface.start` shall exist.⌋

Class	***DiagnosticAbstractRoutineInterface*** (abstract)			
Note	This meta-class serves as the abstract base class of PortInterfaces dedicated to routine execution on the AUTOSAR adaptive platform. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	DiagnosticRoutineGenericInterface, DiagnosticRoutineInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.58: DiagnosticAbstractRoutineInterface

4.10.2.2 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 diagnostic-related data in the application software.

[TPS_MANI_01243] Semantics of DiagnosticDataIdentifierInterface
⌈DiagnosticDataIdentifierInterface 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.⌋

[constr_10030] Existence of DiagnosticDataIdentifierInterface.read

Imposition time: IT_BefAraApiGen

⌈Attribute DiagnosticDataIdentifierInterface.read shall exist.⌋

[constr_10634] ClientServerOperation aggregated by meta-class DiagnosticDataIdentifierInterface or meta-class DiagnosticDataElementInterface

Imposition time: IT_BefAraApiGen

⌈Any ClientServerOperation aggregated by a DiagnosticDataIdentifierInterface or a DiagnosticDataElementInterface shall **not** define the following attributes:

- fireAndForget
- possibleApError
- possibleApErrorSet

⌋

Class	***DiagnosticAbstractDataIdentifierInterface*** (abstract)
Note	This meta-class serves as the abstract base class of PortInterfaces dedicated to the access of diagnostic data identifiers on the AUTOSAR adaptive platform. This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*
Subclasses	DiagnosticDataElementInterface, DiagnosticDataIdentifierGenericInterface, DiagnosticDataIdentifierInterface
Aggregated by	ARPackage.element

▽

△

Class	***DiagnosticAbstractDataIdentifierInterface*** (abstract)			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.59: DiagnosticAbstractDataIdentifierInterface

Class	**DiagnosticDataIdentifierInterface**			
Note	This meta-class represents the ability to implement a DID-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractDataIdentifierInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
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.Splitkey=read, read.shortName
write	ClientServerOperation	0..1	aggr	This represents the method to write the contents of a diagnostic data identifier. **Tags:** atp.Splitkey=write, write.shortName

Table 4.60: DiagnosticDataIdentifierInterface

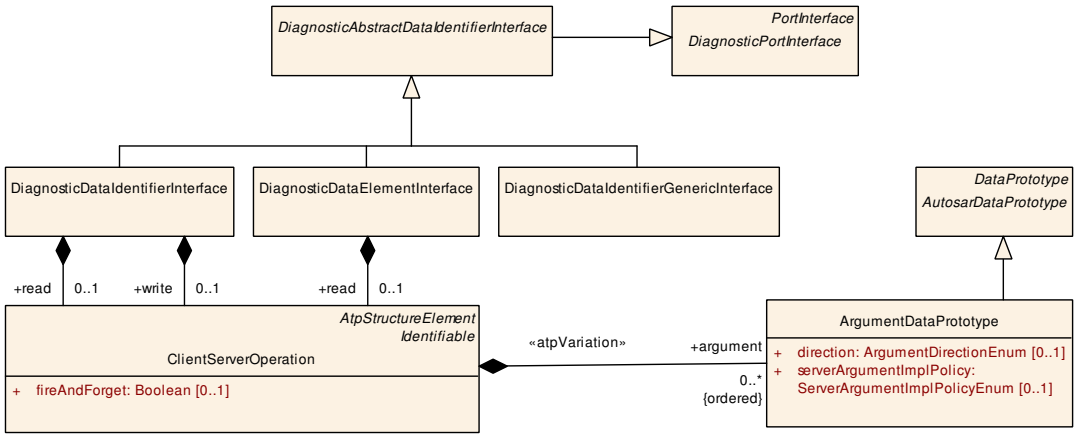


Figure 4.33: Modeling of `DiagnosticDataIdentifierInterface`

[TPS_MANI_01244] Semantics of meta-class `DiagnosticDataElement-Interface` ⌈`DiagnosticDataElementInterface` is used to access the content of an element within a given DID.

For this purpose, a `ClientServerOperation` is aggregated in the role `read`.⌋

The `DiagnosticDataElementInterface` *intentionally* does not support a write operation because the consistency of the data in principle cannot be ensured if it is sent 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			
Note	This meta-class represents the ability to implement a element-of-DID-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractDataIdentifierInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
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.Splitkey=read, read.shortName

Table 4.61: DiagnosticDataElementInterface

[TPS_MANI_01245] Semantics of meta-class DiagnosticDataIdentifierGenericInterface ⌈DiagnosticDataIdentifierGenericInterface 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.⌋

Class	DiagnosticDataIdentifierGenericInterface			
Note	This meta-class represents the ability to implement a generic DID-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractDataIdentifierInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.62: DiagnosticDataIdentifierGenericInterface

It is necessary to put some restrictions on the argument **unless** a given DiagnosticDataIdentifierInterface aggregates **only one** ClientServerOperation in **either** the role read **or** write.

[constr_1697] Restriction for `ClientServerOperation` aggregated by a `DiagnosticDataIdentifierInterface`

Imposition time: `IT_BefAraApiGen`

⌈If meta-class `DiagnosticDataIdentifierInterface` aggregates two `ClientServerOperation`s, then

- The two `ClientServerOperation`s shall have the same number of `argument`s.
- The `argument`s on the nth position in the collection of `argument`s 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 the role `DiagnosticDataIdentifierInterface.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`.

⌋

4.10.2.3 Security Level Interface

[TPS_MANI_01251] Semantics of `DiagnosticSecurityLevelInterface` ⌈The usage of the `DiagnosticSecurityLevelInterface` is intended for software that implements the checks for the clearance of a given security level.⌋

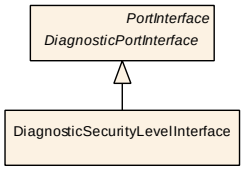


Figure 4.34: Modeling of `DiagnosticSecurityLevelInterface`

The `DiagnosticSecurityLevelInterface` does not require any further details in its formalization.

Class	**DiagnosticSecurityLevelInterface**
Note	This meta-class represents the ability to implement a security-level-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.

▽

△

Class	DiagnosticSecurityLevelInterface			
Base	ARElement, ARObject, AtpBlueprint, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 4.63: DiagnosticSecurityLevelInterface

4.10.2.4 Service Validation Interface

[TPS_MANI_01252] Semantics of `DiagnosticServiceValidationInterface` ⌈The usage of the `DiagnosticServiceValidationInterface` is foreseen for software that implements the checks for clearance on manufacturer or supplier level.⌋

The `DiagnosticServiceValidationInterface` does not require any further details in its formalization.

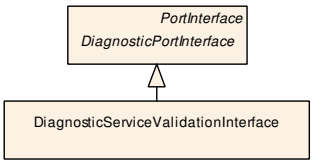


Figure 4.35: Modeling of `DiagnosticServiceValidationInterface`

Class	DiagnosticServiceValidationInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for service validation on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, AtpBlueprint, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 4.64: DiagnosticServiceValidationInterface

4.10.2.5 Generic UDS Interface

[TPS_MANI_01254] Semantics of `DiagnosticGenericUdsInterface` ⌈The AUTOSAR diagnostic communication API also foresees the existence of one `DiagnosticPortInterface` that supports the implementation of a completely generic handler of a UDS service.⌋

Class	**DiagnosticGenericUdsInterface**			
Note	This meta-class represents the ability to implement a generic UDS PortInterface for diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.65: DiagnosticGenericUdsInterface

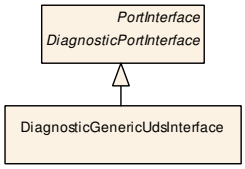


Figure 4.36: Modeling of `DiagnosticGenericUdsInterface`

The `DiagnosticGenericUdsInterface` does not require any further details in its formalization.

4.10.2.6 Diagnostic Interfaces for Upload and Download

[TPS_MANI_01265] Semantics of `DiagnosticUploadInterface` and `DiagnosticDownloadInterface` ⌈The AUTOSAR diagnostic communication API also foresees the existence of `DiagnosticPortInterface`s 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`

⌋

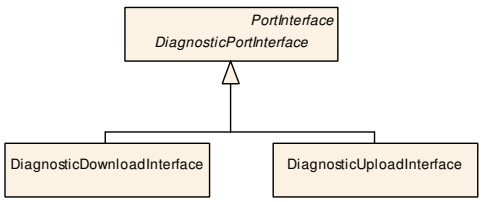


Figure 4.37: Modeling of `DiagnosticUploadInterface` and `DiagnosticDownloadInterface`

The `DiagnosticUploadInterface` and `DiagnosticDownloadInterface` do not require any further details in its formalization.

Class	**DiagnosticUploadInterface**			
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.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.66: DiagnosticUploadInterface

Class	**DiagnosticDownloadInterface**			
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.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.67: DiagnosticDownloadInterface

4.10.2.7 Interface to support managing the EcuReset

[TPS_MANI_01332] Semantics of `DiagnosticEcuResetInterface` ⌈Meta-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.⌋

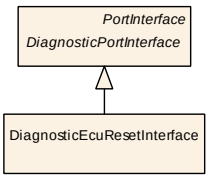


Figure 4.38: Modeling of `DiagnosticEcuResetInterface`

Class	**DiagnosticEcuResetInterface**			
Note	This meta-class represents the ability to implement a focused PortInterface for handling the diagnostic service EcuReset on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.68: DiagnosticEcuResetInterface

4.10.2.8 Diagnostic Authentication Interface

[TPS_MANI_01359] Semantics of `DiagnosticAuthenticationInterface` ⌈The ability to support the diagnostic authentication in the application software is formalized in another sub-class of `DiagnosticPortInterface`: `DiagnosticAuthenticationInterface`.⌋

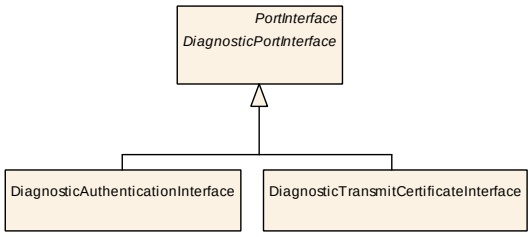


Figure 4.39: Modeling of `DiagnosticAuthenticationInterface`

Class	**DiagnosticAuthenticationInterface**			
Note	This meta-class represents the ability to implement a focused PortInterface for handling the diagnostic service "authentication" on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.69: DiagnosticAuthenticationInterface

[TPS_MANI_01452] Semantics of meta-class`DiagnosticTransmitCertificateInterface` ⌈The `DiagnosticTransmitCertificateInterface` can be taken to type a `PPortPrototype` over which the application software can manage the reception of new certificates.⌋

[constr_10526] DiagnosticAuthenticationPortMapping refers to a PPortPrototype that is typed by a DiagnosticTransmitCertificateInterface

Imposition time: IT_DiagDes

⌈If a DiagnosticAuthenticationPortMapping refers in the role pPortPrototypeInExecutable to a PPortPrototype that is typed by a DiagnosticTransmitCertificateInterface, then the DiagnosticAuthenticationPortMapping shall only refer in the role diagnosticAuthentication to a DiagnosticAuthTransmitCertificate (and vice versa).⌋

There are different ways to allow for the same PPortPrototype typed by a DiagnosticTransmitCertificateInterface to handle more than one evaluationId:

- One DiagnosticAuthenticationPortMapping exists that refers to a DiagnosticAuthTransmitCertificate that in turn aggregates more than one certificateEvaluation, each with a different setting of attribute evaluationId.
- Different DiagnosticAuthenticationPortMappings exist that each refer to the same PPortPrototype typed by a DiagnosticTransmitCertificateInterface and each refers to a different DiagnosticAuthTransmitCertificate that in turn aggregates only one certificateEvaluation.

Class	**DiagnosticTransmitCertificateInterface**			
Note	This meta-class represents the ability to implement the transmit-certificate functionality on application software level. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.70: DiagnosticTransmitCertificateInterface

4.10.2.9 Diagnostic External Authentication Interface

[TPS_MANI_01353] Semantics of DiagnosticExternalAuthenticationInterface ⌈The ability to support the authentication of a diagnostic client in the application software is formalized in another sub-class of DiagnosticPortInterface: DiagnosticExternalAuthenticationInterface.⌋

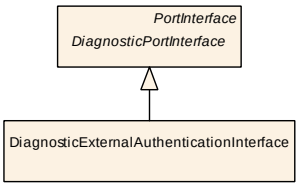


Figure 4.40: Modeling of `DiagnosticExternalAuthenticationInterface`

Class	**DiagnosticExternalAuthenticationInterface**			
Note	This meta-class represents the ability to implement a focused PortInterface for handling the diagnostic client authentication (i.e. convey the Authentication state to the Diagnostic Server instance of the DM) on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.71: DiagnosticExternalAuthenticationInterface

4.10.2.10 Diagnostic Communication Control Interface

[TPS_MANI_01363] Semantics of `DiagnosticComControlInterface` ⌈The ability to support the activation and deactivation of communication with a diagnostic client is formalized in another sub-class of `DiagnosticPortInterface`: `DiagnosticComControlInterface`.⌋

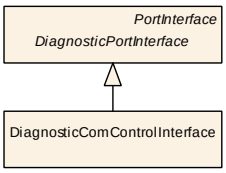


Figure 4.41: Modeling of `DiagnosticComControlInterface`

Class	**DiagnosticComControlInterface**			
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.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.72: DiagnosticComControlInterface

4.10.2.11 Diagnostic Request File Transfer Interface

[TPS_MANI_01373] Semantics of meta-class `DiagnosticRequestFileTransferInterface` ⌈The usage of the `DiagnosticRequestFileTransferInterface` is foreseen for software that implements a handler for requests for file transfer.⌋

The `DiagnosticRequestFileTransferInterface` does not require any further details in its formalization.

Class	**DiagnosticRequestFileTransferInterface**			
Note	This meta-class represents the ability to implement a PortInterface to process requests for file transfer using diagnostic channels on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.73: DiagnosticRequestFileTransferInterface

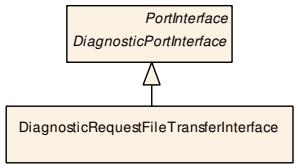


Figure 4.42: Modeling of `DiagnosticRequestFileTransferInterface`

4.10.2.12 Diagnostic Input/Output Control Interface

[TPS_MANI_01525] Semantics of `DiagnosticInputOutputControlInterface` ⌈The usage of the `DiagnosticInputOutputControlInterface` is foreseen for application software that implements a typed handler for I/O control requests.⌋

[TPS_MANI_01526] Semantics of `DiagnosticInputOutputControlGenericInterface` ⌈The usage of the `DiagnosticInputOutputControlGenericInterface` is foreseen for application software that implements a generic handler for I/O control requests.

The term “generic” means that for the purpose of executing an I/O control request, methods will be defined with a control 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⌋

The DiagnosticInputOutputControlInterface does not require any further details in its formalization.

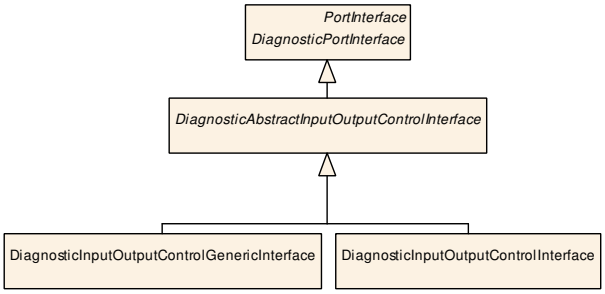


Figure 4.43: Modeling of DiagnosticInputOutputControlInterface

Class	***DiagnosticAbstractInputOutputControlInterface*** (abstract)			
Note	This meta-class serves as an abstract base class for PortInterfaces related to the UDS service 0x2F (I/O Control). This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	DiagnosticInputOutputControlGenericInterface, DiagnosticInputOutputControlInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.74: DiagnosticAbstractInputOutputControlInterface

Class	**DiagnosticInputOutputControlInterface**			
Note	This meta-class represents the ability to implement an I/O Control PortInterface for UDS diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractInputOutputControlInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.75: DiagnosticInputOutputControlInterface

Class	**DiagnosticInputOutputControlGenericInterface**
Note	This meta-class represents the ability to implement a generic I/O-Control-focused PortInterface for UDS diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractInputOutputControlInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*
Aggregated by	ARPackage.element

△

Class	DiagnosticInputOutputControlGenericInterface			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.76: DiagnosticInputOutputControlGenericInterface

4.10.3 Diagnostic Interfaces for Fault Memory

4.10.3.1 Interface for providing Data to Extended Data Record

The ability to contribute data for the creation of an extended data record from within the application software is provided by means of the `DiagnosticExtendedDataRecordInterface`.

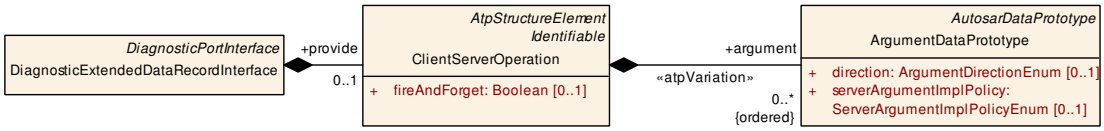


Figure 4.44: Modeling of `DiagnosticExtendedDataRecordInterface`

[constr_10682] Restriction of the multiplicity of `DiagnosticExtendedDataRecordInterface.provide.argument`

Imposition time: `IT_BefAraApiGen`

⌈A `ClientServerOperation` aggregated in the role `DiagnosticExtendedDataRecordInterface.provide` shall only aggregate exactly one `argument`.⌋

[constr_10683] Restriction for the data type of `DiagnosticExtendedDataRecordInterface.provide.argument`

Imposition time: `IT_BefAraApiGen`

⌈`DiagnosticExtendedDataRecordInterface.provide.argument` shall only be typed by **either**

- an `ApplicationPrimitiveDataType` **or**
- an `ApplicationArrayDataType` where attribute `element` is typed by an `ApplicationPrimitiveDataType` **or**
- a `CppImplementationDataType` where attribute `category` is set to the value `VALUE` **or**
- a `CppImplementationDataType` where attribute `category` is set to the value `ARRAY` that defines exactly one `subElement` where attribute `category` is set to the value `VALUE`.

⌋

Class	**DiagnosticExtendedDataRecordInterface**			
Note	This meta-class represents the ability to implement a focused PortInterface for providing values to an extended data record from within an application. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
provide	ClientServerOperation	0..1	aggr	This represents the method to proovide the content of an element of a diagnostic extended data record. **Tags:** atp.Splitkey=provide, provide.shortName

Table 4.77: DiagnosticExtendedDataRecordInterface

4.10.3.2 Interface to diagnostic Events

AUTOSAR defines several subclasses of `DiagnosticPortInterface` that are dedicated to the handling of diagnostic events.

[TPS_MANI_01246] Semantics of `DiagnosticMonitorInterface` ⌈Meta-class `DiagnosticMonitorInterface` represents the ability to report diagnostic events to the AUTOSAR `Diagnostic Manager`.⌋

Class	**DiagnosticMonitorInterface**			
Note	This meta-class represents the ability to implement a monitor-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.78: DiagnosticMonitorInterface

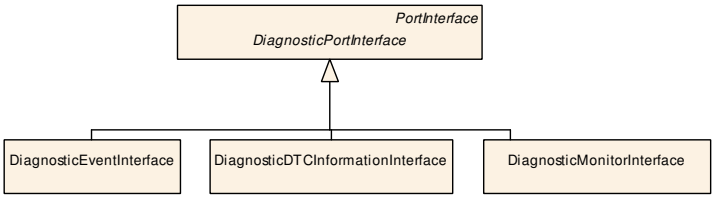


Figure 4.45: Modeling of `DiagnosticEventInterface`

[TPS_MANI_01247] Semantics of `DiagnosticDTCInformationInterface` ⌈Meta-class `DiagnosticDTCInformationInterface` represents the ability to retrieve information about a given diagnostic trouble code.⌋

Class	DiagnosticDTCInformationInterface			
Note	This meta-class represents the ability to implement a PortInterface to access the properties of DTCs on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 4.79: DiagnosticDTCInformationInterface

[TPS_MANI_01248] **Semantics of `DiagnosticEventInterface`** ⌈Meta-class `DiagnosticEventInterface` represents the ability to retrieve information about a given diagnostic event.⌋

Class	DiagnosticEventInterface			
Note	This meta-class represents the ability to implement a PortInterface to access the properties of diagnostic events on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 4.80: DiagnosticEventInterface

4.10.3.3 Interface to diagnostic Condition

[TPS_MANI_01249] Semantics of `DiagnosticConditionInterface` ⌈AUTOSAR 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*.⌋

The `DiagnosticConditionInterface` does not get specific about what the role[3] of the condition would be.

This understanding of the role is in general not known to the application which just provides the status of the condition, but does not know whether the condition will be used as an enable condition or clear condition.

This is the reason why there are no dedicated `PortInterface`s for enable condition and clear condition.

[3] enable condition vs. clear condition

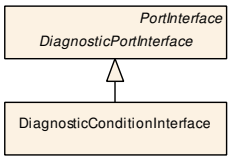


Figure 4.46: Modeling of `DiagnosticConditionInterface`

The `DiagnosticConditionInterface` does not require any further details in its formalization.

Class	**DiagnosticConditionInterface**			
Note	This meta-class represents the ability to implement a PortInterface to process requests for diagnostic conditions on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.81: DiagnosticConditionInterface

4.10.3.4 Interface for multiple diagnostic Resources

There are cases where software on application level needs to handle anonymous diagnostic resources, namely `DiagnosticEvent`s and `DiagnosticCondition`s in bulk and without knowing the identity of the specific resources at the time when the application is designed.

This means that the application would define a single `PortPrototype` (along with an appropriate `PortInterface`) over which e.g. the reporting of bulk diagnostic events is executed. This section defines the `PortInterface`s used for the purpose of handling bulk diagnostic resources.

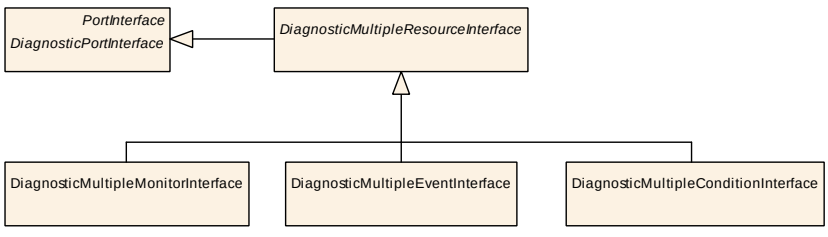


Figure 4.47: Modeling of `DiagnosticMultipleResourceInterface`

The definition of these `PortInterface`s serves the special use case discussed above and therefore does not represent a “generic” replacement for the existing `PortInterface`s over which single `DiagnosticEvent`s or `DiagnosticCondition`s are handled.

AUTOSAR defines an abstract subclass of DiagnosticPortInterface named DiagnosticMultipleResourceInterface that is acting as a base class for DiagnosticPortInterfaces that affect multiple diagnostic resources (e.g. diagnostic events) at the same time.

Class	**DiagnosticMultipleResourceInterface** (abstract)			
Note	This meta-class acts as an abstract base class for diagnostic port interfaces that affect multiple diagnostic resources (e.g. multiple events in one port interface). This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	DiagnosticMultipleConditionInterface, DiagnosticMultipleEventInterface, DiagnosticMultipleMonitorInterface			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.82: DiagnosticMultipleResourceInterface

4.10.3.4.1 Interface for multiple diagnostic Events

Two subclasses of DiagnosticMultipleResourceInterface are dedicated to the handling of diagnostic events:

- DiagnosticMultipleMonitorInterface provides the ability to report the status of multiple diagnostic monitors in the context of one PortInterface.
- DiagnosticMultipleEventInterface provides the ability to retrieve the status of multiple diagnostic events in the context of one PortInterface.

Class	**DiagnosticMultipleMonitorInterface**			
Note	This meta-class represents the ability to implement a monitor-focused PortInterface for diagnostics on the adaptive platform. In contrast to the DiagnosticMonitorInterface, the DiagnosticMultipleMonitor Interface allows for handling more than one event in the scope of a single PortPrototype. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticMultipleResourceInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.83: DiagnosticMultipleMonitorInterface

Class	DiagnosticMultipleEventInterface			
Note	This meta-class represents the ability to implement a event-focused PortInterface for diagnostics on the adaptive platform. In contrast to the DiagnosticEventInterface, the DiagnosticMultipleMonitorInterface allows for handling more than one event in the scope of a single PortPrototype. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticMultipleResourceInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.84: DiagnosticMultipleEventInterface

4.10.3.4.2 Interface for multiple diagnostic Conditions

As documented in Figure 4.47, another subclass of `DiagnosticMultipleResourceInterface`, the `DiagnosticMultipleConditionInterface` is dedicated to the ability to handle multiple diagnostic conditions in the context of one `PortInterface`.

Class	DiagnosticMultipleConditionInterface			
Note	This meta-class represents the ability to implement a condition-focused PortInterface for diagnostics on the adaptive platform. In contrast to the DiagnosticConditionInterface, the DiagnosticMultipleCondition Interface allows for handling more than one condition in the scope of a single PortPrototype. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticMultipleResourceInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.85: DiagnosticMultipleConditionInterface

4.10.3.5 Indicator Interface

[TPS_MANI_01250] Semantics of `DiagnosticIndicatorInterface` ⌈The usage of the `DiagnosticIndicatorInterface` is foreseen for software that implements a diagnostic indicator (i.e. a warning light on the dashboard).⌋

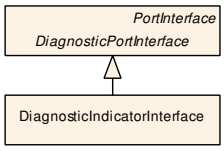


Figure 4.48: Modeling of `DiagnosticIndicatorInterface`

Class	DiagnosticIndicatorInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement indicator functionality on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 4.86: DiagnosticIndicatorInterface

The `DiagnosticIndicatorInterface` does not require any further details in its formalization.

4.10.3.6 Operation Cycle Interface

[TPS_MANI_01253] Semantics of `DiagnosticOperationCycleInterface` ⌈The usage of the `DiagnosticOperationCycleInterface` is foreseen for software that implements the manages the operation cycles.⌋

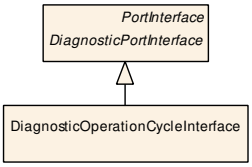


Figure 4.49: Modeling of `DiagnosticOperationCycleInterface`

Class	DiagnosticOperationCycleInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for operation cycles on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 4.87: DiagnosticOperationCycleInterface

The `DiagnosticOperationCycleInterface` does not require any further details in its formalization.

4.10.4 Diagnostic Interfaces for SOVD

4.10.4.1 Overview

The support for SOVD requires the definition of a number of PortInterfaces. These are entirely modeled as sub-classes to the abstract base-class DiagnosticSovdPortInterface.

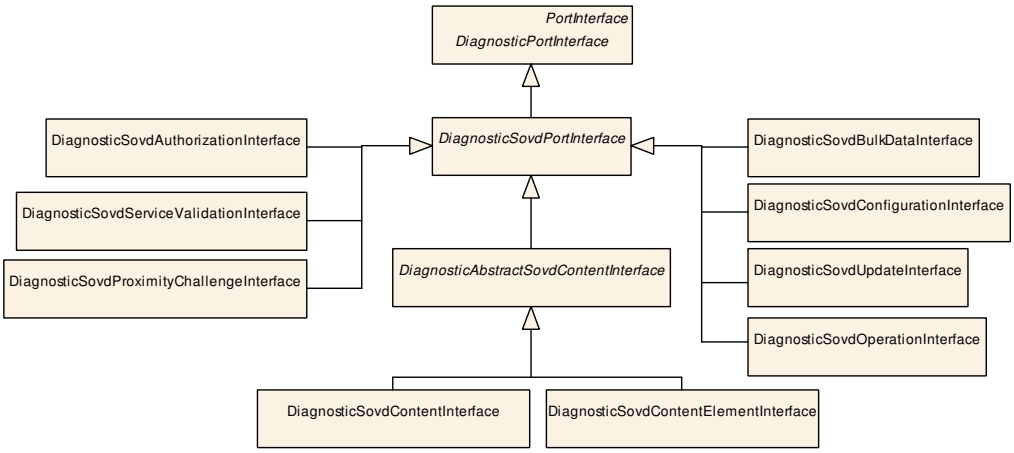


Figure 4.50: Modeling of sub-classes of DiagnosticSovdPortInterface

Class	***DiagnosticSovdPortInterface*** (abstract)			
Note	This abstract meta-class acts as a base class for all diagnostics-related PortInterfaces used in the context of SOVD. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	*DiagnosticAbstractSovdContentInterface*, DiagnosticSovdAuthorizationInterface, DiagnosticSovdBulkDataInterface, DiagnosticSovdConfigurationInterface, DiagnosticSovdOperationInterface, DiagnosticSovdProximityChallengeInterface, DiagnosticSovdServiceValidationInterface, DiagnosticSovdUpdateInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.88: DiagnosticSovdPortInterface

The various sub-classes of DiagnosticSovdPortInterface are explained in the following sub-chapters.

4.10.4.2 Port Interface for SOVD Client Authorization

[TPS_MANI_01394] Support for the authorization of SOVD clients ⌈The ability to support the authorization of SOVD clients is formalized in meta-class DiagnosticSovdAuthorizationInterface, modeled as a sub-class of abstract base class DiagnosticSovdPortInterface. Application software exposes a PPortPrototype for the support of authorization.⌋

Class	**DiagnosticSovdAuthorizationInterface**			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD authorization. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.89: DiagnosticSovdAuthorizationInterface

4.10.4.3 Port Interface for SOVD Proximity Challenge

[TPS_MANI_01395] Support for the proximity challenge in the context of SOVD
⌈The ability to support the proximity challenge in the context of SOVD is formalized in meta-class DiagnosticSovdProximityChallengeInterface, modeled as a sub-class of abstract base class DiagnosticSovdPortInterface. Application software exposes a PPortPrototype for the proximity challenge.⌋

Class	**DiagnosticSovdProximityChallengeInterface**			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD proximity challenge. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.90: DiagnosticSovdProximityChallengeInterface

4.10.4.4 Port Interface for SOVD Bulk Data

[TPS_MANI_01420] Support for the transport of bulk data in the context of SOVD
⌈The ability to transport bulk data in the context of SOVD is formalized in meta-class DiagnosticSovdBulkDataInterface, modeled as a sub-class of abstract base class DiagnosticSovdPortInterface. Application software exposes a PPort-Prototype for the bulk data transport.⌋

Class	**DiagnosticSovdBulkDataInterface**			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD bulk data transmission. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.91: DiagnosticSovdBulkDataInterface

4.10.4.5 Port Interface for SOVD Service Validation

[TPS_MANI_01421] Support for service validation in the context of SOVD ⌈The ability to execute a service validation in the context of SOVD is formalized in meta-class DiagnosticSovdServiceValidationInterface, modeled as a sub-class of abstract base class DiagnosticSovdPortInterface. Application software exposes a PPortPrototype for the service validation.⌋

Class	**DiagnosticSovdServiceValidationInterface**			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD service validation. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.92: DiagnosticSovdServiceValidationInterface

4.10.4.6 Port Interface for SOVD Update

[TPS_MANI_01422] Support for update procedures in the context of SOVD ⌈The ability to execute update procedures in the context of SOVD is formalized in meta-class DiagnosticSovdUpdateInterface, modeled as a sub-class of abstract base class DiagnosticSovdPortInterface. Application software exposes a PPort-Prototype for the update procedure.⌋

Class	**DiagnosticSovdUpdateInterface**			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD update procedure. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.93: DiagnosticSovdUpdateInterface

4.10.4.7 Port Interface for SOVD Configuration

[TPS_MANI_01423] Support for configuration in the context of SOVD ⌈The ability to execute configuration in the context of SOVD is formalized in meta-class DiagnosticSovdConfigurationInterface, modeled as a sub-class of abstract base class DiagnosticSovdPortInterface. Application software exposes a PPortPrototype for configuration purposes.⌋

Class	**DiagnosticSovdConfigurationInterface**			
Note	This meta-class is used to configure a PortInterface for the exchange of configuration content. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.94: DiagnosticSovdConfigurationInterface

4.10.4.8 Port Interface for SOVD Operation

[TPS_MANI_01487] Support for the execution of operations in the context of SOVD ⌈The ability to execute operations in the context of SOVD is formalized in meta-class DiagnosticSovdOperationInterface, modeled as a sub-class of abstract base class DiagnosticSovdPortInterface. The application software exposes a PPortPrototype for operation execution purposes.⌋

[TPS_MANI_01488] Modeling of the DiagnosticSovdOperationInterface ⌈Meta-class DiagnosticSovdOperationInterface aggregates DiagnosticSovdAccessOperation in three roles:

- `start` represents the ability to **start** the execution of the operation represented by the enclosing `DiagnosticSovdOperationInterface`. This aggregation is always needed in each `DiagnosticSovdOperationInterface`.
- `terminate` represents the ability to **terminate** the execution of the operation represented by the enclosing `DiagnosticSovdOperationInterface`.
- `readStatus` represents the ability to **query the status** of the execution of the operation represented by the enclosing `DiagnosticSovdOperationInterface`.

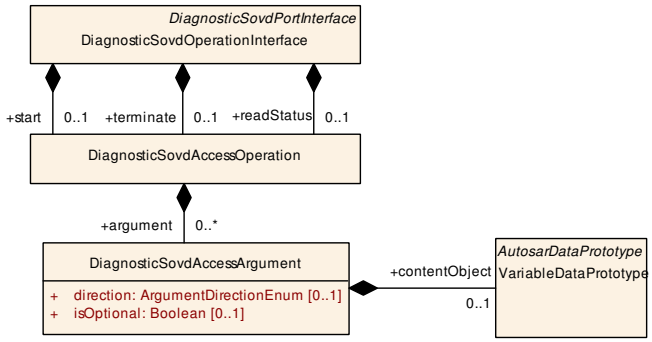


Figure 4.51: Modeling of sub-classes of `DiagnosticSovdOperationInterface`

Class	**DiagnosticSovdOperationInterface**			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD operation. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
readStatus	DiagnosticSovdAccessOperation	0..1	aggr	This role represents the "readStatus" primitive within an "SOVD operation" in the context of the enclosing DiagnosticSovdOperationInterface. **Tags:** atp.Splitkey=readStatus xml.sequenceOffset=120
start	DiagnosticSovdAccessOperation	0..1	aggr	This role represents the "start" primitive within an "SOVD operation" in the context of the enclosing DiagnosticSovdOperationInterface. **Tags:** atp.Splitkey=start xml.sequenceOffset=100

△

Class	DiagnosticSovdOperationInterface			
terminate	DiagnosticSovdAccessOperation	0..1	aggr	This role represents the "terminate" primitive within an "SOVD operation" in the context of the enclosing DiagnosticSovdOperationInterface. **Tags:** atp.Splitkey=terminate xml.sequenceOffset=110

Table 4.95: DiagnosticSovdOperationInterface

[constr_10692] Existence of aggregation in the role DiagnosticSovdOperationInterface.start

Imposition time: IT_BefAraApiGen

⌈For each DiagnosticSovdOperationInterface, the aggregation in the role start shall exist.⌋

Class	DiagnosticSovdAccessOperation			
Note	This meta-class represents a directional access to data for SOVD. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	DiagnosticSovdContentElementInterface.read, DiagnosticSovdContentInterface.read, DiagnosticSovdContentInterface.write, DiagnosticSovdOperationInterface.readStatus, DiagnosticSovdOperationInterface.start, DiagnosticSovdOperationInterface.terminate			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
argument	DiagnosticSovdAccessArgument	*	aggr	This riole represents a specific data element used for reading or writing with SOVD. **Tags:** atp.Splitkey=argument

Table 4.96: DiagnosticSovdAccessOperation

Class	DiagnosticSovdAccessArgument			
Note	This meta-class represents a single data element used in the context of reading or writing data using SOVD This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	DiagnosticSovdAccessOperation.argument			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
contentObject	VariableDataPrototype	0..1	aggr	This aggregation represents the ability to specify further information about the argument. **Tags:** atp.Splitkey=contentObject, contentObject.shortName
direction	ArgumentDirectionEnum	0..1	attr	This attribute specifies the direction of the argument.
isOptional	Boolean	0..1	attr	This attribute defines whether the enclosing DiagnosticSovdAccessArgument is considered optional in the context of reading or writing data using SOVD.

Table 4.97: DiagnosticSovdAccessArgument

[constr_10693] Existence of reference in the role DiagnosticSovdAccessArgument.contentObject

Imposition time: IT_BefAraApiGen

⌈For each DiagnosticSovdAccessArgument, the reference in the role contentObject shall exist.⌋

4.10.4.9 Port Interface for SOVD Content Access

[TPS_MANI_01489] Support for data access in the context of SOVD ⌈The ability to access data in the context of SOVD is formalized in meta-class DiagnosticSovdContentInterface and DiagnosticSovdContentElementInterface, modeled as a sub-class of abstract base class DiagnosticAbstractSovdContentInterface. The application software exposes a PPortPrototype for data access purposes.⌋

It is necessary to put some restrictions on the argument **unless** a given DiagnosticSovdContentInterface aggregates **only one** DiagnosticSovdAccessOperation in **either** the role read **or** write.

[constr_10694] Restriction for DiagnosticSovdAccessOperation aggregated by a DiagnosticSovdContentInterface

Imposition time: IT_BefAraApiGen

⌈If the meta-class DiagnosticSovdContentInterface aggregates the two supported roles of DiagnosticSovdAccessOperations then

- The two DiagnosticSovdAccessOperations 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 DiagnosticSovdAccessArgument aggregated by a DiagnosticSovdAccessOperation that is itself aggregated in the role DiagnosticSovdContentInterface.read shall set attribute direction to out.
  - Any DiagnosticSovdAccessArgument aggregated by a DiagnosticSovdAccessOperation that is itself aggregated in the role DiagnosticSovdContentInterface.write shall set attribute direction to in.

⌋

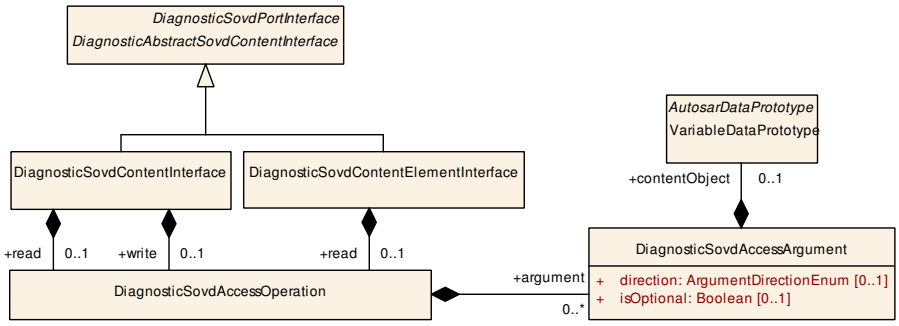


Figure 4.52: Modeling of sub-classes of `DiagnosticAbstractSovdContentInterface`

[constr_10695] Existence of `DiagnosticSovdAccessOperation.argument` if `DiagnosticSovdAccessOperation` that is aggregated in the role `DiagnosticSovdContentElementInterface.read`

Imposition time: `IT_BefAraApiGen`

⌈For each `DiagnosticSovdAccessOperation` that is aggregated in the role `DiagnosticSovdContentElementInterface.read`, the role `DiagnosticSovdAccessOperation.argument` shall exist exactly once and the attribute `argument.direction` shall be set to `ArgumentDirectionEnum.out`.⌋

Class	***DiagnosticAbstractSovdContentInterface*** (abstract)			
Note	This abstract meta-class acts as a base class for all data-related PortInterfaces used in the context of SOVD. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	DiagnosticSovdContentElementInterface, DiagnosticSovdContentInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.98: DiagnosticAbstractSovdContentInterface

Class	**DiagnosticSovdContentInterface**			
Note	This meta-class is used to type a PPortPrototype for reading and/or writing data using SOVD. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractSovdContentInterface*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
read	DiagnosticSovdAccessOperation	0..1	aggr	This role represents the read access to "SOVD data" in the context of the enclosing DiagnosticSovdContentInterface. **Tags:** atp.Splitkey=read

▽

△

Class	**DiagnosticSovdContentInterface**			
write	DiagnosticSovdAccessOperation	0..1	aggr	This role represents the write access to "SOVD data" in the context of the enclosing DiagnosticSovdContentInterface. **Tags:** atp.Splitkey=write

Table 4.99: DiagnosticSovdContentInterface

Class	**DiagnosticSovdContentElementInterface**			
Note	This meta-class is used to type a PPortPrototype for reading and/or writing a single data element as part of a bigger data packet using SOVD. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractSovdContentInterface*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
read	DiagnosticSovdAccessOperation	0..1	aggr	This role represents the read access to content elements in the context of the enclosing DiagnosticSovdContentElementInterface. **Tags:** atp.Splitkey=read

Table 4.100: DiagnosticSovdContentElementInterface

4.10.5 Diagnostic Interfaces for DoIP

[TPS_MANI_01255] Semantics of DoIP `DiagnosticPortInterface`s
⌈The AUTOSAR diagnostic communication API also foresees the existence of `DiagnosticPortInterface`s 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`
- `DiagnosticDoIPEntityIdentificationInterface`
- `DiagnosticDoIPCustomPayloadTxInterface`
- `DiagnosticDoIPCustomPayloadRxInterface`

⌋

Class	DiagnosticDoIPGroupIdentificationInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIP Group Identification on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.101: DiagnosticDoIPGroupIdentificationInterface

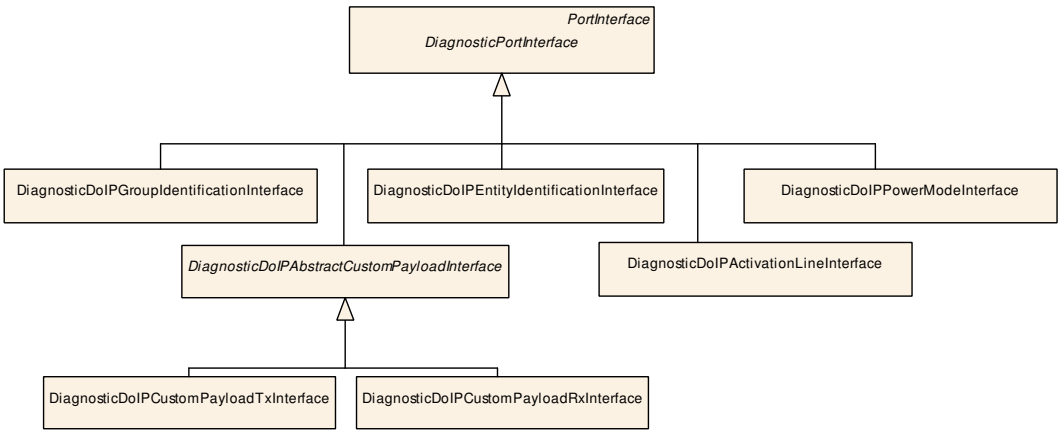


Figure 4.53: Modeling of DoIP `DiagnosticPortInterface`s

Class	DiagnosticDoIPPowerModeInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIP Power Mode on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.102: DiagnosticDoIPPowerModeInterface

Class	DiagnosticDoIPActivationLineInterface
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIPActivationLine on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*
Aggregated by	ARPackage.element

△

Class	DiagnosticDoIPActivationLineInterface			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.103: DiagnosticDoIPActivationLineInterface

Class	DiagnosticDoIPEntityIdentificationInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIP Entity Identification on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.104: DiagnosticDoIPEntityIdentificationInterface

Class	DiagnosticDoIPCustomPayloadRxInterface			
Note	This meta-class is used to type a `PPortPrototype` over which custom DoIp payloads can be received for further processing in the application software. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticDoIPAbstractCustomPayloadInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.105: DiagnosticDoIPCustomPayloadRxInterface

Class	DiagnosticDoIPCustomPayloadTxInterface			
Note	This meta-class is used to type an `RPortPrototype` over which custom DoIP payloads can be sent out by the application software. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticDoIPAbstractCustomPayloadInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.106: DiagnosticDoIPCustomPayloadTxInterface

Class	**DiagnosticDoIPAbstractCustomPayloadInterface** (abstract)			
Note	This meta-class serves as an abstract base class for all DiagnosticPortInterfaces that handle custom DoIP payloads. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	DiagnosticDoIPCustomPayloadRxInterface, DiagnosticDoIPCustomPayloadTxInterface			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.107: DiagnosticDoIPAbstractCustomPayloadInterface

The meta-classes

- DiagnosticDoIPGroupIdentificationInterface
- DiagnosticDoIPPowerModeInterface
- DiagnosticDoIPActivationLineInterface
- DiagnosticDoIPEntityIdentificationInterface
- DiagnosticDoIPCustomPayloadTxInterface
- DiagnosticDoIPCustomPayloadRxInterface

do not require any further details in their formalization.

4.11 Crypto Interfaces

4.11.1 Interaction with Crypto Software

[TPS_MANI_03253] Interaction with crypto software

Status: DRAFT

⌈Interaction 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.⌋

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 [18, SWS Cryptography]. 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 target-configuration model to a specific crypto object, e.g. CryptoKeySlot in a CryptoKeyStorage.

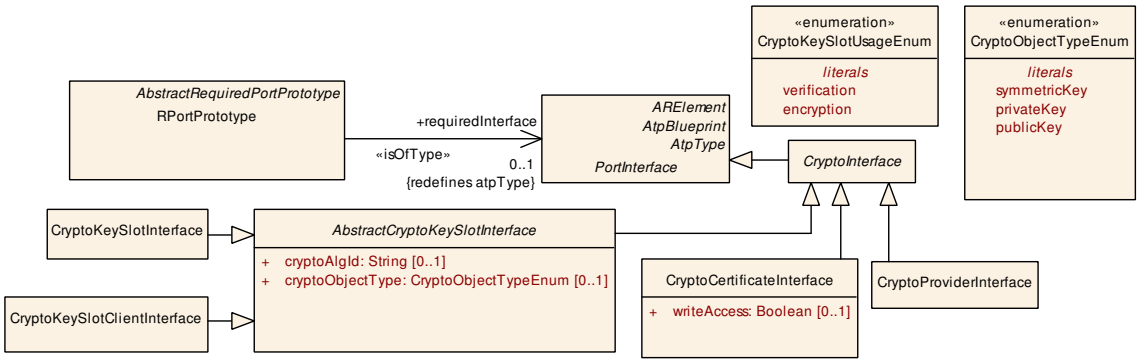


Figure 4.54: `CryptoInterface`s for modeling of the interaction of the Application with the Crypto software

Class	***CryptoInterface*** (abstract)			
Note	This meta-class provides the abstract ability to define a PortInterface for the support of crypto use cases. **Tags:** atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	*AbstractCryptoKeySlotInterface*, CryptoCertificateGroupInterface, CryptoCertificateInterface, CryptoProviderInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.108: CryptoInterface

Figure 4.54 depicts all meta-classes that directly inherit from `CryptoInterface`.

4.11.2 Crypto Key Slot Interface

[TPS_MANI_01466] Two ways to interact with crypto key slots

Status: DRAFT

⌈There are two ways in which the application software can interact with a `CryptoKeySlot`:

- The application software acts as the **owner** of the `CryptoKeySlot`. In this case an `RPortPrototype` typed by a **`CryptoKeySlotInterface`** is used.
- The application software is just the **user** (or client) of the `CryptoKeySlot`. In this case an `RPortPrototype` typed by a **`CryptoKeySlotClientInterface`** is used.

⌋

The assignment of a `CryptoKeySlot` to a `CryptoProvider` is described in the target-configuration model (Machine Manifest). With this mapping, the assignment of

the CryptoKeySlot to a CryptoPrimitive of a CryptoProvider is also established.

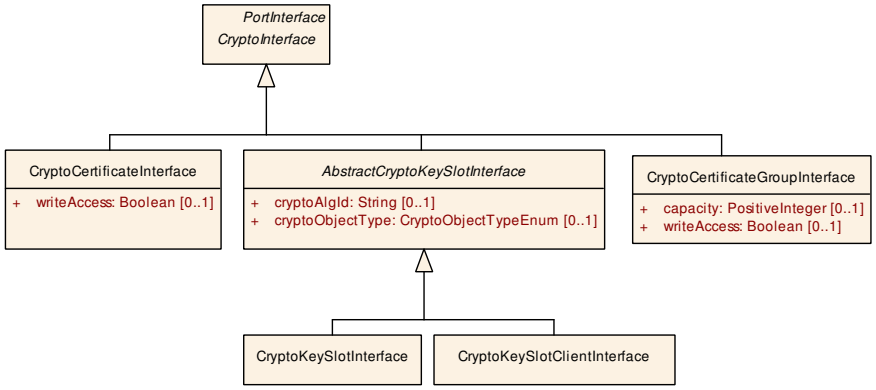


Figure 4.55: CryptoInterfaces for modeling of the interaction of the Application with the Crypto key slots and crypto certificates

To support crypto algorithms that are not well known or standardized today, AUTOSAR does not provide a concrete list of crypto algorithm identifiers and does not assume the usage of numerical identifiers.

Instead, the supplier of the crypto provider 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 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. The Application Design model settings are transferred into the target-configuration model and are therefore also available at run-time as described in Chapter 12.10.2.

Class	***AbstractCryptoKeySlotInterface*** (abstract)			
Note	This abstract meta-class acts as a base class for crypto key slot port interfaces. **Tags:** atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	CryptoKeySlotClientInterface, CryptoKeySlotInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	***AbstractCryptoKeySlotInterface*** (abstract)			
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=candidate
cryptoObject Type	CryptoObjectTypeEnum	0..1	attr	Object type that can be stored in the slot. **Tags:** atp.Status=candidate

Table 4.109: AbstractCryptoKeySlotInterface

Enumeration	**CryptoObjectTypeEnum**
Note	Enumeration of all types of crypto objects, i.e. types of content that can be stored to a key slot. **Tags:** atp.Status=candidate This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	*AbstractCryptoKeySlotInterface*.cryptoObjectType
Literal	***Description***
privateKey	cryp::PrivateKey object **Tags:** atp.EnumerationLiteralIndex=2 atp.Status=candidate
publicKey	cryp::PublicKey object **Tags:** atp.EnumerationLiteralIndex=3 atp.Status=candidate
symmetricKey	cryp::SymmetricKey object **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=candidate

Table 4.110: CryptoObjectTypeEnum

Class	**CryptoKeySlotClientInterface**			
Note	This meta-class provides the ability to define a PortInterface restricted to **using** Crypto Key Slots. **Tags:** atp.Status=candidate atp.recommendedPackage=CryptoInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractCryptoKeySlotInterface*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.111: CryptoKeySlotClientInterface

Class	**CryptoKeySlotInterface**			
Note	This meta-class provides the ability to define a PortInterface for **using and modifying** Crypto Key Slots. **Tags:** atp.Status=candidate atp.recommendedPackage=CryptoInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractCryptoKeySlotInterface*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.112: CryptoKeySlotInterface

4.11.3 Crypto Certificate Interface

[TPS_MANI_03256] Modeling of application that accesses a Crypto Certificate

Status: DRAFT

⌈Application software that accesses a Crypto Certificate is modeled as a `AdaptiveApplicationSwComponentType` with an `RPortPrototype` that is typed by a `CryptoCertificateInterface`.⌋

Class	**CryptoCertificateInterface**			
Note	This meta-class provides the ability to define a PortInterface for a CryptoCertificate. **Tags:** atp.Status=candidate atp.recommendedPackage=CryptoInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
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=candidate

Table 4.113: CryptoCertificateInterface

4.11.4 Crypto Certificate Group Interface

[TPS_MANI_01486] Modeling of application that accesses a Crypto Certificate Group

Status: DRAFT

⌈Application software that accesses a Crypto Certificate Group is modeled as a `AdaptiveApplicationSwComponentType` with an `RPortPrototype` that is typed by a `CryptoCertificateGroupInterface`.⌋

Class	**CryptoCertificateGroupInterface**			
Note	This meta-class provides the ability to define a PortInterface for a group of CryptoCertificates. **Tags:** atp.Status=candidate atp.recommendedPackage=CryptoInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
capacity	PositiveInteger	0..1	attr	Capacity of the certificate group in bytes of DER serialized certificates. **Tags:** atp.Status=candidate
writeAccess	Boolean	0..1	attr	This attribute defines whether the application has write-access to the CryptoCertificateGroup (true) or only read-access (false).

Table 4.114: CryptoCertificateGroupInterface

4.11.5 Crypto Provider Interface

[TPS_MANI_03257] Modeling of application that accesses a Crypto Provider

Status: DRAFT

⌈Application software that accesses a Crypto Provider is modeled as a `AdaptiveApplicationSwComponentType` with an `RPortPrototype` that is typed by a `CryptoProviderInterface`.⌋

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**			
Note	This meta-class provides the ability to define a PortInterface for a CryptoProvider. **Tags:** atp.Status=candidate atp.recommendedPackage=CryptoInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.115: CryptoProviderInterface

4.12 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. SOME/IP serialization does not make sense in such a scenario, nor would the modeling of `AutosarDataType`s, 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 `RawDataStream-ClientInterface` or a `RawDataStreamServerInterface`.

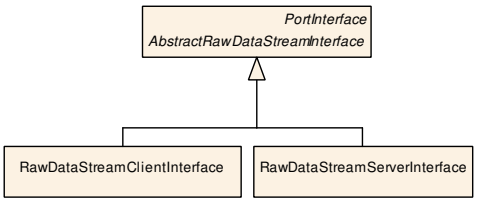


Figure 4.56: Modeling of `PortInterface`s for raw data stream

This kind of `PortInterface` does neither support nor require any elements with a modeled data type, i.e. an `AutosarDataType`.

The `RawDataStreamClientInterface` and `RawDataStreamServerInterface` do not expect any further knowledge on the underlying data transport. The associated APIs handle the connection and are able to send and receive any payload data.

If detailed protocol information and transport knowledge is required, further specialized derivations of meta-class `AbstractRawDataStreamInterface`, e.g. meta-classes `IEEE1722RawDataStreamConsumerInterface` and `IEEE1722 RawDataStreamProducerInterface` are available.

One aspect of this refinement is that more specialized APIs can be used for the interaction with the specific protocol details. Another aspect is that `RPortPrototype`

typed by a specialized `AbstractRawDataStreamInterface` may require specific `RawDataStreamMapping`s (see Chapter 14.2).

Class	***AbstractRawDataStreamInterface*** (abstract)			
Note	This meta-class serves as an abstract base class for PortInterfaces related to raw data streams. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	*MacLayerRawDataStreamInterface*, RawDataStreamClientInterface, RawDataStreamServerInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.116: AbstractRawDataStreamInterface

Class	**RawDataStreamClientInterface**			
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.recommendedPackage=RawDataStreamInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractRawDataStreamInterface*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.117: RawDataStreamClientInterface

Class	**RawDataStreamServerInterface**			
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.recommendedPackage=RawDataStreamInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractRawDataStreamInterface*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.118: RawDataStreamServerInterface

Class	***MacLayerRawDataStreamInterface*** (abstract)			
Note	This meta-class serves as an abstract base class for PortInterfaces related to MAC layer raw data streams. **Tags:** atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractRawDataStreamInterface*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	IEEE1722RawDataStreamConsumerInterface, IEEE1722RawDataStreamProducerInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.119: MacLayerRawDataStreamInterface

The meta-classes `IEEE1722RawDataStreamConsumerInterface` and `IEEE1722RawDataStreamProducerInterface` refine the abstract base class `AbstractRawDataStreamInterface` with specific focus on [19, IEEE1722] stream transport.

The specific APIs and handling of the IEEE1722 subtypes are defined in [20, SWS Raw Data Stream].

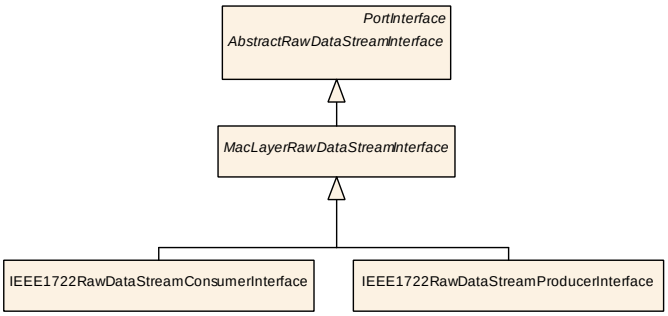


Figure 4.57: Modeling of `PortInterface`s for IEEE1722 raw data stream

Class	**IEEE1722RawDataStreamConsumerInterface**			
Note	This meta-class represents the necessary capabilities for IEEE1722 raw data streaming on the consumer side, i.e. the streaming of data that do not undergo any serialization. **Tags:** atp.Status=candidate atp.recommendedPackage=RawDataStreamInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractRawDataStreamInterface*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MacLayerRawDataStreamInterface*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.120: IEEE1722RawDataStreamConsumerInterface

Class	IEEE1722RawDataStreamProducerInterface			
Note	This meta-class represents the necessary capabilities for IEEE1722 raw data streaming on the producer side, i.e. the streaming of data that do not undergo any serialization. **Tags:** atp.Status=candidate atp.recommendedPackage=RawDataStreamInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractRawDataStreamInterface*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MacLayerRawDataStreamInterface*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.121: IEEE1722RawDataStreamProducerInterface

4.13 Intrusion Detection Interface

This section provides documentation for `PortInterface`s used in the context of intrusion detection. These `PortInterface`s are modeled as sub-classes of the abstract base class `IdsmAbstractPortInterface`.

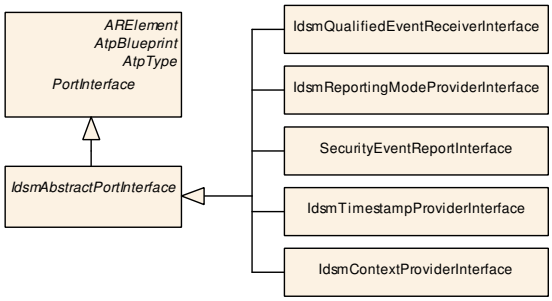


Figure 4.58: Abstract base-class `IdsmAbstractPortInterface`

Class	***IdsmAbstractPortInterface*** (abstract)			
Note	This abstract meta-class acts as a base class for all kinds of PortInterfaces related to security event handling. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	IdsmContextProviderInterface, IdsmQualifiedEventReceiverInterface, IdsmReportingModeProviderInterface, IdsmTimestampProviderInterface, SecurityEventReportInterface			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.122: IdsmAbstractPortInterface

4.13.1 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**			
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.recommendedPackage=SecurityEventReportInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *IdsmAbstractPortInterface*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.123: SecurityEventReportInterface

[TPS_MANI_01340] Semantics of `SecurityEventReportInterface` ⌈Each `RPortPrototype` typed by a `SecurityEventReportInterface` is able to report exactly one security event.⌋

4.13.2 Intrusion Detection Timestamp Provider Interface

[TPS_MANI_01439] Semantics of meta-class `IdsmTimestampProviderInterface` ⌈A `PPortPrototype` that is typed by a `IdsmTimestampProviderInterface` implements a *time-stamp provider* for the *Intrusion Detection System Management*.⌋

Class	**IdsmTimestampProviderInterface**			
Note	This meta-class provides the ability to define a PortInterface for providing a timestamp for security events in the context of the intrusion detection system. **Tags:** atp.recommendedPackage=IdsmPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *IdsmAbstractPortInterface*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.124: IdsmTimestampProviderInterface

4.13.3 Intrusion Detection Context Provider Interface

[TPS_MANI_01442] Semantics of meta-class `IdsmContextProviderInterface`
⌈A `PPortPrototype` that is typed by a `IdsmContextProviderInterface` implements a *context provider* for the *Intrusion Detection System Management*.⌋

Class	**IdsmContextProviderInterface**			
Note	This meta-class provides the ability to define a PortInterface for providing a Context for security events in the context of the intrusion detection system. **Tags:** atp.recommendedPackage=IdsmPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *IdsmAbstractPortInterface*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.125: IdsmContextProviderInterface

4.13.4 Intrusion Detection Reporting Mode Provider Interface

[TPS_MANI_01454] Semantics of meta-class `IdsmReportingModeProviderInterface`
⌈Each `RPortPrototype` typed by a `IdsmReportingModeProviderInterface` is able to set and get the reporting mode for security events.⌋

Class	**IdsmReportingModeProviderInterface**			
Note	This meta-class provides the ability to define a PortInterface for setting and getting the reporting mode for security events in the context of the intrusion detection system. **Tags:** atp.recommendedPackage=IdsmPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *IdsmAbstractPortInterface*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.126: IdsmReportingModeProviderInterface

4.13.5 Intrusion Detection Qualified Events Receiver Interface

[TPS_MANI_01475] Semantics of meta-class `IdsmQualifiedEventReceiverInterface`
⌈Each `RPortPrototype` typed by a `IdsmQualifiedEventReceiverInterface` is able to receive qualified security events.⌋

Class	**IdsmQualifiedEventReceiverInterface**			
Note	This meta-class provides the ability to define a PortInterface for receiving qualified security events in the context of the intrusion detection system. **Tags:** atp.recommendedPackage=IdsmPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *IdsmAbstractPortInterface*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.127: IdsmQualifiedEventReceiverInterface

4.14 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**			
Note	This meta-class provides the ability to implement a PortInterface for support of Logging or Tracing. **Tags:** atp.recommendedPackage=PortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.128: LogAndTraceInterface

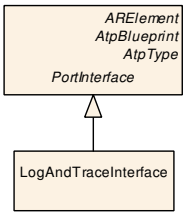


Figure 4.59: Specification of the `LogAndTraceInterface`

[TPS_MANI_03284] Semantics of `LogAndTraceInterface` ⌈Each `RPortPrototype` typed by a `LogAndTraceInterface` is able to forward logging information onto the external Dlt Log Viewer.⌋

[constr_5290] PPortPrototype is not allowed to be typed by LogAndTraceInterface

Imposition time: IT_BefAraApiGen

⌈A PPortPrototype is not allowed to reference a LogAndTraceInterface in the role providedInterface.⌋

The mapping of an RPortPrototype typed by a LogAndTraceInterface to a DltLogSink is described in the target configuration model in Chapter 12.6.2.

Class	**LTMessageCollectionToPortPrototypeMapping**			
Note	This mapping element assigns a collection of Log or Trace messages to a PortPrototype of an application. **Tags:** atp.recommendedPackage=LTMessageCollectionToPortPrototypeMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
logAndTrace Message CollectionSet	LogAndTraceMessage CollectionSet	0..1	ref	Reference to a Collection of Log or Trace messages **Tags:** atp.Splitkey=logAndTraceMessageCollectionSet
rPort	RPortPrototype	0..1	iref	Reference to the RPortPrototype to which Log or Trace messages are assigned. **Tags:** atp.Splitkey=rPort, rPort.contextRootSw ComponentPrototype, rPort.contextComponentPrototype, rPort.targetRPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 4.129: LTMessageCollectionToPortPrototypeMapping

Class	**LogAndTraceMessageCollectionSet**			
Note	Collection of DltMessages **Tags:** atp.recommendedPackage=LogAndTraceMessageCollectionSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dltMessage	DltMessage	*	aggr	Collection of DltMessages in the DltMessageCollection Set. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=dltMessage, dltMessage.shortName, dlt Message.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime

Table 4.130: LogAndTraceMessageCollectionSet

[constr_5291] Allowed usage of LTMessageCollectionToPortPrototype-Mapping.rPort

Imposition time: IT_BefAraApiGen

⌈An LTMessageCollectionToPortPrototypeMapping shall (in the role rPort) only refer to a RPortPrototype that is typed by a LogAndTraceInterface.⌋

[TPS_MANI_03285] **Semantics of meta-class `LTMessageCollectionToPortPrototypeMapping`** ⌈With the `LTMessageCollectionToPortPrototypeMapping` it is possible to describe at application design time a collection of Log or Trace messages that will be used by the application.⌋

4.15 Firewall Interface

On the *AUTOSAR adaptive platform*, a dedicated `PortInterface` named `FirewallStateSwitchInterface` is defined for the interaction of application-layer software with the Firewall Cluster.

Class	**FirewallStateSwitchInterface**			
Note	This meta-class provides the ability to implement a PortInterface for interaction with the Firewall mode. **Tags:** atp.recommendedPackage=FirewallStateSwitchPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
firewallStateMachine	ModeDeclarationGroupPrototype	*	aggr	The state machine of this firewall interface. **Tags:** atp.Splitkey=firewallStateMachine, firewallStateMachine.shortName

Table 4.131: FirewallStateSwitchInterface

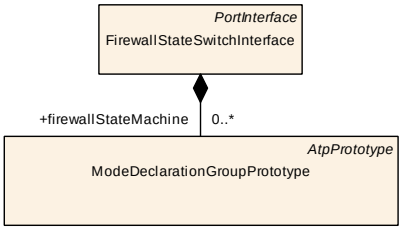


Figure 4.60: Specification of the `FirewallStateSwitchInterface`

[TPS_MANI_03321] **Semantics of `FirewallStateSwitchInterface`** ⌈The `RPortPrototype` typed by a `FirewallStateSwitchInterface` provides the functionality to switch the firewall state.⌋

[TPS_MANI_03295] **Semantics of `FirewallStateSwitchInterface.firewallStateMachine`** ⌈The `ModeDeclarationGroupPrototype` aggregated in the `FirewallStateSwitchInterface` in the role `firewallStateMachine` defines the state machine of the Firewall Cluster.⌋

[constr_10528] **Existence of `FirewallStateSwitchInterface.firewallStateMachine`**

Imposition time: `IT_BefAraApiGen`

⌈For any given `FirewallStateSwitchInterface`, attribute `firewallStateMachine` shall exist at most once.⌋

4.16 Network Management Interface

[TPS_MANI_01419] Semantics of NetworkManagementPortInterface

Status: DRAFT

⌈The NetworkManagementPortInterface is intended to be used by state management to interact with the network management for cases that are intentionally (e.g. because of the complexity of the rules to be applied for a specific case) not modeled in the context of StateManagementModuleInstantiation.⌋

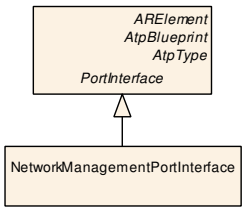


Figure 4.61: Specification of the NetworkManagementPortInterface

Class	**NetworkManagementPortInterface**			
Note	This PortInterface shall be used to submit triggers to the state management **Tags:** atp.recommendedPackage=NetworkManagementInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.132: NetworkManagementPortInterface

4.17 Suspend to RAM Interface

[TPS_MANI_01507] Semantics of AbstractSuspendToRamInterface

Status: DRAFT

⌈Subclasses of the AbstractSuspendToRamInterface are intended to be used by state management to interact with the suspend-to-RAM functionality.⌋

[TPS_MANI_01509] Semantics of SuspendToRamSatelliteInterface

Status: DRAFT

⌈The SuspendToRamSatelliteInterface is used to type PPortPrototypes in the application software in order to provide an endpoint for suspend-to-RAM related interaction with state management.⌋

[TPS_MANI_01511] Semantics of SuspendToRamHubInterface

Status: DRAFT

⌈The SuspendToRamHubInterface is used to type PPortPrototypes in the state management to provide an endpoint for suspend-to-RAM related interaction with application software.⌋

A sketch of the interaction between “hub” and “satellite” is depicted in Figure 12.85.

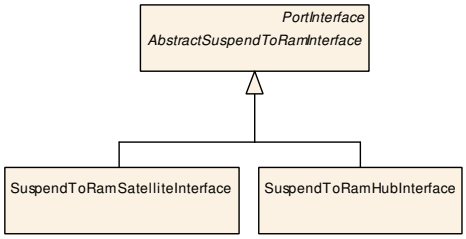


Figure 4.62: Specification of the AbstractSuspendToRamInterface

Class	***AbstractSuspendToRamInterface*** (abstract)			
Note	This meta-class provides an abstract base class for all sub-classes related to suspend-to-RAM functionality. **Tags:** atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	SuspendToRamHubInterface, SuspendToRamSatelliteInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.133: AbstractSuspendToRamInterface

Class	**SuspendToRamSatelliteInterface**			
Note	This meta-class represents a satellite-side PortInterface for the implementation of suspend-to-RAM functionality. **Tags:** atp.Status=candidate atp.recommendedPackage=SuspendToRamInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractSuspendToRamInterface*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.134: SuspendToRamSatelliteInterface

Class	**SuspendToRamHubInterface**			
Note	This meta-class represents a hub-side PortInterface for the implementation of suspend-to-RAM functionality. **Tags:** atp.Status=candidate atp.recommendedPackage=SuspendToRamInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractSuspendToRamInterface*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.135: SuspendToRamHubInterface

5 Diagnostic Design

The configuration of diagnostics on the *AUTOSAR adaptive platform* will typically be done by creating a Diagnostic Extract documented within the specification of the [3, TPS Diagnostic Extract Template] that is also used on the *AUTOSAR classic platform*.

Therefore, concepts within the Diagnostic Extract should be applicable uniformly to models on both platforms.

It can even be expected that a given Diagnostic Extract can be divided into parts that apply to ECUs built on top of the *AUTOSAR classic platform* and parts that apply to ECUs built on top of the *AUTOSAR adaptive platform*, all of which belong to the same vehicle.

The `PortInterface`s used for diagnostic purposes are documented in Chapter 4.10.

5.1 Extensions to the Diagnostic Common Props

The definition of the `DiagnosticCommonProps` is extended exclusively for the use on the *AUTOSAR adaptive platform*. The extension improves the description of authentication of diagnostic clients as well as the specification of diagnostic addresses.

Class	«atpVariation» **DiagnosticCommonProps**			
Note	This meta-class aggregates a number of common properties that are shared among a diagnostic extract. **Tags:** vh.latestBindingTime=codeGenerationTime			
Base	*ARObject*			
Aggregated by	DiagnosticContributionSet.commonProperties			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authentication Timeout	TimeValue	0..1	attr	This attribute defines the time (in seconds) that the authentication state is maintained in default-session if there is no communication from the authenticated client.
debounce AlgorithmProps	DiagnosticDebounce AlgorithmProps	*	aggr	Defines the used debounce algorithms relevant in the context of the enclosing DiagnosticCommonProps. Usually, there is a variety of debouncing algorithms to take into account and therefore the multiplicity of this aggregation is set to 0..*. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=debounceAlgorithmProps, debounce AlgorithmProps.shortName, debounceAlgorithm Props.variationPoint.shortLabel vh.latestBindingTime=postBuild
default Endianness	ByteOrderEnum	0..1	attr	Defines the default endianness of the data belonging to a DID or RID which is applicable if the DiagnosticData Element does not define the endianness via the swData DefProps.baseType attribute.

▽

△

Class	«atpVariation» **DiagnosticCommonProps**			
diagnostic Address	SoftwareCluster DiagnosticAddress	*	aggr	"This aggregation represents the collection of diagnostic addresses that apply for the SoftwareClusterDesign. **Tags:** atp.Splitkey=diagnosticAddress, diagnostic Address.componentQualifier, diagnostic Address.diagnosticAddress xml.namePlural=DIAGNOSTIC-ADDRESSES This Attribute is only used by the AUTOSAR Adaptive Platform.
event Combination Reporting Behavior	DiagnosticEvent CombinationReporting BehaviorEnum	0..1	attr	In case of EventCombination on Retrieval, this attribute specifies if a specific order of reporting is to be maintained.
external Authentication	DiagnosticExternal Authentication Identification	*	aggr	This aggregation supports the configuration of the authentication of diagnostic clients. **Tags:** atp.Splitkey=externalAuthentication This Attribute is only used by the AUTOSAR Adaptive Platform.
maxNumberOf Request Correctly Received Response Pending	PositiveInteger	0..1	attr	Maximum number of negative responses with response code 0x78 (requestCorrectlyReceived-ResponsePending) allowed per request. DCM will send a negative response with response code 0x10 (generalReject), in case the limit value gets reached. Value 0xFF means that no limit number of NRC 0x78 response apply.
occurrence Counter Processing	DiagnosticOccurrence CounterProcessing Enum	0..1	attr	This attribute defines the consideration of the fault confirmation process for the occurrence counter.
resetConfirmed BitOnOverflow	Boolean	0..1	attr	This attribute defines whether the confirmed bit is reset or not while an event memory entry will be displaced or could not acquire an entry in the event memory.
resetPendingBit OnOverflow	Boolean	0..1	attr	This attribute defines whether the pending bit is reset or not while an event memory entry will be displaced or could not acquire an entry in the event memory. In order to be compliant to ISO 14229-1 [21], this parameter needs to be set to "false".
responseOnAll RequestSids	Boolean	0..1	attr	If set to FALSE the DCM will not respond to diagnostic request that contains a service ID which is in the range from 0x40 to 0x7F or in the range from 0xC0 to 0xFF (Response IDs).
responseOn Second Declined Request	Boolean	0..1	attr	Defines the reaction upon a second request (ClientB) that can not be processed (e.g. due to priority assessment). TRUE: when the second request (Client B) can not be processed, it shall be answered with NRC21 BusyRepeat Request. FALSE: when the second request (Client B) can not be processed, it shall not be responded.
sovdProperties	DiagnosticSovdProps	0..1	aggr	This aggregation contains the definition of common properties that are specific for SOVD. **Tags:** atp.Splitkey=sovdProperties This Attribute is only used by the AUTOSAR Adaptive Platform.
typeOfEvent Combination Supported	DiagnosticEvent CombinationBehavior Enum	0..1	attr	Select type of Event Combination support.

Table 5.1: DiagnosticCommonProps

Class	**DiagnosticDebounceAlgorithmProps**			
Note	Defines properties for the debounce algorithm class.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DiagnosticCommonProps.debounceAlgorithmProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
debounce Algorithm	DiagEventDebounce Algorithm	0..1	aggr	This represents the actual debounce algorithm. **Tags:** atp.Splitkey=debounceAlgorithm, debounce Algorithm.shortName

Table 5.2: DiagnosticDebounceAlgorithmProps

[constr_10663] Existence of primitive attribute DiagnosticCommonProps.authenticationTimeout

Imposition time: IT_DiagDes

⌈If a DiagnosticContributionSet that aggregates the DiagnosticCommonProps also references a DiagnosticAuthenticationPortMapping or DiagnosticExternalAuthenticationPortMapping, then attribute DiagnosticCommonProps.authenticationTimeout shall exist.⌋

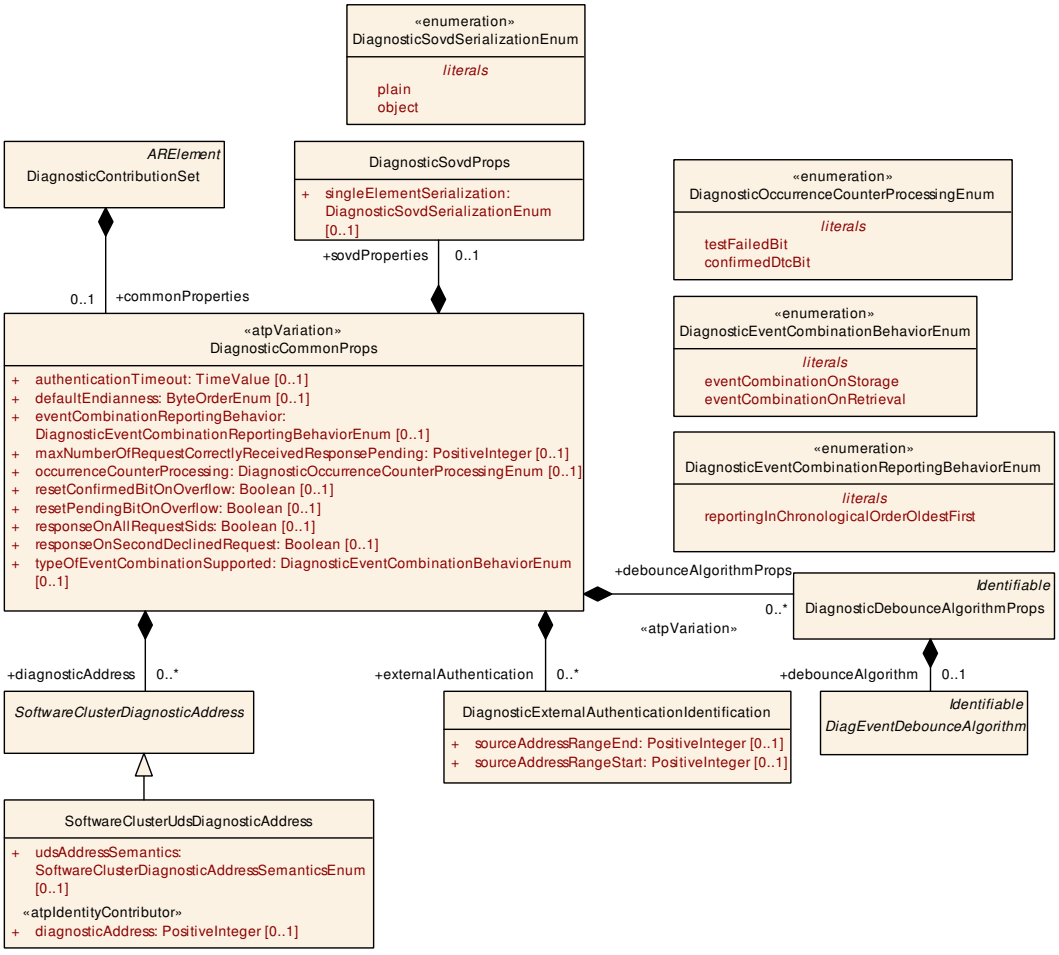


Figure 5.1: Modeling of DiagnosticCommonProps on the *AUTOSAR adaptive platform*

5.1.1 Diagnostic SOVD Props

Meta-class DiagnosticSovdProps collects all global-level attributes that have an impact on the configuration of SOVD.

[TPS_MANI_01514] Semantics of DiagnosticSovdProps.singleElement-Serialization ⌈Attribute DiagnosticSovdProps.singleElement-Serialization controls the behavior of the serialization (and, by extension, deserialization) of SOVD content for the case that only a single DiagnosticSovd-Content.element exists. This degree of freedom is officially supported by the [22, ISO-17978-3].

In principle, there are two different approaches for the serialization, as formalized in the definition of DiagnosticSovdSerializationEnum:

- The value DiagnosticSovdSerializationEnum.plain means that the value is provided “as is”, i.e. without wrapping it in curly braces. Example: “data”: 42
- The value DiagnosticSovdSerializationEnum.object means that the value is wrapped in curly braces. Example: “data”: {42}

⌋

Class	**DiagnosticSovdProps**			
Note	This meta-class collects common attributes that are specific for SOVD. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	DiagnosticCommonProps.sovdProperties			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
singleElement Serialization	DiagnosticSovd SerializationEnum	0..1	attr	This attribute defines the serialization strategy that is valid if and only if the contet consists of a single element.

Table 5.3: DiagnosticSovdProps

Enumeration	**DiagnosticSovdSerializationEnum**
Note	This enumeration provides possible values for the serialization approach in SOVD. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	DiagnosticSovdProps.singleElementSerialization
Literal	***Description***
object	The content is serialized as an object. **Tags:** atp.EnumerationLiteralIndex=1
plain	The content is serialized as a plain element. **Tags:** atp.EnumerationLiteralIndex=0

Table 5.4: DiagnosticSovdSerializationEnum

5.1.2 Diagnostic Address

[TPS_MANI_01434] Diagnostic Address defined in the context of a DiagnosticCommonProps ⌈It is possible to front-load the definition of diagnostic addresses as part of the configuration of diagnostic management.

Therefore, the definition of the DiagnosticCommonProps needs to provide information about the diagnostic address(es) to which the contained diagnostic management component shall respond.

This information is formalized by means of the attribute DiagnosticCommonProps.diagnosticAddress.

A DiagnosticCommonProps may be required to respond to multiple (i.e. several functional plus one physical) diagnostic addresses, thus the multiplicity of diagnosticAddress is set to 0..*.⌋

The goal of the specific modeling decision described in [TPS_MANI_01434] has been to make the modeling of the diagnostic address extensible such that the idiomatic ways to define diagnostic addresses can also be supported by adding respective subclasses of SoftwareClusterDiagnosticAddress.

Class	***SoftwareClusterDiagnosticAddress*** (abstract)			
Note	This meta-class represents the ability to define a diagnostic address in an abstract form. Sub-classes are supposed to clarify how the diagnostic address shall be defined according to the applicable addressing scheme (DoIP vs. CAN TP vs. ...). This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Subclasses	SoftwareClusterUdsDiagnosticAddress			
Aggregated by	DiagnosticCommonProps.diagnosticAddress			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 5.5: SoftwareClusterDiagnosticAddress

[constr_10487] Only one physical address per DiagnosticCommonProps

Imposition time: IT_DiagDes

⌈Each DiagnosticCommonProps shall only aggregate one SoftwareClusterDiagnosticAddress where the value of attribute udsAddressSemantics is set to SoftwareClusterDiagnosticAddressSemanticsEnum.physicalAddress.⌋

[TPS_MANI_01405] Semantics of sub-classes of meta-class SoftwareClusterDiagnosticAddress ⌈The definition of the actual diagnostic address is done by means of the applicable sub-class of the abstract base-class SoftwareClusterDiagnosticAddress: if the diagnostic session is executed on the basis of **UDS**, then the diagnostic address shall be specified by means of SoftwareClusterUdsDiagnosticAddress.diagnosticAddress.⌋

Enumeration	**SoftwareClusterDiagnosticAddressSemanticsEnum**
Note	This meta-class defines a list of semantics for the interpretation of diagnostic addresses in the context of a SoftwareCluster. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	SoftwareClusterUdsDiagnosticAddress.udsAddressSemantics
Literal	***Description***
functionalAddress	This address represents a functional address. **Tags:** atp.EnumerationLiteralIndex=1
physicalAddress	This address represents a physical address. **Tags:** atp.EnumerationLiteralIndex=0

Table 5.6: SoftwareClusterDiagnosticAddressSemanticsEnum

Class	**SoftwareClusterUdsDiagnosticAddress**			
Note	This meta-class represents the ability to define a diagnostic address specifically for the UDS case This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *SoftwareClusterDiagnosticAddress*			
Aggregated by	DiagnosticCommonProps.diagnosticAddress			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnostic Address	PositiveInteger	0..1	attr	This attribute represents the collection of diagnostic addresses the SoftwareCluster occupies. **Stereotypes:** atpIdentityContributor
udsAddress Semantics	SoftwareCluster DiagnosticAddress SemanticsEnum	0..1	attr	This attribute clarifies whether the address value shall be interpreted as a physical or a functional address.

Table 5.7: SoftwareClusterUdsDiagnosticAddress

[constr_10753] Multiplicity of attribute `SoftwareClusterUdsDiagnosticAddress.udsAddressSemantics`

Imposition time: `IT_Mani`

⌈For each `SoftwareClusterUdsDiagnosticAddress`, the attribute `udsAddressSemantics` shall exist.⌋

5.1.3 Diagnostic Authentication Configuration

[TPS_MANI_01435] Semantics of the meta-class `DiagnosticExternalAuthenticationIdentification` ⌈Meta-class `DiagnosticExternalAuthenticationIdentification` is used to define source address information for the authentication of diagnostic clients.⌋

It is possible to support diagnostic clients with a fixed source address or with a range of source addresses by means of the aggregation of `DiagnosticExternalAuthenticationIdentification` at `DiagnosticCommonProps` with a multiplicity of 0..*.

Class	**DiagnosticExternalAuthenticationIdentification**			
Note	This meta-class represents the ability to support the authentication of diagnostic clients. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	DiagnosticCommonProps.externalAuthentication			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sourceAddress RangeEnd	PositiveInteger	0..1	attr	The source address defines the logical address of the tester that starts the communication. This element is able to define a source address range (without gaps) with the two attributes sourceAddressRangeStart and source AddressRangeEnd. This attribute defines the end of the range.
sourceAddress RangeStart	PositiveInteger	0..1	attr	The source address defines the logical address of the tester that starts the communication. This element is able to define a source address range (without gaps) with the two attributes sourceAddressRangeStart and source AddressRangeEnd. This attribute defines the start of the range.

Table 5.8: DiagnosticExternalAuthenticationIdentification

5.2 Diagnostic Mapping

5.2.1 Overview

In terms of applicability to this document, one specific 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 5.2 describes a very simplistic situation where two different `PPortPrototype`s typed by possibly two different `DiagnosticDataIdentifierInterface` exposed by an `Adaptive-ApplicationSwComponentType` are accessed by the `AUTOSAR Adaptive Diagnostic Management` on the *AUTOSAR adaptive platform* with the purpose of accessing an entire DID.

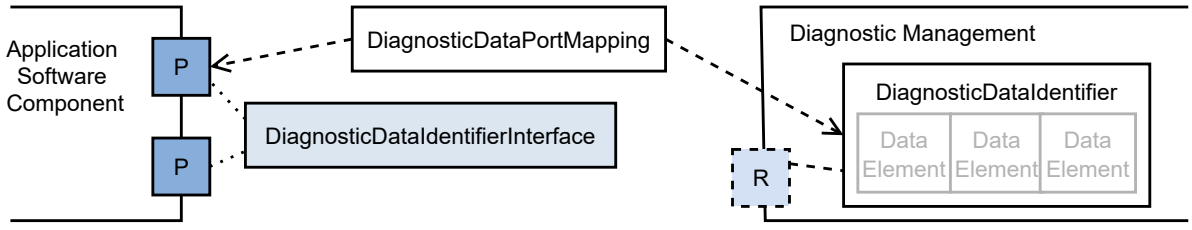


Figure 5.2: Example data exchange for diagnostic purpose

In particular, a subclass of `DiagnosticSwMapping` (in this specific case: `DiagnosticDataPortMapping`) formalizes the “connection” between both ends of the communication.

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	***DiagnosticSwMapping*** (abstract)			
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	*AbstractDoIpPortMapping*, DiagnosticAuthenticationPortMapping, DiagnosticClearConditionPortMapping, DiagnosticDataPortMapping, *DiagnosticDoIPCustomPayloadAbstractPortMapping*, DiagnosticEdrSenderPortMapping, DiagnosticEdrServerPortMapping, DiagnosticEnableConditionPortMapping, DiagnosticEventPortMapping, DiagnosticExtendedDataRecordClientPortMapping, DiagnosticExternalAuthenticationPortMapping, DiagnosticFimFunctionMapping, DiagnosticIndicatorPortMapping, DiagnosticInputOutputControlPortMapping, DiagnosticMemoryDestinationPortMapping, DiagnosticMonitorPortMapping, *DiagnosticMultipleResourcePortMapping*, DiagnosticOperationCyclePortMapping, DiagnosticSecurityLevelPortMapping, DiagnosticServiceDataMapping, DiagnosticServiceGenericMapping, DiagnosticServiceSwMapping, DiagnosticServiceValidationMapping, DiagnosticSovdAuthorizationPortMapping, DiagnosticSovdBulkDataPortMapping, DiagnosticSovdConfigurationPortMapping, DiagnosticSovdContentPortMapping, DiagnosticSovdOperationPortMapping, DiagnosticSovdProximityChallengePortMapping, DiagnosticSovdServiceValidationPortMapping, DiagnosticSovdUpdatePortMapping, ISignalPortToDiagnosticEventMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 5.9: DiagnosticSwMapping

Class	**DiagnosticDataIdentifier**			
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*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	DiagnosticParameter	*	aggr	This is the dataElement associated with the Diagnostic DataIdentifier. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=dataElement, dataElement.bitOffset, dataElement.ident.shortName, dataElement.variationPoint.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	DiagnosticSupportInfoByte	0..1	aggr	This attribute represents the supported information associated with the DiagnosticDataIdentifier. **Tags:** atp.Splitkey=supportInfoByte

Table 5.10: DiagnosticDataIdentifier

Class	**DiagnosticSupportInfoByte**			
Note	This meta-class defines the support information (typically byte A) to declare the usability of the Data Elements within the so-called packeted PIDs (e.g. PID$68).			
Base	*ARObject*			
Aggregated by	DiagnosticDataIdentifier.supportInfoByte, DiagnosticParameterIdentifier.supportInfoByte			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
position	PositiveInteger	0..1	attr	This represents the position of the supportInfo in the PID. Unit: byte.
size	PositiveInteger	0..1	attr	This represents the size of the supportInfo within the PID. Unit: byte.

Table 5.11: DiagnosticSupportInfoByte

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, if 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 ≪instanceRef≫s 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. Consequently, the diagnostic management can be configured in the presence of mappings to an application software model that 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≫.

The mapping targets[1] 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 3.2.6.

[1] on the end of the application software

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 this requirement, i.e. a relation between a diagnostic mapping and the ProcessDesign needs to be established.

[constr_10002] Only one mapping per PortPrototype

Imposition time: IT_DiagDes

⌈If one instance of the following sub-classes of DiagnosticSwMapping – that refers to a given ProcessDesign – refers to a PortPrototype, then no other instance of DiagnosticSwMapping that refers to the same ProcessDesign 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 a 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.

⌋

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 Chapter 5.2.3.3.1) shall be applied.

[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 DiagnosticServiceGeneric-Mapping *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.

5.2.1.1 Mapping of multiple Diagnostic Resources to a PortPrototype

The “regular” mapping of diagnostic resources foresees a 1:1 mapping of a specific diagnostic resource to a PortPrototype.

There are, however, use cases where the intention is to map multiple diagnostic resources to a single PortPrototype. The following use cases exist:

- The software wants to report the status of multiple events to the DM.
- The software wants to retrieve the status of multiple events from the DM.
- The software wants to set multiple diagnostic conditions.

Using the “regular” mapping approach, the software would need to define a dedicated PortPrototype for, e.g. reporting the status of each event. This could mean that a large number of PortPrototypes have to be defined in such a case.

If the PortPrototype is typed by a DiagnosticMultipleResourceInterface, then the PortPrototype can become the target of references from multiple DiagnosticMultipleResourcePortMappings.

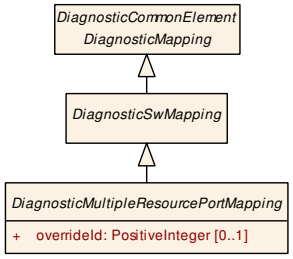


Figure 5.3: Modeling of DiagnosticMultipleResourcePortMapping

This means, for example, that for each DiagnosticEvent that shall have its status reported to the DM, a single DiagnosticMultipleMonitorPortMapping (see Chapter 5.2.2.3) shall be defined.

In other words, this approach minimizes the number of PortPrototypes that have to be defined, but has no impact on the overall number of diagnostic mappings.

Each DiagnosticMultipleResourcePortMapping can only map **one** PortPrototype to **one** diagnostic resource (e.g. a DiagnosticEvent).

Class	***DiagnosticMultipleResourcePortMapping*** (abstract)			
Note	This abstract base class enables the mapping of diagnostic PortInterfaces that deal with multiple diagnostic resources. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticMultipleConditionPortMapping, DiagnosticMultipleEventPortMapping, DiagnosticMultiple MonitorPortMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
overrideId	PositiveInteger	0..1	attr	This attribute shall be used to define the value of a manually override of the automatic generated handle Id value.

Table 5.12: DiagnosticMultipleResourcePortMapping

But it is possible to define multiple DiagnosticMultipleResourcePortMapping that refer to the **same** PortPrototype and **different** diagnostic resources of the same kind. Thereby the multiple resource mapping semantics is achieved.

[TPS_MANI_01415] Semantics of attribute DiagnosticMultipleResourcePortMapping.overrideId ⌈The DM will assign an ID for each diagnostic resource mapped to a PortPrototype that is typed by a DiagnosticMultipleResourceInterface.

If the user prefers to override the algorithm applied for the assignment of the ID, it is possible to assign a dedicated ID by means of setting the optional attribute overrideId to the intended value. The DM will respect this setting for the assignment of the ID.⌋

5.2.2 Fault-Memory-related Diagnostic Mapping

This section contains a documentation of DiagnosticMappings in the context of fault memory configuration.

5.2.2.1 Diagnostic Monitor to Port Mapping

[TPS_MANI_01351] Reporting the status of a DiagnosticEvent on the *AUTOSAR adaptive platform* ⌈For the purpose of reporting the status of a diagnostic event on the *AUTOSAR adaptive platform*, the relation between a DiagnosticEvent and one RPortPrototype 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.

⌋

Class	**DiagnosticMonitorPortMapping**			
Note	Defines to which SWC service port the Diagnostic Monitor is mapped. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to the DiagnosticEvent that is assigned to SWC service ports. **Tags:** atp.Splitkey=diagnosticEvent
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. **Tags:** atp.Splitkey=process
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.Splitkey=rPortPrototypeInExecutable, rPort PrototypeInExecutable.contextRootSwComponent Prototype, rPortPrototypeInExecutable.context ComponentPrototype, rPortPrototypeInExecutable.target RPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 5.13: DiagnosticMonitorPortMapping

The use case that supports the reporting of a diagnostic event is depicted in Figure 5.4.

[constr_10047] Restriction for the applicability of DiagnosticMonitorPortMapping

Imposition time: IT_DiagDes

⌈If an RPortPrototype is referenced by a DiagnosticMonitorPortMapping, then the RPortPrototype shall be typed by a DiagnosticMonitorInterface.⌋

[constr_10048] Existence of reference from DiagnosticMonitorPortMapping to DiagnosticEvent

Imposition time: IT_DiagDes

⌈Each DiagnosticEvent shall only be referenced by exactly one DiagnosticMonitorPortMapping or DiagnosticMultipleMonitorPortMapping.⌋

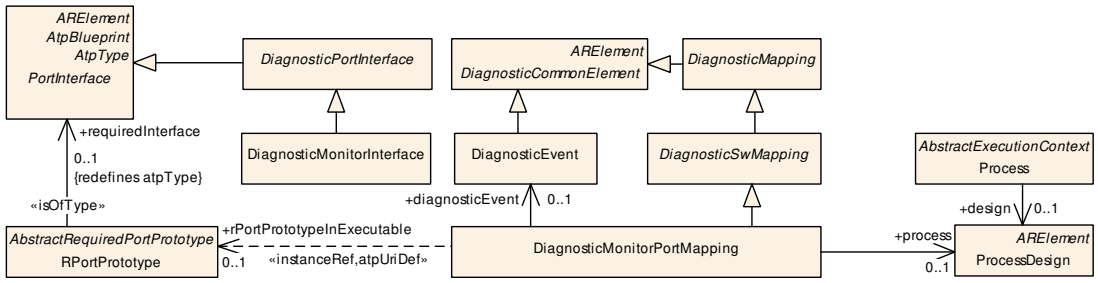


Figure 5.4: Modeling of `DiagnosticMonitorPortMapping` for reporting the status of diagnostic events on the *AUTOSAR adaptive platform*

Class	**DiagnosticEvent**			
Note	This element is used to configure DiagnosticEvents. **Tags:** atp.recommendedPackage=DiagnosticEvents			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
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
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=postBuild
connected Indicator	DiagnosticConnected Indicator	*	aggr	Event specific description of Indicators. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=connectedIndicator, connected Indicator.shortName, connectedIndicator.variation Point.shortLabel vh.latestBindingTime=postBuild
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 fFalse: Prestorage of FreezeFrames is not supported

△

Class	**DiagnosticEvent**			
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 5.14: DiagnosticEvent

Enumeration	**DiagnosticClearEventAllowedBehaviorEnum**
Note	This enumeration defines the possible behavior for clear event allowed
Aggregated by	DiagnosticEvent.clearEventAllowedBehavior
Literal	*Description*
noStatusByte Change	The event status byte keeps unchanged. **Tags:** atp.EnumerationLiteralIndex=0
onlyThisCycleAnd Readiness	The OperationCycle and readiness bits of the event status byte are reset. **Tags:** atp.EnumerationLiteralIndex=1

Table 5.15: DiagnosticClearEventAllowedBehaviorEnum

Class	**DiagnosticConnectedIndicator**			
Note	Description of indicators that are defined per DiagnosticEvent.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DiagnosticEvent.connectedIndicator			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
behavior	DiagnosticConnected IndicatorBehaviorEnum	0..1	attr	Behavior of the linked indicator.
healingCycle Counter Threshold	PositiveInteger	0..1	attr	This attribute defines the number of healing cycles for the WarningIndicatorOffCriteria **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
indicator	DiagnosticIndicator	0..1	ref	Reference to the used indicator. **Tags:** atp.Splitkey=indicator
indicatorFailure CycleCounter Threshold	PositiveInteger	0..1	attr	This attribute defines the number of failure cycles for the WarningIndicatorOnCriteria. Please note that this attribute is not relevant for the Adaptive Platform.

Table 5.16: DiagnosticConnectedIndicator

5.2.2.1.1 Multiple Diagnostic Monitor to Port Mapping

[TPS_MANI_01416] Reporting the status of multiple `DiagnosticEvent`s in the context of one `PortInterface` on the *AUTOSAR adaptive platform* ⌈For the purpose of reporting the status of multiple diagnostic events on the *AUTOSAR adaptive*

platform, the relation between a collection of DiagnosticEvents and one RPort-Prototype is created by using the DiagnosticMultipleMonitorPortMapping that refers to

- a DiagnosticEvent in the role diagnosticEvent,
- a ProcessDesign in the role process, and
- an RPortPrototype typed by a DiagnosticMultipleMonitorInterface 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.

⌋

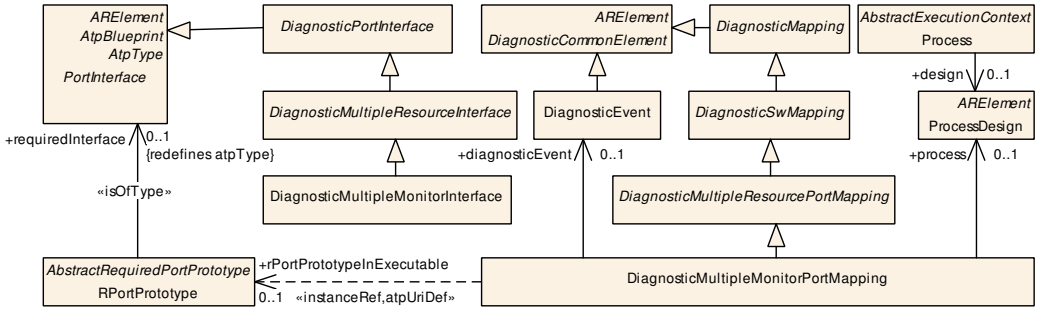


Figure 5.5: Modeling of DiagnosticMultipleMonitorPortMapping for reporting the status of multiple diagnostic events on the *AUTOSAR adaptive platform*

Class	**DiagnosticMultipleMonitorPortMapping**			
Note	Defines to which SWC service port that can handle a collection of monitors the specific event is mapped **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticMultipleResourcePortMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to the DiagnosticEvent that is assigned to a SWC service port. **Tags:** atp.Splitkey=diagnosticEvent
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. **Tags:** atp.Splitkey=process

▽

△

Class	**DiagnosticMultipleMonitorPortMapping**			
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic MonitorMultipleMonitorPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=rPortPrototypeInExecutable, rPort PrototypeInExecutable.contextRootSwComponent Prototype, rPortPrototypeInExecutable.context ComponentPrototype, rPortPrototypeInExecutable.target RPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 5.17: DiagnosticMultipleMonitorPortMapping

[constr_10436] Restriction for the applicability of `DiagnosticMultipleMonitorPortMapping`

Imposition time: `IT_DiagDes`

⌈If an `RPortPrototype` is referenced by a `DiagnosticMultipleMonitorPortMapping`, then the `RPortPrototype` shall be typed by a `DiagnosticMultipleMonitorInterface`.⌋

5.2.2.2 Diagnostic Event to Port Mapping

[TPS_MANI_01048] Retrieving the status of a `DiagnosticEvent` to `PortPrototype`(s) on the *AUTOSAR adaptive platform* ⌈For the purpose of retrieving the status of a diagnostic event on the *AUTOSAR adaptive platform*, the relation between a `DiagnosticEvent` and one `RPortPrototype` 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.

⌋

[constr_10049] Restriction for the applicability of `DiagnosticEventPortMapping`

Imposition time: `IT_DiagDes`

⌈If an `RPortPrototype` is referenced by a `DiagnosticEventPortMapping`, then the `RPortPrototype` shall be typed by a `DiagnosticEventInterface`.⌋

The use case that supports the retrieval of information about a diagnostic event is depicted in Figure 5.6.

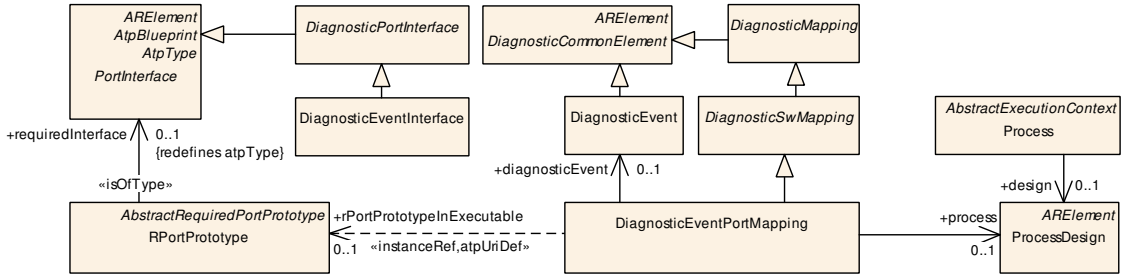


Figure 5.6: Modeling of `DiagnosticEventPortMapping` to retrieve information about a diagnostic event on the *AUTOSAR adaptive platform*

Class	**DiagnosticEventPortMapping**			
Note	Defines to which SWC service ports the DiagnosticEvent is mapped. **Tags:** atp.recommendedPackage=DiagnosticPortMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to the DiagnosticEvent that is assigned to SWC service ports. **Tags:** atp.Splitkey=diagnosticEvent
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. **Tags:** atp.Splitkey=process This Attribute is only used by the AUTOSAR Adaptive Platform.
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.Splitkey=rPortPrototypeInExecutable, rPort PrototypeInExecutable.contextRootSwComponent Prototype, rPortPrototypeInExecutable.context ComponentPrototype, rPortPrototypeInExecutable.target RPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef This Attribute is only used by the AUTOSAR Adaptive Platform.

Table 5.18: DiagnosticEventPortMapping

5.2.2.3 Multiple Diagnostic Event to Port Mapping

[TPS_MANI_01417] Retrieving the status of multiple `DiagnosticEvent`s to `PortPrototype`(s) on the *AUTOSAR adaptive platform* ⌈For the purpose of retrieving the status of a collection of diagnostic events on the *AUTOSAR adaptive platform*, the relation between the collection of `DiagnosticEvent`s and one `RPortPrototype` is created by using the `DiagnosticMultipleEventPortMapping` 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.

⌋

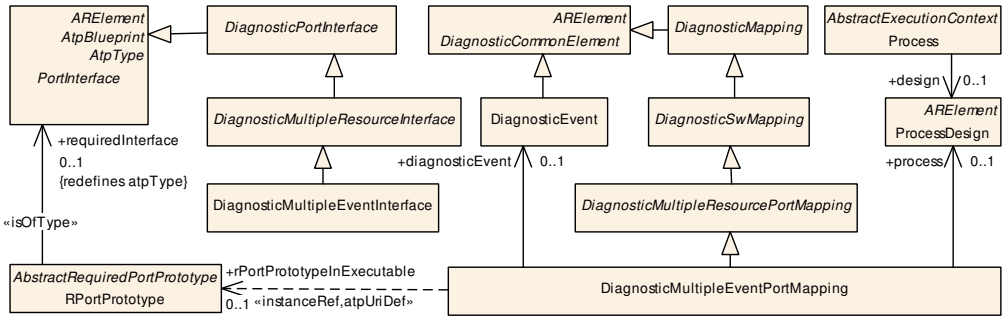


Figure 5.7: Modeling of `DiagnosticMultipleEventPortMapping` for retrieving the status of multiple diagnostic events on the *AUTOSAR adaptive platform*

Class	**DiagnosticMultipleEventPortMapping**			
Note	Defines to which SWC service port that can handle a collection of event status the specific event is mapped. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticMultipleResourcePortMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to the DiagnosticEvent that is assigned to a SWC service port. **Tags:** atp.Splitkey=diagnosticEvent

▽

△

Class	DiagnosticMultipleEventPortMapping			
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. **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic MonitorMultipleEventPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=rPortPrototypeInExecutable, rPort PrototypeInExecutable.contextRootSwComponent Prototype, rPortPrototypeInExecutable.context ComponentPrototype, rPortPrototypeInExecutable.target RPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 5.19: DiagnosticMultipleEventPortMapping

[constr_10437] Restriction for the applicability of `DiagnosticMultipleEventPortMapping`

Imposition time: `IT_DiagDes`

⌈If an `RPortPrototype` is referenced by a `DiagnosticMultipleEventPortMapping`, then the `RPortPrototype` shall be typed by a `DiagnosticMultipleEventInterface`.⌋

5.2.2.4 Diagnostic Operation Cycle to Port Mapping

[TPS_MANI_01049] Mapping of `DiagnosticOperationCycle` to `PortPrototype`(s) on the *AUTOSAR adaptive platform* ⌈On 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.

⌋

[constr_10050] Restriction for the applicability of DiagnosticOperationCyclePortMapping

Imposition time: IT_DiagDes

⌈If an RPortPrototype is referenced by a DiagnosticOperationCyclePortMapping, then the RPortPrototype shall be typed by a DiagnosticOperationCycleInterface.⌋

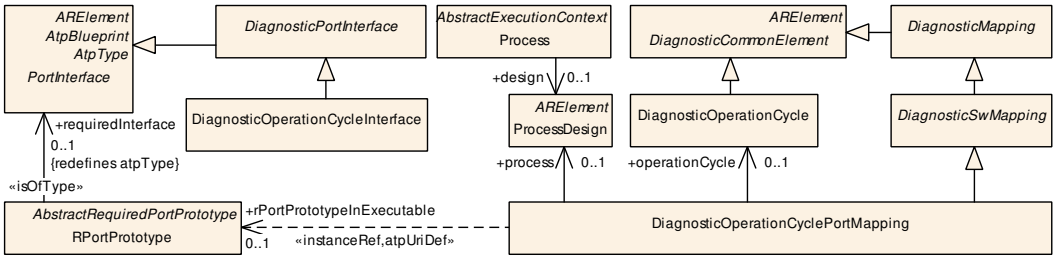


Figure 5.8: Modeling of DiagnosticOperationCyclePortMapping for the usage on the *AUTOSAR adaptive platform*

[constr_10051] Existence of reference from DiagnosticOperationCyclePortMapping to DiagnosticOperationCycle

Imposition time: IT_DiagDes

⌈Each DiagnosticOperationCycle shall only be referenced by exactly one DiagnosticOperationCyclePortMapping.⌋

Class	**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*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
type	DiagnosticOperation CycleTypeEnum	0..1	attr	Operation cycles types for the Dem.

Table 5.20: DiagnosticOperationCycle

Enumeration	**DiagnosticOperationCycleTypeEnum**
Note	Operation cycles types used to identify certain Operation cycles with a certain semantics.
Aggregated by	DiagnosticOperationCycle.type
Literal	***Description***
ignition	Ignition ON / OFF cycle **Tags:** atp.EnumerationLiteralIndex=0
obdDrivingCycle	OBD Driving cycle **Tags:** atp.EnumerationLiteralIndex=1

▽

△

Enumeration	**DiagnosticOperationCycleTypeEnum**
other	further operation cycle **Tags:** atp.EnumerationLiteralIndex=2
propulsionSystem Active	Zero Emission Vehicle Propulsion System Active operation cycle **Tags:** atp.EnumerationLiteralIndex=6
warmup	OBD Warm up cycle **Tags:** atp.EnumerationLiteralIndex=5

Table 5.21: DiagnosticOperationCycleTypeEnum

Class	**DiagnosticOperationCyclePortMapping**			
Note	Defines to which SWC service ports the DiagnosticOperationCycle is mapped. **Tags:** atp.recommendedPackage=DiagnosticPortMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
operationCycle	DiagnosticOperation Cycle	0..1	ref	Reference to the DiagnosticOperationCycle that is assigned to SWC service ports. **Tags:** atp.Splitkey=operationCycle
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. **Tags:** atp.Splitkey=process This Attribute is only used by the AUTOSAR Adaptive Platform.
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.Splitkey=rPortPrototypeInExecutable, rPort PrototypeInExecutable.contextRootSwComponent Prototype, rPortPrototypeInExecutable.context ComponentPrototype, rPortPrototypeInExecutable.target RPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef This Attribute is only used by the AUTOSAR Adaptive Platform.

Table 5.22: DiagnosticOperationCyclePortMapping

5.2.2.5 Diagnostic Enable Condition to Port Mapping

[TPS_MANI_01050] Mapping of `DiagnosticEnableCondition` to `PortPrototype`(s) on the *AUTOSAR adaptive platform* ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticEnableCondition` and one `RPortPrototype` is created by using the `DiagnosticEnableConditionPortMapping` 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.

⌋

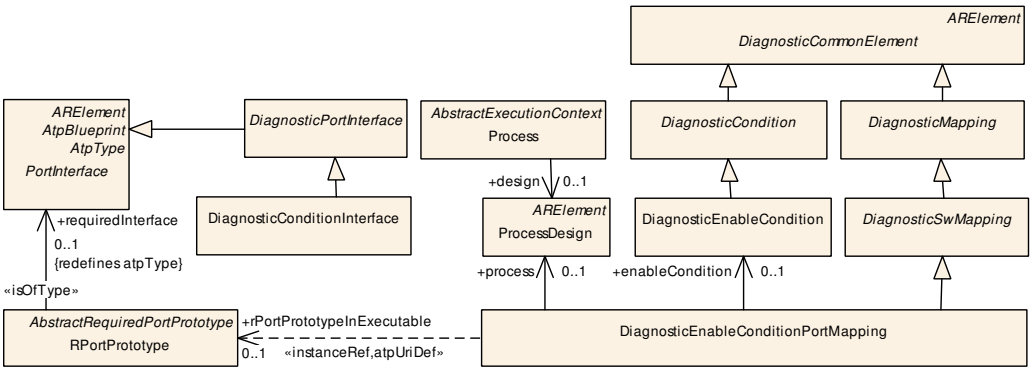


Figure 5.9: Modeling of DiagnosticEnableConditionPortMapping for the usage on the *AUTOSAR adaptive platform*

[constr_10052] Restriction for the applicability of DiagnosticEnableConditionPortMapping

Imposition time: IT_DiagDes

⌈If an RPortPrototype is referenced by a DiagnosticEnableConditionPortMapping, then the RPortPrototype shall be typed by a DiagnosticConditionInterface.⌋

[constr_10053] Existence of reference from DiagnosticEnableConditionPortMapping to DiagnosticEnableCondition

Imposition time: IT_DiagDes

⌈Each DiagnosticEnableCondition shall only be referenced by at most one DiagnosticEnableConditionPortMapping.⌋

Class	**DiagnosticEnableCondition**			
Note	Specification of an enable condition. **Tags:** atp.recommendedPackage=DiagnosticConditions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticCondition*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 5.23: DiagnosticEnableCondition

Class	**DiagnosticEnableConditionPortMapping**			
Note	Defines to which SWC service ports the DiagnosticEnableCondition is mapped. **Tags:** atp.recommendedPackage=DiagnosticPortMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
enableCondition	DiagnosticEnable Condition	0..1	ref	Reference to the EnableCondition which is mapped to a SWC service port. **Tags:** atp.Splitkey=enableCondition
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. **Tags:** atp.Splitkey=process This Attribute is only used by the AUTOSAR Adaptive Platform.
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.Splitkey=rPortPrototypeInExecutable, rPort PrototypeInExecutable.contextRootSwComponent Prototype, rPortPrototypeInExecutable.context ComponentPrototype, rPortPrototypeInExecutable.target RPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef This Attribute is only used by the AUTOSAR Adaptive Platform.

Table 5.24: DiagnosticEnableConditionPortMapping

5.2.2.6 Diagnostic Clear Condition to Port Mapping

[TPS_MANI_01259] Mapping of `DiagnosticClearCondition` to `PortPrototype`(s) on the *AUTOSAR adaptive platform* ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticClearCondition` and one `RPortPrototype` is 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.

⌋

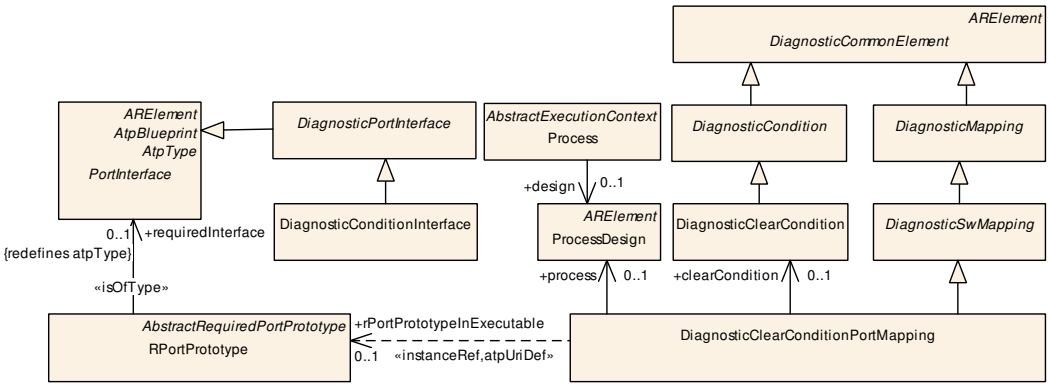


Figure 5.10: Modeling of `DiagnosticClearConditionPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_10054] Restriction for the applicability of meta-class `DiagnosticClearConditionPortMapping`

Imposition time: `IT_DiagDes`

⌈If an `RPortPrototype` is referenced by a `DiagnosticClearConditionPortMapping`, then the `RPortPrototype` shall be typed by a `DiagnosticConditionInterface`.⌋

[constr_10055] Existence of reference from `DiagnosticClearConditionPortMapping` to `DiagnosticClearCondition`

Imposition time: `IT_DiagDes`

⌈Each `DiagnosticClearCondition` shall only be referenced by at most one `DiagnosticClearConditionPortMapping`.⌋

Class	**DiagnosticClearConditionPortMapping**			
Note	Defines to which SWC service ports the DiagnosticClearCondition is mapped. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clearCondition	DiagnosticClear Condition	0..1	ref	Reference to the ClearCondition which is mapped to a SWC service port. **Tags:** atp.Splitkey=clearCondition
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. **Tags:** atp.Splitkey=process

▽

△

Class	DiagnosticClearConditionPortMapping			
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.Splitkey=rPortPrototypeInExecutable, rPort PrototypeInExecutable.contextRootSwComponent Prototype, rPortPrototypeInExecutable.context ComponentPrototype, rPortPrototypeInExecutable.target RPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 5.25: DiagnosticClearConditionPortMapping

[constr_10162] Multiplicity of reference in the role `DiagnosticClearConditionPortMapping.clearCondition`

Imposition time: `IT_DiagDes`

⌈For each `DiagnosticClearConditionPortMapping`, the reference in the role `clearCondition` shall exist.⌋

5.2.2.7 Multiple Diagnostic Condition to Port Mapping

[TPS_MANI_01418] Mapping of multiple `DiagnosticCondition`s to a `PortPrototype` on the *AUTOSAR adaptive platform* ⌈On the *AUTOSAR adaptive platform*, the relation between a collection of `DiagnosticCondition`s and one `RPortPrototype` is created by using the `DiagnosticMultipleConditionPortMapping` that refers to

- a collection of `DiagnosticCondition`s in the role `diagnosticCondition`,
- a `ProcessDesign` in the role `process`, and
- an `RPortPrototype` typed by a `DiagnosticMultipleConditionInterface` 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.

⌋

The nature of the reference in the role `DiagnosticMultipleConditionPortMapping.diagnosticCondition` allows for referencing a mixed collection of *enable conditions* and *clear conditions*, if applicable.

But since the concrete reference only points to the concrete sub-classes of `DiagnosticCondition`, the semantics of each individual reference can easily be determined.

Class	**DiagnosticMultipleConditionPortMapping**			
Note	Defines to which SWC service port that can handle a collection of diagostic conditions the specific condition is mapped. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticMultipleResourcePortMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnostic Condition	DiagnosticCondition	0..1	ref	Reference to the DiagnosticCondition which is mapped to a SWC service port. **Tags:** atp.Splitkey=diagnosticCondition
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. **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic ConditionPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=rPortPrototypeInExecutable, rPort PrototypeInExecutable.contextRootSwComponent Prototype, rPortPrototypeInExecutable.context ComponentPrototype, rPortPrototypeInExecutable.target RPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 5.26: DiagnosticMultipleConditionPortMapping

[constr_10438] Restriction for the applicability of meta-class DiagnosticMultipleConditionPortMapping

Imposition time: IT_DiagDes

⌈If an RPortPrototype is referenced by a DiagnosticMultipleConditionPortMapping, then the RPortPrototype shall be typed by a DiagnosticMultipleConditionInterface.⌋

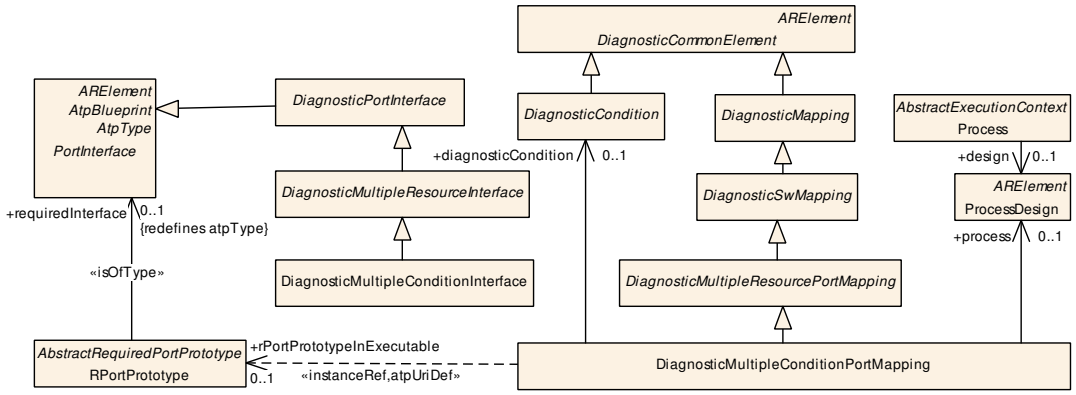


Figure 5.11: Modeling of DiagnosticMultipleConditionPortMapping for reporting the status of multiple diagnostic events on the *AUTOSAR adaptive platform*

5.2.2.8 Diagnostic Indicator to Port Mapping

[TPS_MANI_01260] Mapping of `DiagnosticIndicator` to `PortPrototype`(s) on the *AUTOSAR adaptive platform* ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticIndicator` and one `RPortPrototype` is 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.

⌋

[constr_10056] Restriction for the applicability of `DiagnosticIndicatorPortMapping`

Imposition time: `IT_DiagDes`

⌈If an `RPortPrototype` is referenced by a `DiagnosticIndicatorPortMapping`, then the `RPortPrototype` shall be typed by a `DiagnosticIndicatorInterface`.⌋

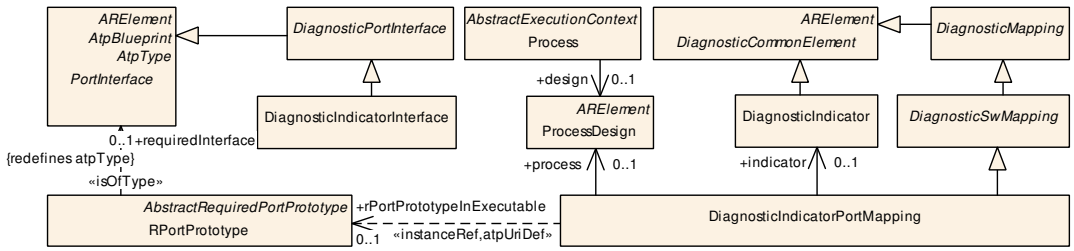


Figure 5.12: Modeling of `DiagnosticIndicatorPortMapping` for the usage on the *AUTOSAR adaptive platform*

Class	**DiagnosticIndicatorPortMapping**			
Note	Defines to which SWC service ports the DiagnosticIndicator is mapped. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	DiagnosticIndicatorPortMapping			
indicator	DiagnosticIndicator	0..1	ref	Reference to the DiagnosticIndicator which is mapped to a SWC service port. **Tags:** atp.Splitkey=indicator
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. **Tags:** atp.Splitkey=process
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.Splitkey=rPortPrototypeInExecutable, rPort PrototypeInExecutable.contextRootSwComponent Prototype, rPortPrototypeInExecutable.context ComponentPrototype, rPortPrototypeInExecutable.target RPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 5.27: DiagnosticIndicatorPortMapping

[constr_10163] Multiplicity of reference in the role `DiagnosticIndicatorPortMapping.indicator`

Imposition time: `IT_DiagDes`

⌈For each `DiagnosticIndicatorPortMapping`, the reference in the role `indicator` shall exist.⌋

5.2.2.9 Diagnostic Memory Destination to Port Mapping

[TPS_MANI_01261] Mapping of `DiagnosticMemoryDestination` to `PortPrototype`(s) on the *AUTOSAR adaptive platform* ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticMemoryDestination` and one `RPortPrototype` is created by using the `DiagnosticMemoryDestinationPortMapping` 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.

⌋

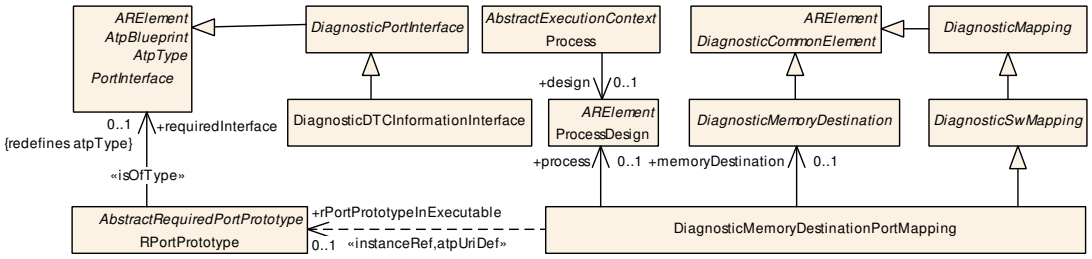


Figure 5.13: Modeling of `DiagnosticMemoryDestinationPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_10057] Restriction for the applicability of meta-class `DiagnosticMemoryDestinationPortMapping`

Imposition time: `IT_DiagDes`

⌈If an `RPortPrototype` is referenced by a `DiagnosticMemoryDestinationPortMapping`, then the `RPortPrototype` shall be typed by a `DiagnosticDTCInformationInterface`.⌋

Class	**DiagnosticMemoryDestinationPortMapping**			
Note	Defines to which SWC service ports the DiagnosticMemoryDestination. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
memory Destination	DiagnosticMemory Destination	0..1	ref	Reference to the MemoryDestination which is mapped to a SWC service port. **Tags:** atp.Splitkey=memoryDestination
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. **Tags:** atp.Splitkey=process
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.Splitkey=rPortPrototypeInExecutable, rPort PrototypeInExecutable.contextRootSwComponent Prototype, rPortPrototypeInExecutable.context ComponentPrototype, rPortPrototypeInExecutable.target RPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 5.28: DiagnosticMemoryDestinationPortMapping

[constr_10164] Multiplicity of reference in the role `DiagnosticMemoryDestinationPortMapping.memoryDestination`

Imposition time: `IT_DiagDes`

⌈For each `DiagnosticMemoryDestinationPortMapping`, the reference in the role `memoryDestination` shall exist.⌋

Class	***DiagnosticMemoryDestination*** (abstract)			
Note	This abstract meta-class represents a possible memory destination for a diagnostic event.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticMemoryDestinationPrimary, DiagnosticMemoryDestinationUserDefined			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
agingRequiresTestedCycle	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. On the adaptive platform, the value of this attribute can be different for each DiagnosticMemoryDestination.
clearDtcLimitation	DiagnosticClearDtcLimitationEnum	0..1	attr	Defines the scope of the DEM_ClearDTC Api. On the adaptive platform, the value of this attribute can be different for each DiagnosticMemoryDestination.
dtcStatusAvailabilityMask	PositiveInteger	0..1	attr	Mask for the supported DTC status bits by the Dem.
eventDisplacementStrategy	DiagnosticEventDisplacementStrategyEnum	0..1	attr	This attribute defines, whether support for event displacement is enabled or not, and which displacement strategy is followed.
maxNumberOfEventEntries	PositiveInteger	0..1	attr	This attribute fixes the maximum number of event entries in the fault memory.
memoryEntryStorageTrigger	DiagnosticMemoryEntryStorageTriggerEnum	0..1	attr	Describes the trigger to allocate an event memory entry.
statusBitHandlingTestFailedSinceLastClear	DiagnosticStatusBitHandlingTestFailedSinceLastClearEnum	0..1	attr	This attribute defines, whether the aging and displacement mechanism shall be applied to the "Test FailedSinceLastClear" status bits. On the adaptive platform, the value of this attribute can be different for each DiagnosticMemoryDestination.
typeOfFreezeFrameRecordNumeration	DiagnosticTypeOfFreezeFrameRecordNumerationEnum	0..1	attr	This attribute defines the type of assigning freeze frame record numbers for event-specific freeze frame records.

Table 5.29: DiagnosticMemoryDestination

Enumeration	**DiagnosticClearDtcLimitationEnum**
Note	Scope of the DEM_ClearDTC Api.
Aggregated by	*DiagnosticMemoryDestination*.clearDtcLimitation
Literal	***Description***

▽

△

Enumeration	**DiagnosticClearDtcLimitationEnum**
allSupportedDtcs	DEM_ClearDtc API accepts all supported DTC values. **Tags:** atp.EnumerationLiteralIndex=0
clearAllDtcs	DEM_ClearDtc API accepts ClearAllDTCs only. **Tags:** atp.EnumerationLiteralIndex=1

Table 5.30: DiagnosticClearDtcLimitationEnum

Enumeration	**DiagnosticEventDisplacementStrategyEnum**
Note	Defines the displacement strategy.
Aggregated by	*DiagnosticMemoryDestination*.eventDisplacementStrategy
Literal	***Description***
full	Event memory entry displacement is enabled, by consideration of priority active/passive status, and occurrence. **Tags:** atp.EnumerationLiteralIndex=0
none	Event memory entry displacement is disabled. **Tags:** atp.EnumerationLiteralIndex=1
prioOcc	Event memory entry displacement is enabled, by consideration of priority and occurrence (but without active/passive status). **Tags:** atp.EnumerationLiteralIndex=2

Table 5.31: DiagnosticEventDisplacementStrategyEnum

Enumeration	**DiagnosticMemoryEntryStorageTriggerEnum**
Note	Trigger types to allocate an event memory entry.
Aggregated by	*DiagnosticMemoryDestination*.memoryEntryStorageTrigger
Literal	***Description***
confirmed	Status information of UDS DTC status bit 3 **Tags:** atp.EnumerationLiteralIndex=0
fdcThreshold	Threshold to allocate an event memory entry and to capture the Freeze Frame. **Tags:** atp.EnumerationLiteralIndex=1
testFailed	Status information of UDS DTC status bit 0. **Tags:** atp.EnumerationLiteralIndex=3

Table 5.32: DiagnosticMemoryEntryStorageTriggerEnum

Enumeration	**DiagnosticStatusBitHandlingTestFailedSinceLastClearEnum**
Note	This enumeration controls whether the aging and displacement mechanism shall be applied to the 'TestFailedSinceLastClear' status bits.
Aggregated by	*DiagnosticMemoryDestination*.statusBitHandlingTestFailedSinceLastClear
Literal	***Description***
statusBitAgingAnd Displacement	The "TestFailedSinceLastClear" status bits are reset to 0, if aging or displacement applies. **Tags:** atp.EnumerationLiteralIndex=0
statusBitNormal	Aging and displacement has no impact on the "TestFailedSinceLastClear" status bits. **Tags:** atp.EnumerationLiteralIndex=1

Table 5.33: DiagnosticStatusBitHandlingTestFailedSinceLastClearEnum

Enumeration	**DiagnosticTypeOfFreezeFrameRecordNumerationEnum**
Note	FreezeFrame record numeration type
Aggregated by	*DiagnosticMemoryDestination*.typeOfFreezeFrameRecordNumeration
Literal	***Description***
calculated	Freeze frame records will be numbered consecutive starting by 1 in their chronological order. **Tags:** atp.EnumerationLiteralIndex=0
configured	Freeze frame records will be numbered based on the given configuration in their chronological order. **Tags:** atp.EnumerationLiteralIndex=1

Table 5.34: DiagnosticTypeOfFreezeFrameRecordNumerationEnum

5.2.2.10 Diagnostic Extended Data Record Port Mapping

[TPS_MANI_01484] Mapping of `DiagnosticExtendedDataRecordElement` to `PortPrototype`(s) on the *AUTOSAR adaptive platform* ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticExtendedDataRecordElement` and one `RPortPrototype` is created by using the `DiagnosticExtendedDataRecordClientPortMapping` that refers to

- a `DiagnosticExtendedDataRecordElement` in the role `recordElement`,
- a `ProcessDesign` in the role `process`, and
- an `RPortPrototype` typed by a `DiagnosticExtendedDataRecordInterface` 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.

⌋

[constr_10684] Restriction for the applicability of `DiagnosticExtendedDataRecordClientPortMapping`

Imposition time: `IT_DiagDes`

⌈If an `RPortPrototype` is referenced by a `DiagnosticExtendedDataRecordClientPortMapping`, then the `RPortPrototype` shall be typed by a `DiagnosticExtendedDataRecordInterface`.⌋

There is no other use case for the utilization of an `RPortPrototype` that is typed by a `DiagnosticExtendedDataRecordInterface`, apart from becoming the target of a `DiagnosticExtendedDataRecordClientPortMapping`. This restriction is formulated in [constr_10685].

[constr_10685] Restriction for the reference to an RPortPrototype typed by a DiagnosticExtendedDataRecordInterface

Imposition time: IT_DiagDes

⌈If an RPortPrototype is typed by a DiagnosticExtendedDataRecord-Interface, it shall only be referenced in the role rPortPrototypeInExecutable by a DiagnosticExtendedDataRecordClientPortMapping.⌋

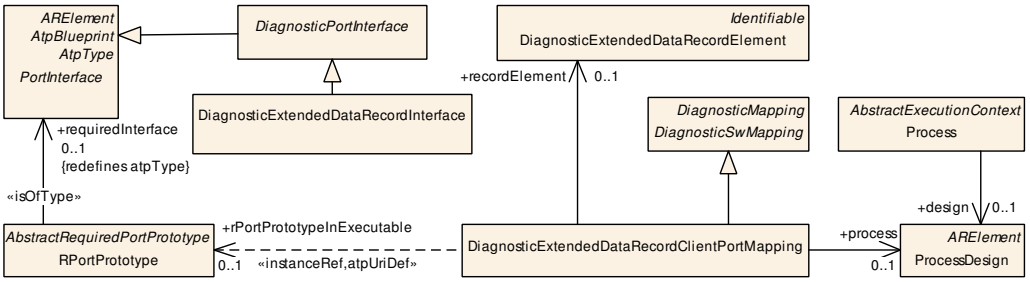


Figure 5.14: Modeling of DiagnosticExtendedDataRecordClientPortMapping for the usage on the *AUTOSAR adaptive platform*

Class	**DiagnosticExtendedDataRecordClientPortMapping**			
Note	This mapping class identifies a PortPrototype in the application software that provides data to an external data record. **Tags:** atp.recommendedPackage=DiagnosticPortMappppings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
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. **Tags:** atp.Splitkey=process
recordElement	DiagnosticExtended DataRecordElement	0..1	ref	This reference identifies the mapped element of an extended data record. **Tags:** atp.Splitkey=recordElement
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This reference identifies the applicable RPortPrototype 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.Splitkey=rPortPrototypeInExecutable, rPort PrototypeInExecutable.contextRootSwComponent Prototype, rPortPrototypeInExecutable.context ComponentPrototype, rPortPrototypeInExecutable.target RPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 5.35: DiagnosticExtendedDataRecordClientPortMapping

[constr_10686] Multiplicity of reference in the role DiagnosticExtendedDataRecordClientPortMapping.recordElement

Imposition time: IT_DiagDes

⌈For each DiagnosticExtendedDataRecordClientPortMapping, the reference in the role recordElement shall exist.⌋

[constr_10732] Existence of the attributes DiagnosticExtendedDataRecord.trigger and update on the *AUTOSAR adaptive platform*

Imposition time: AP: IT_DiagDes

⌈For each DiagnosticExtendedDataRecord, attributes trigger and update shall only exist if at least one DiagnosticExtendedDataRecordElement is aggregated by a DiagnosticExtendedDataRecord in the role element that is referenced in the role DiagnosticExtendedDataRecordClientPortMapping.recordElement.⌋

[constr_10732] is the equivalent of [constr_1756], see [3, TPS Diagnostic Extract Template] Chapter 4.4.5.

5.2.3 UDS-related Diagnostic Mapping

This section contains a documentation of DiagnosticMappings in the context of UDS service configuration.

5.2.3.1 Diagnostic Security to Port Mapping

[TPS_MANI_01262] Mapping of DiagnosticSecurityLevel to PortPrototype(s) on the *AUTOSAR adaptive platform* ⌈On the *AUTOSAR adaptive platform*, the relation between a DiagnosticSecurityLevel and one PPortPrototype is created by using the DiagnosticSecurityLevelPortMapping that refers to

- a DiagnosticSecurityLevel in the role securityLevel,
- a ProcessDesign in the role process, and
- a 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.

⌋

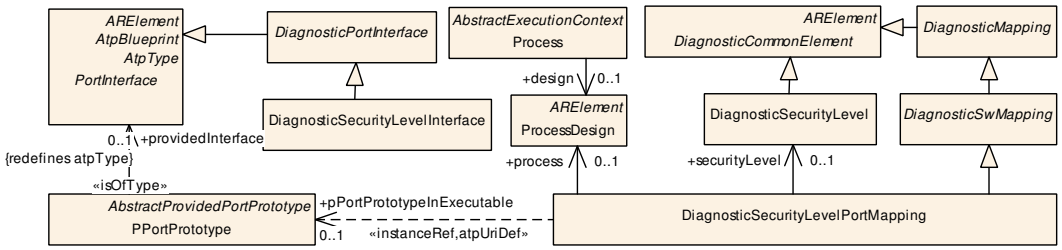


Figure 5.15: Modeling of `DiagnosticSecurityLevelPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_10058] Restriction for the applicability of `DiagnosticSecurityLevelPortMapping`

Imposition time: `IT_DiagDes`

⌈If a `PPortPrototype` is referenced by a `DiagnosticSecurityLevelPortMapping`, then the `PPortPrototype` shall be typed by a `DiagnosticSecurityLevelInterface`.⌋

[constr_10059] Existence of reference from `DiagnosticSecurityLevelPortMapping` to `DiagnosticSecurityLevel`

Imposition time: `IT_DiagDes`

⌈Each `DiagnosticSecurityLevel` shall only be referenced by exactly one `DiagnosticSecurityLevelPortMapping`.⌋

Class	**DiagnosticSecurityLevelPortMapping**			
Note	Defines to which SWC service ports the DiagnosticSecurityLevel is mapped. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
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.Splitkey=pPortPrototypeInExecutable, pPort PrototypeInExecutable.contextRootSwComponent Prototype, pPortPrototypeInExecutable.context ComponentPrototype, pPortPrototypeInExecutable.target PPortPrototype **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. **Tags:** atp.Splitkey=process

▽

△

Class	DiagnosticSecurityLevelPortMapping			
securityLevel	DiagnosticSecurityLevel	0..1	ref	Reference to the SecurityLevel which is mapped to a SWC service port. **Tags:** atp.Splitkey=securityLevel

Table 5.36: DiagnosticSecurityLevelPortMapping

Class	DiagnosticSecurityLevel			
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*			
Aggregated by	ARPackage.element			
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 5.37: DiagnosticSecurityLevel

5.2.3.2 Diagnostic Data Identifier to Port Mapping

The DM on the *AUTOSAR adaptive platform* has the ability to access entire `DiagnosticDataIdentifier`s at once. For supporting this ability, a dedicated mapping class named `DiagnosticDataPortMapping` is introduced.

[TPS_MANI_01263] Mapping of `DiagnosticDataIdentifier` or `DiagnosticDataElement` to `PortPrototype`(s) on the *AUTOSAR adaptive platform* ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticDataIdentifier` or `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`.

⌋

Class	**DiagnosticDataElement**			
Note	This meta-class represents the ability to describe a concrete piece of data to be taken into account for diagnostic purposes.			
Base	*ARObject*, *DiagnosticServiceMappingDiagTarget*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*DiagnosticAbstractParameter*.dataElement			
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.
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. **Tags:** atp.Splitkey=swDataDefProps

Table 5.38: DiagnosticDataElement

As depicted in Figure 5.16, `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] Restriction for the existence of attribute `DiagnosticDataPortMapping.diagnosticDataIdentifier` vs. attribute `DiagnosticDataPortMapping.diagnosticDataElement`

Imposition time: `IT_DiagDes`

⌈For each `DiagnosticDataPortMapping`, **either** the reference in the role `diagnosticDataIdentifier` **or** `diagnosticDataElement` shall exist.⌋

[constr_10060] `PortInterface` of `PPortPrototype` that is referenced by `DiagnosticDataPortMapping`

Imposition time: `IT_DiagDes`

⌈Any particular `PPortPrototype` that is referenced in the role `DiagnosticDataPortMapping.pPortPrototypeInExecutable` shall be typed by either of

- `DiagnosticDataIdentifierInterface`
- `DiagnosticDataElementInterface`
- `DiagnosticDataIdentifierGenericInterface`

⌋

[constr_10061] Mapping to PPortPrototype that is typed by DiagnosticDataIdentifierInterface, DiagnosticDataElementInterface, or DiagnosticDataIdentifierGenericInterface

Imposition time: IT_DiagDes

⌈All PPortPrototypes typed by either

- DiagnosticDataIdentifierInterface
- DiagnosticDataElementInterface
- DiagnosticDataIdentifierGenericInterface

shall **only** be referenced by a DiagnosticDataPortMapping. No other subclass of DiagnosticSwMapping is eligible for this purpose.⌋

[TPS_MANI_01347] Definition of a DiagnosticDataElement used in the context of a DID obtained from a PPortPrototype typed by a DiagnosticDataElementInterface ⌈If 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 of the values

- 8,
- 16,
- 32, or
- 64

The usage of attribute baseTypeEncoding is regulated by [constr_10483].

In the case addressed by this spec item, 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.⌋

[TPS_MANI_01348] Definition of a DiagnosticDataElement used in the context of a DID obtained from a PPortPrototype typed by a DiagnosticDataIdentifierInterface ⌈If 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,
- 32, or

- 64

The usage of attribute baseTypeEncoding is regulated by [constr_10483].

In the case addressed by this spec item, 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.⌋

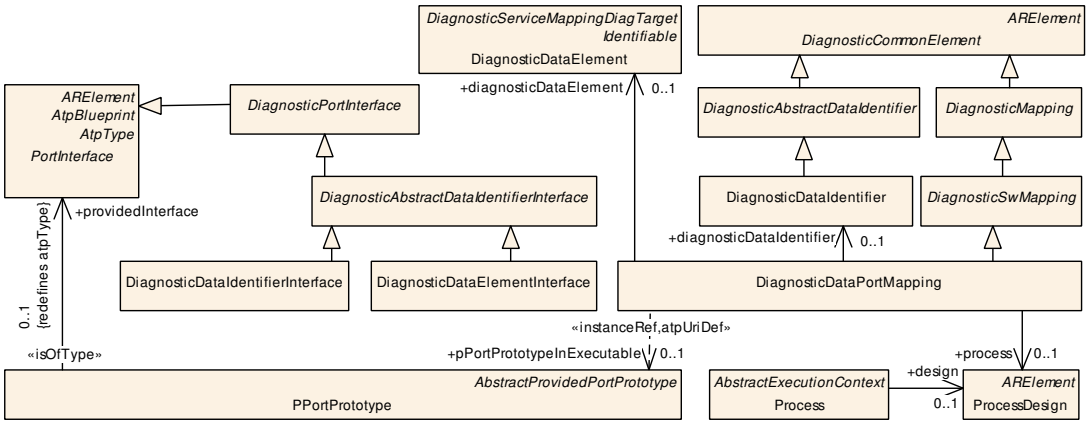


Figure 5.16: Modeling of DiagnosticDataPortMapping for the usage on the *AUTOSAR adaptive platform*

[constr_10165] Multiplicity of reference in the role DiagnosticDataPortMapping.process

Imposition time: IT_DiagDes

⌈For each DiagnosticDataPortMapping, the reference in the role process shall exist.⌋

Class	**DiagnosticDataPortMapping**			
Note	This meta-class provides the ability to define a diagnostic access to an entire DID. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnosticData Element	DiagnosticDataElement	0..1	ref	This reference represents the applicable DiagnosticData Element. **Tags:** atp.Splitkey=diagnosticDataElement
diagnosticData Identifier	DiagnosticDataIdentifier	0..1	ref	This reference represents the applicable DiagnosticData Identfiier. **Tags:** atp.Splitkey=diagnosticDataIdentifier

△

Class	**DiagnosticDataPortMapping**			
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.Splitkey=pPortPrototypeInExecutable, pPort PrototypeInExecutable.contextRootSwComponent Prototype, pPortPrototypeInExecutable.context ComponentPrototype, pPortPrototypeInExecutable.target PPortPrototype **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. **Tags:** atp.Splitkey=process

Table 5.39: DiagnosticDataPortMapping

[constr_10490] `DiagnosticDataElement` shall be directly or indirectly the target of a reference owned by a subclass of `DiagnosticMapping`

Imposition time: `IT_DiagDes`

⌈Each `DiagnosticDataElement` shall be directly or indirectly considered by a `DiagnosticMapping`. Therefore, exactly one of the following conditions shall be fulfilled:

- A `DiagnosticDataPortMapping` exists that refers to the respective `DiagnosticDataElement` in the role `diagnosticDataElement`.
- A `DiagnosticDataPortMapping` exists that refers in the role `diagnosticDataIdentifier` to the `DiagnosticDataIdentifier` that aggregates the respective `DiagnosticDataElement` in the role `dataElement`.`dataElement`.

⌋

As also explained by [TPS_MANI_01263], Meta-class `DiagnosticDataPortMapping` fulfills two purposes at once:

- map a `PPortPrototype` that is typed by a `DiagnosticDataIdentifierInterface` to a `DiagnosticDataIdentifier`.
- map a `PPortPrototype` that is typed by a `DiagnosticDataElementInterface` to a `DiagnosticDataElement`.

Mixing the two cases explained above is not supported, thus [constr_10630] and [constr_10631] exist.

[constr_10630] `PPortPrototype` that is typed by a `DiagnosticDataIdentifierInterface` shall be mapped to a `DiagnosticDataIdentifier`

Imposition time: `IT_DiagDes`

⌈Any `DiagnosticDataPortMapping` that refers to a `PPortPrototype` that is typed by a `DiagnosticDataIdentifierInterface` shall only refer to a `DiagnosticDataIdentifier` in the role `diagnosticDataIdentifier`.

In this case, the existence of the reference in the role `DiagnosticDataPortMapping.diagnosticDataElement` is not permitted.⌋

[constr_10631] `PPortPrototype` that is typed by a `DiagnosticDataElementInterface` shall be mapped to a `DiagnosticDataElement`

Imposition time: `IT_DiagDes`

⌈Any `DiagnosticDataPortMapping` that refers to a `PPortPrototype` that is typed by a `DiagnosticDataElementInterface` shall only refer to a `DiagnosticDataElement` in the role `diagnosticDataElement`.

In this case, the existence of the reference in the role `DiagnosticDataPortMapping.diagnosticDataIdentifier` is not permitted.⌋

5.2.3.3 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, each taking 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 `SoftwareCluster`s (see Chapter 16.3).

5.2.3.3.1 Diagnostic Generic Mapping

[TPS_MANI_01326] Generic Mapping to a `DiagnosticServiceInstance` on the AUTOSAR adaptive platform ⌈On the AUTOSAR adaptive platform, the relation between a `DiagnosticServiceInstance` and one `PPortPrototype` for selected 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.

⌋

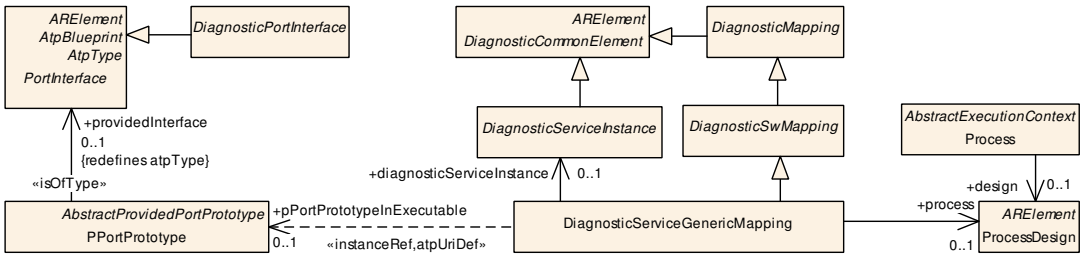


Figure 5.17: Modeling of the diagnostic generic mapping

[TPS_MANI_01453] DiagnosticServiceInstances that can be mapped by a DiagnosticServiceGenericMapping ⌈DiagnosticServiceGenericMapping shall always be used for the following list of DiagnosticServiceInstances:

- DiagnosticEcuReset
- DiagnosticComControl
- DiagnosticRoutineControl
- DiagnosticRequestUpload
- DiagnosticRequestDownload
- DiagnosticDataTransfer
- DiagnosticTransferExit
- DiagnosticRequestFileTransfer

⌋

[constr_10546] PortPrototype typed by DiagnosticGenericUdsInterface shall only be referenced by DiagnosticServiceGenericMapping

Imposition time: IT_DiagDes

⌈A PPortPrototype that is typed by DiagnosticGenericUdsInterface shall **only** be referenced in the role pPortPrototypeInExecutable by a DiagnosticServiceGenericMapping.⌋

Services DiagnosticDataTransfer and DiagnosticTransferExit are mapped dynamically according to the UDS request sequence for upload and download.

Class	**DiagnosticEcuReset**			
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*			
Aggregated by	ARPackage.element			
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. **Tags:** atp.Splitkey=ecuResetClass

Table 5.40: DiagnosticEcuReset

Class	**DiagnosticEcuResetClass**			
Note	This meta-class contains attributes shared by all instances of the "Ecu Reset" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticEcuResets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
respondTo Reset	DiagnosticResponseTo EcuResetEnum	0..1	attr	This attribute defines whether the response to the Ecu Reset service shall be transmitted before or after the actual reset.

Table 5.41: DiagnosticEcuResetClass

Class	**DiagnosticComControl**			
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*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
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. **Tags:** atp.Splitkey=comControlClass
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 5.42: DiagnosticComControl

Class	DiagnosticComControlClass			
Note	This meta-class contains attributes shared by all instances of the "Communication Control" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticCommunicationControls			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
allChannels	CommunicationCluster	*	ref	This reference represents the semantics that all available channels shall be affected. It is still necessary to refer to individual CommunicatuionClusters because there could be private CommunicationClusters in the System Extract that are not subject to the service "communication control". By referring to the applicable CommunicationClusters it can be made sure that only the affected Communication Clusters are accessed. **Tags:** atp.Splitkey=allChannels
allPhysical Channels	EthernetPhysical Channel	*	ref	This reference represents the semantics that all available channels shall be affected. It is still necessary to refer to individual EthernetPhysicalChannels because there could be private VLANs (and thus private EthernetPhysical Channels) in the System Extract that are not subject to the service "communication control". By referring to the applicable EthernetPhysicalChannels it can be made sure that only the affected EthernetPhysical Channels are accessed. **Tags:** atp.Splitkey=allPhysicalChannels xml.namePlural=ALL-PHYSICAL-CHANNELS
specificChannel	DiagnosticComControl SpecificChannel	*	aggr	This represents the ability to add additional attributes to the case that only specific channels are supposed to be considered, **Tags:** atp.Splitkey=specificChannel
subNode Channel	DiagnosticComControl SubNodeChannel	*	aggr	This attribute represents the ability to add further attributes to the definition of a specific sub-node channel that is subject to the diagnostic service "communication control". **Tags:** atp.Splitkey=subNodeChannel

Table 5.43: DiagnosticComControlClass

Class	DiagnosticRoutineControl			
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*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
routine	DiagnosticRoutine	0..1	ref	This refers to the applicable DiagnosticRoutine. **Tags:** atp.Splitkey=routine

▽

△

Class	DiagnosticRoutineControl			
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. **Tags:** atp.Splitkey=routineControlClass

Table 5.44: DiagnosticRoutineControl

Class	DiagnosticRoutineControlClass			
Note	This meta-class contains attributes shared by all instances of the "Routine Control" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticRoutineControls			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 5.45: DiagnosticRoutineControlClass

Class	DiagnosticCustomServiceInstance			
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*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
customService Class	DiagnosticCustom ServiceClass	0..1	ref	Reference to the corresponding DiagnosticCustom ServiceClass. **Tags:** atp.Splitkey=customServiceClass

Table 5.46: DiagnosticCustomServiceInstance

Class	DiagnosticCustomServiceClass			
Note	This represents the ability to define a custom diagnostic service class and assign an ID to it. Further configuration is not foreseen from the point of view of the diagnostic extract and consequently needs to be done on the level of ECUC. **Tags:** atp.recommendedPackage=DiagnosticCustomServiceClasses			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
customService Id	PositiveInteger	0..1	attr	This attribute may only be used for the definition of custom services. The values shall not overlap with existing standardized service IDs.

Table 5.47: DiagnosticCustomServiceClass

Class	DiagnosticRequestUpload			
Note	This represents an instance of the "Request Upload" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticMemoryByAdresss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMemoryAddressableRangeAccess*, *DiagnosticMemoryByAddress*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
requestUploadClass	DiagnosticRequestUploadClass	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. **Tags:** atp.Splitkey=requestUploadClass

Table 5.48: DiagnosticRequestUpload

Class	DiagnosticRequestUploadClass			
Note	This meta-class contains attributes shared by all instances of the "Request Upload" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticMemoryByAdresss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 5.49: DiagnosticRequestUploadClass

Class	DiagnosticRequestDownload			
Note	This represents an instance of the "Request Download" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticMemoryByAdresss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMemoryAddressableRangeAccess*, *DiagnosticMemoryByAddress*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
requestDownloadClass	DiagnosticRequestDownloadClass	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. **Tags:** atp.Splitkey=requestDownloadClass

Table 5.50: DiagnosticRequestDownload

Class	DiagnosticRequestDownloadClass
Note	This meta-class contains attributes shared by all instances of the "Request Download" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticMemoryByAdresss
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

Class	DiagnosticRequestDownloadClass			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 5.51: DiagnosticRequestDownloadClass

Class	DiagnosticServiceGenericMapping			
Note	This meta-class represents the ability to implement a generic generic mapping for select diagnostics services on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
diagnostic ServiceInstance	DiagnosticService Instance	0..1	ref	Reference to the ServiceInstance mapped to a SWC service port. **Tags:** atp.Splitkey=diagnosticServiceInstance
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.Splitkey=pPortPrototypeInExecutable, pPort PrototypeInExecutable.contextRootSwComponent Prototype, pPortPrototypeInExecutable.context ComponentPrototype, pPortPrototypeInExecutable.target PPortPrototype **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. **Tags:** atp.Splitkey=process

Table 5.52: DiagnosticServiceGenericMapping

Class	DiagnosticRequestFileTransfer			
Note	This diagnostic service instance implements the UDS service 0x38. **Tags:** atp.recommendedPackage=DiagnosticRequestFileTransfers			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
requestFile TransferClass	DiagnosticRequestFile TransferClass	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 DiagnosticRequestFile Transfer in the given context. **Tags:** atp.Splitkey=requestFileTransferClass

Table 5.53: DiagnosticRequestFileTransfer

Class	**DiagnosticRequestFileTransferClass**			
Note	This meta-class contains attributes shared by all instances of the "Request File transfer" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticRequestFileTransfers			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 5.54: DiagnosticRequestFileTransferClass

5.2.3.4 Diagnostic Service Validation Mapping

[TPS_MANI_01352] Definition of `DiagnosticServiceValidationMapping` ⌈On 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 `PPortPrototype`s 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 target-configuration level.

This is the main reason why the `DiagnosticServiceValidationMapping` (besides the reference to the `ProcessDesign`) has no reference into another design object in the context of the DEXT.⌋

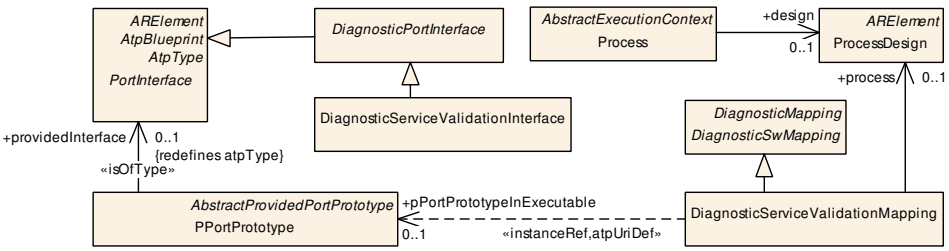


Figure 5.18: Modeling of the diagnostic validation mapping

[constr_10063] Possible values for `DiagnosticServiceValidationMapping.category`

Imposition time: `IT_DiagDes`

⌈The 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.

⌋

[constr_10064] Existence of `DiagnosticServiceValidationMapping.pPortPrototypeInExecutable`

Imposition time: `IT_DiagDes`

⌈A `PPortPrototype` referenced in the role `DiagnosticServiceValidationMapping.pPortPrototypeInExecutable` shall be typed by a `DiagnosticServiceValidationInterface`.⌋

The final decision about the order in which the validations are executed is made in the target-configuration phase, see Chapter 16.3.2.3.

Class	**DiagnosticServiceValidationMapping**			
Note	This meta-class provides the ability to specify manufacturer/supplier checks to be executed before diagnostic services can be processed. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
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.Splitkey=pPortPrototypeInExecutable, pPort PrototypeInExecutable.contextRootSwComponent Prototype, pPortPrototypeInExecutable.context ComponentPrototype, pPortPrototypeInExecutable.target PPortPrototype **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. **Tags:** atp.Splitkey=process

Table 5.55: DiagnosticServiceValidationMapping

5.2.3.5 Diagnostic Authentication Mapping

[TPS_MANI_01360] Creation of two diagnostic mappings that fulfill different roles in the context of authenticating a diagnostic client ⌈The support for the diagnostic service Authentication (0x29) on the *AUTOSAR adaptive platform* requires the creation of two diagnostic mappings that 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 as an authentication manager.

- The meta-class `DiagnosticExternalAuthenticationPortMapping` is used to convey the authentication state of a diagnostic client to the diagnostic server instance of DM.

⌋

Class	**DiagnosticAuthenticationPortMapping**			
Note	This mapping class identifies the PortPrototype in the application software that handles the client authentication. **Tags:** atp.recommendedPackage=DiagnosticPortMapppings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
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.Splitkey=diagnosticAuthentication
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic AuthenticationPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=pPortPrototypeInExecutable, pPort PrototypeInExecutable.contextRootSwComponent Prototype, pPortPrototypeInExecutable.context ComponentPrototype, pPortPrototypeInExecutable.target PPortPrototype **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. **Tags:** atp.Splitkey=process

Table 5.56: DiagnosticAuthenticationPortMapping

The meta-class `DiagnosticExternalAuthenticationPortMapping` is described in Chapter 5.2.3.6.

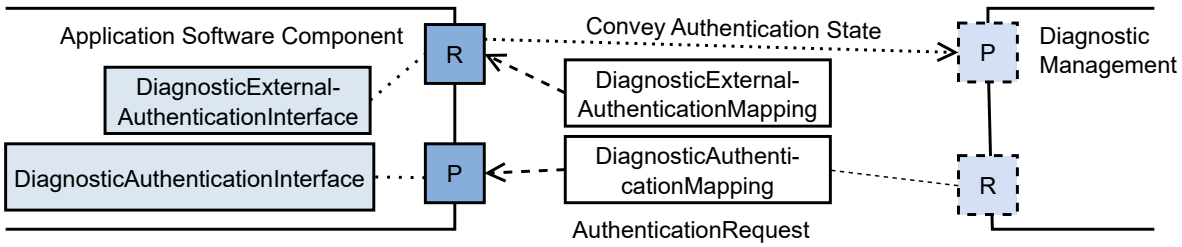


Figure 5.19: Example data exchange for diagnostic authentication

The two different roles mentioned in [TPS_MANI_01360] for the authentication of a diagnostic client are depicted in Figure 5.19.

[TPS_MANI_01361] Supporting the authentication request of a diagnostic client
⌈For the purpose of supporting the authentication request of a diagnostic client on the

AUTOSAR adaptive platform, the relation between a DiagnosticAuthentication and one PPortPrototype is created by using the DiagnosticAuthentication-PortMapping 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.

⌋

[constr_10093] Existence of the reference from DiagnosticAuthentication-PortMapping to DiagnosticAuthentication

Imposition time: IT_DiagDes

⌈Each DiagnosticAuthentication shall only be referenced by exactly one DiagnosticAuthenticationPortMapping.⌋

Class	***DiagnosticAuthentication*** (abstract)			
Note	This meta-class represents the ability to configure the usage of the UDS service Authentication in the Diagnostic extract.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticAuthTransmitCertificate, DiagnosticAuthenticationConfiguration, DiagnosticDeAuthentication, DiagnosticProofOfOwnership, DiagnosticVerifyCertificateBidirectional, DiagnosticVerifyCertificate Unidirectional			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authentication Class	Diagnostic AuthenticationClass	0..1	ref	This represents the corresponding "class", i.e. this meta-class provides properties that are shared among all instances of applicable sub-classes of DiagnosticService Instance. The subclasses that affected by this pattern implement references to the applicable "class"-role that substantiate this abstract reference. **Tags:** atp.Splitkey=authenticationClass

Table 5.57: DiagnosticAuthentication

Class	**DiagnosticAuthenticationClass**			
Note	This meta-class contains configuration shared by all instances of the Authentication diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticAuthentications			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 5.58: DiagnosticAuthenticationClass

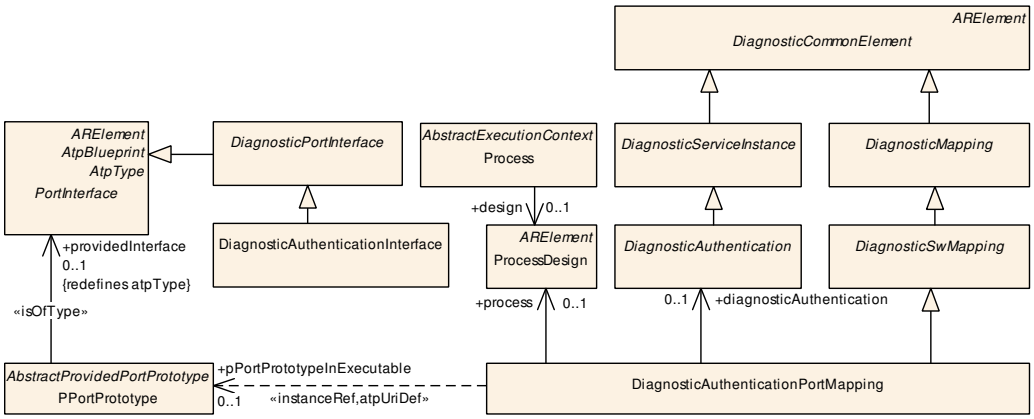


Figure 5.20: Modeling of `DiagnosticAuthenticationPortMapping` to forward the authentication request of a diagnostic client to an authentication manager

[constr_10092] Restriction for the applicability of meta-class `DiagnosticAuthenticationPortMapping`

Imposition time: `IT_DiagDes`

⌈If a `PPortPrototype` is referenced by a `DiagnosticAuthenticationPortMapping`, then the `PPortPrototype` shall be typed by a `DiagnosticAuthenticationInterface`.⌋

5.2.3.6 Diagnostic External Authentication Mapping

[TPS_MANI_01362] Conveying the authentication state of a diagnostic client to the diagnostic server instance ⌈For the purpose of conveying the authentication state of a diagnostic client to the diagnostic server instance of DM on the *AUTOSAR adaptive platform*, the relation between a `ProcessDesign` and one `RPortPrototype` is created by using the `DiagnosticExternalAuthenticationPortMapping` that refers to

- 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.

There is no model element in the context of the `Diagnostic Extract` to which the `DiagnosticExternalAuthenticationPortMapping` could reliably refer to (external authentication is not bound to the existence of UDS service 0x29).

This is the reason why the `DiagnosticExternalAuthenticationPortMapping` is restricted to references to `RPortPrototype` and `ProcessDesign`.⌋

[constr_10094] Restriction for the applicability of DiagnosticExternal-AuthenticationPortMapping

Imposition time: IT_DiagDes

⌈If an RPortPrototype is referenced by a DiagnosticExternal-AuthenticationPortMapping, then the RPortPrototype shall be typed by a DiagnosticExternalAuthenticationInterface.⌋

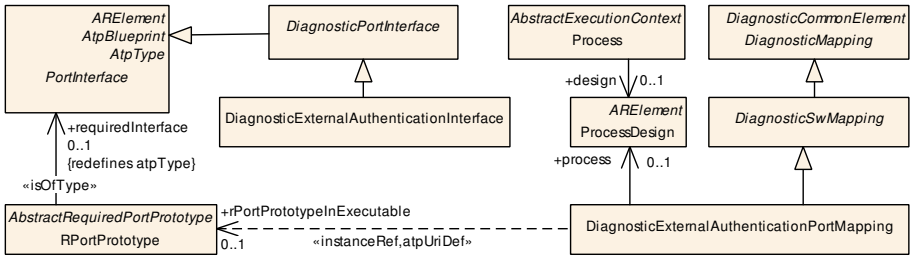


Figure 5.21: Modeling of DiagnosticExternalAuthenticationPortMapping to transmit the response to an authentication request back to the DM

Class	**DiagnosticExternalAuthenticationPortMapping**			
Note	This mapping class identifies the PortPrototype in the application software that handles the external authentication. **Tags:** atp.recommendedPackage=DiagnosticPortMapppings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
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. **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic ClientAuthenticationPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=rPortPrototypeInExecutable, rPort PrototypeInExecutable.contextRootSwComponent Prototype, rPortPrototypeInExecutable.context ComponentPrototype, rPortPrototypeInExecutable.target RPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 5.59: DiagnosticExternalAuthenticationPortMapping

5.2.3.7 Diagnostic Input/Output Control Mapping

[TPS_MANI_01527] Executing an I/O control request of the DM on the application software component ⌈For the purpose of executing an I/O control request of the DM

on the application software component on the *AUTOSAR adaptive platform*, the relation between a ProcessDesign and one PPortPrototype is created by using the DiagnosticInputOutputControlPortMapping that refers to

- a ProcessDesign in the role process,
- a DiagnosticDataIdentifier that carries the payload of the I/O control request, and
- a PPortPrototype, typed by a DiagnosticInputOutputControlInterface or a DiagnosticInputOutputControlGenericInterface, 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.

⌋

[constr_10796] Restriction for the applicability of DiagnosticInputOutputControlPortMapping

Imposition time: IT_DiagDes

⌈If a PPortPrototype is referenced by a DiagnosticInputOutputControlPortMapping, then the PPortPrototype shall be typed by either

- a DiagnosticInputOutputControlInterface or
- a DiagnosticInputOutputControlGenericInterface.

⌋

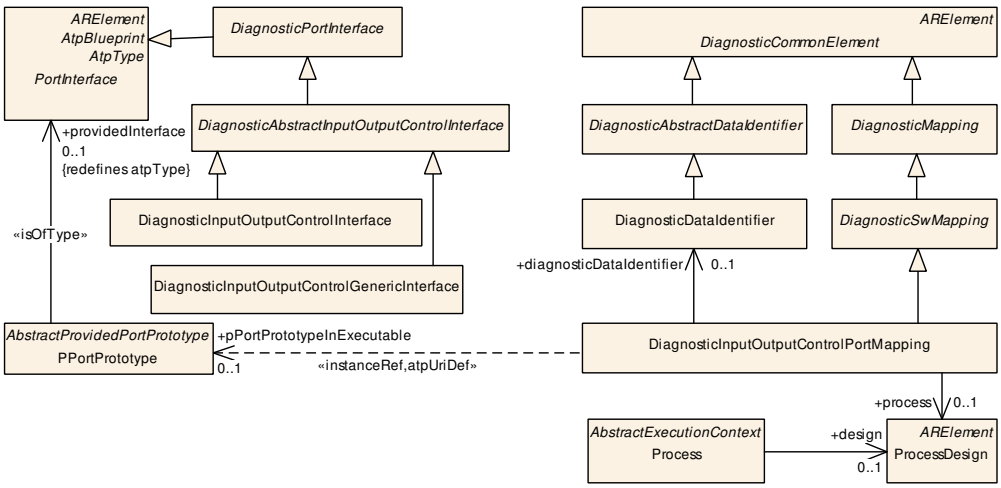


Figure 5.22: Modeling of DiagnosticInputOutputControlPortMapping to execute a diagnostic Input/Output control request of the DM on the application software component

Class	**DiagnosticInputOutputControlPortMapping**			
Note	This meta-class provides the ability to implement a DID-focused I/O Control. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnosticData Identifier	DiagnosticDataIdentifier	0..1	ref	This reference represents the applicable DiagnosticData Identfiier. **Tags:** atp.Splitkey=diagnosticDataIdentifier
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This reference identifies the applicable PPortPrototype on which the I/O Control is executed. The reference has the ability to point into the component hierarchy (under possible consideration of the rootSoftwareComposition). **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=pPortPrototypeInExecutable, pPort PrototypeInExecutable.contextRootSwComponent Prototype, pPortPrototypeInExecutable.context ComponentPrototype, pPortPrototypeInExecutable.target PPortPrototype **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	This reference represents the identification of the design-time representation for the Process that owns the reference. **Tags:** atp.Splitkey=process

Table 5.60: DiagnosticInputOutputControlPortMapping

5.2.4 SOVD-related Diagnostic Mapping

This chapter contains a documentation of some `DiagnosticMapping`s that are specific for the context of Service-Oriented Vehicle Diagnostics (SOVD).

5.2.4.1 Diagnostic SOVD Proximity Challenge Mapping

[TPS_MANI_01424] Semantics of `DiagnosticSovdProximityChallengePortMapping` ⌈On the *AUTOSAR adaptive platform*, the relation between a `PPortPrototype` and a `ProcessDesign` is created by using the `DiagnosticSovdProximityChallengePortMapping` that refers to

- a `ProcessDesign` in the role `process`, and
- a `PPortPrototype` typed by a `DiagnosticSovdProximityChallengeInterface` 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.

⌋

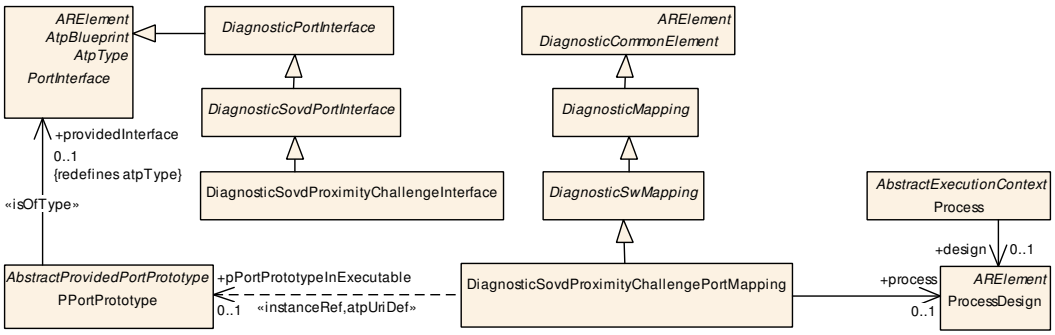


Figure 5.23: Modeling of the DiagnosticSovdProximityChallengePortMapping

[constr_10442] Restriction for the applicability of DiagnosticSovdProximity-ChallengePortMapping

Imposition time: IT_DiagDes

⌈If a PPortPrototype is referenced by a DiagnosticSovdProximity-ChallengePortMapping in the role pPortPrototypeInExecutable, then the PPortPrototype shall be typed by a DiagnosticSovdProximityChallenge-Interface.⌋

[constr_10443] Restriction for the applicability of DiagnosticSovdProximity-ChallengeInterface

Imposition time: IT_DiagDes

⌈If a PPortPrototype is typed by a DiagnosticSovdProximityChallenge-Interface, then the PortPrototype shall only be referenced in the role pPort-PrototypeInExecutable by a DiagnosticSovdProximityChallengePort-Mapping.⌋

Only one implementation of an SOVD proximity challenge is allowed in the context of a DiagnosticContributionSet. This restriction is formalized by means of [constr_10444].

[constr_10444] Existence of DiagnosticSovdProximityChallengePort-Mapping

Imposition time: IT_DiagDes

⌈Each DiagnosticContributionSet shall only reference at most one DiagnosticSovdProximityChallengePortMapping in the role element.⌋

Class	**DiagnosticSovdProximityChallengePortMapping**			
Note	This mapping class identifies the PortPrototype in the application software that handles the SOVD proximity challenge. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdProximityChallengePortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=pPortPrototypeInExecutable, pPort PrototypeInExecutable.contextRootSwComponent Prototype, pPortPrototypeInExecutable.context ComponentPrototype, pPortPrototypeInExecutable.target PPortPrototype **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. **Tags:** atp.Splitkey=process

Table 5.61: DiagnosticSovdProximityChallengePortMapping

[constr_10445] Existence of the reference in the role `DiagnosticSovdProximityChallengePortMapping.pPortPrototypeInExecutable`

Imposition time: `IT_DiagDes`

⌈For each `DiagnosticSovdProximityChallengePortMapping`, the reference in the role `DiagnosticSovdProximityChallengePortMapping.pPortPrototypeInExecutable` shall exist.⌋

[constr_10446] Existence of the reference in the role `DiagnosticSovdProximityChallengePortMapping.process`

Imposition time: `IT_DiagDes`

⌈For each `DiagnosticSovdProximityChallengePortMapping`, the reference in the role `DiagnosticSovdProximityChallengePortMapping.process` shall exist.⌋

5.2.4.2 Diagnostic SOVD Authorization Port Mapping

[TPS_MANI_01425] Semantics of `DiagnosticSovdAuthorizationPortMapping` ⌈On the *AUTOSAR adaptive platform*, the relation between a `PPortPrototype` and a `ProcessDesign` is created by using the `DiagnosticSovdAuthorizationPortMapping` that refers to

- a `ProcessDesign` in the role `process`, and

- a PPortPrototype typed by a DiagnosticSovdAuthorization-Interface 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.

⌋

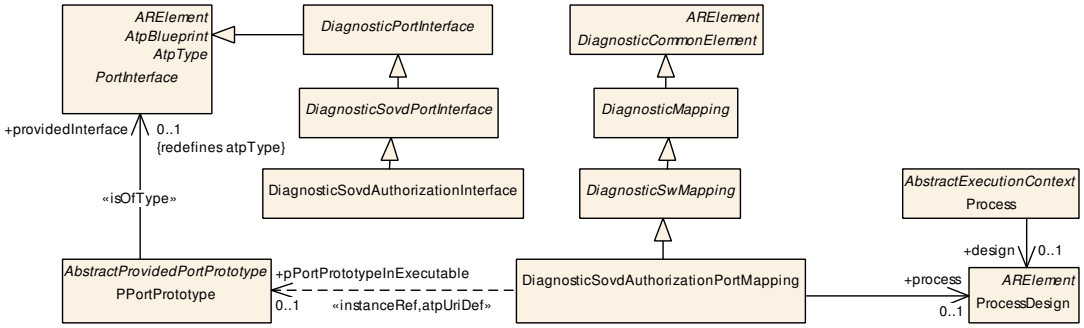


Figure 5.24: Modeling of the DiagnosticSovdAuthorizationPortMapping

[constr_10447] Restriction for the applicability of DiagnosticSovd-AuthorizationPortMapping

Imposition time: IT_DiagDes

⌈If a PPortPrototype is referenced in the role pPortPrototypeInExecutable by a DiagnosticSovdAuthorizationPortMapping, then the PPortPrototype shall be typed by a DiagnosticSovdAuthorizationInterface.⌋

[constr_10448] Restriction for the applicability of DiagnosticSovd-AuthorizationInterface

Imposition time: IT_DiagDes

⌈If a PPortPrototype is typed by a DiagnosticSovdAuthorization-Interface, then the PortPrototype shall only be referenced in the role pPortPrototypeInExecutable by a DiagnosticSovdAuthorizationPort-Mapping.⌋

Only one implementation of an SOVD authorization is allowed in the context of a DiagnosticContributionSet. This restriction is formalized by means of [constr_10449].

[constr_10449] Existence of DiagnosticSovdAuthorizationPortMapping

Imposition time: IT_DiagDes

⌈Each DiagnosticContributionSet shall only reference in the role element at most one DiagnosticSovdAuthorizationPortMapping.⌋

Class	DiagnosticSovdAuthorizationPortMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the SOVD authorization. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdAuthorizationPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=pPortPrototypeInExecutable, pPort PrototypeInExecutable.contextRootSwComponent Prototype, pPortPrototypeInExecutable.context ComponentPrototype, pPortPrototypeInExecutable.target PPortPrototype **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. **Tags:** atp.Splitkey=process

Table 5.62: DiagnosticSovdAuthorizationPortMapping

[constr_10450] Existence of the reference in the role `DiagnosticSovdAuthorizationPortMapping.pPortPrototypeInExecutable`

Imposition time: `IT_DiagDes`

⌈For each `DiagnosticSovdAuthorizationPortMapping`, the reference in the role `DiagnosticSovdAuthorizationPortMapping.pPortPrototypeInExecutable` shall exist.⌋

[constr_10451] Existence of the reference in the role `DiagnosticSovdAuthorizationPortMapping.process`

Imposition time: `IT_DiagDes`

⌈For each `DiagnosticSovdAuthorizationPortMapping`, the reference in the role `DiagnosticSovdAuthorizationPortMapping.process` shall exist.⌋

5.2.4.3 Diagnostic SOVD Bulk Data Port Mapping

[TPS_MANI_01426] Mapping of `DiagnosticSovdBulkData` to `PortPrototype` on the *AUTOSAR adaptive platform* ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticSovdBulkData` and one `PPortPrototype` is created by using the `DiagnosticSovdBulkDataPortMapping` that refers to

- a `DiagnosticSovdBulkData` in the role `serviceInstance`,
- a `ProcessDesign` in the role `process`, and

- a PPortPrototype typed by a DiagnosticSovdBulkDataInterface 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.

⌋

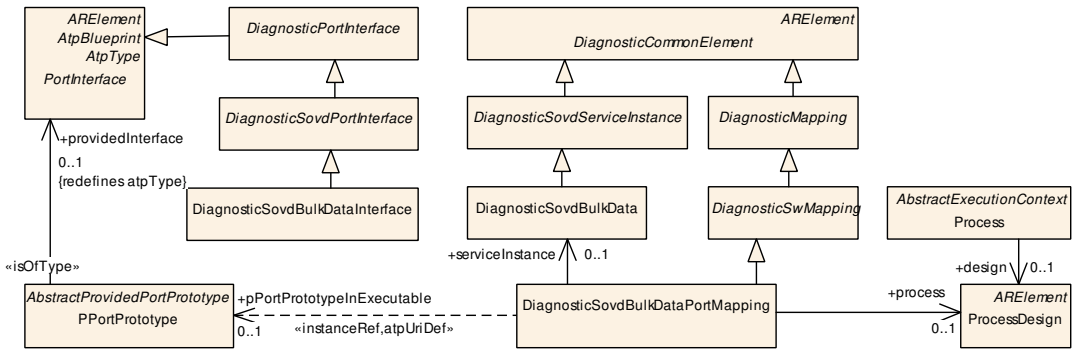


Figure 5.25: Modeling of the DiagnosticSovdBulkDataPortMapping

[constr_10452] Restriction for the applicability of DiagnosticSovdBulkDataPortMapping

Imposition time: IT_DiagDes

⌈If a PPortPrototype is referenced in the role pPortPrototypeInExecutable by a DiagnosticSovdBulkDataPortMapping, then the PPortPrototype shall be typed by a DiagnosticSovdBulkDataInterface.⌋

[constr_10453] Restriction for the applicability of DiagnosticSovdBulkDataInterface

Imposition time: IT_DiagDes

⌈If a PPortPrototype is typed by a DiagnosticSovdBulkDataInterface, then the PortPrototype shall only be referenced in the role pPortPrototypeInExecutable by a DiagnosticSovdBulkDataPortMapping.⌋

[constr_10454] Uniqueness of reference from DiagnosticSovdBulkDataPortMapping to DiagnosticSovdBulkData

Imposition time: IT_DiagDes

⌈Each instance of DiagnosticSovdBulkData shall only be referenced in the role serviceInstance by exactly one DiagnosticSovdBulkDataPortMapping.⌋

Class	**DiagnosticSovdBulkDataPortMapping**			
Note	This mapping associates a PPortPrototype typed by a DiagnosticSovdBulkDataInterface to the corresponding SOVD service instance that is modeled as DiagnosticSovdBulkData. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdBulkDataPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=pPortPrototypeInExecutable, pPort PrototypeInExecutable.contextRootSwComponent Prototype, pPortPrototypeInExecutable.context ComponentPrototype, pPortPrototypeInExecutable.target PPortPrototype **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. **Tags:** atp.Splitkey=process
serviceInstance	DiagnosticSovdBulk Data	0..1	ref	This reference identifies the applicable diagnostic SOVD service instance. **Tags:** atp.Splitkey=serviceInstance

Table 5.63: DiagnosticSovdBulkDataPortMapping

[constr_10455] Existence of the reference in the role DiagnosticSovdBulkDataPortMapping.serviceInstance

Imposition time: IT_DiagDes

⌈For each DiagnosticSovdBulkDataPortMapping, the reference in the role DiagnosticSovdBulkDataPortMapping.serviceInstance shall exist.⌋

[constr_10456] Existence of the reference in the role DiagnosticSovdBulkDataPortMapping.pPortPrototypeInExecutable

Imposition time: IT_DiagDes

⌈For each DiagnosticSovdBulkDataPortMapping, the reference in the role DiagnosticSovdBulkDataPortMapping.pPortPrototypeInExecutable shall exist.⌋

[constr_10457] Existence of the reference in the role DiagnosticSovdBulkDataPortMapping.process

Imposition time: IT_DiagDes

⌈For each DiagnosticSovdBulkDataPortMapping, the reference in the role DiagnosticSovdBulkDataPortMapping.process shall exist.⌋

5.2.4.4 Diagnostic SOVD Update Port Mapping

[TPS_MANI_01427] Mapping of `DiagnosticSovdUpdate` to `PortPrototype` on the *AUTOSAR adaptive platform* ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticSovdUpdate` and one `PPortPrototype` is created by using the `DiagnosticSovdUpdatePortMapping` that refers to

- a `DiagnosticSovdUpdate` in the role `serviceInstance`,
- a `ProcessDesign` in the role `process`, and
- a `PPortPrototype` typed by a `DiagnosticSovdUpdateInterface` 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.

⌋

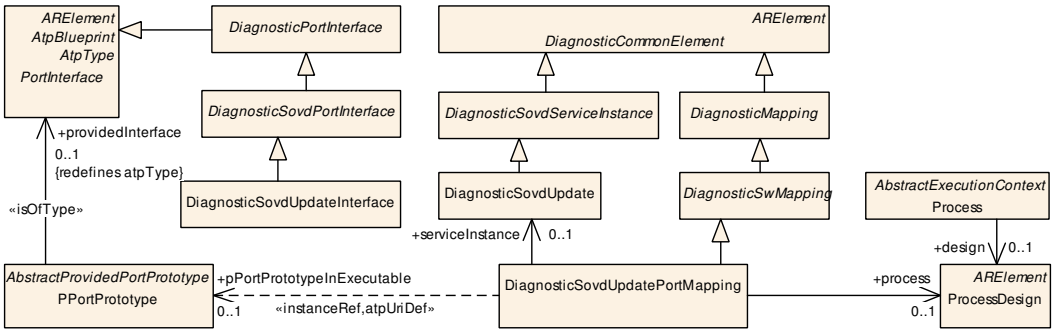


Figure 5.26: Modeling of the `DiagnosticSovdUpdatePortMapping`

[constr_10458] Restriction for the applicability of `DiagnosticSovdUpdatePortMapping`

Imposition time: `IT_DiagDes`

⌈If a `PPortPrototype` is referenced in the role `pPortPrototypeInExecutable` by a `DiagnosticSovdUpdatePortMapping`, then the `PPortPrototype` shall be typed by a `DiagnosticSovdUpdateInterface`.⌋

[constr_10459] Restriction for the applicability of `DiagnosticSovdUpdateInterface`

Imposition time: `IT_DiagDes`

⌈If a `PPortPrototype` is typed by a `DiagnosticSovdUpdateInterface`, then the `PortPrototype` shall only be referenced in the role `pPortPrototypeInExecutable` by a `DiagnosticSovdUpdatePortMapping`.⌋

[constr_10460] Uniqueness of reference from DiagnosticSovdUpdatePortMapping to DiagnosticSovdUpdate

Imposition time: IT_DiagDes

⌈Each instance of DiagnosticSovdUpdate shall only be referenced in the role serviceInstance by exactly one DiagnosticSovdUpdatePortMapping.⌋

Class	**DiagnosticSovdUpdatePortMapping**			
Note	This mapping associates a PPortPrototype typed by an DiagnosticSovdUpdateInterface with the corresponding SOVD service instance that is modeled as a DiagnosticSovdUpdate. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdUpdatePortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=pPortPrototypeInExecutable, pPort PrototypeInExecutable.contextRootSwComponent Prototype, pPortPrototypeInExecutable.context ComponentPrototype, pPortPrototypeInExecutable.target PPortPrototype **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. **Tags:** atp.Splitkey=process
serviceInstance	DiagnosticSovdUpdate	0..1	ref	This reference identifies the applicable diagnostic SOVD service instance. **Tags:** atp.Splitkey=serviceInstance atp.Status=candidate

Table 5.64: DiagnosticSovdUpdatePortMapping

[constr_10461] Existence of the reference in the role DiagnosticSovdUpdatePortMapping.serviceInstance

Imposition time: IT_DiagDes

⌈For each DiagnosticSovdUpdatePortMapping, the reference in the role DiagnosticSovdUpdatePortMapping.serviceInstance shall exist.⌋

[constr_10462] Existence of the reference in the role DiagnosticSovdUpdatePortMapping.pPortPrototypeInExecutable

Imposition time: IT_DiagDes

⌈For each DiagnosticSovdUpdatePortMapping, the reference in the role DiagnosticSovdUpdatePortMapping.pPortPrototypeInExecutable shall exist.⌋

[constr_10463] Existence of the reference in the role DiagnosticSovdUpdatePortMapping.process

Imposition time: IT_DiagDes

⌈For each DiagnosticSovdUpdatePortMapping, the reference in the role DiagnosticSovdUpdatePortMapping.process shall exist.⌋

5.2.4.5 Diagnostic SOVD Service Validation Port Mapping

[TPS_MANI_01428] Mapping of DiagnosticSovdServiceInstance to PortPrototype on the *AUTOSAR adaptive platform* ⌈On the *AUTOSAR adaptive platform*, the relation between a PPortPrototype and a ProcessDesign is created by using the DiagnosticSovdServiceValidationPortMapping that refers to

- a ProcessDesign in the role process, and
- a PPortPrototype typed by a DiagnosticSovdServiceValidationInterface 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.

⌋

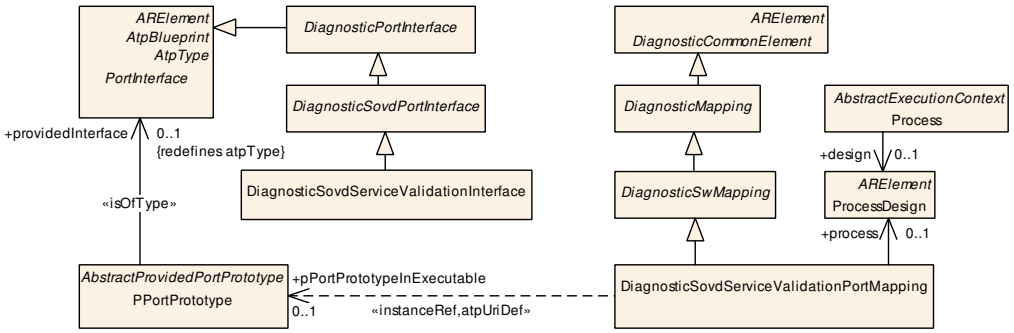


Figure 5.27: Modeling of the DiagnosticSovdServiceValidationPortMapping

[constr_10464] Restriction for the applicability of DiagnosticSovdServiceValidationPortMapping

Imposition time: IT_DiagDes

⌈If a PPortPrototype is referenced in the role pPortPrototypeInExecutable by a DiagnosticSovdServiceValidationPortMapping, then the PPortPrototype shall be typed by a DiagnosticSovdServiceValidationInterface.⌋

[constr_10465] Restriction for the applicability of DiagnosticSovdService-ValidationInterface

Imposition time: IT_DiagDes

⌈If a PPortPrototype is typed by a DiagnosticSovdServiceValidation-Interface, then the PortPrototype shall only be referenced in the role pPort-PrototypeInExecutable by a DiagnosticSovdServiceValidationPort-Mapping.⌋

Class	**DiagnosticSovdServiceValidationPortMapping**			
Note	This mapping class identifies the PortPrototype in the application software that handles the SOVD service validation. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdValidationPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=pPortPrototypeInExecutable, pPort PrototypeInExecutable.contextRootSwComponent Prototype, pPortPrototypeInExecutable.context ComponentPrototype, pPortPrototypeInExecutable.target PPortPrototype **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. **Tags:** atp.Splitkey=process

Table 5.65: DiagnosticSovdServiceValidationPortMapping

[constr_10466] Existence of the reference in the role DiagnosticSovdService-ValidationPortMapping.pPortPrototypeInExecutable

Imposition time: IT_DiagDes

⌈For each DiagnosticSovdServiceValidationPortMapping, the reference in the role DiagnosticSovdServiceValidationPortMapping.pPort-PrototypeInExecutable shall exist.⌋

[constr_10467] Existence of the reference in the role DiagnosticSovdService-ValidationPortMapping.process

Imposition time: IT_DiagDes

⌈For each DiagnosticSovdServiceValidationPortMapping, the reference in the role DiagnosticSovdServiceValidationPortMapping.process shall exist.⌋

5.2.4.6 Diagnostic SOVD Configuration Port Mapping

[TPS_MANI_01429] Mapping of `DiagnosticSovdConfiguration` to `PortPrototype` on the *AUTOSAR adaptive platform* ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticSovdConfiguration` and one `PPortPrototype` is created by using the `DiagnosticSovdConfigurationPortMapping` that refers to

- a `DiagnosticSovdConfiguration` in the role `serviceInstance`,
- a `ProcessDesign` in the role `process`, and
- a `PPortPrototype` typed by a `DiagnosticSovdConfigurationInterface` 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.

⌋

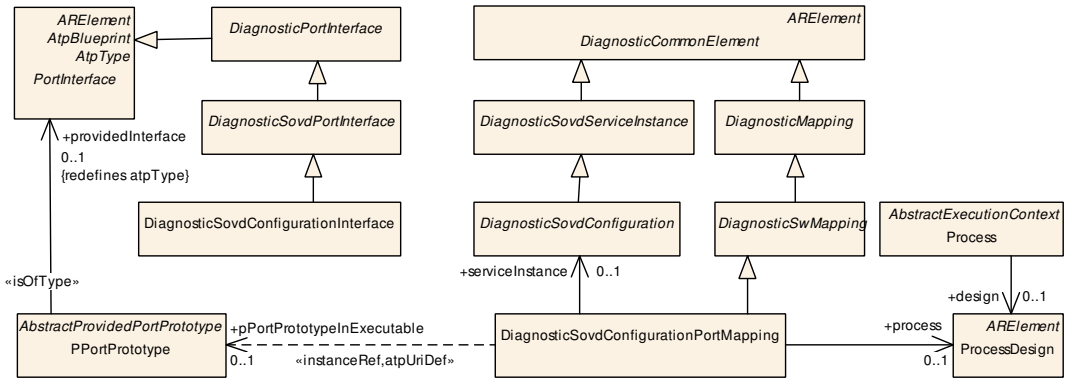


Figure 5.28: Modeling of the `DiagnosticSovdConfigurationPortMapping`

[constr_10468] Restriction for the applicability of meta-class `DiagnosticSovdConfigurationPortMapping`

Imposition time: `IT_DiagDes`

⌈If a `PPortPrototype` is referenced in the role `pPortPrototypeInExecutable` by a `DiagnosticSovdConfigurationPortMapping`, then the `PPortPrototype` shall be typed by a `DiagnosticSovdConfigurationInterface`.⌋

[constr_10469] Restriction for the applicability of meta-class `DiagnosticSovdConfigurationInterface`

Imposition time: `IT_DiagDes`

⌈If a `PPortPrototype` is typed by a `DiagnosticSovdConfigurationInterface`, then the `PortPrototype` shall only be referenced in the role `pPortPrototypeInExecutable` by a `DiagnosticSovdConfigurationPortMapping`.⌋

[constr_10470] Uniqueness of the reference in the role DiagnosticSovdConfigurationPortMapping.serviceInstance

Imposition time: IT_DiagDes

⌈Each instance of a sub-class of DiagnosticSovdConfiguration shall only be referenced in the role serviceInstance by exactly one DiagnosticSovdConfigurationPortMapping.⌋

Class	**DiagnosticSovdConfigurationPortMapping**			
Note	This mapping associates a PPortPrototype typed by a DiagnosticSovdConfigurationInterface to the corresponding SOVD service instance that is modeled as DiagnosticSovdConfiguration. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
pPortPrototypeInExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdConfigurationPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=pPortPrototypeInExecutable, pPortPrototypeInExecutable.contextRootSwComponentPrototype, pPortPrototypeInExecutable.contextComponentPrototype, pPortPrototypeInExecutable.targetPPortPrototype **InstanceRef implemented by:** PPortPrototypeInExecutableInstanceRef
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. **Tags:** atp.Splitkey=process
serviceInstance	DiagnosticSovdConfiguration	0..1	ref	This reference identifies the applicable diagnostic SOVD service instance. **Tags:** atp.Splitkey=serviceInstance

Table 5.66: DiagnosticSovdConfigurationPortMapping

[constr_10471] Existence of the reference in the role DiagnosticSovdConfigurationPortMapping.serviceInstance

Imposition time: IT_DiagDes

⌈For each DiagnosticSovdConfigurationPortMapping, the reference in the role DiagnosticSovdConfigurationPortMapping.serviceInstance shall exist.⌋

[constr_10472] Existence of the reference in the role DiagnosticSovdConfigurationPortMapping.pPortPrototypeInExecutable

Imposition time: IT_DiagDes

⌈For each DiagnosticSovdConfigurationPortMapping, the reference in the role DiagnosticSovdConfigurationPortMapping.pPortPrototypeInExecutable shall exist.⌋

[constr_10473] Existence of the reference in the role DiagnosticSovdConfigurationPortMapping.process

Imposition time: IT_DiagDes

⌈For each DiagnosticSovdConfigurationPortMapping, the reference in the role DiagnosticSovdConfigurationPortMapping.process shall exist.⌋

5.2.4.7 Diagnostic SOVD Data Port Mapping

Mapping a PPortPrototype for the purpose of defining the exchange of either SOVD Data or SOVD Data Element can be done by means of meta-class DiagnosticSovdContentPortMapping.

From the technical perspective, this is achieved by defining abstract meta-classes for the applicable PortInterface (in the form of the DiagnosticAbstractSovdContentInterface) and also for the "diagnostic content" (in the form of DiagnosticAbstractSovdContent).

Nevertheless, constraints are necessary to make sure that on both ends matching sub-classes of the mentioned abstract meta-classes become actual targets of a DiagnosticSovdContentPortMapping.

[TPS_MANI_01490] Mapping of DiagnosticAbstractSovdContent to PortPrototype on the *AUTOSAR adaptive platform* ⌈On the *AUTOSAR adaptive platform*, the relation between a DiagnosticSovdContent and one PPortPrototype is created by using the DiagnosticSovdContentPortMapping that refers to

- a DiagnosticAbstractSovdContent in the role content,
- a ProcessDesign in the role process, and
- a PPortPrototype typed by a DiagnosticAbstractSovdContentInterface 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.

⌋

[constr_10696] Target of the reference DiagnosticSovdContentPortMapping.pPortPrototypeInExecutable

Imposition time: IT_DiagDes

⌈Any DiagnosticSovdContentPortMapping shall **only** reference in the role pPortPrototypeInExecutable a PPortPrototype that is typed by either

- a DiagnosticSovdContentInterface or

- a DiagnosticSovdContentElementInterface.

⌋

[constr_10697] DiagnosticSovdContentPortMapping maps on the level of SOVD Data

Imposition time: IT_DiagDes

⌈If a DiagnosticSovdContentPortMapping references in the role pPortPrototypeInExecutable a PPortPrototype that is typed by a DiagnosticSovdContentInterface, then the DiagnosticSovdContentPortMapping shall also refer in the role content to a DiagnosticSovdContent, and vice versa.⌋

[constr_10698] DiagnosticSovdContentPortMapping maps on the level of SOVD Data Element

Imposition time: IT_DiagDes

⌈If a DiagnosticSovdContentPortMapping references in the role pPortPrototypeInExecutable a PPortPrototype that is typed by a DiagnosticSovdContentElementInterface, then the DiagnosticSovdContentPortMapping shall also refer in the role content to a DiagnosticSovdContentElement, and vice versa.⌋

For the time being, there are no formal constraints regarding a match between the elements of a PPortPrototype typed by either a DiagnosticSovdContentInterface or a DiagnosticSovdContentElementInterface and the DiagnosticAbstractSovdContent referenced in the role DiagnosticSovdContentPortMapping.content.

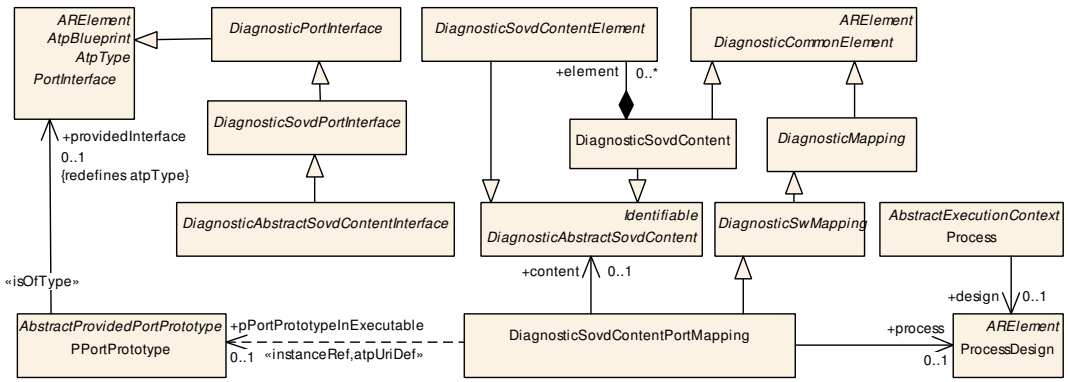


Figure 5.29: Modeling of the DiagnosticSovdContentPortMapping

Class	**DiagnosticSovdContentPortMapping**
Note	This mapping associates a PPortPrototype typed by either a DiagnosticSovdContentInterface or a DiagnosticSovdContentElementInterface to the corresponding SOVD content that is modeled as DiagnosticAbstractSovdContent. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.

▽

△

Class	DiagnosticSovdContentPortMapping			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
content	DiagnosticAbstractSovdContent	0..1	ref	This reference identifies the applicable content for the mapping. **Tags:** atp.Splitkey=content
pPortPrototypeInExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdContentPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=pPortPrototypeInExecutable, pPortPrototypeInExecutable.contextRootSwComponentPrototype, pPortPrototypeInExecutable.contextComponentPrototype, pPortPrototypeInExecutable.targetPPortPrototype **InstanceRef implemented by:** PPortPrototypeInExecutableInstanceRef
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. **Tags:** atp.Splitkey=process

Table 5.67: DiagnosticSovdContentPortMapping

[constr_10699] Existence of the reference in the role DiagnosticSovdContentPortMapping.pPortPrototypeInExecutable

Imposition time: IT_DiagDes

⌈For each DiagnosticSovdContentPortMapping, the reference in the role DiagnosticSovdContentPortMapping.pPortPrototypeInExecutable shall exist.⌋

[constr_10700] Existence of the reference in the role DiagnosticSovdContentPortMapping.content

Imposition time: IT_DiagDes

⌈For each DiagnosticSovdContentPortMapping, the reference in the role DiagnosticSovdContentPortMapping.content shall exist.⌋

[constr_10701] Existence of the reference in the role DiagnosticSovdContentPortMapping.process

Imposition time: IT_DiagDes

⌈For each DiagnosticSovdContentPortMapping, the reference in the role DiagnosticSovdContentPortMapping.process shall exist.⌋

5.2.4.8 Diagnostic SOVD Operation Port Mapping

[TPS_MANI_01491] Mapping of `DiagnosticSovdOperation` to a `PPortPrototype` on the *AUTOSAR adaptive platform* ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticSovdOperation` and one `PPortPrototype` is created by using the `DiagnosticSovdOperationPortMapping` that refers to

- a `DiagnosticSovdOperation` in the role `sovdOperation`,
- a `ProcessDesign` in the role `process`, and
- an `PPortPrototype` typed by a `DiagnosticSovdOperationInterface` 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.

⌋

[constr_10713] Target of the reference `DiagnosticSovdOperationPortMapping.pPortPrototypeInExecutable`

Imposition time: `IT_DiagDes`

⌈Any `DiagnosticSovdOperationPortMapping` shall **only** reference in the role `DiagnosticSovdOperationPortMapping.pPortPrototypeInExecutable` a `PPortPrototype` that is typed by a `DiagnosticSovdOperationInterface`.⌋

[constr_10714] `PPortPrototype` that is typed by a `DiagnosticSovdOperationInterface` as a mapping target

Imposition time: `IT_DiagDes`

⌈Any `PPortPrototype` that is typed by a `DiagnosticSovdOperationInterface` shall only be referenced in the role `DiagnosticSovdOperationPortMapping.pPortPrototypeInExecutable`.⌋

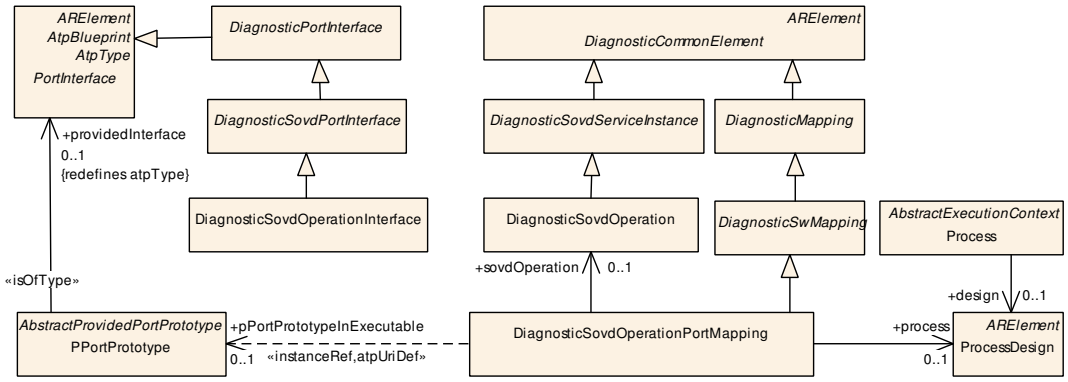


Figure 5.30: Modeling of the `DiagnosticSovdOperationPortMapping`

Class	DiagnosticSovdOperationPortMapping			
Note	This mapping associates a PPortPrototype typed by a DiagnosticSovdOperationInterface to the corresponding SOVD operation. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdOperationPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=pPortPrototypeInExecutable, pPort PrototypeInExecutable.contextRootSwComponent Prototype, pPortPrototypeInExecutable.context ComponentPrototype, pPortPrototypeInExecutable.target PPortPrototype **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. **Tags:** atp.Splitkey=process
sovdOperation	DiagnosticSovd Operation	0..1	ref	This reference identifies the applicable DiagnosticSovd Operation. **Tags:** atp.Splitkey=sovdOperation

Table 5.68: DiagnosticSovdOperationPortMapping

[constr_10702] Existence of the reference in the role `DiagnosticSovdOperationPortMapping.pPortPrototypeInExecutable`

Imposition time: `IT_DiagDes`

⌈For each `DiagnosticSovdOperationPortMapping`, the reference in the role `DiagnosticSovdOperationPortMapping.pPortPrototypeInExecutable` shall exist.⌋

[constr_10703] Existence of the reference in the role `DiagnosticSovdOperationPortMapping.sovdOperation`

Imposition time: `IT_DiagDes`

⌈For each `DiagnosticSovdOperationPortMapping`, the reference in the role `DiagnosticSovdOperationPortMapping.sovdOperation` shall exist.⌋

[constr_10704] Existence of the reference in the role `DiagnosticSovdOperationPortMapping.process`

Imposition time: `IT_DiagDes`

⌈For each `DiagnosticSovdOperationPortMapping`, the reference in the role `DiagnosticSovdOperationPortMapping.process` shall exist.⌋

5.2.4.9 Diagnostic SOVD Configuration to SOVD Content Mapping

[TPS_MANI_01515] Mapping of `DiagnosticSovdConfigurationParameter` to `DiagnosticSovdContent` on the *AUTOSAR adaptive platform* ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticSovdConfigurationParameter` and one `DiagnosticSovdContent` is created by using the `DiagnosticSovdConfigContentMapping` that refers to

- a `DiagnosticSovdConfigurationParameter` in the role `serviceInstance` and
- a `DiagnosticSovdContent` in the role `content`.

⌋

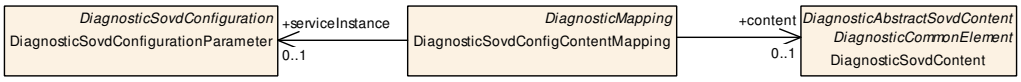


Figure 5.31: Modeling of the `DiagnosticSovdConfigContentMapping`

[constr_10733] Uniqueness of reference from `DiagnosticSovdConfigContentMapping` to `DiagnosticSovdConfigurationParameter`

Imposition time: `IT_DiagDes`

⌈Each instance of a sub-class of `DiagnosticSovdConfigurationParameter` shall only be referenced by exactly one `DiagnosticSovdConfigContentMapping`.⌋

Class	**DiagnosticSovdConfigContentMapping**			
Note	This mapping associates an SOVD configuration parameter service instance to an SOVD content. **Tags:** atp.recommendedPackage=DiagnosticMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
content	DiagnosticSovdContent	0..1	ref	This reference identifies the applicable SOVD content. **Tags:** atp.Splitkey=content
serviceInstance	DiagnosticSovd ConfigurationParameter	0..1	ref	This reference identifies the applicable service instance. **Tags:** atp.Splitkey=serviceInstance

Table 5.69: DiagnosticSovdConfigContentMapping

[constr_10734] Existence of the reference in the role `DiagnosticSovdConfigContentMapping.serviceInstance`

Imposition time: `IT_DiagDes`

⌈For each `DiagnosticSovdConfigContentMapping`, the reference in the role `DiagnosticSovdConfigContentMapping.serviceInstance` shall exist.⌋

[constr_10735] Existence of the reference in the role `DiagnosticSovdConfigContentMapping.content`

Imposition time: `IT_DiagDes`

⌈For each `DiagnosticSovdConfigContentMapping`, the reference in the role `DiagnosticSovdConfigContentMapping.content` shall exist.⌋

5.2.5 DoIP-related Diagnostic Mapping

5.2.5.1 Diagnostic DoIP Mapping

This section contains a description of the available diagnostic mappings that are applicable to DoIP communication. The mappings do not fully follow the pattern defined by other diagnostic mappings described in this document.

The difference is that the DoIP-related mapping classes (beyond referencing a `PortPrototype` and a `ProcessDesign`) do not reference any model element in the context of the diagnostic extract.

This works only because the *Diagnostic Management* searches for the existence of these mappings.

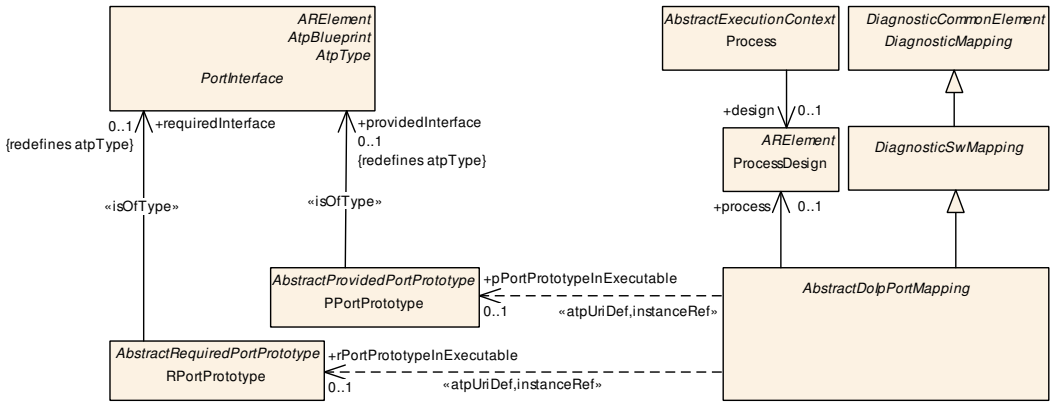


Figure 5.32: Modeling of `AbstractDoIpPortMapping` as an abstract base class for DoIP-related mappings

Because each of the mappings described in this section is supposed to exist **only once** in the context of the enclosing `DiagnosticContributionSet`, the *Diagnostic Management* is able to figure out the underlying semantics **based on the mere existence** of the respective mappings.

The DoIP-related diagnostic mapping is supported by the existence of the abstract base class `AbstractDoIpPortMapping`, that implements the heavy-lifting parts of the mapping relationship.

Class	***AbstractDoIpPortMapping*** (abstract)			
Note	This meta-class acts as an abstract base class for DoIP-related mappings. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticDoIpActivationLinePortMapping, DiagnosticDoIpEntityIdentificationPortMapping, DiagnosticDoIpGroupIdentificationPortMapping, DiagnosticDoIpPowerModePortMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
pPortPrototypeInExecutable	PPortPrototype	0..1	iref	This identifies the PPortPrototype for which the Abstract DoIpPortMapping is applicable. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=pPortPrototypeInExecutable, pPortPrototypeInExecutable.contextRootSwComponentPrototype, pPortPrototypeInExecutable.contextComponentPrototype, pPortPrototypeInExecutable.targetPPortPrototype **InstanceRef implemented by:** PPortPrototypeInExecutableInstanceRef
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. **Tags:** atp.Splitkey=process
rPortPrototypeInExecutable	RPortPrototype	0..1	iref	This identifies the RPortPrototype for which the Abstract DoIpPortMapping is applicable. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=rPortPrototypeInExecutable, rPortPrototypeInExecutable.contextRootSwComponentPrototype, rPortPrototypeInExecutable.contextComponentPrototype, rPortPrototypeInExecutable.targetRPortPrototype **InstanceRef implemented by:** RPortPrototypeInExecutableInstanceRef

Table 5.70: AbstractDoIpPortMapping

[constr_10609] Existence of references in the role `AbstractDoIpPortMapping.rPortPrototypeInExecutable` and `AbstractDoIpPortMapping.pPortPrototypeInExecutable`

Imposition time: `IT_DiagDes`

⌈For each `AbstractDoIpPortMapping`, only one of the references in the role

- `AbstractDoIpPortMapping.rPortPrototypeInExecutable`
- `AbstractDoIpPortMapping.pPortPrototypeInExecutable`

shall exist.⌋

5.2.5.2 Diagnostic DoIP Group Identification Mapping

[TPS_MANI_01460] Retrieving the status of the DoIP group identification to the diagnostic server instance ⌈For the purpose of retrieving the status of the DoIP group identification to the diagnostic server instance on the *AUTOSAR adaptive platform*, the

relation between a ProcessDesign and one PPortPrototype is created by using the DiagnosticDoIpGroupIdentificationPortMapping that refers to

- a ProcessDesign in the role process, and
- a PPortPrototype typed by a DiagnosticDoIPGroupIdentification-Interface 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.

There is no model element in the context of the Diagnostic Extract to which the DiagnosticDoIpGroupIdentificationPortMapping could reliably refer to.

This is the reason why the DiagnosticDoIpGroupIdentificationPortMapping is restricted to references to PPortPrototype and ProcessDesign.⌋

[constr_10586] Restriction for the applicability of DiagnosticDoIpGroup-IdentificationPortMapping

Imposition time: IT_DiagDes

⌈If a PPortPrototype is referenced by a DiagnosticDoIpGroup-IdentificationPortMapping, then the PPortPrototype shall be typed by a DiagnosticDoIPGroupIdentificationInterface.⌋

[constr_10587] Restriction for the applicability of DiagnosticDoIPGroup-IdentificationInterface

Imposition time: IT_DiagDes

⌈If a PPortPrototype is typed by a DiagnosticDoIPGroupIdentification-Interface, then the reference in the role pPortPrototypeInExecutable to the PPortPrototype shall only originate from a DiagnosticDoIpGroup-IdentificationPortMapping.⌋

For more information, see [SWS_DM_00813].

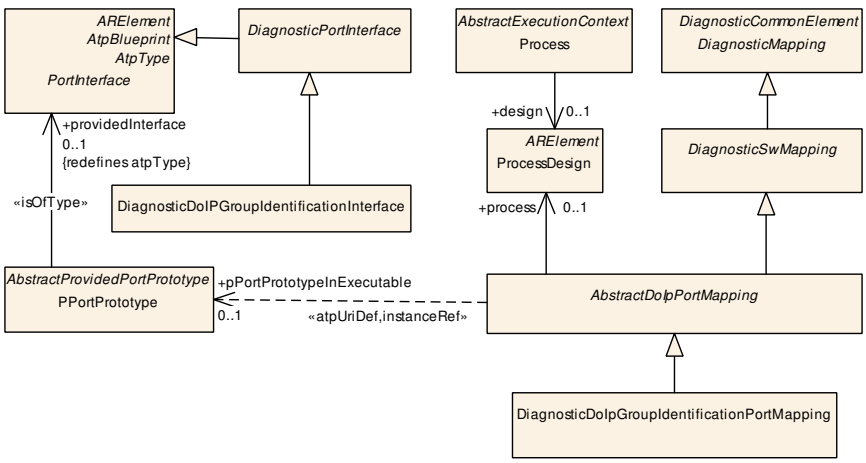


Figure 5.33: Modeling of DiagnosticDoIpGroupIdentificationPortMapping to indicate the existence of a handler for the DoIP group identification

Class	DiagnosticDoIpGroupIdentificationPortMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the DoIP group identification. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractDoIpPortMapping*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 5.71: DiagnosticDoIpGroupIdentificationPortMapping

[constr_10588] Existence of DiagnosticDoIpGroupIdentificationPortMapping.pPortPrototypeInExecutable

Imposition time: IT_DiagDes

⌈For each DiagnosticDoIpGroupIdentificationPortMapping, attribute pPortPrototypeInExecutable shall exist.⌋

[constr_10589] Existence of DiagnosticDoIpGroupIdentificationPortMapping.process

Imposition time: IT_DiagDes

⌈For each DiagnosticDoIpGroupIdentificationPortMapping, attribute process shall exist.⌋

5.2.5.3 Diagnostic DoIP Activation Line Mapping

[TPS_MANI_01461] Retrieving the status of the DoIP activation line to the diagnostic server instance ⌈For the purpose of retrieving the status of the DoIP activation line to the diagnostic server instance on the *AUTOSAR adaptive platform*, the relation between a ProcessDesign, a DoIpNetworkConfigurationDesign, and one PPortPrototype is created by using the DiagnosticDoIpActivationLinePortMapping that refers to

- a ProcessDesign in the role process,
- a DoIpNetworkConfigurationDesign in the role networkConfigurationDesign, and
- a PPortPrototype typed by a DiagnosticDoIPActivationLineInterface 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.

⌋

[constr_10590] Restriction for the applicability of DiagnosticDoIp-ActivationLinePortMapping

Imposition time: IT_DiagDes

⌈If a PPortPrototype is referenced in the role pPortPrototypeInExecutable by a DiagnosticDoIpActivationLinePortMapping, then the PPortPrototype shall be typed by a DiagnosticDoIPActivationLineInterface.⌋

[constr_10591] Restriction for the applicability of DiagnosticDo-IPActivationLineInterface

Imposition time: IT_DiagDes

⌈If a PPortPrototype is typed by a DiagnosticDoIPActivationLine-Interface, then the reference in the role pPortPrototypeInExecutable to the PPortPrototype shall only originate from a DiagnosticDoIpActivationLine-PortMapping.⌋

For more information, see [SWS_DM_00816].

[constr_10648] Target of the reference in the role DiagnosticDoIpActivation-LinePortMapping.networkConfigurationDesign

Imposition time: IT_DiagDes

⌈A DoIpNetworkConfiguration shall only be referenced at most once in the role DiagnosticDoIpActivationLinePortMapping.networkConfiguration-Design.⌋

[constr_10649] Existence of reference in the role DiagnosticDoIpActivation-LinePortMapping.networkConfigurationDesign

Imposition time: IT_DiagDes

⌈If a DoIpNetworkConfiguration exists where attribute isActivationLine-Dependent exists and is set to true, then a DiagnosticDoIpActivationLine-PortMapping shall exist that references the DoIpNetworkConfiguration in the role networkConfigurationDesign.⌋

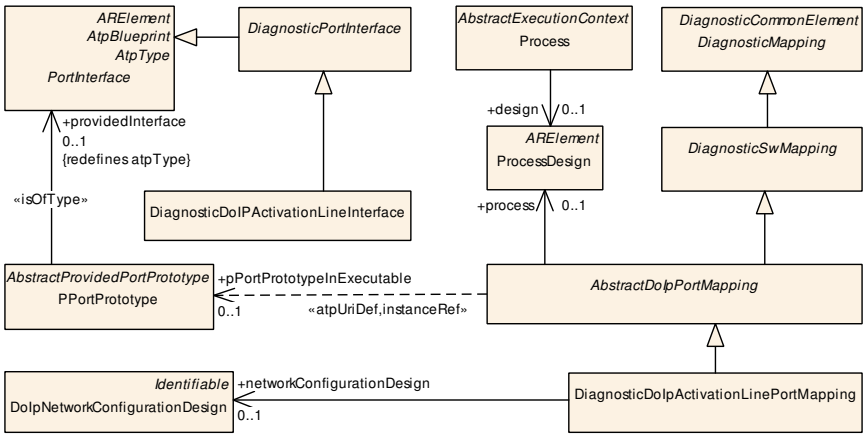


Figure 5.34: Modeling of DiagnosticDoIpActivationLinePortMapping to indicate the existence of a handler for the DoIP activation line

Class	DiagnosticDoIpActivationLinePortMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the DoIP activation line. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractDoIpPortMapping*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
network Configuration Design	DoIpNetwork ConfigurationDesign	0..1	ref	This reference identifies the applicable DoIpNetwork ConfigurationDesign. **Tags:** atp.Splitkey=networkConfigurationDesign

Table 5.72: DiagnosticDoIpActivationLinePortMapping

[constr_10592] Existence of the reference in the role DiagnosticDoIp-ActivationLinePortMapping.pPortPrototypeInExecutable

Imposition time: IT_DiagDes

⌈For each DiagnosticDoIpActivationLinePortMapping, attribute pPort-PrototypeInExecutable shall exist.⌋

[constr_10593] Existence of the reference in the role DiagnosticDoIp-ActivationLinePortMapping.process

Imposition time: IT_DiagDes

⌈For each DiagnosticDoIpActivationLinePortMapping, attribute process shall exist.⌋

5.2.5.4 Diagnostic DoIP Entity Identification Mapping

[TPS_MANI_01463] Retrieving the DoIP entity identification (EID) to the diagnostic server instance ⌈For the purpose of retrieving the DoIP entity identification (EID) to the diagnostic server instance on the *AUTOSAR adaptive platform*, the relation between a ProcessDesign and one PPortPrototype is created by using the DiagnosticDoIpEntityIdentificationPortMapping that refers to

- a ProcessDesign in the role process, and
- a PPortPrototype typed by a DiagnosticDoIPEntityIdentification-Interface 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.

There is no model element in the context of the Diagnostic Extract to which the DiagnosticDoIpEntityIdentificationPortMapping could reliably refer to.

This is the reason why the DiagnosticDoIpEntityIdentificationPort-Mapping is restricted to references to PPortPrototype and ProcessDesign.⌋

[constr_10598] Restriction for the applicability of DiagnosticDoIpEntity-IdentificationPortMapping

Imposition time: IT_DiagDes

⌈If a PPortPrototype is referenced in the role pPortPrototypeInExecutable by a DiagnosticDoIpEntityIdentificationPortMapping, then the PPort-Prototype shall be typed by a DiagnosticDoIPEntityIdentification-Interface.⌋

[constr_10599] Restriction for the applicability of DiagnosticDoIPEntity-IdentificationInterface

Imposition time: IT_DiagDes

⌈If a PPortPrototype is typed by a DiagnosticDoIPEntityIdentification-Interface, then the reference in the role pPortPrototypeInExecutable to the PPortPrototype shall only originate from a DiagnosticDoIpEntity-IdentificationPortMapping.⌋

For more information, see [SWS_DM_01361].

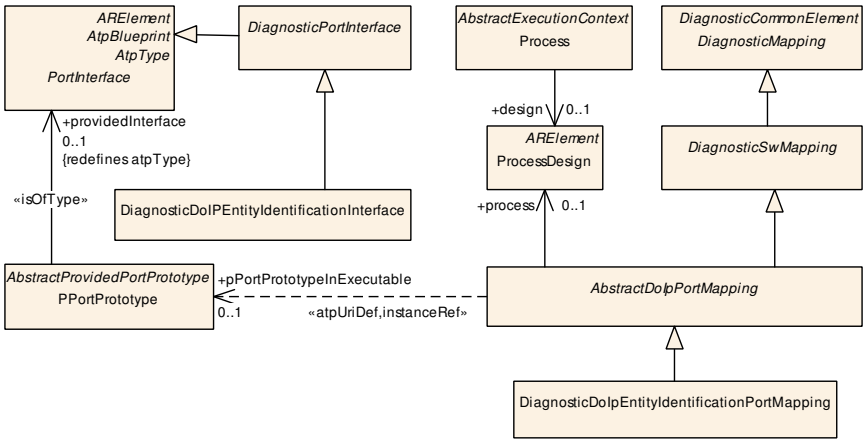


Figure 5.35: Modeling of DiagnosticDoIpEntityIdentificationPortMapping to indicate the existence of a handler for the DoIP entity identification

Class	**DiagnosticDoIpEntityIdentificationPortMapping**
Note	This mapping class identifies the PortPrototype in the application software that handles the DoIP identity identification. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARElement*, *ARObject*, *AbstractDoIpPortMapping*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*
Aggregated by	ARPackage.element

△

Class	DiagnosticDoIpEntityIdentificationPortMapping			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 5.73: DiagnosticDoIpEntityIdentificationPortMapping

[constr_10600] Existence of DiagnosticDoIpEntityIdentificationPortMapping.pPortPrototypeInExecutable

Imposition time: IT_DiagDes

⌈For each DiagnosticDoIpEntityIdentificationPortMapping, attribute pPortPrototypeInExecutable shall exist.⌋

[constr_10601] Existence of DiagnosticDoIpEntityIdentificationPortMapping.process

Imposition time: IT_DiagDes

⌈For each DiagnosticDoIpEntityIdentificationPortMapping, attribute process shall exist.⌋

5.2.5.5 Diagnostic DoIP Power Mode Mapping

[TPS_MANI_01464] Retrieving the status of the vehicle's power mode to the diagnostic server instance ⌈For the purpose of retrieving the DoIP power mode to the diagnostic server instance on the *AUTOSAR adaptive platform*, the relation between a ProcessDesign and one PPortPrototype is created by using the DiagnosticDoIpPowerModePortMapping that refers to

- a ProcessDesign in the role process, and
- a PPortPrototype typed by a DiagnosticDoIPPowerModeInterface 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.

There is no model element in the context of the Diagnostic Extract to which the DiagnosticDoIpPowerModePortMapping could reliably refer to.

This is the reason why the DiagnosticDoIpPowerModePortMapping is restricted to references to PPortPrototype and ProcessDesign.⌋

[constr_10602] Restriction for the applicability of DiagnosticDoIpPowerMode-PortMapping

Imposition time: IT_DiagDes

⌈If a PPortPrototype is referenced in the role pPortPrototypeInExecutable by a DiagnosticDoIpPowerModePortMapping, then the PPortPrototype shall be typed by a DiagnosticDoIPPowerModeInterface.⌋

[constr_10603] Restriction for the applicability of DiagnosticDoIPPowerMode-Interface

Imposition time: IT_DiagDes

⌈If a PPortPrototype is typed by a DiagnosticDoIPPowerModeInterface, then the reference in the role pPortPrototypeInExecutable to the PPortPrototype shall only originate from a DiagnosticDoIpPowerModePortMapping.⌋

For more information, see [SWS_DM_00814].

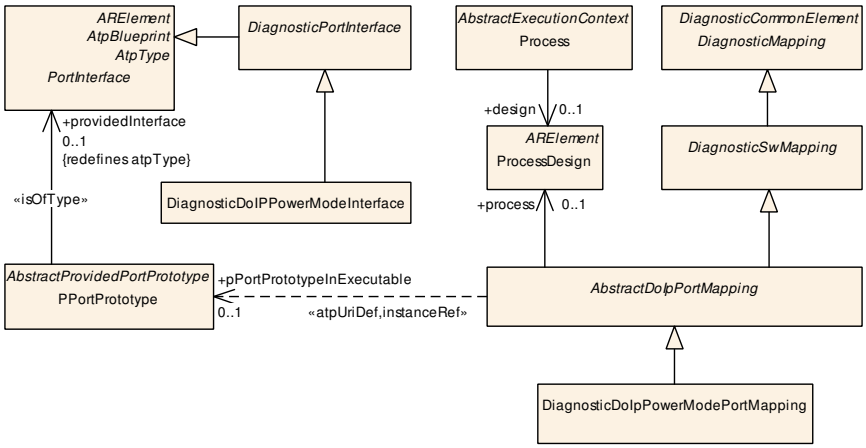


Figure 5.36: Modeling of DiagnosticDoIpPowerModePortMapping to indicate the existence of a handler for the DoIP power mode

Class	**DiagnosticDoIpPowerModePortMapping**			
Note	This mapping class identifies the PortPrototype in the application software that handles the DoIP power mode. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractDoIpPortMapping*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 5.74: DiagnosticDoIpPowerModePortMapping

[constr_10604] Existence of DiagnosticDoIpPowerModePortMapping.pPortPrototypeInExecutable

Imposition time: IT_DiagDes

⌈For each DiagnosticDoIpPowerModePortMapping, attribute pPortPrototypeInExecutable shall exist.⌋

[constr_10605] Existence of DiagnosticDoIpPowerModePortMapping.process

Imposition time: IT_DiagDes

⌈For each DiagnosticDoIpPowerModePortMapping, attribute process shall exist.⌋

5.2.5.6 Diagnostic DoIP Custom Payload Mapping

The DoIP standard [23] foresees the possibility to exchange custom payloads between a DoIP client and a DoIP server.

For this purpose, the DiagnosticDoIPCustomPayloadTxInterface and the DiagnosticDoIPCustomPayloadRxInterface are defined to support the implementation of functionalities in the context of the application software.

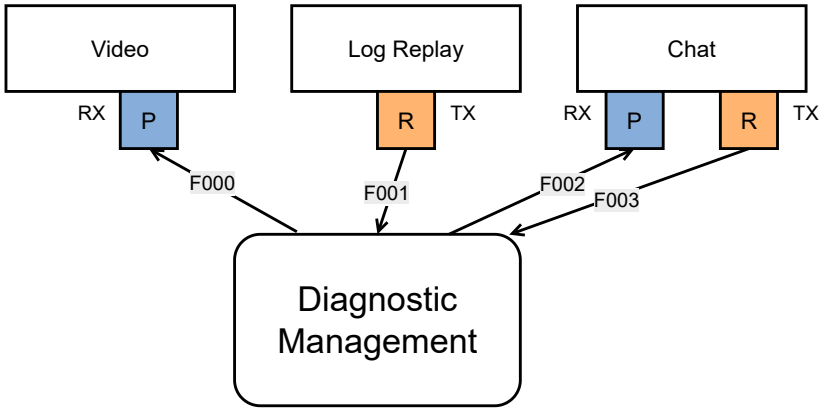


Figure 5.37: Use cases for the exchange of custom payloads between a DoIP client and a DoIP server

As already indicated by the existence of the DiagnosticDoIPCustomPayloadTxInterface and the DiagnosticDoIPCustomPayloadRxInterface, the application software on the *AUTOSAR adaptive platform* is able to send and receive custom payloads to/from a DoIP client. Examples for the use of custom payloads in both Tx and Rx direction are shown in Figure 5.37.

[constr_10780] Uniqueness of the value of DiagnosticDoIPCustomPayloadRxPortMapping.payloadTypeId

Imposition time: IT_DiagDes

⌈In the context of a DiagnosticContributionSet, no two DiagnosticDoIPCustomPayloadRxPortMapping shall have the same value of attribute payloadTypeId.⌋

Class	***DiagnosticDoIPCustomPayloadAbstractPortMapping*** (abstract)			
Note	This meta-class serves as an abstract base class for all mapping classes that deal with custom DoIP payloads. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticDoIPCustomPayloadRxPortMapping, DiagnosticDoIPCustomPayloadTxPortMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
payloadTypeId	PositiveInteger	0..1	attr	This attribute provides the numerical type Id of the payload reserved for manufacturer-specific use according to the ISO DoIP standard [23].
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. **Tags:** atp.Splitkey=process

Table 5.75: DiagnosticDoIPCustomPayloadAbstractPortMapping

5.2.5.6.1 Diagnostic DoIP Custom Payload Rx Mapping

[TPS_MANI_01520] Receiving custom payloads from a DoIP client ⌈For the purpose of receiving custom payloads from a DoIP client, the relation between a ProcessDesign and one PPortPrototype is created by using the DiagnosticDoIPCustomPayloadRxPortMapping that refers to

- a ProcessDesign in the role process, and
- a PPortPrototype typed by a DiagnosticDoIPCustomPayloadRxInterface 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.

There is no model element in the context of the Diagnostic Extract to which the DiagnosticDoIPCustomPayloadRxPortMapping could reliably refer to. This is the reason why the DiagnosticDoIPCustomPayloadRxPortMapping is restricted to references to PPortPrototype and ProcessDesign.⌋

[constr_10781] Restriction for the applicability of DiagnosticDoIPCustomPayloadRxPortMapping

Imposition time: IT_DiagDes

⌈If a PPortPrototype is referenced in the role pPortPrototypeInExecutable by a DiagnosticDoIPCustomPayloadRxPortMapping, then the PPortPrototype shall be typed by a DiagnosticDoIPCustomPayloadRxInterface.⌋

[constr_10782] Restriction for the applicability of DiagnosticDoIPCustomPayloadRxInterface

Imposition time: IT_DiagDes

⌈If a PPortPrototype is typed by a DiagnosticDoIPCustomPayloadRxInterface, then the reference in the role pPortPrototypeInExecutable to the PPortPrototype shall only originate from a DiagnosticDoIPCustomPayloadRxPortMapping.⌋

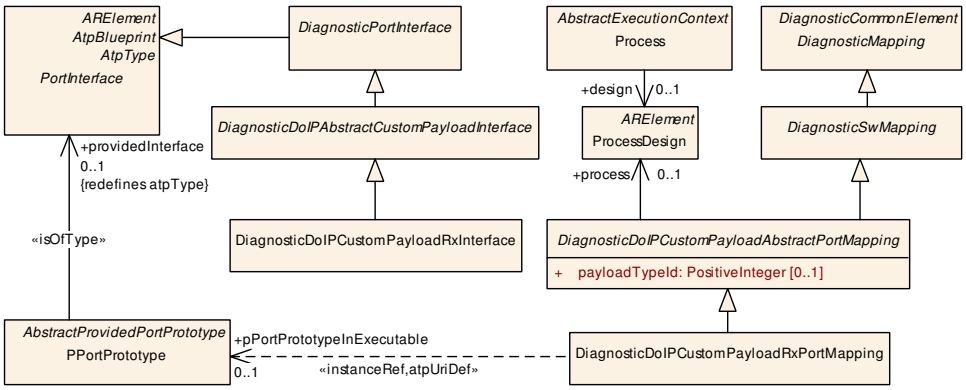


Figure 5.38: Modeling of DiagnosticDoIPCustomPayloadRxPortMapping to indicate the existence of application software that is able to receive custom payloads from a DoIP client

[constr_10783] Existence of DiagnosticDoIPCustomPayloadRxPortMapping.pPortPrototypeInExecutable

Imposition time: IT_DiagDes

⌈For each DiagnosticDoIPCustomPayloadRxPortMapping, attribute pPortPrototypeInExecutable shall exist.⌋

[constr_10784] Existence of DiagnosticDoIPCustomPayloadAbstractPortMapping.process

Imposition time: IT_DiagDes

⌈For each DiagnosticDoIPCustomPayloadRxPortMapping, attribute process shall exist.⌋

[constr_10785] Existence of DiagnosticDoIPCustomPayloadAbstractPortMapping.payloadTypeId

Imposition time: IT_DiagDes

⌈For each DiagnosticDoIPCustomPayloadRxPortMapping, attribute payloadTypeId shall exist.⌋

Class	**DiagnosticDoIPCustomPayloadRxPortMapping**			
Note	This mapping class identifies a PPortPrototype in the application software that sends out custom Do IP payload. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticDoIPCustomPayloadAbstractPortMapping*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
pPortPrototypeInExecutable	PPortPrototype	0..1	iref	This identifies the PPortPrototype for which the DiagnosticDoIPCustomPayloadAbstractPortMapping is applicable. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=pPortPrototypeInExecutable, pPortPrototypeInExecutable.contextRootSwComponentPrototype, pPortPrototypeInExecutable.contextComponentPrototype, pPortPrototypeInExecutable.targetPPortPrototype **InstanceRef implemented by:** PPortPrototypeInExecutableInstanceRef

Table 5.76: DiagnosticDoIPCustomPayloadRxPortMapping

5.2.5.6.2 Diagnostic DoIP Custom Payload Tx Mapping

[TPS_MANI_01521] Sending custom payloads to a DoIP client ⌈For the purpose of sending custom payloads to a DoIP client, the relation between a ProcessDesign and one PPortPrototype is created by using the DiagnosticDoIPCustomPayloadTxPortMapping that refers to

- a ProcessDesign in the role process, and
- a RPortPrototype typed by a DiagnosticDoIPCustomPayloadTxInterface 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.

There is no model element in the context of the Diagnostic Extract to which the DiagnosticDoIPCustomPayloadTxPortMapping could reliably refer to. This is the reason why the DiagnosticDoIPCustomPayloadTxPortMapping is restricted to references to RPortPrototype and ProcessDesign.⌋

[constr_10786] Restriction for the applicability of DiagnosticDoIPCustomPayloadTxPortMapping

Imposition time: IT_DiagDes

⌈If a RPortPrototype is referenced in the role rPortPrototypeInExecutable by a DiagnosticDoIPCustomPayloadTxPortMapping, then the RPortPrototype shall be typed by a DiagnosticDoIPCustomPayloadTxInterface.⌋

[constr_10787] Restriction for the applicability of DiagnosticDoIPCustomPayloadTxInterface

Imposition time: IT_DiagDes

⌈If a RPortPrototype is typed by a DiagnosticDoIPCustomPayloadTxInterface, then the reference in the role rPortPrototypeInExecutable to the RPortPrototype shall only originate from a DiagnosticDoIPCustomPayloadTxPortMapping.⌋

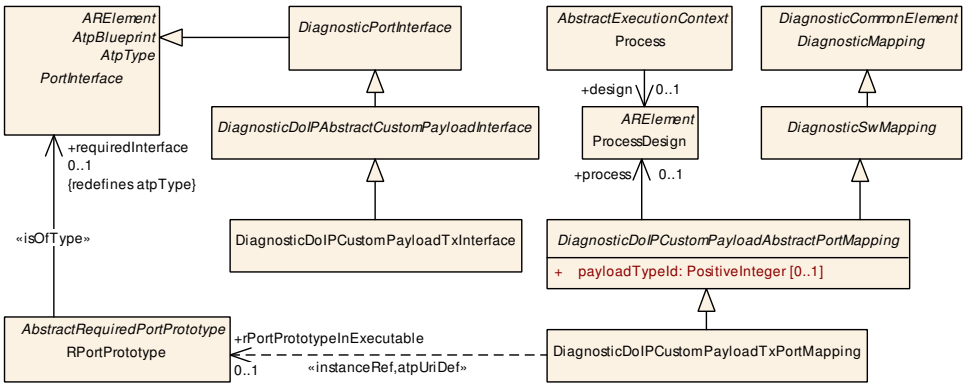


Figure 5.39: Modeling of DiagnosticDoIPCustomPayloadTxPortMapping to indicate the existence of application software that is able to send custom payloads to a DoIP client

[constr_10788] Existence of reference DiagnosticDoIPCustomPayloadTxPortMapping.rPortPrototypeInExecutable

Imposition time: IT_DiagDes

⌈For each DiagnosticDoIPCustomPayloadTxPortMapping, attribute rPortPrototypeInExecutable shall exist.⌋

[constr_10789] Existence of reference DiagnosticDoIPCustomPayloadTxPortMapping.process

Imposition time: IT_DiagDes

⌈For each DiagnosticDoIPCustomPayloadTxPortMapping, attribute process shall exist.⌋

[constr_10790] Existence of reference DiagnosticDoIPCustomPayloadTx-PortMapping.payloadTypeId

Imposition time: IT_DiagDes

⌈For each DiagnosticDoIPCustomPayloadTxPortMapping, attribute payload-TypeId shall exist.⌋

Class	**DiagnosticDoIPCustomPayloadTxPortMapping**			
Note	This mapping class identifies an RPortPrototype in the application software that sends out custom Do IP payload. **Tags:** atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticDoIPCustom PayloadAbstractPortMapping*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This identifies the RPortPrototype for which the Abstract DoIpCustomPayloadTxPortMapping is applicable. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=rPortPrototypeInExecutable, rPort PrototypeInExecutable.contextRootSwComponent Prototype, rPortPrototypeInExecutable.context ComponentPrototype, rPortPrototypeInExecutable.target RPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 5.77: DiagnosticDoIPCustomPayloadTxPortMapping

5.3 Diagnostic Clear Condition

On the *AUTOSAR adaptive platform*, a new model element similar to DiagnosticEnableCondition is introduced: DiagnosticClearCondition.

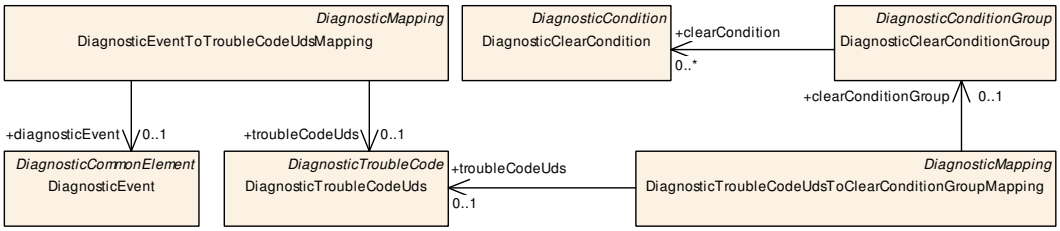


Figure 5.40: Modeling of the diagnostic clear condition

In contrast to DiagnosticEnableCondition, DiagnosticClearCondition is not mapped to a DiagnosticEvent but (via the aggregation by DiagnosticClear-ConditionGroup) to a DiagnosticTroubleCodeUds.

For this purpose, meta-class DiagnosticTroubleCodeUdsToClearCondition-GroupMapping has been defined.

Class	DiagnosticClearCondition			
Note	This meta-class describes a clear condition for diagnostic purposes. **Tags:** atp.recommendedPackage=DiagnosticConditions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticCondition*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 5.78: DiagnosticClearCondition

Class	DiagnosticClearConditionGroup			
Note	Clear condition group which includes one or several clear conditions. **Tags:** atp.recommendedPackage=DiagnosticClearConditionGroups This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticConditionGroup*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
clearCondition	DiagnosticClear Condition	*	ref	This aggregation represents the collection of Diagnostic ClearConditions that belong to the DiagnosticClear ConditionGroup. **Tags:** atp.Splitkey=clearCondition

Table 5.79: DiagnosticClearConditionGroup

Class	DiagnosticTroubleCodeUdsToClearConditionGroupMapping			
Note	This meta-class provides the ability to map a DiagnosticClearConditionGroup to a collection of Diagnostic TroubleCodeUds. **Tags:** atp.recommendedPackage=DiagnosticMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
clearCondition Group	DiagnosticClear ConditionGroup	0..1	ref	Thi reference identifies the applicable DiagnosticClear ConditionGroup. **Tags:** atp.Splitkey=clearConditionGroup
troubleCodeUds	DiagnosticTroubleCode Uds	0..1	ref	This reference identifies the DiagnosticTroubleCodeUds that are relevant for the mapping. **Tags:** atp.Splitkey=troubleCodeUds

Table 5.80: DiagnosticTroubleCodeUdsToClearConditionGroupMapping

[constr_1658] Number of `DiagnosticTroubleCodeUdsToClearCondition-GroupMapping` elements per `DiagnosticTroubleCodeUds`

Imposition time: `IT_DiagDes`

⌈The mapping element `DiagnosticTroubleCodeUdsToClearConditionGroup-Mapping` 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.⌋

5.4 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.

Class	**DiagnosticSecurityAccessClass**			
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*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
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. This Attribute is only used by the AUTOSAR Adaptive Platform.

Table 5.81: DiagnosticSecurityAccessClass

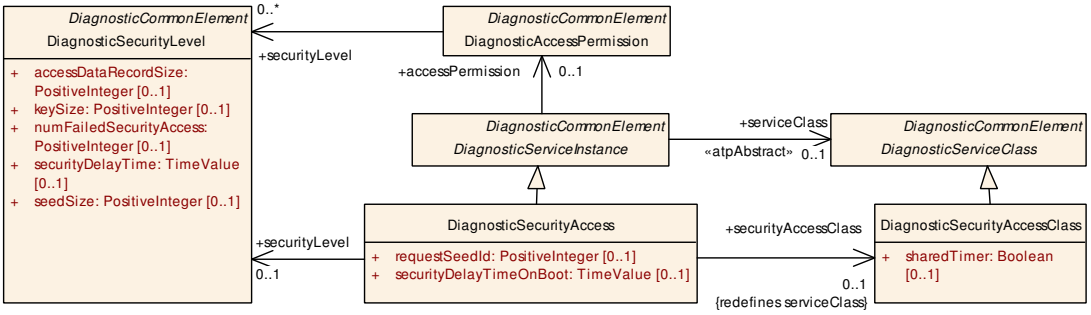


Figure 5.41: Refined modeling of the diagnostic security access

5.5 Diagnostic Provided Data Mapping

[TPS_MANI_01230] Semantics of DiagnosticProvidedDataMapping ⌈The purpose of the DiagnosticProvidedDataMapping is to provide internal values of the diagnostic stack to specific DiagnosticDataElements.

This is why 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.

The specific nature of this mapping is that the second element (the DiagnosticProvidedDataMapping.dataProvider) that is supposed to take place in the mapping (by being an internal value of the diagnostic stack) cannot precisely be modeled as a meta-class at all.

Therefore, there is no better way than to model the DiagnosticProvidedDataMapping.dataProvider by a NameToken.⌋

The DiagnosticProvidedDataMapping is not to be confused with the DiagnosticEdrDataProviderMapping. The former delivers values to DiagnosticDataElements for access by the UDS service “ReadDataByIdentifier”, while the latter delivers data to the content of an extended data record.

The list of standardized values of attribute DiagnosticProvidedDataMapping.dataProvider is defined in the [24, SWS Diagnostics].

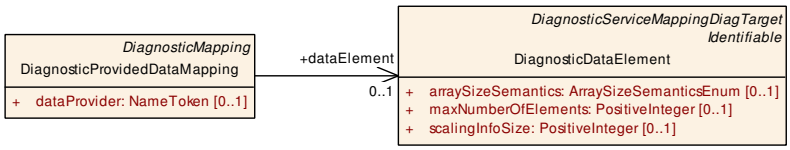


Figure 5.42: Modeling of DiagnosticProvidedDataMapping

Class	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.recommendedPackage=DiagnosticMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dataElement	DiagnosticDataElement	0..1	ref	This represents the DiagnosticDataElement to which the internal data is mapped, further qualified by the DiagnosticProvidedDataMapping.dataProvider. **Tags:** atp.Splitkey=dataElement
dataProvider	NameToken	0..1	attr	This represents the ability to further specify the data provider.

Table 5.82: DiagnosticProvidedDataMapping

[constr_10166] Multiplicity of attribute DiagnosticProvidedDataMapping.dataProvider

Imposition time: IT_DiagDes

⌈For each DiagnosticProvidedDataMapping, the attribute dataProvider shall exist.⌋

5.6 Usage of Diagnostic Environmental Data Element Condition

The definition of environmental conditions is explained in the document [3, TPS Diagnostic Extract Template].

Specifically for the *AUTOSAR adaptive platform*, it is possible to obtain data from application software (fetched from a `PPortPrototype` typed by a `DiagnosticDataElementInterface`) as a reference value for the computation of the environmental condition.

The reference `DiagnosticEnvDataElementCondition.pPortPrototype` is decorated with stereotype ≪`atpUriDef`≫ because the model of the application software does not necessarily have to exist when the diagnostic design is created.

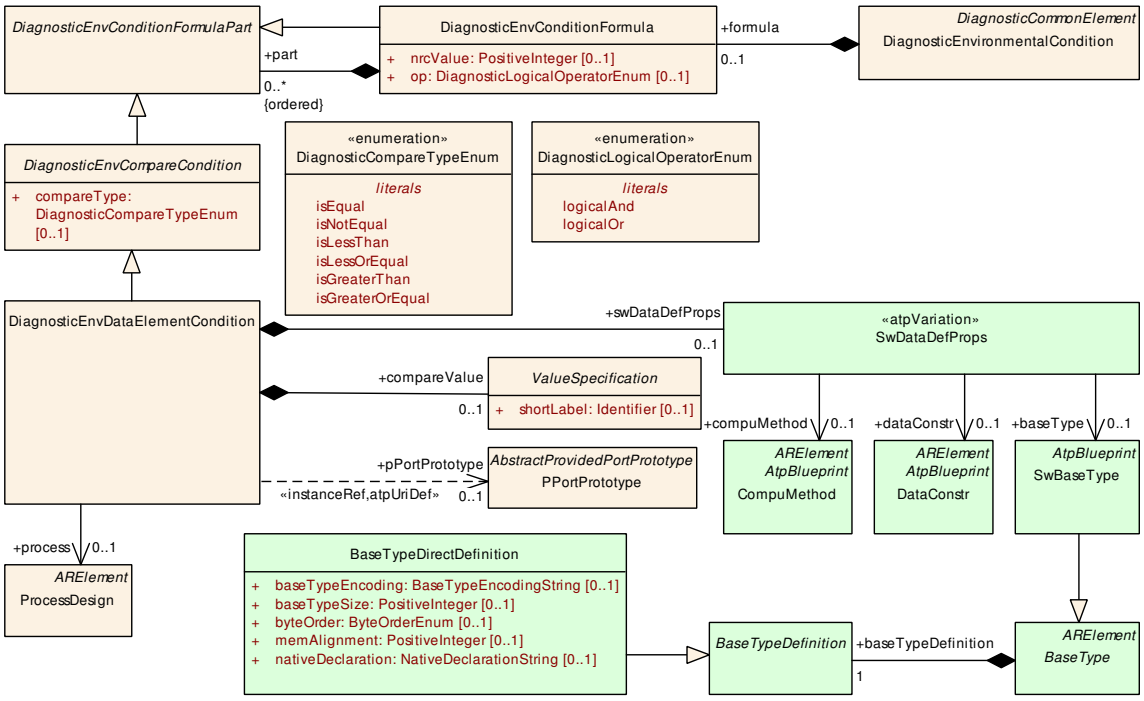


Figure 5.43: Leverage of `DiagnosticEnvironmentalCondition` on the *AUTOSAR adaptive platform*

[TPS_MANI_01372] Properties of data obtained from the application software ⌈The properties of the data obtained from the application software can be configured in the context of `DiagnosticEnvDataElementCondition`, namely by means of the aggregation of `SwDataDefProps` in the role `swDataDefProps`.⌋

[constr_10114] Existence of attributes of `DiagnosticEnvDataElementCondition` if the reference in the role `pPortPrototype` exists

Imposition time: `IT_DiagDes`

⌈If the reference in the role `DiagnosticEnvDataElementCondition.pPortPrototype` exists, then the aggregations in the roles `compareValue`, `process`, and `swDataDefProps` shall exist.⌋

Class	**DiagnosticEnvironmentalCondition**			
Note	The meta-class DiagnosticEnvironmentalCondition formalizes the idea of a condition which is evaluated during runtime of the ECU by looking at "environmental" states (e.g. one such condition is that the vehicle is not driving, i.e. vehicle speed == 0). **Tags:** atp.recommendedPackage=DiagnosticEnvironmentalConditions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
formula	DiagnosticEnvCondition Formula	0..1	aggr	This attribute represents the formula part of the DiagnosticEnvironmentalCondition. **Tags:** atp.Splitkey=formula
modeElement	DiagnosticEnvMode Element	*	aggr	This aggregation contains a representation of Mode Declarations in the context of a DiagnosticEnvironmental Condition. **Tags:** atp.Splitkey=modeElement, modeElement.short Name

Table 5.83: DiagnosticEnvironmentalCondition

Class	**DiagnosticEnvDataElementCondition**			
Note	This meta-class represents the ability to formulate a diagnostic environment condition based on the value of a data element owned by the application software.			
Base	*ARObject*, *DiagnosticEnvCompareCondition*, *DiagnosticEnvConditionFormulaPart*			
Aggregated by	DiagnosticEnvConditionFormula.part			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compareValue	ValueSpecification	0..1	aggr	This aggregation represents the definition of the compare value against which the value taken from the application software shall be compared.
dataPrototype	DataPrototype	0..1	iref	This instanceRef represent the ability to access a data element owned by the application software on the AUTOSAR classic platform. **InstanceRef implemented by:** DataPrototypeInSystem InstanceRef
pPortPrototype	PPortPrototype	0..1	iref	This instanceRef identifies the PortPrototype from which the relevant information for the environment condition can be obtained. This InstanceRef is only relevant for the adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef This Attribute is only used by the AUTOSAR Adaptive Platform.
process	ProcessDesign	0..1	ref	This reference identifies the applicable ProcessDesign. This Attribute is only used by the AUTOSAR Adaptive Platform.
swDataDef Props	SwDataDefProps	0..1	aggr	Via this aggregation it is possible to describe the properties of the data that is obtained from the application for the environmental condition.

Table 5.84: DiagnosticEnvDataElementCondition

Class	***DiagnosticEnvCompareCondition*** (abstract)			
Note	DiagnosticCompareConditions are atomic conditions. They are based on the idea of a comparison at runtime of some variable data with something constant. The type of the comparison (==, !=, <, <=, ...) is specified in DiagnosticCompareCondition.compareType.			
Base	*ARObject*, *DiagnosticEnvConditionFormulaPart*			
Subclasses	DiagnosticEnvDataCondition, DiagnosticEnvDataElementCondition, DiagnosticEnvModeCondition, DiagnosticEnvSovdDataCondition			
Aggregated by	DiagnosticEnvConditionFormula.part			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compareType	DiagnosticCompare TypeEnum	0..1	attr	This attributes represents the concrete type of the comparison.

Table 5.85: DiagnosticEnvCompareCondition

Class	**DiagnosticEnvConditionFormula**			
Note	A DiagnosticEnvConditionFormula embodies the computation instruction that is to be evaluated at runtime to determine if the DiagnosticEnvironmentalCondition is currently present (i.e. the formula is evaluated to true) or not (otherwise). The formula itself consists of parts which are combined by the logical operations specified by DiagnosticEnvConditionFormula.op. If a diagnostic functionality cannot be executed because an environmental condition fails then the diagnostic stack shall send a negative response code (NRC) back to the client. The value of the NRC is directly related to the specific formula and is therefore formalized in the attribute DiagnosticEnvCondition Formula.nrcValue.			
Base	*ARObject*, *DiagnosticEnvConditionFormulaPart*			
Aggregated by	DiagnosticEnvConditionFormula.part, DiagnosticEnvironmentalCondition.formula			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
nrcValue	PositiveInteger	0..1	attr	This attribute represents the concrete NRC value that shall be returned if the condition fails.
op	DiagnosticLogical OperatorEnum	0..1	attr	This attribute represents the concrete operator (supported operators: and, or) of the condition formula.
part (ordered)	DiagnosticEnvCondition FormulaPart	*	aggr	This aggregation represents the collection of formula parts that can be combined by logical operators.

Table 5.86: DiagnosticEnvConditionFormula

Class	***DiagnosticEnvConditionFormulaPart*** (abstract)			
Note	A DiagnosticEnvConditionFormulaPart can either be a atomic condition, e.g. a DiagnosticEnvCompare Condition, or a DiagnosticEnvConditionFormula, again, which allows arbitrary nesting.			
Base	*ARObject*			
Subclasses	*DiagnosticEnvCompareCondition*, DiagnosticEnvConditionFormula			
Aggregated by	DiagnosticEnvConditionFormula.part			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 5.87: DiagnosticEnvConditionFormulaPart

Class	***DiagnosticEnvModeElement*** (abstract)			
Note	All ModeDeclarations that are referenced in a DiagnosticEnvModeCondition shall be defined as a DiagnosticEnvModeElement of this DiagnosticEnvironmentalCondition. This concept keeps the ARXML clean: It avoids that the DiagnosticEnvConditionFormula is cluttered by lengthy InstanceRef definitions. Furthermore, it allows that an InstanceRef only needs to be defined once and can be used multiple times in the different DiagnosticEnvModeConditions.			
Base	*ARObject*, *Referrable*			
Subclasses	DiagnosticEnvBswModeElement, DiagnosticEnvSwcModeElement			
Aggregated by	DiagnosticEnvironmentalCondition.modeElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 5.88: DiagnosticEnvModeElement

Enumeration	**DiagnosticCompareTypeEnum**
Note	Enumeration for the type of a comparison of values usually expressed by the following operators: ==, !=, <, <=, >, >=
Aggregated by	*DiagnosticEnvCompareCondition*.compareType
Literal	***Description***
isEqual	equal **Tags:** atp.EnumerationLiteralIndex=0
isGreaterOrEqual	greater than or equal **Tags:** atp.EnumerationLiteralIndex=5
isGreaterThan	greater than **Tags:** atp.EnumerationLiteralIndex=4
isLessOrEqual	less than or equal **Tags:** atp.EnumerationLiteralIndex=3
isLessThan	less than **Tags:** atp.EnumerationLiteralIndex=2
isNotEqual	not equal **Tags:** atp.EnumerationLiteralIndex=1

Table 5.89: DiagnosticCompareTypeEnum

Enumeration	**DiagnosticLogicalOperatorEnum**
Note	Logical AND and OR operation (&&, ||)
Aggregated by	DiagnosticEnvConditionFormula.op
Literal	***Description***
logicalAnd	Logical AND **Tags:** atp.EnumerationLiteralIndex=0
logicalOr	Logical OR **Tags:** atp.EnumerationLiteralIndex=1

Table 5.90: DiagnosticLogicalOperatorEnum

5.7 SOVD Configuration Elements

This chapter documents the meaning of a number of meta-classes that are specific to the usage of Service-Oriented Vehicle Diagnostics (SOVD).

5.7.1 SOVD Service Instance

5.7.1.1 Overview

[TPS_MANI_01431] Definition of SOVD service instance ⌈Similar to UDS, SOVD supports the concept of a *service instance*. The SOVD service instance is formalized by means of abstract meta-class DiagnosticSovdServiceInstance.⌋

Concrete SOVD services are modeled as concrete sub-classes of DiagnosticSovd-ServiceInstance. The “big picture” of the modeling of SOVD service instances is sketched in Figure 5.44.

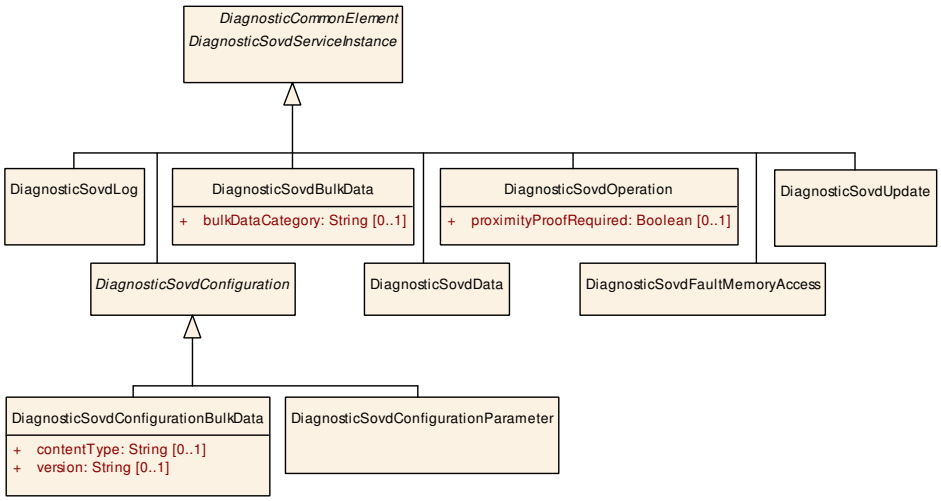


Figure 5.44: Modeling of DiagnosticSovdServiceInstance

Class	**DiagnosticSovdBulkData**			
Note	This meta-class represents a "Bulk Data" SOVD service instance. **Tags:** atp.recommendedPackage=DiagnosticSovdServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticSovdService Instance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bulkData Category	String	0..1	attr	This attribute provides a categorization of the bulk data.

Table 5.91: DiagnosticSovdBulkData

Class	**DiagnosticSovdConfiguration** (abstract)			
Note	This abstract meta-class represents a "configuration" SOVD service instance. The concrete nature of the service instance is defined by sub-classes. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticSovdService Instance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticSovdConfigurationBulkData, DiagnosticSovdConfigurationParameter			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 5.92: DiagnosticSovdConfiguration

Class	**DiagnosticSovdConfigurationParameter**			
Note	This meta-class represents a "Configuration Parameter" SOVD service instance. **Tags:** atp.recommendedPackage=DiagnosticSovdServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticSovdConfiguration*, *DiagnosticSovdServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 5.93: DiagnosticSovdConfigurationParameter

Class	**DiagnosticSovdConfigurationBulkData**			
Note	This meta-class represents a "Configuration Bulk Data" SOVD service instance. **Tags:** atp.recommendedPackage=DiagnosticSovdServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticSovdConfiguration*, *DiagnosticSovdServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
contentType	String	0..1	attr	This attribute represents the setting of the content (i.e. MIME) type of the bulk data. Please note that the number of supported content types is not restricted by the ASAM SOVD standard and therefore it is unfortunately not possible to constrain the possible settings of this attribute.
version	String	0..1	attr	This attribute represents a version information. Please note that the structure of the version information is not restricted by the ASAM SOVD standard and therefore the structure of the content of this attribute cannot be constrained further in the scope of AUTOSAR.

Table 5.94: DiagnosticSovdConfigurationBulkData

5.7.1.2 SOVD Log

The `SOVD` log service is based on the existence of just a single instance of `DiagnosticSovdLog` for which no matching subclass of `DiagnosticSovdPort-Interface` is defined.

This means that the application software does not have access to the log service[2] and therefore access to SOVD logging is restricted to the case where diagnostic management (DM) uses the logging service internally.

The restriction to just a single instance of DiagnosticSovdLog is formalized by means of [constr_10479].

Class	***DiagnosticSovdServiceInstance*** (abstract)			
Note	This abstract class serves as the base class for the configuration of SOVD service instances. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticSovdBulkData, *DiagnosticSovdConfiguration*, DiagnosticSovdData, DiagnosticSovdFaultMemoryAccess, DiagnosticSovdLog, DiagnosticSovdOperation, DiagnosticSovdUpdate			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
method	DiagnosticSovdMethod	0..1	ref	This reference identifies the applicable SOVD method. **Tags:** atp.Splitkey=method

Table 5.95: DiagnosticSovdServiceInstance

[constr_10477] Existence of DiagnosticSovdLog

Imposition time: IT_DiagDes

⌈Each DiagnosticContributionSet shall at most reference a single DiagnosticSovdLog in the role element.⌋

Class	**DiagnosticSovdLog**			
Note	This meta-class represents a "Log" SOVD service instance. **Tags:** atp.recommendedPackage=DiagnosticSovdServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticSovdServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 5.96: DiagnosticSovdLog

5.7.1.3 SOVD Update

A similar restriction is in place for the existence of DiagnosticSovdUpdate, see [constr_10480].

[2]This also explains why there is no mapping class defined for the SOVD logging.

[constr_10478] Existence of DiagnosticSovdUpdate

Imposition time: IT_DiagDes

⌈Each DiagnosticContributionSet shall at most reference a single DiagnosticSovdUpdate in the role element.⌋

The meta-class DiagnosticSovdFaultMemoryAccess can be used to access the fault memory. Access to the primary memory is protected by the access permission associated with the DiagnosticSovdMethod.get and delete DiagnosticSovdMethodPrimitives.

In addition, access to user-defined fault memories may require diagnostic authentication.

Class	**DiagnosticSovdUpdate**			
Note	This meta-class represents a "Update" SOVD service instance. **Tags:** atp.recommendedPackage=DiagnosticSovdServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticSovdServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 5.97: DiagnosticSovdUpdate

5.7.1.4 SOVD Fault Memory Access

The meta-class DiagnosticSovdFaultMemoryAccess can be used to access the fault memory. Access to the primary memory is protected by the access permission associated with the DiagnosticSovdMethod.get and delete DiagnosticSovdMethodPrimitives.

[constr_10791] Existence of DiagnosticSovdMethodPrimitive in the context of a DiagnosticSovdFaultMemoryAccess

Imposition time: IT_DiagDes

⌈If a DiagnosticSovdFaultMemoryAccess is referencing a DiagnosticSovdMethod in the role method, then the following restriction for the existence of the attributes of the DiagnosticSovdMethod applies:

put not allowed

post not allowed

get allowed

delete allowed

⌋

From the tester's perspective, there are two access scenarios using the `get` or `delete` primitive on the fault memory:

- Access all entries of the fault memory
- Access a single entry of the fault memory.

The distinction between the two scenarios is encoded in the `category` of the `DiagnosticSovdMethodPrimitive`, see [TPS_MANI_01522].

[TPS_MANI_01522] Access scenarios for the fault memory ⌈The distinction between the two scenarios is encoded in the `category` of the `DiagnosticSovdMethodPrimitive`.

For `DiagnosticSovdMethodPrimitive`s that are aggregated (in either the role `get` or `delete`) by a `DiagnosticSovdMethod` that is referenced from a `DiagnosticSovdFaultMemoryAccess`, the following values of the attribute `category` are allowed:

`FAULT_MEMORY_ACCESS_ALL` - The `DiagnosticSovdMethodPrimitive` is used to access all entries of the fault memory.

`FAULT_MEMORY_ACCESS_SINGLE` - The `DiagnosticSovdMethodPrimitive` is used to access a single entry of the fault memory.

⌋

This means that, in total, four combinations of `category` and `DiagnosticSovdMethodPrimitive` are supported:

Delete all entries in the fault memory - set `category` = `FAULT_MEMORY_ACCESS_ALL` and use `DiagnosticSovdMethod.delete`

Delete a single entry in the fault memory - set `category` = `FAULT_MEMORY_ACCESS_SINGLE` and use `DiagnosticSovdMethod.delete`

Get all entries in the fault memory - set `category` = `FAULT_MEMORY_ACCESS_ALL` and use `DiagnosticSovdMethod.get`

Get a single entry in the fault memory - set `category` = `FAULT_MEMORY_ACCESS_SINGLE` and use `DiagnosticSovdMethod.get`

In addition, access to user-defined fault memories may require diagnostic authentication.

Class	**DiagnosticSovdFaultMemoryAccess**
Note	This meta-class represents the ability to access (read/get and delete) the fault memory. **Tags:** atp.recommendedPackage=DiagnosticSovdServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticSovdServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

Class	DiagnosticSovdFaultMemoryAccess			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 5.98: DiagnosticSovdFaultMemoryAccess

5.7.1.5 Diagnostic SOVD Data

The usage of the SOVD Data service is connected with the **mandatory** definition of a so-called *SOVD data category* as well as the definition of an optional collection of so-called *SOVD groups*.

The latter does not represent a group in the literal sense, i.e. that a grouping of model elements is intended. The semantics is more like a secondary filter criterion, where the primary filter criterion is represented by the *SOVD data category*, which is required to exist for each DiagnosticSovdData, see [constr_10706].

For illustration, the tester can query data from the diagnostic server using the *SOVD data category* as a filter criterion. The query can be further refined by specifying a collection of *SOVD groups*.

The relation between the *SOVD service instance*, the *SOVD data category* and the *SOVD groups* is a formal one, therefore the relation is modeled in the form of meta-classes

- DiagnosticSovdData
- DiagnosticSovdDataCategory
- DiagnosticSovdGroup

Constraints apply, see below.

The semantics of the DiagnosticSovdDataCategory is standardized by AUTOSAR by defining standardized values for attribute DiagnosticSovdDataCategory.category.

In other words, the shortName is not relevant. This aspect is covered by the spec item [TPS_MANI_01492].

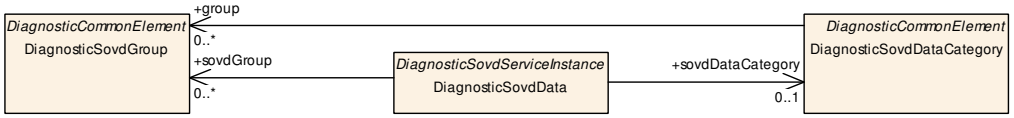


Figure 5.45: Modeling of DiagnosticSovdData

[TPS_MANI_01492] Standardized values for attribute DiagnosticSovdDataCategory.category ⌈The following values of attribute DiagnosticSovdDataCategory.category are standardized by AUTOSAR:

IDENT_DATA - Read access to fixed parameters that identify entities within an *extended vehicle* (ExVe), e.g., part number or VIN.

CURRENT_DATA - Read access to dynamically changing values, e.g., battery voltage.

STORED_DATA - Read and write access to parameters in the ExVe.

SYS_INFO - Read access to dynamically changing system resources, e.g., CPU load.

It is possible to define further values for attribute DiagnosticSovdDataCategory.category, provided that a custom prefix is used to avoid potential name clashes with further extensions of the AUTOSAR standard.⌋

The possible values of the *SOVD groups* are not standardized and therefore the intended semantics is carried by the shortName. There are, however, restrictions in place.

In particular, the DiagnosticSovdGroups referenced from the DiagnosticSovdData shall be a subset of the DiagnosticSovdGroups referenced from the DiagnosticSovdDataCategory that is referenced by the enclosing DiagnosticSovdData. This restriction is formalized in [constr_10705].

[constr_10705] Restriction regarding references in the roles DiagnosticSovdDataCategory.group and DiagnosticSovdData.sovdGroup

Imposition time: IT_DiagDes

⌈In the scope of any given DiagnosticSovdData, every DiagnosticSovdGroup that is referenced in the role DiagnosticSovdData.sovdGroup shall also be referenced in the role DiagnosticSovdDataCategory.group from the DiagnosticSovdDataCategory that is referenced in the role sovdDataCategory from the enclosing DiagnosticSovdData.⌋

Class	**DiagnosticSovdData**			
Note	This meta-class represents a "Data" SOVD service instance. **Tags:** atp.recommendedPackage=DiagnosticSovdServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticSovdService Instance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sovdData Category	DiagnosticSovdData Category	0..1	ref	This reference identifies the SOVD category that applies to the enclosing SOVD data. **Tags:** atp.Splitkey=sovdDataCategory atp.Status=candidate

▽

△

Class	DiagnosticSovdData			
sovdGroup	DiagnosticSovdGroup	*	ref	This reference identifies the collection of SOVD groups in which the enclosing SOVD data is a "member". **Tags:** atp.Splitkey=sovdGroup atp.Status=candidate
target	DiagnosticSovdContent	0..1	ref	Thsi reference identifies the content that the Diagnostic SovdData service instance is responsible for. **Tags:** atp.Splitkey=target

Table 5.99: DiagnosticSovdData

[constr_10706] Existence of reference in the role DiagnosticSovdData.sovdDataCategory

Imposition time: IT_DiagDes

⌈For each DiagnosticSovdData, the reference in the role sovdDataCategory shall exist.⌋

Class	DiagnosticSovdDataCategory			
Note	This meta-class represents the primary filter criterion for the access to SOVD data. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticDataCategorys This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
group	DiagnosticSovdGroup	*	ref	This reference identifies the SOVD groups in which the SOVD category is a "member". **Tags:** atp.Splitkey=group atp.Status=candidate

Table 5.100: DiagnosticSovdDataCategory

Class	DiagnosticSovdGroup			
Note	This meta-class represents the secondary filter criterion for the access to SOVD data **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticSovdGroups This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 5.101: DiagnosticSovdGroup

5.7.1.6 SOVD Operation

An SOVD operation (formalized by meta-class DiagnosticSovdOperation) can be used to execute arbitrary diagnostic-related code in the context of the application software.

The modeling of the DiagnosticSovdOperation and related meta-classes is depicted in Figure 5.46.

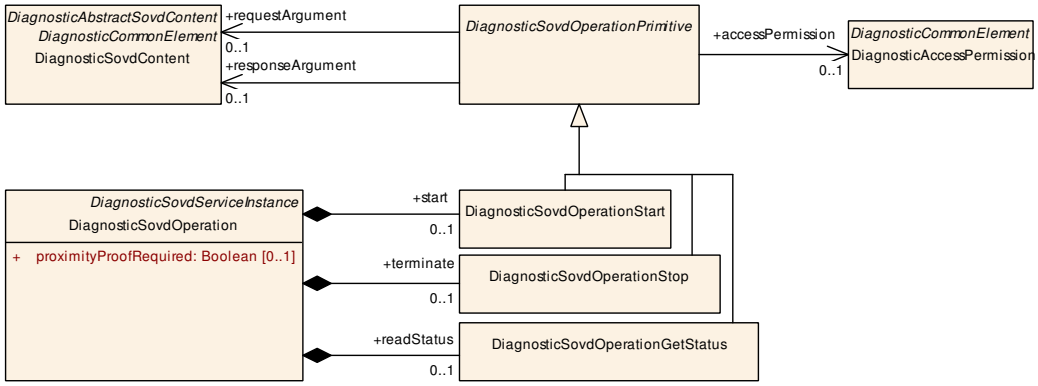


Figure 5.46: Modeling of DiagnosticSovdOperation and DiagnosticSovdOperationPrimitive

The details of other DiagnosticSovdServiceInstance are typically modeled by means of DiagnosticSovdMethod and aggregated classes.

This pattern does not apply to DiagnosticSovdOperation because the implementation of the latter (in terms of what otherwise would be clarified by DiagnosticSovdMethodPrimitives) is fixed and cannot be modified. As a consequence of this observation, [constr_10715] exists.

[constr_10715] DiagnosticSovdOperation shall not refer to DiagnosticSovdMethod

Imposition time: IT_DiagDes

⌈Any DiagnosticSovdOperation shall not refer to DiagnosticSovdMethod.⌋

Because the usage of DiagnosticSovdMethod (see Chapter 5.7.2) is not applicable for the definition of an DiagnosticSovdOperation, the specification of DiagnosticAccessPermission for the execution of DiagnosticSovdOperation is connected to the definition of a DiagnosticSovdOperationPrimitive.

This means that different DiagnosticSovdOperationPrimitives may be associated with different DiagnosticAccessPermission. Specifically, it is very typical that a specific DiagnosticAccessPermission is required to terminate a running DiagnosticSovdOperation.

The relation between DiagnosticSovdOperationPrimitive and DiagnosticAccessPermission (as defined by [TPS_MANI_01502]) is very similar to the configuration of UDS diagnostic routines, as explained in [3, TPS Diagnostic Extract Template] Chapter 4.3.8.9.

[TPS_MANI_01502] Definition of a reference to DiagnosticAccessPermission for DiagnosticSovdOperation ⌈The DiagnosticSovdOperationPrimitives are also used to define a reference to DiagnosticAccessPermissions.⌋

Class	**DiagnosticSovdOperation**			
Note	This meta-class represents a "Operation" SOVD service instance. **Tags:** atp.recommendedPackage=DiagnosticSovdServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticSovdService Instance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
proximityProof Required	Boolean	0..1	attr	If true, the execution of the operation requires proof of co-location, and the offline capability description includes the attribute x-sovd-proximity-proof-required in the Path Item object. An SOVD client can use this information to filter operations in advance if a co-location proof cannot be provided. **Tags:** atp.Status=candidate
readStatus	DiagnosticSovd OperationGetStatus	0..1	aggr	This aggregation identifies the "read status" operation primitive. **Tags:** atp.Splitkey=readStatus
start	DiagnosticSovd OperationStart	0..1	aggr	This aggregation identifies the "start" operation primitive. **Tags:** atp.Splitkey=start
terminate	DiagnosticSovd OperationStop	0..1	aggr	This aggregation identifies the "terminate" operation primitive. **Tags:** atp.Splitkey=terminate

Table 5.102: DiagnosticSovdOperation

[constr_10716] Existence of DiagnosticSovdOperation.start

Imposition time: IT_DiagDes

⌈The aggregation in the role DiagnosticSovdOperation.start shall always exist.⌋

[constr_10717] Existence of DiagnosticSovdOperation.proximityProofRequired

Imposition time: IT_DiagDes

⌈The aggregation in the role DiagnosticSovdOperation.proximityProofRequired shall always exist.⌋

[TPS_MANI_01503] Semantics of meta-class DiagnosticSovdOperationPrimitive ⌈Abstract meta-class DiagnosticSovdOperationPrimitive has been defined to provide the references in the roles

- requestArgument
- responseArgument

to sub-classes of DiagnosticSovdOperationPrimitive.⌋

Class	**DiagnosticSovdOperationPrimitive** (abstract)			
Note	This meta-class represents a primitive of an SOVD operation. The class is modeled as an abstract class to be able to contribute its properties to the concrete subclasses. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Subclasses	DiagnosticSovdOperationGetStatus, DiagnosticSovdOperationStart, DiagnosticSovdOperationStop			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
access Permission	DiagnosticAccess Permission	0..1	ref	This reference identifies the applicable access permission. **Tags:** atp.Splitkey=accessPermission xml.sequenceOffset=100
request Argument	DiagnosticSovdContent	0..1	ref	Ths reference identifies the arguments applicable for the operation request. **Tags:** atp.Splitkey=requestArgument xml.sequenceOffset=50
response Argument	DiagnosticSovdContent	0..1	ref	Ths reference identifies the arguments applicable for the operation response. **Tags:** atp.Splitkey=responseArgument xml.sequenceOffset=60

Table 5.103: DiagnosticSovdOperationPrimitive

Class	**DiagnosticSovdOperationStart**			
Note	This class represents the "start" primitive of an SOVD operation. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *DiagnosticSovdOperationPrimitive*			
Aggregated by	DiagnosticSovdOperation.start			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 5.104: DiagnosticSovdOperationStart

Class	**DiagnosticSovdOperationStop**
Note	This class represents the "stop" primitive of an SOVD operation. This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARObject*, *DiagnosticSovdOperationPrimitive*
Aggregated by	DiagnosticSovdOperation.terminate

▽

△

Class	**DiagnosticSovdOperationStop**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 5.105: DiagnosticSovdOperationStop

Class	**DiagnosticSovdOperationGetStatus**			
Note	This class represents the "get status" primitive of an SOVD operation. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *DiagnosticSovdOperationPrimitive*			
Aggregated by	DiagnosticSovdOperation.readStatus			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 5.106: DiagnosticSovdOperationGetStatus

5.7.2 SOVD Method

As sketched in Figure 5.47, the modeling of the behavior of a `DiagnosticSovdServiceInstance` by means of meta-class `DiagnosticSovdMethod` follows a pattern similar to the modeling of a diagnostic routine [3, TPS Diagnostic Extract Template] in that primitive actions are defined that represent atomic activities in the execution of the `SOVD` service.

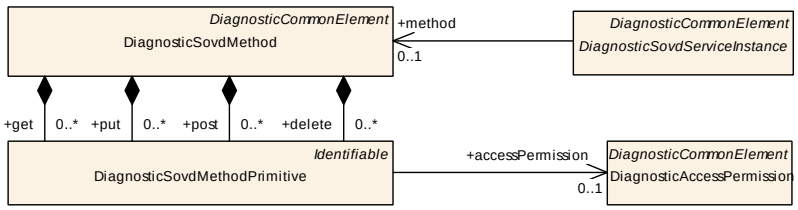


Figure 5.47: Modeling of `DiagnosticSovdMethod` and the relation to `DiagnosticAccessPermission`

[TPS_MANI_01432] Semantics of `DiagnosticSovdMethodPrimitive` ⌈Each of the primitive activities, modeled as `DiagnosticSovdMethodPrimitive` has a reference to `DiagnosticAccessPermission`. The basic semantics implemented in the context of a `DiagnosticSovdMethod` is modeled by means of the aggregation roles

- `get`
- `put`
- `post`
- `delete`

The `DiagnosticSovdMethodPrimitive`s are the building blocks out of which the semantics **and access permission**s of each `SOVD` service can be defined.⌋

[constr_10479] Restriction on values of attribute DiagnosticSovdMethodPrimitive.category aggregated in the role put in the context of SOVD Update

Imposition time: IT_DiagDes

⌈If a DiagnosticSovdMethod is referenced from a DiagnosticSovdUpdate and the attribute put.category exists, then the value of the attribute category within any DiagnosticSovdMethodPrimitive aggregated in the role put is restricted to the following values:

- PREPARE
- EXECUTE
- AUTOMATED

⌋

The meaning of the values of DiagnosticSovdMethodPrimitive.category is described in [24, SWS Diagnostics].

[constr_10480] Restriction on value of attribute DiagnosticSovdMethodPrimitive.category aggregated in the role get in the context of SOVD Update

Imposition time: IT_DiagDes

⌈If a DiagnosticSovdMethod is referenced from a DiagnosticSovdUpdate and the attribute get.category exists, then the value of the attribute category within any DiagnosticSovdMethodPrimitive aggregated in the role get is restricted to the value STATUS.⌋

[constr_10481] Restriction on values of attribute DiagnosticSovdMethodPrimitive.category aggregated in the role get in the context of SOVD Log

Imposition time: IT_DiagDes

⌈If a DiagnosticSovdMethod is referenced from a DiagnosticSovdLog and the attribute get exists, then the value of the attribute category within any DiagnosticSovdMethodPrimitive aggregated in the role get is restricted to the values

- CONFIG
- ENTRIES

⌋

[constr_10482] Restriction on values of attribute DiagnosticSovdMethodPrimitive.category aggregated in the roles put and delete in the context of SOVD Log

Imposition time: IT_DiagDes

⌈If a DiagnosticSovdMethod is referenced from a DiagnosticSovdLog and the attributes

- put

- delete

exist, then the value of the attribute category within any DiagnosticSovdMethod-Primitive aggregated in the role

- put
- delete

is restricted to the value CONFIG.⌋

[constr_10567] Existence of DiagnosticSovdMethod.get resp. delete in the context of DiagnosticSovdFaultMemoryAccess

Imposition time: IT_DiagDes

⌈If a DiagnosticSovdMethod is referenced from a DiagnosticSovdFault-MemoryAccess, then the following restriction for the existence and multiplicity of the attributes of the DiagnosticSovdMethod exist:

- put [0]
- post [0]
- get [0..1]
- delete [0..1]

⌋

Class	**DiagnosticSovdMethod**			
Note	A DiagnosticSovdMethod represents a re-usable complex operation (that consists of primitive operations) in the context of the communication of an SOVD server. **Tags:** atp.recommendedPackage=DiagnosticSovdMethods This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
delete	DiagnosticSovdMethodPrimitive	*	aggr	This represents the "delete" method primitive. **Tags:** atp.Splitkey=delete, delete.shortName
get	DiagnosticSovdMethodPrimitive	*	aggr	This represents the "get" method primitive. **Tags:** atp.Splitkey=get, get.shortName
post	DiagnosticSovdMethodPrimitive	*	aggr	This represents the "post" method primitive. **Tags:** atp.Splitkey=post, post.shortName
put	DiagnosticSovdMethodPrimitive	*	aggr	This represents the "delete" method primitive. **Tags:** atp.Splitkey=put, put.shortName

Table 5.107: DiagnosticSovdMethod

Class	**DiagnosticSovdMethodPrimitive**			
Note	This meta-class represents a primitive operation inside a DiagnosticSovdMethod. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DiagnosticSovdMethod.delete, DiagnosticSovdMethod.get, DiagnosticSovdMethod.post, DiagnosticSovdMethod.put			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
access Permission	DiagnosticAccessPermission	0..1	ref	This reference identifes the applicable access permission. **Tags:** atp.Splitkey=accessPermission

Table 5.108: DiagnosticSovdMethodPrimitive

5.7.3 SOVD Lock

[TPS_MANI_01396] Semantics of meta-class `DiagnosticSovdLock` ⌈SOVD does not integrate with the existing UDS-based session handling. But nevertheless, it is necessary to provide the ability to lock the access for one SOVD client such that no other SOVD or UDS client can access the software that is subject to the diagnostic activity.

For this purpose, meta-class DiagnosticSovdLock has been defined. It is referenced by DiagnosticAccessPermission as an additional criterion for computing the access permission for a diagnostic client.⌋

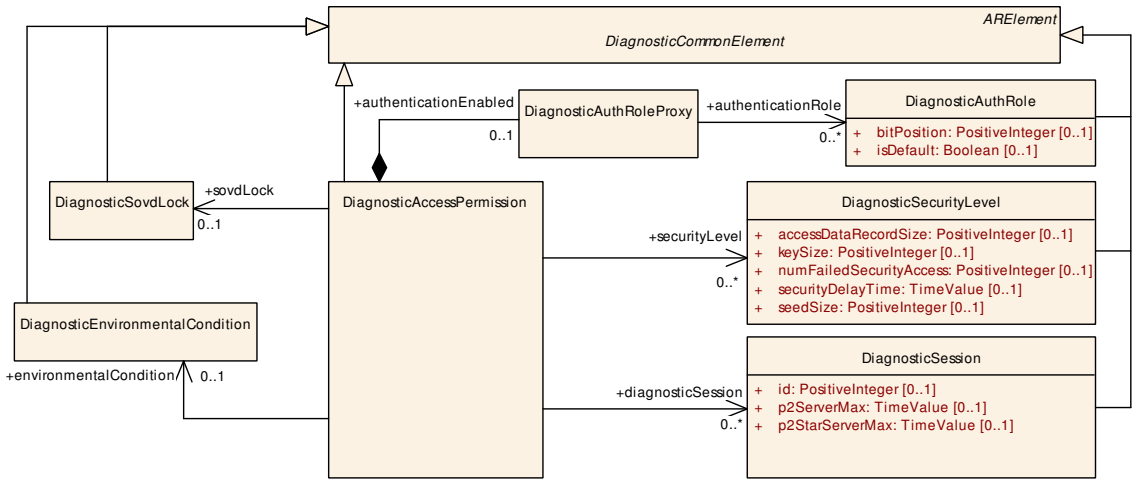


Figure 5.48: Modeling of SovdLock in the context of `DiagnosticAccessPermission`

Class	**DiagnosticSovdLock**
Note	This meta-class represents the ability to define an SOVD lock to be considered for the computation of a diagnostic access permission. **Tags:** atp.recommendedPackage=DiagnosticSovdLocks This Class is only used by the AUTOSAR Adaptive Platform.

△

Class	DiagnosticSovdLock			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 5.109: DiagnosticSovdLock

5.7.4 SOVD Content

5.7.4.1 Overview

Data Access via SOVD needs a detailed definition about the content of a data package exchanged between an SOVD client and the corresponding server (and vice versa). In other words, something like the established `DiagnosticDataIdentifier` is needed, but without the UDS-specific characteristic.

Unfortunately, there is no technical term (like “DID” in the UDS case) defined in the context of SOVD for this purpose that is more specific than “data”.

But using the generic term “data” specifically for this use case within SOVD can easily lead to misunderstandings because (without further qualification) it takes a lot of context for the reader to unambiguously distinguish “data” in the SOVD meaning of the word from “data” used as a generic concept.

Therefore, it is necessary to define the technical term `SOVD Data` that unambiguously clarifies that the specific meaning of “data in the SOVD meaning” without depending on a more or less obvious context.

A direct usage of existing UDS concepts for the purpose of describing “SOVD data” is considered confusing for the audience and there is also a significant risk that any overlapping between UDS and SOVD may break (and cause significant effort for fixing) as the two approaches evolve.

This means that analogous to a `DiagnosticDataIdentifier` as the representation of a UDS DID, `SOVD Data` is represented by the `DiagnosticAbstractSovdContent` and its subclasses: `DiagnosticSovdContent` and `DiagnosticSovdContentElement`.

The elements of a UDS `DiagnosticDataIdentifier` are modeled as a combination of `DiagnosticParameter` and `DiagnosticDataElement`. The equivalent concept on the side of SOVD shall be referred to as a `SOVD Data Element`.

[TPS_MANI_01493] Semantics of meta-class `DiagnosticSovdContent` ⌈Meta-class `DiagnosticSovdContent` represents the definition of the payload of a data-defined SOVD message that is transmitted between an SOVD Client and an SOVD server (or vice versa).

One example for the usage of DiagnosticSovdContent would be for the purpose of configuring a DiagnosticSovdData diagnostic service, comparable to a DID (DiagnosticDataIdentifier) on UDS.

Another example could be the configuration of the primitives of an DiagnosticSovd-Operation, comparable to a RID on UDS.⌋

[TPS_MANI_01494] Semantics of meta-class DiagnosticSovdContentElement ⌈Meta-class DiagnosticSovdContentElement represents a SOVD Data Element, comparable to a DID Element (DiagnosticDataElement) on UDS.⌋

The order inside a DID, as established by a DiagnosticParameter in the UDS case, is not required on SOVD because elements of the data package are identified by name instead of by something like a bitOffset.

The ability to define deeply nested data structures inside a DID, as available in the UDS case, is also available in the SOVD case, albeit using a different modeling approach[3].

In particular, the approach is that SOVD Data Element is contained in either SOVD Data or in another SOVD Data Element. A more detailed explanation is given in [TPS_MANI_01495] and in Figure 5.49.

[TPS_MANI_01495] Modeling of hierarchical data structures using DiagnosticSovdContentElement ⌈The modeling approach for defining hierarchical data structures in the context of DiagnosticSovdContent is based on the definition of DiagnosticSovdContentElement as an abstract base class, from which two other meta-classes inherit:

- DiagnosticSovdCompositeContentElement is again an abstract base class (representing one hierarchy level in a nested data structure), from which the following meta-classes inherit:
  - DiagnosticSovdRecordContentElement represents a structure that contains a collection of elements (represented by the role element with multiplicity 0..*) that in turn can be modeled as an aggregation of arbitrary subclasses of DiagnosticSovdContentElement.
  - DiagnosticSovdArrayContentElement represents an array of elements defined by the aggregation in the role element. This role has the multiplicity 0..1 to indicate that it just represents a definition of the nature of the elements of the array instead of an actual array element.
- DiagnosticSovdPrimitiveContentElement represents an atomic leaf element of the data structure. This means that DiagnosticSovdPrimitive-ContentElement could be aggregated either
  - directly in the role DiagnosticSovdContent.element or

[3]The ability for the creation of nested data structures was added later to the UDS modeling, which could only be done while observing a number of structural constraints to keep the backwards-compatibility of the AUTOSAR XML Schema

– somewhere deep in the hierarchical structure, i.e. in either of the roles

* DiagnosticSovdRecordContentElement.element
* DiagnosticSovdArrayContentElement.element

⌋

An overview of the modeling of DiagnosticSovdContent and its sub-structures is provided by Figure 5.49.

[constr_10742] Reference to DiagnosticSovdContent

Imposition time: IT_DiagDes

⌈Any DiagnosticSovdContent shall only be referenced by one of

- DiagnosticSovdConfigContentMapping
- DiagnosticSovdContentPortMapping

⌋

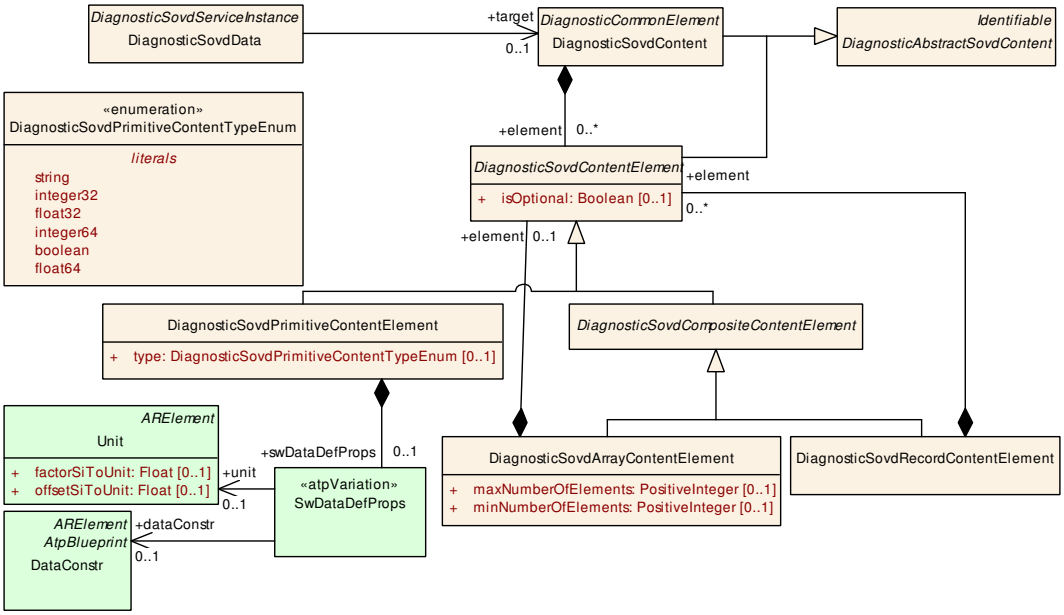


Figure 5.49: Modeling of DiagnosticSovdContent and DiagnosticSovdContentElement

Class	***DiagnosticAbstractSovdContent*** (abstract)
Note	This class acts as an abstract base class of it subclasses so that the modeling of references to all subclasses can be simplified. This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*
Subclasses	DiagnosticSovdContent, *DiagnosticSovdContentElement*

▽

△

Class	**DiagnosticAbstractSovdContent** (abstract)			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 5.110: DiagnosticAbstractSovdContent

Class	DiagnosticSovdContent			
Note	This meta-class represents a data packet exchanged between am SOVD Tester and an SOVD server. The semantics is very close to the semantics of a UDS DID, with some extensions. **Tags:** atp.recommendedPackage=DiagnosticSovdContents This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticAbstractSovdContent*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
element	DiagnosticSovdContentElement	*	aggr	This aggregation identifies the collection of elements of the enclosing DiagnosticSovdContent. **Tags:** atp.Splitkey=element, element.shortName

Table 5.111: DiagnosticSovdContent

Class	***DiagnosticSovdContentElement*** (abstract)			
Note	This meta-class represents an element of a DiagnosticSovdContent. When compared to the modeling of a UDS DID, this class would correspond to a mixture of DiagnosticParameter and DiagnosticDataElement. The concrete semantics of a DiagnosticSovdContentElement is defined by its subclasses. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *DiagnosticAbstractSovdContent*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*DiagnosticSovdCompositeContentElement*, DiagnosticSovdPrimitiveContentElement			
Aggregated by	DiagnosticSovdArrayContentElement.element, DiagnosticSovdContent.element, DiagnosticSovdRecordContentElement.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
isOptional	Boolean	0..1	attr	This attribute defines whether the enclosing DiagnosticSovdContentElement is considered optional. Note that array elements cannot be declared optional.

Table 5.112: DiagnosticSovdContentElement

5.7.4.2 SOVD Composite Content

[constr_10707] Relation of the values of attributes `minNumberOfElements` and `maxNumberOfElements`

Imposition time: `IT_DiagDes`

⌈The value of attribute `DiagnosticSovdArrayContentElement.minNumberOfElements` shall be **smaller or equal** to the value of `DiagnosticSovdArrayContentElement.maxNumberOfElements`.⌋

[TPS_MANI_01496] `DiagnosticSovdArrayContentElement` can have fixed or variable size ⌈The content of a `DiagnosticSovdArrayContentElement` may consist of either

- a fixed number of elements or
- a number of elements that may range between a configured lower bound and upper bound.

If the value of `minNumberOfElements` and `maxNumberOfElements` is identical, the enclosing `DiagnosticSovdArrayContentElement` is intended to be of **fixed size**.

If the value of `minNumberOfElements` and `maxNumberOfElements` is different while fulfilling [constr_10708], the enclosing `DiagnosticSovdArrayContentElement` is intended to be of **variable size**.⌋

[TPS_MANI_01497] `DiagnosticSovdContentElement` can be declared optional ⌈It is possible to declare `DiagnosticSovdContentElement`s as *optional* on nearly (see [constr_10707]) every level of the hierarchy by means of the attribute `isOptional`.

As long as [constr_10707] is fulfilled, it is possible to place non-optional `DiagnosticSovdContentElement`s inside an optional `DiagnosticSovdContentElement`.⌋

It does not make sense to declare the definition of the nature of an array element as optional. The mechanism to define “optional” array elements is to specify that the array shall have a variable number of elements, see [TPS_MANI_01495].

[constr_10708] `DiagnosticSovdContentElement` aggregated in the role `DiagnosticSovdArrayContentElement.element` shall not be declared *optional*

Imposition time: `IT_DiagDes`

⌈Any `DiagnosticSovdContentElement` aggregated in the role `DiagnosticSovdArrayContentElement.element` shall not set the value of attribute `isOptional` to `true`.⌋

Class	***DiagnosticSovdCompositeContentElement*** (abstract)			
Note	This abstract class represents the ability to define a hierarchical structure within a DiagnosticSovd Content. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *DiagnosticAbstractSovdContent*, *DiagnosticSovdContentElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	DiagnosticSovdArrayContentElement, DiagnosticSovdRecordContentElement			
Aggregated by	DiagnosticSovdArrayContentElement.element, DiagnosticSovdContent.element, DiagnosticSovdRecordContentElement.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 5.113: DiagnosticSovdCompositeContentElement

Class	DiagnosticSovdArrayContentElement			
Note	This meta-class represents the ability to define a DiagnosticSovdCompositeContentElement with array semantics. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARObject, DiagnosticAbstractSovdContent, DiagnosticSovdCompositeContentElement, DiagnosticSovdContentElement, Identifiable, MultilanguageReferrable, Referrable			
Aggregated by	DiagnosticSovdArrayContentElement.element, DiagnosticSovdContent.element, DiagnosticSovdRecordContentElement.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
element	DiagnosticSovdContentElement	0..1	aggr	This aggregation represents the specification of the nature of an element of an enclosing DiagnosticSovdArrayContentElement. It is only necessary to aggregate one element because the array semantics requires that all elements of the array have the same nature. **Tags:** atp.Splitkey=element, element.shortName xml.sequenceOffset=100
maxNumberOfElements	PositiveInteger	0..1	attr	This attribute specifies the maximum number of elements expected in the array. **Tags:** xml.sequenceOffset=20
minNumberOfElements	PositiveInteger	0..1	attr	This attribute specifies the minimum number of elements expected in the array. **Tags:** xml.sequenceOffset=10

Table 5.114: DiagnosticSovdArrayContentElement

[advisory_01019] Existence of the attribute `DiagnosticSovdArrayContentElement.element` ⌈For each `DiagnosticSovdArrayContentElement`, attribute `element` should exist.⌋

[advisory_01020] Existence of the attribute `DiagnosticSovdArrayContentElement.minNumberOfElements`

Imposition time: `IT_DiagDes`

⌈For each `DiagnosticSovdArrayContentElement`, attribute `minNumberOfElements` should exist.⌋

[constr_10711] Existence of attribute `DiagnosticSovdArrayContentElement.maxNumberOfElements`

Imposition time: `IT_DiagDes`

⌈For each `DiagnosticSovdArrayContentElement`, attribute `maxNumberOfElements` shall exist.⌋

Class	DiagnosticSovdRecordContentElement
Note	This meta-class represents the ability to define a DiagnosticSovdCompositeContentElement with record semantics. This Class is only used by the AUTOSAR Adaptive Platform.
Base	ARObject, DiagnosticAbstractSovdContent, DiagnosticSovdCompositeContentElement, DiagnosticSovdContentElement, Identifiable, MultilanguageReferrable, Referrable

▽

△

Class	DiagnosticSovdRecordContentElement			
Aggregated by	DiagnosticSovdArrayContentElement.element, DiagnosticSovdContent.element, DiagnosticSovdRecord ContentElement.element			
Attribute	Type	Mult.	Kind	Note
element	DiagnosticSovdContent Element	*	aggr	This aggregation represents the collection of record elements. **Tags:** atp.Splitkey=element, element.shortName

Table 5.115: DiagnosticSovdRecordContentElement

[advisory_01021] Existence of the attribute DiagnosticSovdRecordContent-Element.element

Imposition time: IT_DiagDes

⌈For each DiagnosticSovdRecordContentElement, attribute element shall exist.⌋

5.7.4.3 SOVD Primitive Content

The serialization approach of SOVD (a JSON object, backed by a JSON schema) is entirely different from the serialization approach for UDS (binary data). This means that the representation of SOVD content on the network does not need to be defined by means of an SwBaseType.

In particular, SOVD conveys the content in physical form such that no conversion rule from a binary representation to physical values is required. The definition of physical properties of the content on the network is still an important part of the design of the SOVD content.

This is the reason why DiagnosticSovdPrimitiveContentElement aggregates swDataDefProps that in turn references Unit and DataConstr.

[TPS_MANI_01498] Semantics of attribute DiagnosticSovdPrimitive-ContentElement.dataConstr.dataConstrRule.physConstrs ⌈For the transport of SOVD content over the network, attribute DiagnosticSovdPrimitive-ContentElement.dataConstr.dataConstrRule.physConstrs is used to provide information regarding the upper and lower bound of the values in the physical domain of the enclosing DiagnosticSovdPrimitiveContentElement.⌋

[TPS_MANI_01499] InternalConstrs is not supported in the context of a DiagnosticSovdPrimitiveContentElement ⌈As a consequence of [TPS_MANI_01498], attribute DiagnosticSovdPrimitiveContentElement.data-Constr.dataConstrRule shall ignore the existence of a potential aggregation in the role internalConstrs.⌋

[TPS_MANI_01500] Semantics of the attribute DiagnosticSovdPrimitive-ContentElement.swDataDefProps.unit ⌈For the transport of SOVD content over

the network, attribute DiagnosticSovdPrimitiveContentElement.swDataDefProps.unit is used to provide information regarding the physical unit of the enclosing DiagnosticSovdPrimitiveContentElement.⌋

[constr_10712] Allowed roles within SwDataDefProps in the context of meta-class DiagnosticSovdPrimitiveContentElement

Imposition time: IT_DiagDes

⌈Only the following roles within SwDataDefProps in the context of DiagnosticSovdPrimitiveContentElement are allowed:

- dataConstr
- unit

Any role that is not mentioned in the list above shall **not** be used.⌋

[TPS_MANI_01501] Semantics of the attribute DiagnosticSovdPrimitiveContentElement.type ⌈Attribute DiagnosticSovdPrimitiveContentElement.type is used to convey the type of the content in terms of an abstract description of the data type in which the content from the JSON payload is expected to be de-serialized.⌋

Class	**DiagnosticSovdPrimitiveContentElement**			
Note	This meta-class represents a primitive leaf element of a DiagnosticSovdContent. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *DiagnosticAbstractSovdContent*, *DiagnosticSovdContentElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DiagnosticSovdArrayContentElement.element, DiagnosticSovdContent.element, DiagnosticSovdRecordContentElement.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
swDataDefProps	SwDataDefProps	0..1	aggr	This aggregation represents the applicable data defnition properties. **Tags:** atp.Splitkey=swDataDefProps xml.sequenceOffset=100
type	DiagnosticSovdPrimitiveContentTypeEnum	0..1	attr	This attribute denotes the type of the primitive content element. **Tags:** xml.sequenceOffset=10

Table 5.116: DiagnosticSovdPrimitiveContentElement

Enumeration	**DiagnosticSovdPrimitiveContentTypeEnum**
Note	This enumeration defines the possible content type on primitive level.
Aggregated by	DiagnosticSovdPrimitiveContentElement.type
Literal	***Description***
boolean	The primitive content represents a boolean value. **Tags:** atp.EnumerationLiteralIndex=4

▽

△

Enumeration	DiagnosticSovdPrimitiveContentTypeEnum
float32	The primitive content represents a 32 bit floating-point number. **Tags:** atp.EnumerationLiteralIndex=2
float64	The primitive content represents a 64 bit floating-point number. **Tags:** atp.EnumerationLiteralIndex=5
integer32	The primitive content represents a 32 bit integer value. **Tags:** atp.EnumerationLiteralIndex=1
integer64	The primitive content represents a 64 bit integer number. **Tags:** atp.EnumerationLiteralIndex=3
string	The primitive content represents a string. **Tags:** atp.EnumerationLiteralIndex=0

Table 5.117: DiagnosticSovdPrimitiveContentTypeEnum

5.7.5 Environmental Data Condition to SOVD Content Element

Similar to obtaining values from a `DiagnosticDataElement`, as described in [3] Chapter 4.3.7.2.1, it is possible to refer to a `DiagnosticSovdContentElement` to evaluate an environmental condition.

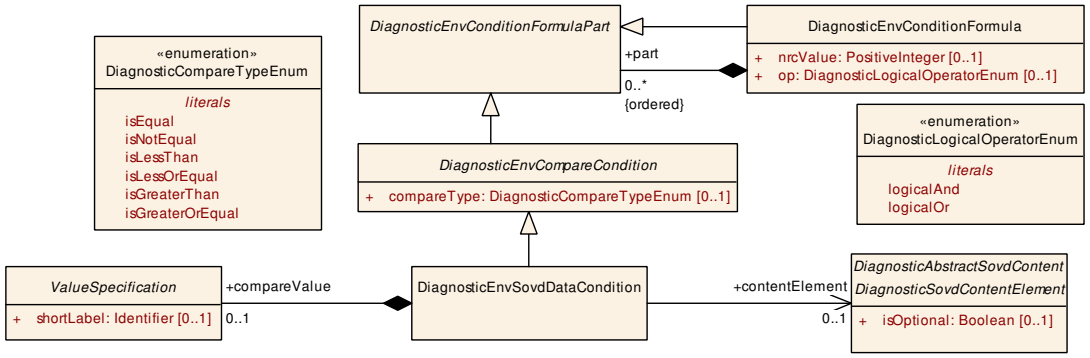


Figure 5.50: Diagnostic environmental condition with reference to a `DiagnosticSovd-ContentElement`

[TPS_MANI_01504] Semantics of `DiagnosticEnvSovdDataCondition` ⌈The meta-class `DiagnosticEnvSovdDataCondition` represents an atomic condition that compares the current value of the referenced `DiagnosticSovdContent-Element` with a constant value defined by the `ValueSpecification` aggregated in the role `compareValue`.⌋

Class	**DiagnosticEnvSovdDataCondition**			
Note	A DiagnosticEnvSovdDataCondition is an atomic condition that compares the current value of the referenced DiagnosticSovdContentElement with a constant value defined by the ValueSpecification. All compareTypes are supported. **Tags:** atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *DiagnosticEnvCompareCondition*, *DiagnosticEnvConditionFormulaPart*			
Aggregated by	DiagnosticEnvConditionFormula.part			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compareValue	ValueSpecification	0..1	aggr	This attribute represents a fixed compare value taken to evaluate the compare condition. **Tags:** atp.Status=candidate
contentElement	DiagnosticSovdContent Element	0..1	ref	This reference represents the related diagnostic SOVD content element. **Tags:** atp.Status=candidate

Table 5.118: DiagnosticEnvSovdDataCondition

[constr_10718] Existence of `DiagnosticEnvSovdDataCondition.compareValue`

Imposition time: `IT_DiagDes`

⌈For each `DiagnosticEnvSovdDataCondition`, that attribute `compareValue` shall exist.⌋

[constr_10719] Existence of `DiagnosticEnvSovdDataCondition.contentElement`

Imposition time: `IT_DiagDes`

⌈For each `DiagnosticEnvSovdDataCondition`, that attribute `contentElement` shall exist.⌋

5.7.6 SOVD Snapshots

[TPS_MANI_01505] Semantics of `DiagnosticTroubleCodeProps.sovdSnapshotContent` ⌈The reference in the role `DiagnosticTroubleCodeProps.sovdSnapshotContent` identifies the SOVD-related snapshot content that is not related to legislation.⌋

[TPS_MANI_01506] Semantics of `DiagnosticTroubleCodeProps.legislatedSovdFreezeFrameContent` ⌈The reference in the role `DiagnosticTroubleCodeProps.legislatedSovdFreezeFrameContent` identifies the SOVD-related snapshot content that is positively related to legislation.⌋

It is possible to configure snapshots for the UDS transport and snapshots for the SOVD transport for the same DTC, at the same time.

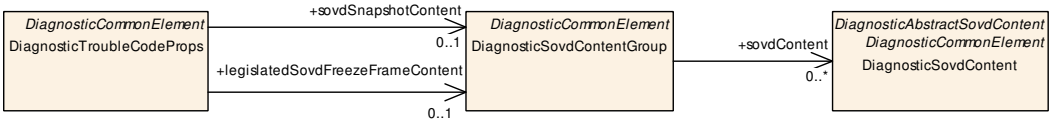


Figure 5.51: Modeling of `DiagnosticSovdContentGroup`

Class	**DiagnosticSovdContentGroup**			
Note	This meta-class represents a collection of SOVD content for the purpose of defining a DTC snapshot for the usage in SOVD. **Tags:** atp.recommendedPackage=SovdContentGroups This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sovdContent	DiagnosticSovdContent	*	ref	Reference to a collection of SOVD Content. **Tags:** atp.Splitkey=sovdContent

Table 5.119: DiagnosticSovdContentGroup

5.8 Diagnostic Miscellanea

[TPS_MANI_01358] Restriction for the configuration of diagnostic debouncing
⌈The restriction for the configuration of the diagnostic debouncing, as formulated in [TPS_DEXT_03004], shall **not** apply on *the AUTOSAR adaptive platform.*⌋

[constr_10483] Applicable values of `baseTypeEncoding` in the context of the definition of a `DiagnosticDataElement`

Imposition time: `IT_DiagDes`

⌈Any `SwBaseType` referenced in the role `DiagnosticDataElement.swDataDefProps.baseType` shall only set one of the following values for attribute `SwBaseType.baseTypeDefinition.baseTypeEncoding`:

- **`BOOLEAN`** (for `baseTypeSize` = 8)
- **`NONE`** (for `baseTypeSize` = 8, 16, 32, or 64)
- **`2C`** (for `baseTypeSize` = 8, 16, 32, or 64)
- **`IEEE754`** (for `baseTypeSize` = 32 or 64)
- **`UTF-8`** (for `baseTypeSize` = 8)

⌋

A description of the standardized values mentioned in [constr_10483] can be found in the specification of the [2, TPS Software Component Template].

The relation between the possible values of `baseTypeEncoding` and the minimum value of `baseTypeSize` is documented in [TPS_SWCT_01876].

6 Machine Design

6.1 Overview

On the *AUTOSAR classic platform*, the design of communication-related aspects is attached to the existence of meta-class `EcuInstance`. A similar approach to support the definition of communication-related aspects early in the workflow (and definitely before the manifest configuration starts) is also needed on the *AUTOSAR adaptive platform*.

Therefore, `MachineDesign` has been introduced 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).

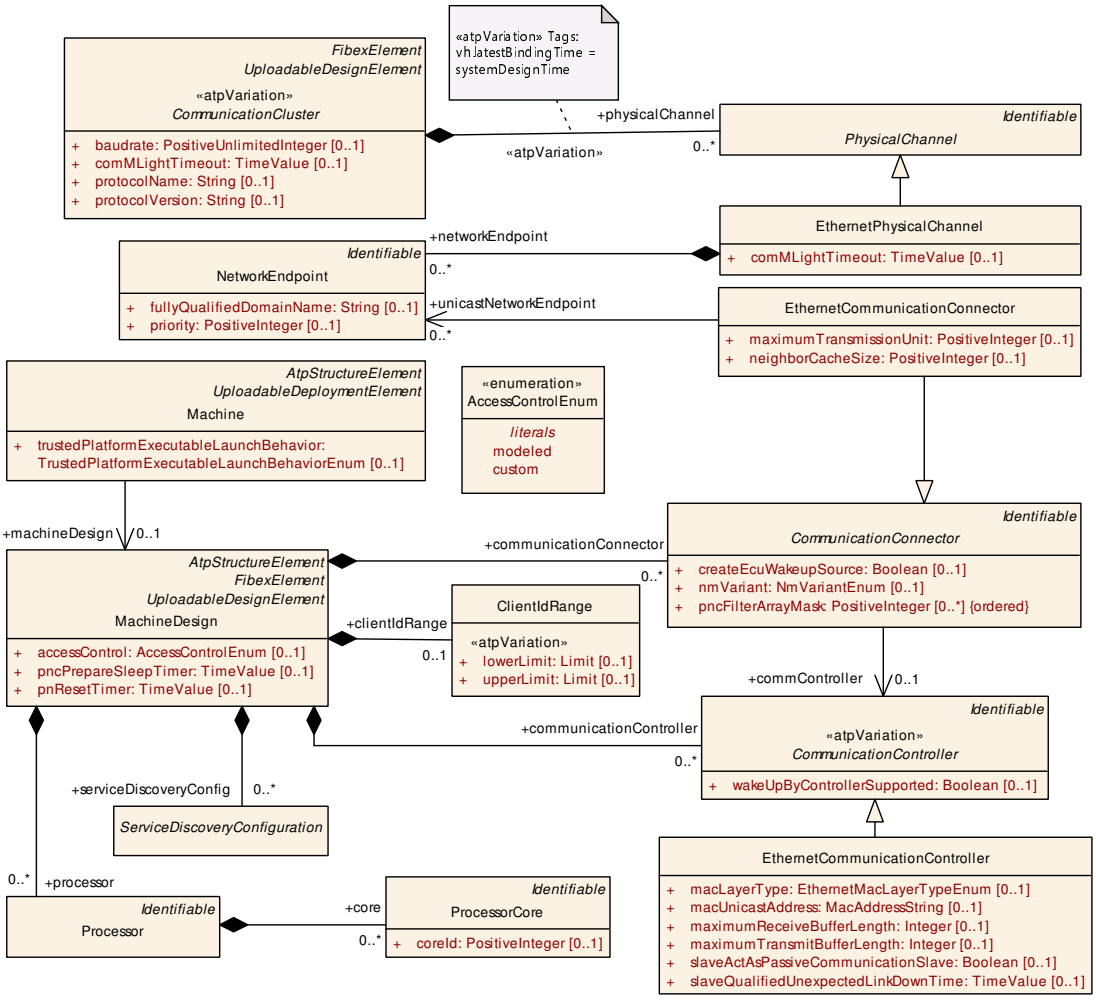


Figure 6.1: MachineDesign

[TPS_MANI_03347] Semantics of `MachineDesign.clientIdRange` ⌈`MachineDesign.clientIdRange` allows the definition of a range of numbers for a `MachineDesign` from which the client identifier is chosen. The client identifier identifies the client of a Service on the Machine.⌋

[constr_9434] Overlap of `ClientIdRange`s in the context of the enclosing `System`

Imposition time: `IT_SysDes`

⌈The `ClientIdRange` defined for a `MachineDesign` shall not overlap with the `ClientIdRange`s of any other `MachineDesign`s or `EcuInstance`s in the context of the enclosing `System`.⌋

As the *AUTOSAR adaptive platform* focuses on the usage of Ethernet for communication, the task of creating the communication design in the context of the `MachineDesign` boils down to the specification of IP addresses (see Chapter 6.5) and the configuration of service discovery, see Chapter 6.6.

6.2 Processor

6.2.1 Overview

The `MachineDesign` is able to aggregate one or several `Processor`s, as depicted by Figure 11.1. Each `Processor` consists of one or several `ProcessorCore`s.

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] Value of `ProcessorCore.coreId`

Imposition time: `IT_SysDes`

⌈The value of `ProcessorCore.coreId` shall be unique in the context of the enclosing `Processor`.⌋

Class	**Processor**			
Note	This represents a processor for the execution of an AUTOSAR adaptive platform This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	Machine.processor, MachineDesign.processor			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
core	ProcessorCore	*	aggr	This represents the collection of cores owned by the enclosing processor. **Tags:** atp.Splitkey=core, core.shortName

Table 6.1: Processor

[constr_10172] Multiplicity of attribute `Processor.core`

Imposition time: `IT_SysDes`

⌈For each `Processor`, the attribute `core` shall exist.⌋

Class	**ProcessorCore**			
Note	This meta-class represents the ability to model a processor core for the execution of an AUTOSAR adaptive platform. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	Processor.core			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
coreId	PositiveInteger	0..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.

Table 6.2: ProcessorCore

[constr_10173] Multiplicity of attribute ProcessorCore.coreId

Imposition time: IT_SysDes

⌈For each ProcessorCore, the attribute coreId shall exist.⌋

6.2.2 Core Affinity

There are two ways to define the core affinity of a Process:

- On *design level*, the references ProcessDesignToMachineDesignMapping.shallRunOn and ProcessDesignToMachineDesignMapping.shallNotRunOn can be used.

  This is a way to define core affinity in the early stages of a project, in the context of an overall system design.

- On *target configuration level*, the references ProcessToMachineMapping.shallRunOn and ProcessToMachineMapping.shallNotRunOn can be used.

  From the standard's perspective, it is in principle possible to override the definition of core affinity on the design level in this step. Projects need to decide whether this capability shall be available.

[TPS_MANI_01477] Description of Core affinity ⌈The meta-class ProcessDesignToMachineDesignMapping provides the ability to restrict the assignment of processes to selected ProcessCores with the two references shallRunOn and shallNotRunOn.⌋

[constr_10659] Consistency of references in the roles ProcessDesignToMachineDesignMapping.machineDesign and shallRunOn resp. shallNotRunOn

Imposition time: IT_SysDes

⌈Any ProcessorCore that is referenced in the roles shallRunOn and shallNotRunOn shall be aggregated (via MachineDesign.processor) by the MachineDesign referenced in the role ProcessDesignToMachineDesignMapping.machineDesign.⌋

[constr_10660] Usage of shallRunOn and shallNotRunOn references

Imposition time: IT_SysDes

⌈The ProcessorCore that is referenced by a ProcessDesignToMachineDesignMapping in the role shallRunOn or shallNotRunOn shall be aggregated by the MachineDesign that is referenced in the role machineDesign by the same ProcessDesignToMachineDesignMapping.⌋

[constr_10661] Consistency of references shallRunOn and shallNotRunOn

Imposition time: IT_SysDes

⌈Within the context of one ProcessDesignToMachineDesignMapping, all ProcessorCores referenced in the role shallRunOn or shallNotRunOn shall be aggregated by the same Processor.⌋

If a model defines that a given ProcessDesign shall run on a selected set of ProcessorCores then there is hardly a use case to (in addition) also specify the opposite, i.e. that the ProcessDesign 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 ProcessDesign 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_10662].

[constr_10662] Mutual exclusive existence of references shallRunOn and shallNotRunOn

Imposition time: IT_SysDes

⌈For any given ProcessDesignToMachineDesignMapping, either the reference in the role shallRunOn or the reference in the role shallNotRunOn may exist.⌋

6.3 Functional Cluster Design

This chapter defines configuration aspects for Functional Clusters on the design level. The design models for individual functional clusters are introduced in the following subchapters.

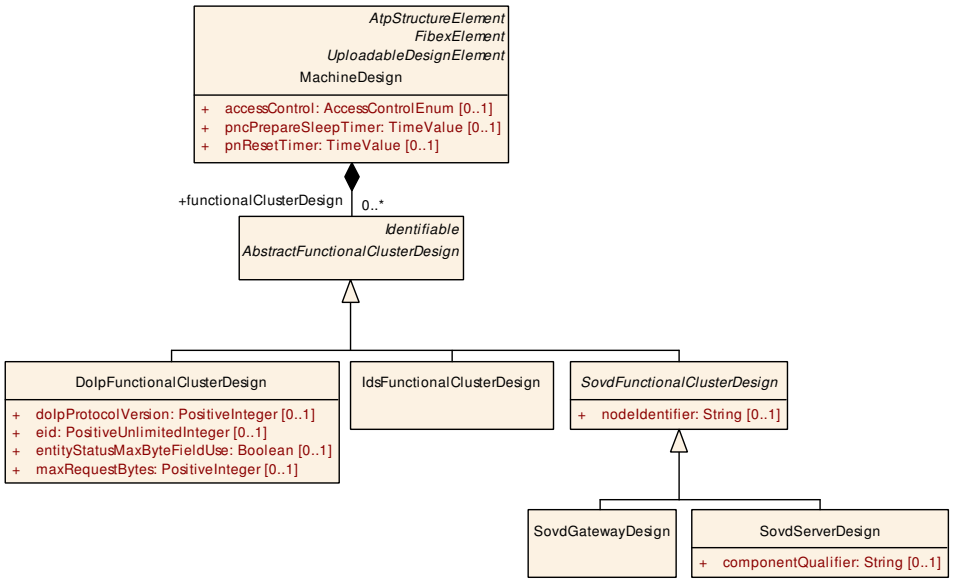


Figure 6.2: Abstract FunctionalCluster Design

Class	***AbstractFunctionalClusterDesign*** (abstract)			
Note	The concrete subclasses of this meta-class are defining configuration settings for Functional Clusters on the machine design level. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	DoIpFunctionalClusterDesign, IdsFunctionalClusterDesign, *SovdFunctionalClusterDesign*			
Aggregated by	MachineDesign.functionalClusterDesign			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 6.3: AbstractFunctionalClusterDesign

6.3.1 Ids Functional Cluster Design

[TPS_MANI_03339] Semantics of `IdsFunctionalClusterDesign`

Status: DRAFT

⌈The design aspects of the `Intrusion Detection System` are modeled by means of the meta-class `IdsFunctionalClusterDesign`.⌋

The modeling of `IdsFunctionalClusterDesign` in the `MachineDesign` allows an OEM to perform the network-specific configuration settings used by the `Intrusion Detection System`.

The security events that are transmitted over the `networkInterface` are defined in the IdsM Target-Configuration as described by [TPS_MANI_03343].

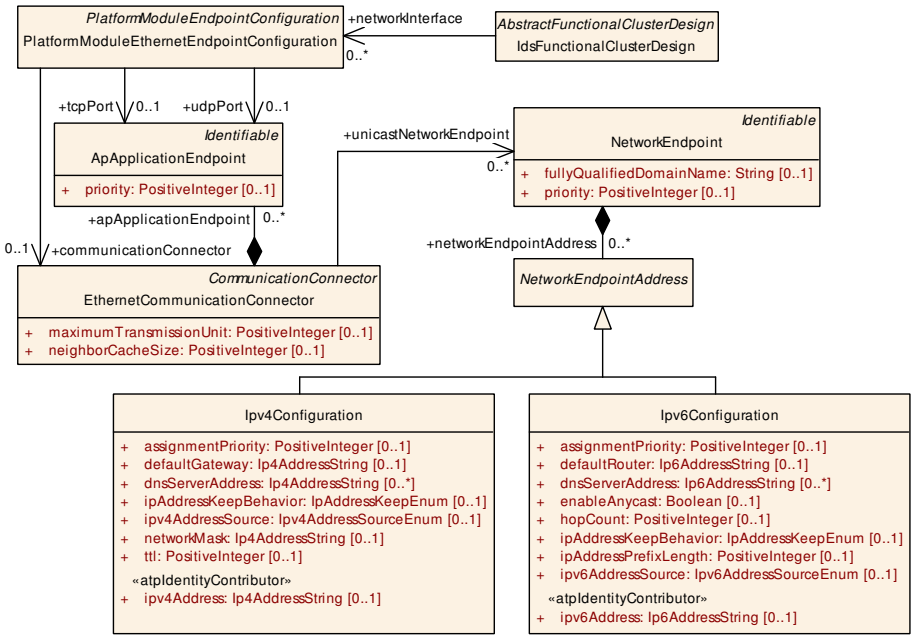


Figure 6.3: Intrusion Detection FunctionalCluster Design

Class	**IdsFunctionalClusterDesign**			
Note	This meta-class defines the intrusion detection system configuration settings in the MachineDesign. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractFunctionalClusterDesign*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	MachineDesign.functionalClusterDesign			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
network Interface	PlatformModule EthernetEndpoint Configuration	*	ref	This association contains the network configuration that shall be applied to an IDS entity. **Tags:** atp.Splitkey=networkInterface

Table 6.4: IdsFunctionalClusterDesign

[constr_9429] Upper multiplicity of reference in the role IdsFunctional-ClusterDesign.networkInterface

Imposition time: IT_MachDes

⌈In the context of IdsFunctionalClusterDesign, the reference in the role networkInterface shall exist at most once.⌋

[TPS_MANI_03340] local ApApplicationEndpoint and NetworkEndpoint for transmission of security events

Status: DRAFT

⌈The ApApplicationEndpoint (Udp or Tcp port number) and NetworkEndpoint over which the security event is transmitted on a VLAN are defined by the IdsFunctionalClusterDesign.networkInterface reference. The referenced PlatformModuleEthernetEndpointConfiguration defines:

- the local NetworkEndpoint via PlatformModuleEthernetEndpoint-Configuration.communicationConnector.unicastNetworkEndpoint

- the udp Port over which the security event is transmitted by PlatformModuleEthernetEndpointConfiguration.udpPort
- the tcp Port over which the security event is transmitted by PlatformModuleEthernetEndpointConfiguration.tcpPort

It is allowed to define either a udpPort or tcpPort or both.⌋

[constr_9329] Condition for the existence of references in the roles PlatformModuleEthernetEndpointConfiguration.tcpPort and PlatformModuleEthernetEndpointConfiguration.udpPort

Imposition time: IT_MachDes

⌈The PlatformModuleEthernetEndpointConfiguration shall only reference ApApplicationEndpoints in the role tcpPort and udpPort that are aggregated by the EthernetCommunicationConnector that is also referenced by the same PlatformModuleEthernetEndpointConfiguration in the role communicationConnector.⌋

[TPS_MANI_03332] Remote credentials for transmission of security events

Status: DRAFT

⌈The remote IP Address, protocol and port number to which the security event is transmitted on a VLAN is defined by PlatformModuleEthernetEndpointConfiguration.remoteConfig. The aggregated RemoteEndpointConfiguration defines:

- the remote IP Address via RemoteEndpointConfiguration.ipv4Address and RemoteEndpointConfiguration.ipv6Address
- the udp Port to which the security event is transmitted by RemoteEndpointConfiguration.udpPort
- the tcp Port to which the security event is transmitted by RemoteEndpointConfiguration.tcpPort

⌋

6.3.2 Sovd Functional Cluster Design

The Service-oriented Vehicle Diagnostics (SOVD) architecture described in [25, ASAM SOVD] foresees the existence of an SOVD gateway that communicates with:

- an SOVD Client outside the vehicle's network
- the collection of SOVD Servers inside the vehicle's network.

Some of the configuration items of SOVD gateway and SOVD Server can be shared, but some are distinct for one of the two SOVD entities. This aspect is reflected in the formalization as SovdServerDesign and SovdGatewayDesign.

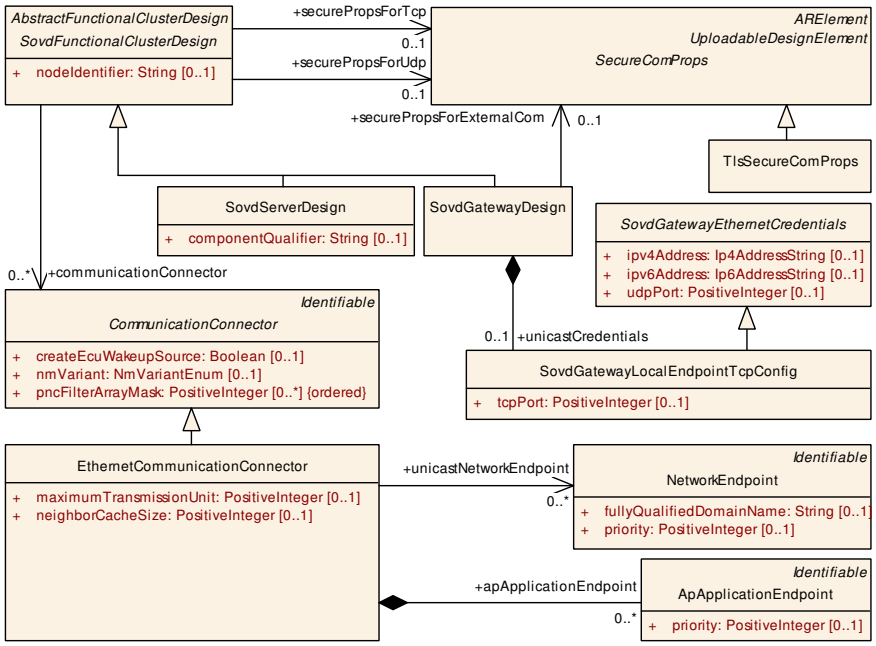


Figure 6.4: Sovd FunctionalCluster Design

[TPS_MANI_03341] Semantics of `SovdFunctionalClusterDesign` ⌈The design aspects of the SOVD architecture are modeled by means of the meta-class SovdFunctionalClusterDesign and the two sub-classes:

- SovdServerDesign: representation of SOVD Server in the MachineDesign reachable via SOVD somewhere in the vehicle network.
- SovdGatewayDesign: representation of the "edge-node" that connects an SOVD Client with the SOVD Servers in the vehicle network. This is a logical role, i.e. the MachineDesign on which an SovdGatewayDesign is located could also host one or more SovdServerDesigns.

⌋

Class	***SovdFunctionalClusterDesign*** (abstract)			
Note	This meta-class defines the Sovd configuration settings in the MachineDesign. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractFunctionalClusterDesign*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	SovdGatewayDesign, SovdServerDesign			
Aggregated by	MachineDesign.functionalClusterDesign			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Connector	Communication Connector	*	ref	This reference identifies the collection of communication connectors used by the SOVD for vehicle-internal communication. **Tags:** atp.Splitkey=communicationConnector

▽

△

Class	***SovdFunctionalClusterDesign*** (abstract)			
nodeIdentifier	String	0..1	attr	This attribute represents the local hostname of the SOVD server to be used during the execution of the DNS-SD protocol.
securePropsForTcp	SecureComProps	0..1	ref	This reference is used to identify the applicable TCP secure com properties for the external communication of the enclosing SOVD server. **Tags:** atp.Splitkey=securePropsForTcp
securePropsForUdp	SecureComProps	0..1	ref	This reference is used to identify the applicable UDP secure com properties for the external communication of the enclosing SOVD server. **Tags:** atp.Splitkey=securePropsForUdp

Table 6.5: SovdFunctionalClusterDesign

[TPS_MANI_01400] Secure communication between SOVD Gateway and SOVD Server ⌈The *unicast* communication between SOVD gateway and SOVD Server is modeled using the same concepts as for other vehicle-internal communication, i.e. by reference to a CommunicationConnector in the role SovdFunctionalClusterDesign.communicationConnector.⌋

[TPS_MANI_01401] Support for TLS in vehicle-internal SOVD communication ⌈The internal SOVD communication can be secured by TLS. The configuration of the TLS credentials is modeled by means of the reference SovdFunctionalClusterDesign.securePropsForTcp.⌋

The usage of TLS over UDP is not applicable to SOVD communication.

6.3.2.1 SovdServerDesign

[TPS_MANI_03342] Semantics of SovdServerDesign ⌈The SOVD Server is represented in the MachineDesign by the SovdServerDesign meta-class.⌋

Class	**SovdServerDesign**			
Note	This meta-class represents the configuration of an SOVD server. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractFunctionalClusterDesign*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SovdFunctionalClusterDesign*			
Aggregated by	MachineDesign.functionalClusterDesign			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
componentQualifier	String	0..1	attr	This attributes described the component qualifier used to compose an SOVD query.

Table 6.6: SovdServerDesign

[TPS_MANI_01402] Identification of the SOVD Server ⌈The identification of the SOVD Server is supported by the configuration of the attribute SovdServerDesign.componentQualifier.

In other words, the SOVD Client communicates with the SOVD Server by putting the value of the componentQualifier into the *component-identifier part* of the query URL.⌋

[constr_10400] Existence of SovdServerDesign.componentQualifier

Imposition time: IT_Mani

⌈For each SovdServerDesign, attribute componentQualifier shall exist.⌋

6.3.2.2 SovdGatewayDesign

The SOVD gateway represents the "edge-node" of the SOVD-related internal communication network towards the tester.

This means that the SOVD gateway's configuration not only consists of the definition of internal communication paths, it also needs to expose credentials for the communication with a SOVD Client as well as the use of search functionality via *multicast DNS* (mDNS), as described in [26, RFC-6762] and *DNS Service Discovery* (DNS-SD), see [27, RFC-6763].

[TPS_MANI_03343] Semantics of SovdGatewayDesign ⌈The SOVD Gateway is represented in the MachineDesign by the SovdGatewayDesign meta-class.⌋

[TPS_MANI_01403] External communication of the SOVD gateway ⌈The specification of external communication with the SOVD Client needs to cover the following cases:

- multicast DNS: usage of standardized mDNS credentials[1].
- DNS-SD: configuration of hostname in the form of SovdFunctionalClusterDesign.nodeIdentifier.
- unicast communication with the diagnostic client: aggregation of SovdGatewayLocalEndpointTcpConfig in the role SovdGatewayDesign.unicastCredentials.

⌋

[TPS_MANI_01404] Secure communication between SOVD gateway and SOVD Client ⌈The communication between SOVD gateway and SOVD Client can be secured by TLS. The TLS configuration for the SOVD gateway can be created by means of the reference in the role SovdGatewayDesign.securePropsForExternalCom.⌋

[1] mDNS uses multicast UDP credentials: 224.0.0.251, port 5353 (IPv4) [28] [29] or FF02::FB (IPv6), port 5353 [30] [29], reserved by IANA.

Class	SovdGatewayDesign			
Note	This meta-class represents the configuration of an SOVD gateway. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractFunctionalClusterDesign*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SovdFunctionalClusterDesign*			
Aggregated by	MachineDesign.functionalClusterDesign			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
securePropsForExternalCom	SecureComProps	0..1	ref	This reference is used to identify the applicable secure com properties for the external communication of the enclosing SOVD gateway. **Tags:** atp.Splitkey=securePropsForExternalCom
unicastCredentials	SovdGatewayLocalEndpointTcpConfig	0..1	aggr	This aggregation is used to describe the unicast credentials for external (i.e. with a tester) communication of the enclosing SOVD gateway. **Tags:** atp.Splitkey=unicastCredentials

Table 6.7: SovdGatewayDesign

Class	SovdGatewayLocalEndpointTcpConfig			
Note	This meta-class provides the ability to define the TCP configuration of a local endpoint for external communication of an SOVD gateway. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *SovdGatewayEthernetCredentials*			
Aggregated by	SovdGatewayDesign.unicastCredentials			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
tcpPort	PositiveInteger	0..1	attr	This attribute describes the port number of the port used for TCP communication.

Table 6.8: SovdGatewayLocalEndpointTcpConfig

Class	*SovdGatewayEthernetCredentials* (abstract)			
Note	This meta-class represents the ability to define Ethernet credentials for the purpose of connecting a client to an SOVD gateway. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Subclasses	SovdGatewayLocalEndpointTcpConfig			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ipv4Address	Ip4AddressString	0..1	attr	This attribute represents the IPv4 address for the case that IPv4 is used for communication between the SOVD gateway and a client.
ipv6Address	Ip6AddressString	0..1	attr	This attribute represents the IPv6 address for the case that IPv6 is used for communication between the SOVD gateway and a client.
udpPort	PositiveInteger	0..1	attr	This attribute describes the port number of the port used for UDP communication.

Table 6.9: SovdGatewayEthernetCredentials

[constr_10401] Existence of `SovdGatewayLocalEndpointTcpConfig.tcpPort`

Imposition time: `IT_Mani`

⌈For each `SovdGatewayLocalEndpointTcpConfig`, attribute `tcpPort` shall exist.⌋

[constr_10402] Existence of attribute `SovdGatewayEthernetCredentials.ipv4Address` vs. `SovdGatewayEthernetCredentials.ipv6Address`

Imposition time: `IT_Mani`

⌈For each `SovdGatewayEthernetCredentials`, at least one of attributes

- `SovdGatewayEthernetCredentials.ipv4Address`
- `SovdGatewayEthernetCredentials.ipv6Address`

shall exist.⌋

[constr_10403] Existence of `SovdGatewayEthernetCredentials.udpPort`

Imposition time: `IT_Mani`

⌈For each `SovdGatewayEthernetCredentials`, attribute `udpPort` shall exist.⌋

6.3.3 DoIp Functional Cluster Design

[TPS_MANI_03326] MachineDesign-specific configuration settings for DoIP ⌈The meta-class `DoIpFunctionalClusterDesign` allows the configuration of the DoIp Functional Cluster on the `MachineDesign` level.⌋

The target configuration of `DoIpInstantiation` is described in Chapter 12.5.

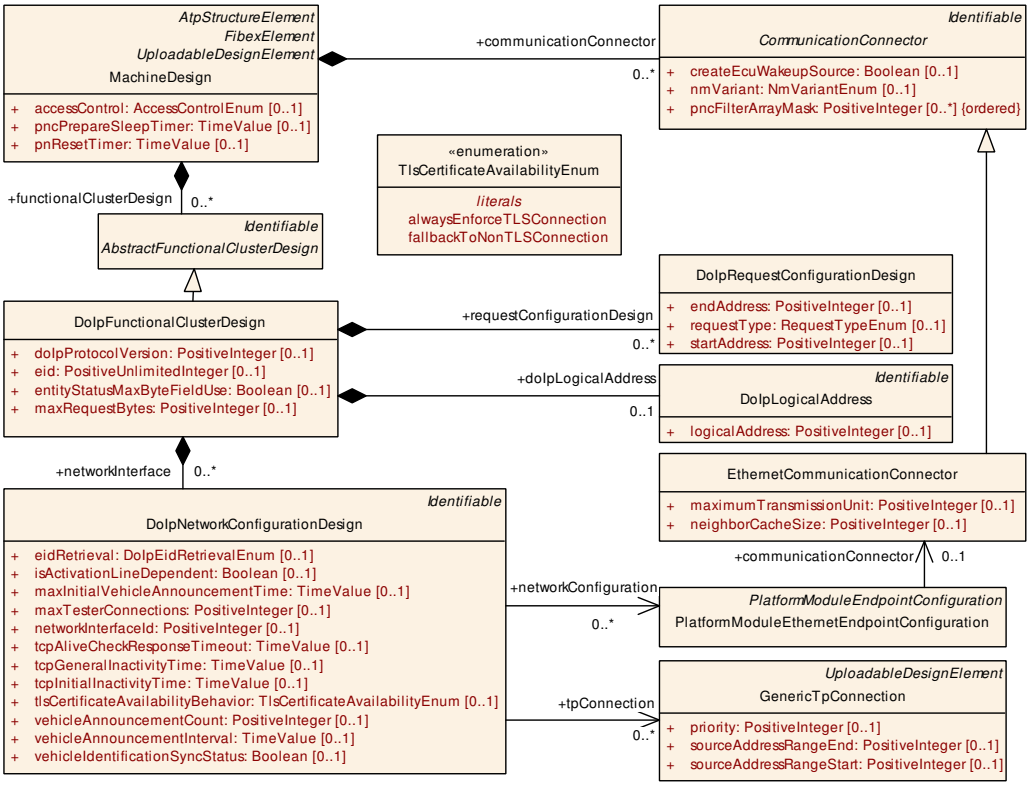


Figure 6.5: DoIp FunctionalCluster Design

[TPS_MANI_03327] Network Interface configuration for DoIP ⌈The meta-class DoIpNetworkConfigurationDesign contains all configuration settings that are specific to a configured network connection. The network connection is configured with the PlatformModuleEthernetEndpointConfiguration that is referenced by the DoIpNetworkConfigurationDesign in the role networkConfiguration.

The tcpPort and udpPort references to the ApApplicationEndpoint are used to configure the Transport Protocol (Udp or Tcp) and the used Port number. The IP Address is configured in the NetworkEndpoint that is referenced by the PlatformModuleEthernetEndpointConfiguration via the EthernetCommunicationConnector.⌋

[TPS_MANI_03323] Support for TLS in DoIP communication ⌈DoIP communication that is secured by TLS is modeled by a DoIpNetworkConfiguration that references the PlatformModuleEthernetEndpointConfiguration that in turn references SecureComProps in the role secureComPropsForTcp, which contains the credentials to secure the DoIP communication over TCP.⌋

Class	**DoIpFunctionalClusterDesign**			
Note	This meta-class defines the attributes for the DoIP configuration settings in the MachineDesign. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractFunctionalClusterDesign*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	MachineDesign.functionalClusterDesign			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
doIpLogical Address	DoIpLogicalAddress	0..1	aggr	This aggregation contains information about the DoIP logical address. **Tags:** atp.Splitkey=doIpLogicalAddress, doIpLogical Address.shortName
doIpProtocol Version	PositiveInteger	0..1	attr	Configures the DoIP protocol version used in the generic DoIP header. The valid range of this parameter is defined by the always latest release of ISO 13400-2 and can be extended with every new release of the ISO document. As example a value of 0x03 defines the ISO 13400-2:2019 release.
eid	PositiveUnlimitedInteger	0..1	attr	Configured EID (Entity ID) used for VehicleIdentification Request.
entityStatusMax ByteFieldUse	Boolean	0..1	attr	This attribute is used to distinguish the optional support of the Max data size element of a diagnostic entity status response.
maxRequest Bytes	PositiveInteger	0..1	attr	Specifies the maximum allowed bytes of a DoIP message request without the DoIP header.
network Interface	DoIpNetwork ConfigurationDesign	*	aggr	This element collects DoIP properties that are network interface specific. **Tags:** atp.Splitkey=networkInterface, network Interface.shortName

▽

△

Class	DoIpFunctionalClusterDesign			
request Configuration Design	DoIpRequest ConfigurationDesign	*	aggr	Request configuration that is used to determine whether an incoming DiagnosticMessage request needs to be interpreted as PHYSICAL or FUNCTIONAL. Any request with target address not within the configured target address range will be rejected. **Tags:** atp.Splitkey=requestConfigurationDesign

Table 6.10: DoIpFunctionalClusterDesign

[constr_10188] Existence of attribute DoIpFunctionalClusterDesign.doIp-LogicalAddress

Imposition time: IT_MachDes

⌈For each DoIpFunctionalClusterDesign, attribute doIpLogicalAddress shall exist.⌋

[constr_10611] Existence of attribute DoIpLogicalAddress.logicalAddress

Imposition time: IT_SysDes

⌈For each DoIpLogicalAddress, attribute logicalAddress shall exist.⌋

Class	DoIpNetworkConfigurationDesign			
Note	This element collects DoIP properties that are network interface specific. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DoIpFunctionalClusterDesign.networkInterface			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
eidRetrieval	DoIpEidRetrievalEnum	0..1	attr	This attribute defines how DoIp Entitiy Identification is retrieved.
isActivationLine Dependent	Boolean	0..1	attr	This attribute defines whether the network interface • is started "on-demand" when an activation line is sensed or • is always available.
maxInitial Vehicle Announcement Time	TimeValue	0..1	attr	Upper bound for the time to wait in [s] for sending first vehicle announcement message after IP address assignment. Represents parameter A_DoIP_Announce_ Wait of ISO 13400-2:2019. The value of this timing shall be determined randomly in the closed interval [0..max InitialVehicleAnnouncementTime].
maxTester Connections	PositiveInteger	0..1	attr	Maximum amount of tester connections that shall be maintained at one time before alive check is performed.
network Configuration	PlatformModule EthernetEndpoint Configuration	*	ref	Network configuration (Protocol, Port, IP Address) for transmission of DoIP messages on a specific VLAN. **Tags:** atp.Splitkey=networkConfiguration
network InterfaceId	PositiveInteger	0..1	attr	This attribute defines the identifier for the DoIPInterface.
tcpAliveCheck Response Timeout	TimeValue	0..1	attr	Timeout in [s] for waiting for a response to an Alive Check request before the connection is considered to be disconnected. Represents parameter T_TCP_AliveCheck of ISO 13400-2:2019.

▽

△

Class	DoIpNetworkConfigurationDesign			
tcpGeneral InactivityTime	TimeValue	0..1	attr	Timeout in [s] for maximum inactivity of a TCP socket connection before the DoIP module will close the according socket connection. Represents parameter T_TCP_General_Inactivity of ISO 13400-2:2019.
tcpInitial InactivityTime	TimeValue	0..1	attr	Timeout in [s] used for initial inactivity of a connected TCP socket connection directly after socket connection. Represents parameter T_TCP_Initial_Inactivity of ISO 13400-2:2019.
tlsCertificate AvailabilityBehavior	TlsCertificateAvailabilityEnum	0..1	attr	Defines the DoIP behavior in case DoIP is configured to require always a TLS connection but no valid TLS certificate is present (available).
tpConnection	GenericTpConnection	*	ref	Reference to a TpConnection that identifies the receiver(s) of a particular communication **Tags:** atp.Splitkey=tpConnection
vehicle AnnouncementCount	PositiveInteger	0..1	attr	Number of vehicle announcement messages on IP address assignment. Represents parameter A_DoIP_Announce_Num of ISO 13400-2:2019.
vehicle AnnouncementInterval	TimeValue	0..1	attr	Time to wait in [s] for sending subsequent vehicle announcement messages. Represents parameter A_DoIP_Announce_Interval of ISO 13400-2:2019.
vehicle IdentificationSyncStatus	Boolean	0..1	attr	Defines if the optional VIN/GID synchronization status is used additionally in the vehicle identification/ announcement.

Table 6.11: DoIpNetworkConfigurationDesign

Enumeration	TlsCertificateAvailabilityEnum
Note	This enumeration provides options for dealing with certificates. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	DoIpNetworkConfigurationDesign.tlsCertificateAvailabilityBehavior
Literal	*Description*
alwaysEnforceTLSConnection	DoIP is always enforcing TLS connection, regardless if a TLS certificate is present or not. If this option is selected the ECU supplier has to ensure, that a valid TLS certificate is always present. Without a TLS certificate no DoIP connection is possible, which would mean the ECU cannot be diagnosed via DoIP/UDS. **Tags:** atp.EnumerationLiteralIndex=0
fallbackToNonTLSConnection	If no valid TLS certificate is present, the DoIP will allow a fallback to a non-TLS secured connection on DoIP. **Tags:** atp.EnumerationLiteralIndex=1

Table 6.12: TlsCertificateAvailabilityEnum

[constr_9323] Restriction of DoIpFunctionalClusterDesign on a MachineDesign

Imposition time: IT_SysDes

⌈Each MachineDesign shall aggregate at most one DoIpFunctionalClusterDesign in the role functionalClusterDesign.⌋

[constr_9324] Allowed PlatformModuleEthernetEndpointConfiguration.communicationConnector references

Imposition time: IT_SysDes

⌈The PlatformModuleEthernetEndpointConfiguration that is referenced by a DoIpNetworkConfigurationDesign shall only reference EthernetCommunicationConnectors that are aggregated by the MachineDesign that is also aggregating the DoIpNetworkConfigurationDesign via the DoIpFunctionalClusterDesign.⌋

[constr_9325] The attribute DoIpFunctionalClusterDesign.doIpLogicalAddress.logicalAddress shall be within the physical address boundaries defined by DoIpRequestConfiguration

Imposition time: IT_SysDes

⌈The value of the attribute DoIpFunctionalClusterDesign.doIpLogicalAddress.logicalAddress shall be within the physical address boundaries defined by DoIpRequestConfiguration that is aggregated by the DoIpFunctionalClusterDesign in the role requestConfigurationDesign.⌋

Class	**DoIpRequestConfigurationDesign**			
Note	This meta-class specifies a range of target addresses and its interpretation as either physical or functional request. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	DoIpFunctionalClusterDesign.requestConfigurationDesign			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
endAddress	PositiveInteger	0..1	attr	End address for range of target-addresses (including this address).
requestType	RequestTypeEnum	0..1	attr	Determines the type of request.
startAddress	PositiveInteger	0..1	attr	Start address for range of target-addresses (including this address).

Table 6.13: DoIpRequestConfigurationDesign

[constr_10197] Multiplicity of attribute DoIpRequestConfigurationDesign.endAddress

Imposition time: IT_MachDes

⌈For each DoIpRequestConfigurationDesign, the attribute endAddress shall exist.⌋

[constr_10198] Multiplicity of attribute DoIpRequestConfigurationDesign.requestType

Imposition time: IT_MachDes

⌈For each DoIpRequestConfigurationDesign, the attribute requestType shall exist.⌋

[constr_10199] Multiplicity of attribute DoIpRequestConfigurationDesign.startAddress

Imposition time: IT_MachDes

⌈For each DoIpRequestConfigurationDesign, the attribute startAddress shall exist.⌋

[TPS_MANI_03344] Possible interval of the attribute DoIpLogicalAddress.logicalAddress ⌈The possible interval of attribute DoIpLogicalAddress.logicalAddress is defined by means of

- DoIpRequestConfigurationDesign.startAddress
- DoIpRequestConfigurationDesign.endAddress

⌋

The DoIpRequestConfigurationDesign corresponds to Table 39 that is defined in [23, ISO-13400-2:2019].

[constr_3499] Supported value range for the primitive attribute DoIpRequestConfigurationDesign.startAddress

Imposition time: IT_MachDes

⌈The supported value range of primitive attribute DoIpRequestConfigurationDesign.startAddress is limited to the interval [0..65535].⌋

[constr_5000] Supported value range for the primitive attribute DoIpRequestConfigurationDesign.endAddress

Imposition time: IT_MachDes

⌈The supported value range of primitive attribute DoIpRequestConfigurationDesign.endAddress is limited to the interval [0..65535].⌋

[constr_9344] Supported value range for the attribute DoIpFunctionalClusterDesign.doIpProtocolVersion

Imposition time: IT_SysDes

⌈The supported value range of attribute DoIpFunctionalClusterDesign.doIpProtocolVersion is limited to the interval [2..4].⌋

[constr_10186] Multiplicity of attribute DoIpFunctionalClusterDesign.entityStatusMaxByteFieldUse

Imposition time: IT_SysDes

⌈For each DoIpFunctionalClusterDesign, the attribute entityStatusMaxByteFieldUse shall exist.⌋

[constr_10189] Multiplicity of attribute DoIpFunctionalClusterDesign.maxRequestBytes

Imposition time: IT_SysDes

⌈For each DoIpFunctionalClusterDesign, the attribute maxRequestBytes shall exist.⌋

[constr_3495] Supported value range for attribute DoIpFunctionalClusterDesign.eid

Imposition time: IT_SysDes

⌈The supported value range of attribute DoIpFunctionalClusterDesign.eid is limited to the interval [0..281474976710655].⌋

[constr_3497] Supported value range for attribute DoIpFunctionalClusterDesign.maxRequestBytes

Imposition time: IT_SysDes

⌈The supported value range of attribute DoIpFunctionalClusterDesign.maxRequestBytes is limited to the interval [0..4294967295].⌋

[constr_9356] Upper multiplicity of reference in the role DoIpNetworkConfigurationDesign.networkConfiguration

Imposition time: IT_SysDes

⌈In the context of DoIpNetworkConfigurationDesign, the reference in the role networkConfiguration shall exist at most once.⌋

[constr_10191] Multiplicity of attribute DoIpNetworkConfigurationDesign.isActivationLineDependent

Imposition time: IT_MachDes

⌈For each DoIpNetworkConfigurationDesign, the attribute isActivationLineDependent shall exist.⌋

[constr_10192] Multiplicity of attribute DoIpNetworkConfigurationDesign.maxInitialVehicleAnnouncementTime

Imposition time: IT_MachDes

⌈For each DoIpNetworkConfigurationDesign, the attribute maxInitialVehicleAnnouncementTime shall exist.⌋

[constr_10193] Multiplicity of attribute DoIpNetworkConfigurationDesign.maxTesterConnections

Imposition time: IT_MachDes

⌈For each DoIpNetworkConfigurationDesign, the attribute maxTesterConnections shall exist.⌋

[constr_10194] Multiplicity of attribute DoIpNetworkConfigurationDesign.networkInterfaceId

Imposition time: IT_MachDes

⌈For each DoIpNetworkConfigurationDesign, the attribute networkInterfaceId shall exist.⌋

It is possible to define several networkInterfaces in a DoIpFunctionalClusterDesign. For each network connection, individual configuration settings can be set with the attributes that are defined in the DoIpNetworkConfigurationDesign element, e.g. it is possible to configure the vehicle announcement for different network connections differently.

[TPS_MANI_03294] Semantics of DoIpNetworkConfigurationDesign.eidRetrieval ⌈If DoIpNetworkConfigurationDesign.eidRetrieval is set to:

- eidUseConfigValue, then take the value from DoIpFunctionalClusterDesign.eid
- eidUseApi, then DoIpFunctionalClusterDesign.eid shall be ignored and the API DiagnosticDoIPEntityIdentificationInterface shall be used to retrieve the eid
- eidUseMac, then DoIpFunctionalClusterDesign.eid shall be ignored and the MAC shall be taken as the eid

⌋

[constr_5343] Usage of DoIpNetworkConfigurationDesign.eidRetrieval

Imposition time: IT_Mani

⌈If DoIpNetworkConfigurationDesign.eidRetrieval is set to eidUseConfigValue then DoIpFunctionalClusterDesign.eid shall exist and a value shall be assigned to it.⌋

[TPS_MANI_03333] Semantics of DoIpNetworkConfigurationDesign.tpConnection ⌈The purpose of tpConnection is to define a DoIP communication connection between the DoIP entity that is defined by the DoIpFunctionalClusterDesign.doIpLogicalAddress and between one or several testers that are defined by the source address range (defined by sourceAddressRangeStart and sourceAddressRangeEnd).⌋

Class	GenericTpConnection
Note	A TpConnection that defines a communication between the logical target address of the DoIP Entitiy and the source address range. **Tags:** atp.recommendedPackage=GenericTpConnections This Class is only used by the AUTOSAR Adaptive Platform.

▽

△

Class	GenericTpConnection			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadableDesignElement, UploadablePackageElement			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
priority	PositiveInteger	0..1	attr	This attribute defines the priority for a particular communication connection. For more details see the SWS Diagnostics specification.
sourceAddressRangeEnd	PositiveInteger	0..1	attr	The DoIP source address defines the logical address of the tester that starts the communication. The GenericTpConnection is able to define a source address range (without gaps) with the two attributes sourceAddressRangeStart and sourceAddressRangeEnd. This attribute defines the end of the range.
sourceAddressRangeStart	PositiveInteger	0..1	attr	The DoIP source address defines the logical address of the tester that starts the communication. The GenericTpConnection is able to define a source address range (without gaps) with the two attributes sourceAddressRangeStart and sourceAddressRangeEnd. This attribute defines the start of the range.

Table 6.14: GenericTpConnection

[constr_9334] Supported value range for attribute `GenericTpConnection.sourceAddressRangeStart`

Imposition time: `IT_Mani`

⌈The supported value range of attribute `GenericTpConnection.sourceAddressRangeStart` is limited to the interval 0..65535.⌋

[constr_9335] Supported value range for attribute `GenericTpConnection.sourceAddressRangeEnd`

Imposition time: `IT_Mani`

⌈The supported value range of attribute `GenericTpConnection.sourceAddressRangeEnd` is limited to the interval 0..65535.⌋

[constr_9338] `GenericTpConnection.sourceAddressRangeStart` shall be less or equal of `GenericTpConnection.sourceAddressRangeEnd`

Imposition time: `IT_Mani`

⌈The value of`GenericTpConnection.sourceAddressRangeStart` shall be less or equal the value of `GenericTpConnection.sourceAddressRangeEnd`.⌋

[constr_9336] `DoIpNetworkConfigurationDesign` shall reference a `GenericTpConnection`

Imposition time: `IT_SysDes`

⌈Each `DoIpNetworkConfigurationDesign` element shall reference at least one `GenericTpConnection` in the role `tpConnection`.⌋

[constr_9337] Same `priority` for overlapping address ranges in the context of the same `MachineDesign`

Imposition time: `IT_SysDes`

⌈All `GenericTpConnection`s referenced in the scope of the same `MachineDesign` that have an overlapping range defined by `sourceAddressRangeStart` and `sourceAddressRangeEnd` shall also define the identical values of attribute `priority`.⌋

6.4 Ethernet Endpoint Configuration for Platform Modules

The `PlatformModuleEthernetEndpointConfiguration` allows the configuration of a socket address consisting of a TCP/UDP port and an IP address for use in the Functional Cluster Design.

Class	**PlatformModuleEthernetEndpointConfiguration**			
Note	This meta-class defines the attributes for the configuration of a port, protocol type and IP address (local address) of the communication on a VLAN. **Tags:** atp.recommendedPackage=PlatformModuleEndpointConfigurations This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PlatformModuleEndpointConfiguration*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Connector	EthernetCommunication Connector	0..1	ref	Reference to the CommunicationConnector (VLAN) for which the network configuration is defined. **Tags:** atp.Splitkey=communicationConnector
remoteConfig	RemoteEndpoint Configuration	*	aggr	Defintion of remote addresses of peers. **Tags:** atp.Splitkey=remoteConfig
secureCom PropsForTcp	SecureComProps	0..1	ref	Reference to communication security configuration settings that are valid for the tcp unicast endpoint (Tcp Port + unicast IP Address) defined by the PlatformModule EthernetEndpointConfiguration. **Tags:** atp.Splitkey=secureComPropsForTcp
secureCom PropsForUdp	SecureComProps	0..1	ref	Reference to communication security configuration settings that are valid for the udp unicast endpoint (Udp Port + unicast IP Address) defined by the PlatformModule EthernetEndpointConfiguration. **Tags:** atp.Splitkey=secureComPropsForUdp
tcpPort	ApApplicationEndpoint	0..1	ref	This reference allows to configure a tcp port number. **Tags:** atp.Splitkey=tcpPort
udpPort	ApApplicationEndpoint	0..1	ref	This reference allows to configure a udp port number. **Tags:** atp.Splitkey=udpPort

Table 6.15: PlatformModuleEthernetEndpointConfiguration

[constr_3414] Allowed use of attributes of meta-class PlatformModuleEthernetEndpointConfiguration for the configuration of network communication in the Functional Cluster Design.

Status: DRAFT

Imposition time: IT_Mani

⌈

	Element		
PlatformModuleEthernetEndpointConfiguration attributes	**Usage in DoIpFunctionalClusterDesign**	**Usage in IdsFunctionalClusterDesign**	**Usage in DltLogSink**
tcpPort	Optional	Optional	Optional
udpPort	Optional	Optional	Optional
communicationConnector	Mandatory	Mandatory	Mandatory
remoteConfig	N/A	Mandatory	Optional
secureComPropsForUdp	N/A	Optional	Optional
secureComPropsForTcp	Optional	Optional	Optional

⌋

[constr_5372] SecureComProps for a PlatformModuleEthernetEndpointConfiguration that contains a UDP configuration

Imposition time: IT_Mani

⌈Only a PlatformModuleEthernetEndpointConfiguration that references an ApApplicationEndpoint in the role udpPort is allowed to reference SecureComProps in the role secureComPropsForUdp.⌋

[constr_5373] SecureComProps for a PlatformModuleEthernetEndpointConfiguration that contains a TCP configuration

Imposition time: IT_Mani

⌈Only a PlatformModuleEthernetEndpointConfiguration that references an ApApplicationEndpoint in the role tcpPort is allowed to reference SecureComProps in the role secureComPropsForTcp.⌋

Class	**RemoteEndpointConfiguration**			
Note	This meta-class is used to define the IP address and port of a peer. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	PlatformModuleEthernetEndpointConfiguration.remoteConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ipv4Address	Ip4AddressString	0..1	attr	remote Unicast or Multicast IPv4 Address
ipv6Address	Ip6AddressString	0..1	attr	remote Unicast or Multicast IPv6 Address
tcpPort	PositiveInteger	0..1	attr	remote tcpPort
udpPort	PositiveInteger	0..1	attr	remote udpPort

Table 6.16: RemoteEndpointConfiguration

[constr_9352] Mutually exclusive existence of attributes `RemoteEndpointConfiguration.tcpPort` and `RemoteEndpointConfiguration.udpPort`

Imposition time: `IT_Mani`

⌈For each `RemoteEndpointConfiguration`, either the attribute `udpPort` or the attribute `tcpPort` shall exist.⌋

[constr_9353] Mutually exclusive existence of attributes `RemoteEndpointConfiguration.ipv4Address` and `RemoteEndpointConfiguration.ipv6Address`

Imposition time: `IT_Mani`

⌈For each `RemoteEndpointConfiguration`, either the attribute `ipv4Address` or the attribute `ipv6Address` shall exist.⌋

6.5 Network Design aspects of Machine Design

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`.

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`. Details of `EthernetCluster` are given in Chapter 7.2.1.

Class	**MachineDesign**			
Note	This meta-class represents the ability to define requirements on a Machine in the context of designing a system. **Tags:** atp.recommendedPackage=MachineDesigns This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element, *AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
accessControl	AccessControlEnum	0..1	attr	This attribute defines how the access restriction to the Service Instance is defined.
clientIdRange	ClientIdRange	0..1	aggr	Optional restriction of Client Identifiers for this Machine to an allowed range of numerical values. **Tags:** atp.Splitkey=clientIdRange
communicationConnector	CommunicationConnector	*	aggr	This aggregation defines the network connection of the machine. **Tags:** atp.Splitkey=communicationConnector, communicationConnector.shortName

▽

△

Class	**MachineDesign**			
communication Controller	Communication Controller	*	aggr	CommunicationControllers of the Machine that are used for description of 10-Base-T1S topologies **Tags:** atp.Splitkey=communicationController, communicationController.shortName
cryptoKeySlot	CryptoKeySlotDesign	*	aggr	This aggregation represents the key slots for which a key slot design is created in the context of the enclosing machine design. **Tags:** atp.Splitkey=cryptoKeySlot, cryptoKeySlot.short Name
ethIpProps	EthIpProps	*	ref	Machine specific IP attributes. **Tags:** atp.Splitkey=ethIpProps
functional ClusterDesign	AbstractFunctional ClusterDesign	*	aggr	Configuration settings for Functional Clusters on the machine design level. **Tags:** atp.Splitkey=functionalClusterDesign, functional ClusterDesign.shortName
pncPrepare SleepTimer	TimeValue	0..1	attr	Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP.
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.
processor	Processor	*	aggr	This represents the collection of processors owned by the enclosing MachineDesign. **Tags:** atp.Splitkey=processor, processor.shortName
service Discovery Config	ServiceDiscovery Configuration	*	aggr	Set of service discovery configuration settings that are defined on the machine for individual Communication Connectors. **Tags:** atp.Splitkey=serviceDiscoveryConfig
tcpIpIcmpProps	EthTcpIpIcmpProps	*	ref	Machine specific ICMP (Internet Control Message Protocol) attributes **Tags:** atp.Splitkey=tcpIpIcmpProps
tcpIpProps	EthTcpIpProps	*	ref	Machine specific TcpIp Stack attributes. **Tags:** atp.Splitkey=tcpIpProps

Table 6.17: MachineDesign

Class	«atpVariation» ***CommunicationController*** (abstract)			
Note	The communication controller is a dedicated hardware device by means of which hosts are sending frames to and receiving frames from the communication medium. **Tags:** vh.latestBindingTime=postBuild			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractCanCommunicationController*, EthernetCommunicationController, FlexrayCommunication Controller, UserDefinedCommunicationController			
Aggregated by	EcuInstance.commController, MachineDesign.communicationController			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	«atpVariation» **CommunicationController** (abstract)			
wakeUpBy Controller Supported	Boolean	0..1	attr	Defines whether the ECU shall be woken up by this CommunicationController. TRUE: wake up is possible FALSE: wake up is not supported Note: If wakeUpByControllerSupported is set to TRUE the feature shall be supported by both hardware and basic software.

Table 6.18: CommunicationController

[constr_3723] Upper multiplicity of reference in the role MachineDesign.tcpIpProps

Imposition time: IT_SysDes

⌈In the context of MachineDesign, the reference in the role tcpIpProps shall exist at most once.⌋

[constr_3724] Upper multiplicity of reference in the role MachineDesign.tcpIpIcmpProps

Imposition time: IT_SysDes

⌈In the context of MachineDesign, the reference in the role tcpIpIcmpProps shall exist at most once.⌋

[constr_3725] Upper multiplicity of reference in the role MachineDesign.ethIpProps

Imposition time: IT_SysDes

⌈In the context of MachineDesign, the reference in the role ethIpProps shall exist at most once.⌋

Class	**PhysicalChannel** (abstract)			
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			
Aggregated by	*CommunicationCluster*.physicalChannel			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 6.19: PhysicalChannel

Class	«atpVariation» **EthernetCluster**			
Note	Ethernet-specific cluster attributes. **Tags:** atp.recommendedPackage=CommunicationClusters			
Base	*ARElement*, *ARObject*, *CollectableElement*, *CommunicationCluster*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *Uploadable PackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
couplingPort Connection	CouplingPort Connection	*	aggr	Specification of connections between CouplingElements and EcuInstances. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=couplingPortConnection, couplingPort Connection.variationPoint.shortLabel 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). **Tags:** atp.Splitkey=macMulticastGroup, macMulticast Group.shortName

Table 6.20: EthernetCluster

Class	**MacMulticastGroup**			
Note	Per EthernetCluster globally defined MacMulticastGroup. One sender can handle many receivers simultaneously if the receivers have all the same macMulticastAddress. The addresses need to be unique for the particular EthernetCluster.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EthernetCluster.macMulticastGroup			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
macMulticast Address	MacAddressString	0..1	attr	A multicast MAC address (Media Access Control address) is a identifier for a group of hosts in a network.

Table 6.21: MacMulticastGroup

Enumeration	**AccessControlEnum**
Note	This enumeration describes the options for the definition of access restriction to resources. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	MachineDesign.accessControl
Literal	**Description**
custom	The access restriction to the resource is defined by a non-AUTOSAR process. **Tags:** atp.EnumerationLiteralIndex=1
modeled	The access restriction to the resource is modeled in the AUTOSAR Application Design model or the AUTOSAR target-configuration model. **Tags:** atp.EnumerationLiteralIndex=0

Table 6.22: AccessControlEnum

[TPS_MANI_03209] The meaning of `MachineDesign.accessControl`

Status: DRAFT

⌈The `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 custom lists that are created by a non-AUTOSAR process.⌋

[constr_3320] Aggregation of `CommunicationConnector` by `MachineDesign`

Imposition time: `IT_MachDes`

⌈Meta-class `MachineDesign` shall only aggregate `EthernetCommunicationConnector`s in the role `communicationConnector`. No other subclass of `CommunicationConnector` shall appear in this aggregation.⌋

[constr_5324] `MachineDesign.communicationController` aggregation restriction

Imposition time: `IT_MachDes`

⌈`MachineDesign` is only allowed to aggregate an `EthernetCommunicationController` in the role `communicationController`.⌋

6.6 Service Discovery Configuration

Service Discovery messages are exchanged between network nodes to announce and 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)			
Note	Service Discovery configuration settings for the middleware transport layer. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Subclasses	SomeipServiceDiscovery			
Aggregated by	MachineDesign.serviceDiscoveryConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 6.23: ServiceDiscoveryConfiguration

6.6.1 SOME/IP Service Discovery Configuration

[TPS_MANI_03064] SOME/IP Service Discovery message exchange configuration ⌈`ProvidedServiceInstance`s on the *AUTOSAR classic platform* and

ProvidedSomeipServiceInstance on the *AUTOSAR adaptive platform* 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).⌋

[constr_5045] Only one SomeipServiceDiscovery configuration per VLAN is allowed

Imposition time: IT_SysDes

⌈Only a single NetworkEndpoint on an EthernetPhysicalChannel (VLAN) is allowed to be referenced by a SomeipServiceDiscovery element in the role multicastSdIpAddress.⌋

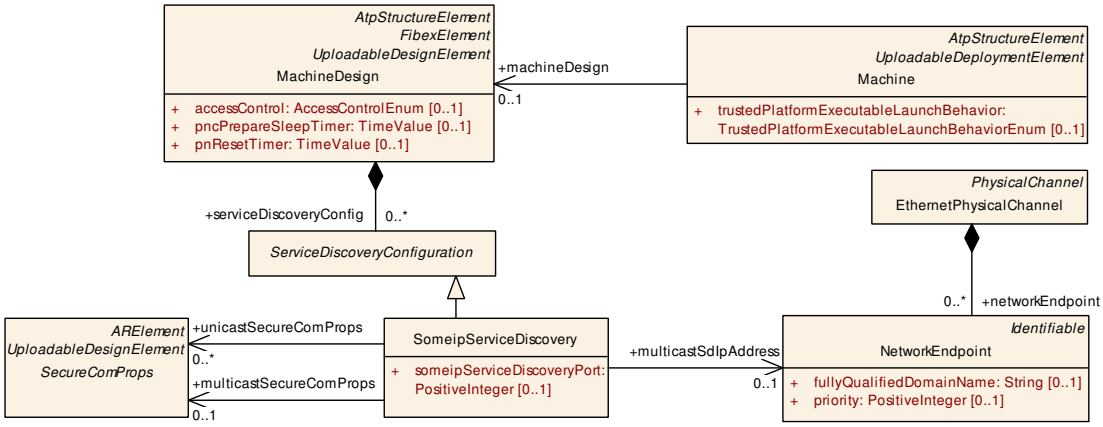


Figure 6.6: SOME/IP Service Discovery Configuration

The SomeipServiceDiscovery can reference SecureComProps to define and configure a security protocol that provides 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.

Class	**SomeipServiceDiscovery**			
Note	This meta-class represents a specialization of the generic service discovery for the SOME/IP case. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *ServiceDiscoveryConfiguration*			
Aggregated by	MachineDesign.serviceDiscoveryConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**SomeipServiceDiscovery**			
multicastSdIp Address	NetworkEndpoint	0..1	ref	This reference identifies the multicast IP address used for service discovery. **Tags:** atp.Splitkey=multicastSdIpAddress
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.Splitkey=multicastSecureComProps
someipService DiscoveryPort	PositiveInteger	0..1	attr	This attribute represents the port number reserved for service discovery.
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.Splitkey=unicastSecureComProps

Table 6.24: SomeipServiceDiscovery

[constr_10167] Multiplicity of attribute SomeipServiceDiscovery.someip-ServiceDiscoveryPort

Imposition time: IT_SysDes

⌈For each SomeipServiceDiscovery, the attribute someipServiceDiscovery-Port shall exist.⌋

6.7 TcpIp stack configuration properties

The MachineDesign references the following elements and allows setting 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

The [4, TPS System Template] defines constraints for the usage of EthIpProps, EthTcpIpProps, EthTcpIpIcmpProps. These constraints are also valid if the MachineDesign references these elements.

7 System Design

7.1 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 therefore has to cover both platform target-configuration 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 indicating 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 system design aspects have in common that they have to cope with both AUTOSAR classic ECUs and adaptive Machines. The basic design aspects of such interdisciplinary systems have to be available in the AUTOSAR classic modeling approach already, otherwise they would not be available to both platforms.

Thus, it is straightforward to take the existing meta-class System as the starting point for the modeling of such mixed systems.

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.

[TPS_MANI_01274] Standardized System Category Definitions for the Adaptive Platform ⌈

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.

⌋

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 uses 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 category SYSTEM_DESIGN_DESCRIPTION, and a definition of the subsystem is provided.

CAUTION

The constraints marked with the imposition time IT_SysDes defined in chapter [4, TPS System Template] 3.1.3 *"Communication Connector"* and chapter [4, TPS System Template] 3.2 *"Communication Clustering"* are valid for this specification as well.

[constr_3421] Fibex elements applicable for a System of category MACHINE_DESIGN_EXTRACT

Imposition time: IT_SysDes

⌈A System with the category MACHINE_DESIGN_EXTRACT is allowed to reference the following fibexElements:

- CommunicationCluster
- MachineDesign
- GlobalTimeDomain
- NmConfig

⌋

[constr_10652] SystemMappings applicable for a System of category MACHINE_DESIGN_EXTRACT

Imposition time: IT_SysDes

⌈A System with the category MACHINE_DESIGN_EXTRACT is only allowed to aggregate a SystemMapping in the role mapping that in turn only aggregates one or more PncMappings in the role pncMapping.

The aggregation of **anything besides** PncMapping shall be ignored.⌋

Class	**System**			
Note	The top level element of the System Description. **Tags:** atp.recommendedPackage=Systems			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element, *AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	System			
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:** atp.Splitkey=fibexElement, fibexElement.fibexElement, fibexElement.variationPoint.shortLabel vh.latestBindingTime=postBuild
mapping	SystemMapping	*	aggr	Aggregation of all mapping aspects relevant in the System Description. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=mapping, mapping.shortName, mapping.variationPoint.shortLabel vh.latestBindingTime=postBuild
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.
systemCom SpecDefinition	SystemComSpec DefinitionSet	*	ref	Reference to the set of ComSpec definitions that are used for inter-ECU communication in the System. **Tags:** atp.Splitkey=systemComSpecDefinition
systemVersion	RevisionLabelString	0..1	attr	Version number of the System Description.

Table 7.1: System

7.2 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.

On the *AUTOSAR classic platform*, the element `EcuInstance` is used to define one ECU in the system design.

On the *AUTOSAR adaptive platform*, the element `MachineDesign` is an entity that represents a specific ECU design with dedicated configurations for e.g. `Processor`s, `ProcessorCore`s.

The meta-class `Machine` represents a model entity that is not in the focus of communication designers and should not be used during system design.

In response to this restriction, meta-class MachineDesign has been introduced, see Chapter 6.

7.2.1 Network connection

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.

[constr_3798] Applicable attributes for an Ethernet node on the adaptive platform

Imposition time: IT_SysDes

⌈If a CouplingPort is aggregated by an EthernetCommunicationController and that EthernetCommunicationController is in turn aggregated by a MachineDesign, then the attributes defined in the column “Endpoint” of [TPS_SYST_02422] are applicable.⌋

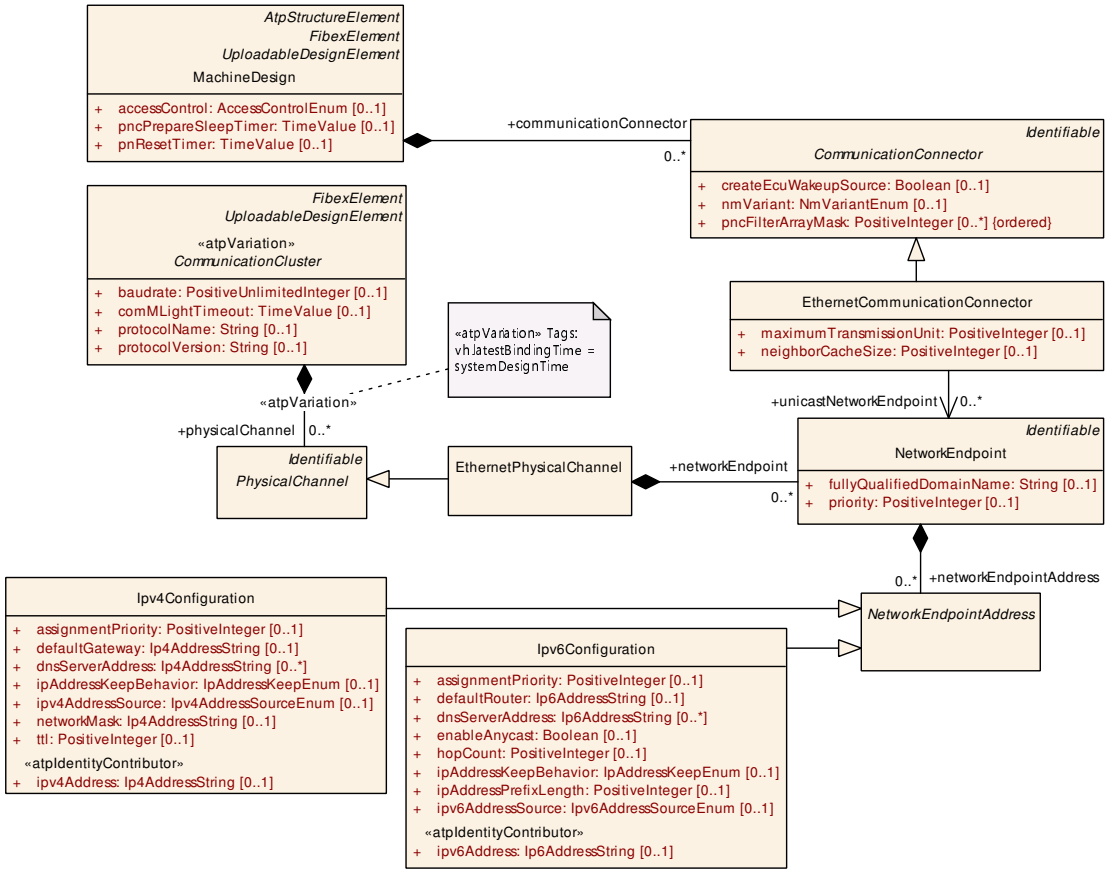


Figure 7.1: Network connection of a MachineDesign

Class	NetworkEndpoint			
Note	The network endpoint defines the network addressing (e.g. IP-Address or MAC multicast address).			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EthernetPhysicalChannel.networkEndpoint			
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. **Tags:** atp.Splitkey=ipSecConfig
network Endpoint Address	NetworkEndpoint Address	*	aggr	Definition of a Network Address. **Tags:** atp.Splitkey=networkEndpointAddress, networkEndpoint Address.ipv4Address, networkEndpointAddress.ipv6 Address, networkEndpointAddress.macMulticastGroup xml.namePlural=NETWORK-ENDPOINT-ADDRESSES
priority	PositiveInteger	0..1	attr	Defines the frame priority where values from 0 (best effort) to 7 (highest) are allowed.

Table 7.2: NetworkEndpoint

More precisely, the particular IP address is configured by means of the aggregation of `Ipv4Configuration` or `Ipv6Configuration` in the role `networkEndpoint-Address`.

The `NetworkEndpoint` is aggregated by the `EthernetPhysicalChannel` that in turn is aggregated by the `EthernetCluster`.

[TPS_MANI_03052] Static IPv4 configuration ⌈If the value of attribute `ipv4 AddressSource` of meta-class `Ipv4Configuration` is set to `Ipv4Address-SourceEnum.fixed`, then the `ipv4Address` defines the static IPv4 address.⌋

Class	Ipv4Configuration			
Note	Internet Protocol version 4 (IPv4) configuration.			
Base	*ARObject*, *NetworkEndpointAddress*			
Aggregated by	NetworkEndpoint.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. **Stereotypes:** atpIdentityContributor

▽

△

Class	Ipv4Configuration			
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 7.3: Ipv4Configuration

Primitive	Ip4AddressString
Note	This is used to specify an IP4 address. Notation: 255.255.255.255 **Tags:** xml.xsd.customType=IP4-ADDRESS-STRING xml.xsd.pattern=(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)|ANY xml.xsd.type=string

Table 7.4: Ip4AddressString

Enumeration	Ipv4AddressSourceEnum
Note	Defines how the node obtains its IPv4-Address.
Aggregated by	Ipv4Configuration.ipv4AddressSource
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 xml.name=AUTO-IP--DOIP
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 7.5: Ipv4AddressSourceEnum

[TPS_MANI_03053] Static IPv6 configuration ⌈If the value of attribute `ipv6AddressSource` of meta-class `Ipv6Configuration` is set to `Ipv6AddressSourceEnum.fixed`, then the `ipv6Address` defines the static IPv6 address.⌋

Class	Ipv6Configuration			
Note	Internet Protocol version 6 (IPv6) configuration.			
Base	*ARObject*, *NetworkEndpointAddress*			
Aggregated by	NetworkEndpoint.networkEndpointAddress			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**Ipv6Configuration**			
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. **Stereotypes:** atpIdentityContributor
ipv6Address Source	Ipv6AddressSource Enum	0..1	attr	Defines how the node obtains its IP address.

Table 7.6: Ipv6Configuration

Primitive	**Ip6AddressString**
Note	This is used to specify an IP6 address. Notation: FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF Alternative notations, short-cuts with duplicate colons like ::, etc. or mixtures using colons and dots, are not allowed. **Tags:** xml.xsd.customType=IP6-ADDRESS-STRING xml.xsd.pattern=[0-9A-Fa-f]{1,4}(:[0-9A-Fa-f]{1,4}){7,7}|ANY xml.xsd.type=string

Table 7.7: Ip6AddressString

Enumeration	**Ipv6AddressSourceEnum**
Note	Defines how the node obtains its IPv6-Address.
Aggregated by	Ipv6Configuration.ipv6AddressSource
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

▽

△

Enumeration	**Ipv6AddressSourceEnum**
linkLocal_doip	Linklocal IPv6 Address Assignment using DoIP Parameters **Tags:** atp.EnumerationLiteralIndex=3 xml.name=LINK-LOCAL--DOIP
router Advertisement	IPv6 Stateless Autoconfiguration. **Tags:** atp.EnumerationLiteralIndex=4

Table 7.8: Ipv6AddressSourceEnum

Enumeration	**IpAddressKeepEnum**
Note	Defines the behavior after a dynamic IP address has been assigned.
Aggregated by	Ipv4Configuration.ipAddressKeepBehavior, Ipv6Configuration.ipAddressKeepBehavior
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 7.9: IpAddressKeepEnum

In addition to `Ipv4Configuration` and `Ipv6Configuration`, a `MacMulticastConfiguration` can also be described as `NetworkEndpointAddress` in a system topology description.

This may be useful for describing endpoints that are not based on IP, e.g. for streaming protocols like AVB (Audio Video Bridging). However, there is no foundation software or ara::com support for such `MacMulticastConfiguration` `NetworkEndpoint`s on the *AUTOSAR adaptive platform*. For SOME/IP communication such `NetworkEndpoint`s are excluded by [constr_3288].

CAUTION

The constraints marked with the imposition time `IT_SysDes` defined in chapter [4, TPS System Template] 3.3.5 *"Ethernet"* are valid for this specification as well.

7.2.1.1 Support of 10BASE-T1S Network Topologies

10BASE-T1S network topology description is supported in the System Design with a `CouplingPortConnection` that points with the `nodePort` reference to `CouplingPort`s that represent the 10Base-T1S PHYs connected to the network.

More details about the modeling of 10BASE-T1S networks can be found in the [4, TPS System Template]. 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.

CAUTION

The constraints marked with the imposition time `IT_SysDes` defined in chapter [4, TPS System Template] 3.3.6 *"10BASE-T1S Ethernet"* are valid for this specification as well.

7.2.1.2 Support of Ethernet tunneling through CAN XL

On AUTOSAR AP, CAN XL supports only the "mapped tunneling" of Ethernet frames through the physical CAN XL network (i.e. SDU Type 0x05h as specified by the *CiA* standardization of CAN XL). Configuration of this tunneling is done in a similar way as in CP by using the category `CAN_XL` for `EthernetCommunicationConnector`.

[TPS_MANI_03668] Supported type of Ethernet tunneling through CAN XL on Adaptive Platform ⌈On the Adaptive Platform, CAN XL only supports SDU Type 0x05h (Ethernet mapped tunneling). If an AP machine receives a CAN XL frame containing an SDU Type other than 0x05h, it shall discard this frame without processing its content.⌋

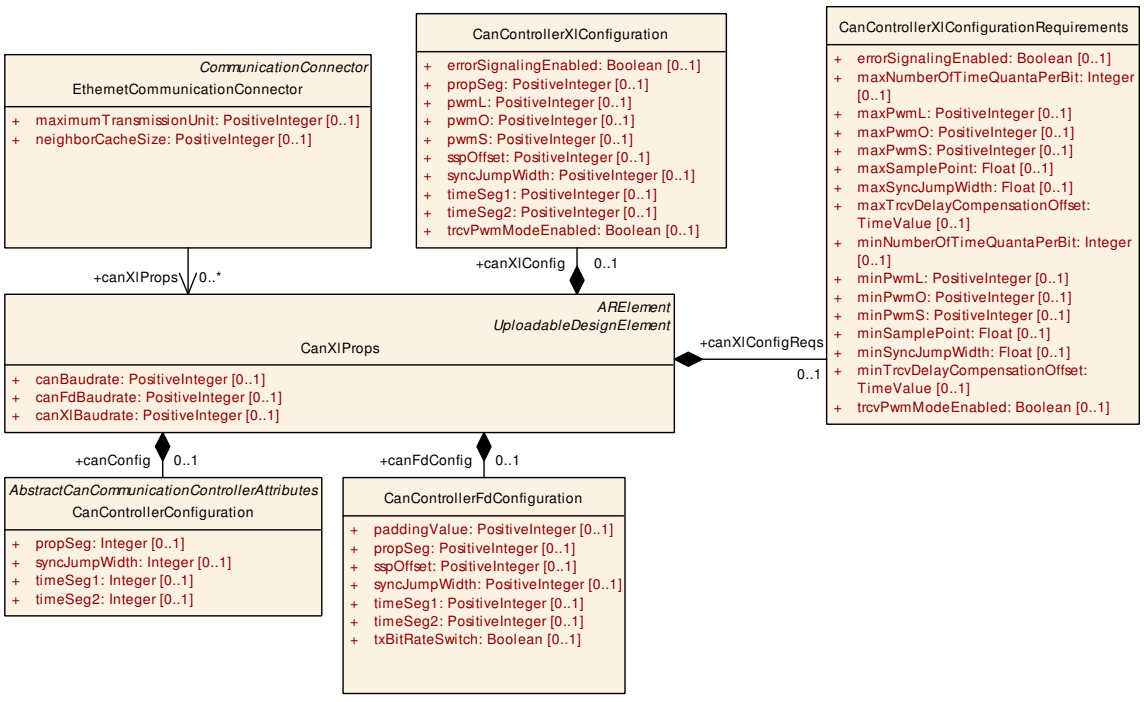


Figure 7.2: Modeling of `CanXlProps`

In other words, CAN XL frames containing (native or tunneled) Classical CAN, CAN FD or CAN XL payload cannot be meaningfully processed by an AP machine because it does not contain a CAN communication stack.

[TPS_MANI_03669] Ethernet tunneling through CAN XL ⌈If an Ethernet connection of a MachineDesign shall be realized by Ethernet frames tunneled through a CAN XL network, the EthernetCommunicationConnector aggregated by the MachineDesign shall set the value of attribute category to CAN_XL and additionally have a reference in the role canXlProps to meta-class CanXlProps which contains the required CAN XL specific configuration parameters, i.e. CanController-XlConfiguration and CanControllerXlConfigurationRequirements.⌋

Class	**CanXlProps**			
Note	This meta-class is used to configure Machine specific CAN XL attributes. **Tags:** atp.recommendedPackage=CanXlProps This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
canBaudrate	PositiveInteger	0..1	attr	Specifies the data segment CAN 2.0 baud rate of the CAN XL controller in bits/s.
canConfig	CanController Configuration	0..1	aggr	CAN 2.0 configuration parameters for the CAN XL controller. **Tags:** atp.Splitkey=canConfig
canFdBaudrate	PositiveInteger	0..1	attr	Specifies the data segment CAN FD baud rate of the CAN XL controller in bits/s.
canFdConfig	CanControllerFd Configuration	0..1	aggr	CAN FD configuration parameters for the CAN XL controller. **Tags:** atp.Splitkey=canFdConfig
canXlBaudrate	PositiveInteger	0..1	attr	Specifies the data segment CAN XL baud rate of the CAN XL controller in bits/s.
canXlConfig	CanControllerXl Configuration	0..1	aggr	CAN XL configuration parameters for the CAN XL controller. **Tags:** atp.Splitkey=canXlConfig
canXlConfig Reqs	CanControllerXl Configuration Requirements	0..1	aggr	CAN XL configuration parameter requirements for the CAN XL controller. **Tags:** atp.Splitkey=canXlConfigReqs

Table 7.10: CanXlProps

[constr_3693] EthernetCommunicationConnector.category is set to CAN_XL

Imposition time: IT_SysDes

⌈If a EthernetCommunicationConnector is aggregated by the MachineDesign where attribute category is set to CAN_XL, then a reference from the Ethernet-CommunicationConnector to a CanXlProps in the role canXlProps shall exist.⌋

[constr_3694] Existence of canXlConfig vs. canXlConfigReqs

Imposition time: IT_SysDes

⌈For each CanXlProps, one of

- canXlConfig or
- canXlConfigReqs

shall exist.⌋

7.2.1.3 IP configuration properties

This chapter introduces meta-classes and attributes that can be used to configure the TCP/IP stack in the Adaptive Platform.

The Internet Protocol (IP) is the main protocol of the TCP/IP stack and is responsible for delivering datagrams from a source host identified by the source address to one or multiple destination hosts identified by the destination address.

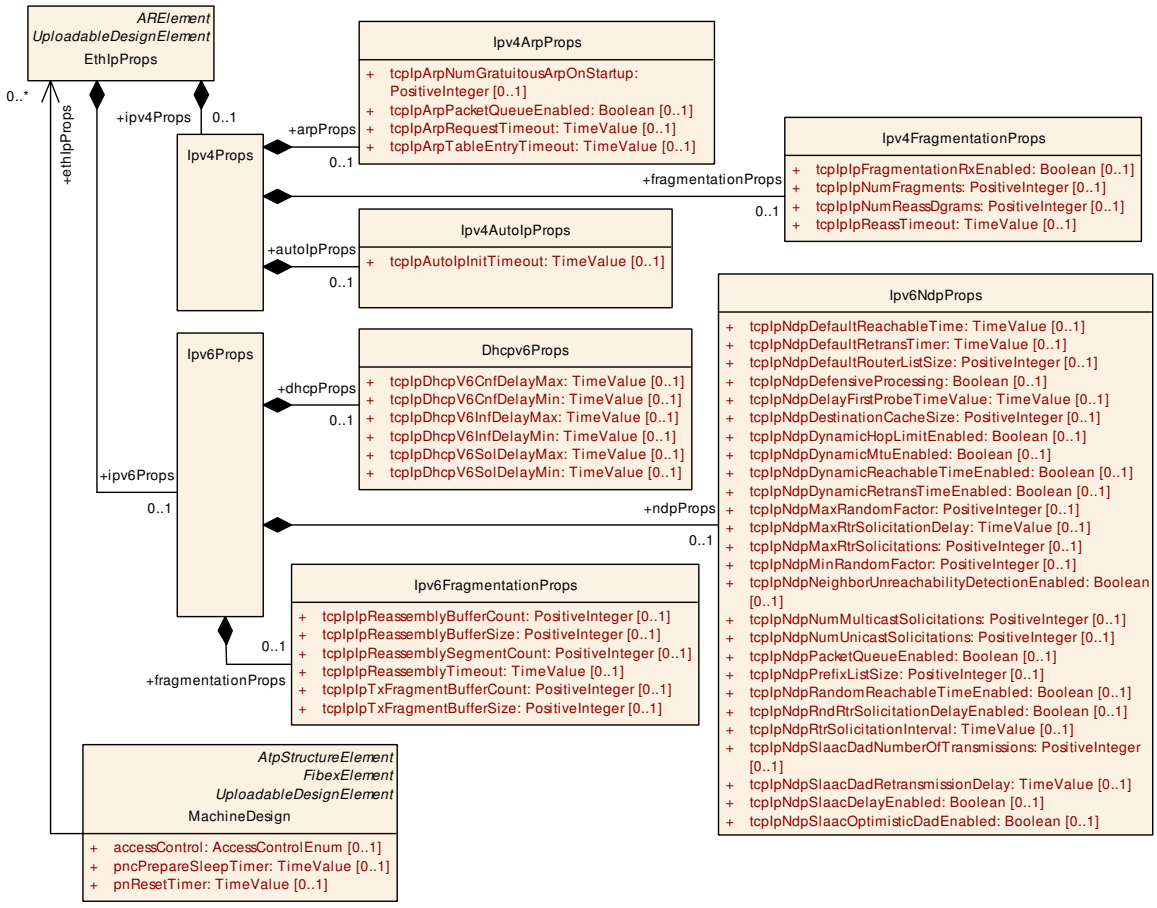


Figure 7.3: Machine specific IP configuration options

AUTOSAR supports the configuration of

- Internet Protocol of version 4 with `Ipv4Props`. This includes the configuration of
  - ARP (Address Resolution Protocol) with `Ipv4ArpProps`
  - Auto IP with `Ipv4AutoIpProps`
  - Packet fragmentation and reassembly with `Ipv4FragmentationProps`
- Internet Protocol of version 6 with `Ipv6Props`. This includes the configuration of

- NDP (Neighbor Discovery Protocol) with `Ipv6NdpProps`
- DHCP (Dynamic Host Configuration Protocol) with `Dhcpv6Props`
- Packet fragmentation and reassembly with `Ipv6FragmentationProps`

For the configuration of the Internet Protocol, the same modeling approach is used as in the Classic Platform (see [4, TPS System Template]).

CAUTION

The constraints marked with the imposition time `IT_SysDes` defined in chapter [4, TPS System Template] 3.3.5.7.1 *"IP configuration properties"* are valid for this specification as well.

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*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ipv4Props	Ipv4Props	0..1	aggr	Configuration options for IPv4. **Tags:** atp.Splitkey=ipv4Props
ipv6Props	Ipv6Props	0..1	aggr	Configuration options for IPv6. **Tags:** atp.Splitkey=ipv6Props

Table 7.11: EthIpProps

Class	**Ipv4Props**			
Note	This meta-class specifies the configuration options for IPv4.			
Base	*ARObject*			
Aggregated by	EthIpProps.ipv4Props			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arpProps	Ipv4ArpProps	0..1	aggr	Configuration properties for the ARP (Address Resolution Protocol). **Tags:** atp.Splitkey=arpProps
autoIpProps	Ipv4AutoIpProps	0..1	aggr	Configuration options for Auto-IP (automatic private IP addressing). **Tags:** atp.Splitkey=autoIpProps
fragmentation Props	Ipv4Fragmentation Props	0..1	aggr	Configuration options for IPv4 packet fragmentation/ reassembly. **Tags:** atp.Splitkey=fragmentationProps

Table 7.12: Ipv4Props

Class	**Ipv4ArpProps**			
Note	Specifies the configuration options for the ARP (Address Resolution Protocol).			
Base	*ARObject*			
Aggregated by	Ipv4Props.arpProps			
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 7.13: Ipv4ArpProps

Class	**Ipv4AutoIpProps**			
Note	Specifies the configuration options for Auto-IP (automatic private IP addressing).			
Base	*ARObject*			
Aggregated by	Ipv4Props.autoIpProps			
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 7.14: Ipv4AutoIpProps

Class	**Ipv4FragmentationProps**			
Note	Specifies the configuration options for IPv4 packet fragmentation/reassembly.			
Base	*ARObject*			
Aggregated by	Ipv4Props.fragmentationProps			
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 7.15: Ipv4FragmentationProps

Class	Ipv6Props			
Note	This meta-class specifies the configuration options for IPv6.			
Base	*ARObject*			
Aggregated by	EthIpProps.ipv6Props			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dhcpProps	Dhcpv6Props	0..1	aggr	Configuration properties for DHCPv6. **Tags:** atp.Splitkey=dhcpProps
fragmentation Props	Ipv6Fragmentation Props	0..1	aggr	Configuration properties for IPv6 packet fragmentation/ reassembly. **Tags:** atp.Splitkey=fragmentationProps
ndpProps	Ipv6NdpProps	0..1	aggr	Configuration properties for the Neighbor Discovery Protocol for IPv6. **Tags:** atp.Splitkey=ndpProps

Table 7.16: Ipv6Props

Class	Ipv6FragmentationProps			
Note	This meta-class specifies the configuration options for IPv6 packet fragmentation/reassembly.			
Base	*ARObject*			
Aggregated by	Ipv6Props.fragmentationProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
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 7.17: Ipv6FragmentationProps

Class	**Dhcpv6Props**			
Note	This meta-class specifies the configuration options for DHCPv6.			
Base	*ARObject*			
Aggregated by	Ipv6Props.dhcpProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpDhcpV6 CnfDelayMax	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.
tcpIpDhcpV6 CnfDelayMin	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 7.18: Dhcpv6Props

Class	**Ipv6NdpProps**			
Note	This meta-class specifies the configuration options for the Neighbor Discovery Protocol for IPv6.			
Base	*ARObject*			
Aggregated by	Ipv6Props.ndpProps			
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	0..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 Value	TimeValue	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.



△

Class	Ipv6NdpProps			
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.
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 7.19: Ipv6NdpProps

7.2.1.4 TCP and UDP configuration properties

The Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are transport protocols that are based on the Internet Protocol (IP) that is located on the network layer in the OSI model.

The Transport Protocol configuration is supported with the `EthTcpIpProps` meta-class and the two aggregated classes:

- `UdpProps` - used for configuration of Udp
- `TcpProps` - used for configuration of Tcp

For the configuration of TCP and UDP, the same modeling approach is used as in the Classic Platform (see [4, TPS System Template]).

CAUTION

The constraints marked with the imposition time `IT_SysDes` defined in chapter [4, TPS System Template] 3.3.5.7.2 *"TCP and UDP configuration properties"* are valid for this specification as well.

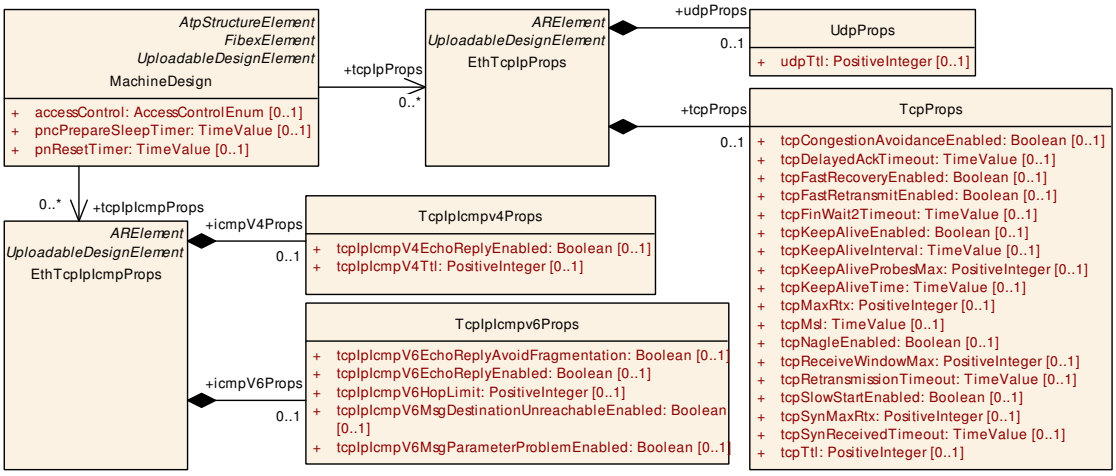


Figure 7.4: Machine specific TCP/UDP and ICMP configuration options

Class	**EthTcpIpProps**			
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*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpProps	TcpProps	0..1	aggr	TCP configuration properties **Tags:** atp.Splitkey=tcpProps
udpProps	UdpProps	0..1	aggr	UDP configuration properties **Tags:** atp.Splitkey=udpProps

Table 7.20: EthTcpIpProps

Class	UdpProps			
Note	This meta-class specifies the configuration options for UDP (User Datagram Protocol).			
Base	*ARObject*			
Aggregated by	EthTcpIpProps.udpProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
udpTtl	PositiveInteger	0..1	attr	Default Time-to-live value of outgoing UDP packets.

Table 7.21: UdpProps

Class	TcpProps			
Note	This meta-class specifies the configuration options for TCP (Transmission Control Protocol).			
Base	*ARObject*			
Aggregated by	EthTcpIpProps.tcpProps			
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.
tcpFinWait2 Timeout	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.

△

Class	TcpProps			
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 7.22: TcpProps

7.2.1.5 ICMP configuration properties

Internet Control Message Protocol (ICMP) is located in the network layer in the OSI model and is used by network devices to diagnose network communication issues, e.g. it is used to determine whether the data is reaching its intended destination in a timely manner.

AUTOSAR supports the configuration of

- the Internet Control Message Protocol version 4 for the Internet Protocol version 4 (IPv4) with `TcpIpIcmpv4Props`
- the Internet Control Message Protocol version 6 for the Internet Protocol version 6 (IPv6) with `TcpIpIcmpv6Props`

For the configuration of ICMP, the same modeling approach is used as in the Classic Platform (see [4, TPS System Template]).

CAUTION

The constraints marked with the imposition time `IT_SysDes` defined in chapter [4, TPS System Template] 3.3.5.7.3 *"ICMP configuration properties"* are valid for this specification as well.

Class	EthTcpIpIcmpProps			
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*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
icmpV4Props	TcpIpIcmpv4Props	0..1	aggr	ICMPv4 configuration properties **Tags:** atp.Splitkey=icmpV4Props
icmpV6Props	TcpIpIcmpv6Props	0..1	aggr	ICMPv6 configuration properties **Tags:** atp.Splitkey=icmpV6Props

Table 7.23: EthTcpIpIcmpProps

Class	**TcpIpIcmpv4Props**			
Note	This meta-class specifies the configuration options for ICMPv4 (Internet Control Message Protocol).			
Base	*ARObject*			
Aggregated by	EthTcpIpIcmpProps.icmpV4Props			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpIcmpV4 EchoReply 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 7.24: TcpIpIcmpv4Props

Class	**TcpIpIcmpv6Props**			
Note	This meta-class specifies the configuration options for ICMPv6 (Internet Control Message Protocol).			
Base	*ARObject*			
Aggregated by	EthTcpIpIcmpProps.icmpV6Props			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpIcmpV6 EchoReply 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.
tcpIpIcmpV6 EchoReply Enabled	Boolean	0..1	attr	This attribute enables or disables transmission of ICMP echo reply message in case of a ICMP echo reception.
tcpIpIcmpV6 HopLimit	PositiveInteger	0..1	attr	Default Hop-Limit value of outgoing ICMPv6 packets.
tcpIpIcmpV6 MsgDestination Unreachable Enabled	Boolean	0..1	attr	This attribute Enables/Disables the transmission of Destination Unreachable Messages.
tcpIpIcmpV6 MsgParameter 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 7.25: TcpIpIcmpv6Props

7.2.2 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 received datagrams are processed 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] Configuration of IPsec ⌈The `IPSecConfig` meta-class 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.⌋

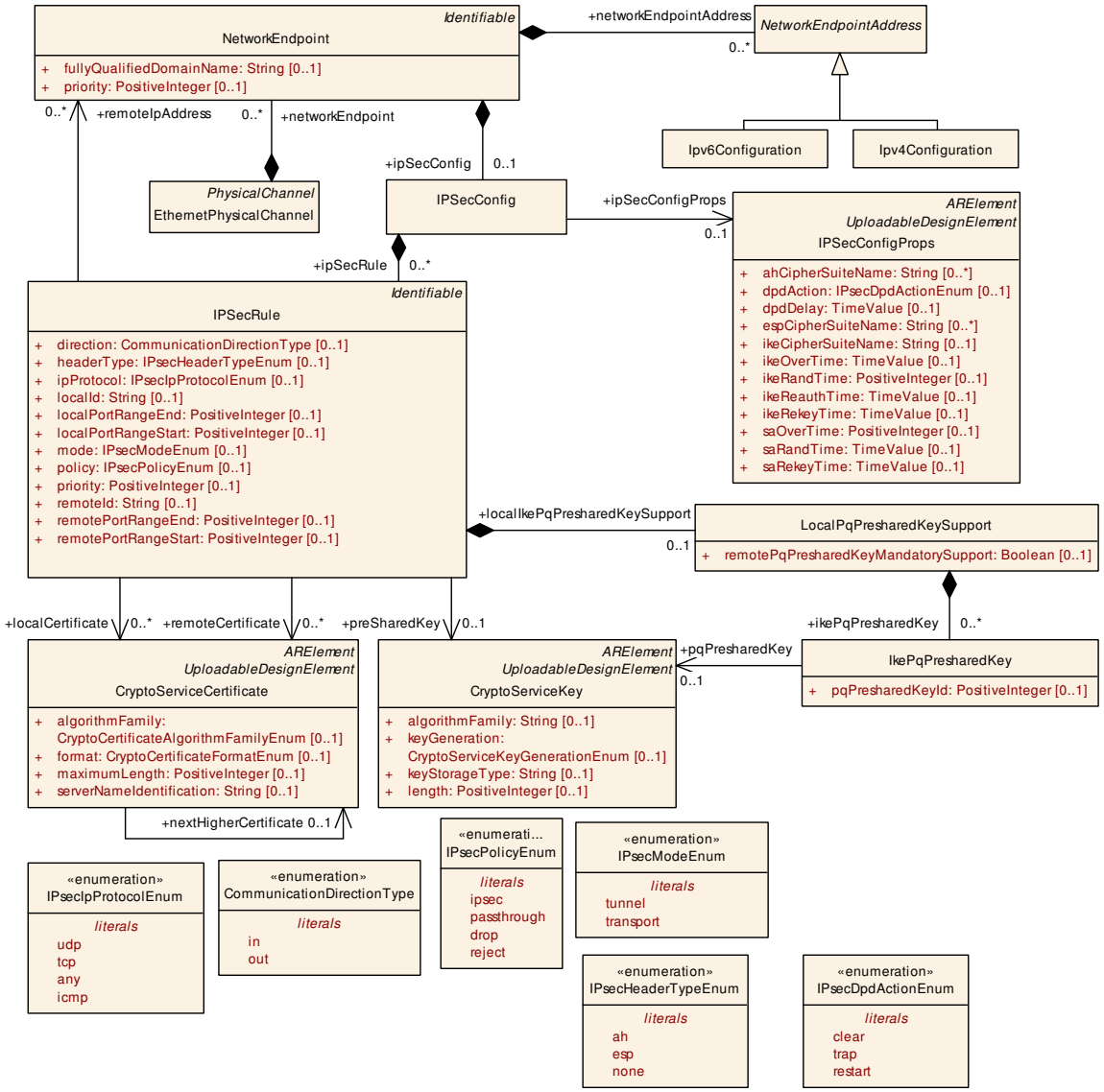


Figure 7.5: IPsec configuration model

[TPS_MANI_03204] Definition of `IPSecRule`s ⌈The `IPSecConfig` meta-class may contain one or several `IPSecRule`s. 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.⌋

Class	**IPSecConfig**			
Note	IPsec is a protocol that is designed to provide "end-to-end" cryptographically-based security for IP network connections.			
Base	*ARObject*			
Aggregated by	NetworkEndpoint.ipSecConfig			
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. **Tags:** atp.Splitkey=ipSecConfigProps
ipSecRule	IPSecRule	*	aggr	IPSec rules and filters that are defined in the IPSecConfig for a specific NetworkEndpoint. **Tags:** atp.Splitkey=ipSecRule, ipSecRule.shortName

Table 7.26: IPSecConfig

Class	**IPSecRule**			
Note	This element defines an IPsec rule that describes communication traffic that is monitored, protected and filtered.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	IPSecConfig.ipSecRule			
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. **Tags:** atp.Splitkey=localCertificate
localId	String	0..1	attr	This attribute defines how the local participant should be identified for authentication.
localIkePq PresharedKey Support	LocalPqPresharedKey Support	0..1	aggr	Post-quantum pre-shared key configuration to support IETF RFC 8784 to ensure resistance against post-quantum attacks by combining a post-quantum pre-shared key (PPK) with the exchanged information (via e.g., DH/ECDH) during key derivation. **Tags:** atp.Splitkey=localIkePqPresharedKeySupport
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.

▽

△

Class	**IPSecRule**			
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 cryptographic key used for authentication. **Tags:** atp.Splitkey=preSharedKey
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".
remote Certificate	CryptoService Certificate	*	ref	This reference identifies the applicable certificate used for a remote authentication. **Tags:** atp.Splitkey=remoteCertificate
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. **Tags:** atp.Splitkey=remoteIpAddress
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 7.27: IPSecRule

Class	**IPSecConfigProps**
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*, *UploadableDesignElement*, *UploadablePackageElement*

▽

△

Class	**IPSecConfigProps**			
Aggregated by	ARPackage.element			
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.
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 7.28: IPSecConfigProps

Enumeration	**IPsecIpProtocolEnum**
Note	Definition of supported TcpIp protocols that are supported in Security Policy Database (SPD) entries in IPSec configurations.
Aggregated by	IPSecRule.ipProtocol
Literal	***Description***

▽

△

Enumeration	IPsecIpProtocolEnum
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 7.29: IPsecIpProtocolEnum

[constr_5102] Usage of remote port ranges in `IPSecRule` is not allowed

Imposition time: `IT_SysDes`

⌈`IPSecRule.remotePortRangeStart` and `IPSecRule.remotePortRangeEnd` shall always be set to the same value.⌋

[constr_5103] Usage of local port ranges in `IPSecRule` is not allowed

Imposition time: `IT_SysDes`

⌈`IPSecRule.localPortRangeStart` and `IPSecRule.localPortRangeEnd` shall always be set to the same value.⌋

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] Definition of general IPsec configuration settings ⌈General configuration properties that are independent of particular `IPSecRule`s are collected in the `IPSecConfigProps` element that is referenced from the `IPSecConfig` in the role `ipSecConfigProps`.⌋

[TPS_MANI_03205] IPsec policy ⌈The `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.⌋

Enumeration	IPsecPolicyEnum
Note	Defines the filter actions that are supported by IPsec.
Aggregated by	IPSecRule.policy
Literal	***Description***
drop	Signifying that packets should be discarded **Tags:** atp.EnumerationLiteralIndex=3
ipsec	Signifying that packets should be protected. **Tags:** atp.EnumerationLiteralIndex=1

▽

△

Enumeration	**IPsecPolicyEnum**
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 7.30: 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 those 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] IPsec mode ⌈The `IPSecRule.mode` attribute defines whether the IP packet is processed in the `transport` or `tunnel` mode.⌋

AUTOSAR currently supports only the `transport mode` as configuration option.

Enumeration	**IPsecModeEnum**
Note	This enumeration describes the supported IPSec modes.
Aggregated by	IPSecRule.mode
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 7.31: 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] IPsec AH and ESP protocol configuration ⌈In 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.⌋

Enumeration	**IPsecHeaderTypeEnum**
Note	IPsec Header Type options
Aggregated by	IPSecRule.headerType
Literal	***Description***
ah	Authentication Header (AH) **Tags:** atp.EnumerationLiteralIndex=0
esp	Encapsulating Security Payloads (ESP) **Tags:** atp.EnumerationLiteralIndex=1
none	No header **Tags:** atp.EnumerationLiteralIndex=2

Table 7.32: IPsecHeaderTypeEnum

[TPS_MANI_03234] IPsec AH and ESP CipherSuites ⌈The attributes ahCipherSuiteName and espCipherSuiteName define the supported AH and ESP algorithms.⌋

The naming convention for ahCipherSuiteName, espCipherSuiteName and IPSecConfigProps.ikeCipherSuiteName shall follow the naming convention for cryptographic primitives that is defined in [18, SWS Cryptography].

[TPS_MANI_03207] IPsec Internet Key Exchange protocol configuration ⌈In 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.localCertificate and IPSecRule.remoteCertificate defines that Digital Signature Authentication is used.⌋

The supported IKE CipherSuites are configured with the IPSecConfigProps.ikeCipherSuiteName. The IPSecConfigProps contains additional IKE specific configuration settings.

Enumeration	**IPsecDpdActionEnum**
Note	Potential Dead Peer Detection (Dpd) Actions
Aggregated by	IPSecConfigProps.dpdAction
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 7.33: IPsecDpdActionEnum

[TPS_MANI_03208] Protection of `AdaptivePlatformServiceInstance` by IPsec ⌈To 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.⌋

IP Multicast protection by IPsec is not supported. It is intentionally 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.

7.2.3 MACsec

MACsec (Media Access Control security) is defined by IEEE standard 802.1AE and operates at the medium access control layer and defines connectionless data confidentiality and integrity for media access independent protocols.

The MACsec standard specifies a set of protocols to meet the security requirements for protecting data traversing Ethernet LANs.

MACsec allows unauthorized LAN connections to be identified and excluded from communication within the network. In common with IPsec and TLS, MACsec defines a security infrastructure to provide data confidentiality, data integrity and data origin authentication.

With security on the MAC layer, protocols not based on IP as well as protocols using multicast can be protected by MACsec.

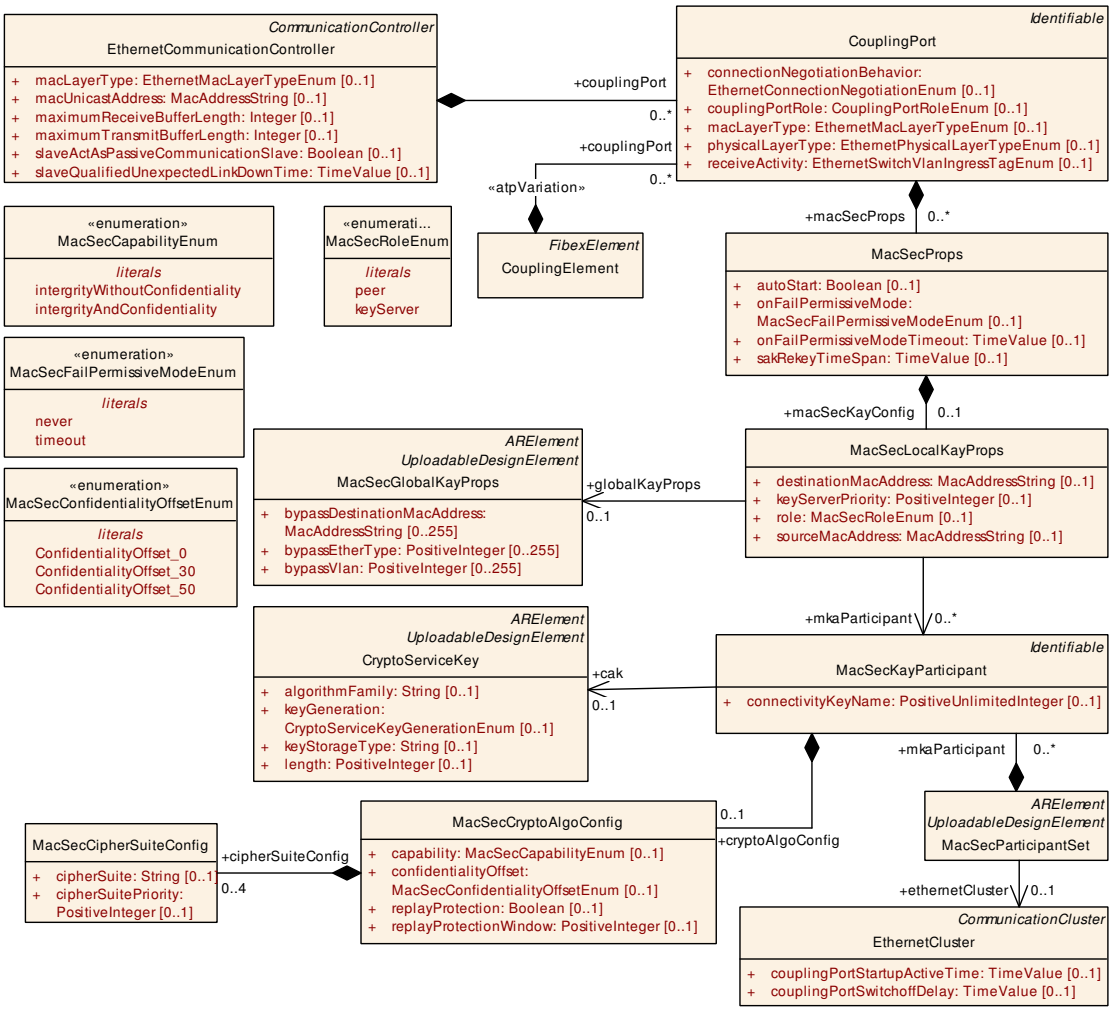


Figure 7.6: MACsec configuration

The MACsec standard relies on the usage of authentic partners, which are called MAC Security Entities (SecYs). The standard organizes this by MACsec Protocol Data Units (MPDUs), which reside in the MAC layer (ISO/OSI layer 2).

The MAC Security Key Agreement Entity (KaY) will take care of properly configuring and managing the MAC Security Entities to enable the secure communication channel.

In the System Design model the configuration of MACsec is supported on `CouplingPort`s that in turn are aggregated either by a `CouplingElement` or by an `EthernetCommunicationController`.

This allows the configuration of MACsec on a switch port or on an `EthernetCommunicationController` of a `MachineDesign`. The same modeling approach is used in the Classic Platform System Template as well.

CAUTION

The constraints marked with the imposition time `IT_SysDes` defined in chapter [4, TPS System Template] 3.3.8 *"MACsec"* are valid for this specification as well.

[TPS_MANI_03317] MACsec configuration

Status: DRAFT

⌈A `CouplingPort` that aggregates the `MacSecProps` in the role `macSecProps` defines a MAC Security Entity.⌋

Constraint [constr_5361] from the System Template is also valid in the System Design model of the Adaptive Platform.

Constraint [constr_3726] from the System Template is also valid in the System Design model of the Adaptive Platform.

[TPS_MANI_03318] MAC Security Key Agreement Entity configuration

Status: DRAFT

⌈The `MacSecLocalKayProps` element that is referenced by the `MacSecProps` in the role `macSecKayConfig` in combination with `MacSecGlobalKayProps` that is referenced by `MacSecLocalKayProps` in the role `globalKayProps` defines the configuration settings for the MACsec Key Agreement that is responsible for the peer discovery and key negotiation to secure the Ethernet link.⌋

Class	**MacSecProps**			
Note	This meta-class allows to configure MACsec (Media access control security) and the MKA (MACsec Key Agreement) for the CouplingPort (PHY). **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	CouplingPort.macSecProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
autoStart	Boolean	0..1	attr	This attribute defines how the Port Access Entity (PAE) is started: • true := Autostart • false := Manual Start **Tags:** atp.Status=candidate
macSecKay Config	MacSecLocalKayProps	0..1	aggr	Properties to configure the MKA instance (KaY) for a controlled CouplingPort (PaE). **Tags:** atp.Splitkey=macSecKayConfig atp.Status=candidate
onFail Permissive Mode	MacSecFailPermissive ModeEnum	0..1	attr	This attribute sets the behavior of the Port Access Entity in case MACsec does not succeed. **Tags:** atp.Status=candidate
onFail Permissive ModeTimeout	TimeValue	0..1	attr	Timeout in seconds to enable the controlled port in case onFailPermissiveMode is set to Timeout. **Tags:** atp.Status=candidate
sakRekeyTime Span	TimeValue	0..1	attr	Time in seconds to trigger the rekey of an in use SAK (Static Secure Association key). If set to 0, the rekey will not be triggered after a time span. **Tags:** atp.Status=candidate

Table 7.34: MacSecProps

Class	MacSecLocalKayProps			
Note	Configuration of the MAC Security Key Agreement Entity (KaY). **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	MacSecProps.macSecKayConfig			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
destinationMac Address	MacAddressString	0..1	attr	This attribute defines the destination MAC Address that is used to calculate the ICV (Integrity Check Value). **Tags:** atp.Status=candidate
globalKayProps	MacSecGlobalKay Props	0..1	ref	Reference to properties that are shared between MAC Security Key Agreement Entities. **Tags:** atp.Splitkey=globalKayProps atp.Status=candidate
keyServer Priority	PositiveInteger	0..1	attr	This attribute defines the key-server priority. **Tags:** atp.Status=candidate
mkaParticipant	MacSecKayParticipant	*	ref	Reference to MKA participant settings supported on the CouplingPort. **Tags:** atp.Splitkey=mkaParticipant atp.Status=candidate
role	MacSecRoleEnum	0..1	attr	Role of the MAC Security Key Agreement Entity **Tags:** atp.Status=candidate
sourceMac Address	MacAddressString	0..1	attr	This attribute defines the source MAC Address that is used to calculate the ICV (Integrity Check Value). **Tags:** atp.Status=candidate

Table 7.35: MacSecLocalKayProps

Class	MacSecGlobalKayProps			
Note	Configuration of the MAC Security Key Agreement Entity properties that are shared by different KaY configurations. **Tags:** atp.Status=candidate atp.recommendedPackage=MacSecGlobalKayProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
bypass DestinationMac Address	MacAddressString	0..255	attr	The attribute is used to define MAC destination addresses that are bypassed by MACsec. The provided MAC addresses will not be MACsec protected. Please note, IEEE 802.1X-2020 chapter 11.1.1 (Table 11-1) define specific MAC destination addresses to be bypassed.
bypassEther Type	PositiveInteger	0..255	attr	This attribute is used to define EtherTypes that are bypassed by MACsec. The providedEtherType will not be MACsec protected. **Tags:** atp.Status=candidate

▽

△

Class	MacSecGlobalKayProps			
bypassVlan	PositiveInteger	0..255	attr	This attribute is used to define VLAN-IDs that are bypassed by MACsec. The provided VLAN-IDs will not be MACsec protected. (VLAN-ID 0 is interpreted as no-VLAN --> Bypass untagged traffic) **Tags:** atp.Status=candidate

Table 7.36: MacSecGlobalKayProps

Class	MacSecParticipantSet			
Note	Collection of MACsec Kay Participants on an Ethernet Link. **Tags:** atp.Status=candidate atp.recommendedPackage=MacSecKayParticipantSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
ethernetCluster	EthernetCluster	0..1	ref	Reference to the EthernetCluster (Link) on which the KaY participants are located **Tags:** atp.Splitkey=ethernetCluster atp.Status=candidate
mkaParticipant	MacSecKayParticipant	*	aggr	Configuration of a MKA Participant. **Tags:** atp.Splitkey=mkaParticipant, mkaParticipant.shortName atp.Status=candidate

Table 7.37: MacSecParticipantSet

Class	MacSecKayParticipant			
Note	This meta-class configures a MKA participant. **Tags:** atp.Status=candidate atp.recommendedPackage=MacSecKayParticipants			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	MacSecParticipantSet.mkaParticipant			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
cak	CryptoServiceKey	0..1	ref	Reference to the Connectivity Association Key (CAK) **Tags:** atp.Splitkey=cak
connectivityKey Name	PositiveUnlimitedInteger	0..1	attr	Connectivity Key Name (CKN) defined as a hexadecimal number
cryptoAlgo Config	MacSecCryptoAlgo Config	0..1	aggr	Cryptography that is used by the MKA Participant. **Tags:** atp.Splitkey=cryptoAlgoConfig atp.Status=candidate

Table 7.38: MacSecKayParticipant

Class	MacSecCryptoAlgoConfig			
Note	This meta-class defines the cryptography configuration for MACsec. **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	MacSecKayParticipant.cryptoAlgoConfig			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
capability	MacSecCapabilityEnum	0..1	attr	This attribute defines the MACsec capability. **Tags:** atp.Status=candidate
cipherSuite Config	MacSecCipherSuite Config	0..4	aggr	Cipher suite configuration to use with MACsec. **Tags:** atp.Splitkey=cipherSuiteConfig atp.Status=candidate
confidentiality Offset	MacSecConfidentiality OffsetEnum	0..1	attr	The MACsec confidentiality offset specifies the number of bytes starting from the frame header. MACsec encrypts only the bytes after the offset in a frame. **Tags:** atp.Status=candidate
replayProtection	Boolean	0..1	attr	This attribute is used to configure the MACsec replay protection. **Tags:** atp.Status=candidate
replayProtection Window	PositiveInteger	0..1	attr	In case replay protection is active, this attribute defines the replay protection window. **Tags:** atp.Status=candidate

Table 7.39: MacSecCryptoAlgoConfig

Class	MacSecCipherSuiteConfig			
Note	This meta-class defines the cipher suite configuration to use with MACsec. cipherSuitePriority is present in case the MKA instance acts as a Key Server to select the cipher suite to use for MACsec. **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	MacSecCryptoAlgoConfig.cipherSuiteConfig			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
cipherSuite	String	0..1	attr	Cipher Suite to use for MACsec. **Tags:** atp.Status=candidate
cipherSuite Priority	PositiveInteger	0..1	attr	In case the MKA instance acts as a Key Server, the priority is used to select the Cipher Suite to use with MACsec from the supported Ciphers. **Tags:** atp.Status=candidate

Table 7.40: MacSecCipherSuiteConfig

[TPS_MANI_03319] Standardized values for the attribute `cipherSuite` of meta-class `MacSecCipherSuiteConfig`

Status: DRAFT

⌈The following values for `MacSecCipherSuiteConfig.cipherSuite` are reserved by the AUTOSAR standard:

- GCM-AES-128
- GCM-AES-256
- GCM-AES-XPN-128

- GCM-AES-XPN-256

⌋

[TPS_MANI_03320] Semantics of MacSecCipherSuiteConfig.cipherSuitePriority

Status: DRAFT

⌈The MacSecCryptoAlgoConfig can define up to four MacSecCipherSuiteConfigs. If more than one MacSecCipherSuiteConfig is defined then the cipherSuitePriority decides about the chosen cipher. The cipherSuitePriority of value 1 means the highest priority and 4 means the lowest priority.⌋

Enumeration	**MacSecConfidentialityOffsetEnum**
Note	This enum defines the MACsec capability options. **Tags:** atp.Status=candidate
Aggregated by	MacSecCryptoAlgoConfig.confidentialityOffset
Literal	***Description***
Confidentiality Offset_0	confidentiality offset of 0. **Tags:** atp.EnumerationLiteralIndex=0 xml.name=CONFIDENTIALITY-OFFSET--0
Confidentiality Offset_30	confidentiality offset of 30. **Tags:** atp.EnumerationLiteralIndex=1 xml.name=CONFIDENTIALITY-OFFSET--30
Confidentiality Offset_50	confidentiality offset of 50. **Tags:** atp.EnumerationLiteralIndex=2 xml.name=CONFIDENTIALITY-OFFSET--50

Table 7.41: MacSecConfidentialityOffsetEnum

Enumeration	**MacSecCapabilityEnum**
Note	This enum defines the MACsec capability options. **Tags:** atp.Status=candidate
Aggregated by	MacSecCryptoAlgoConfig.capability
Literal	***Description***
intergrityAnd Confidentiality	Option that ensures confidentiality and integrity **Tags:** atp.EnumerationLiteralIndex=1
intergrityWithout Confidentiality	Option that ensures integrity without confidentiality **Tags:** atp.EnumerationLiteralIndex=0

Table 7.42: MacSecCapabilityEnum

Enumeration	**MacSecRoleEnum**
Note	This enum defines the MACsec Role options. **Tags:** atp.Status=candidate
Aggregated by	MacSecLocalKayProps.role

▽

△

Enumeration	**MacSecRoleEnum**
Literal	***Description***
keyServer	Port acts in the KeyServer role **Tags:** atp.EnumerationLiteralIndex=1
peer	Port acts in the peer role **Tags:** atp.EnumerationLiteralIndex=0

Table 7.43: MacSecRoleEnum

Enumeration	**MacSecFailPermissiveModeEnum**
Note	Behavior options of the Port Access Entity in case MACsec does not succeed. **Tags:** atp.Status=candidate
Aggregated by	MacSecProps.onFailPermissiveMode
Literal	***Description***
never	The controlled port will never be set to enabled if the participants cannot establish and successfully use a MACsec Secure Channel. **Tags:** atp.EnumerationLiteralIndex=0
timeout	The controlled port will be set to enabled and MACsec will not be used in the port if the timeout value (onFailPermissiveModeTimeout) is reached and the following conditions apply: - A participant belonging to the same CA was recognized and authenticated. - A secure channel could be established. - Both participants can transmit and receive MACsec protected traffic through the SC. **Tags:** atp.EnumerationLiteralIndex=1

Table 7.44: MacSecFailPermissiveModeEnum

For MACsec to start up, a key distribution according to the MAC Security Key Agreement (MKA) protocol runs first. The Connectivity Association Key (CAK) used for MKA is referenced as `MacSecKayParticipant.cak`. The Connectivity Key Name (CKN), which identifies the CAK, is described by the parameter `MacSecKayParticipant.connectivityKeyName` (CKN). The CAK and the CKN shall match on both ends of a link to create a MACsec-secured link.

In addition, a user may choose to allow additional Ethernet frames unprotected, e.g., for installing keys into ECUs. These frames can be selected via so-called bypass rules based on e.g., EtherType or VLAN-ID defined in `MacSecGlobalKayProps`.

Each `MacSecKayParticipant` that is referenced by `MacSecLocalKayProps` in the role `mkaParticipant` defines the Connectivity Association Key and the `MacSecCryptoAlgoConfig`. The `MacSecKayParticipant` settings in the connectivity association need to match on both `CouplingPort`s of the link to enable MACsec.

For better control and predictability, the MAC Security Key Agreement participant that will get the KeyServer role (the node generating the MACsec Key for this link) can be configured by `role`.

7.2.4 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] Modeling of a Partial Network Cluster ⌈A 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.⌋

[TPS_MANI_03225] References to VLANs in PncMapping ⌈There 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.⌋

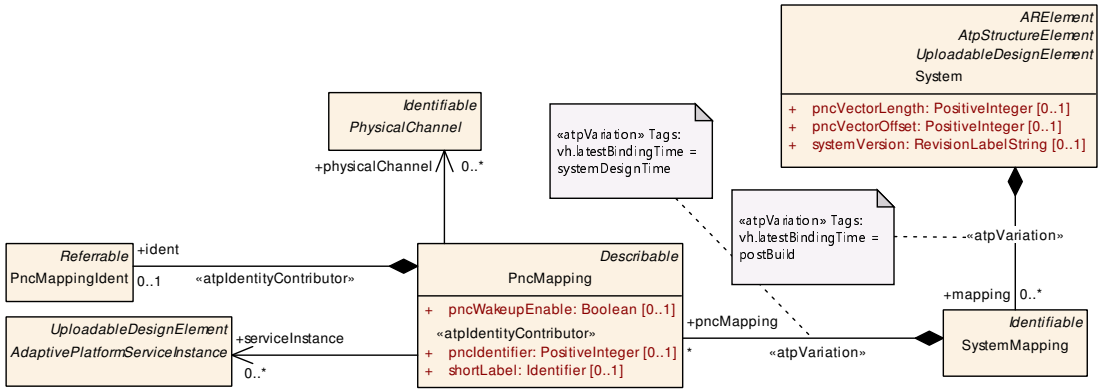


Figure 7.7: PncMapping with collection of ServiceInstances that are participating in the Partial Network Cluster

Class	SystemMapping			
Note	The system mapping aggregates all mapping aspects that are relevant in the System Description.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	System.mapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
pncMapping	PncMapping	*	aggr	Mappings between Virtual Function Clusters and Partial Network Clusters. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=pncMapping, pncMapping.ident.shortName, pncMapping.pncIdentifier, pncMapping.shortLabel, pncMapping.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime

Table 7.45: SystemMapping

Class	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*			
Aggregated by	SystemMapping.pncMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ident	PncMappingIdent	0..1	aggr	This adds the ability to become referrable to PncMapping. **Stereotypes:** atpIdentityContributor
physicalChannel	PhysicalChannel	*	ref	This reference maps the partial network to a communication channel. **Tags:** atp.Splitkey=physicalChannel
pncConsumedProvidedServiceInstanceGroup	ConsumedProvidedServiceInstanceGroup	*	ref	ConsumedProvidedServiceInstanceGroup used in a Partial Network Cluster. This reference is optional, since this could be used for starting and stopping ConsumedProvidedServiceInstanceGroup according the requested partial network, but is not necessarily needed. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=pncConsumedProvidedServiceInstanceGroup, pncConsumedProvidedServiceInstanceGroup.consumedProvidedServiceInstanceGroup, pncConsumedProvidedServiceInstanceGroup.variationPoint.shortLabel vh.latestBindingTime=postBuild
pncIdentifier	PositiveInteger	0..1	attr	Identifer of the Partial Network Cluster. This number represents the absolute bit position of this Partial Network Cluster in the NM Pdu. **Stereotypes:** atpIdentityContributor
pncWakeupEnable	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. **Tags:** atp.Status=obsolete



△

Class	**PncMapping**			
serviceInstance	AdaptivePlatform ServiceInstance	*	ref	Reference to ServiceInstances that are participating in a Partial Network Cluster. **Tags:** atp.Splitkey=serviceInstance This Attribute is only used by the AUTOSAR Adaptive Platform.
shortLabel	Identifier	0..1	attr	This attribute specifies an identifying shortName for the PncMapping. It shall be unique in the System scope. **Stereotypes:** atpIdentityContributor
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. **Tags:** atp.Splitkey=vfc, vfc.contextComposition, vfc.contextComponent, vfc.target **InstanceRef implemented by:** PortGroupInSystem InstanceRef

Table 7.46: PncMapping

7.3 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] Allowed components in system description with `category SYSTEM_DESIGN_DESCRIPTION`. ⌈`SwComponentPrototype`s 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.⌋

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*

▽

△

Class	**RootSwCompositionPrototype**			
Aggregated by	*AtpClassifier*.atpFeature, System.rootSoftwareComposition			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
software Composition	CompositionSw ComponentType	0..1	tref	We assume that there is exactly one top-level composition that includes all Component instances of the system. **Stereotypes:** isOfType **Tags:** atp.Splitkey=softwareComposition

Table 7.47: RootSwCompositionPrototype

If a Software Component communicates over service-oriented communication and provides or requires a `ServiceInterface`, the opposite communication end is not always known upfront. In the `System` with `category SYSTEM_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.

[TPS_MANI_03114] Usage of `AssemblySwConnector`s in the System Design model ⌈In the `System` with `category SYSTEM_DESIGN_DESCRIPTION` it is allowed to indicate the service-oriented communication between two communication endpoints by `AssemblySwConnector`s 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`.⌋

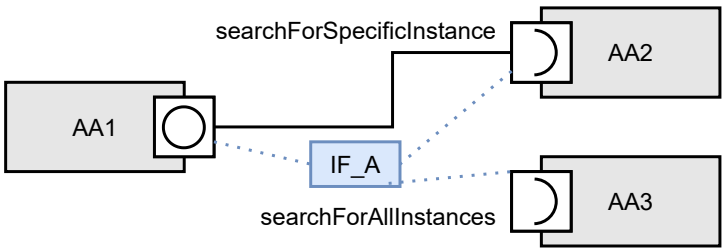


Figure 7.8: Example for Assembly connectors in System Design model

7.4 Modeling of Service-Oriented Communication between Classic and Adaptive Platform

AUTOSAR classic platform does not support `ServiceInterface`s 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 `ClientServerInterface`s, `SenderReceiverInterface`s or `TriggerInterface`s 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 mapping elements of PortInterfaces of the Classic Platform to a single ServiceInterface of the Adaptive Platform.

Class	InterfaceMapping			
Note	This meta-class collects the mappings of elements of a single ServiceInterface to PortInterface elements of the AUTOSAR Classic Platform. **Tags:** atp.recommendedPackage=InterfaceMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
eventMapping	EventMapping	*	aggr	Mapping of a VariableDataPrototype in a SenderReceiver Interface to an Event in a ServiceInterface. **Tags:** atp.Splitkey=eventMapping, eventMapping.shortName
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.Splitkey=fieldMapping, fieldMapping.shortName
fireAndForgetMethodMapping	FireAndForgetMethodMapping	*	aggr	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.Splitkey=fireAndForgetMethodMapping, fireAndForgetMethodMapping.shortName
methodMapping	MethodMapping	*	aggr	Mapping of a ClientServerOperation in a ClientServer Interface to a Method in a ServiceInterface. **Tags:** atp.Splitkey=methodMapping, methodMapping.shortName

Table 7.48: InterfaceMapping

[constr_3370] InterfaceMapping shall map all elements of a single ServiceInterface

Imposition time: IT_SysDes

⌈The 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.⌋

Figure 7.10 shows a possible System Design modeling approach where a piece of application software communicates 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.

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.

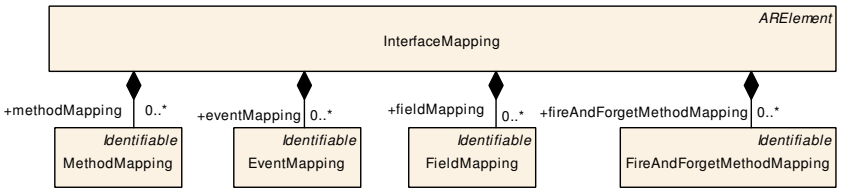


Figure 7.9: InterfaceMapping Overview

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 `PortInterface`s are used several times on a network, the network configuration needs to be taken into account to emulate how the service discovery behaves on the network. From this information, the actual communication relations on software level can be deduced.

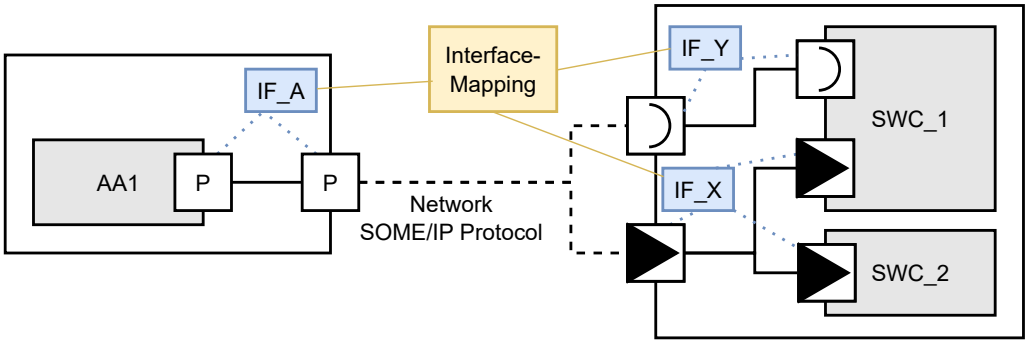


Figure 7.10: Example for a modeling of service-oriented communication between application software on the *AUTOSAR adaptive platform* and Software Components of the Classic Platform

7.4.1 Method Mapping

[TPS_MANI_03111] Mapping between `method` and `ClientServerOperation` located in a `ClientServerInterface` ⌈The mapping between a `method` located in a `ServiceInterface` and a `ClientServerOperation` located in a `ClientServerInterface` is provided by the class `MethodMapping`.⌋

Class	**MethodMapping**			
Note	Mapping of a ClientServerOperation that is located in a ClientServerInterface to a Method that is located in a ServiceInterface. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	InterfaceMapping.methodMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	MethodMapping			
clientServer Operation	ClientServerOperation	0..1	ref	Reference to a ClientSeverOperation that is located in a ClientSeverInterface. **Tags:** atp.Splitkey=clientServerOperation
method	ClientServerOperation	0..1	ref	Reference to a Method that is located in a Service Interface. **Tags:** atp.Splitkey=method

Table 7.49: MethodMapping

[advisory_01000] Existence of the reference `MethodMapping.clientServerOperation`

Imposition time: `IT_SysDes`

⌈For each `MethodMapping`, the reference in the role `MethodMapping.clientServerOperation` should exist.⌋

[advisory_01001] Existence of reference `MethodMapping.method`

Imposition time: `IT_SysDes`

⌈For each `MethodMapping`, the reference in the role `MethodMapping.method` should exist.⌋

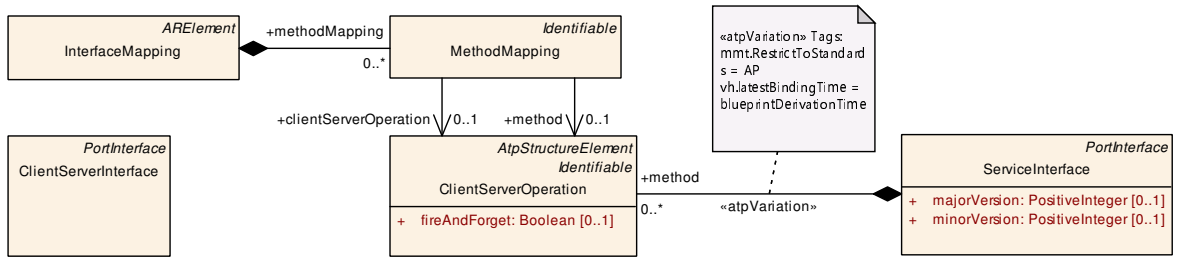


Figure 7.11: Mapping of a Method to a ClientServerOperation

7.4.2 Event Mapping

[TPS_MANI_03112] Mapping between an `event` and a `dataElement` ⌈The mapping between an `event` located in a `ServiceInterface` and a `dataElement` located in a `SenderReceiverInterface` is provided by the class `EventMapping`.⌋

Class	EventMapping
Note	Mapping of a VariableDataPrototype that is located in a SenderReceiverInterface to an Event that is located in a ServiceInterface. This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*
Aggregated by	InterfaceMapping.eventMapping

▽

△

Class	**EventMapping**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	VariableDataPrototype	0..1	ref	Reference to a VariableDataPrototype that is located in a SenderReceiverInterface. **Tags:** atp.Splitkey=dataElement
event	VariableDataPrototype	0..1	ref	Reference to an Event that is located in a Service Interface. **Tags:** atp.Splitkey=event

Table 7.50: EventMapping

[advisory_01002] Existence of reference `EventMapping.dataElement`

Imposition time: `IT_SysDes`

⌈For each `EventMapping`, the reference in the role `EventMapping.dataElement` should exist.⌋

[advisory_01003] Existence of reference `EventMapping.event`

Imposition time: `IT_SysDes`

⌈For each `EventMapping`, the reference in the role `EventMapping.event` should exist.⌋

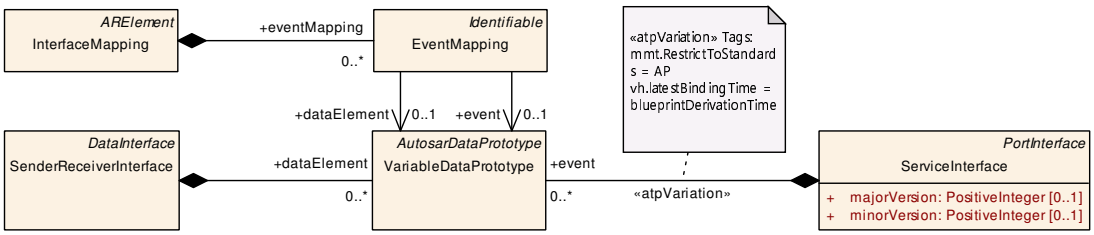


Figure 7.12: Mapping between an `event` and a `dataElement`

7.4.3 FieldMapping

[TPS_MANI_03113] Mapping between a `field` and elements of Classic Platform `PortInterface`s

⌈The mapping between a `field` located in a `ServiceInterface` and elements of Classic Platform `PortInterface`s 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 `operation`s that are located in a `ClientServerInterface`.⌋

[constr_3367] `FieldMapping.notifierDataElement` reference

Imposition time: `IT_SysDes`

⌈The `FieldMapping` shall only contain the `notifierDataElement` reference if the `hasNotifier` attribute in the referenced `field` is set to true.⌋

[constr_3368] FieldMapping.getterOperation reference

Imposition time: IT_SysDes

⌈The FieldMapping shall only contain the getterOperation reference if the hasGetter attribute in the referenced field is set to true.⌋

[constr_3369] FieldMapping.setterOperation reference

Imposition time: IT_SysDes

⌈The FieldMapping shall only contain the setterOperation reference if the hasSetter attribute in the referenced field is set to true.⌋

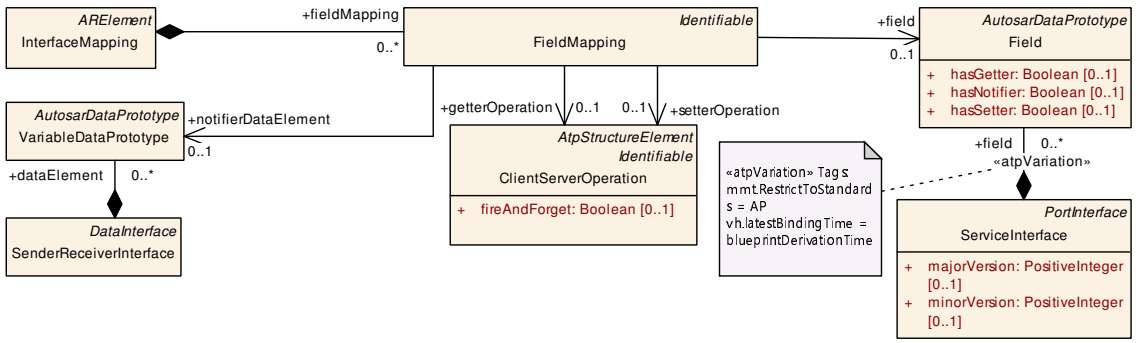


Figure 7.13: Mapping between a field and elements of Classic Platform PortInterfaces

Class	**FieldMapping**			
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. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	InterfaceMapping.fieldMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
field	Field	0..1	ref	Reference to a field that is located in a ServiceInterface. **Tags:** atp.Splitkey=field
getterOperation	ClientServerOperation	0..1	ref	Reference to a ClientServerOperation that represents the getter Method in the Field. **Tags:** atp.Splitkey=getterOperation
notifierData Element	VariableDataPrototype	0..1	ref	Reference to a VariableDataPrototype that represents the notifier in the Field. **Tags:** atp.Splitkey=notifierDataElement
setterOperation	ClientServerOperation	0..1	ref	Reference to a ClientServerOperation that represents the setter Method in the Field. **Tags:** atp.Splitkey=setterOperation

Table 7.51: FieldMapping

[advisory_01004] Existence of reference FieldMapping.field

Imposition time: IT_SysDes

⌈For each FieldMapping, the reference in the role FieldMapping.field should exist.⌋

[advisory_01005] Existence of references `FieldMapping.getterOperation`, `setterOperation`, and `notifierDataElement`

Imposition time: `IT_SysDes`

⌈For each `FieldMapping`, at least one of the references in the role

- `FieldMapping.getterOperation`
- `FieldMapping.setterOperation`
- `FieldMapping.notifierDataElement`

should exist.⌋

7.4.4 Fire And Forget Method Mapping

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 4.2.5.1.

On the *AUTOSAR adaptive platform*, 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].

On the *AUTOSAR classic platform*, a fire and forget method cannot 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.

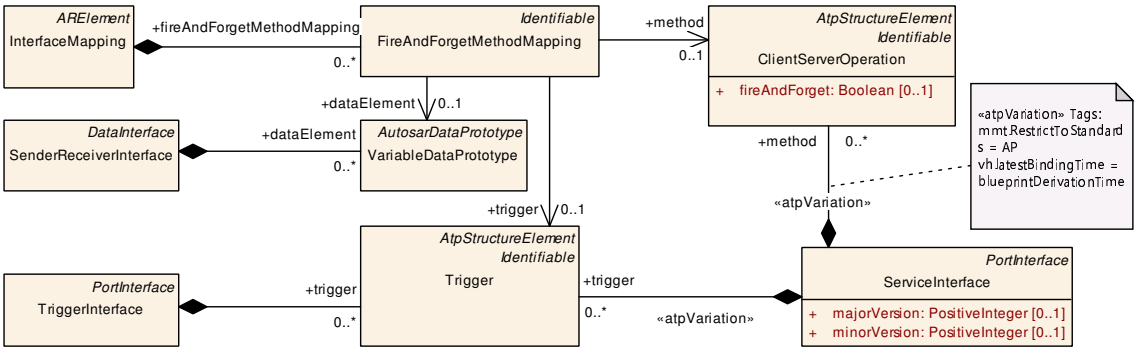


Figure 7.14: Mapping between a fire and forget `method` and elements of Classic Platform `PortInterface`s

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. The order of elements in the Structure shall be the same as the order of `ArgumentDataPrototype`s within the `ClientServerOperation`.

This representation ensures that the SOME/IP serialization results in the same byte stream as in the Adaptive Platform where all `argument`s which have the `direction`

in are serialized according to the order of the ArgumentDataPrototypes within the ClientServerOperation.

If the fire and forget method has no parameters, a Trigger is used to describe such a method on the *AUTOSAR classic platform*.

The SOME/IP MessageType shall be set to REQUEST_NO_RETURN if a fire and forget method is transmitted over SOME/IP.

[TPS_MANI_03115] Mapping between a fire and forget method and elements of Classic Platform PortInterfaces ⌈The 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 FireAndForgetMethodMapping.

If the fire and forget method is represented on the *AUTOSAR 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 on the *AUTOSAR classic platform* by a Trigger then this trigger is mapped to a method located in a ServiceInterface.⌋

[constr_3371] Mutually exclusive existence of FireAndForgetMethodMapping.dataElement reference and FireAndForgetMethodMapping.trigger reference

Imposition time: IT_SysDes

⌈A FireAndForgetMethodMapping shall never reference a dataElement and a trigger at the same time.⌋

[constr_3376] FireAndForgetMethodMapping shall reference only fire and forget methods

Imposition time: IT_SysDes

⌈A FireAndForgetMethodMapping is only allowed to reference a ClientServerOperation in role method for which the value of attribute method.fireAndForget is set to true.⌋

Class	**FireAndForgetMethodMapping**			
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. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	InterfaceMapping.fireAndForgetMethodMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**FireAndForgetMethodMapping**			
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.Splitkey=dataElement
method	ClientServerOperation	0..1	ref	Reference to a Fire&Forget Method that is located in a ServiceInterface. **Tags:** atp.Splitkey=method
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.Splitkey=trigger

Table 7.52: FireAndForgetMethodMapping

[advisory_01006] Existence of reference `FireAndForgetMethodMapping.method`

Imposition time: `IT_SysDes`

⌈For each `FireAndForgetMethodMapping`, the reference in the role `FireAndForgetMethodMapping.method` should exist.⌋

[advisory_01007] Existence of references `FireAndForgetMethodMapping.dataElement` and `trigger`

Imposition time: `IT_SysDes`

⌈For each `FireAndForgetMethodMapping`, at least one of the references in the role

- `FireAndForgetMethodMapping.dataElement`
- `FireAndForgetMethodMapping.trigger`

should exist.⌋

7.5 Crypto System Design

A cryptographic key is being used in different roles, as sketched in Figure 7.15:

1. Code that accesses the cryptographic key can be one of several **owner**s of the cryptographic key and therefore assume read and write access to the cryptographic key.
2. Code that accesses the cryptographic key may be restricted to read access, without the permission to change. This code would be one of several **user**s of the cryptographic key.

The benefit of formally describing the relation between cryptographic keys (represented by meta-class `CryptoKeySlotDesign`) and the users of these keys (represented by meta-class `CryptoKeySlotUserDesign`) via the `CryptoKeySlotUsageDesign` is that a formal check for completeness and consistency becomes possible.

Another aspect is that the modeling of these relations can be taken as the recipe for handling cryptographic keys in a further design step, in which CryptoKeySlot-UserDesign corresponds to meta-class Executable and CryptoKeySlotUsage-Design corresponds to RPortPrototype.

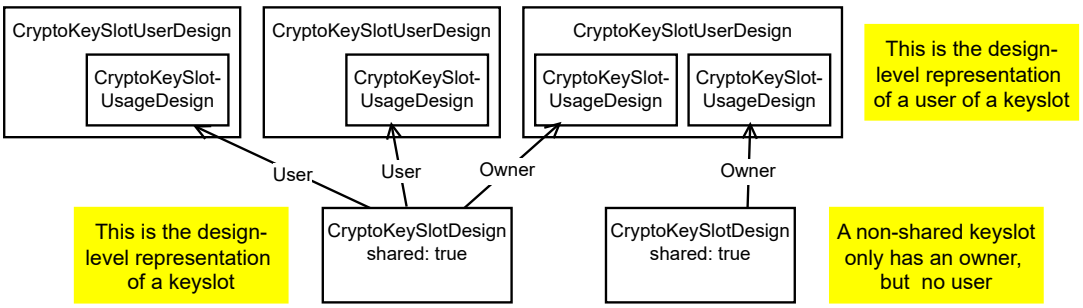


Figure 7.15: Example for a high-level crypto design as part of the System Design model

[constr_10612] Standardized values for attribute CryptoKeySlotDesign-Description.cryptoObjectType

Imposition time: IT_SysDes

⌈In the context of CryptoKeySlotDesignDescription.cryptoObjectType, **only** the following values shall be used:

- PUBLIC_KEY
- PRIVATE_KEY
- SYMMETRIC_KEY

⌋

The correspondence between CryptoKeySlotDesign and meta-class CryptoKey-Slot can be tracked even further down to the *target configuration phase*.

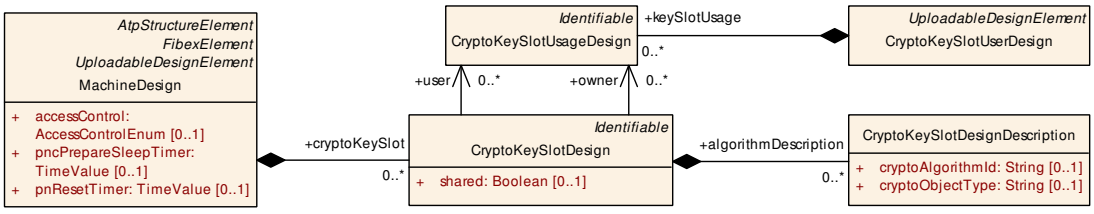


Figure 7.16: High-level crypto system design

The values for attribute CryptoKeySlotDesignDescription.crypto-AlgorithmId listed in [TPS_MANI_01470] are **standardized** by AUTOSAR.

The list of crypto algorithms contained in [TPS_MANI_01470] represents an open list. This means that it is possible to use a custom value in attribute CryptoKey-SlotDesignDescription.cryptoAlgorithmId, provided that the custom value is guaranteed not to clash with future extensions of the standard.

This protection against a future name clash can be achieved by adding a name fragment (e.g. a company name) to the custom value that is highly unlikely to appear in a future extension of the standard.

[TPS_MANI_01470] Standardized values of attribute `CryptoKeySlotDesign-Description.cryptoAlgorithmId` ⌈

Algorithm ID	Description
Symmetric Algorithms	
AES_128	AES-128
AES_256	AES-256
CHACHA	ChaCha
SM4	SM4
HMAC_128	HMAC/128
HMAC_256	HMAC/256
Asymmetric Algorithms	
RSA_2048	RSA-2048
RSA_4096	RSA-4096
ED25519	ed25519
ED448	ed448
SECP256R1	secp256r1
SECP384R1	secp384r1
SECP521R1	secp521r1
Key Exchange	
X25519	x25519
X448	x448
PQC	
ML_DSA_44	ML-DSA/44
ML_DSA_65	ML-DSA/65
ML_DSA_87	ML-DSA/87
SLH_DSA_128	SLH-DSA/128
SLH_DSA_192	SLH-DSA/192
SLH_DSA_256	SLH-DSA/256
ML_KEM_512	ML-KEM/512
ML_KEM_768	ML-KEM/768
ML_KEM_1024	ML-KEM/1024

⌋

[constr_10613] `CryptoKeySlotDesign` where attribute `shared` is set to `false`

Imposition time: `IT_SysDes`

⌈A `CryptoKeySlotDesign` where attribute `shared` is set to `false` shall not refer to a `CryptoKeySlotUsageDesign` in the role `user`.⌋

Class	**CryptoKeySlotDesign**			
Note	This meta-class represents a key slot on the design level. This class is used to create a model of the usage of key slots so that owners and users of the key slot can be described, as the basis for consistency checking of the overall design. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	MachineDesign.cryptoKeySlot			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
algorithm Description	CryptoKeySlotDesign Description	*	aggr	This aggregation contains the collection of crypto algorithm descriptions that can be used in the context of the enclosing crypto key slot design. **Tags:** atp.Splitkey=algorithmDescription
owner	CryptoKeySlotUsage Design	*	ref	This reference attributes an owner of the enclosing crypto key slot design. **Tags:** atp.Splitkey=owner
shared	Boolean	0..1	attr	This attribute controls whether the cryptographic key is shared with at least one user.
user	CryptoKeySlotUsage Design	*	ref	This reference attributes a user of the enclosing crypto key slot design. **Tags:** atp.Splitkey=user

Table 7.53: CryptoKeySlotDesign

[constr_10614] Existence of attribute `CryptoKeySlotDesign.shared`

Imposition time: `IT_SysDes`

⌈For each `CryptoKeySlotDesign`, the attribute `shared` shall exist.⌋

Class	**CryptoKeySlotUserDesign**			
Note	This meta-class represents an abstracted user (application) of a crypto key slot on design level. **Tags:** atp.recommendedPackage=CryptoKeySlotDesigns This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
keySlotUsage	CryptoKeySlotUsage Design	*	aggr	This aggregation represents the collection of key slot usages in the context of the enclosing CryptoKeySlotUser Design. **Tags:** atp.Splitkey=keySlotUsage, keySlotUsage.short Name

Table 7.54: CryptoKeySlotUserDesign

[advisory_01016] Existence of the aggregation in the role `CryptoKeySlotUserDesign.keySlotUsage`

Imposition time: `IT_SysDes`

⌈For each `CryptoKeySlotUserDesign`, the aggregation in the role `keySlotUsage` should exist.⌋

Class	CryptoKeySlotDesignDescription			
Note	This meta-class is used to provide properties of the crypto algorithm on design level. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	CryptoKeySlotDesign.algorithmDescription			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
cryptoAlgorithmId	String	0..1	attr	This attribute describes the identity of the crypto algorithm associated with the key stored in this key slot.
cryptoObjectType	String	0..1	attr	This attribute describes the type of the cryptographic object to be stored in this key slot.

Table 7.55: CryptoKeySlotDesignDescription

Class	CryptoKeySlotUsageDesign			
Note	This meta-class represents the concrete usage (on design level) of a cryptographic key. **Tags:** atp.recommendedPackage=CryptoKeySlotDesigns This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	CryptoKeySlotUserDesign.keySlotUsage			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 7.56: CryptoKeySlotUsageDesign

7.6 Process Design to Machine Design Mapping

[TPS_MANI_01229] Pre-allocation of a given `ProcessDesign` on a specific `MachineDesign` ⌈It is possible to pre-allocate a given `ProcessDesign` on a specific `MachineDesign`. For this purpose meta-class `ProcessDesignToMachineDesignMapping` exists.⌋

[constr_1693] Relation of `Executable`, `ProcessDesign`, and `Process`

Imposition time: `IT_SysDes`

⌈Any `Executable` that is referenced by a `ProcessDesign` shall also be referenced by every `Process` that references the `ProcessDesign`.⌋

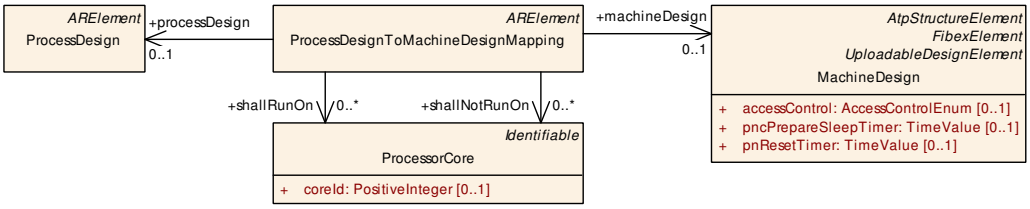


Figure 7.17: Modeling of the `ProcessDesignToMachineDesignMapping`

Class	**ProcessDesignToMachineDesignMapping**			
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 target-configuration phase. **Tags:** atp.recommendedPackage=ProcessDesignToMachineDesignMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
machineDesign	MachineDesign	0..1	ref	This reference identifies the MachineDesign in the context of the ProcessDesignToMachineDesignMapping. **Tags:** atp.Splitkey=machineDesign
processDesign	ProcessDesign	0..1	ref	This reference identifies the ProcessDesign in the context of the ProcessDesignToMachineDesignMapping. **Tags:** atp.Splitkey=processDesign
shallNotRunOn	ProcessorCore	*	ref	This reference indicates a collection of cores onto which the mapped process shall not be executing. **Tags:** atp.Splitkey=shallNotRunOn
shallRunOn	ProcessorCore	*	ref	This reference indicates a collection of cores onto which the mapped process shall be executing. **Tags:** atp.Splitkey=shallRunOn

Table 7.57: ProcessDesignToMachineDesignMapping

[constr_10154] Multiplicity of reference in the role `ProcessDesignToMachineDesignMapping.processDesign`

Imposition time: `IT_SysDes`

⌈For each `ProcessDesignToMachineDesignMapping`, the reference in the role `processDesign` shall exist.⌋

[constr_10657] Multiplicity of reference in the role `ProcessDesignToMachineDesignMapping.machineDesign`

Imposition time: `IT_SysDes`

⌈For each `ProcessDesignToMachineDesignMapping`, the reference in the role `machineDesign` shall exist.⌋

An intended `ProcessDesignToMachineDesignMapping` may not be possible for utilization of the target machine, and therefore a different `ProcessToMachineMapping` may be created in the target-configuration phase.

7.7 Ethernet Synchronized Time Design

As the *AUTOSAR adaptive platform* supports Ethernet as a communication network, time synchronization using Ethernet is also supported.

To configure the behavior of the Ethernet time synchronization, the specific sub-classes are used as shown in Figure 7.18.

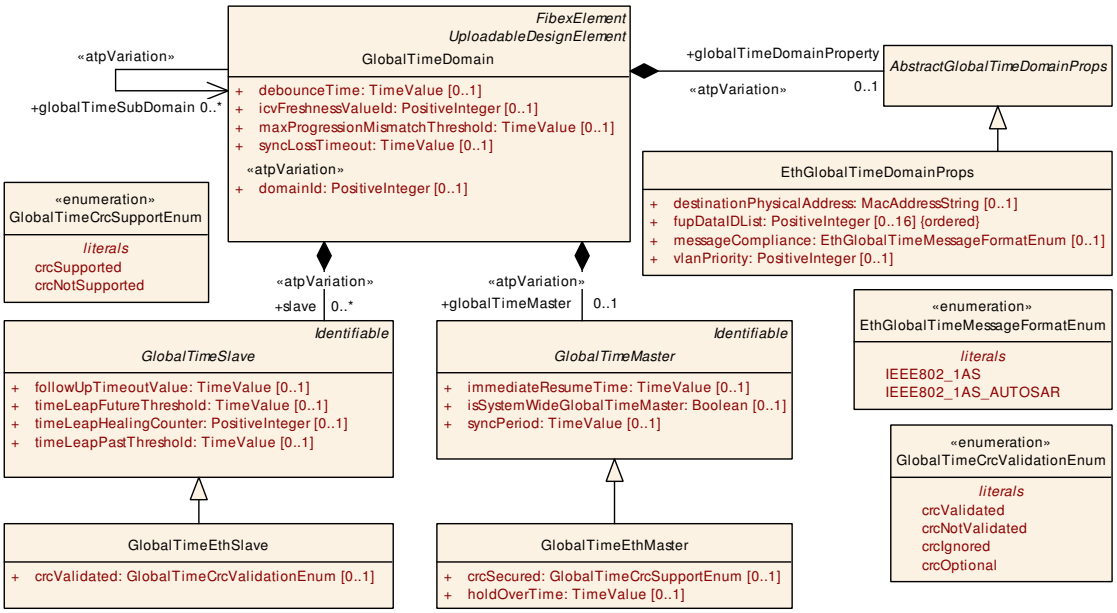


Figure 7.18: Modeling of Ethernet synchronized time

Class	**EthGlobalTimeDomainProps**			
Note	Enables the definition of Ethernet Global Time specific properties.			
Base	*ARObject*, *AbstractGlobalTimeDomainProps*			
Aggregated by	GlobalTimeDomain.globalTimeDomainProperty			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
crcFlags	EthTSynCrcFlags	0..1	aggr	Defines the fields of the message which shall be taken into account for CRC calculation and verification. **Tags:** atp.Splitkey=crcFlags
destination Physical Address	MacAddressString	0..1	attr	Defines the MAC multicast address the Ethernet time sync messages are communicated on.
fupDataIDList (ordered)	PositiveInteger	0..16	attr	The DataIDList for FUP messages to calculate CRC.
managed CouplingPort	EthGlobalTime ManagedCouplingPort	*	aggr	Collection of CouplingPorts which are managed in the scope of this Ethernet GlobalTimeDomain. **Tags:** atp.Splitkey=managedCouplingPort
message Compliance	EthGlobalTimeMessage FormatEnum	0..1	attr	Defines the compliance of the Ethernet time sync messages to specific standards.
vlanPriority	PositiveInteger	0..1	attr	Defines which VLAN priority shall be assigned to a time sync message in case the message is sent using a VLAN tag.

Table 7.58: EthGlobalTimeDomainProps

Class	**EthTSynCrcFlags**			
Note	Defines the fields of the message which shall be taken into account for CRC calculation and verification.			
Base	*ARObject*			
Aggregated by	EthGlobalTimeDomainProps.crcFlags			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	**EthTSynCrcFlags**			
crcCorrection Field	Boolean	0..1	attr	CorrectionField from the Follow_Up Message Header shall be included in CRC calculation.
crcDomain Number	Boolean	0..1	attr	DomainNumber from the Follow_Up Message Header shall be included in CRC calculation.
crcMessage Length	Boolean	0..1	attr	MessageLength from the Follow_Up Message Header shall be included in CRC calculation.
crcPrecise Origin Timestamp	Boolean	0..1	attr	PreciseOriginTimestamp from the Follow_Up Message Field shall be included in CRC calculation.
crcSequenceId	Boolean	0..1	attr	SequenceId from the Follow_Up Message Header shall be included in CRC calculation.
crcSourcePort Identity	Boolean	0..1	attr	SourcePortIdentity from the Follow_Up Message Header shall be included in CRC calculation.

Table 7.59: EthTSynCrcFlags

Primitive	**MacAddressString**
Note	This primitive specifies a Mac Address. Notation: FF:FF:FF:FF:FF:FF Alternative notations, e.g. using dash instead of colon, or another grouping of numbers, is not allowed. **Tags:** xml.xsd.customType=MAC-ADDRESS-STRING xml.xsd.pattern=([0-9a-fA-F]{2}:){5}[0-9a-fA-F]{2} xml.xsd.type=string

Table 7.60: MacAddressString

Class	**EthGlobalTimeManagedCouplingPort**			
Note	Specifies a CouplingPort which is managed by an Ethernet Global Time Domain.			
Base	*ARObject*			
Aggregated by	EthGlobalTimeDomainProps.managedCouplingPort			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
couplingPort	CouplingPort	0..1	ref	Defines which CouplingPort is managed by this EthGlobal TimeManagedCouplingPort. **Tags:** atp.Splitkey=couplingPort
globalTimePort Role	GlobalTimePortRole Enum	0..1	attr	This attribute defines the port behavior.
globalTimeTx Period	TimeValue	0..1	attr	This attribute defines the TX period in seconds
pdelayLatency Threshold	TimeValue	0..1	attr	Threshold for calculated Pdelay. If a measured Pdelay exceeds pdelayLatencyThreshold, the measured Pdelay value is discarded.
pdelayRequest Period	TimeValue	0..1	attr	Defines the period for the pdelay request messages.
pdelayRespAnd RespFollowUp Timeout	TimeValue	0..1	attr	Timeout value for Pdelay_Resp and Pdelay_Resp_ Follow_Up after a Pdelay_Req has been transmitted resp. a Pdelay_Resp has been received. A value of 0 or not defining this attribute deactivates this timeout observation.
pdelay Response Enabled	Boolean	0..1	attr	Defines whether PDELAY RESPONSE and PDELAY RESPONSE FOLLOW UP shall be sent on this Coupling Port.

Table 7.61: EthGlobalTimeManagedCouplingPort

Enumeration	**EthGlobalTimeMessageFormatEnum**
Note	Specifies which message formats are available to for the Ethernet time sync protocol.
Aggregated by	EthGlobalTimeDomainProps.messageCompliance
Literal	*Description*
IEEE802_1AS	Message format according to IEEE 802.1AS standard. **Tags:** atp.EnumerationLiteralIndex=0 xml.name=IEEE802-1AS
IEEE802_1AS_AUTOSAR	Message format according to IEEE 802.1AS standard with AUTOSAR extensions. **Tags:** atp.EnumerationLiteralIndex=1 xml.name=IEEE802-1AS-AUTOSAR

Table 7.62: EthGlobalTimeMessageFormatEnum

Class	**GlobalTimeEthSlave**			
Note	This represents the specialization of the GlobalTimeSlave for Ethernet communication.			
Base	*ARObject*, *GlobalTimeSlave*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	GlobalTimeDomain.slave			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
crcValidated	GlobalTimeCrcValidationEnum	0..1	attr	Definition of whether or not validation of the CRC is supported.

Table 7.63: GlobalTimeEthSlave

Enumeration	**GlobalTimeCrcValidationEnum**
Note	This enumeration provides values for the evaluation of the CRC
Aggregated by	GlobalTimeCanSlave.crcValidated, GlobalTimeEthSlave.crcValidated, GlobalTimeFrSlave.crcValidated
Literal	*Description*
crcIgnored	The CRC is supposed to be ignored **Tags:** atp.EnumerationLiteralIndex=0
crcNotValidated	The CRC is not supposed to be present. If CRC is present the message is ignored. **Tags:** atp.EnumerationLiteralIndex=1
crcOptional	Either the CRC is present and then shall be validated or the CRC is not present and no CRC check is done. **Tags:** atp.EnumerationLiteralIndex=3
crcValidated	This CRC is supposed to be validated. **Tags:** atp.EnumerationLiteralIndex=2

Table 7.64: GlobalTimeCrcValidationEnum

Class	**GlobalTimeEthMaster**			
Note	This represents the specialization of the GlobalTimeMaster for Ethernet communication.			
Base	*ARObject*, *GlobalTimeMaster*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	GlobalTimeDomain.globalTimeMaster			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
crcSecured	GlobalTimeCrcSupportEnum	0..1	attr	Definition of whether or not CRC is supported. This is only relevant for selected bus systems.

▽

△

Class	**GlobalTimeEthMaster**			
holdOverTime	TimeValue	0..1	attr	This attribute defines the timeout for transmission of Sync and Follow_Up messages on Master ports in absence of reception of Sync and Follow_Up messages on Slave port.
subTlvConfig	EthTSynSubTlvConfig	0..1	aggr	Defines the subTLV fields which shall be included in the time sync message. **Tags:** atp.Splitkey=subTlvConfig

Table 7.65: GlobalTimeEthMaster

Enumeration	**GlobalTimeCrcSupportEnum**
Note	This enumeration is used to define whether and how CRC on the TX side shall be utilized.
Aggregated by	GlobalTimeCanMaster.crcSecured, GlobalTimeEthMaster.crcSecured, GlobalTimeFrMaster.crc Secured
Literal	*Description*
crcNotSupported	This indicates that CRC is not supported **Tags:** atp.EnumerationLiteralIndex=0
crcSupported	This indicates that CRC is supported **Tags:** atp.EnumerationLiteralIndex=1

Table 7.66: GlobalTimeCrcSupportEnum

Class	**EthTSynSubTlvConfig**			
Note	Defines the subTLV fields which shall be included in the time sync message.			
Base	*ARObject*			
Aggregated by	GlobalTimeEthMaster.subTlvConfig			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
statusSubTlv	Boolean	0..1	attr	Defines whether an AUTOSAR Follow_Up TLV Status Sub-TLV is used.
timeSubTlv	Boolean	0..1	attr	Defines whether an AUTOSAR Follow_Up TLV Time Sub-TLV is used.
userDataSubTlv	Boolean	0..1	attr	Defines whether an AUTOSAR Follow_Up TLV UserData Sub-TLV is used.

Table 7.67: EthTSynSubTlvConfig

7.8 SOVD System Design

The SOVD standard [22, ISO-17978-3] foresees the usage of a [31, REST] API for the interaction between a diagnostic client and a diagnostic server.

Following the REST paradigm, SOVD implements a component architecture in which resources existing in the architecture can be addressed by means of a specifically crafted URI that is structured along the elements of the component architecture.

This means that support for the definition of the component architecture is required to craft a deterministic way to access resources that is shared between diagnostic client and diagnostic server.

The URI has a rough structure consisting of static parts and parts that are derived from a configuration model:

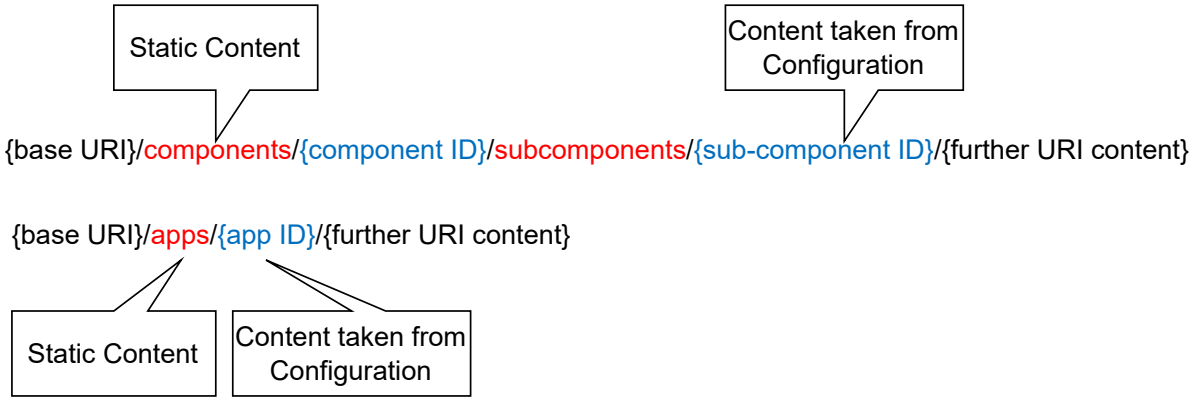


Figure 7.19: Example SOVD content URIs

It is very likely that the individual components of the component architecture are developed in a distributed way, which has an impact on the structure of the model. In particular, relations between parts of the component structure are modeled as reference instead of as aggregation.

This approach is very similar to the design philosophy of e.g. the AUTOSAR diagnostic extract ([3, TPS Diagnostic Extract Template]).

[TPS_MANI_01528] Root element of the SOVD system design ⌈The SOVD system design uses the meta-class `SovdSystemDesign` as the root element. The role of `SovdSystemDesign` for the SOVD system design is very similar to the role of the `DiagnosticContributionSet` in the context of the definition of the diagnostic extract.⌋

[TPS_MANI_01529] Access to diagnostic content from within a `SovdAbstractContributingEntity` ⌈Access to diagnostic content (typically formalized in the "further URI content" in Figure 7.19) associated with a subclass of `SovdAbstractContributingEntity` is possible by following the roles `softwareClusterDesign.diagnosticContribution`.⌋

[constr_10797] Reference in the role `SovdAbstractContributingEntity.softwareClusterDesign`

Imposition time: `IT_SovdSysDes`

⌈Each `SoftwareClusterDesign` shall only be referenced by a single `SovdAbstractContributingEntity` in the role `softwareClusterDesign`.⌋

It is possible for several `SoftwareClusterDesign`s to refer to the same `DiagnosticContributionSet`.

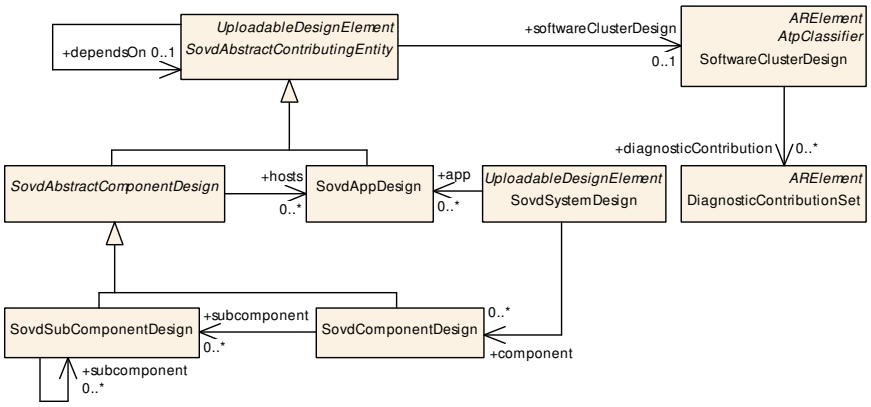


Figure 7.20: Modeling of the SOVD system design

[TPS_MANI_01530] Usage of `SovdAbstractContributingEntity.dependsOn`
⌈The AUTOSAR standard does not foresee a standardized meaning for the reference in the role `SovdAbstractContributingEntity.dependsOn`.

Such a reference has originally been foreseen by the SOVD standard [22, ISO-17978-3], albeit without a clear definition of its intended semantics.

This reference has only been introduced to AUTOSAR to give users the opportunity to utilize this role for a proprietary purpose.⌋

Class	***SovdAbstractContributingEntity*** (abstract)			
Note	This meta-class acts as an abstract base class for meta-classes that contribute to the structure of an SOVD system design.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Subclasses	*SovdAbstractComponentDesign*, SovdAppDesign			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dependsOn	SovdAbstract ContributingEntity	0..1	ref	This reference expresses a relationship between two contributing entities. The semantics of this reference is not defined by the AUTOSAR standard (or the SOVD standard, either) and can therefore be used for establishing a proprietary relation. **Tags:** atp.Splitkey=dependsOn This Attribute is only used by the AUTOSAR Adaptive Platform.
softwareCluster Design	SoftwareClusterDesign	0..1	ref	This reference identifies the "implementation" of the SOVD Contributing Entity. **Tags:** atp.Splitkey=softwareClusterDesign This Attribute is only used by the AUTOSAR Adaptive Platform.

Table 7.68: SovdAbstractContributingEntity

Class	**SovdAbstractComponentDesign** (abstract)			
Note	This meta-class acts as an abstract base class for the component structure of an SOVD system design. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SovdAbstractContributingEntity*, *UploadableDesignElement*, *UploadablePackageElement*			
Subclasses	SovdComponentDesign, SovdSubComponentDesign			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
hosts	SovdAppDesign	*	ref	This reference indicates that the SOVD App is hosted on the enclosing abstract SOVD component **Tags:** atp.Splitkey=hosts

Table 7.69: SovdAbstractComponentDesign

Class	**SovdAppDesign**			
Note	This meta-class represents an SOVD App **Tags:** atp.recommendedPackage=SovdAppDesigns This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SovdAbstractContributingEntity*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 7.70: SovdAppDesign

Class	**SovdComponentDesign**			
Note	This meta-class represents an SOVD component. **Tags:** atp.recommendedPackage=SovdComponentDesigns This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SovdAbstractComponentDesign*, *SovdAbstractContributingEntity*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
subcomponent	SovdSubComponentDesign	*	ref	This reference represents the collection of SOVD subcomponents located directly under the SOVD Component. **Tags:** atp.Splitkey=subcomponent

Table 7.71: SovdComponentDesign

Class	**SovdSubComponentDesign**
Note	This meta-class represents an SOB sub-component that itself can have sub-components all the way down. **Tags:** atp.recommendedPackage=SovdSubComponentDesigns This Class is only used by the AUTOSAR Adaptive Platform.



△

Class	**SovdSubComponentDesign**			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *SovdAbstractComponentDesign*, *SovdAbstractContributingEntity*, *Uploadable DesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
subcomponent	SovdSubComponent Design	*	ref	This reference allows for the creation of an arbitrarily nested hierarchy of subcomponents. **Tags:** atp.Splitkey=subcomponent

Table 7.72: SovdSubComponentDesign

Class	**SovdSystemDesign**			
Note	This meta-class represents the root element of a an SOVD system design, the scope of which i typically limited to the enclosing vehicle. **Tags:** atp.recommendedPackage=SovdSystemDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
app	SovdAppDesign	*	ref	This reference represents the collection of SOVD Apps in the overall SOVD system design. **Tags:** atp.Splitkey=app This Attribute is only used by the AUTOSAR Adaptive Platform.
component	SovdComponentDesign	*	ref	This reference represents the collection of SOVD Components in the overall SOVD system design. **Tags:** atp.Splitkey=component This Attribute is only used by the AUTOSAR Adaptive Platform.

Table 7.73: SovdSystemDesign

8 Sub-System Design

8.1 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 needs 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 subcontracted 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 the 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.

8.2 Software Cluster Design

The creation of a `SoftwareCluster` (see Chapter 16.3) represents a complex workflow that has to be done in the target-configuration phase of the overall methodology for the *AUTOSAR adaptive platform*.

In order to reduce the complexity of this step in the workflow, AUTOSAR supports a front-loading approach for the definition of a `SoftwareCluster`, the modeling of `SoftwareClusterDesign`. In this context, the term “front-loading” means that some model elements defined in the context of a `SoftwareClusterDesign` can be (directly) taken over for the creation of a corresponding `SoftwareCluster`.

The membership of model elements to a given SoftwareClusterDesign is expressed by means of references to the respective model elements. The referenced model elements are “claimed” by the referencing SoftwareClusterDesign.

The pattern established by these references is very similar to the pattern established by means of the modeling of System.fibexElement or DiagnosticContributionSet.element.

The modeling of SoftwareClusterDesign prioritizes the usage of references over the usage of aggregations because the migration from SoftwareClusterDesign to SoftwareCluster is much facilitated as no actual copying of model elements is required; only the content of references has to be taken over in the migration.

[TPS_MANI_01112] Semantics of SoftwareClusterDesign ⌈The 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.

On the other hand, the SoftwareClusterDesign also imposes a contract with respect to certain model elements. In other words, the requirements imposed by an OEM shall be fulfilled in the downstream processing.⌋

As an example for the contractual relevance of a SoftwareClusterDesign, if a SoftwareCluster references a given SoftwareClusterDesign in the role design, then any Process that is referenced by the SoftwareCluster in the role containedProcess shall also refer to a ProcessDesign that is aggregated by the referenced SoftwareClusterDesign.

[constr_1560] Usage of SoftwareClusterDesign.requiredARElement

Imposition time: IT_SubSysDes

⌈The reference SoftwareClusterDesign.requiredARElement shall not be used to refer to another SoftwareClusterDesign or even SoftwareCluster.⌋

[TPS_MANI_01369] Semantics of reference SoftwareClusterDesign.requiredARElement ⌈The reference SoftwareClusterDesign.requiredARElement is used to identify all model elements that are typed by a sub-class of meta-class ARElement. This way, it is possible to express which model elements belong to the enclosing SoftwareClusterDesign.⌋

[TPS_MANI_01211] Specification of executable software within SoftwareClusterDesign ⌈One 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 SoftwareClusterDesign.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 Executable.⌋

[TPS_MANI_01331] Standardized values of attribute SoftwareClusterDesign.category and SoftwareCluster.category ⌈AUTOSAR standardizes the following values of attribute SoftwareClusterDesign.category and SoftwareCluster.category.

- **PLATFORM_CORE**: a SoftwareClusterDesign/SoftwareCluster of this category typically represents any kind of platform software, e.g. bootloader, hypervisor, OS, adaptive platform module.

  Such a SoftwareClusterDesign/SoftwareCluster cannot be removed by a UCM, but updates are possible. It is not forbidden that such a SoftwareClusterDesign/SoftwareCluster contains application-level software.

- **PLATFORM**: a SoftwareClusterDesign/SoftwareCluster of this category represents the parts of the platform software (e.g. configuration of functional clusters) that could be installed, removed, and updated.

- **APPLICATION_LAYER**: a SoftwareClusterDesign/SoftwareCluster of this category represents a driving-relevant function on application level, e.g. a lane keeping assistant, window lift controller, seat positioning. Such a SoftwareClusterDesign/SoftwareCluster can be installed, removed, and updated.

⌋

[TPS_MANI_01410] Semantics of attribute SoftwareClusterDesign.installationBehavior ⌈Attribute SoftwareClusterDesign.installationBehavior determines the behavior of a derived SoftwareCluster in terms of whether it will be allowed to remove a derived SoftwareCluster from the target machine.

The attribute SoftwareClusterDesign.installationBehavior actually applies to a derived SoftwareCluster. But the definition of the installation behavior is considered an important design decision (and is thus made during the creation of a SoftwareClusterDesign) that cannot be left to the considerations on target configuration level (where actual SoftwareClusters are made).

The decision to declare a SoftwareClusterDesign non-removable is most likely limited to SoftwareClusterDesigns that correspond to SoftwareClusters with a significant share of platform-level software.⌋

[constr_10410] Value of SoftwareClusterDesign.installationBehavior for a SoftwareClusterDesign of category PLATFORM_CORE

Imposition time: IT_Mani

⌈In a SoftwareClusterDesign of category PLATFORM_CORE, the attribute installationBehavior shall exist and its value shall be set to cannotBeRemoved.⌋

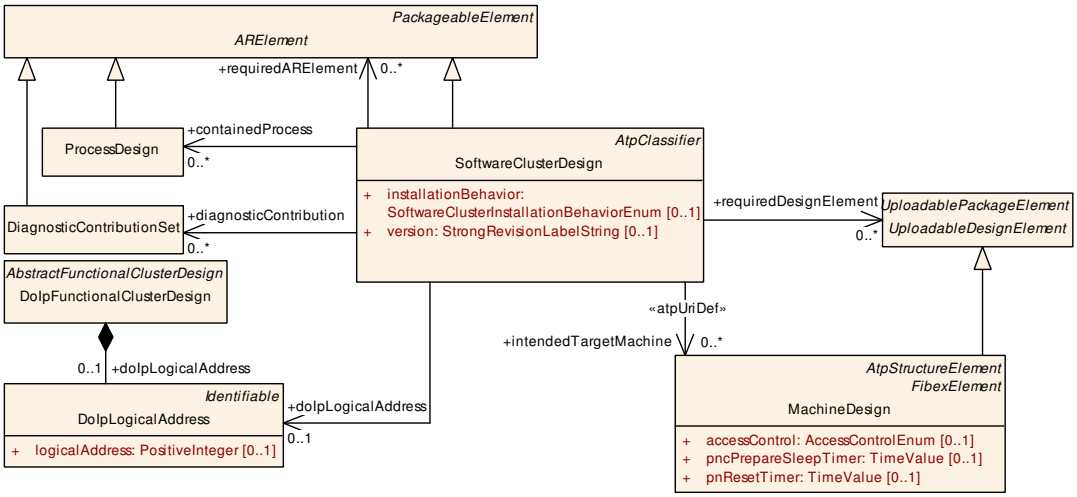


Figure 8.1: Modeling of SoftwareClusterDesign

Class	**SoftwareClusterDesign**			
Note	This meta-class represents the ability for the OEM to design the grouping of software uploadable to a specific target Machine. **Tags:** atp.recommendedPackage=SoftwareClusterDesigns This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
conflictsTo	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation handles conflicts. If it yields true then the SoftwareClusterDesign shall not be installed. **Tags:** atp.Splitkey=conflictsTo
contained Process	ProcessDesign	*	ref	This reference represent the ProcessDesigns contained in the enclosing SoftwareCluster. **Tags:** atp.Splitkey=containedProcess
dependsOn	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation can be taken to identify a dependency for the enclosing SoftwareClusterDesign. **Tags:** atp.Splitkey=dependsOn
diagnostic Contribution	DiagnosticContribution Set	*	ref	This reference identifies the corresponding collection of DiagnosticContributionSet. **Tags:** atp.Splitkey=diagnosticContribution

△

Class	**SoftwareClusterDesign**			
doIpLogical Address	DoIpLogicalAddress	0..1	ref	This reference identifies the DoIp logical address that shall be used to contact the diagnostic management in the SoftwareCluster derived from the enclosing Software ClusterDesign. **Tags:** atp.Splitkey=doIpLogicalAddress
installation Behavior	SoftwareCluster InstallationBehavior Enum	0..1	attr	This attribute controls the behavior of the SoftwareCluster in terms of installation.
intendedTarget Machine	MachineDesign	*	ref	This reference can be taken to identify the Machine Design for which the final SoftwareCluster shall be developed. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=intendedTargetMachine
required ARElement	ARElement	*	ref	This reference represents the collection of ARElements that are required for the completeness of the definition of the SoftwareCluster. **Tags:** atp.Splitkey=requiredARElement
requiredDesign Element	UploadableDesign Element	*	ref	This reference points to uploadable design elements that have been identified as relevant in the context of the enclosing SoftwareClusterDesign. **Tags:** atp.Splitkey=requiredDesignElement
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.Splitkey=rootComposition, root Composition.shortName
version	StrongRevisionLabel String	0..1	attr	This attribute can be used to describe a version information for the enclosing SoftwareClusterDesign.

Table 8.1: SoftwareClusterDesign

Enumeration	**SoftwareClusterInstallationBehaviorEnum**
Note	This enumeration defines possible approaches for the installation behavior of a SoftwareCluster. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	SoftwareClusterDesign.installationBehavior
Literal	***Description***
canBeRemoved	The enclosing SoftwareCluster can be removed from the target Machine or updated with a newer version. **Tags:** atp.EnumerationLiteralIndex=0
cannotBeRemoved	The enclosing SoftwareCluster cannot be removed from the target Machine. It can only be updated with a newer version. **Tags:** atp.EnumerationLiteralIndex=1

Table 8.2: SoftwareClusterInstallationBehaviorEnum

[constr_3727] Upper multiplicity of reference in the role `SoftwareClusterDesign.intendedTargetMachine`

Imposition time: `IT_SubSysDes`

⌈In the context of `SoftwareClusterDesign`, the reference in the role `intendedTargetMachine` shall exist at most once.⌋

[TPS_MANI_01117] Semantics of SoftwareClusterDesign.intendedTargetMachine ⌈The 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.⌋

SoftwareCluster does not 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] Relation between meta-classes SoftwareClusterDesign and DiagnosticContributionSet ⌈An important aspect of the definition of a SoftwareClusterDesign is the question which 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.⌋

[TPS_MANI_01189] Software Cluster and DiagnosticContributionSet.category ⌈A DiagnosticContributionSet used in the context of a SoftwareCluster shall set the value of attribute category to DIAGNOSTICS_SWCL_EXTRACT.⌋

The difference between SoftwareCluster and SoftwareClusterDesign in terms of the relation to DiagnosticContributionSet is intentional.

In other words, the multiplicity of the references to DiagnosticContributionSet intentionally differs.

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] Reference to model elements from SoftwareClusterDesign ⌈SoftwareClusterDesign 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 UploadableDesignElement are formalized by way of SoftwareClusterDesign.requiredDesignElement.
- references to meta-classes derived from ARElement are formalized by way of SoftwareClusterDesign.requiredARElement.

⌋

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 of time.

8.3 Relevance of Software Cluster Design for Diagnostics

It is possible to define several diagnostics-related configuration elements on design level and within the scope of the SoftwareClusterDesign. One example is the definition of the logical address used by the DoIP communication.

[TPS_MANI_01465] Semantics of reference SoftwareClusterDesign.doIpLogicalAddress ⌈The definition of the logical address of the *Diagnostic Management* associated with a given SoftwareClusterDesign needs to be clarified in an early stage of the project workflow and represents information that is typically defined in an OEM-role.

In the meta-model, this relation is established by the role SoftwareClusterDesign.doIpLogicalAddress.⌋

8.4 Software Cluster Dependency

8.4.1 Overview

The installation or update of a given SoftwareCluster (see Chapter 16.3) may be subject to dependencies on other SoftwareClusters. This means that the installation or update may only be permitted if dedicated version requirements are fulfilled.

The definition of dependencies among SoftwareClusters is exclusively done on the design level, i.e. attached to SoftwareClusterDesign. The latter is generally not mandatory, i.e. SoftwareClusters can be created without a reference to SoftwareClusterDesign.

But if dependencies among SoftwareClusters are needed, then the definition of related SoftwareClusterDesigns cannot be avoided.

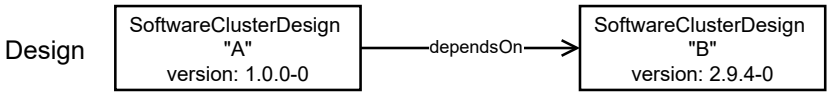


Figure 8.2: Definition of dependencies among SoftwareClusterDesign

This could even mean that SoftwareClusterDesigns are created for the sole purpose of expressing dependencies. This might look like an overshoot, but the ability to

analyze and understand the dependencies among SoftwareClusters on the design level is considered an important asset that cannot be delayed to the target configuration phase.

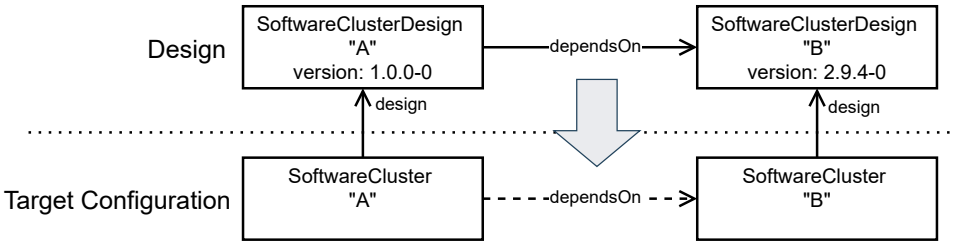


Figure 8.3: Dependencies on the level of SoftwareClusterDesign are reflected on the level of SoftwareClusters

The dashed arrow indicates that the dependency is “inherited” rather than explicitly modeled on the target configuration level.

8.4.2 Modeling of a Software Cluster Dependency

The modeling of a dependency among SoftwareClusterDesigns is formulated by meta-class SoftwareClusterDependencyFormula.

[TPS_MANI_01479] Semantics of meta-class SoftwareClusterDependency-Formula ⌈Meta-class SoftwareClusterDependencyFormula allows for the definition of a formal condition that can be taken to decide about the dependency to or the conflict with a SoftwareCluster.

The modeling of SoftwareClusterDependencyFormula allows for the definition of nested conditions. The attribute operator is applied on the results of the evaluation of the parts.⌋

Class	**SoftwareClusterDependencyFormula**			
Note	This meta-class represents the ability to define a dependency among SoftwareClusters. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *SoftwareClusterDependencyFormulaPart*			
Aggregated by	SoftwareClusterDependencyFormula.part, SoftwareClusterDesign.conflictsTo, SoftwareClusterDesign.dependsOn			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
operator	SoftwareClusterDependencyLogicalOperatorEnum	0..1	attr	This logical operator can be used to relate the results of different SoftwareClusterDependencyParts.
part (ordered)	SoftwareClusterDependencyFormulaPart	*	aggr	This aggregation represents the ordered collection of the parts of the SoftwareClusterDependencyFormula.

Table 8.3: SoftwareClusterDependencyFormula

Enumeration	**SoftwareClusterDependencyLogicalOperatorEnum**
Note	This enumeration provides a set of operators to be used in a SoftwareClusterDependencyFormula. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	SoftwareClusterDependencyFormula.operator
Literal	***Description***
logicalAnd	logical and **Tags:** atp.EnumerationLiteralIndex=0
logicalOr	logical or **Tags:** atp.EnumerationLiteralIndex=1

Table 8.4: SoftwareClusterDependencyLogicalOperatorEnum

[TPS_MANI_01216] Semantics of meta-class `SoftwareClusterDependencyFormulaPart` ⌈Meta-class `SoftwareClusterDependencyFormulaPart` represents a part of a `SoftwareClusterDependencyFormula`. The order of the parts of a `SoftwareClusterDependencyFormula` is significant.⌋

Class	***SoftwareClusterDependencyFormulaPart*** (abstract)			
Note	This meta-class represents an abstract base class for the definition of different formula parts of a SoftwareClusterDependencyFormula. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Subclasses	SoftwareClusterDependencyCompareCondition, SoftwareClusterDependencyFormula			
Aggregated by	SoftwareClusterDependencyFormula.part			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 8.5: SoftwareClusterDependencyFormulaPart

At the same time, `SoftwareClusterDependencyFormulaPart` is the base class of `SoftwareClusterDependencyFormula`.

This means that the `SoftwareClusterDependencyFormula` can aggregate all subclasses of `SoftwareClusterDependencyFormulaPart`, i.e. `SoftwareClusterDependencyFormula` and `SoftwareClusterDependencyCompareCondition`.

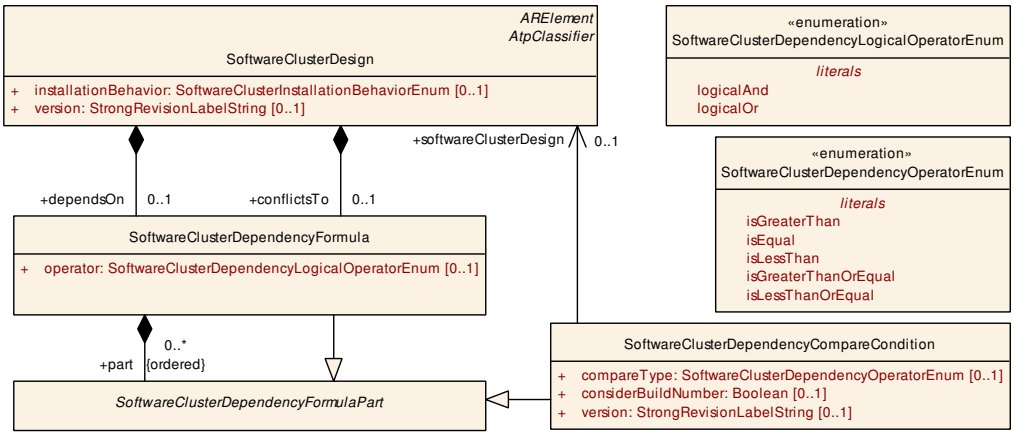


Figure 8.4: Modeling of dependencies in the context of a `SoftwareCluster`

[TPS_MANI_01480] Semantics of `SoftwareClusterDesign.dependsOn` ⌈A `SoftwareClusterDesign` has the ability to express a dependency to other `SoftwareCluster`s in the role `dependsOn`.

The semantics is to express that the functionality of the `SoftwareCluster` that references (in the role `design`) the enclosing `SoftwareClusterDesign` **depends on** the installation of the `SoftwareCluster` that references (in the role `design`) the `SoftwareClusterDesign` referenced in the role `SoftwareClusterDependencyCompareCondition.softwareClusterDesign`.

Attribute `SoftwareClusterDesign.dependsOn` allows for the definition of a **formal** (potentially nested) dependency condition. The dependency shall be applicable only if the condition defined by `dependsOn` yields `true`.⌋

[TPS_MANI_01481] Semantics of meta-class `SoftwareClusterDependencyCompareCondition` ⌈Meta-class `SoftwareClusterDependencyCompareCondition` allows for the definition of a formal condition to compare against the version of the referenced `softwareClusterDesign` using a given `compareType`.

The ability to decide whether to consider the build number in the comparison is implemented by means of attribute `considerBuildNumber`.⌋

Class	**SoftwareClusterDependencyCompareCondition**			
Note	This meta-class represents the ability to specify a concrete dependency condition in the context of a SoftwareClusterDependencyFormula. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *SoftwareClusterDependencyFormulaPart*			
Aggregated by	SoftwareClusterDependencyFormula.part			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compareType	SoftwareCluster DependencyOperator Enum	0..1	attr	This attribute identifies the semantics of the compare operator.
considerBuild Number	Boolean	0..1	attr	If this attribute is set to true then the build number shall be taken into account for the comparison. Build numbers don't have to be consecutive but could be created by some kind of hashing algorithm. In such a case it might make sense to include the build number in a test for equality but it is probably not reasonable to apply e.g. a less-than comparison.
softwareCluster Design	SoftwareClusterDesign	0..1	ref	This reference identifies the SoftwareClusterDesign to which the dependency/conflict applies.
version	StrongRevisionLabel String	0..1	attr	This attribute represents the value of a version against which the comparison shall be executed.

Table 8.6: SoftwareClusterDependencyCompareCondition

[constr_10237] Multiplicity of attribute `SoftwareClusterDependencyCompareCondition.compareType`

Imposition time: `IT_Mani`

⌈For each `SoftwareClusterDependencyCompareCondition`, the attribute `compareType` shall exist.⌋

[constr_10253] Multiplicity of attribute SoftwareClusterDependencyCompareCondition.considerBuildNumber

Imposition time: IT_Mani

⌈For each SoftwareClusterDependencyCompareCondition, the attribute considerBuildNumber shall exist.⌋

[constr_10254] Multiplicity of attribute SoftwareClusterDependencyCompareCondition.version

Imposition time: IT_Mani

⌈For each SoftwareClusterDependencyCompareCondition, the attribute version shall exist.⌋

Enumeration	**SoftwareClusterDependencyOperatorEnum**
Note	This enumeration provides a choice of operators for comparison within a SoftwareCluster DependencyCompareCondition. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	SoftwareClusterDependencyCompareCondition.compareType
Literal	***Description***
isEqual	equal **Tags:** atp.EnumerationLiteralIndex=1
isGreaterThan	greater than **Tags:** atp.EnumerationLiteralIndex=0
isGreaterThanOr Equal	greater than or equal **Tags:** atp.EnumerationLiteralIndex=3
isLessThan	less than **Tags:** atp.EnumerationLiteralIndex=2
isLessThanOrEqual	less than or equal **Tags:** atp.EnumerationLiteralIndex=4

Table 8.7: SoftwareClusterDependencyOperatorEnum

[TPS_MANI_01482] Semantics of SoftwareClusterDesign.conflictsTo ⌈A SoftwareClusterDesign has the ability to express a conflict to other SoftwareClusters in the role conflictsTo.

The semantics is to express that the functionality of the SoftwareCluster that references (in the role design) the enclosing SoftwareClusterDesign **inhibits** the installation of the SoftwareCluster that references (in the role design) the SoftwareClusterDesign referenced in the role SoftwareClusterDependencyCompareCondition.softwareClusterDesign.

Attribute SoftwareClusterDesign.conflictsTo allows for the definition of a **formal** (potentially nested) dependency condition. The dependency shall be applicable only if the condition defined by conflictsTo yields false.⌋

An example definition of dependencies among SoftwareClusters can be found in Chapter B.11.1.

8.5 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[1], 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.

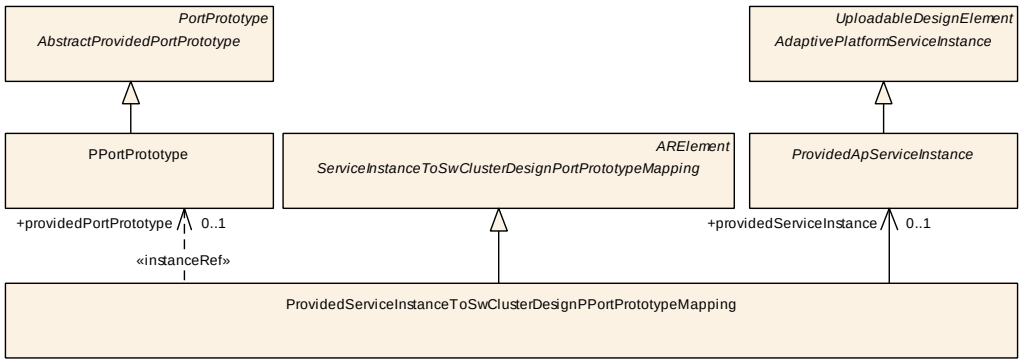


Figure 8.5: Modeling of the ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping

For the specific case of the RootSwClusterDesignComponentPrototype, the referenced SwComponentType is expected to represent a CompositionSwComponentType without any further detailing. Detailing is unnecessary because the only purpose is the exposure of PortPrototypes to which AdaptivePlatformServiceInstances can be mapped.

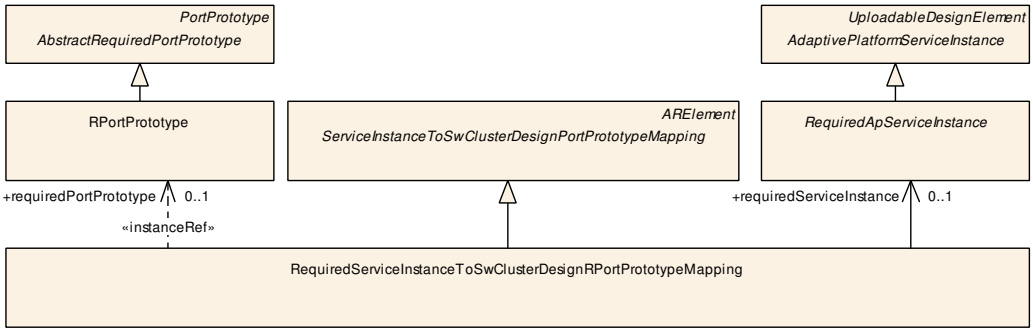


Figure 8.6: Modeling of the RequiredServiceInstanceToSwClusterDesignRPortPrototypeMapping

A dedicated mapping class, ServiceInstanceToSwClusterDesignPortPrototypeMapping, is defined to support the creation of the described relation between PortPrototype and AdaptivePlatformServiceInstance.

[TPS_MANI_01275] Semantics of meta-class ServiceInstanceToSwClusterDesignPortPrototypeMapping ⌈The software-component used to type the

[1] For more information, see Chapter 16.3.

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.⌋

In Figure 8.7, 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 Chapter 8.6.

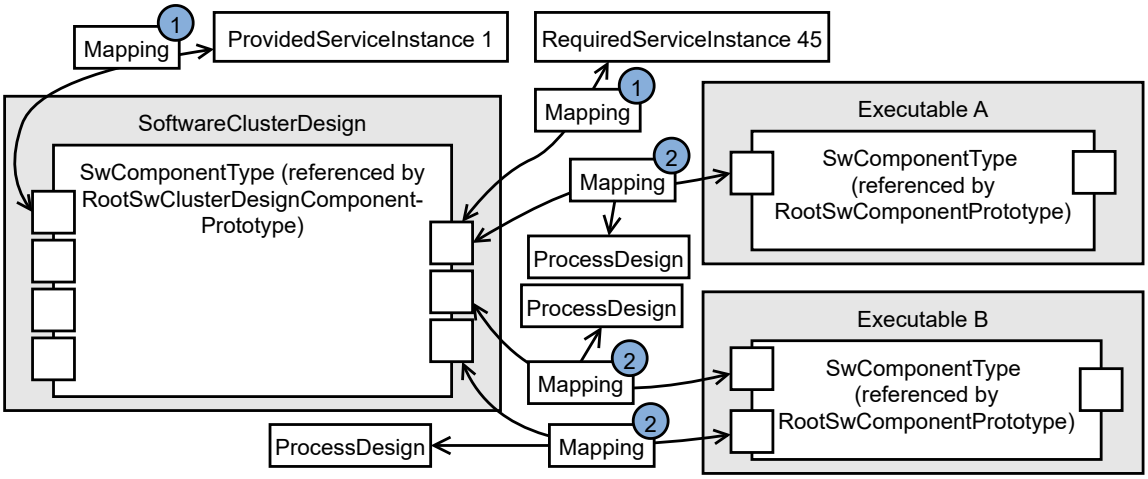


Figure 8.7: Modeling of mappings in the context of SoftwareClusterDesign

Class	**RootSwClusterDesignComponentPrototype**			
Note	This meta-class represents the ability to define the service endpoints in the scope of a SwClusterDesign. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SoftwareClusterDesign.rootComposition			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
applicationType	SwComponentType	0..1	tref	This SwComponentType acts as the Type of the RootSw ClusterDesignComponentPrototype. **Stereotypes:** isOfType **Tags:** atp.Splitkey=applicationType

Table 8.8: RootSwClusterDesignComponentPrototype

The existence of the reference in the role RootSwClusterDesignComponentPrototype.applicationType is governed by [constr_2633].

[advisory_01008] Multiplicity of reference in the role RootSwClusterDesignComponentPrototype.applicationType

Imposition time: IT_SubSysDes

⌈For each RootSwClusterDesignComponentPrototype, the reference in the role applicationType should exist.⌋

Class	***ServiceInstanceToSwClusterDesignPortPrototypeMapping*** (abstract)			
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. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping, RequiredServiceInstanceToSw ClusterDesignRPortPrototypeMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 8.9: ServiceInstanceToSwClusterDesignPortPrototypeMapping

Class	**RequiredServiceInstanceToSwClusterDesignRPortPrototypeMapping**			
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.recommendedPackage=ServiceInstanceToSwClusterDesignPortPrototypeMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInstanceToSwClusterDesignPortPrototypeMapping*			
Aggregated by	ARPackage.element			
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.Splitkey=requiredPortPrototype, requiredPort Prototype.contextRootSwClusterDesignComponent Prototype, requiredPortPrototype.contextSwComponent Prototype, requiredPortPrototype.targetRPortPrototype **InstanceRef implemented by:** RPortPrototypeIn SoftwareClusterDesignInstanceRef
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.Splitkey=requiredServiceInstance

Table 8.10: RequiredServiceInstanceToSwClusterDesignRPortPrototypeMapping

Class	**ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping**			
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.recommendedPackage=ServiceInstanceToSwClusterDesignPortPrototypeMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInstanceToSwClusterDesignPortPrototypeMapping*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping**			
providedPort Prototype	PPortPrototype	0..1	iref	This reference identifies the applicable PortPrototype in the scope of the SwClusterDesign. **Tags:** atp.Splitkey=providedPortPrototype, providedPort Prototype.contextRootSwClusterDesignComponent Prototype, providedPortPrototype.contextSwComponent Prototype, providedPortPrototype.targetPPortPrototype **InstanceRef implemented by:** PPortPrototypeIn SoftwareClusterDesignInstanceRef
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.Splitkey=providedServiceInstance

Table 8.11: ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping

8.6 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.

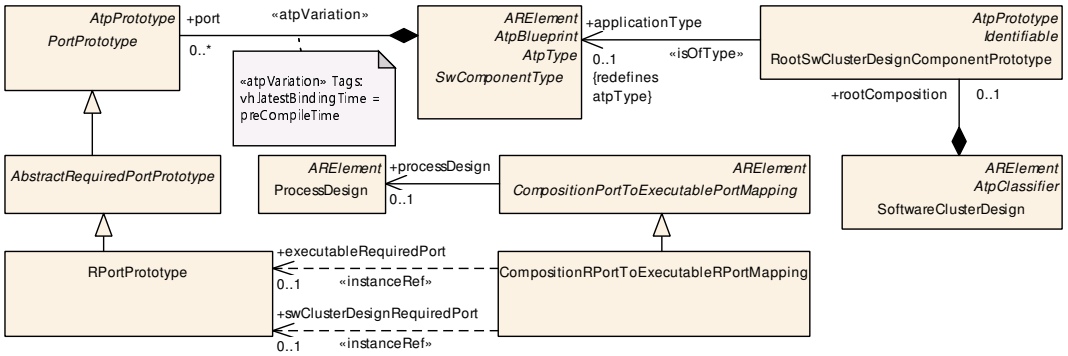


Figure 8.8: Modeling of the RootSwClusterDesignComponentPrototype and the CompositionRPortToExecutableRPortMapping

[TPS_MANI_01276] Semantics of CompositionRPortToExecutableRPortMapping and CompositionPPortToExecutablePPortMapping ⌈In 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.⌋

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 Chapter 13.3.

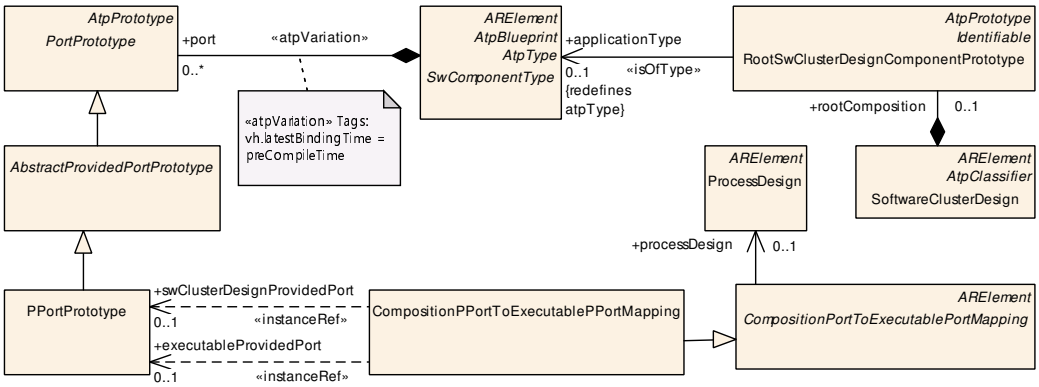


Figure 8.9: Modeling of the RootSwClusterDesignComponentPrototype and the CompositionPPortToExecutablePPortMapping

[TPS_MANI_01282] Semantics of reference CompositionPortToExecutablePortMapping.processDesign ⌈The reference in the role 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.⌋

The statement made by [TPS_MANI_01282] is further explained in Figure 8.7. Two CompositionPortToExecutablePortMapping refer to the same PortPrototype on the surface of the Executable B.

Each of these CompositionPortToExecutablePortMappings refers to a different ProcessDesign.

This means that, at run-time, the two CompositionPortToExecutablePortMappings apply to different instances of the Executable B launched as different Processes (that each, in turn, refer to one of the ProcessDesigns referenced by the Executable B).

Class	***CompositionPortToExecutablePortMapping*** (abstract)			
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. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	CompositionPPortToExecutablePPortMapping, CompositionRPortToExecutableRPortMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	***CompositionPortToExecutablePortMapping*** (abstract)			
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.Splitkey=processDesign

Table 8.12: CompositionPortToExecutablePortMapping

Class	**CompositionRPortToExecutableRPortMapping**			
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.recommendedPackage=CompositionPortToExecutablePortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *CompositionPortToExecutablePortMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
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.Splitkey=executableRequiredPort, executable RequiredPort.contextRootSwComponentPrototype, executableRequiredPort.contextComponentPrototype, executableRequiredPort.targetRPortPrototype **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.Splitkey=swClusterDesignRequiredPort, sw ClusterDesignRequiredPort.contextRootSwClusterDesign ComponentPrototype, swClusterDesignRequired Port.contextSwComponentPrototype, swClusterDesign RequiredPort.targetRPortPrototype **InstanceRef implemented by:** RPortPrototypeIn SoftwareClusterDesignInstanceRef

Table 8.13: CompositionRPortToExecutableRPortMapping

Class	**CompositionPPortToExecutablePPortMapping**			
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.recommendedPackage=CompositionPortToExecutablePortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *CompositionPortToExecutablePortMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	CompositionPPortToExecutablePPortMapping			
executable ProvidedPort	PPortPrototype	0..1	iref	This reference identifies the applicable PortPrototype in the context on an Executable. **Tags:** atp.Splitkey=executableProvidedPort, executableProvidedPort.contextRootSwComponentPrototype, executableProvidedPort.contextComponentPrototype, executableProvidedPort.targetPPortPrototype **InstanceRef implemented by:** PPortPrototypeInExecutableInstanceRef
swCluster DesignProvided Port	PPortPrototype	0..1	iref	This reference identifies the applicable PPortPrototype in the context of the SwClusterDesign. **Tags:** atp.Splitkey=swClusterDesignProvidedPort, swClusterDesignProvidedPort.contextRootSwClusterDesignComponentPrototype, swClusterDesignProvidedPort.contextSwComponentPrototype, swClusterDesignProvidedPort.targetPPortPrototype **InstanceRef implemented by:** PPortPrototypeInSoftwareClusterDesignInstanceRef

Table 8.14: CompositionPPortToExecutablePPortMapping

9 Signal-based Communication

9.1 Overview

The applications on the adaptive platform communicate with each other in a service-oriented manner. But there is also a use case where applications on the *AUTOSAR adaptive platform* need to communicate with software-components running on the *AUTOSAR classic platform*.

If the remote ECU on the *AUTOSAR classic platform* communicates via SOME/IP in a service-oriented manner and uses the SOME/IP transformer to serialize its data, then the communication with the `Machine` on the *AUTOSAR adaptive platform* can be established directly without any adaptations of either the ECU or the `Machine`.

If the counterpart on the *AUTOSAR classic platform* ECU communicates using signal-based communication over, e.g., CAN or FlexRay, the translation of the signal-based content into `ServiceInterface`s needs to be established. The preconditions for this use-case are defined in Chapter 9.2.

Such a `signal/service translation` may happen in a Gateway that is implemented on an ECU on the *AUTOSAR classic platform*. Such a solution is out of scope of this document since it is handled using the *AUTOSAR classic platform* configuration means. This approach is defined in the [4, TPS System Template] for the Classic platform.

It is up to the vehicle architecture design to choose whether the `signal/service translation` shall be implemented on a Classic platform ECU or on an Adaptive platform Machine.

Another alternative for this translation is to happen directly on the `Machine` on the *AUTOSAR adaptive platform* by an Application that is running in the Process, as sketched in Figure 9.1.

This Application communicates with other applications on the *AUTOSAR adaptive platform* in the service-oriented way over ara::com; but it is also able to transmit and receive `ISignal`s as well as communicate signal-based with remote ECUs on the *AUTOSAR classic platform*.

In order to make this possible, software that conforms to the specification of the COM stack on the *AUTOSAR classic platform* needs to be executed on the `Machine` on the *AUTOSAR adaptive platform*.

For the configuration of this software, the System Description based on the System Template on the *AUTOSAR classic platform* is used that contains a communication matrix description with `Pdu`s and `ISignal`s.

This chapter introduces a modeling that creates a bridge between the service-oriented communication based on `ServiceInterface`s of the *AUTOSAR adaptive platform* and the signal-based communication involving the definition of `Pdu`s and `ISignal`s that are used on the *AUTOSAR classic platform*.

The signal/service translation mapping, together with the *AUTOSAR classic platform* System Description, allows to configure the communication between a Machine on the *AUTOSAR adaptive platform* and an ECU on the *AUTOSAR classic platform*. In a setup like the one sketched in Figure 9.1, the *AUTOSAR classic platform* System Description would also contain a Pdu or Signal Gateway configuration between the Ethernet and the CAN network to forward the PDUs between the networks.

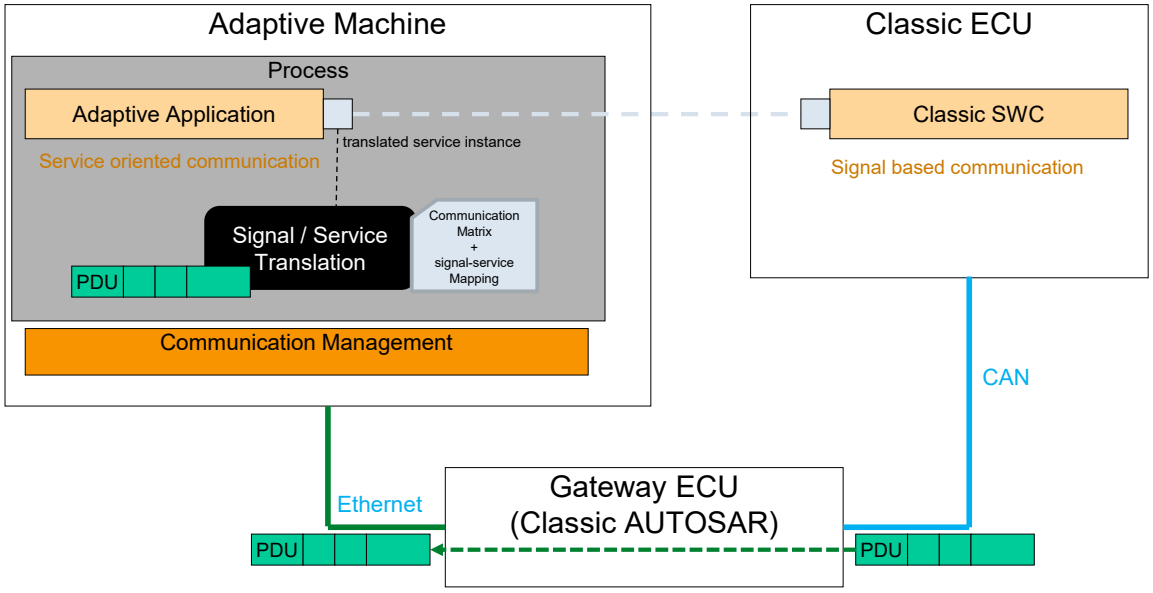


Figure 9.1: Signal-service-translation in Application on Adaptive Machine

The *translated service instance* is accessed from the application software using the ara::com API. The translation is designed as a network binding similar to the binding of SOME/IP. So the communication direction is

- if the signal-based payload is received by the Machine then the application software has a RPortPrototype
- if the signal-based payload is sent by the Machine then the application software has a PPortPrototype.

Further, the transport of the signal-based information via an IEEE1722 stream encapsulated as IEEE1722 ACF messages is supported. In this case, the Machine receives or sends an IEEE1722 stream of subtype TSCF or NTSCF according to [19, IEEE 1722-2016]. CAN or LIN frames which may contain signal-based ISignalIPdus are encapsulated in IEEE1722 ACF messages. Using this information, it is possible for the signal/service translation to gain access to the included signals.

An example setup for an IEEE1722 stream which transports a bus frame encapsulated as IEEE1722 ACF message is illustrated in Figure 9.2. In this sketch, a *CAN frame* is sent by a *Classic ECU* and received at the *IEEE1722 Stream Data Producer*. The *IEEE1722 Stream Data Producer* transfers the *CAN frame* to an IEEE1722 ACF message and puts it inside an *IEEE1722 Stream*.

The *Adaptive Machine* consumes the *IEEE1722 Stream* with an `IEEE1722 ACF message`, fetches the *CAN frame*, and performs the `signal/service translation` on the contained `ISignalIPdu`.

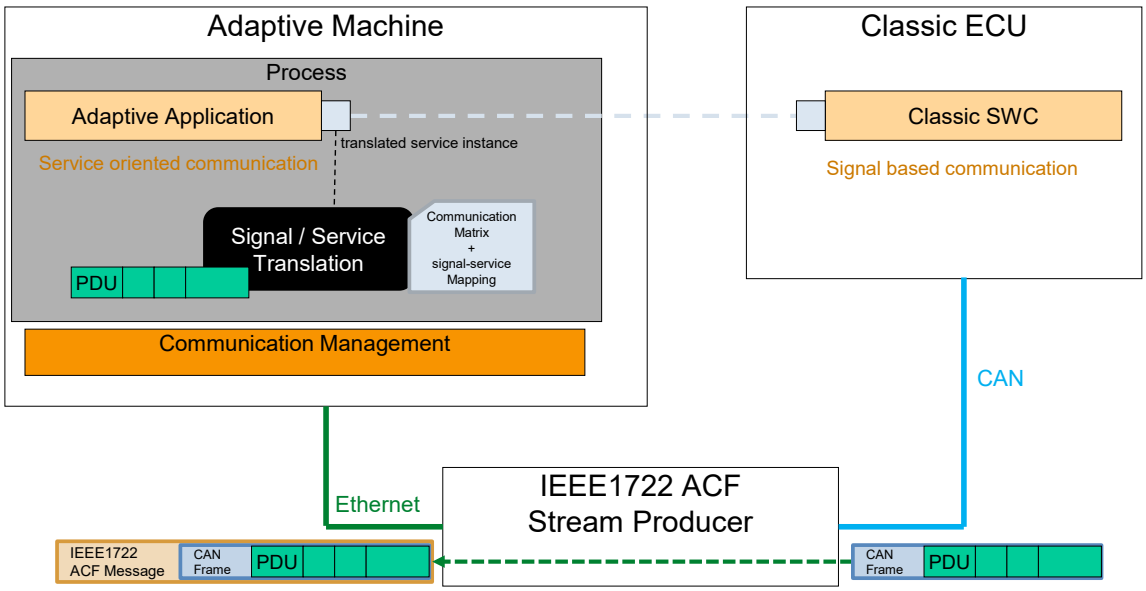


Figure 9.2: `Signal-service-translation` transport of a CAN frame as encapsulated `IEEE1722 ACF message` within an `IEEE1722 stream`

Another approach for the usage of `signal/service translation` is the interaction of a Classic AUTOSAR Instance and an Adaptive AUTOSAR Instance running on the same virtualization technology.

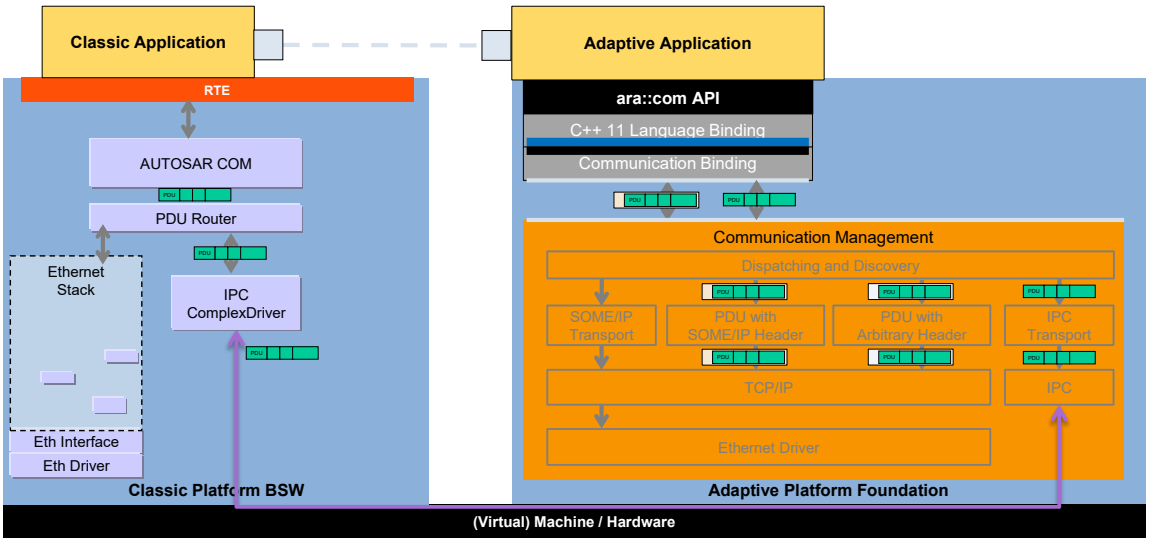


Figure 9.3: `Signal/service translation` using IPC technology

In such a setup, the IPC technology is able to transport `ISignalIPdu`s between the two platform instances. It is not defined whether the IPC technology requires any Message Header (according to Chapter 9.2.1) being part of the payload.

Thus, no dedicated configuration for this Message Header (specifically the configuration of the Header ID) is required if this information can already be derived from the protocol information of the underlying IPC technology (e.g. dedicated IPC channels for different messages).

Figure 9.3 sketches an example setup where a virtualization technology enables the transport of messages between a Classic and an Adaptive Platform instance.

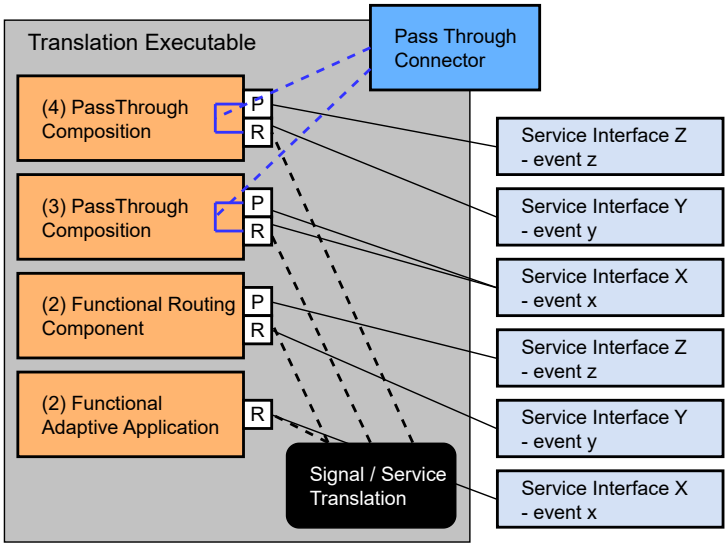


Figure 9.4: `Signal/service translation` and PassThroughConnector

Based on the `signal/service translation` network binding there are several approaches for further processing the translated information (illustrated in Figure 9.4):

- The application software on the *AUTOSAR adaptive platform* directly consumes/produces[1] the translated service (1)
  This is the typical approach if there is only one piece of application software on the *AUTOSAR adaptive platform* interested in the translated data.
- The application software on the *AUTOSAR adaptive platform* manages and performs a functional routing (2)
  This is the approach if the translated data shall be available as a service again for further processing. Here it is up to the implementation of the application software on the *AUTOSAR adaptive platform* how the translated data is routed to the secondary service, especially whether data combination and data conversion are applied.
- Pass Through Connectors: The translated service is passed through to be available to further application software on the *AUTOSAR adaptive platform*. This is the approach if there exist several further application software on the *AUTOSAR adaptive platform* which are using the translated service. In this case the translation only has to be performed once. Two cases can be distinguished:

[1] In the explanations the direction `signal-service-translation` is usually used, the `service-signal-translation` direction is supported as well. For simplicity this is only mentioned explicitly if the mapping behavior is not symmetrical.

- The translated service and the secondary service use the same ServiceInterface (3)
- The translated service and the secondary service use different (but compatible) ServiceInterfaces (4)

The methodological approach is that out of the signal/service translation mapping descriptions it is possible to automatically generate the network binding code for the signal/service translation.

If there is a CompositionSwComponentType defined with PassThroughSwConnectors then also the AdaptiveApplicationSwComponentType implementing the pass through behavior can be generated (see Chapter 9.7).

The configuration of such signal-based communication on an adaptive machine may be solved in two different ways:

1. The communication matrix definition (ARXML System Description) and the signal/service translation mapping is available on the target machine and is interpreted at run-time (like the manifest approach).
2. The communication matrix definition (ARXML System Description) and the signal/service translation mapping is built off-board and the application executable gets uploaded to the target Machine in response to changes in the communication matrix.

The Executable utilizing the signal/service translation and/or implementing the PassThroughSwConnectors is considered to belong to the APPLICATION_LEVEL domain as defined in [constr_1605]. This also applies to cases where the code implementing the PassThroughSwConnectors has been generated automatically by some tooling.

[TPS_MANI_03643] Translation-Executable category ⌈The category value of the Executable utilizing the signal/service translation and/or implementing the PassThroughSwConnectors shall be set to APPLICATION_LEVEL.⌋

The following terminology is used in the context of signal/service translation:

Signal/service translation defines the feature this chapter is concerned with. It does not prescribe a specific translation direction.

Signal-service-translation defines the translation direction from a signal-based to a service oriented representation.

Service-signal-translation defines the translation direction from a service oriented to a signal-based representation.

9.2 Signal-based Prerequisites

The identification of the received and sent messages and their actual length needs to be determined for each individual Pdu. Depending on the transport technology and configuration, a Message Header might be required. If the transport technology can determine the kind of Pdu and the length out of its protocol information, then the Message Header may be omitted.

It depends on the specific transport technology whether a dedicated Message Header and Message Length information is required or not. The SOME/IP header already provides enough information to identify the contained payload. The IEEE1722 ACF stream as well is capable to identify each transported CAN and LIN frame. If a custom transport technology is used (e.g. IPC between a Classic Platform instance and an Adaptive Platform instance), it depends on that specific transport technology whether a Message Header and length information is required or not.

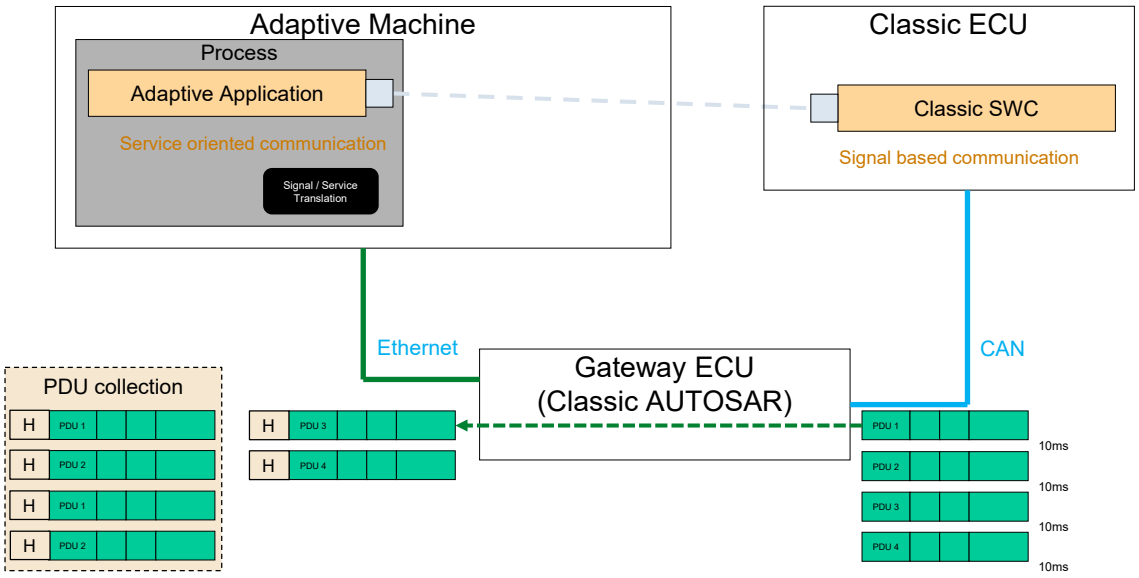


Figure 9.5: Network architecture for signal-service-translation

As the sole communication network currently supported is Ethernet, the Pdus have to be transported on that network.

Although there is in theory the possibility to directly put the Pdu on the Ethernet, this approach would require an individual Socket per Pdu. This approach is not feasible for Classic platform ECUs, which do not have enough resources available to allow individual Sockets per Pdu. Nevertheless, there is also support by the signal/service translation to have no Message Header defined for Ethernet transport.

In the case of Ethernet transport and the approach of several Pdus sharing a Socket, there is a need to use a dedicated Message Header per Pdu. This is illustrated in Figure 9.5. The Classic Platform ECU routes the Pdus from a CAN network to the Ethernet.

In order to have efficient transport of many Pdus on Ethernet additionally the SOME/IP collection mechanism may be used (i.e. definition of pduCollectionTrigger, pduCollectionSemantics, pduCollectionPduTimeout).

9.2.1 Message Header for IP based transport

In the case of Ethernet IP based transport and the approach of several Pdus sharing a Socket, the identification of the received and sent messages is done using a 64 bit wide Message Header. This Message Header is composed of a 32 bit wide Message ID and a 32 bit wide Message Length field.

[TPS_MANI_03652] Signal/service translation header format ⌈If a Message Header is used then the message-transporting signal-based Pdus shall have a 32 bit wide Message ID field and a 32 bit wide Message Length field.⌋

[TPS_MANI_03653] Signal/service translation header endianness ⌈The header of a message-transporting signal-based Pdus shall be encoded in big endian.⌋

There are three variants supported by the signal/service translation approach how the Message Header is interpreted:

Partial SOME/IP Header : 32 bit wide Message ID field and 32 bit wide Length field (just Message ID and Length, then signal-based payload. Message ID according to the rules of SOME/IP: ServiceId and MethodId [PRS_SOMEIP_00245])

Arbitrary Message Header : 32 bit wide Message ID field and 32 bit wide Length field (e.g. CAN-Id taken as Message Id) with signal-based payload

No Message Header : No standardized Message Header is defined.

In Figure 9.6, the *Full SOME/IP Header* is shown for illustration purposes. In the context of signal/service translation the *Full SOME/IP Header* is not used. The *Signal-Based Header* can either be configured according to the Partial SOME/IP Header, the Arbitrary Message Header, or the No Message Header approach.

Which Message Header variant is used for a dedicated message is defined according to the input configuration:

[TPS_MANI_03654] Definition of Partial SOME/IP Header ID ⌈If there is a RequiredSomeipServiceInstance of category SIGNALBASED_WITH_HEADER defined for the signal-service-translation (resp. ProvidedSomeipServiceInstance for the service-signal-translation), then the Message ID shall be interpreted according to the SOME/IP Message ID rules [PRS_SOMEIP_00245]. Namely the Message ID is composed of the SomeipServiceInterfaceDeployment.serviceInterfaceId and the SomeipEventDeployment.eventId + 0x8000.⌋

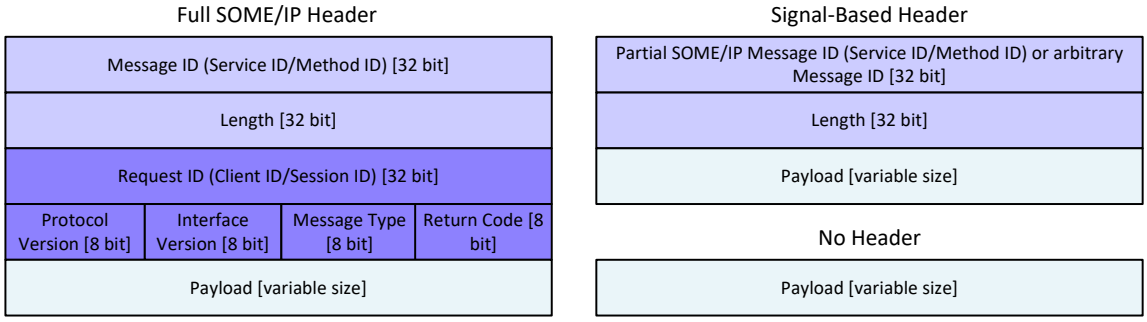


Figure 9.6: SOME/IP Message Header Format

The `Partial SOME/IP Header` as defined in [TPS_MANI_03654] does not contain the SOME/IP serializer fields (Request ID, Protocol Version, Interface Version, Message Type, Return Code). The only identification criterion is the Message ID.

Most of the missing fields are not used for Event communication, thus their lack on the network has no consequences:

- Request ID is composed of Client ID (not used for events) and Session ID (only used for events if used in combination with SOME/IP-TP, thus no issue for `signal/service translation` communication)
- Message Type is fixed for events, thus no issue for `signal/service translation` communication
- Return Code is fixed for events, thus no issue for `signal/service translation` communication

But Protocol Version and Interface Version actually are applicable for Event communication and the Service communication design needs to compensate for these missing fields.

The SOME/IP Protocol Version is a fixed value of 1 according to [PRS_SOMEIP_00051], thus there is currently no configuration attribute defined at the `SomeipServiceInterfaceDeployment`.

The SOME/IP Interface Version is used to transport the `SomeipServiceInterfaceDeployment.serviceInterfaceVersion.majorVersion` according to [PRS_SOMEIP_00053]. Thus there is the need for [constr_3825] to have unique `SomeipEventDeployment.eventId` of all `signalBased` serialized events belonging to one `SomeipServiceInterfaceDeployment.serviceInterfaceId` on one VLAN.

[constr_3825] ProvidedSomeipServiceInstances with SomeipEventDeployment.serializer = signalBased shall be unique with respect to Message ID on one VLAN

Imposition time: IT_Mani

⌈Every SomeipEventDeployment with SomeipEventDeployment.serializer = SerializationTechnologyEnum.signalBased aggregated by SomeipServiceInterfaceDeployment where the SomeipServiceInterfaceDeployment.serviceInterfaceId is identical, shall have a unique SomeipEventDeployment.eventId defined in the scope of one VLAN.⌋

[TPS_MANI_03577] Definition of Arbitrary Message Header ID

Status: DRAFT

⌈If there is a RequiredUserDefinedServiceInstance of category SIGNALBASED_WITH_HEADER defined for the signal-service-translation (resp. ProvidedUserDefinedServiceInstance for the service-signal-translation), then the Message ID shall be taken from the SoConIPduIdentifier.headerId attribute.⌋

[constr_3650] headerId required in the case of Arbitrary Message Header

Status: DRAFT
Imposition time: IT_Mani

⌈If [TPS_MANI_03577] applies, then the respective SoConIPduIdentifier.headerId shall be defined.⌋

There is also the possibility to define the No Message Header option:

[TPS_MANI_03655] Definition of No Message Header - implicit SOME/IP Message Identification

Status: DRAFT

⌈If there is a RequiredSomeipServiceInstance of category SIGNALBASED_NO_HEADER defined for the signal-service-translation (resp. ProvidedSomeipServiceInstance for the service-signal-translation), then no Message Header shall be used.

In this case it is either possible to derive the Message identification from the assignment to unambiguous communication channels or the Message identification is derived from protocol information of the underlying IPC technology.⌋

[TPS_MANI_03656] Definition of No Message Header - implicit Arbitrary Message Identification

Status: DRAFT

⌈If there is a RequiredUserDefinedServiceInstance of category SIGNALBASED_NO_HEADER defined for the signal-service-translation (resp. ProvidedUserDefinedServiceInstance for the service-signal-translation), then no Message Header shall be used.

In this case it is either possible to derive the Message identification from the assignment to unambiguous communication channels or the Message identification is derived from protocol information of the underlying IPC technology.⌋

9.2.2 Message Identification for IEEE1722 ACF based transport

If the signal based ISignalIPdu is transported using an IEEE1722 ACF stream according to [19, IEEE 1722-2016] (as depicted in Figure 9.2), then the message identification is fully provided by the IEEE1722 ACF stream and the corresponding modeling in the IEEE1722TpAcfConnection. Details on the IEEE1722TpAcfConnection are provided in the [4, TPS System Template].

[TPS_MANI_03682] IEEE1722TpAcfConnection contributes to signal/service translation

Status: DRAFT

⌈If

- an ISignalTriggering is referenced by an SignalBasedEventElementToISignalTriggeringMapping in the role iSignalTriggering,
- and that ISignalTriggering is part of an ISignalIPdu,
- and that ISignalIPdu is referenced by a PduTriggering,
- and that PduTriggering is referenced by an IEEE1722TpAcfCanPart or IEEE1722TpAcfLinPart,

then the enclosing IEEE1722 ACF stream (defined by the IEEE1722TpAcfConnection) is subject to signal/service translation.⌋

The IEEE1722 ACF stream identification is done based on the attributes IEEE1722TpConnection.macAddressStreamId and IEEE1722TpConnection.uniqueStreamId (see also [TPS_SYST_03098]).

If the subtype of the IEEE1722 ACF stream is IEEE1722TpAcfConnection, then further identification is based on the IEEE1722TpAcfBus. The specific IEEE1722 ACF message may contain several parts, each part belonging to a specific IEEE1722TpAcfBus: IEEE1722TpAcfCan or IEEE1722TpAcfLin.

The attribute IEEE1722TpAcfBus.busId is defined per IEEE1722TpAcfCan and IEEE1722TpAcfLin, thus it is possible to have an instance of IEEE1722TpAcfCan with busId = 1 and also an IEEE1722TpAcfLin with busId = 1.

The final identification which CAN or LIN frame (and thus which ISignalIPdu) is transported is based on the bus type and is defined by:

- IEEE1722TpAcfCanPart.canIdentifier
- IEEE1722TpAcfLinPart.linIdentifier.

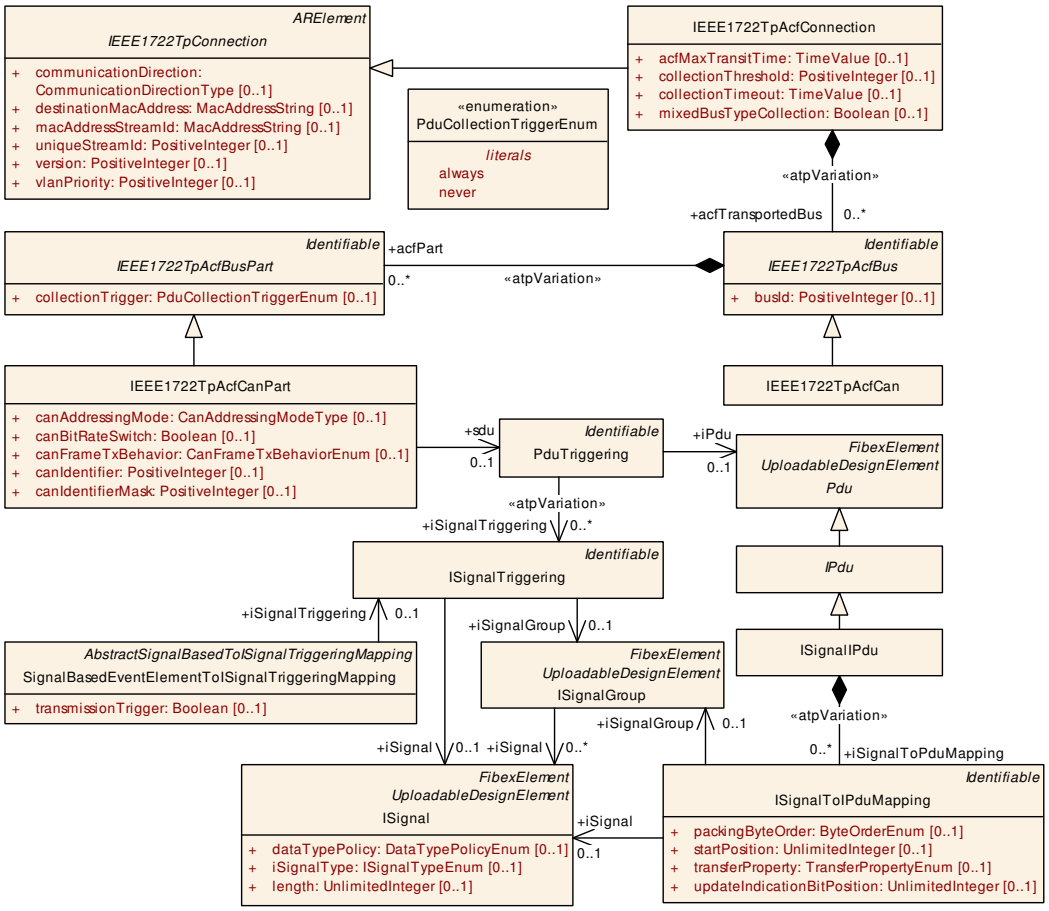


Figure 9.7: Relation of `SignalBasedEventElementToISignalTriggeringMapping` and `IEEE1722TpAcfConnection`

[TPS_MANI_03683] Identification of an `ISignalIPdu` inside an `IEEE1722 ACF stream`

Status: DRAFT

⌈The following attributes need to be respected to unambiguously identify an `ISignal-IPdu` transported over an `IEEE1722 ACF stream`:

- `IEEE1722TpConnection.macAddressStreamId`
- `IEEE1722TpConnection.uniqueStreamId`
- `busId`
- Frame Id:
  - `IEEE1722TpAcfCanPart.canIdentifier` or
  - `IEEE1722TpAcfLinPart.linIdentifier`

⌋

Which EthernetCommunicationConnector shall be used for the communication of an IEEE1722 ACF stream is defined by the UserDefinedServiceInstanceToMachineMapping.communicationConnector reference.

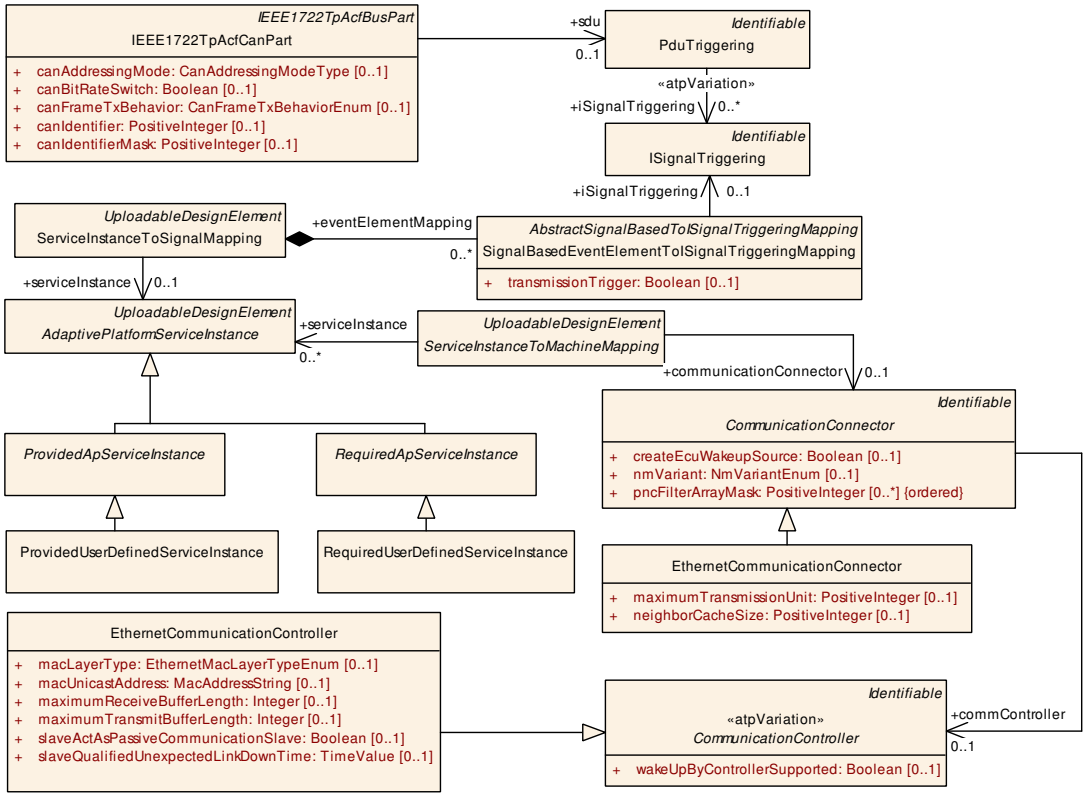


Figure 9.8: Definition which EthernetCommunicationConnector is considered for an IEEE1722 ACF stream

[TPS_MANI_03684] Mapping of an IEEE1722 ACF stream to an EthernetCommunicationConnector

Status: DRAFT

⌈If a UserDefinedServiceInstanceToMachineMapping refers to a RequiredUserDefinedServiceInstance or a ProvidedUserDefinedServiceInstance of category SIGNALBASED_IEEE1722_ACF in the role serviceInstance, then the referenced EthernetCommunicationConnector shall be used for the transport of the IEEE1722 ACF stream.⌋

9.3 Signal-based Deployment

The signal-service-translation is embedded in the SOME/IP deployment using SomeipServiceInterfaceDeployment. The attribute SomeipEventDeployment.serializer defines whether the someip or the signalBased serialization shall be used for a specific event (see also Figure 13.4).

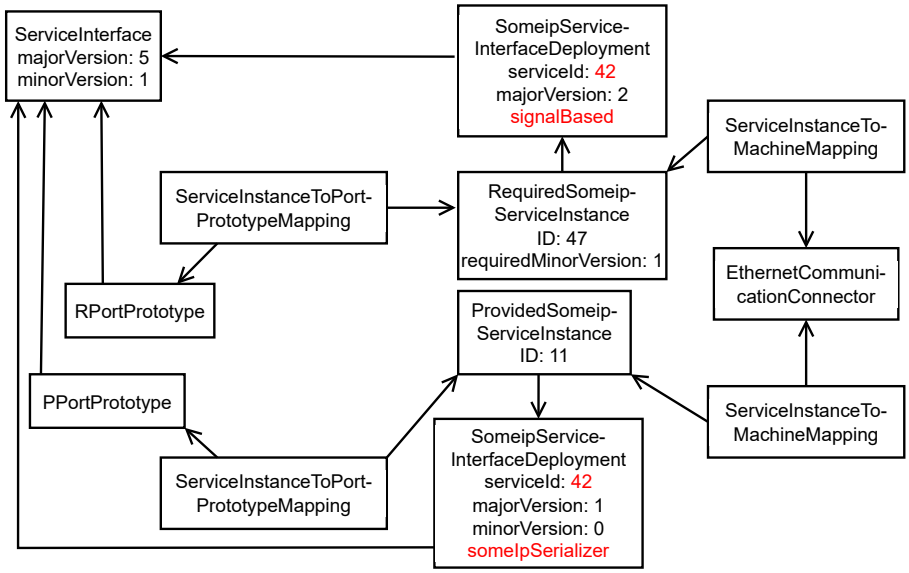


Figure 9.9: Example of 1 `PPortPrototype` mapped to one VLAN

[TPS_MANI_03578] Signal-based `ServiceInterface` binding over `Ethernet` ⌈In the case of signal-based communication over `Ethernet`, the `SomeipService-InterfaceDeployment` is used to define the `Pdu` transport over the network.⌋

This aspect is described in Chapter 9.2.

[TPS_MANI_03579] Signal-based `ServiceEventDeployment` over `Ethernet` ⌈If the attribute `SomeipEventDeployment.serializer` equals `signalBased` then the `event` referenced by `ServiceEventDeployment.event` shall be handled using signal-service-translation.⌋

Figure 9.9 illustrates a use-case where a `ServiceInterface` has two `Someip-ServiceInterfaceDeployment`s, one with `signalBased serializer` technology and one with `someip`.

The translation use-case is defined by having one `RPortPrototype` receiving the `signalBased` serialized messages and one `PPortPrototype` providing the `someip` serialized service.

9.4 `Signal/Service Translation` Mapping

This chapter describes the mapping of `ServiceInterface` elements of a specific `AdaptivePlatformServiceInstance` to `ISignalTriggering`s.

According to [TPS_MANI_03555], the same Ethernet socket (via `ISignal-Triggering` and `PduTriggering`) may be used for signal-based and service-oriented communication at the same time.

This allows to define one service instance which consists of events with different serialization technologies (i.e. someip and signalBased).

[TPS_MANI_03627] No signal/service translation for methods ⌈As methods and field getters/setters are already serialized using the someip serialization, there is no need for signal/service translation for them.⌋

Therefore, methods and field getters/setters are directly accessible using ara::com via the *Signal Based Network Binding* as defined in [32, SWS Communication Management]. To pass methods through, refer to Chapter 9.7.

A ServiceInstanceToSignalMapping refers to an AdaptivePlatformServiceInstance in the role ServiceInstanceToSignalMapping.serviceInstance. While it would be possible to define the mappings for all the events, fields, and triggers of an AdaptivePlatformServiceInstance in the scope of one ServiceInstanceToSignalMapping, it is rather advised to create several ServiceInstanceToSignalMappings referring to the same AdaptivePlatformServiceInstance, one for each service member.

[advisory_02000] Define a ServiceInstanceToSignalMapping per member of an AdaptivePlatformServiceInstance

Imposition time: IT_Mani

⌈If an AdaptivePlatformServiceInstance is referenced by a ServiceInstanceToSignalMapping in the role ServiceInstanceToSignalMapping.serviceInstance, then there should be one instance of ServiceInstanceToSignalMapping defined for every member of the corresponding ServiceInterface (event, field, trigger).⌋

9.4.1 Handling of ISignalGroups

ISignalGroups are used in the definition of ISignalIPdus to express that certain ISignals belong together and form a consistent set of data. The definition of ISignalGroups is part of the signal based communication matrix definition.

ISignalGroups do not need to be explicitly mapped when defining the ServiceInstanceToSignalMappings, as they are implicitly covered in the mapping of the individual ISignals belonging to the ISignalGroup.

[advisory_02001] No explicit signal/service translation for ISignalGroups

Imposition time: IT_Mani

⌈The mapping elements of a ServiceInstanceToSignalMapping:

- SignalBasedEventElementToISignalTriggeringMapping.iSignalTriggering

- SignalBasedTriggerToISignalTriggeringMapping.iSignal-Triggering
- SignalBasedFieldToISignalTriggeringMapping.notifierSignal-Triggering
- SignalBasedFireAndForgetMethodToISignalTriggeringMapping.iSignalTriggering

should not refer to an ISignalTriggering that in turn refers to an ISignalGroup.⌋

9.4.2 Direction of Signal/Service Translation Mapping

The definition of the translation mappings has no explicitly modeled direction (has no source or target point of view). Each individual mapping (AbstractSignalBased-ToISignalTriggeringMapping) takes one element from a ServiceInterface and one ISignalTriggering.

[TPS_MANI_03635] Determination of translation direction ⌈The translation direction is determined by

- the communicationDirection of the referenced ISignalPort which is referenced by the ISignalTriggering
- the kind of service instance which is referenced from the ServiceInstanceTo-SignalMapping in the role serviceInstance. It can be either a Provided-ApServiceInstance or a RequiredApServiceInstance.

⌋

[constr_3636] Consistent ISignal communication direction in and Required-ApServiceInstance

Imposition time: IT_Mani

⌈If the ServiceInstanceToSignalMapping.serviceInstance refers to a RequiredApServiceInstance then any ServiceInstanceToSignal-Mapping.eventElementMapping (respectively ServiceInstanceToSignal-Mapping.fieldMapping) shall refer to an ISignalTriggering which in turn refers to an ISignalPort with communicationDirection equal to in.⌋

[constr_3637] Consistent ISignal communication direction out and Provided-ApServiceInstance

Imposition time: IT_Mani

⌈If the ServiceInstanceToSignalMapping.serviceInstance refers to a ProvidedApServiceInstance then any ServiceInstanceToSignal-Mapping.eventElementMapping (respectively ServiceInstanceToSignal-Mapping.fieldMapping) shall refer to an ISignalTriggering which in turn refers to an ISignalPort with communicationDirection equal to out.⌋

Class	**ServiceInstanceToSignalMapping**			
Note	This meta-class is defined for a specific ServiceInstance and contains the mappings of elements of a ServiceInterface for which the ServiceInstance is defined to individual ISignalTriggerings. **Tags:** atp.recommendedPackage=ServiceInstanceToSignalMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventElement Mapping	SignalBasedEvent ElementToISignal TriggeringMapping	*	aggr	Mapping of an event or an element inside of the event to an ISignalTriggering. **Tags:** atp.Splitkey=eventElementMapping, eventElement Mapping.shortName
fieldMapping	SignalBasedFieldTo ISignalTriggering Mapping	*	aggr	Mapping of a field to ISignalTriggerings. **Tags:** atp.Splitkey=fieldMapping, fieldMapping.short Name
fireAndForget MethodMapping	SignalBasedFireAnd ForgetMethodToISignal TriggeringMapping	*	aggr	Mapping of an ISignalTriggering being part of a fire and forget message to a ClientServerOperation. **Tags:** atp.Splitkey=fireAndForgetMethodMapping, fireAnd ForgetMethodMapping.shortName
serviceInstance	AdaptivePlatform ServiceInstance	0..1	ref	Reference to a ServiceInstance from which the corresponding ServiceInterface elements will be transported in the signal-based way over a communication medium. **Tags:** atp.Splitkey=serviceInstance
triggerMapping	SignalBasedTriggerTo ISignalTriggering Mapping	*	aggr	Mapping of a trigger to an ISignalTriggering. **Tags:** atp.Splitkey=triggerMapping, triggerMapping.short Name

Table 9.1: ServiceInstanceToSignalMapping

The `ServiceInstanceToSignalMapping` refers to an `AdaptivePlatformServiceInstance` and thereby defines which `serviceInterfaceDeployment` elements will be mapped by the aggregated `eventElementMapping`, and/or `fieldMapping` to `ISignalTriggering`s. This is described in detail in the following chapters.

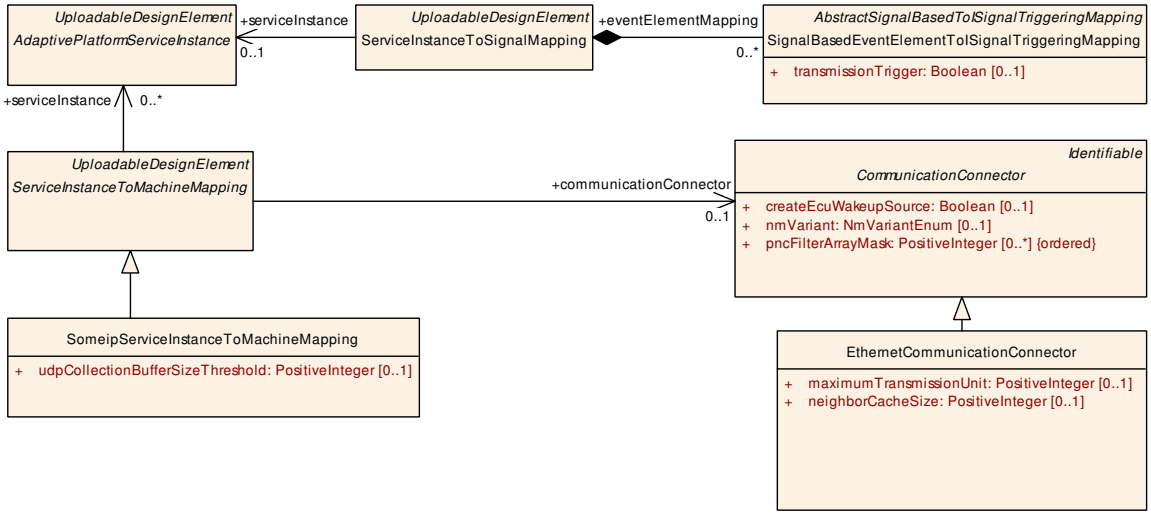


Figure 9.10: Relation of `ServiceInstanceToSignalMapping` and `CommunicationConnector`

The ServiceInstanceToMachineMapping which refers to the AdaptivePlatformServiceInstance (which in turn is referenced by the ServiceInstanceToSignalMapping) defines on which CommunicationConnector (i.e. network / VLAN) the signal based communication shall be performed.

[TPS_MANI_03629] Relation of meta-classes ServiceInstanceToSignalMapping and CommunicationConnector ⌈The meta-class ServiceInstanceToMachineMapping referring to the AdaptivePlatformServiceInstance defines on which CommunicationConnector the AdaptivePlatformServiceInstance shall be communicated.

If a ServiceInstanceToSignalMapping refers to the same AdaptivePlatformServiceInstance then the signal based communication shall be performed on the referenced CommunicationConnector.⌋

9.4.3 SignalBasedEvent Mapping

It is required that every event with signalBased serialization has a ServiceInstanceToSignalMapping defined.

[constr_3550] Existence of ServiceInstanceToSignalMapping for an event with signalBased serialization

Imposition time: IT_Mani

⌈If

- an event is referenced by a SomeipEventDeployment in the role event and
- the attribute SomeipEventDeployment.serializer is set to signalBased,

then a ServiceInstanceToSignalMapping shall exist with eventElementMapping referring to the event in the role dataPrototypeInServiceInterfaceRef.⌋

For composite payload, the mapping definition needs to be complete from the target point of view:

[constr_3551] Full mapping of target ISignalGroup

Imposition time: IT_Mani

⌈If

- an ISignalTriggering is part of a ServiceInstanceToSignalMapping and
- the ISignalTriggering refers to an ISignalPort with communicationDirection equals out and
- the ISignalTriggering refers to an ISignalGroup in the role iSignalGroup,

then a SignalBasedEventElementToISignalTriggeringMapping shall exist for every ISignal referenced by the ISignalGroup in the role iSignal.⌋

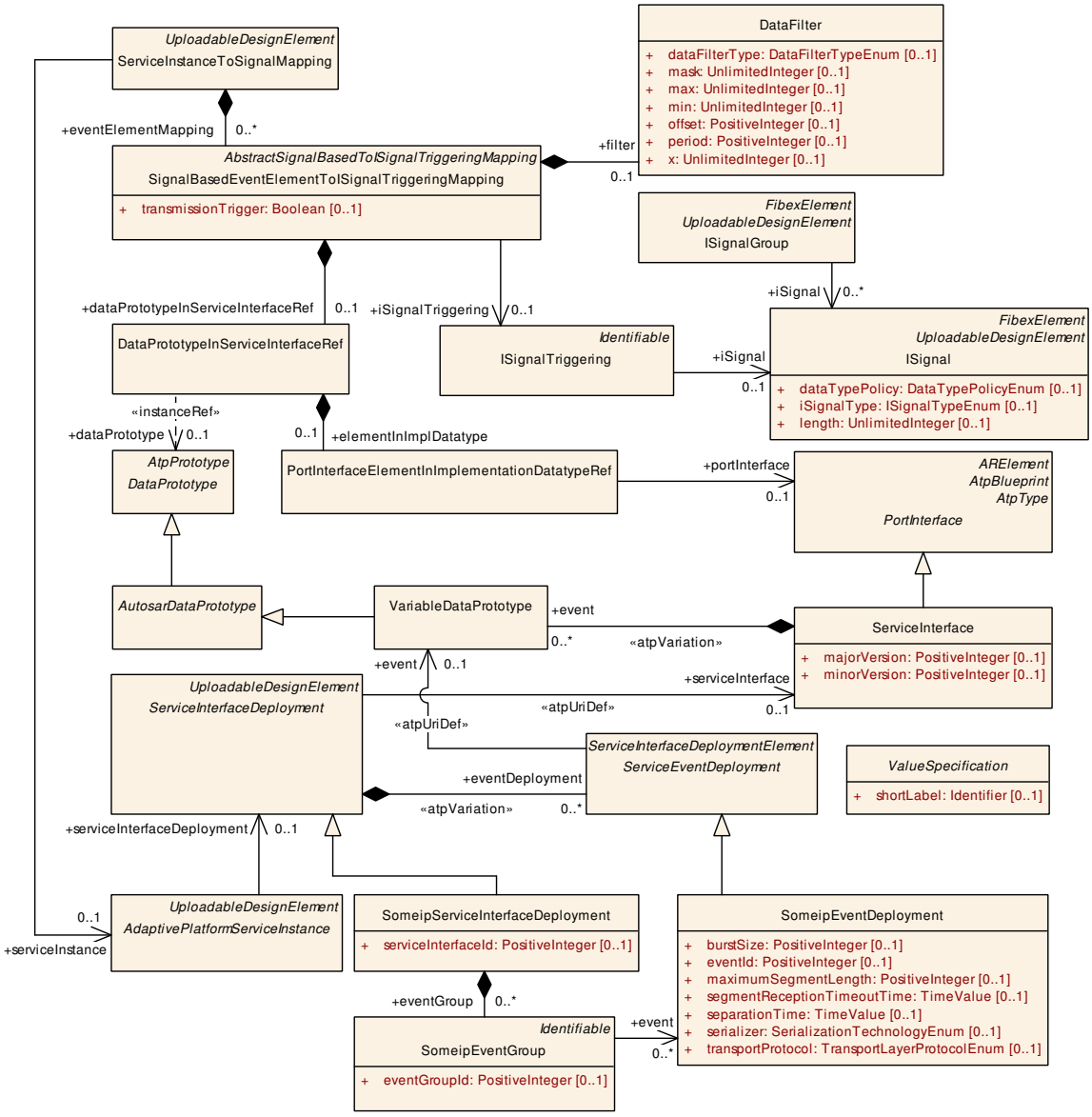


Figure 9.11: Mapping of Event elements to ISignals

[constr_3552] Full mapping of target event

Imposition time: IT_Mani

⌈If

- the ServiceInstanceToSignalMapping refers to a ProvidedSomeip-ServiceInstance and
- the dataPrototypeInServiceInterfaceRef refers to a DataPrototype which is part of a composite data type,

then a SignalBasedEventElementToISignalTriggeringMapping shall exist for every DataPrototype that is part of the composite data type.⌋

[TPS_MANI_03124] ServiceInterface.event to ISignalTriggering mapping ⌈The SignalBasedEventElementToISignalTriggeringMapping meta-class provides the ability to map a DataPrototype defined in the context of a ServiceInterface to one ISignalTriggering of the ISignal or ISignalGroup.⌋

Class	**SignalBasedEventElementToISignalTriggeringMapping**			
Note	This meta-class defines the mapping of a ServiceInterface event or an element that is defined inside of the event in case that the datatype is composite to an ISignalTriggering. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractSignalBasedToISignalTriggeringMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ServiceInstanceToSignalMapping.eventElementMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataPrototypeIn ServiceInterface Ref	DataPrototypeInService InterfaceRef	0..1	aggr	Reference to a DataPrototype or to an internal structure of a DataPrototype in the context of a ServiceInterface. **Tags:** atp.Splitkey=dataPrototypeInServiceInterfaceRef
filter	DataFilter	0..1	aggr	Defines an optional filter to be applied during translation. **Tags:** atp.Splitkey=filter
iSignal Triggering	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport a piece of data of an event that is defined in a ServiceInterface in a signal-based way over a communication channel. **Tags:** atp.Splitkey=iSignalTriggering
transmission Trigger	Boolean	0..1	attr	Defines whether the source element triggers the sending of the respective payload.

Table 9.2: SignalBasedEventElementToISignalTriggeringMapping

[constr_10252] Multiplicity of attribute SignalBasedEventElementToISignalTriggeringMapping.dataPrototypeInServiceInterfaceRef

Imposition time: IT_Mani

⌈For each SignalBasedEventElementToISignalTriggeringMapping, the attribute dataPrototypeInServiceInterfaceRef shall exist.⌋

Class	***AbstractSignalBasedToISignalTriggeringMapping*** (abstract)			
Note	This meta-class is the common class for all SIgnalBased to ISignalTRiggering mappings. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	SignalBasedEventElementToISignalTriggeringMapping, SignalBasedFieldToISignalTriggeringMapping, SignalBasedFireAndForgetMethodToISignalTriggeringMapping, SignalBasedTriggerToISignalTriggering Mapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 9.3: AbstractSignalBasedToISignalTriggeringMapping

A ServiceInstanceToSignalMapping with several SignalBasedEvent-ElementToISignalTriggeringMappings is used to map the data leaves of *TestEvent* to the corresponding ISignalTriggerings.

An example of a mapping of event content to a signal is described in Chapter B.10.1.1.

9.4.4 SignalBasedTrigger Mapping

It is required that every trigger with signalBased serialization has a Service-InstanceToSignalMapping defined:

[constr_5318] Existence of ServiceInstanceToSignalMapping for an trigger with signalBased serialization

Imposition time: IT_Mani

⌈If a trigger is referenced by a SomeipEventDeployment in the role trigger and the attribute SomeipEventDeployment.serializer is set to signalBased then a ServiceInstanceToSignalMapping shall exist with triggerMapping referring to the trigger in the role trigger.⌋

[TPS_MANI_03287] ServiceInterface.trigger to ISignalTriggering mapping ⌈The SignalBasedTriggerToISignalTriggeringMapping meta-class provides the ability to map a trigger defined in the context of a ServiceInterface to one ISignalTriggering of the ISignal.⌋

[constr_1198] and [constr_1199] that are defined in the [4, TPS System Template] apply for the ISignal to which the trigger is mapped via the ISignalTriggering by the SignalBasedTriggerToISignalTriggeringMapping. This means that the length of the ISignal shall be 0 and an update bit shall be defined for the ISignal if DataTransformation is not used.

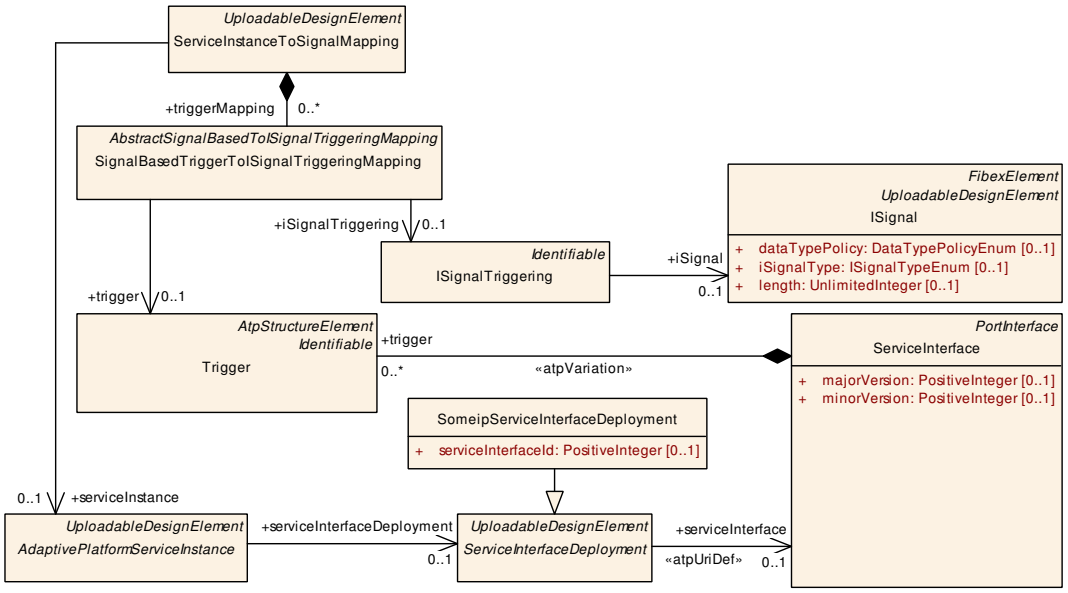


Figure 9.12: Mapping of a Trigger to an ISignal

Class	**SignalBasedTriggerToISignalTriggeringMapping**			
Note	This meta-class defines the mapping of a ServiceInterface trigger to an ISignalTriggering. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractSignalBasedToISignalTriggeringMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ServiceInstanceToSignalMapping.triggerMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iSignal Triggering	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport the trigger that is defined in a ServiceInterface in a signal-based way over a communication channel. **Tags:** atp.Splitkey=iSignalTriggering
trigger	Trigger	0..1	ref	Reference to a trigger defined in the context of a Service Interface. **Tags:** atp.Splitkey=trigger

Table 9.4: SignalBasedTriggerToISignalTriggeringMapping

9.4.5 SignalBasedField Mapping

[advisory_02002] `ServiceInterface.field` mapping to `ISignalTriggering`s for primitive data type `field`s

Imposition time: `IT_Mani`

⌈If `Field.hasNotifier` equals true, then one `SignalBasedFieldToISignalTriggeringMapping` should be used to map a primitive data type `field` to one `ISignalTriggering` for the `ISignal` representing the primitive Notifier element using the role `notifierSignalTriggering`.⌋

[advisory_02003] `ServiceInterface.field` mapping to `ISignalTriggering`s for composite data type `field`s

Imposition time: `IT_Mani`

⌈If `Field.hasNotifier` equals true, then a separate `SignalBasedFieldToISignalTriggeringMapping` should be used to map each `ISignal` belonging to the `ISignalGroup` used for the transport of the notifier using the role `notifierSignalTriggering`.⌋

This means that several `SignalBasedFieldToISignalTriggeringMapping`s may be necessary to map a `field` to the corresponding `ISignalTriggering`s.

It is required that every `field` using a `SomeipFieldDeployment.notifier` with `signalBased` serialization has a `ServiceInstanceToSignalMapping` defined:

[constr_3553] Existence of `ServiceInstanceToSignalMapping` for an `field` with `signalBased` serialization

Imposition time: IT_Mani

⌈If a field is referenced by a SomeipFieldDeployment in the role field and that SomeipFieldDeployment aggregates a SomeipEventDeployment in the role notifier and the SomeipEventDeployment has an attribute SomeipEvent-Deployment.serializer set to signalBased then there shall exist a Service-InstanceToSignalMapping with a fieldMapping referring to the field in the role dataPrototypeInServiceInterfaceRef and the SignalBasedFieldTo-ISignalTriggeringMapping shall refer to a ISignalTriggering in the role notifierSignalTriggering.⌋

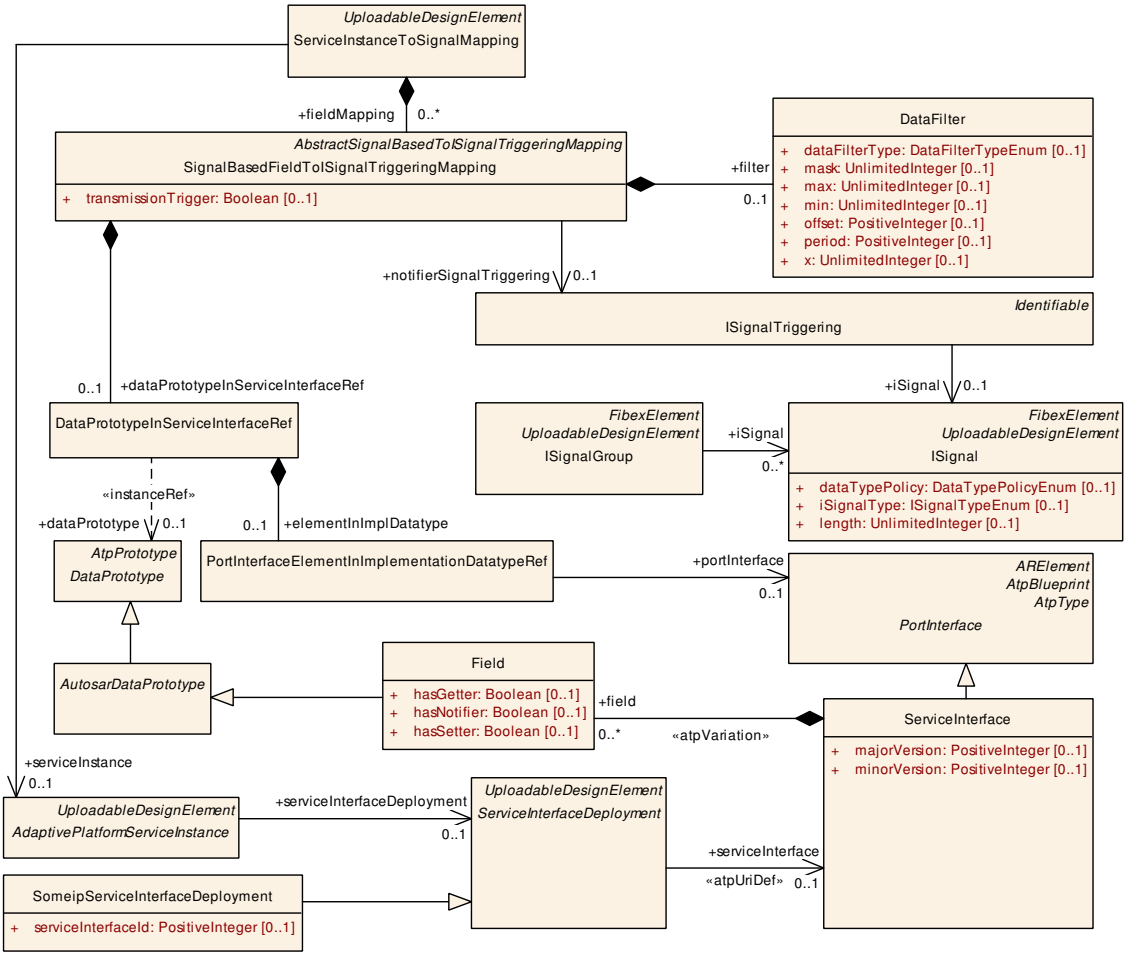


Figure 9.13: Mapping of Fields to ISignals

[constr_10228] Multiplicity of attribute `SignalBasedFieldToISignal-TriggeringMapping.dataPrototypeInServiceInterfaceRef`

Imposition time: IT_Mani

⌈For each SignalBasedFieldToISignalTriggeringMapping, the attribute dataPrototypeInServiceInterfaceRef shall exist.⌋

Class	**SignalBasedFieldToISignalTriggeringMapping**			
Note	This meta-class defines the mapping of a ServiceInterface field to ISignalTriggerings that represent the notifier elements on a signal-based communication channel. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractSignalBasedToISignalTriggeringMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ServiceInstanceToSignalMapping.fieldMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataPrototypeIn ServiceInterface Ref	DataPrototypeInService InterfaceRef	0..1	aggr	Reference to a DataPrototype or to an internal structure of a DataPrototype in the context of a ServiceInterface. **Tags:** atp.Splitkey=dataPrototypeInServiceInterfaceRef
filter	DataFilter	0..1	aggr	Defines an optional filter to be applied during translation. **Tags:** atp.Splitkey=filter
notifierSignal Triggering	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport a piece of data of a notifier in a signal-based way over a communication channel. **Tags:** atp.Splitkey=notifierSignalTriggering
transmission Trigger	Boolean	0..1	attr	Defines whether the source notifier element triggers the sending of the respective payload.

Table 9.5: SignalBasedFieldToISignalTriggeringMapping

An example of a mapping of a field to signals is described in Chapter B.10.1.2.

9.4.6 SignalBased Fire and Forget Method Mapping

[TPS_MANI_03665] **`ServiceInterface.method` with `fireAndForget` equals `true` to `ISignalTriggering`s mapping** ⌈The `SignalBasedFireAndForgetMethodToISignalTriggeringMapping` meta-class provides the ability to map a part of a `method` with `ClientServerOperation.fireAndForget` equals `true` to one `ISignalTriggering` for the `ISignal` representing one part of the “fire & forget” method message.⌋

[constr_3692] **`DataPrototypeInServiceInterfaceInstanceRef.targetDataPrototype` in the context of a `SignalBasedFireAndForgetMethodToISignalTriggeringMapping`**

Imposition time: `IT_Mani`

⌈If a `DataPrototypeInServiceInterfaceInstanceRef` is aggregated by a `SignalBasedFireAndForgetMethodToISignalTriggeringMapping` in the role `dataPrototypeInMethodArgumentInstanceRef`, then the reference `DataPrototypeInServiceInterfaceInstanceRef.targetDataPrototype` shall refer to an `ArgumentDataPrototype`.⌋

On the *AUTOSAR classic platform*, the “call” of a “fire & forget method” is modeled by means of a `dataElement` within a `SenderReceiverInterface`. In other words, the `dataElement` carries the entire collection of arguments of the “fire & forget method call” to ensure atomic transport over the network.

If

- a “fire & forget method call” has more than one argument or
- one argument of the “fire & forget method call” is typed by a composite data type,

then the transmission of the arguments of the “fire & forget method call”, modeled as a `dataElement`, involves the definition of an `ISignalGroup` that bundles all the `ISignal`s that are used to transport individual primitive arguments or the elements of composite arguments of the corresponding `ClientServerOperation` on the side of the *AUTOSAR adaptive platform*.

This important model relation is enforced by [constr_10540] and [constr_10541].

[constr_10540] Arguments of a `ClientServerOperation` shall be fully covered by `SignalBasedFireAndForgetMethodToISignalTriggeringMapping`s

Imposition time: `IT_Mani`

⌈If

- an `ArgumentDataPrototype` (if it has a primitive data type) or
- a leaf element of an `ArgumentDataPrototype` (if it has a composite data type)

owned by a `ClientServerOperation` is referenced in the role `dataPrototypeInMethodArgumentInstanceRef` by a `SignalBasedFireAndForgetMethodToISignalTriggeringMapping`, then **every other**

- `ArgumentDataPrototype` (if it has a primitive data type) or
- leaf element of an `ArgumentDataPrototype` (if it has a composite data type)

of the enclosing `ClientServerOperation` shall also be referenced in the role `dataPrototypeInMethodArgumentInstanceRef`.⌋

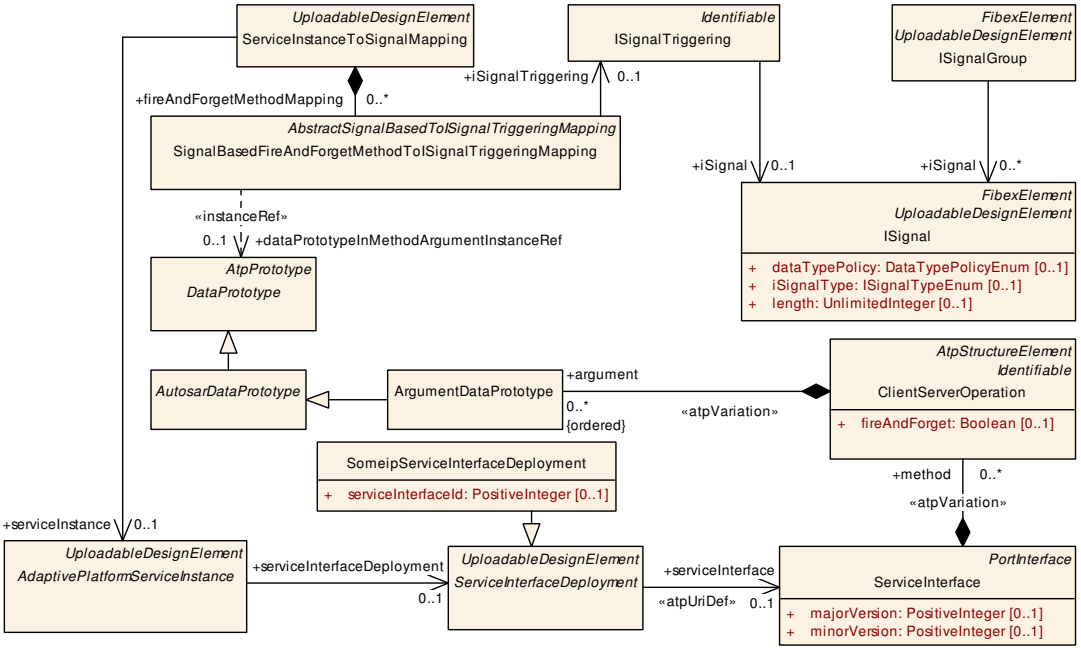


Figure 9.14: Mapping of “fire & forget” methods to `ISignal`s

[constr_10541] Consistent assignment of ISignalTriggerings to an ISignalGroup for SignalBasedFireAndForgetMethodToISignalTriggeringMapping

Imposition time: IT_Mani

⌈Each SignalBasedFireAndForgetMethodToISignalTriggeringMapping affected by [constr_10540] shall also refer to an ISignalTriggering that references an ISignal that in turn is also referenced in the role iSignal by the **same** ISignalGroup.⌋

Class	**SignalBasedFireAndForgetMethodToISignalTriggeringMapping**			
Note	This meta-class defines the mapping of a ServiceInterface fire and forget method part to an ISignal Triggering. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractSignalBasedToISignalTriggeringMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ServiceInstanceToSignalMapping.fireAndForgetMethodMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataPrototypeIn Method Argument InstanceRef	DataPrototype	0..1	iref	Instance reference to a (potentially structured) member of a ClientServerOperation. **Tags:** atp.Splitkey=dataPrototypeInMethodArgument InstanceRef, dataPrototypeInMethodArgumentInstance Ref.rootDataPrototype, dataPrototypeInMethodArgument InstanceRef.contextDataPrototype, dataPrototypeIn MethodArgumentInstanceRef.targetDataPrototype **InstanceRef implemented by:** DataPrototypeInService InterfaceInstanceRef
iSignal Triggering	ISignalTriggering	0..1	ref	Reference to an ISignalTriggering being part of a fire and forget message. **Tags:** atp.Splitkey=iSignalTriggering

Table 9.6: SignalBasedFireAndForgetMethodToISignalTriggeringMapping

9.5 Service Discovery and Static Communication Configuration

Although the ISignalIPdus, which are candidates for signal/service translation, may be communicated to/from CAN or FlexRay networks (where there is no service discovery support available), the messages still might be candidates for service discovery and event subscription on the Ethernet network. In such cases the gateway ECU connecting the CAN/FlexRay networks to the Ethernet network is responsible for providing service discovery and event subscription support as part of the gateway operation.

Only SOME/IP supports service discovery in combination with signal/service translation. Whether or not service discovery shall be done in the case of the Partial SOME/IP Header approach is defined by the category of the RequiredSomeipServiceInstance or ProvidedSomeipServiceInstance.

Even in cases where SOME/IP service discovery defined by the category of the RequiredSomeipServiceInstance or ProvidedSomeipServiceInstance is

configured there may still be a static configuration applied as defined in [TPS_MANI_03295] and [TPS_MANI_03298].

If no SOME/IP service discovery is used, then there is the possibility to define a static configuration as described in Chapter 9.5.4.

[TPS_MANI_03657] `Signal-service-translation` SOME/IP Service Discovery Find ⌈If there is a `RequiredSomeipServiceInstance` of `category` `SIGNAL-BASED_WITH_HEADER` defined for the `signal-service-translation`, then this `RequiredSomeipServiceInstance` shall be handled using SOME/IP service discovery.⌋

[TPS_MANI_03658] `Service-signal-translation` SOME/IP Service Discovery Offer ⌈If there is a `ProvidedSomeipServiceInstance` of `category` `SIGNAL-BASED_WITH_HEADER` defined for the `service-signal-translation`, then this `ProvidedSomeipServiceInstance` shall be handled using SOME/IP service discovery.⌋

If the source `ISignalIPdu` in a `signal-service-translation` scenario is composed of several parts (`ISignalGroup`s and/or individual `ISignal`s) and these parts shall be mapped to several `event`s or field notifiers (according to [TPS_MANI_03667]) and a service discovery shall be performed, then a specific configuration of the `SomeipServiceInterfaceDeployment` and `SomeipEventDeployment` is required.

As in the case of [TPS_MANI_03654] the SOME/IP Message ID is derived from the `SomeipServiceInterfaceDeployment.serviceInterfaceId` and the `SomeipEventDeployment.eventId`, all the `event`s mapped from one `ISignal-IPdu` shall have identical `SomeipServiceInterfaceDeployment.service-InterfaceId`s and `SomeipEventDeployment.eventId`s.

[TPS_MANI_03666] Service Discovery for multi-event `ISignalIPdu`s ⌈If the `ISignal`s of one `ISignalIPdu` are mapped to more than one `event`s (using `SignalBasedEventElementToISignalTriggeringMapping.iSignal-Triggering`) or field notifiers (using `SignalBasedFieldToISignal-TriggeringMapping.notifierSignalTriggering`),

then each of the `SignalBasedEventElementToISignalTriggeringMapping`s and `SignalBasedFieldToISignalTriggeringMapping`s referring to `ISignal`s of that `ISignalIPdu` shall be aggregated by one or more `ServiceInstance-ToSignalMapping`s where the reference `ServiceInstanceToSignalMapping.serviceInstance` refers to a `RequiredSomeipServiceInstance`

and that `RequiredSomeipServiceInstance` shall refer to a `SomeipService-InterfaceDeployment` where

- each of the referenced `SomeipServiceInterfaceDeployment`s has the identical value for `SomeipServiceInterfaceDeployment.service-InterfaceId` defined

- every event taking part in that signal-service-translation has the attribute SomeipEventDeployment.serializer set to SerializationTechnologyEnum.signalBased and has the identical value for:
  - SomeipEventDeployment.eventId
  - SomeipEventDeployment.transportProtocol

⌋

[TPS_MANI_03659] No signal-service-translation SOME/IP Service Discovery Find

Status: DRAFT

⌈If there is a RequiredSomeipServiceInstance of category SIGNAL-BASED_NO_HEADER defined for the signal-service-translation, then this RequiredSomeipServiceInstance shall not be handled using SOME/IP service discovery.⌋

[TPS_MANI_03660] No signal-service-translation SOME/IP Service Discovery Offer

Status: DRAFT

⌈If there is a ProvidedSomeipServiceInstance of category SIGNAL-BASED_NO_HEADER defined for the service-signal-translation, then this ProvidedSomeipServiceInstance shall not be handled using SOME/IP service discovery.⌋

9.5.1 Service discovery control

The signal/service translation needs to determine when the translated service (service which originates in signal-based messages) shall actually be offered/subscribed.

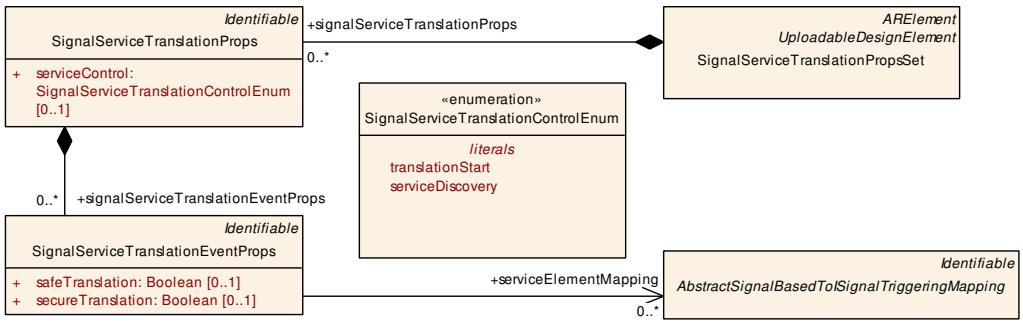


Figure 9.15: Signal/service translation properties

Attributes defining the behavior of signal/service translation are available at the SignalServiceTranslationProps. The reference serviceElement-Mapping determines to which service instance these settings apply.

Class	SignalServiceTranslationPropsSet			
Note	Collection of SignalServiceTranslationProps. **Tags:** atp.recommendedPackage=SignalServiceTranslationProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
signalService Translation Props	SignalService TranslationProps	*	aggr	Collection of SignalServiceTranslationProps. **Tags:** atp.Splitkey=signalServiceTranslationProps, signal ServiceTranslationProps.shortName

Table 9.7: SignalServiceTranslationPropsSet

Class	SignalServiceTranslationProps			
Note	This element allows to define the properties which are applicable for the signal/service translation service.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SignalServiceTranslationPropsSet.signalServiceTranslationProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
serviceControl	SignalService TranslationControlEnum	0..1	attr	Defines how the service instance control shall behave.
signalService Translation EventProps	SignalService TranslationEventProps	*	aggr	Defines properties for a single translated event. **Tags:** atp.Splitkey=signalServiceTranslationEventProps, signalServiceTranslationEventProps.shortName

Table 9.8: SignalServiceTranslationProps

[constr_10255] Multiplicity of attribute `SignalServiceTranslationProps.serviceControl`

Imposition time: `IT_Mani`

⌈For each `SignalServiceTranslationProps`, the attribute `serviceControl` shall exist.⌋

Class	SignalServiceTranslationEventProps			
Note	This element allows to define the properties which are applicable for the signal/service translation event.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SignalServiceTranslationProps.signalServiceTranslationEventProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
safeTranslation	Boolean	0..1	attr	Defined whether the translation shall happen in a safe way.
secure Translation	Boolean	0..1	attr	Defined whether the translation shall happen in a secure way.
serviceElement Mapping	AbstractSignalBasedTo ISignalTriggering Mapping	*	ref	Reference to the collection of SignalBased to ISignal Triggerung mappings the properties apply to. **Tags:** atp.Splitkey=serviceElementMapping This Attribute is only used by the AUTOSAR Adaptive Platform.

Table 9.9: SignalServiceTranslationEventProps

[constr_10230] Multiplicity of attribute SignalServiceTranslationEventProps.safeTranslation

Imposition time: IT_Mani

⌈For each SignalServiceTranslationEventProps, the attribute safeTranslation shall exist.⌋

[constr_10231] Multiplicity of attribute SignalServiceTranslationEventProps.secureTranslation

Imposition time: IT_Mani

⌈For each SignalServiceTranslationEventProps, the attribute secureTranslation shall exist.⌋

At which point in time a specific service instance (originating from signal/service translation) is actually offered / subscribed at the service discovery can be defined per service instance:

Possible approaches for service availability/subscription are:

- translationStart - right after translation software is started
- serviceDiscovery - availability of related service instance

The attribute SignalServiceTranslationProps.serviceControl defines the service instance control behavior.

Enumeration	**SignalServiceTranslationControlEnum**
Note	This enumeration allows to define how the service instance offer/subscribe control shall behave.
Aggregated by	SignalServiceTranslationProps.serviceControl
Literal	***Description***
serviceDiscovery	Defines the start of service control when other service is available. **Tags:** atp.EnumerationLiteralIndex=2
translationStart	Defines the start of service control at translation start. **Tags:** atp.EnumerationLiteralIndex=0

Table 9.10: SignalServiceTranslationControlEnum

9.5.2 Service control right after translation start

If the availability of the signal-based PDUs is not controlled then the respective translated service offers and subscriptions may be activated immediately at start of the translation software.

[TPS_MANI_03580] Service offer at startup ⌈For a provided translated service instance, if the SignalServiceTranslationProps.serviceControl equals translationStart then the translation software shall - right after translation software start - *offer* the respective translated service instance.⌋

[TPS_MANI_03581] Service find at startup ⌈For a required translated service instance, if the SignalServiceTranslationProps.serviceControl equals translationStart then the translation shall right after translation software start issue the *find* of the respective translated service instance and *subscribe* to its event groups.⌋

9.5.3 Service control due to availability of related service instance

There are scenarios where the signal-based PDUs are actually controlled using SOME/IP Service Discovery. Services are defined using events, methods, or fields, and the service instances are offered / subscribed using the SOME/IP Service Discovery, but the payload of such services is not serialized according to the SOME/IP transformer rules (or a subset of events uses signal-based serialization). Therefore, a signal/service translation is required for the payload.

9.5.3.1 Signal-service-translation

[TPS_MANI_03582] Service find for required signal ⌈For a required translated service instance, if the SignalServiceTranslationProps.serviceControl equals serviceDiscovery then upon startup the translation software component shall issue a *service find* for the RequiredSomeipServiceInstance which is referenced by the ServiceInstanceToSignalMapping in the role serviceInstance.⌋

[TPS_MANI_03583] Service subscribe for required signal ⌈For a required translated service instance, if the SignalServiceTranslationProps.service-Control equals serviceDiscovery and the *service find* of [TPS_MANI_03582] was successful then the translation software component shall issue a *subscribe* to all SomeipServiceInterfaceDeployment.eventGroups.⌋

9.5.3.2 Service-signal-translation

[TPS_MANI_03606] Service offer for provided signal ⌈For a provided translated service instance, if the SignalServiceTranslationProps.serviceControl equals serviceDiscovery then upon startup the translation software component shall issue an *offer* for the ProvidedSomeipServiceInstance which is referenced by the ServiceInstanceToSignalMapping in the role serviceInstance.⌋

9.5.4 Static Communication Configuration

If no service discovery shall be applied, it is possible to define a static configuration for non SOME/IP communication. The actual representation of the required information

is not standardized, thus the information can be provided using the model extension mechanism, as exemplified in [8, TPS Generic Structure Template].

9.5.4.1 Static Communication Configuration for the Server

If a ProvidedUserDefinedServiceInstance shall be statically configured for communication on Ethernet, then several elements need to be defined:

- localUdpPortNumber: a port number for the local UDP port needs to be defined on which the service will be provided using UDP protocol.
- localTcpPortNumber: a port number for the local TCP port needs to be defined on which the service will be provided using TCP protocol.
- UserDefinedServiceInstanceToMachineMapping: a mapping needs to be defined (for the local connection where the service will be provided) that associates the ProvidedUserDefinedServiceInstance to an EthernetCommunicationConnector (VLAN).
- remotePeers: a list of remote IP-Address and Port combinations needs to be defined where the messages will be sent to. A mix of unicast and multicast IP-Addresses is supported.
- udpCollectionBufferTimeout: Maximum time an outgoing message may be delayed due to data collection.
- udpCollectionTrigger: Defines whether the element contributes to the triggering of the UDP data transmission if data collection is enabled.
- udpCollectionBufferSizeThreshold: Specifies the amount of data in bytes that shall be buffered for data transmission over the UDP connection if data collection is enabled.

9.5.4.2 Static Communication Configuration for the Client

If a RequiredUserDefinedServiceInstance shall be statically configured for communication on Ethernet, then several elements need to be defined:

- localUdpPortNumber: a port number for the local UDP port needs to be defined on which the service will be consumed using UDP protocol.
- localTcpPortNumber: a port number for the local TCP port needs to be defined on which the service will be consumed using TCP protocol.
- UserDefinedServiceInstanceToMachineMapping: a mapping needs to be defined (for the local connection where the service will be consumed) that associates the RequiredUserDefinedServiceInstance to an EthernetCommunicationConnector (VLAN).

- eventMulticastUdpPort: a port number for the local UDP port in the case of multicast reception may be defined.
- multicastIpAddress: an IP-Address where the events will be consumed in the case of multicast reception.
- remotePeer: an IP-Address and Port combination needs to be defined for the remote provider of the service.

9.6 Translation Behavior

The signal/service translation is defined as a network binding for the [32, SWS Communication Management] and it is not specified at which exact point in time or in which context the data transformation (i.e. signal-service-translation or service-signal-translation) will be executed.

The behavior of signal-service-translation is governed by the application software which calls ara::com-APIs to interact with the communication management.

9.6.1 ServiceInterface Representation of Translation Data

Due to update-bits (see Chapter 9.6.2) and data filtering (see Chapter 9.6.3), the ServiceInterface – which is used to represent the signal/service translated information – needs to be able to represent partially available information. For such cases, the AUTOSAR data type system has defined the isOptional feature of structured data (see also the [2, TPS Software Component Template] for details).

For ApplicationDataTypes, the optionality is configured at the ApplicationRecordElement.isOptional attribute.

For CppImplementationDataTypes, the optionality is configured at the CppImplementationDataTypeElement.isOptional attribute.

The goal of the signal/service translation mapping is to define the proper ServiceInterface representation for the corresponding ISignalIPdu, respecting update bits and filtering.

9.6.1.1 Optional Elements in the Case signal-service-translation

The approach for the signal-service-translation direction is: if a received ISignal or ISignalGroup has an update-bit defined (see Chapter 9.6.2) or the reception is guarded by a reception-filter (see Chapter 9.6.3), then the corresponding element in the ServiceInterface event shall be defined as ApplicationRecordElement.isOptional (resp. CppImplementationDataTypeElement.isOptional).

An example for the handling of optional elements in signal/service translation is given in Chapter B.10.2.1.

9.6.1.2 Optional Elements in the Case `service-signal-translation`

The approach for the `service-signal-translation` direction is: if the `ServiceInterface` event has a member with `CppImplementationDataTypeElement.isOptional`, then the corresponding `ISignal` or `ISignalGroup` shall have an update-bit defined. During runtime, it will be checked which optional elements are actually defined in the event / field and the corresponding values are put into the `ISignalIPdu`. Also the update-bits are set accordingly.

An example for the handling of optional elements in signal/service translation is given in Chapter B.10.2.2.

9.6.2 Event and update-bit Semantics

A variety of sources may exist where the `ISignalIPdu`s originate that are candidates for `signal/service translation`. The `ISignalIPdu`s produced by the `signal/service translation` may be routed to many different targets in the vehicle network. The semantics of the received and sent data in the scope of the `signal/service translation` are essential.

Often `ISignalIPdu`s are transported using periodic transmission in order to not only transport data but also have a liveliness indication of the information producer and the communication path.

If the sender would just transmit an `ISignalIPdu` when the value changes, a receiver would not be able to detect whether the sender just did not send anything because there is no new information or whether the sender (or the transport path) has a malfunction and therefore an intended sent message was not able to get through.

In the simplest case, the arrival of one `ISignalIPdu` actually represents the functional reception of all the included information. This means that the resulting event (resp. field) for the application software can be derived from the reception of that `ISignalIPdu`. In the opposite direction, the notification of an event (resp. field) from the application software results in the sending of the `signal/service translation ISignalIPdu`.

But there are scenarios where the layout of the `ISignalIPdu` is composed of information from different sources, or the information is produced at different points in time. In order to indicate which parts of the `ISignalIPdu` actually have been updated and which parts have just been sent because they are part of the `ISignalIPdu`, the *update-bits* are used.

These update-bits are part of the ISignalIPdu payload and indicate whether the associated information has actually been sent intentionally or just has been sent because the ISignalIPdu was sent as a whole.

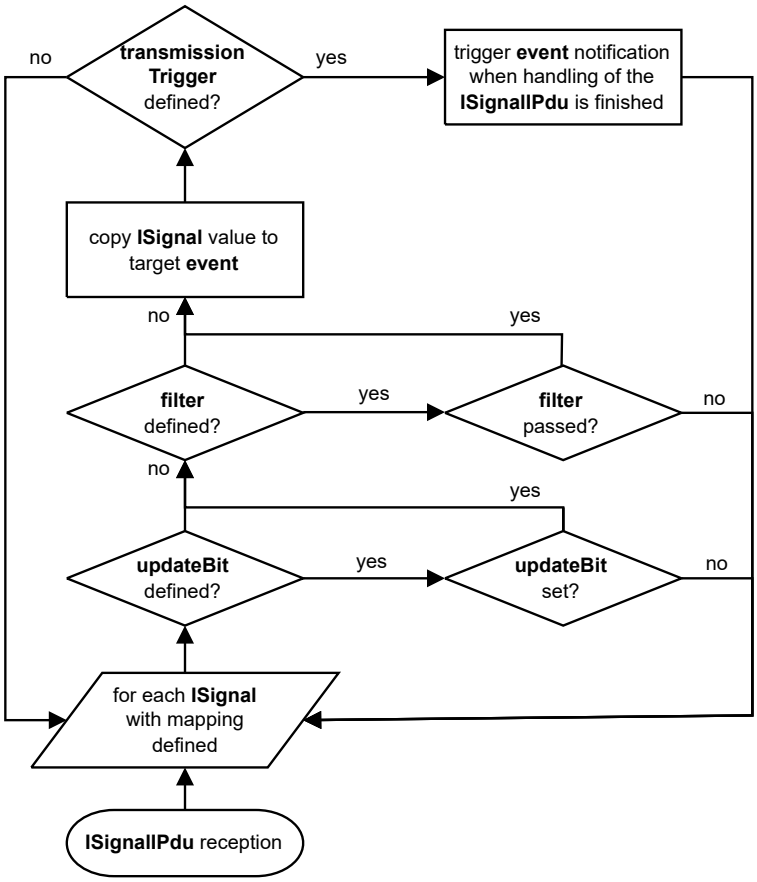


Figure 9.16: Processing order reception

If no update-bits are used, then every arrival of an ISignalIPdu has to be considered as being updated by the producer as a whole.

If update-bits are used, then not all of the included information may actually have been produced by the sender. In an extreme case – if all information of the ISignalIPdu has associated update-bits – all of the update-bits might not be set, resulting in reception of an ISignalIPdu but without any effect on the signal-service-translation.

Another way to influence the translation behavior is the usage of reception data filters during signal-service-translation (see also Chapter 9.6.3). This also may lead to the situation that the reception of an ISignalIPdu only triggers a subset of the contained information (or none at all).

The third check during signal-service-translation is the evaluation of the SignalBasedEventElementToISignalTriggeringMapping.transmissionTrigger (resp. SignalBasedFieldToISignalTriggeringMapping.

transmissionTrigger). If an ISignal was actually taken to update the resulting event, the transmissionTrigger defines whether that update actually causes the notification of the mapped event.

One use-case may be to define which signal-service-translation elements actually cause the sending of an event. For example, if – due to update-bits and filters – only the least important elements have been translated, it can be defined that the event is not triggered.

[constr_3644] No transmissionTrigger support for service-signal-translation direction

Imposition time: IT_Mani

⌈If a ServiceInstanceToSignalMapping.serviceInstance refers to a ProvidedApServiceInstance then

- every SignalBasedEventElementToISignalTriggeringMapping aggregated in the role eventElementMapping and
- every SignalBasedFieldToISignalTriggeringMapping aggregated in the role fieldMapping

shall not have a SignalBasedEventElementToISignalTriggering-Mapping.transmissionTrigger (respectively SignalBasedFieldToISignal-TriggeringMapping.transmissionTrigger) defined.⌋

[TPS_MANI_03646] Definition of transmission triggers for signal-service-translation ⌈The attribute SignalBasedEventElementToISignal-TriggeringMapping.transmissionTrigger, respectively SignalBased-FieldToISignalTriggeringMapping.transmissionTrigger, defines which translation parts contribute to the transmission triggering for the mapped payload.⌋

[TPS_MANI_03647] Full translation before transmission triggering ⌈if there has been a transmission trigger caused by a source signal, the signal-service-translation shall first process all other mapped source signals for that target before sending out the target.⌋

[TPS_MANI_03648] Transmission trigger for signal-service-translation ⌈If the attribute SignalBasedEventElementToISignalTriggering-Mapping.transmissionTrigger, respectively SignalBasedFieldToISignal-TriggeringMapping.transmissionTrigger, equals *true* then the translation of the respective source signal causes the sending of the target (after all mapped sources have been processed, see [TPS_MANI_03647]).⌋

[TPS_MANI_03649] No transmission trigger for signal-service-translation ⌈If the attribute SignalBasedEventElementToISignalTriggering-Mapping.transmissionTrigger, respectively SignalBasedFieldToISignal-TriggeringMapping.transmissionTrigger, is not defined or has the value

false, then the translation of that respective source signal does not cause the sending of the target.⌋

If none of the SignalBasedEventElementToISignalTriggeringMapping or SignalBasedFieldToISignalTriggeringMapping contributing to the target of a signal-service-translation has the attribute SignalBasedEvent-ElementToISignalTriggeringMapping.transmissionTrigger (respectively SignalBasedFieldToISignalTriggeringMapping.transmissionTrigger) set to *true*, then the translation target will not be sent.

The combination of update-bits, filters, and transmission triggers may be configured for one ISignal – resulting in a sequential processing according to Figure 9.16.

9.6.3 Data Filtering

Filtering can be used to potentially reduce the amount of generated event notifications in the case of signal-service-translation direction. There is no filtering support available for the service-signal-translation direction, although the transmission mode selection can be used to influence the amount of sent ISignalIPdus (see Chapter 9.8.5).

[constr_3643] No filter support for service-signal-translation direction

Imposition time: IT_Mani

⌈If a ServiceInstanceToSignalMapping.serviceInstance refers to a ProvidedApServiceInstance then

- every SignalBasedEventElementToISignalTriggeringMapping aggregated in the role eventElementMapping and
- every SignalBasedFieldToISignalTriggeringMapping aggregated in the role fieldMapping

shall not have a SignalBasedEventElementToISignalTriggering-Mapping.filter (resp. SignalBasedFieldToISignalTriggering-Mapping.filter) defined.⌋

If a data filtering shall be applied during the signal-service-translation then it is possible to define a DataFilter at the SignalBasedEvent-ElementToISignalTriggeringMapping and SignalBasedFieldToISignal-TriggeringMapping in the role filter.

[TPS_MANI_03621] Data filtering during the signal-service-translation ⌈If there is a SignalBasedEventElementToISignalTriggeringMapping.filter (resp. SignalBasedFieldToISignalTriggeringMapping.filter) defined this filtering shall be implemented during the signal-service-translation.

This is the only supported application of filters for signal-service-translation.⌋

[TPS_MANI_03644] Supported reception data filters ⌈The following `DataFilter` settings are supported by the `signal-service-translation`:

- `always`
- `maskedNewEqualsX`
- `maskedNewDiffersX`
- `maskedNewDiffersMaskedOld`

⌋

The setting of `maskedNewDiffersMaskedOld` requires the translation software to keep the previous (*old*) value for comparison.

The possibility to define an `always` filter allows to specify a filter for each mapped signal, even if that particular signal shall not contribute to the filter result - as it always yields *passed*. This allows to define a full set of filters and thus gives the possibility to check for complete filter definitions.

The usage of a *never* filter is not supported by the `signal-service-translation`. Such a filter would always yield *not passed*, which is equivalent to not having a `SignalBasedEventElementToISignalTriggeringMapping` defined at all for this specific `ISignal`.

Potentially there could be several `SignalBasedEventElementToISignal-TriggeringMapping.filter`s (resp. `SignalBasedFieldToISignal-TriggeringMapping.filter`s) defined in the context of one target element (in the example Figure 9.17, the target is the *event xyz* with the elements *x1* and *z*).

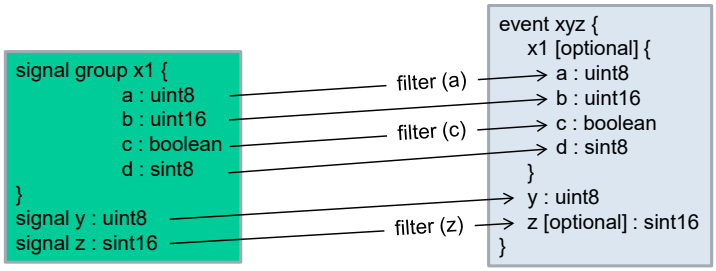


Figure 9.17: Filter example during `signal-service-translation`

In such a case all of the defined filters for the `scoped filtered element` have to pass in order for the `signal/service translation` to take effect. For the optional element *x1* to actually be set, both the *filter (a)* and *filter (c)* have to pass. The optional element *z* will only be set if the *filter (z)* passes.

If the target event of the `signal-service-translation` has no optional elements defined and at least one filter does not pass, then the whole event is not sent. This is illustrated in Figure 9.18, where the *event xyz* is considered to be the scoped element.

The `scoped filtered element` is defined as either

- the filtered element, if it is defined as *optional*
- the closest *optional* element traversing from the filtered element and going towards the root element
- the root element itself, if there is no *optional* element traversing from the filtered element and going towards the root element.

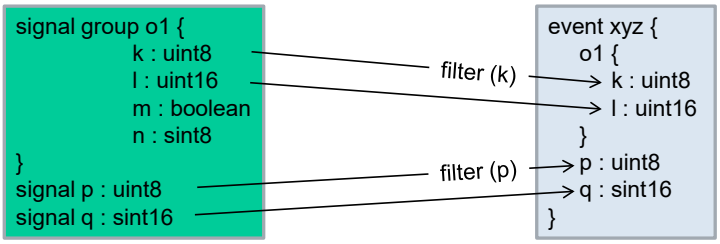


Figure 9.18: Filter example during `signal-service-translation` with no optional target elements

[TPS_MANI_03645] Applicability of a filter to the `scoped filtered element` ⌈If a filter does not pass then the `scoped filtered element` shall not be considered for `signal-service-translation`.⌋

9.6.4 Translation from one Source

The simplest translation approach is a one source translation. If the translation target is primitive then it has by nature only one source. If the translation target is a composite data type then it depends on the structure of the sources whether a one source translation is possible.

[TPS_MANI_03620] Service discovery control ⌈If the service discovery control is enabled for a specific service instance then any payload coming from or going to that service instance shall be translated and forwarded to the mapped output(s).⌋

Every time an input signal or event arrives it is translated to the mapped output path. Since this is a one source translation, the arrival of the input signal or event is the trigger for the translation. Therefore, if all the translation mappings have been performed the target payload can be sent out.

The example in Figure 9.19 illustrates the full approach of a `signal-service-translation` with application component forwarding of the payload.

On the reception side, the `ISignalIPdu` *uv1* is the source of the `signal-service-translation`. The payload of the `ISignalIPdu` is translated into an instance of the `event` *uv2*. The `event` is part of a `ServiceInterface` that types the `RPort-Prototype` which defines the ara::com API at that `RPortPrototype`.

It is the job of the application component code to use the ara::com API and fetch the information from the `RPortPrototype` and forward it to the `PPortPrototype`. In

this example, it is not further defined how that forwarding is implemented, but for such a simple use-case the usage of a PassThroughSwConnector could be used to express the desired forwarding. For further details on this approach, refer to Chapter 9.7. A custom code implementation can also be used to fetch the data and forward it.

At the PPortPrototype a ServiceInterface with an event *uv3* is defined. This is the basis to provide the information in a service-oriented manner to further software components. What kind of network binding is used for the transport of the PPortPrototype payload is not in scope of this example. However, the PPortPrototype could be mapped to any kind of network binding (SOME/IP, DDS, IPC, or even another service-signal-translation).

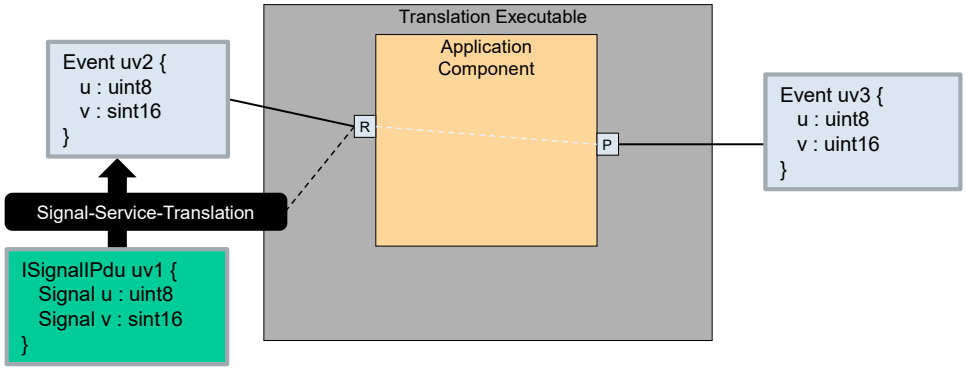


Figure 9.19: Example signal-service-translation and forwarding from one source

9.6.5 Translation from several Sources

There is also the use-case to support the composition of data from different sources. However, because the sources will most likely not be received in a coordinated way, it would be hard to define an automatic behavior based on the usage of PassThroughSwConnectors for this setup. Thus, for the usage of PassThroughSwConnectors, it is required by the signal/service translation that the input for the signal/service translation comes from one source.

The merging of data from several sources can be implemented on application component level and still utilize the signal/service translation features to interface with ISignals and ISignalIPdus. The code, however, which reads several sources of data and produces a consistent set of information is considered to be custom code.

An example for signal/service translation from different sources can be found in Chapter B.10.2.3.1. Conversely, an example for service/signal translation from different sources is documented in Chapter B.10.2.3.2.

9.6.6 Signal-Service-Translation to several Targets

If the source ISignalIPdu in a signal-service-translation scenario is composed of several parts (ISignalGroups and/or individual ISignals), then it is possible to map the parts of that ISignalIPdu to several targets (events or field notifiers).

[TPS_MANI_03667] Mapping from one ISignalIPdu to several events in the case of signal-service-translation ⌈The ISignals of one ISignalIPdu can be mapped to one or more events (using SignalBasedEventElementToISignalTriggeringMapping.iSignalTriggering) or field notifiers (using SignalBasedFieldToISignalTriggeringMapping.notifierSignalTriggering).⌋

The individual SignalBasedEventElementToISignalTriggeringMappings and SignalBasedFieldToISignalTriggeringMappings will define in their DataPrototypeInServiceInterfaceRef the target of the signal-service-translation.

One scenario requires mapping parts of an ISignalIPdu to different events. This applies if the source ISignalIPdu is composed of several parts (ISignalGroups and/or individual ISignals) and at least one of the ISignalGroups is end-to-end protected.

In such cases it is required to get the end-to-end status per protected ISignalGroup, thus merging the whole ISignalIPdu's content into one event does not suffice, as only a part of the ISignalIPdu is actually end-to-end protected, and the ara::com status APIs are available on an event level.

The modeling approach according to Chapter 9.4.3 results in one or more ServiceInstanceToSignalMappings which distribute the content of one ISignalIPdu to several events.

An example for the signal/service translation to several targets can be found in Chapter B.10.2.4.

9.7 Translation pass-through Composition

If a *Pass-through Composition* is defined (see Figure 9.4), the behavior for the defined pass-through definition shall be implemented. It is possible to automatically generate the implementation of such a pass-through behavior out of the information given in the *Pass-through Composition*.

The mapping approach is already used in Chapter 4.3 where the PassThroughSwConnector is used to define the matching PortPrototypes for the *facade* use-case. Specifically [constr_5056] applies as well for the signal/service translation.

SOME/IP services may consist of a mixture of events, fields and methods. While events and field notifiers are (potentially) subject to signal/service translation it is

well-defined that methods (methods, getter- and setter-methods, and fire-and-forget methods) are serialized according to the SOME/IP serialization rules.

However, the mixed nature of a specific service instance means that methods also need to be considered in the *Pass-through Composition*. It is not so much about the translation of the payload serialization, but the general wrapping of events, fields and methods in one service instance makes it necessary that methods get *passed through* as well.

9.8 Expected Features of Classic Platform

Classic AUTOSAR SWS-COM Specification provides a variety of possibilities to pack signals into PDUs – from a structural as well as from a behavioral perspective. Also, the further layers in the COM-Stack may influence the PDUs layout.

The goal of this section is to specify which of these Classic COM-Stack features shall also be available on the Adaptive Platform `signal/service translation`.

9.8.1 Processing Order

For the features of the Classic platform COM-Stack there is a well-defined processing order in which the actions are performed upon Tx and Rx of data. It is required for the `signal/service translation` to ensure the same processing order.

[TPS_MANI_03585] Processing order of COM-Stack features ⌈For the COM-Stack features the `signal/service translation` supports the processing order shall be the same as on the Classic platform.⌋

9.8.2 Reception Data Filter

The reception filter support for `signal-service-translation` is defined in Chapter 9.6.3.

The potentially defined reception filters at an `ISignalPort` shall be ignored, as [TPS_MANI_03621] is the only supported way to define `filter`s for `signal-service-translation`.

[TPS_MANI_03589] Reception data filter of COM-Stack ⌈If the `ISignal-Triggering` refers to an `ISignalPort` and that `ISignalPort` has a `dataFilter` defined then the `signal-service-translation` shall ignore the filter definition.⌋

9.8.3 Reception of Invalid Signal

[TPS_MANI_03592] **`ISignal` invalidation of COM-Stack** ⌈If the `ISignalTriggering` refers to an `ISignalPort` with `communicationDirection` equal `in` then the following values for `ISignalPort.handleInvalid` shall be supported by the `signal/service translation`:

- `dontInvalidate`
- `replace`

⌋

[TPS_MANI_03593] **`handleInvalid` = `dontInvalidate` behavior of COM-Stack** ⌈If the `ISignalTriggering` refers to an `ISignalPort` with `communicationDirection` equal `in` and the `ISignalPort.handleInvalid` equals `dontInvalidate` then the `signal/service translation` shall not perform any invalidation handling.⌋

[TPS_MANI_03594] **`handleInvalid` = `replace` behavior of COM-Stack** ⌈If the `ISignalTriggering` refers to an `ISignalPort` with `communicationDirection` equal `in` and the `ISignalPort.handleInvalid` equals `replace` and the received value of the `ISignal` equals the `ISignal.networkRepresentationProps.invalidValue` then the `signal/service translation` shall *replace* the `invalidValue` with the `ISignal.initValue`.⌋

9.8.4 Update Bit Handling

For details on the handling of update bits refer to Chapter 9.6.

[TPS_MANI_03595] **Update Bit support for `ISignal`** ⌈If the `ISignalTriggering` refers to an `ISignalPort` with `communicationDirection` set to the value `in` and the `ISignalTriggering` refers to an `ISignal` and that `ISignal` is mapped into an `ISignalIPdu` with a `ISignalToIPduMapping.updateIndicationBitPosition` defined, then the following rules apply:

- If the received *update bit* is *true*, then the respective `ISignal` shall be considered for reception.
- If the received *update bit* is *false*, then the respective `ISignal` shall not be considered for reception.

⌋

9.8.5 Transfer Properties and Transmission Modes for Service-Signal-Translation

[TPS_MANI_03590] Transfer properties and transmission modes of COM-Stack
⌈If the ISignalTriggering refers to an ISignalPort with communicationDirection equal out and the ISignalTriggering refers to an ISignal and that ISignal is mapped into an ISignalIPdu with an iPduTimingSpecification and the IPduTiming has a transmissionModeDeclaration defined and the TransmissionModeDeclaration has a transmissionModeCondition defined and the following values for TransmissionModeCondition.dataFilter shall be supported by the service-signal-translation:

- always
- never
- maskedNewEqualsX
- maskedNewDiffersX

⌋

Class	**TransmissionModeCondition**			
Note	Possibility to attach a condition to each signal within an I-PDU. If at least one condition evaluates to true, TRANSMISSION MODE True shall be used for this I-Pdu. In all other cases, the TRANSMISSION MODE FALSE shall be used.			
Base	*ARObject*			
Aggregated by	TransmissionModeDeclaration.transmissionModeCondition			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataFilter	DataFilter	0..1	aggr	Possibilities to define conditions **Tags:** atp.Splitkey=dataFilter
iSignalInIPdu	ISignalToIPduMapping	0..1	ref	Reference to a signal to which a condition is attached. **Tags:** atp.Splitkey=iSignalInIPdu

Table 9.11: TransmissionModeCondition

Class	**TransmissionModeDeclaration**			
Note	AUTOSAR COM provides the possibility to define two different TRANSMISSION MODES (True and False) for each I-PDU. As TransmissionMode selector the signal content can be evaluated via transmissionModeCondition (implemented directly in the COM module) or mode conditions can be defined with the modeDrivenTrue Condition or modeDrivenFalseCondition (evaluated by BswM and invoking Com_SwitchIpduTxMode COM API). If modeDrivenTrueCondition and modeDrivenFalseCondition are defined they shall never evaluate to true both at the same time. The mixing of Transmission Mode Switch via API and signal value is not allowed.			
Base	*ARObject*			
Aggregated by	IPduTiming.transmissionModeDeclaration			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**TransmissionModeDeclaration**			
modeDriven FalseCondition	ModeDriven TransmissionMode Condition	*	aggr	Defines the trigger for the Com_SwitchIpduTxMode Transmission Mode switch. Only if all defined modeDriven FalseConditions evaluate to true (AND associated) the transmissionModeFalseTiming shall be activated. mode DrivenTrueCondition and modeDrivenFalseCondition shall never evaluate to true both at the same time. **Tags:** atp.Splitkey=modeDrivenFalseCondition
modeDriven TrueCondition	ModeDriven TransmissionMode Condition	*	aggr	Defines the trigger for the Com_SwitchIpduTxMode Transmission Mode switch. Only if all defined modeDriven TrueConditions evaluate to true (AND associated) the transmissionModeTrueTiming shall be activated. mode DrivenTrueCondition and modeDrivenFalseCondition shall never evaluate to true both at the same time. **Tags:** atp.Splitkey=modeDrivenTrueCondition
transmission ModeCondition	TransmissionMode Condition	*	aggr	The Transmission Mode Selector evaluates the conditions for a subset of signals and decides which transmission mode should be used. In case only one transmission mode is used there is no need for the "TransmissionMode Condition" and its sub-structure. In case the transmission mode shall be switched using the COM-API "Com_Switch IpduTxMode" there is no need for the "TransmissionMode Condition" and its sub-structure. **Tags:** atp.Splitkey=transmissionModeCondition
transmission ModeFalse Timing	TransmissionMode Timing	0..1	aggr	Timing Specification if the COM Transmission Mode is false. The Transmission Mode Selector is defined to be false, if all Conditions evaluate to false. **Tags:** atp.Splitkey=transmissionModeFalseTiming
transmission ModeTrue Timing	TransmissionMode Timing	0..1	aggr	Timing Specification if the COM Transmission Mode is true. The Transmission Mode Selector is defined to be true, if at least one Condition evaluates to true. **Tags:** atp.Splitkey=transmissionModeTrueTiming

Table 9.12: TransmissionModeDeclaration

9.8.6 Deadline Monitoring

The application software is responsible for the deadline monitoring.

9.8.7 Signal and IPdu Transmission

The application software is responsible for the periodic production of data and triggering of sending.

9.8.8 IPdu Multiplexing

On low-payload networks, IPdu Multiplexing is used to handle the limited number of identifiers for the respective network transport (e.g. CAN-Ids). The potential alternatives of a MultiplexedIPdu shall be represented as optional alternative CppImplementationDataType of category STRUCTURE in the ServiceInterface event or field.

[TPS_MANI_03597] Support for MultiplexedIPdu ⌈The signal/service translation shall support the handling of MultiplexedIPdu defined payload. The support shall be available for sending and receiving of MultiplexedIPdus.⌋

[TPS_MANI_03638] Mapping of MultiplexedIPdu ⌈The content of the MultiplexedIPdu shall be mapped to a CppImplementationDataType of category STRUCTURE consisting of the elements:

- MultiplexedIPdu static part: directly mapped into the top-level structure
- MultiplexedIPdu dynamic part: each dynamic part alternative shall be mapped inside a CppImplementationDataTypeElement of category STRUCTURE

⌋

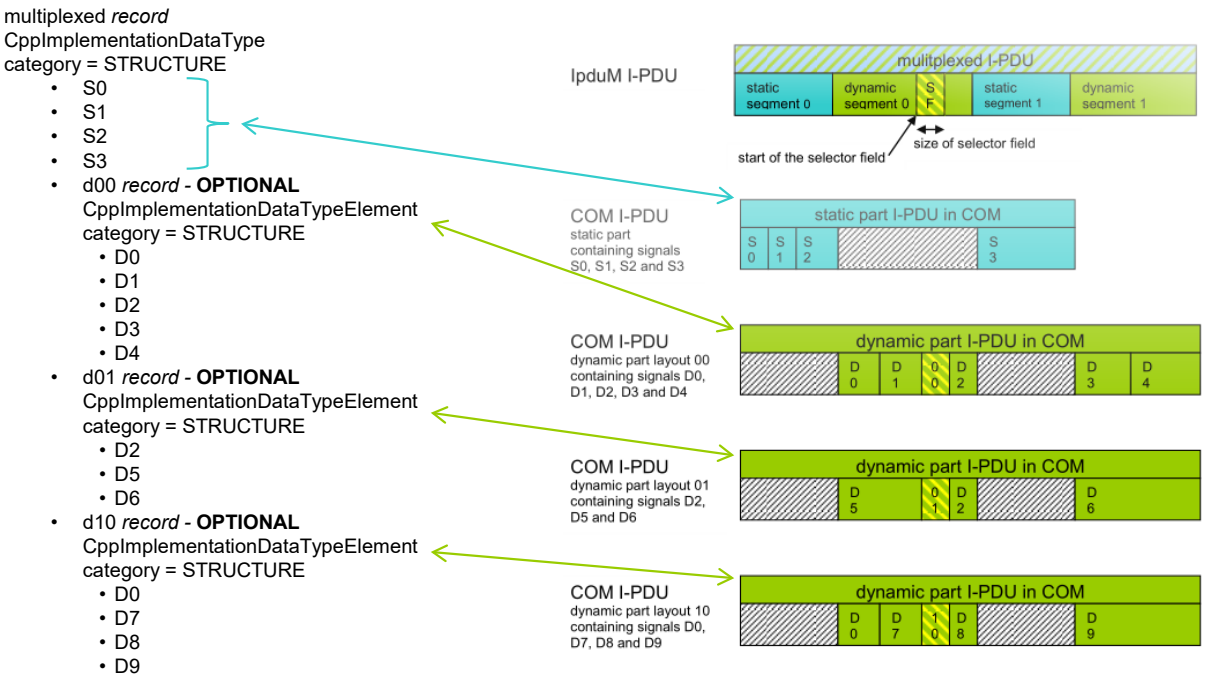


Figure 9.20: Example mapping for MultiplexedIPdu

[TPS_MANI_03639] Mapping of MultiplexedIPdu static part ⌈The content of the MultiplexedIPdu static part shall be mapped to a CppImplementationDataType of category STRUCTURE defined for the whole MultiplexedIPdu in [TPS_MANI_03638].⌋

[TPS_MANI_03640] Mapping of `MultiplexedIPdu` dynamic part ⌈The content of each `MultiplexedIPdu` dynamic part shall be mapped to one of the optional alternatives of the `CppImplementationDataTypeElement` of `category STRUCTURE` with `CppImplementationDataTypeElement.isOptional` set to *true* for each dynamic part in [TPS_MANI_03638].⌋

Even in cases where there is just one member mapped from a `MultiplexedIPdu` dynamic part there shall still be a wrapping `CppImplementationDataTypeElement` of `category STRUCTURE` defined according to [TPS_MANI_03640].

[TPS_MANI_03641] No mapping of `MultiplexedIPdu` selector field value ⌈The selector field value of a `MultiplexedIPdu` shall not be mapped in the context of `signal-service-translation` (or `service-signal-translation`).⌋

In order to keep the status of the event or field of the `ServiceInterface` consistent, exactly one of the optional alternatives shall be used.

[TPS_MANI_03642] Exactly one alternative defined for mapped event / field during runtime ⌈Exactly one option (as defined in [TPS_MANI_03640] for the dynamic part) shall be defined in the event / field representing the `MultiplexedIPdu` during runtime.⌋

In Figure 9.20 an example of the mapping between a `MultiplexedIPdu` and the composite data type is illustrated.

9.9 End-to-End considerations

The `signal/service translation` on the Adaptive platform follows a single-sided approach on ara::com. To the application software there is no difference whether the service is communicated using SOME/IP or DDS, or whether the service originates in a `signal/service translation`. As the translation is running in the same `Process` as the application software itself, the application software has direct access to the translated service payload.

If there is an E2E header attached and/or a secure communication defined, the translation software needs to check the validity (for reception) or calculate the CRC/MAC (for sending) of signal-based payload.

The information whether and which E2E / Security profile is configured is defined in the [4, TPS System Template] part belonging to the mapping information of Chapter 9.4.3 and Chapter 9.4.5.

The assignment of the `ISignal`s to `ISignalIPdu`s which in turn are assigned to `SecuredIPdu`s defines the security aspects.

The assignment of `ISignal`s to `EndToEndTransformationDescription`s define the safety aspects.

9.9.1 Safety

The attribute SignalServiceTranslationEventProps.safeTranslation is used to explicitly require that the translation shall be configured in a safe transport and that the translation software shall handle the translation activity in an end-to-end preserving way.

[TPS_MANI_03607] Handling of safe signal/service translation in one Executable ⌈It is required that the signal-service-translation (and service-signal-translation) of one Service/SignalGroup pair which are mapped to each other, shall be handled in one Executable to also cover a closed mapping from one E2E profile to another, if necessary.

The signal/service translation of different (independent) Services/SignalGroups may be handled by different Executables.⌋

[TPS_MANI_03608] Support for safe signal/service translation ⌈The translation of E2E protected data shall be supported in both directions, signal-service-translation and service-signal-translation.⌋

[TPS_MANI_03609] Support for safe signal/service translation with same or different E2E profiles ⌈The translation of E2E protected data shall support the occurrence of

- the same E2E profile on both sides of the communication and
- different E2E profiles on each side of the communication.

⌋

[TPS_MANI_03610] 1:n mapping for E2E protected data ⌈It shall be possible to map the same E2E protected source data to several E2E protected target data (1:n).⌋

[TPS_MANI_03611] E2E protected target out of E2E protected sources ⌈The content of one E2E protected target shall only be composed out of data from E2E protected sources.⌋

The rationale for [TPS_MANI_03611] is to support the use-case where target data shall be E2E protected and it is composed of several sources.

[TPS_MANI_03614] No translation of not OK E2E protected data out of several sources ⌈If an E2E protected source data is mapped into a composed E2E protected target data (according to [TPS_MANI_03611]) and
if the E2E-Check for the source data returns any E2E error (not *E_OK*) then
this source data shall not be forwarded to the respective target data and (if applicable) shall not trigger the transmission of the target.⌋

If source data is not verified as *E_OK* it is not translated. If the translated E2E protected data comes from several sources there may occur correlation and synchronicity issues during translation.

[TPS_MANI_03612] Sufficient ASIL level of translation software ⌈If the `SignalServiceTranslationEventProps.safeTranslation` equals true then the implementation of the translation software shall fulfill a sufficient ASIL.⌋

[constr_3554] E2E protection configuration check

Imposition time: `IT_Mani`

⌈If the `SignalServiceTranslationEventProps.safeTranslation` equals true then the signal-based payload shall have an EndToEnd profile defined.⌋

The current EndToEnd profiles for the classic platform rely on a periodic communication paradigm. For the translation software of the adaptive platform, this requires the translation software to know the specified period the payload has to be updated / checked for.

[TPS_MANI_03598] Expected check period of E2E-Protected payload ⌈If the `RPortPrototype` has a `ReceiverComSpec.receptionProps.dataUpdatePeriod` defined for an `event` then the application software calling the `ara::com` APIs shall check periodically for updates using the specified period.⌋

[TPS_MANI_03599] Expected update period of E2E-Protected payload ⌈If the `PPortPrototype` has a `SenderComSpec.transmissionProps.dataUpdatePeriod` defined for an `event` then the application software calling the `ara::com` APIs shall periodically update the `event` using the specified period.⌋

Class	**ReceptionComSpecProps**			
Note	This meta-class defines a set of reception attributes which the application software is assumed to implement.			
Base	*ARObject*			
Aggregated by	*ReceiverComSpec*.receptionProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataUpdatePeriod	TimeValue	0..1	attr	This attribute defines the period in which the application shall check for updated data. This attribute is used for the configuration of the E2E protection, but may also indicate a general data reception period.
timeout	TimeValue	0..1	attr	This attribute defines the time interval after which the application shall assume that the to be received data reception has timed out, i.e. the respective data has not been received for that amount of time.

Table 9.13: ReceptionComSpecProps

Class	**TransmissionComSpecProps**			
Note	This meta-class defines a set of transmission attributes which the application software is assumed to implement.			
Base	*ARObject*			
Aggregated by	*SenderComSpec*.transmissionProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**TransmissionComSpecProps**			
dataUpdate Period	TimeValue	0..1	attr	This attribute defines the period in which the application is assumed to transmit the respective data.
minimumSend Interval	TimeValue	0..1	attr	This attribute defines the minimum interval between two consecutive transmissions of the respective data the application is assumed to ensure.
onChangeData Prototype	DataPrototype Reference	*	aggr	This reference defines which DataPrototypes trigger the onChange transmission of the data. **Tags:** atp.Splitkey=onChangeDataPrototype
transmission Mode	TransmissionMode DefinitionEnum	0..1	attr	The attribute defines the mode in which the application is assumed to transmit the respective data.

Table 9.14: TransmissionComSpecProps

Enumeration	**TransmissionModeDefinitionEnum**
Note	This meta-class defines possible settings for the transmission mode.
Aggregated by	TransmissionComSpecProps.transmissionMode
Literal	***Description***
cyclic	The data is assumed to be transmitted in a cyclic manner. The cycle is defined by dataUpdatePeriod. **Tags:** atp.EnumerationLiteralIndex=0
cyclicAndOn Change	The data is assumed to be transmitted in a cyclic manner (with cycle time dataUpdatePeriod) and additionally there may be arbitrary transmission if the data value changes (minimumSendInterval to be respected, if defined). Which data prototypes trigger the transmission if their value changes is defined by TransmissionComSpecProps.onChangeDataPrototype. **Tags:** atp.EnumerationLiteralIndex=2
onChange	The data is assumed to be transmitted in an arbitrary manner only if the data value changes (minimumSendInterval to be respected, if defined). Which data prototypes trigger the transmission if their value changes is defined by TransmissionComSpecProps.onChangeDataPrototype. **Tags:** atp.EnumerationLiteralIndex=3
triggered	The data is assumed to be transmitted in an arbitrary manner (minimumSendInterval to be respected, if defined). **Tags:** atp.EnumerationLiteralIndex=1

Table 9.15: TransmissionModeDefinitionEnum

9.9.1.1 `Signal-service-translation`

[TPS_MANI_03600] `Signal-service-translation` **of E2E protected payload** ⌈`Signal-service-translation` shall check the end-to-end status of every received signal-based payload. If the E2E-Check returns *E_OK* for the received payload then the data shall be forwarded to the respective sending of the translation software.⌋

Error handling:

[TPS_MANI_03601] `Signal-service-translation` **of E2E protected payload - timeout handling** ⌈If no message is received within the specified message cycle time (timeout is detected), then no data shall be transmitted to the service-based part.⌋

[TPS_MANI_03602] `Signal-service-translation` **of E2E protected payload - error handling** ⌈If the E2E-Check returns any E2E error (not *E_OK*), then the service-based message shall reflect that E2E error.⌋

This is necessary to provide E2E information to the application software and support an End-to-End view on the data exchange from sender to receiver / provider to consumer.

9.9.1.2 `Service-signal-translation`

[TPS_MANI_03603] `Service-signal-translation` **of E2E protected payload** ⌈`Service-signal-translation` shall check the end-to-end status of every received service-oriented payload. If the E2E-Check returns *E_OK* for the received payload then the data can be forwarded to the respective sending of the translation software.⌋

Error handling:

[TPS_MANI_03604] `Service-signal-translation` **of E2E protected payload - timeout handling** ⌈If no message is received within the specified message cycle time (timeout is detected), then no data shall be transmitted to the signal-based part.⌋

[TPS_MANI_03605] `Service-signal-translation` **of E2E protected payload - error handling** ⌈If the service-oriented payload is handed over with any E2E error (not *E_OK*), then the newly created signal-based E2E protected message shall reflect that E2E error.⌋

This is necessary to provide E2E information to the receiving application and support an End-to-End view on the data exchange from sender to receiver/provider to consumer.

9.9.2 Security

In the context of Signal Service Translation the Secure Onboard Communication *SecOC* (see [33, SWS Secure Onboard Communication]) is the major security technology. Further technologies (like *IPSec* or *TLS*) have not been included in the considerations for `signal/service translation`.

The configuration of *SecOC* on the signal-based communication is defined by having the `ISignalTriggering` used in one of the `signal/service translation` mappings refer to an `ISignal` and that `ISignal` is part of an `ISignalIPdu`. A `PduTriggering` of this `ISignalIPdu` is referenced by a `SecuredIPdu` in the role `payload`.

The `SecuredIPdu` defines all the details which are required to *generate* / *verify* the cryptographic information. The description of the configuration is provided in the [4, TPS System Template] of Classic platform.

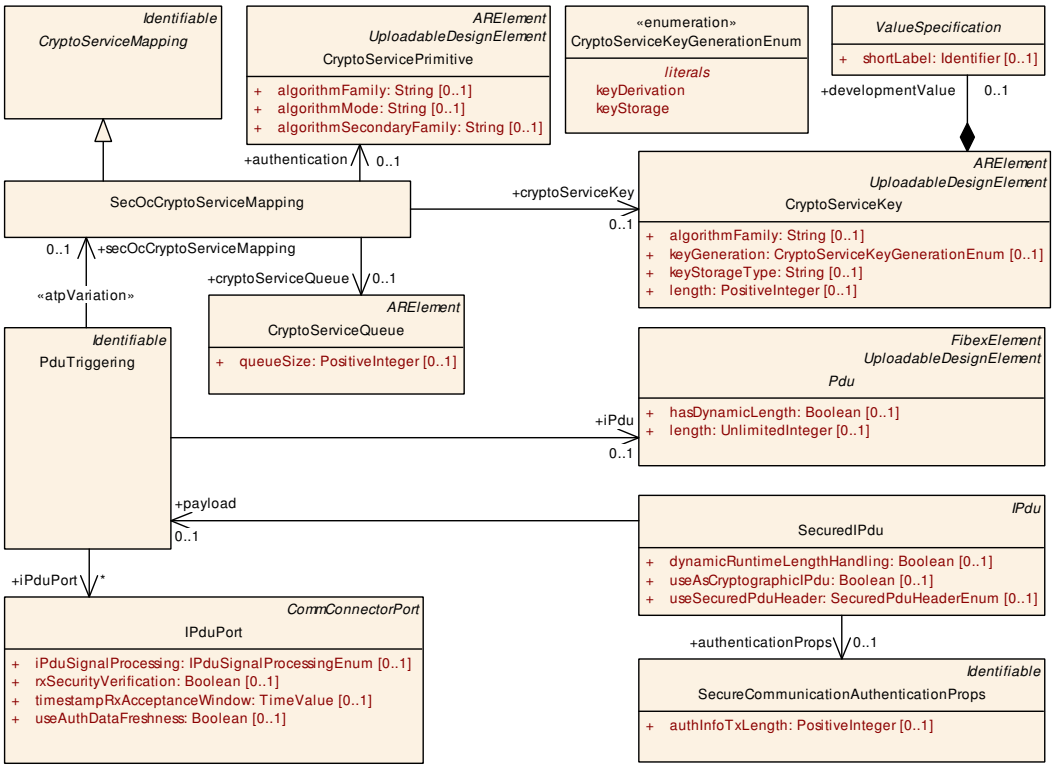


Figure 9.21: SecOC configuration

As the *SecOC* is highly embedded into the Classic platform architecture the `signal/service translation` approach on security is to use the same architecture for its specification.

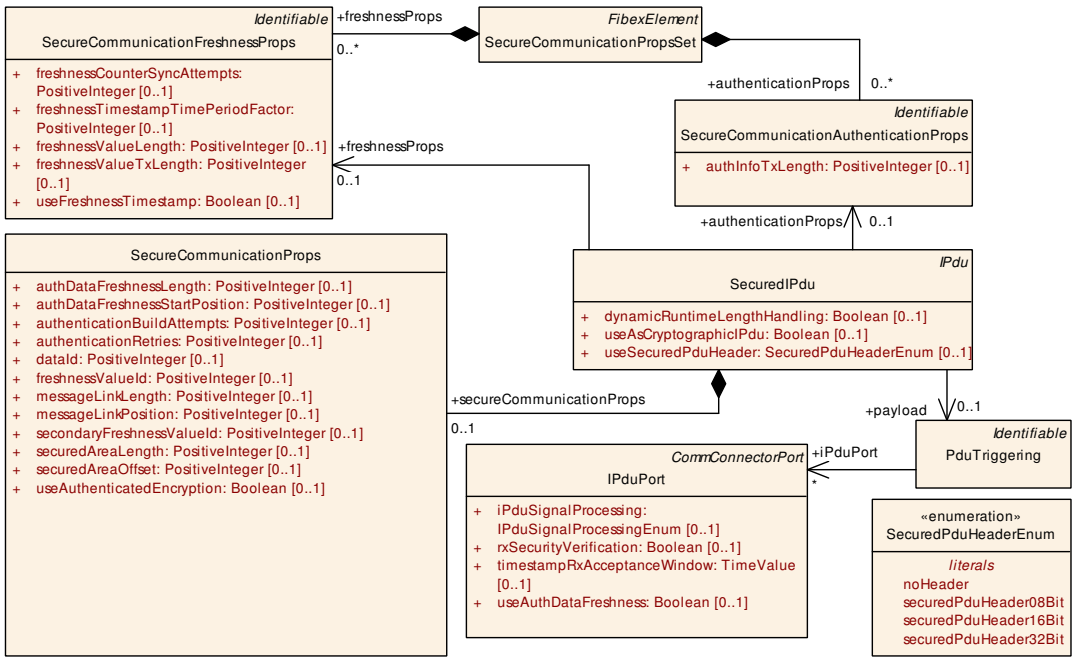


Figure 9.22: Details on `SecuredIPdu`

9.9.2.1 Restrictions on Classic Platform SecOC

[constr_3555] No support for `SecuredIPdu.useAsCryptographicIPdu` set to `true`

Imposition time: IT_Mani

⌈If a PduTriggering is referencing a SecuredIPdu where the attribute useAs-CryptographicIPdu is set to the value true, then no subclass of Abstract-SignalBasedToISignalTriggeringMapping shall refer to an ISignal-Triggering that in turn refers to an ISignal that is referenced in the role iSignal by any ISignalToIPduMapping that is owned by the ISignalIPdu referenced by the SecuredIPdu in the role payload.⌋

The rationale for [constr_3555] is that the separate handling of two PDUs for the transport of one secured message is not supported by the SOME/IP protocol. In such cases the signal/service translation has to be performed on a Classic platform gateway ECU.

9.9.2.2 Link between Classic Platform SecOC and `CryptoKeySlot`

While most of the information on how to configure the SecOC behavior is available in the Classic platform model, there is also the need to define which CryptoKeySlot shall be used for the processing of the SecOC message.

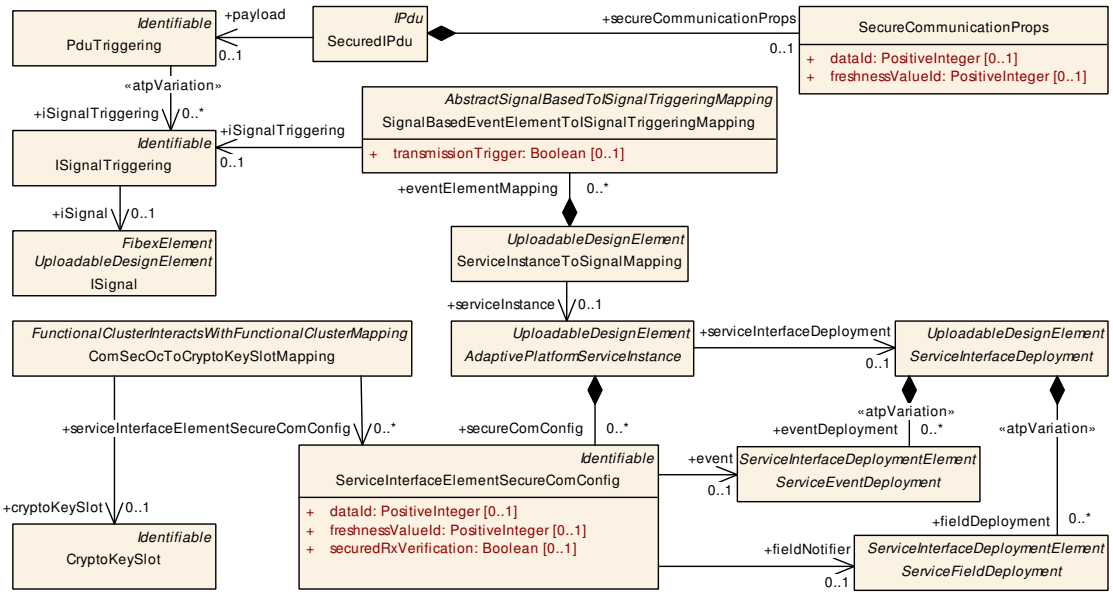


Figure 9.23: Classic SecOC and Adaptive `CryptoKeySlot`

[TPS_MANI_03636] Definition of ServiceInterfaceElementSecureComConfig in the context of signal/service translation ⌈A ServiceInterfaceElementSecureComConfig at the AdaptivePlatformServiceInstance shall be created for each Event or FieldNotifier if the following conditions apply:

- the AdaptivePlatformServiceInstance is the target of a ServiceInstanceToSignalMapping in the role serviceInstance
- the ServiceInstanceToSignalMapping has a SignalBasedEventElementToISignalTriggeringMapping (resp. SignalBasedFieldToISignalTriggeringMapping) that refers to an ISignalTriggering that is part of an ISignalIPdu
- that ISignalIPdu is referenced by a PduTriggering which in turn is referenced by a SecuredIPdu in the role payload
- the SecuredIPdu is referenced by a PduTriggering and that PduTriggering refers to an IPduPort that has the attribute rxSecurityVerification set to true.

⌋

[TPS_MANI_03637] Ignored attributes of ServiceInterfaceElementSecureComConfig in the context of signal/service translation ⌈If a ServiceInterfaceElementSecureComConfig qualifies according to [TPS_MANI_03636] then any attributes (e.g. dataId and freshnessValueId) - except the reference in the role event or fieldNotifier - shall be ignored. The information how to configure the SecOC shall be taken from the SecuredIPdu.⌋

[TPS_MANI_03663] Ignored references of ServiceInstanceToMachineMapping in the context of signal/service translation ⌈If an AdaptivePlatformServiceInstance qualifies according to [TPS_MANI_03636] and a ServiceInstanceToMachineMapping exists that refers to that AdaptivePlatformServiceInstance in the role serviceInstance and in the context of that ServiceInstanceToMachineMapping any of the following references exist:

- ServiceInstanceToMachineMapping.secureComPropsForUdp
- ServiceInstanceToMachineMapping.secureComPropsForTcp
- ServiceInstanceToMachineMapping.secOcComPropsForMulticast,

they shall be ignored.⌋

The information of SecOcSecureComProps is available from the SecuredIPdu description in:

- SecureCommunicationAuthenticationProps.authInfoTxLength
- SecureCommunicationFreshnessProps.freshnessValueLength

- SecureCommunicationFreshnessProps.freshnessValueTxLength
- SecOcCryptoServiceMapping.authentication.

10 Execution Manifest

10.1 Overview

The purpose of the execution manifest is to provide information that is needed for the actual target-configuration of an application (formally modeled as an SwComponentType) onto the AUTOSAR adaptive platform.

One aspect of the target-configuration 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 target-configuration scenario in order to increase the odds that the application software can be reused in different target-configuration 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 Chapter 13.

10.2 Process

The top-level element of the Execution Manifest definition is the Process, in reference to the fact that the unit of target-configuration on the *AUTOSAR adaptive platform* is a binary that, at runtime, makes a POSIX process.

[TPS_MANI_01308] Process is not designed for re-usability ⌈Meta-class Process has **not** been created with the goal of reusing it on different Machines.⌋

Class	**Process**			
Note	This meta-class provides information required to execute the referenced Executable. **Tags:** atp.recommendedPackage=Processes This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractExecutionContext*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *Uploadable PackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**Process**			
design	ProcessDesign	0..1	ref	This reference represents the identification of the design-time representation for the Process that owns the reference. **Tags:** atp.Splitkey=design
executable	Executable	*	ref	Reference to executable that is executed in the process. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=executable
functionCluster Affiliation	String	0..1	attr	This attribute specifies which functional cluster the Process is affiliated with.
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
posixUser	PosixUser	0..1	ref	This reference identifies the POSIX user ID under which the Process shall be launched. **Tags:** atp.Splitkey=posixUser
preMapping	Boolean	0..1	attr	This attribute describes whether the executable is preloaded into the memory.
primaryGroup	PosixGroup	0..1	ref	This reference identifies the collection of primary group IDs under which the Process shall be launched. **Tags:** atp.Splitkey=primaryGroup
processState Machine	ModeDeclarationGroup Prototype	0..1	aggr	Set of Process States that are defined for the process. This attribute is used to support the modeling of execution dependencies that utilize the condition of process state. Please note that the process states may not be modeled arbitrarily at any stage of the AUTOSAR workflow because the supported states are standardized in the context of the SWS Execution Management [13]. **Tags:** atp.Splitkey=processStateMachine, processState Machine.shortName
stateDependent StartupConfig	StateDependentStartup Config	*	aggr	Applicable startup configurations. **Tags:** atp.Splitkey=stateDependentStartupConfig
supplementary Group	PosixGroup	*	ref	This reference identifies the collection of supplementary group IDs under which the Process shall be launched. **Tags:** atp.Splitkey=supplementaryGroup

Table 10.1: Process

[constr_3732] Upper multiplicity of reference in the role `Process.executable`

Imposition time: `IT_Mani`

⌈In the context of `Process`, the reference in the role `executable` shall exist at most once.⌋

Class	***AbstractExecutionContext*** (abstract)
Note	This meta-class acts as a base class for entities that execute code on different levels, e.g. container, process, thread, fiber. This Class is only used by the AUTOSAR Adaptive Platform.

▽

△

Class	**AbstractExecutionContext** (abstract)			
Base	ARElement, ARObject, AtpClassifier, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadableDeploymentElement, UploadablePackageElement			
Subclasses	Process			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 10.2: AbstractExecutionContext

[TPS_MANI_01337] Standardized values for attribute `Process.functionClusterAffiliation` ⌈The following values of attribute `Process.functionClusterAffiliation` are standardized by AUTOSAR:

- **`STATE_MANAGEMENT`**
- **`PLATFORM_HEALTH_MANAGEMENT`**

⌋

Values other than those in the standardized set can also be used for attribute `Process.functionClusterAffiliation`.

However, proprietary values of this attribute should be 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 the [13, SWS Execution Management], specifically in the context of [SWS_EM_02109].

10.2.1 Process relates to Executable

[TPS_MANI_01011] Connection between application design and application target-configuration ⌈The connection between the *application design* and the *application target-configuration* 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 target-configuration level and, at the same time, bind the target-configuration to a specific design.⌋

The meta-model, as depicted in Figure 10.1, supports the existence of two or more `Process`es that reference the same `Executable`.

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 10.2.

The startup conditions and startup parameters of different Processes are likely to differ, in order to achieve a variation of the functionality of the Executable.

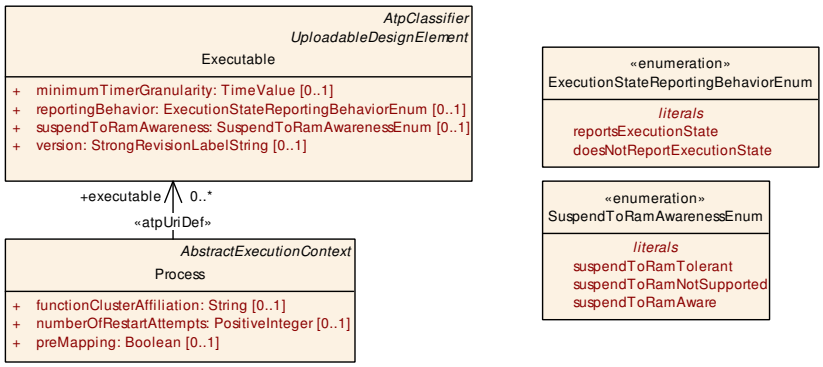


Figure 10.1: Relation of meta-classes Executable and Process

Therefore, it is necessary to allow for the definition of startup configurations on a per-Process-basis. This aspect is described in Chapter 10.3.

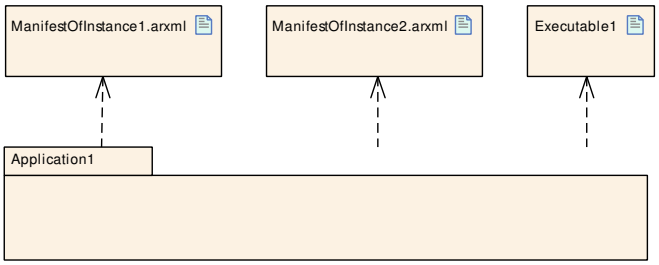


Figure 10.2: Example target-configuration where one Executable is bundled with two ARXML files that each contain the description of one Process

10.2.2 Process States

A Process is a state-full entity. The state machine associated with a Process is modeled based on the aggregation of a ModeDeclarationGroupPrototype in the role processStateMachine, as sketched by Figure 10.3.

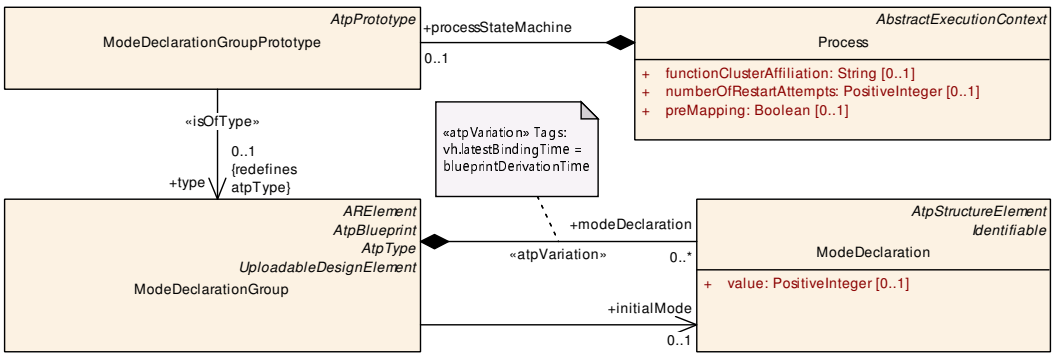


Figure 10.3: Modeling of process states

Process states may not be modeled arbitrarily at any stage of the AUTOSAR workflow, because the supported states are standardized in the context of the [13, SWS Execution Management].

It is possible to define a so-called *execution dependency*, such that one `Process` can only launch if a referenced other `Process` is in a given process state. The details are explained in Chapter 10.3.6.

10.3 Startup Configuration

The configuration of startup behavior is an essential part of the execution manifest.

[TPS_MANI_01012] Formal modeling of application startup behavior ⌈The formal modeling of application startup behavior is implemented by means of the aggregation of meta-class `StateDependentStartupConfig` in the role `Process.stateDependentStartupConfig`.⌋

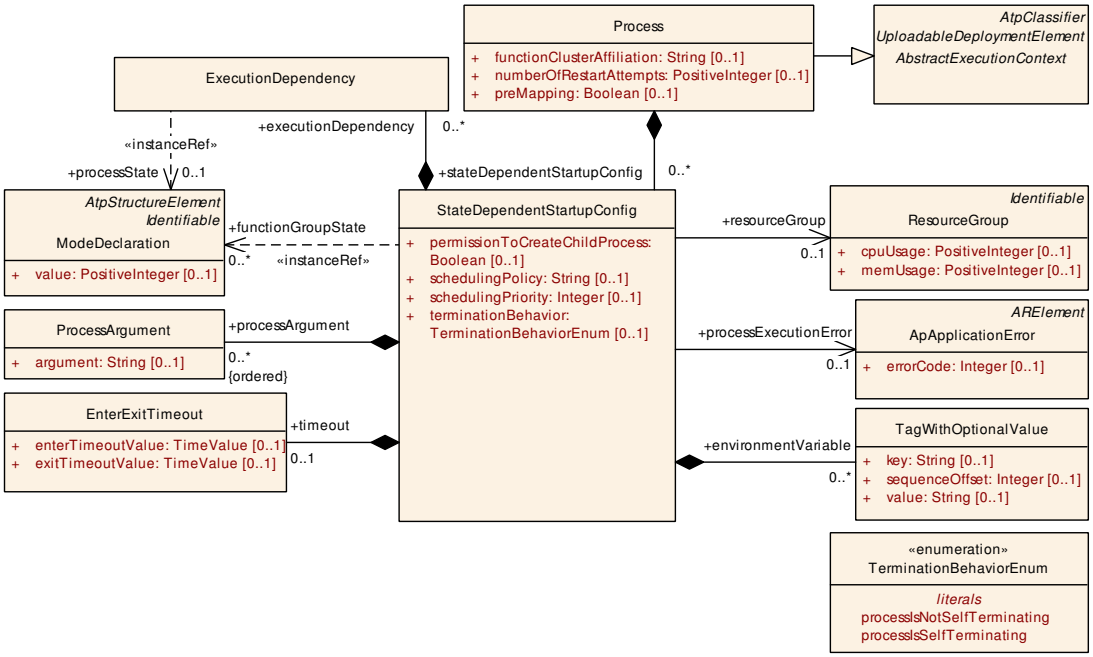


Figure 10.4: Content of a `Process`

10.3.1 State-dependent Startup Configuration

[TPS_MANI_01013] Semantics of meta-class `StateDependentStartupConfig` ⌈The purpose of meta-class `StateDependentStartupConfig` is to qualify the startup configuration for specific `ModeDeclaration`s.

In other words, the intention is to express that the `StateDependentStartupConfig` is applicable if the state machines that control the startup are in the states

represented by the ModeDeclaration referenced in the role StateDependentStartupConfig.functionGroupState.⌋

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 are deployed.

[constr_3423] StateDependentStartupConfig of a Process shall reference a functionGroupState

Imposition time: IT_Mani

⌈Each StateDependentStartupConfig of a Process shall reference at least one ModeDeclaration in the role functionGroupState.⌋

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] StateDependentStartupConfig shall only refer to Function Group States of the same Function Group

Imposition time: IT_Mani

⌈For 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).⌋

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] Number of Process.stateDependentStartupConfig that refer to the same functionGroupState

Imposition time: IT_Mani

⌈Within the context of a given Process, no two StateDependentStartupConfigs shall refer to the same ModeDeclaration in the role functionGroupState.⌋

[TPS_MANI_01046] Semantics of StateDependentStartupConfig.functionGroupState ⌈The 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.⌋

[constr_3424] `StateDependentStartupConfig` shall never reference the `functionGroupState` `Off`

Imposition time: `IT_Mani`

⌈A `StateDependentStartupConfig` shall never reference the `ModeDeclaration` that has the `shortName` `Off` in the role `functionGroupState`. The `Off` `ModeDeclaration` is a special state in a `Function Group` as defined by [TPS_MANI_03195].⌋

[TPS_MANI_01209] Definition of environment variables in process scope ⌈It is possible to define environment variables in the scope of any given `Process`.

For this purpose the aggregation of `TagWithOptionalValue` in the role `StateDependentStartupConfig.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 includes the ability to define environment variables with empty values. For this purpose, the attribute `TagWithOptionalValue.value` shall be omitted.⌋

Class	**StateDependentStartupConfig**			
Note	This meta-class defines the startup configuration for the process depending on a collection of machine states. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	Process.stateDependentStartupConfig			
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.Splitkey=environmentVariable
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.Splitkey=executionDependency
functionGroup State	ModeDeclaration	*	iref	This represent the applicable functionGroupMode. **Tags:** atp.Splitkey=functionGroupState, functionGroup State.contextModeDeclarationGroupPrototype, function GroupState.targetModeDeclaration **InstanceRef implemented by:** FunctionGroupStateIn FunctionGroupSetInstanceRef
permissionTo CreateChild Process	Boolean	0..1	attr	This attribute defines if Process is permitted to create child Processes. When setting this parameter to true two things should be kept in mind: 1) safety and security implication of this configuration, 2) the fact that Process will assume management responsibilities for child Processes (i.e. it will be responsible for terminating Processes that it creates).

▽

△

Class	**StateDependentStartupConfig**			
process Argument (ordered)	ProcessArgument	*	aggr	This aggregation represents the collection of command-line arguments applicable to the enclosing StateDependentStartupConfig.
process ExecutionError	ApApplicationError	0..1	ref	This reference is used to identify the applicable execution error. **Tags:** atp.Splitkey=processExecutionError
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.Splitkey=resourceConsumption, resource Consumption.shortName
resourceGroup	ResourceGroup	0..1	ref	Reference to an applicable resource group. **Tags:** atp.Splitkey=resourceGroup
scheduling Policy	String	0..1	attr	This attribute represents the ability to define the scheduling policy for the initial thread of the application.
scheduling Priority	Integer	0..1	attr	This is the scheduling priority requested by the application itself.
startupConfig	StartupConfig	0..1	ref	Reference to a reusable startup configuration with startup parameters. **Tags:** atp.Splitkey=startupConfig atp.Status=obsolete
supplementary Group	PosixGroup	*	ref	This reference identifies the collection of supplementary group IDs. **Tags:** atp.Splitkey=supplementaryGroup
termination Behavior	TerminationBehavior Enum	0..1	attr	This attribute defines the termination behavior of the Process.
timeout	EnterExitTimeout	0..1	aggr	This aggregation can be used to specify the timeouts for launching and terminating the process depending on the StateDependentStartupConfig. **Tags:** atp.Splitkey=timeout

Table 10.3: StateDependentStartupConfig

[constr_10175] Multiplicity of attribute StateDependentStartupConfig.resourceGroup

Imposition time: IT_Mani

⌈For each StateDependentStartupConfig, the attribute resourceGroup shall exist.⌋

[constr_1618] Ability to shut down

Imposition time: IT_Mani

⌈In the context of one Machine, at least one Process shall have a stateDependentStartupConfig.functionGroupState that has the shortName Shutdown.⌋

[constr_1619] Ability to restart

Imposition time: IT_Mani

⌈In the context of one Machine, at least one Process shall have a stateDependentStartupConfig.functionGroupState that has the shortName Restart.⌋

Class	**StartupConfig**			
Note	This meta-class represents a reusable startup configuration for processes.. **Tags:** atp.Status=obsolete atp.recommendedPackage=StartupConfigs This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
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.Splitkey=environmentVariable atp.Status=obsolete
permissionTo CreateChild Process	Boolean	0..1	attr	This attribute defines if Process is permitted to create child Processes. When setting this parameter to true two things should be kept in mind: 1) safety and security implication of this configuration, 2) the fact that Process will assume management responsibilities for child Processes (i.e. it will be responsible for terminating Processes that it creates). **Tags:** atp.Status=obsolete
process Argument (ordered)	ProcessArgument	*	aggr	This aggregation represents the collection of command-line arguments applicable to the enclosing StartupConfig. **Tags:** atp.Status=obsolete
process ExecutionError	ApApplicationError	0..1	ref	this reference is used to identify the applicable execution error. **Tags:** atp.Splitkey=processExecutionError atp.Status=obsolete
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=obsolete
scheduling Priority	Integer	0..1	attr	This is the scheduling priority requested by the application itself. **Tags:** atp.Status=obsolete
termination Behavior	TerminationBehavior Enum	0..1	attr	This attribute defines the termination behavior of the Process. **Tags:** atp.Status=obsolete

▽

△

Class	StartupConfig			
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.Splitkey=timeout atp.Status=obsolete

Table 10.4: StartupConfig

[TPS_MANI_01277] Definition of a start-up timeout for a `StateDependentStartupConfig` of a `Process` ⌈Meta-class `StateDependentStartupConfig` provides the ability to define a start-up timeout for a `Process` by means of the attribute `enterTimeoutValue` that is aggregated by meta-class `EnterExitTimeout` that is aggregated by the `StateDependentStartupConfig` in the role `timeout`.⌋

[TPS_MANI_01278] Definition of a termination timeout for a `StateDependentStartupConfig` of a `Process` ⌈Meta-class `StateDependentStartupConfig` 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 `StateDependentStartupConfig` in the role `timeout`.⌋

10.3.2 Process Attributes

Several meta-classes are defined to allow for the specification of so-called process attributes.

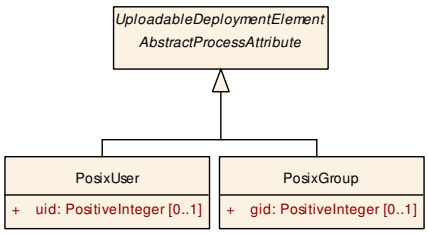


Figure 10.5: Modeling of abstract base class for `Process` attributes

Class	***AbstractProcessAttribute*** (abstract)			
Note	This meta-class acts as an abstract base class for the modeling of process attributes. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	PosixGroup, PosixUser			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 10.5: AbstractProcessAttribute

[TPS_MANI_01518] Definition of launch attributes for `Process` in the context of `Process` ⌈It is possible to configure a number of attributes applicable to the launched `Process` at run-time:

- User ID: this value can be configured in `Process.posixUser.uid`
- Group ID: this value can be configured in `Process.primaryGroup.gid`
- Supplementary Group ID: this value can be configured in `Process.supplementaryGroup.gid`

⌋

[TPS_MANI_01519] Definition of launch attributes for `Process` in the context of `StateDependentStartupConfig` ⌈It is possible to configure an array of Supplementary Group ID for a to-be-launched `Process`. This value can be configured in `StateDependentStartupConfig.supplementaryGroup.gid`⌋

[constr_10747] Simultaneous existence of `Process.posixUser` and `Process.primaryGroup`

Imposition time: `IT_Mani`

⌈If one of

- `Process.posixUser`
- `Process.primaryGroup`

exists, then the respective other reference shall also exist.⌋

The definition of the process user and the process group always go together. If one exists, then the other one shall also exist.

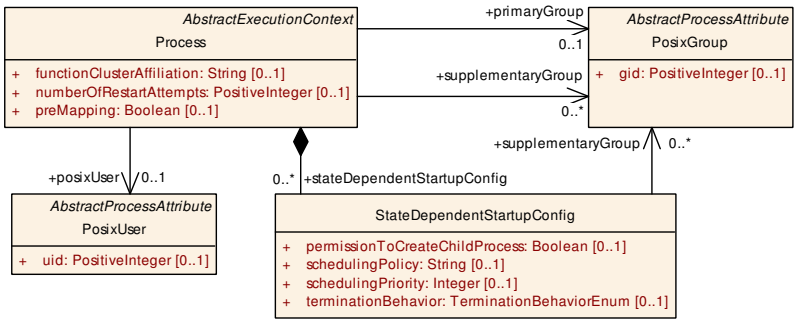


Figure 10.6: Modeling of `Process` attributes

Class	**PosixUser**
Note	This meta-class represents a process user used in the context of creating the launch configuration of a process. **Tags:** atp.recommendedPackage=PosixUsers This Class is only used by the AUTOSAR Adaptive Platform.

▽

△

Class	PosixUser			
Base	ARElement, ARObject, AbstractProcessAttribute, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadableDeploymentElement, UploadablePackageElement			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
uid	PositiveInteger	0..1	attr	This attribute represents the user ID under which the process shall be launched.

Table 10.6: PosixUser

[constr_10748] Existence of PosixUser.uid

Imposition time: IT_Mani

⌈For each PosixUser, attribute uid shall exist.⌋

Class	PosixGroup			
Note	This meta-class represents a process group used in the context of creating the launch configuration of a process. **Tags:** atp.recommendedPackage=PosixGroups This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, AbstractProcessAttribute, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadableDeploymentElement, UploadablePackageElement			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
gid	PositiveInteger	0..1	attr	This attribute represents the value of the ID for the POSIX group or supplementary process group.

Table 10.7: PosixGroup

[constr_10749] Existence of PosixGroup.gid

Imposition time: IT_Mani

⌈For each PosixGroup, attribute gid shall exist.⌋

10.3.3 Scheduling

[TPS_MANI_01061] Requirements on scheduling ⌈The attributes StateDependentStartupConfig.schedulingPolicy and StateDependentStartupConfig.schedulingPriority make requirements on the scheduling of the main thread of a process that is created out of launching the corresponding Executable.⌋

[TPS_MANI_01328] Standardized values for attribute StateDependentStartupConfig.schedulingPolicy ⌈The following values are standardized for attribute StateDependentStartupConfig.schedulingPolicy:

- **SCHED_RR**

- **SCHED_FIFO**
- **SCHED_OTHER**

⌋

It is possible to use a custom, non-standardized value for the attribute StateDependentStartupConfig.schedulingPolicy but this option comes with the obligation to use a value that is guaranteed not to clash with possible future extensions of the collection of standardized values.

[TPS_MANI_01188] Semantics of attribute schedulingPriority ⌈The value of attribute StateDependentStartupConfig.schedulingPriority shall be interpreted such that the higher values represent a higher scheduling priority.⌋

[constr_1692] Value of schedulingPriority

Imposition time: IT_Mani

⌈The value of attribute StateDependentStartupConfig.schedulingPriority shall be set to a positive integer value.⌋

10.3.4 Process Arguments

More information about the interpretation of ProcessArgument is provided in the [13, SWS Execution Management].

[constr_1769] Existence of ProcessArgument.argument

Imposition time: IT_Mani

⌈For each ProcessArgument, attribute argument shall exist.⌋

Class	**ProcessArgument**			
Note	This meta-class has the ability to define command line arguments for processing by the Main function. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	StartupConfig.processArgument, StateDependentStartupConfig.processArgument			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
argument	String	0..1	attr	This represents one command-line argument to be processed by the executable software.

Table 10.8: ProcessArgument

10.3.5 Association with Resource Group

Meta-class StateDependentStartupConfig also supports the specification of a relation to a resource group.

[TPS_MANI_01017] Relation of startup configuration to resource group ⌈The 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 StateDependentStartup-Config to ResourceGroup in the role resourceGroup.⌋

[constr_3413] StateDependentStartupConfig of a Process is mapped to exactly one ResourceGroup

Imposition time: IT_Mani

⌈Each StateDependentStartupConfig of a Process shall be assigned to exactly one ResourceGroup that is defined in the Machine Manifest.⌋

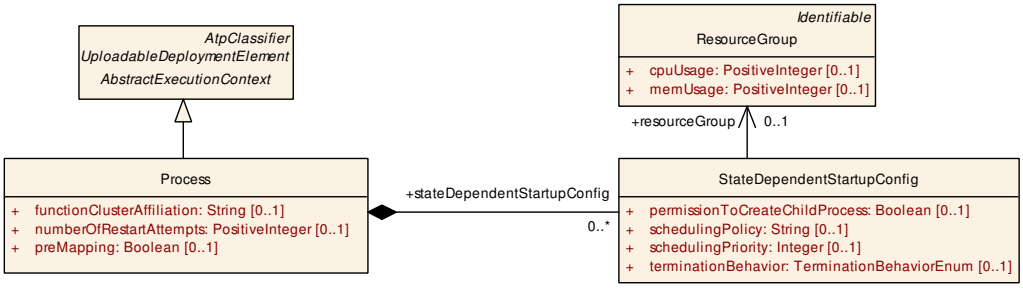


Figure 10.7: Modeling of how Process relates to ResourceGroup

10.3.6 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] Startup configuration supports the definition of a launch sequence dependency ⌈The modeling of startup configuration also supports the definition of a launch sequence dependency, formalized by the meta-class Execution-Dependency that is aggregated by StateDependentStartupConfig in the role executionDependency.

The ExecutionDependency allows the definition of 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.⌋

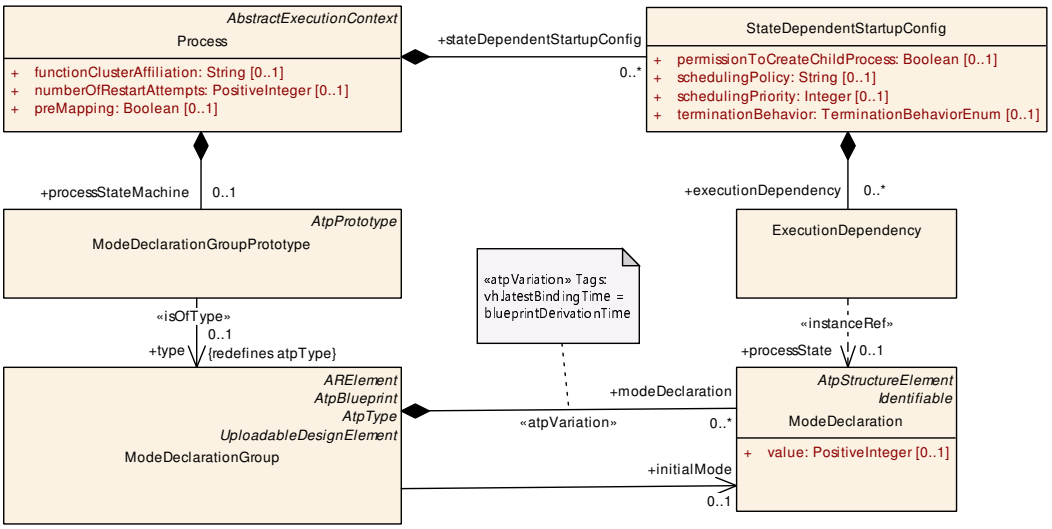


Figure 10.8: Modeling of how `Process` relates to `ModeDeclaration` owned by another `Process`

Figure 10.9 provides an exemplary explanation of [constr_1689]. In this example, `Process` "B" (the referencing `Process` as of [constr_1689]) defines an `execution-Dependency` 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` "A", and if "A" refers to `Function Group State` "MD" within `Function Group` "FG" and if "B" refers to `Function Group State` "MD" within "FG" then the constraint [constr_1689] is fulfilled.

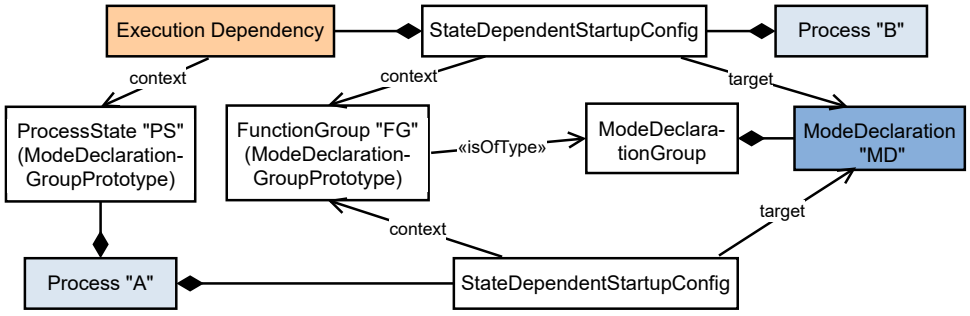


Figure 10.9: Explanation of dependencies from one `Process` to another

[constr_1689] Modeling of a startup dependency between different `Process`es

Imposition time: `IT_Mani`

⌈The existence of the aggregation in the role `Process.stateDependentStartup-Config.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.⌋

There are scenarios where potential references to more than one functionGroupState in the presence of an ExecutionDependency lead to a conflict at runtime (specifically if the Process referenced in the ExecutionDependency is self-terminating). The prevention of such a scenario is the reason for the existence of [constr_10411].

[constr_10411] Existence of ExecutionDependency and references to Function Group States

Imposition time: IT_Mani

⌈Each StateDependentStartupConfig that aggregates at least one ExecutionDependency in the role executionDependency shall reference at most one ModeDeclaration in the role functionGroupState.⌋

For the exploration of such a scenario, Figure 10.10 sketches a scenario that is not allowed in the presence of [constr_10411].

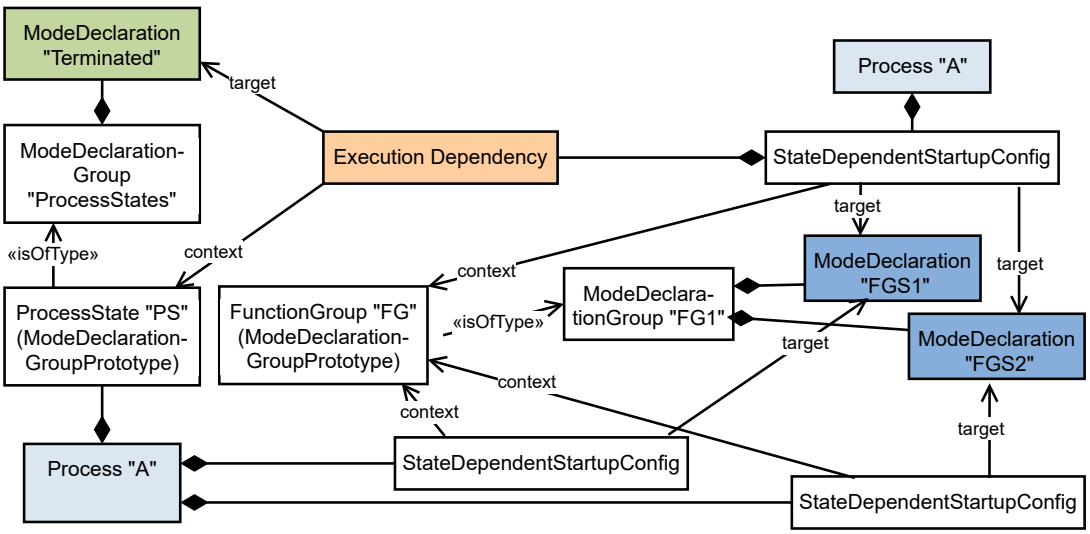


Figure 10.10: StateDependentStartupConfig references more than one Function Group State (invalid configuration)

In the sketched scenario in Figure 10.10, the execution of Process "B" depends on the launch and subsequent termination of Process "A".

Process “A” owns two StateDependentStartupConfigs, one that references the Function Group State “FGS1” and another that references Function Group State “FGS2”.

At runtime, this means that in the event of switching the Function Group to Function Group State “FGS1” (or “FGS2”) both Processes “A” and “B” want to be executed.

Thanks to the existence of the ExecutionDependency, a POSIX-process that represents Process “A” is launched first, then a POSIX-process that represents Process “B”.

Now, if the Function Group State is switched from “FGS1” to “FGS2” (or vice versa), it is (with respect to the modeling of the example) expected that the same sequence of execution will be repeated. But instead, the following sequence will occur:

- a new POSIX-process that represents Process “A” will be launched and
- the POSIX-process that represents Process “B” will **not** be terminated and relaunched because Process “B” aggregates a StateDependentStartupConfig that references both Function Group States “FGS1” and “FGS2”.

As a result, the order of execution (“A” launches and terminates, then “B” launches) as defined by the ExecutionDependency cannot be implemented in the event of a switch of Function Group States between the two Function Group States referenced by the StateDependentStartupConfig aggregated by Process “B”.

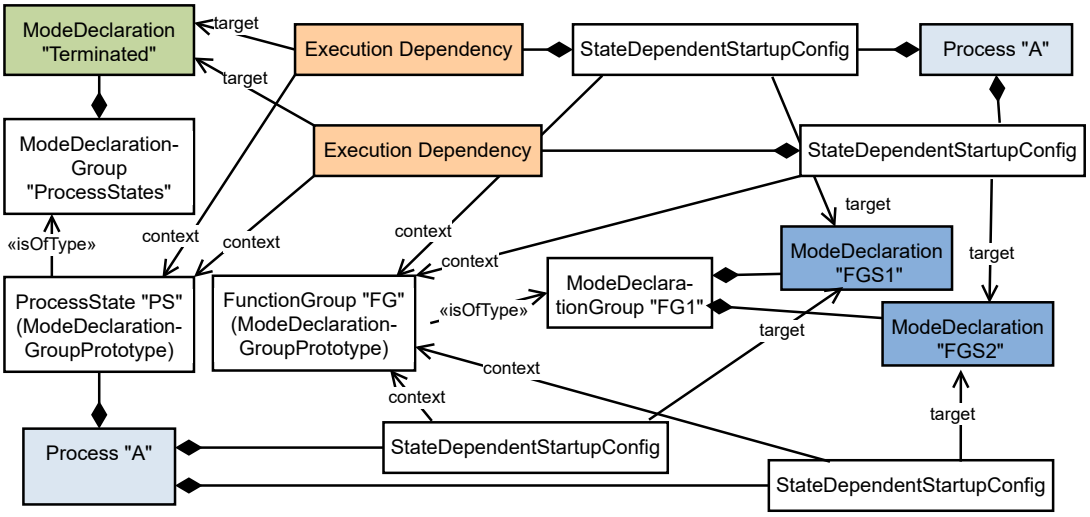


Figure 10.11: Two StateDependentStartupConfigs that each reference just one Function Group State

In order to prevent this contradiction between modeled semantics and real-world behavior, [constr_10411] has been created.

On the other hand, the scenario depicted in Figure 10.11 is permitted according to [constr_10411].

In the example depicted in Figure 10.11 it is expected that, upon Function Group State "FGS1" becoming active, a POSIX-process representing Process "A" is launched and, at some time later, terminates.

Then, a POSIX-process representing Process "B" is executed. The same sequence would occur when the Function Group State is switched from any other Function Group State except "FGS1" to "FGS2".

When the Function Group is switched to "FGS2", the POSIX-process that represents Process "B" terminates as the exit from "FGS1" happens.

As "FGS2" is entered, a POSIX-process representing Process "A" is launched. After it terminates, a POSIX-process that represents Process "B" starts being executed, which is exactly the expected sequence.

To summarize the example behavior, the intended execution dependency is realized when

- one of the referenced Function Group States becomes active or
- a switch from one referenced Function Group State to the other referenced Function Group State occurs.

[constr_1606] Processes with mutual ExecutionDependencys

Imposition time: IT_Mani

⌈A 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.⌋

10.3.7 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 Chapter 10.4 in more detail. The usage of Function Groups is described in more detail in [13, SWS Execution Management].

[TPS_MANI_03152] Assignment of a StateDependentStartupConfig to a Function Group State ⌈The StateDependentStartupConfig is assigned to a Function Group State with the functionGroupState reference.⌋

10.3.8 Resource Consumption Boundaries

[TPS_MANI_01269] Specification of boundaries for resource consumption ⌈It is possible to specify boundaries for resource consumption, specifically in terms of memory consumption for memory of a given startup configuration of a `Process`: the formalization of process memory usage is represented by meta-class `MemoryUsage`, aggregated via meta-class `ResourceConsumption` at `StateDependentStartupConfig`.

The actual value of the memory usage is computed out of the sum of all aggregated `ResourceConsumption.memoryUsage`.⌋

The ability to define resource consumption boundaries for a single `Process` differs from the ability to associate a `Process` with a `ResourceGroup`, which can also define resource consumption boundaries, albeit on a more coarse-grained level.

In contrast, the `StateDependentStartupConfig.resourceConsumption` allows for a fine-grained definition that can even observe the differences in resource consumption with respect to different startup configurations.

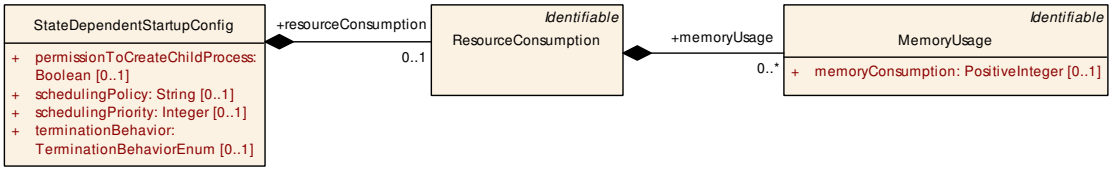


Figure 10.12: Modeling of resource consumption boundaries for a given `Process`

Class	**ResourceConsumption**			
Note	Description of consumed resources by one implementation of a software.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcuResourceEstimation.bswResourceEstimation, EcuResourceEstimation.rteResourceEstimation, *Implementation*.resourceConsumption, StateDependentStartupConfig.resourceConsumption			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
memoryUsage	MemoryUsage	*	aggr	Collection of the memory allocated by the owner. **Tags:** atp.Splitkey=memoryUsage, memoryUsage.shortName This Attribute is only used by the AUTOSAR Adaptive Platform.

Table 10.9: ResourceConsumption

Class	**MemoryUsage**			
Note	This meta-class is used to describe the memory consumption. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ResourceConsumption.memoryUsage			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
memory Consumption	PositiveInteger	0..1	attr	Provides a formal worst case system usage. The unit is byte.

Table 10.10: MemoryUsage

10.3.9 Error and Termination Behavior

[TPS_MANI_01334] Semantics of attribute `StateDependentStartupConfig.terminationBehavior` ⌈The attribute `StateDependentStartupConfig.terminationBehavior` defines the termination behavior of the `Process` in terms of whether (or not) the `Process` that aggregates the `StateDependentStartupConfig` in the role `stateDependentStartupConfig` is configured to self-terminate.⌋

Enumeration	**TerminationBehaviorEnum**
Note	This enumeration provides options for controlling of how a Process terminates. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	StartupConfig.terminationBehavior, StateDependentStartupConfig.terminationBehavior
Literal	***Description***
processIsNotSelf Terminating	The Process terminates only on request from Execution Management. **Tags:** atp.EnumerationLiteralIndex=0
processIsSelf Terminating	The Process is allowed to terminate without request from Execution Management. **Tags:** atp.EnumerationLiteralIndex=1

Table 10.11: TerminationBehaviorEnum

10.4 Function Groups

10.4.1 Semantics of Function Group

`Function Group`s with `Function Group State`s individually control groups of Application processes. The `Process` state may depend on a mode that is defined in the `Function Group` if the `StateDependentStartupConfig` refers to the `Function Group State` with the `functionGroupState` reference.

The usage of `Function Group`s is described in more detail in [13, SWS Execution Management].

[TPS_MANI_03145] Description of a `Function Group` ⌈By 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 `ModeDeclaration`s and a designated `initialMode`.⌋

[TPS_MANI_03194] `Function Group State` ⌈A `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`.⌋

The modeling described in [TPS_MANI_03145] and [TPS_MANI_03194] is depicted in Figure 10.13.

[constr_1786] Restriction to use `functionGroup` in terms of `SoftwareCluster`

Imposition time: `IT_Mani`

⌈Each `functionGroup` shall only be referenced in the role `claimedFunctionGroup` by **at most one** `SoftwareCluster`.⌋

[TPS_MANI_03195] Off state in `Function Group` ⌈Each `functionGroup` shall define a `ModeDeclaration` with the `shortName` `Off`. This `ModeDeclaration` shall also be referenced in the role `initialMode` by the `ModeDeclarationGroup` that types the respective `functionGroup`.⌋

Each `FunctionGroupSet.functionGroup` modeled as a `ModeDeclarationGroupPrototype` that is typed by a `ModeDeclarationGroup` shall have a defined `ModeDeclarationGroup.initialMode` according to [constr_1973] [2, TPS Software Component Template] that is also valid for this specification.

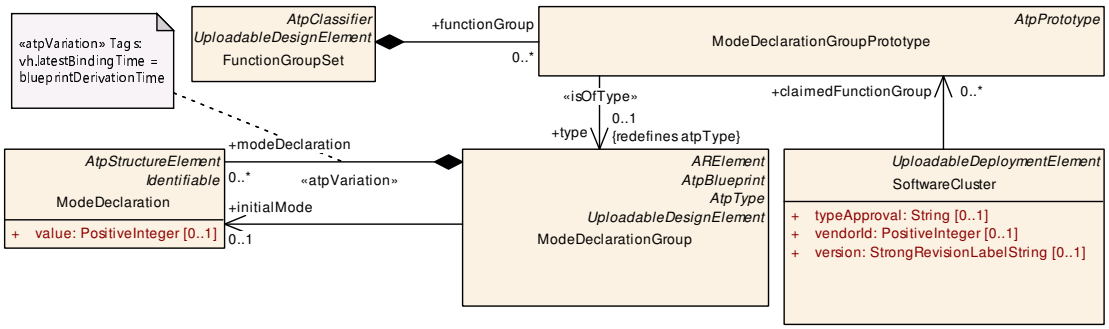


Figure 10.13: Configuration of `Function Group`s

[constr_1787] Restricted use of `Function Group`s in the context of a `SoftwareCluster`

Imposition time: `IT_Mani`

⌈All `Process`es referenced by a `SoftwareCluster` in the role `containedProcess` shall only aggregate `StateDependentStartupConfig`s where the reference `functionGroupState` refers to a `ModeDeclarationGroupPrototype` (as context) that is also referenced by the same `SoftwareCluster` in the role `claimedFunctionGroup`.⌋

The description of `SoftwareCluster` can be found in Chapter 16.

[constr_10023] Mandatory content of any `functionGroup`

Imposition time: `IT_Mani`

⌈All `ModeDeclarationGroupPrototype`s aggregated by a `FunctionGroupSet` in the role `functionGroup` shall refer to a `ModeDeclarationGroup` that contains one `ModeDeclaration` with the `shortName` **`Verify`**.⌋

Class	FunctionGroupSet			
Note	This meta-class provides the ability to create arbitrary collections of function groups. **Tags:** atp.recommendedPackage=FunctionGroupSets This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
functionGroup	ModeDeclarationGroup Prototype	*	aggr	This aggregation represents the collection of function groups. **Tags:** atp.Splitkey=functionGroup, functionGroup.short Name

Table 10.12: FunctionGroupSet

Class	ModeDeclarationGroupPrototype			
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*			
Aggregated by	*AtpClassifier*.atpFeature, BswModuleDescription.providedModeGroup, BswModuleDescription.requiredModeGroup, FirewallStateSwitchInterface.firewallStateMachine, FunctionGroupSet.functionGroup, ModeSwitchInterface.modeGroup, Process.processStateMachine, StateManagementStateNotification.stateMachine			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
type	ModeDeclarationGroup	0..1	tref	The "collection of ModeDeclarations" (= ModeDeclarationGroup) supported by a component **Stereotypes:** isOfType **Tags:** atp.Splitkey=type

Table 10.13: ModeDeclarationGroupPrototype

Class	ModeDeclarationGroup			
Note	A collection of Mode Declarations. Also, the initial mode is explicitly identified. **Tags:** atp.recommendedPackage=ModeDeclarationGroups			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
initialMode	ModeDeclaration	0..1	ref	The initial mode of the ModeDeclarationGroup. This mode is active before any mode switches occurred. **Tags:** atp.Splitkey=initialMode
mode Declaration	ModeDeclaration	*	aggr	The ModeDeclarations collected in this ModeDeclarationGroup. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=modeDeclaration, modeDeclaration.shortName, modeDeclaration.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime

Table 10.14: ModeDeclarationGroup

Class	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*			
Aggregated by	*AtpClassifier*.atpFeature, ModeDeclarationGroup.modeDeclaration			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
value	PositiveInteger	0..1	attr	The RTE shall take the value of this attribute for generating the source code representation of this Mode Declaration.

Table 10.15: ModeDeclaration

10.5 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 (see Figure 12.58) as well as for functional clusters (see Figure 15.12) with the exception of the Execution Manager (because the Execution Manager is itself not modeled as a `Process`).

More information about the semantics and usage of security events is provided in the [34, TPS Security Extract Template].

Class	SecurityEventDefinition			
Note	This meta-class defines a security-related event as part of the intrusion detection system. **Tags:** atp.recommendedPackage=SecurityEventDefinitions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *IdsCommonElement*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
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. **Tags:** atp.Splitkey=eventSymbolName, eventSymbol Name.shortName
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.
securityEvent ContextData Definition	SecurityEventContext DataDefinition	*	ref	Definition of additional context data that is reported with the security event in order to better support the analysis of a possible security threat. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=securityEventContextDataDefinition, security EventContextDataDefinition.securityEventContextData Definition, securityEventContextDataDefinition.variation Point.shortLabel vh.latestBindingTime=systemDesignTime

Table 10.16: SecurityEventDefinition

11 Machine Manifest

11.1 Machine

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 7.2) 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 6.1).

[TPS_MANI_03035] Content of the Machine configuration ⌈The purpose of the Machine is to provide machine-specific configuration settings.⌋

An overview of the Machine meta-class is sketched in Figure 11.1.

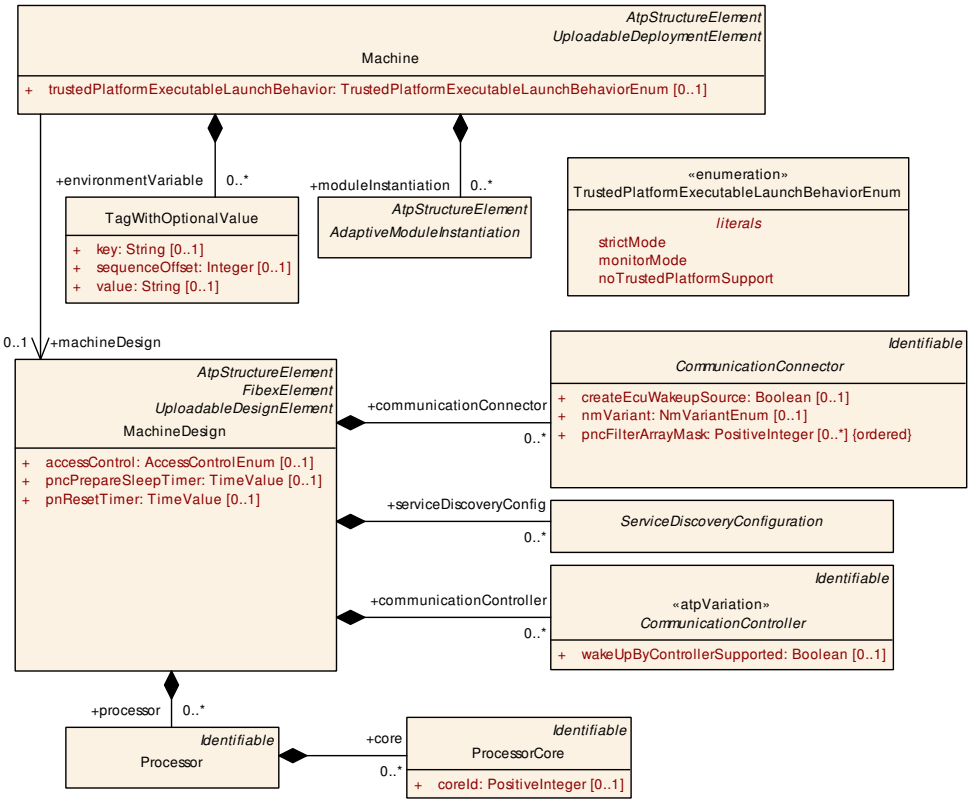


Figure 11.1: Overview about the content of the Machine configuration

[TPS_MANI_01273] Support for trusted Platform ⌈If 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.trustedPlatformExecutable-LaunchBehavior.

Class	Machine			
Note	Machine that represents an Adaptive Autosar Software Stack. **Tags:** atp.recommendedPackage=Machines This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element, *AtpClassifier*.atpFeature			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
default Application Timeout	EnterExitTimeout	0..1	aggr	This aggregation defines a default timeout in the context of a given Machine with respect to the launching and termination of applications. **Tags:** atp.Splitkey=defaultApplicationTimeout
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. **Tags:** atp.Splitkey=environmentVariable
machineDesign	MachineDesign	0..1	ref	Reference to the MachineDesign this Machine is implementing. **Tags:** atp.Splitkey=machineDesign
module Instantiation	AdaptiveModule Instantiation	*	aggr	Configuration of Adaptive Autosar module instances that are running on the machine. **Tags:** atp.Splitkey=moduleInstantiation, module Instantiation.shortName
processor	Processor	*	aggr	This represents the collection of processors owned by the enclosing machine. **Tags:** atp.Splitkey=processor, processor.shortName
secure Communication Deployment	SecureCommunication Deployment	*	aggr	Target-configuration of secure communication protocol configuration settings to crypto module entities. **Tags:** atp.Splitkey=secureCommunicationDeployment, secure CommunicationDeployment.shortName atp.Status=obsolete
trustedPlatform Executable LaunchBehavior	TrustedPlatform ExecutableLaunch BehaviorEnum	0..1	attr	This attribute controls the behavior of how authentication affects the ability to launch for each Executable.

Table 11.1: Machine

Class	***SecureCommunicationDeployment*** (abstract)			
Note	The meta-class represents the ability to define a target-configuration of secure communication protocol configuration settings to crypto module entities. **Tags:** atp.Status=obsolete This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	SecOcDeployment, TlsDeployment			
Aggregated by	Machine.secureCommunicationDeployment			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 11.2: SecureCommunicationDeployment

[constr_10169] Multiplicity of reference in the role Machine.machineDesign

Imposition time: IT_Mani

⌈For each Machine, the reference in the role machineDesign shall exist.⌋

[constr_10170] Existence of attribute Machine.trustedPlatformExecutable-LaunchBehavior

Imposition time: IT_Mani

⌈For each Machine, the primitive attribute trustedPlatformExecutableLaunch-Behavior shall exist.⌋

[constr_10171] Multiplicity of attribute Machine.processor

Imposition time: IT_Mani

⌈For each Machine, at least one aggregation on the role processor shall exist.⌋

Enumeration	**TrustedPlatformExecutableLaunchBehaviorEnum**
Note	This enumeration provides options for controlling the behavior of how authentication affects the ability to launch an Executable. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	Machine.trustedPlatformExecutableLaunchBehavior
Literal	***Description***
monitorMode	An Executable shall always launch, even if the corresponding authentication fails **Tags:** atp.EnumerationLiteralIndex=1
noTrustedPlatform Support	This value shall be used if there is no TrustedPlatform support on the Machine **Tags:** atp.EnumerationLiteralIndex=2
strictMode	An Executable shall not launch if the corresponding authentication fails. **Tags:** atp.EnumerationLiteralIndex=0

Table 11.3: TrustedPlatformExecutableLaunchBehaviorEnum

11.2 Environment Variable

[TPS_MANI_01208] Definition of environment variables in the scope of a Machine
⌈It 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 TagWith-OptionalValue.value.

This includes the ability to define environment variables with empty values. For this purpose, the attribute TagWithOptionalValue.value shall be omitted.⌋

The aggregation Machine.environmentVariable may comprise contributions to the definition of environment variables from **different sources**.

Chapter B.8.1 contains a description of a scenario for the definition of environment variables in different partial models.

11.3 Process To Machine Mapping

11.3.1 General Modeling Approach

[TPS_MANI_03147] Mapping of a `Process` to a `Machine` ⌈The meta-class `ProcessToMachineMapping` provides the ability to map a `Process` to a `Machine`.⌋

[constr_1553] Restriction for `ProcessToMachineMapping`

Imposition time: `IT_Mani`

⌈The 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 `ProcessToMachine-Mapping` in the role `process`.

⌋

[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`.

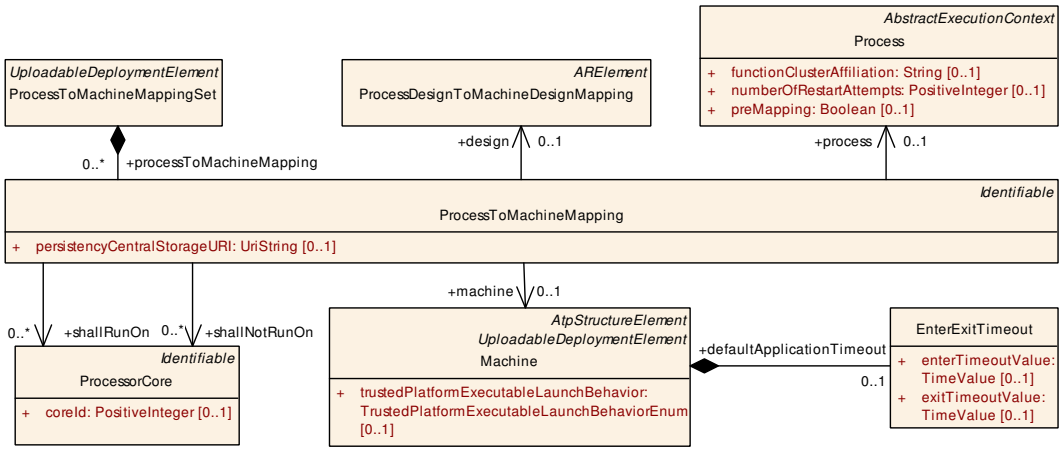


Figure 11.2: Mapping of a `Process` to a `Machine`

[constr_5004] Mapping of a `Process` to a `Machine` is mandatory in the Execution Manifest

Imposition time: `IT_Mani`

⌈Each `Process` shall be mapped by a `ProcessToMachineMapping` to one `Machine`.⌋

[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 for processing the model in many cases.

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.

[constr_10090] Existence of ProcessToMachineMapping.persistencyCentralStorageURI

Imposition time: IT_Mani

⌈Attribute ProcessToMachineMapping.persistencyCentralStorageURI shall exist if the Process referenced in the role ProcessToMachineMapping.process is also referenced by at least one of

- PersistencyPortPrototypeToDeploymentMapping in the role process
- FunctionalClusterInteractsWithPersistencyDeploymentMapping in the role process

⌋

Class	**ProcessToMachineMappingSet**			
Note	This meta-class acts as a bucket for collecting ProcessToMachineMappings. **Tags:** atp.recommendedPackage=ProcessToMachineMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
processTo Machine Mapping	ProcessToMachine Mapping	*	aggr	This represents the collection of ProcessToMachine Mappings of the enclosing ProcessToMachineMapping Set. **Tags:** atp.Splitkey=processToMachineMapping, process ToMachineMapping.shortName

Table 11.4: ProcessToMachineMappingSet

Class	**ProcessToMachineMapping**			
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. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ProcessToMachineMappingSet.processToMachineMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ProcessToMachineMapping**			
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.Splitkey=design
machine	Machine	0..1	ref	This reference identifies the Machine in the context of the ProcessToMachineMapping. **Tags:** atp.Splitkey=machine
nonOsModule Instantiation	NonOsModule Instantiation	0..1	ref	This supports the optional case that the process represents a platform module. **Tags:** atp.Splitkey=nonOsModuleInstantiation
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.
process	Process	0..1	ref	This reference identifies the Process in the context of the ProcessToMachineMapping. **Tags:** atp.Splitkey=process
shallNotRunOn	ProcessorCore	*	ref	This reference indicates a collection of cores onto which the mapped process shall not be executing. **Tags:** atp.Splitkey=shallNotRunOn
shallRunOn	ProcessorCore	*	ref	This reference indicates a collection of cores onto which the mapped process shall be executing. **Tags:** atp.Splitkey=shallRunOn

Table 11.5: ProcessToMachineMapping

Primitive	**UriString**
Note	A Uniform Resource Identifier (URI), is a compact string of characters used to identify or name a resource. **Tags:** xml.xsd.customType=URI-STRING xml.xsd.type=string

Table 11.6: UriString

[constr_10174] Multiplicity of the reference in the role `ProcessToMachineMapping.process`

Imposition time: `IT_Mani`

⌈For each `ProcessToMachineMapping`, the reference in the role `process` shall exist.⌋

[constr_10658] Multiplicity of the reference in the role `ProcessToMachineMapping.machine`

Imposition time: `IT_Mani`

⌈For each `ProcessToMachineMapping`, the reference in the role `machine` shall exist.⌋

11.3.2 Core Affinity

There are two ways to define the core affinity of a Process:

- On *design level*, the references ProcessDesignToMachineDesignMapping.shallRunOn and ProcessDesignToMachineDesignMapping.shallNotRunOn can be used.

  This is a way to define core affinity in the early stages of a project, in the context of an overall system design.

- On *target configuration level*, the references ProcessToMachineMapping.shallRunOn and ProcessToMachineMapping.shallNotRunOn can be used.

  From the standard's perspective, it is in principle possible to override the definition of core affinity on the design level in this step. Projects need to decide whether this capability shall be available.

[TPS_MANI_03148] Description of Core affinity ⌈The meta-class ProcessToMachineMapping provides the ability to restrict the assignment of processes to selected ProcessorCores with the two references shallRunOn and shallNotRunOn.⌋

[constr_3393] Usage of shallRunOn and shallNotRunOn references

Imposition time: IT_Mani

⌈The 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.⌋

[constr_1676] Consistency of references shallRunOn and shallNotRunOn

Imposition time: IT_Mani

⌈Within the context of one ProcessToMachineMapping, all ProcessorCores referenced in the role shallRunOn or shallNotRunOn shall be aggregated by the same Processor.⌋

If a model defines that a given Process shall run on a selected set of ProcessorCores then there is hardly a use case to additionally 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] Mutual exclusive existence of references shallRunOn and shallNotRunOn

Imposition time: IT_Mani

⌈For any given ProcessToMachineMapping, either the reference in the role shallRunOn or the reference in the role shallNotRunOn may exist.⌋

11.3.3 Default Start-up and Termination Timeout

[TPS_MANI_03151] Default value for termination timeout ⌈The meta-class 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.⌋

[TPS_MANI_01412] Default value for startup timeout ⌈The meta-class Machine provides the ability to define a default value for startup timeout of applications in the context of the Machine with the attribute enterTimeoutValue that is available in the EnterExitTimeout meta-class that is aggregated by the Machine in the role defaultApplicationTimeout.⌋

[constr_10430] Existence of attribute Machine.defaultApplicationTimeout

Imposition time: IT_Mani

⌈For each Machine, the attribute defaultApplicationTimeout shall exist.⌋

[constr_10431] Existence of the attribute Machine.defaultApplicationTimeout.enterTimeoutValue

Imposition time: IT_Mani

⌈For each Machine, the attribute defaultApplicationTimeout.enterTimeoutValue shall exist.⌋

[constr_10432] Existence of the attribute Machine.defaultApplicationTimeout.exitTimeoutValue

Imposition time: IT_Mani

⌈For each Machine, the attribute defaultApplicationTimeout.exitTimeoutValue shall exist.⌋

Class	**EnterExitTimeout**
Note	This meta-class represents the ability to specify a pair of timeouts, one for entering, and one for exiting. This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARObject*
Aggregated by	Machine.defaultApplicationTimeout, StartupConfig.timeout, StateDependentStartupConfig.timeout

▽

△

Class	**EnterExitTimeout**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
enterTimeout Value	TimeValue	0..1	attr	This attribute represents the value of the enter timeout in seconds.
exitTimeout Value	TimeValue	0..1	attr	This attribute represents the value of the exit timeout in seconds.

Table 11.7: EnterExitTimeout

12 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.moduleInstantiation, 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.

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.

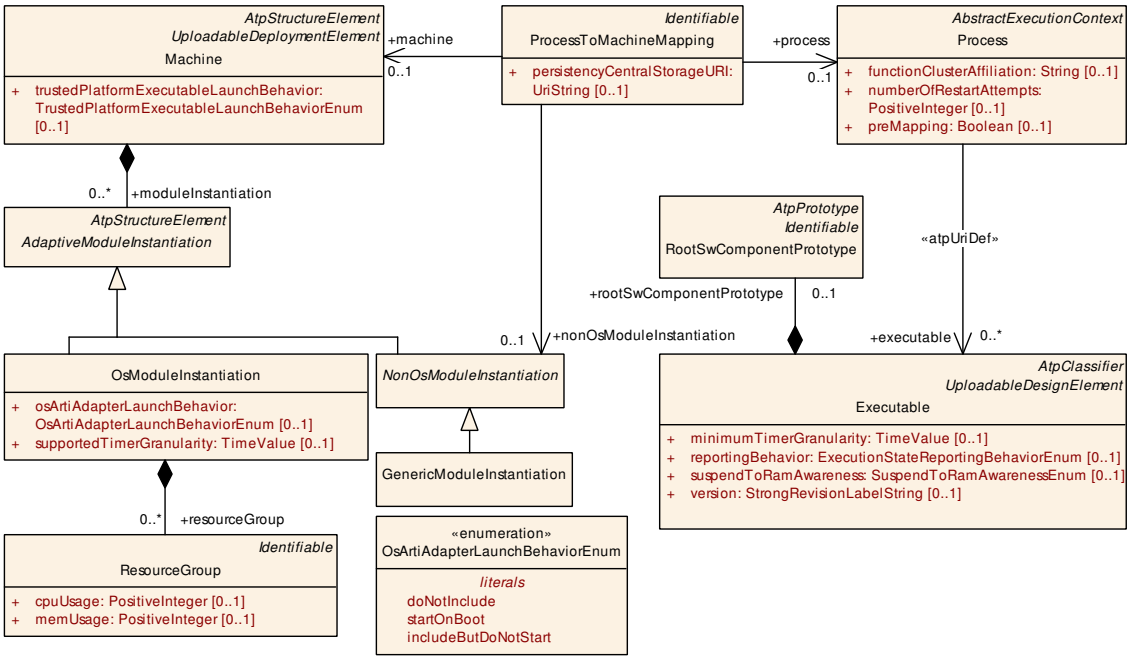


Figure 12.1: Adaptive Autosar Module Configuration

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.

Class	***AdaptiveModuleInstantiation*** (abstract)			
Note	This meta-class defines the abstract attributes for the configuration of an adaptive autosar module instance on a specific machine. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*NonOsModuleInstantiation*, OsModuleInstantiation			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 12.1: AdaptiveModuleInstantiation

Each Adaptive AUTOSAR module other than OS can be assigned to a Process with the ProcessToMachineMapping.

[constr_1490] Allowed value for Executable.category if ProcessToMachineMapping references a NonOsModuleInstantiation

Imposition time: IT_Mani

⌈If 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]).⌋

The model relation described in [constr_1490] is sketched in Figure 12.1.

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.

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, refer to the [8, TPS Generic Structure Template].

[TPS_MANI_03096] `Machine`-specific configuration settings for a generic module ⌈The `Machine`-specific configuration settings for a generic module are collected in `GenericModuleInstantiation` where the value of attribute `category` value denotes the module.⌋

Class	**GenericModuleInstantiation**			
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. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 12.2: GenericModuleInstantiation

12.1 OS Module Configuration

[TPS_MANI_03098] `Machine`-specific configuration settings for the OS module ⌈The `Machine`-specific configuration settings for the OS module are collected in `OsModuleInstantiation`.⌋

Class	**OsModuleInstantiation**			
Note	This meta-class defines the attributes for the OS configuration on a specific machine. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
osArtiAdapter LaunchBehavior	OsArtiAdapterLaunch BehaviorEnum	0..1	attr	This attribute controls the startup behavior of the OS/ ARTI Adapter. **Tags:** atp.Status=candidate
resourceGroup	ResourceGroup	*	aggr	This represents the collection of ResourceGroups owned by the enclosing OsModuleImplementation. **Tags:** atp.Splitkey=resourceGroup, resourceGroup.short Name
supportedTimer Granularity	TimeValue	0..1	attr	This attribute describes the supported timer granularity (TimeValue of one tick).

Table 12.3: OsModuleInstantiation

[TPS_MANI_01433] Semantics of attribute `OsModuleInstantiation.osArtiAdapterLaunchBehavior` ⌈The attribute `OsModuleInstantiation.osArtiAdapterLaunchBehavior` defines launch behavior of the OS/ARTI Adapter that is

used in the system for tracing the states of the operating system, as described in [35, AUTOSAR TR Operating System Tracing Interface].⌋

Enumeration	**OsArtiAdapterLaunchBehaviorEnum**
Note	This enumeration provides options to start the OS/ARTI Adapter when starting the machine. **Tags:** atp.Status=candidate This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	OsModuleInstantiation.osArtiAdapterLaunchBehavior
Literal	***Description***
doNotInclude	Do not include the OS/ARTI Adapter when building the system. **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=candidate
includeButDoNot Start	Include the OS/ARTI Adapter when building the system but do not start it automatically. **Tags:** atp.EnumerationLiteralIndex=2 atp.Status=candidate
startOnBoot	Include the OS/ARTI Adapter when building the system and start it when booting the system. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=candidate

Table 12.4: OsArtiAdapterLaunchBehaviorEnum

Class	***NonOsModuleInstantiation*** (abstract)			
Note	This meta-class defines the abstract attributes for the configuration of an adaptive autosar module other than the OS module. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	AdaptiveFirewallModuleInstantiation, CmModuleInstantiation, CryptoModuleInstantiation, DoIp Instantiation, GenericDiagnosticTransportInstantiation, GenericModuleInstantiation, *IdsPlatform Instantiation*, LogAndTraceInstantiation, NmInstantiation, *SovdModuleInstantiation*, StateManagement ModuleInstantiation, SuspendToRamModuleInstantiation, TimeSyncModuleInstantiation, *UcmModule Instantiation*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 12.5: NonOsModuleInstantiation

AUTOSAR supports the configuration of `ResourceGroup`s in the `OsModuleInstantiation` of the `Machine` that correspond for example to cgroups (aka control groups) in Linux. `ResourceGroup`s 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] Multiplicity of `OsModuleInstantiation.resourceGroup`

Imposition time: `IT_Mani`

⌈Any given `OsModuleInstantiation` shall always define at least one `resourceGroup`.⌋

The rationale for [constr_1661] is that the StateDependentStartupConfig requires a reference to a ResourceGroup.

Class	ResourceGroup			
Note	This meta-class represents a resource group that limits the resource usage of a collection of processes. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	OsModuleInstantiation.resourceGroup			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
cpuUsage	PositiveInteger	0..1	attr	CPU resource limit in percentage of the total CPU capacity on the machine.
memUsage	PositiveInteger	0..1	attr	Memory limit in bytes.

Table 12.6: ResourceGroup

12.2 Persistency Target-Configuration

Disclaimer

As of AUTOSAR R25-11, the canonical approach for the target configuration of the persistency functional cluster is by usage of the APMC model documented in [15, SWS Persistency] Chapter 10.1. The description of target configuration via the M2 model (as described in this chapter) is still provided, but may be discontinued at a later point in time.

12.2.1 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] Semantics of meta-class PersistencyDeployment ⌈Abstract meta-class PersistencyDeployment provides shared attributes to more specific specializations.⌋

[TPS_MANI_01380] Usage of meta-classPersistencyDeployment ⌈Meta-class PersistencyDeployment can be used in different scenarios:

- Persistency is used in combination with the application software by means of PersistencyPortPrototypeToDeploymentMapping.
- Persistency is used in combination with another functional cluster by means of FunctionalClusterInteractsWithPersistencyDeploymentMapping.

⌋

Class	***PersistencyDeployment*** (abstract)			
Note	This abstract meta-class serves as a base class for concrete classes representing different aspects of persistency. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadableExclusivePackageElement*, *UploadablePackageElement*			
Subclasses	PersistencyFileStorage, PersistencyKeyValueStorage			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
deploymentUri (ordered)	PersistencyDeployment Uri	*	aggr	This aggregation represents the collection of URIs relevant for the enclosing PersistencyDeployment.
maximum AllowedSize	PositiveUnlimitedInteger	0..1	attr	The value of this attribute represents the maximum size (unit: bytes) allowed at target-configuration time for the enclosing PersistencyDeployment.
minimum SustainedSize	PositiveInteger	0..1	attr	The value of this attribute represents the minimum size (unit: bytes) guaranteed at deployment time for the enclosing PersistencyDeployment.
redundancy Handling	PersistencyRedundancy Handling	*	aggr	This aggregation represents the chosen approaches to handle redundancy. **Tags:** atp.Splitkey=redundancyHandling
updateStrategy	PersistencyCollection LevelUpdateStrategy Enum	0..1	attr	This attribute shall be used to specify the update strategy of the respective PersistencyDeployment as a whole.
version	StrongRevisionLabel String	0..1	attr	The attribute represents the version of the `PersistencyFileStorage` or `PersistencyKeyValueStorage`.

Table 12.7: PersistencyDeployment

[constr_10177] Multiplicity of the attribute `PersistencyDeployment.updateStrategy`

Imposition time: `IT_Mani`

⌈For each `PersistencyDeployment`, the attribute `updateStrategy` shall exist.⌋

[constr_10365] Existence of `PersistencyDeployment.deploymentUri`

Imposition time: `IT_Mani`

⌈For each concrete sub-class of `PersistencyDeployment`, attribute `deploymentUri` shall exist.⌋

[constr_10509] Existence of the attribute `PersistencyDeployment.maximumAllowedSize`

Imposition time: `IT_Mani`

⌈For each concrete sub-class of `PersistencyDeployment`, attribute `maximumAllowedSize` shall exist.⌋

[constr_10510] Existence of the attribute `PersistencyDeployment.minimumSustainedSize`

Imposition time: `IT_Mani`

⌈For each concrete sub-class of `PersistencyDeployment`, attribute `minimumSustainedSize` shall exist.⌋

Class	**PersistencyDeploymentUri**			
Note	This meta-class represents the ability to contain URIs relevant for the persistency deployment. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	*PersistencyDeployment*.deploymentUri			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
uri	UriString	0..1	attr	This attribute holds the storage location for the concrete subclass of PersistencyDeployment, e.g. file on the file system.

Table 12.8: PersistencyDeploymentUri

The possibility that attribute `deploymentUri` has a multiplicity greater than 1 is only foreseen to be utilized if redundancy handling on the basis of m-out-of-n is activated, i.e. the aggregation of `PersistencyRedundancyMOutOfN` is aggregated in the role `PersistencyDeployment.redundancyHandling`.

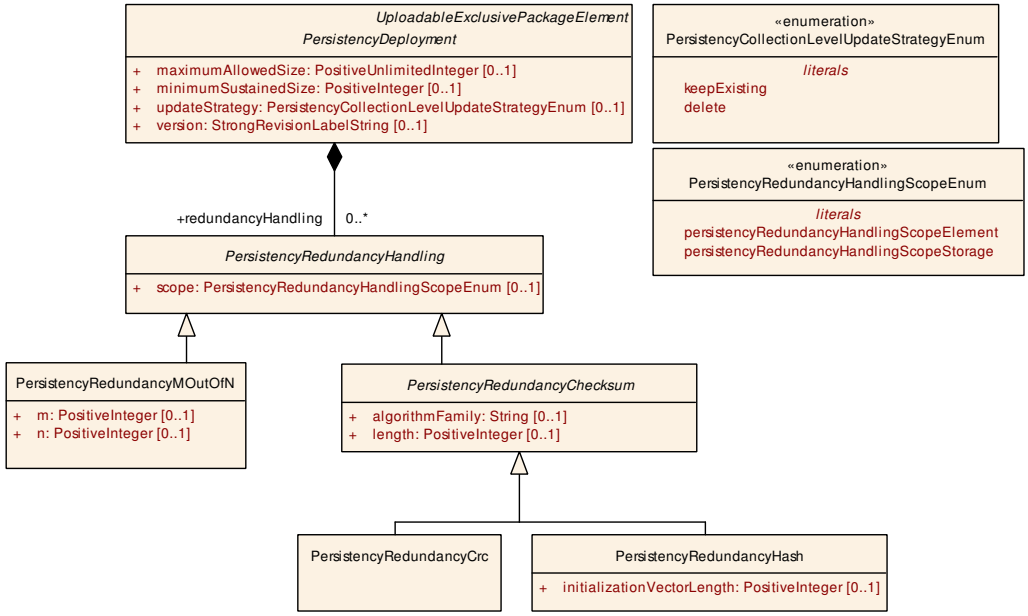


Figure 12.2: Modeling of the abstract base class `PersistencyDeployment`

In this case, the actual multiplicity of `PersistencyDeployment.deploymentUri` in an AUTOSAR model shall be one of

- 1: in this case all copies are stored in one location.
- 2: in this case, the “main” copy is stored in the location `deploymentUri`[0] and all other copies are stored at the location `deploymentUri`[1]

- the value of PersistencyRedundancyMOutOfN.n: each copy is stored in the location deploymentUri[i], where $0 <= i < n$

Please find more details about the storage strategy in the [15, SWS Persistency].

[constr_10366] Allowed multiplicities of the aggregation PersistencyDeployment.deploymentUri

Imposition time: IT_Mani

⌈Possible multiplicities of PersistencyDeployment.deploymentUri shall be one of

- 1
- 2
- value of attribute PersistencyRedundancyMOutOfN.n

⌋

[constr_10367] Condition for the multiplicity of the attribute PersistencyDeployment.deploymentUri

Imposition time: IT_Mani

⌈The multiplicity of attribute PersistencyDeployment.deploymentUri shall only be greater than 1 if meta-class PersistencyRedundancyMOutOfN is aggregated in the role PersistencyDeployment.redundancyHandling and attribute PersistencyDeployment.redundancyHandling.scope is set to the value PersistencyRedundancyHandlingScopeEnum.persistencyRedundancyHandlingScopeStorage.⌋

[constr_10035] Completeness of the PersistencyDeployment.version

Imposition time: IT_Mani

⌈The PersistencyDeployment.version shall contain all the following parts:

- Major version
- Minor version
- Patch version
- Additional labels for pre-release version and build metadata

⌋

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] Semantics of meta-class `PersistencyDeploymentElement`
⌈Meta-class `PersistencyDeploymentElement` represents an abstract base class for the modeling of different aspects of persistency on element level.⌋

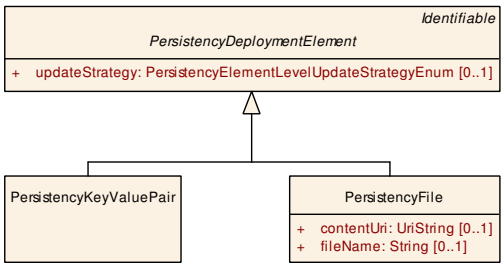


Figure 12.3: Modeling of the abstract base class `PersistencyDeploymentElement`

Class	***PersistencyDeploymentElement*** (abstract)			
Note	This abstract meta-class serves as a base class for concrete classes representing different aspects of elements of a `PersistencyDeployment`. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	PersistencyFile, PersistencyKeyValuePair			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
updateStrategy	PersistencyElementLevelUpdateStrategyEnum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyDeploymentElement.

Table 12.9: PersistencyDeploymentElement

[constr_10426] Multiplicity of attribute `PersistencyDeploymentElement.updateStrategy`

Imposition time: `IT_Mani`

⌈For each `PersistencyDeploymentElement`, the attribute `updateStrategy` shall exist.⌋

[TPS_MANI_01322] Semantics of the meta-class `PersistencyPortPrototypeToDeploymentMapping` ⌈The meta-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`.⌋

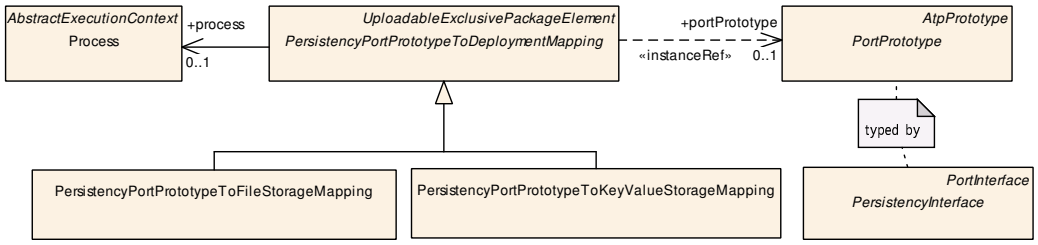


Figure 12.4: Modeling of the abstract base class `PersistencyPortPrototypeToDeploymentMapping`

Class	***PersistencyPortPrototypeToDeploymentMapping*** (abstract)			
Note	This abstract bas class implements the shared functionality of all mapping between a `PortPrototype`, a `Process`, and a specific subclass of `PersistencyDeployment`. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadableExclusivePackageElement*, *UploadablePackageElement*			
Subclasses	PersistencyPortPrototypeToFileStorageMapping, PersistencyPortPrototypeToKeyValueStorageMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
portPrototype	PortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **Tags:** atp.Splitkey=portPrototype, portPrototype.context RootSwComponentPrototype, portPrototype.context ComponentPrototype, portPrototype.targetPortPrototype **InstanceRef implemented by:** PortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference represents the process required as context for the mapping. **Tags:** atp.Splitkey=process

Table 12.10: PersistencyPortPrototypeToDeploymentMapping

[constr_10178] Multiplicity of the reference in the role `PersistencyPortPrototypeToDeploymentMapping.process`

Imposition time: `IT_Mani`

⌈For each `PersistencyPortPrototypeToDeploymentMapping`, the reference in the role `process` shall exist.⌋

12.2.1.1 Redundancy Handling

[TPS_MANI_01206] Modeling of redundancy in the context of `PersistencyDeployment` ⌈The target-configuration 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`.⌋

[constr_1710] Consistency of values of attributes `PersistencyInterface.redundancy` and `PersistencyRedundancyHandling.scope`

Imposition time: `IT_Mani`

⌈If attribute`PersistencyInterface.redundancy` is set to value `PersistencyRedundancyEnum.redundantPerElement`, then the attribute `PersistencyRedundancyHandling.scope` shall be set to value `PersistencyRedundancyHandlingScopeEnum.persistencyRedundancyHandlingScopeElement` for at least one `PersistencyRedundancyHandling` aggregated by the corresponding `PersistencyDeployment`.⌋

[TPS_MANI_01371] Attribute `PersistencyDeployment.redundancyHandling` does not exist ⌈If no aggregation in the role `redundancyHandling` exists for a specific `PersistencyDeployment`, then the configuration of the redundancy handling shall be taken from the value of `PersistencyInterface.redundancyHandling`, if this aggregation exists.⌋

12.2.1.2 Update Handling

[TPS_MANI_01155] The attribute `PersistencyDeployment.updateStrategy` overrides `PersistencyInterface.updateStrategy` ⌈The value of the attribute `PersistencyDeployment.updateStrategy` shall overrule the value of attribute `PersistencyInterface.updateStrategy` for any combination of `PersistencyInterface` mapped to a `PersistencyDeployment` by means of a `PersistencyPortPrototypeToDeploymentMapping`.⌋

[TPS_MANI_01147] Semantics of `PersistencyDeployment.updateStrategy` ⌈The 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 `PersistencyDeploymentElement`s that define their own `updateStrategy`.⌋

[TPS_MANI_01157] Semantics of `updateStrategy` on collection level ⌈

`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

⌋

[TPS_MANI_01313] Definition of `updateStrategy` on element level ⌈The 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.⌋

[TPS_MANI_01159] Semantics of updateStrategy on element level ⌈

updateStrategy	Use Case: Installation	Use Case: Update
delete	don't create	remove
keepExisting	create	do nothing
overwrite	create	replace

⌋

[TPS_MANI_01148] Semantics of PersistencyDeploymentElement.updateStrategy ⌈The attribute PersistencyDeploymentElement.updateStrategy can be used to specify the strategy for updating the actual persistent element that corresponds to PersistencyDeploymentElement.⌋

[TPS_MANI_01156] PersistencyDeploymentElement.updateStrategy overrides PersistencyDeployment.updateStrategy ⌈The value specified for PersistencyDeploymentElement.updateStrategy overrides the value of PersistencyDeployment.updateStrategy for this specific PersistencyDeploymentElement.⌋

[TPS_MANI_01182] Value of attribute PersistencyDeploymentElement.updateStrategy overrides the value of PersistencyInterfaceElement.updateStrategy ⌈The value of attribute PersistencyDeploymentElement.updateStrategy overrides the value of attribute PersistencyInterfaceElement.updateStrategy⌋

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.

12.2.1.3 Size Handling

[TPS_MANI_01196] Semantics of attribute PersistencyDeployment.minimumSustainedSize ⌈Attribute 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.⌋

[TPS_MANI_01197] Semantics of attribute PersistencyDeployment.maximumAllowedSize ⌈Attribute 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 attribute PersistencyDeployment.maximumAllowedSize does not constitute a binding requirement to the platform that this amount of storage shall be available at any time.⌋

For explanation, the amount of storage available shall be at least the sum of the values of attribute `PersistencyDeployment.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.

12.2.1.4 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 15.4 and Chapter 15.5.

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.

The `PersistencyDeploymentToCryptoKeySlotMapping` is able to define a `verificationHash` that shall be used by the PersistencyCluster to verify the data. The same is true for the `PersistencyDeploymentElementToCryptoKeySlotMapping.verificationHash`.

12.2.1.5 Preference Rules

In the context of the modeling of persistency design (see Chapter 4.7) and target-configuration (Chapter 12.2), information can be available on the *design* and *target-configuration* level, and on *storage* and *element* level. In general, the following rules apply regarding the preference of this information:

- if a configuration item exists in the context of the *design element level*, then the value of this configuration item shall be preferred over the value of a corresponding configuration item available on the *design storage level*.
- if a configuration item exists in the context of the *target-configuration element level*, then the value of this configuration item shall be preferred over the value of a corresponding configuration item available on the *target-configuration storage level*.

- if a configuration item exists in the context of the *target-configuration storage level*, then the value of this configuration item shall be preferred over the value of a corresponding configuration item available on the *design storage level*.
- if a configuration item exists in the context of the *target-configuration element level*, then the value of this configuration item shall be preferred over the value of a corresponding configuration item available on the *design element level*.

In other words, there is a growing priority of the value of a configuration item starting from *design storage level*, to *design element level*, to *target-configuration storage level*, and finally to *target-configuration element level*.

On the other hand, models may exist that leave out certain information on the target-configuration level in favor of implicitly taking over the corresponding information from design level. This approach is not encouraged because (where possible) model semantics should preferably be defined explicitly rather than implied by rules.

Nevertheless, the following rules apply:

- if a configuration item exists on the *target-configuration storage level* but is missing on the *target-configuration element level*, then the *target-configuration element level* shall implicitly take over the value from the *target-configuration storage level*.
- if a configuration item exists on the *design storage level* but is missing on the *target-configuration storage level*, then the *target-configuration storage level* shall implicitly take over the value from the *design storage level*.
- if a configuration item exists on the *design element level* but is missing on the *target-configuration element level*, then the *target-configuration element level* shall implicitly take over the value from the *design element level*.
- if a configuration item exists on the *design storage level* but is missing on the *design element level*, then the *design element level* implicitly takes over the value from the *design storage level*.

12.2.2 Target-Configuration of Persistent Key-Value Storage

[TPS_MANI_01079] Semantics of meta-class `PersistencyKeyValueStorage`
⌈Meta-class `PersistencyKeyValueStorage` represents an actual key-value storage used for persistently storing data.⌋

Class	**PersistencyKeyValueStorage**
Note	This meta-class represents the ability to model a key-value storage on target-configuration level. **Tags:** atp.recommendedPackage=PersistencyKeyValueStorages This Class is only used by the AUTOSAR Adaptive Platform.

▽

△

Class	PersistencyKeyValueStorage			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, PersistencyDeployment, Referrable, UploadableDeploymentElement, UploadableExclusivePackageElement, UploadablePackageElement			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
keyValuePair	PersistencyKeyValuePair	*	aggr	This aggregation represents the key-value-pairs owned by the enclosing PersistencyKeyValueStorage. **Tags:** atp.Splitkey=keyValuePair, keyValuePair.shortName

Table 12.11: PersistencyKeyValueStorage

[TPS_MANI_01144] Semantics of PersistencyKeyValuePair ⌈Meta-class PersistencyKeyValuePair represents an **entry** to a key-value storage (formalized by PersistencyKeyValueStorage) used for persistently storing data.⌋

[constr_10102] Existence of initial values for PersistencyKeyValuePair

Imposition time: IT_Mani

⌈For each PersistencyKeyValuePair, if the value of attribute updateStrategy is set to either of the values

- keepExisting or
- overwrite,

then attribute PersistencyKeyValuePair.initValue shall exist.⌋

Class	PersistencyKeyValuePair			
Note	This meta-class represents the ability to formally model a key-value pair in the context of the target-configuration of persistency. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARObject, Identifiable, MultilanguageReferrable, PersistencyDeploymentElement, Referrable			
Aggregated by	PersistencyKeyValueStorage.keyValuePair			
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.Splitkey=initValue
valueDataType	AbstractImplementationDataType	0..1	ref	This reference represents the data type applicable for the value of the key-value pair. **Tags:** atp.Splitkey=valueDataType

Table 12.12: PersistencyKeyValuePair

[constr_10179] Multiplicity of attribute PersistencyKeyValuePair.valueDataType

Imposition time: IT_Mani

⌈For each PersistencyKeyValuePair, the attribute valueDataType shall exist.⌋

[constr_10083] Existence of initial values for PersistencyKeyValuePair

Imposition time: IT_Mani

⌈For each PersistencyKeyValuePair, if the value of attribute updateStrategy is set to the value delete, then attribute PersistencyKeyValuePair.initValue shall not exist.⌋

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.

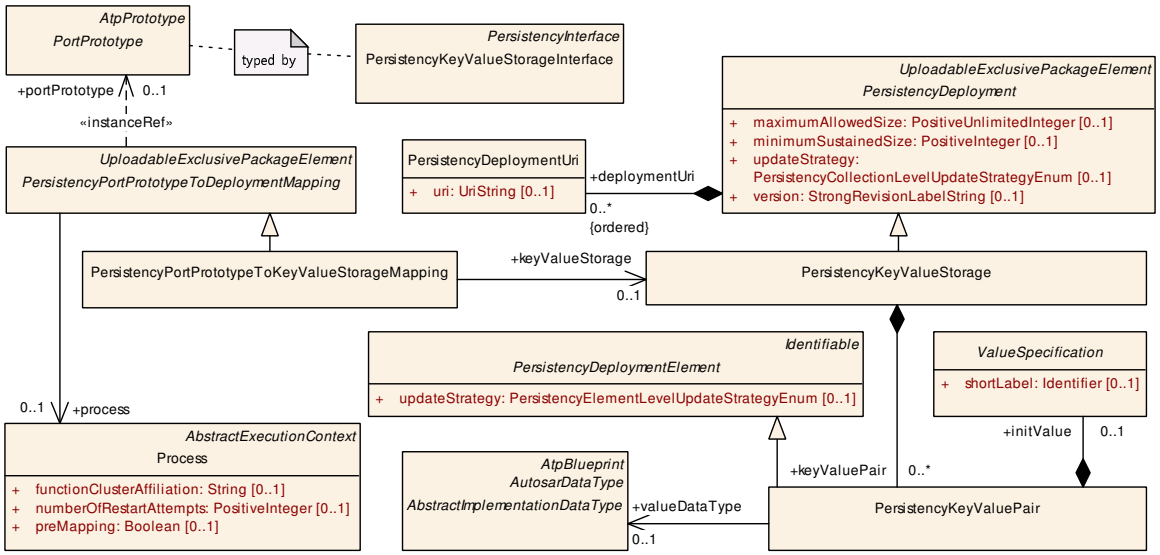


Figure 12.5: Connect a specific PortPrototype to a PersistencyKeyValueStorage

[TPS_MANI_01078] Semantics of PersistencyPortPrototypeToKeyValueStorageMapping ⌈Meta-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.⌋

[constr_1555] Restriction applicable for PersistencyPortPrototypeToKeyValueStorageMapping.portPrototype

Imposition time: IT_Mani

⌈The reference in the role PersistencyPortPrototypeToKeyValueStorageMapping.portPrototype shall only be used for a PortPrototype typed by a PersistencyKeyValueStorageInterface.⌋

Class	PersistencyPortPrototypeToKeyValueStorageMapping			
Note	This meta-class represents the ability to define a mapping between a PortPrototype and a key-value storage. **Tags:** atp.recommendedPackage=PersistencyPortPrototypeToKeyValueStorageMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyPortPrototypeToDeploymentMapping*, *Referrable*, *UploadableDeploymentElement*, *UploadableExclusivePackageElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
keyValue Storage	PersistencyKeyValue Storage	0..1	ref	This reference represents the mapped key-value storage. **Tags:** atp.Splitkey=keyValueStorage

Table 12.13: PersistencyPortPrototypeToKeyValueStorageMapping

[constr_10180] Multiplicity of the reference in the role PersistencyPortPrototypeToKeyValueStorageMapping.keyValueStorage

Imposition time: IT_Mani

⌈For each PersistencyPortPrototypeToKeyValueStorageMapping, the reference in the role keyValueStorage shall exist.⌋

Typically, the existence of PersistencyKeyValueStorage.keyValuePair depends on the existence of PersistencyKeyValueStorageInterface.dataElement.

On the other hand, if a PersistencyKeyValueStorage contains PersistencyKeyValuePairs that do not correspond to any dataElements of the enclosing PersistencyKeyValueStorageInterface that is mapped (indirectly) via PersistencyPortPrototypeToKeyValueStorageMapping then those keyValuePairs are created within the PersistencyKeyValueStorage.

[TPS_MANI_01146] Initial value for PersistencyKeyValuePair ⌈It is possible to define an initial value for a given PersistencyKeyValuePair by means of the aggregation of ValueSpecification in the role initValue.⌋

[constr_1554] Restriction regarding the value of attribute PersistencyKeyValuePair.initValue

Imposition time: IT_Mani

⌈The concrete sub-class of meta-class ValueSpecification aggregated in the role PersistencyKeyValuePair.initValue shall not (after resolving a possible

redirection by means of meta-class ConstantReference) be a ReferenceValue-Specification.⌋

[TPS_MANI_01315] The value of attribute PersistencyKeyValuePair.initValue overrides the value of attribute PersistencyDataRequiredComSpec.initValue ⌈The value of attribute PersistencyKeyValuePair.initValue shall overrule the value of PersistencyDataRequiredComSpec.initValue for any combination of PersistencyKeyValueStorageInterface mapped to a PersistencyKeyValueStorage by means of a PersistencyPortPrototype-ToKeyValueStorageMapping.⌋

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.

[constr_1582] The value of attribute PersistencyKeyValuePair.valueData-Type shall match to AbstractImplementationDataType for the corresponding PersistencyDataElement

Imposition time: IT_Mani

⌈Each PersistencyKeyValuePair.valueDataType shall match the Abstract-ImplementationDataType that either directly or indirectly (via the applicable Data-TypeMap) types the corresponding (based on identical values of the respective shortName) PersistencyDataElement.⌋

[constr_1666] References from meta-class PersistencyPortPrototypeTo-KeyValueStorageMapping to PersistencyKeyValueStorage

Imposition time: IT_Mani

⌈Each PersistencyKeyValueStorage shall only be referenced by at most one PersistencyPortPrototypeToKeyValueStorageMapping.⌋

[TPS_MANI_01323] Identify matching pairs of PersistencyDataElement and PersistencyKeyValuePair ⌈Matching pairs of PersistencyDataElement and PersistencyKeyValuePair shall be identified by having the identical value of attribute shortName within the scope of a PersistencyKeyValueStorage-Interface (or a PortPrototype typed by the PersistencyKeyValueStorage-Interface) mapped to a PersistencyKeyValueStorage by means of a PersistencyPortPrototypeToKeyValueStorageMapping.⌋

12.2.3 Target-Configuration of File Storage

[TPS_MANI_01150] Semantics of meta-class PersistencyFileStorage ⌈A PortPrototype typed by a PersistencyFileStorageInterface actually builds an abstraction for an entire directory of files.

This abstraction is also visible in the target-configuration by means of the existence of the companion meta-class PersistencyFileStorage.

This approach allows for the dynamic creation and/or deletion of files during runtime while still keeping the structural model of the file interaction static.⌋

At one point, however, it is necessary to boil down the relation of such a PortPrototype typed by a PersistencyFileStorageInterface to individual files and how these individual files are represented on the file system themselves.

This aspect is covered by the modeling of meta-class PersistencyPortPrototypeToFileStorageMapping, as depicted in Figure 12.6.

Class	**PersistencyFileStorage**			
Note	This meta-class comes with the ability to define a collection of single files (directory) that creates the deployment-side counterpart to a PortPrototype typed by a PersistencyFileStorageInterface. **Tags:** atp.recommendedPackage=PersistencyFileStorages This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PersistencyDeployment*, *Referrable*, *UploadableDeploymentElement*, *UploadableExclusive PackageElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
deleteStrategy	PersistencyDelete StrategyEnum	0..1	attr	This attribute can be used to configure the reaction to the existence of too many files in the file storage
file	PersistencyFile	*	aggr	This aggregation represents the collection of files aggregated by the PersistencyFileStorage. **Tags:** atp.Splitkey=file, file.shortName
maxNumberOf Files	PositiveInteger	0..1	attr	Upper bound for the number of files in this PersistencyFile Storage.

Table 12.14: PersistencyFileStorage

Enumeration	**PersistencyDeleteStrategyEnum**
Note	This enumeration provides values for configuring the delete strategy for persistency file storage. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	PersistencyFileStorage.deleteStrategy
Literal	***Description***
deleteOldest	Delete the oldest file when a new file would exceed the configured threshold for the maximum number of files **Tags:** atp.EnumerationLiteralIndex=1
keepAndFail	Report error if the configured maximum number of files is exceeded **Tags:** atp.EnumerationLiteralIndex=0

Table 12.15: PersistencyDeleteStrategyEnum

[TPS_MANI_01080] Semantics of meta-class PersistencyPortPrototypeToFileStorageMapping ⌈Meta-class PersistencyPortPrototypeToFileStorageMapping creates a mapping between a PortPrototype referenced in the role portPrototype to a PersistencyFileStorage referenced in the role fileStorage under consideration of a Process referenced in the role process.⌋

Class	PersistencyPortPrototypeToFileStorageMapping			
Note	This meta-class represents the ability to define a mapping between a collection of files on deployment level to a given PortPrototype. **Tags:** atp.recommendedPackage=PersistencyPortPrototypeToFileStorageMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, PersistencyPortPrototypeToDeploymentMapping, Referrable, UploadableDeploymentElement, UploadableExclusivePackageElement, UploadablePackageElement			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
fileStorage	PersistencyFileStorage	0..1	ref	This reference represents the mapped file storage. **Tags:** atp.Splitkey=fileStorage

Table 12.16: PersistencyPortPrototypeToFileStorageMapping

[constr_10182] Multiplicity of the reference in the role PersistencyPortPrototypeToFileStorageMapping.fileStorage

Imposition time: IT_Mani

⌈For each PersistencyPortPrototypeToFileStorageMapping, the reference in the role fileStorage shall exist.⌋

[TPS_MANI_01149] Semantics of PersistencyFileStorage.file

⌈The usage of PersistencyFileStorage.file allows for the explicit modeling of elements of the PersistencyFileStorage.

The creation of this aggregation is optional. It can be used to define the update strategy and/or initial content of selected files.⌋

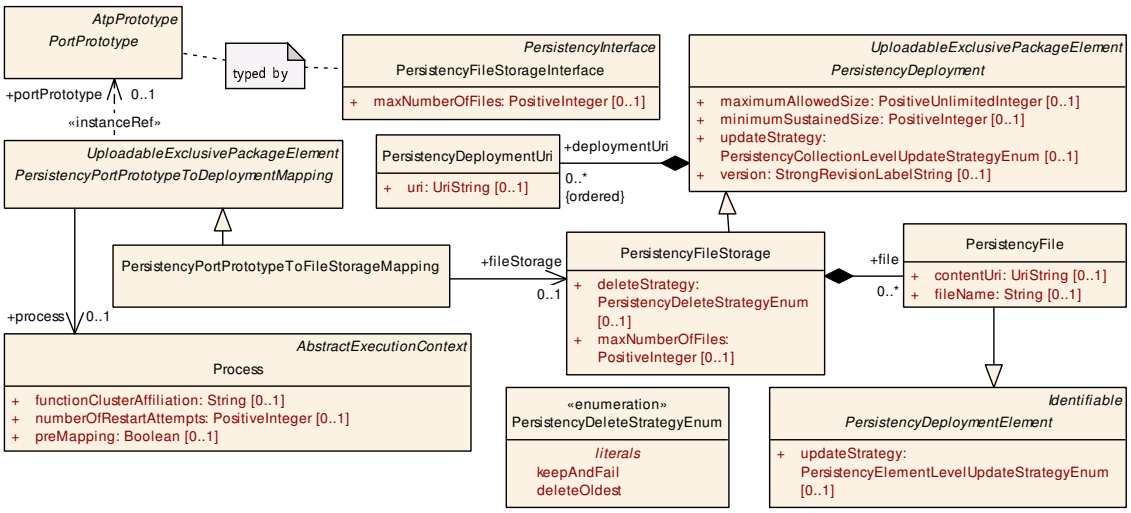


Figure 12.6: Connect a specific PortPrototype to a PersistencyFile

[constr_10103] Existence of initial values for `PersistencyFile`

Imposition time: `IT_Mani`

⌈For each `PersistencyFile`, if the value of attribute `updateStrategy` is set to either of the values

- `keepExisting` or
- `overwrite`,

then attribute `PersistencyFile.contentUri` shall exist.⌋

[constr_1556] Restriction applicable for `PersistencyPortPrototypeToFileStorageMapping.portPrototype`

Imposition time: `IT_Mani`

⌈The reference `PersistencyPortPrototypeToFileStorageMapping.portPrototype` shall only be used for a `PortPrototype` typed by a `PersistencyFileStorageInterface`.⌋

Class	**PersistencyFile**			
Note	This meta-class represents the model of a file as part of the persistency on deployment level. **Tags:** atp.recommendedPackage=PersistencyFiles This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PersistencyDeploymentElement*, *Referrable*			
Aggregated by	PersistencyFileStorage.file			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contentUri	UriString	0..1	attr	This attribute represents the URI that identifies the initial content of the PersistencyFile.
fileName	String	0..1	attr	This attribute holds filename part of the storage location for the PersistencyFile, e.g. file on the file system.

Table 12.17: PersistencyFile

[constr_10183] Multiplicity of attribute `PersistencyFile.fileName`

Imposition time: `IT_Mani`

⌈For each `PersistencyFile`, the attribute `fileName` shall exist.⌋

[constr_10082] Existence of initial values for `PersistencyFile`

Imposition time: `IT_Mani`

⌈For each `PersistencyFile`, if the value of attribute `updateStrategy` is set to the value `delete`, then attribute `PersistencyFile.contentUri` shall not exist.⌋

[TPS_MANI_01179] Semantics of PersistencyFileElement.contentUri/PersistencyFile.contentUri vs. PersistencyDeployment.deploymentUri and PersistencyFileElement.fileName/PersistencyFile.fileName ⌈Attributes PersistencyFileElement.contentUri and (after target-configuration) PersistencyFile.contentUri describe the URI of the file storage that is used to initialize the PersistencyFile (used during install or update).

On the other hand, the combination of PersistencyDeployment.deploymentUri and the PersistencyFileElement.fileName or (after target-configuration) PersistencyFile.fileName denote the position of the PersistencyFile in the ECU (used at run-time).⌋

[constr_1589] Value of file.fileName

Imposition time: IT_Mani

⌈Within the scope of any given PersistencyFileStorage, the value of all file.fileName shall be unique.

A fileName is considered unique if there are no other fileNames with **exactly** the same sequence of characters[1].⌋

[TPS_MANI_01187] Matching pairs of PersistencyFileElement and PersistencyFile ⌈Matching pairs of PersistencyFileElement and PersistencyFile shall be identified by having the identical value of attribute shortName within the scope of a PersistencyFileStorageInterface (or a PortPrototype typed by the PersistencyFileStorageInterface) mapped to a PersistencyFileStorage by means of a PersistencyPortPrototypeToFileStorageMapping.⌋

[constr_1613] File name of matching pairs of PersistencyFileElement and PersistencyFile

Imposition time: IT_Mani

⌈The value of attributes PersistencyFileElement.fileName and PersistencyFile.fileName shall be identical for matching pairs (as identified by the application of [TPS_MANI_01187]) of PersistencyFileStorage and PersistencyFile.⌋

[constr_1667] References from PersistencyPortPrototypeToFileStorageMapping to PersistencyFileStorage

Imposition time: IT_Mani

⌈Each PersistencyFileStorage shall only be referenced by at most one PersistencyPortPrototypeToFileStorageMapping.⌋

[1] The characters “x” and “X” are not considered as identical characters for this purpose.

12.3 Platform Health Management Target-Configuration

12.3.1 Overview

This chapter explains the interaction of application software with the [14, SWS Platform Health Management].

The `PlatformHealthManagementContribution` allows describing aspects for the target-configuration of configuration how the Platform Health Management shall behave during runtime.

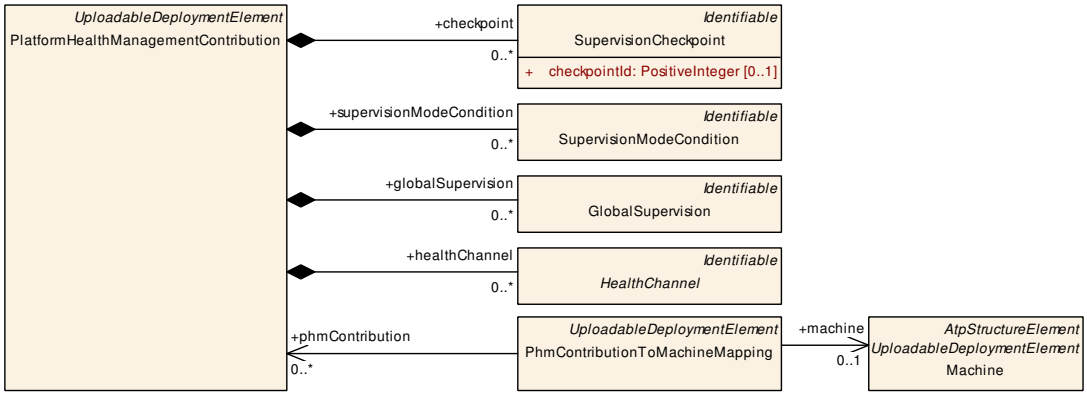


Figure 12.7: Modeling of `PlatformHealthManagementContribution`

[TPS_MANI_03544] Definition of `PlatformHealthManagementContribution`
⌈The meta-class `PlatformHealthManagementContribution` allows to define a set of configuration entities for the Platform Health Management.⌋

The `PlatformHealthManagementContribution` is structured into several aspects which will be described in the following sections:

- Supervision (Chapter 12.3.3)
- Recovery Notification (Chapter 12.3.5)

Class	**PlatformHealthManagementContribution**			
Note	This element defines a contribution to the Platform Health Management. **Tags:** atp.recommendedPackage=PlatformHealthManagementContributions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checkpoint	SupervisionCheckpoint	*	aggr	Collection of checkpoints in the context of a Platform HealthManagementContribution. **Tags:** atp.Splitkey=checkpoint, checkpoint.shortName xml.sequenceOffset=10

▽

△

Class	**PlatformHealthManagementContribution**			
global Supervision	GlobalSupervision	*	aggr	Collection of GlobalSupervisions in the context of a PlatformHealthManagementContribution. **Tags:** atp.Splitkey=globalSupervision, globalSupervision.shortName xml.sequenceOffset=30
healthChannel	HealthChannel	*	aggr	Collection of HealthChannels in the context of a PlatformHealthManagementContribution. **Tags:** atp.Splitkey=healthChannel, healthChannel.shortName xml.sequenceOffset=40
supervision ModeCondition	SupervisionMode Condition	*	aggr	Collection of SupervisionModeConditions in the context of a PlatformHealthManagementContribution. **Tags:** atp.Splitkey=supervisionModeCondition, supervision ModeCondition.shortName xml.sequenceOffset=20

Table 12.18: PlatformHealthManagementContribution

[TPS_MANI_03502] Enabling of `PlatformHealthManagementContribution` on a `Machine` ⌈To enable an instance of `PlatformHealthManagementContribution` on a specific `Machine`, the `PlatformHealthManagementContribution` shall be mapped to the `Machine` via a `PhmContributionToMachineMapping`.⌋

[constr_3568] No support for references across `PlatformHealthManagementContribution`

Imposition time: `IT_Mani`

⌈All references originating on elements aggregated by one `PlatformHealthManagementContribution` shall only refer to elements that are part of the same `PlatformHealthManagementContribution` aggregation chain.⌋

Class	**PhmContributionToMachineMapping**			
Note	This element associates one or more PlatformHealthManagementContributions with a Machine. **Tags:** atp.recommendedPackage=PhmContributionToMachineMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
machine	Machine	0..1	ref	This reference identifies the Machine in the context of the PhmContributionToMachineMapping. **Tags:** atp.Splitkey=machine

▽

△

Class	PhmContributionToMachineMapping			
phm Contribution	PlatformHealth Management Contribution	*	ref	This reference identifies one or more PlatformHealth ManagementContributions in the context of a Phm ContributionToMachineMapping. **Tags:** atp.Splitkey=phmContribution

Table 12.19: PhmContributionToMachineMapping

An application software can define the usage of several Platform Health Management supervisions (see Chapter 4.9.2).

In order to define the interaction between the application software and the Platform Health Management the `PlatformHealthManagementContribution` creates its own representations of the `RPortPrototype`s typed by the `PhmSupervised-EntityInterface` and creates relations to the application software `RPort-Prototype`s (see Figure 12.8).

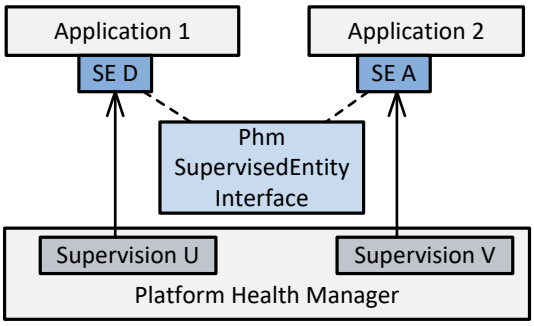


Figure 12.8: Interaction of application software with the platform health manager

In Chapter 4.9.2 it is explained that the application software just calls methods in the context of the respective `RPortPrototype`s to interact with the Platform Health Management. From the application developer these methods have no addressing information, because the identity of the `RPortPrototype` is the identification in the scope of the application software.

The deployed structure (according to Figure 12.7) however requires more information when an `API` at the Platform Health Manager is called, namely:

- `RPortPrototype.shortName` i.e. `InstanceSpecifier`
- `Process` identification during runtime.

These additional arguments have to be injected to the `API` by the implementation of the interaction between the software component and the Platform Health Management (which implements the relations from Figure 12.7). The order of this argument injection is determined by the specification of the Platform Health Management APIs.

12.3.2 Relation between design and Target-Configuration

The application design in Chapter 4.9.2 uses the declarations provided in the Interface definitions for `PhmSupervisedEntityInterface`. Specifically the handling of ids (`checkpointId`) requires a synchronized usage with the respective Interface definition.

It is required to establish a contract between the application code and the Phm target-configuration. The application code shall only use such id values which are declared at the respective Interface definitions.

During the configuration of the PHM the PHM artifacts are created. At target-configuration state there is no access to the design model available, thus the numeric values used for `checkpointId` at the Interface definition (`PhmSupervisedEntity-Interface`) are not available to the Phm.

Therefore the numeric values for `checkpointId` are replicated in the target-configuration model of the PHM: `SupervisionCheckpoint.checkpointId` replicates `PhmCheckpoint.checkpointId`

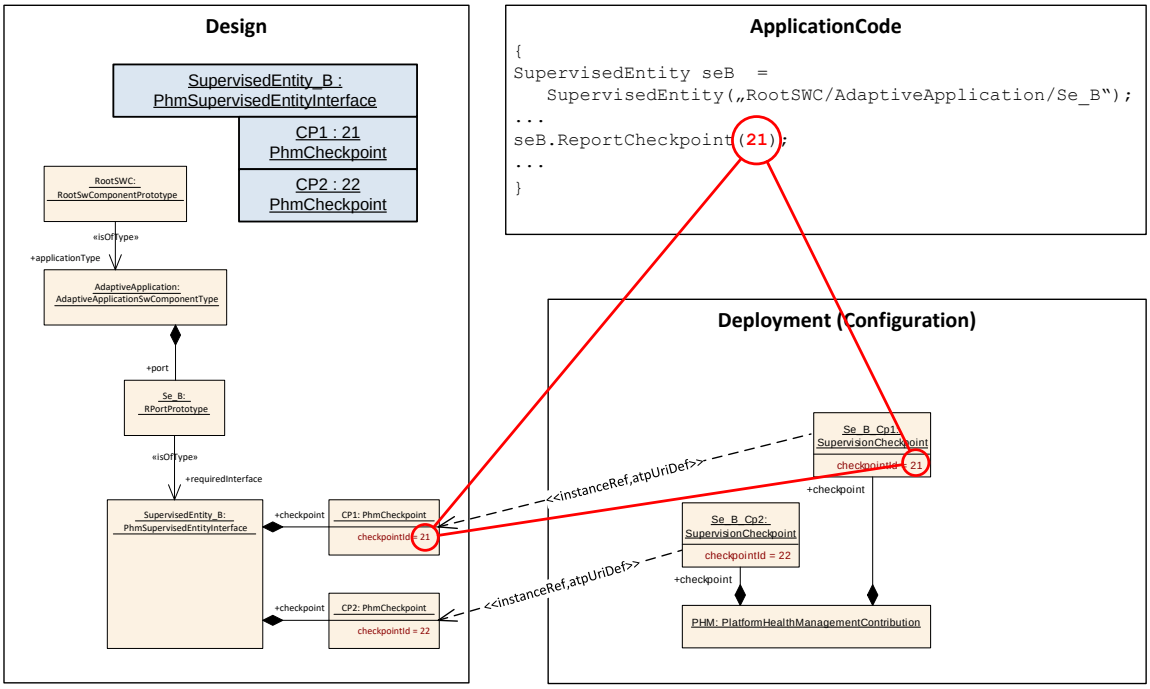


Figure 12.9: Example how IDs have to be in sync

It is a methodological task to make sure that the Ids correspond to each other and match in value. This consistency can be checked using specific tooling on the target-configuration model and the design model.

The example in Figure 12.9 illustrates that the used IDs have to in sync in order to let design, code, and configuration work together.

12.3.3 Supervision Target-Configuration

In the application design chapter of this document the declaration of supervised entities and checkpoints has been described (see Chapter 4.9.2). These declarations provide the view on supervision from the application software code point. Since the application `Executable` can be started multiple times (via individual `Process`es), the configuration of the Platform Health Management needs to cope with these individual `Executable` instances.

[TPS_MANI_03503] Applicability of checkpoints to a specific `Process` ⌈The reference `SupervisionCheckpoint.process` defines to which specific `Process` this `SupervisionCheckpoint` definition shall be applied to.⌋

This means that only if a `PhmCheckpoint` is reported from the context of this `Process` it is considered to be this `SupervisionCheckpoint`.

If the `SupervisionCheckpoint` represents the supervision of the *Execution Management* process, it will not be possible to configure `SupervisionCheckpoint.process` (unless there is a dummy configuration of *Execution Management* process) as the process for Execution Management is not a modeled `Process`.

It is up to the Adaptive Platform Stack Vendor to define the mechanism how to identify the `SupervisionCheckpoint` (thereby the supervisions `AliveSupervision`, `DeadlineSupervision`, and `LogicalSupervision`) corresponding to the *Execution Management*.

For the Platform Health Management supervision to take effect it is required to define the instance the application is executed in, thus the reference to a `Process` has to be taken into account. In the model the `Process` also defines under which conditions (`StateDependentStartupConfig`) and with which arguments (`ProcessArgument`) the `Executable` will be started.

To also define the conditions for the supervision the state dependency is defined at the supervisions as well, see Chapter 12.3.3.2.

For the configuration of the Platform Health Management the definition of `SupervisionCheckpoint` is used to stand in for the corresponding `PhmCheckpoint` including the execution context of the respective `Process`.

The attribute `SupervisionCheckpoint.phmCheckpoint` replicates the value of the referenced `PhmCheckpoint.checkpointId`. During target-configuration the `PhmSupervisedEntityInterface` and its content is no longer available and therefore needs to be made available to the Phm.

[TPS_MANI_03626] Consistency of `SupervisionCheckpoint.phmCheckpoint` and `PhmCheckpoint.checkpointId` ⌈The value of `SupervisionCheckpoint.phmCheckpoint` shall be identical to the value of `PhmCheckpoint.checkpointId` which is referenced in `SupervisionCheckpoint.phmCheckpoint`.⌋

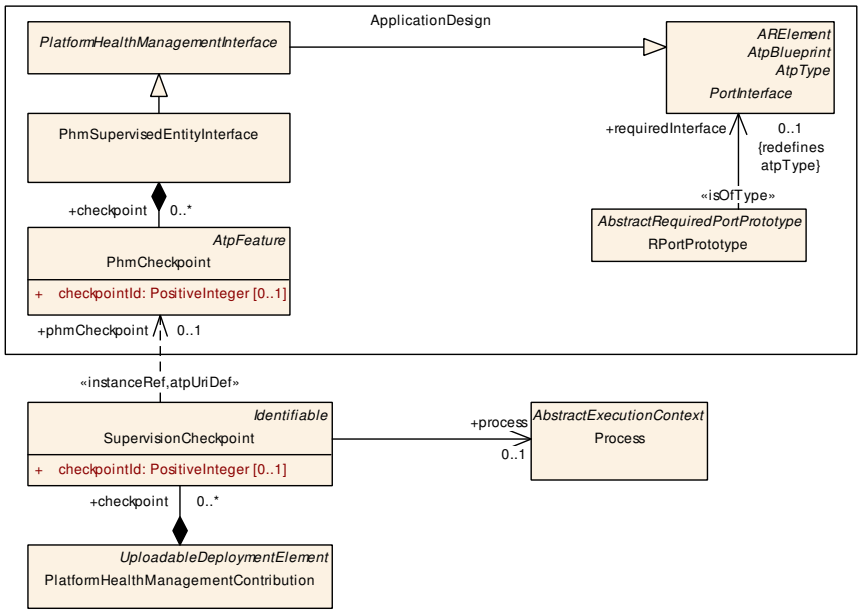


Figure 12.10: Modeling of `SupervisionCheckpoint`

[TPS_MANI_03505] Existence of `SupervisionCheckpoint` ⌈For each `PhmCheckpoint` in the scope of a `Process` and `RPortPrototype` typed by a `PhmSupervisedEntityInterface` in the application definition, a `SupervisionCheckpoint` shall be defined. The correspondence of the two is defined by the instance reference `SupervisionCheckpoint.phmCheckpoint`.⌋

Since every possibly reported `PhmCheckpoint` is represented by a corresponding `SupervisionCheckpoint`, it is possible to explicitly exclude a set of `SupervisionCheckpoint`s from the supervision activities using the `NoCheckpointSupervision` definition (see Chapter 12.3.3.9).

[TPS_MANI_03651] Mandatory reference in the role `SupervisionCheckpoint.process` ⌈Unless the `SupervisionCheckpoint` refers to a `PhmCheckpoint` which is defined in the scope of the *Execution Management*, each `SupervisionCheckpoint` shall define a process reference.⌋

[constr_3649] Consistent `SupervisionCheckpoint.process` reference

Imposition time: `IT_Mani`

⌈If a `SupervisionCheckpoint` refers to a `Process` in the role `process`, then

- the `SupervisionCheckpoint` shall refer to a `PhmCheckpoint` which is defined in a `PhmSupervisedEntityInterface` and
- that `PhmSupervisedEntityInterface` types an `RPortPrototype` of an `AdaptiveApplicationSwComponentType` and
- that `AdaptiveApplicationSwComponentType` is used in the scope of an `Executable` and

- that Executable is referenced by the same Process as SupervisionCheckpoint.process.

⌋

Class	SupervisionCheckpoint			
Note	This element contains an instance reference to a RPortPrototype representing a checkpoint for Platform Health Management. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	PlatformHealthManagementContribution.checkpoint			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
checkpointId	PositiveInteger	0..1	attr	Defines the numeric value which is used to identify the reporting of this SupervisionCheckpoint to the Phm.
phmCheckpoint	PhmCheckpoint	0..1	iref	Instance reference to the PhmCheckpoint defined in the context of a PortInterface. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=phmCheckpoint, phm Checkpoint.contextRootSwComponentPrototype, phm Checkpoint.contextComponentPrototype, phm Checkpoint.contextRPortPrototype, phm Checkpoint.targetPhmCheckpoint **InstanceRef implemented by:** PhmCheckpointIn ExecutableInstanceRef
process	Process	0..1	ref	Reference to the Process this checkoint shall be monitored. **Tags:** atp.Splitkey=process

Table 12.20: SupervisionCheckpoint

[constr_1742] Multiplicity of reference in the role SupervisionCheckpoint.phmCheckpoint

Imposition time: IT_Mani

⌈At the time of target-configuration of a SupervisionCheckpoint, one reference to meta-class PhmCheckpoint in the role phmCheckpoint shall exist.⌋

[constr_1764] Counterpart of PhmCheckpoint

Imposition time: IT_Mani

⌈Each PhmCheckpoint shall be referenced once and only once in the role target-PhmCheckpoint by a PhmCheckpointInExecutableInstanceRef with the same Executable and chain of contextComponentPrototype and contextRPort-Prototype that is aggregated by a SupervisionCheckpoint in combination with a specific Process. This reference shall exist.⌋

This means that each SupervisionCheckpoint is the unique representation of the corresponding PhmCheckpoint instance in the context of a PhmCheckpoint-InExecutableInstanceRef and one Process.

The detailed modeling of the ≪InstanceRef≫ from SupervisionCheckpoint to PhmCheckpoint is documented in Chapter D.7.

12.3.3.1 Global supervision

The GlobalSupervision collects PhmSupervisions

- AliveSupervisions,
- LogicalSupervisions,
- DeadlineSupervisions,
- NoCheckpointSupervisions and
- NoSupervisions

and aggregates the individual states of these Alive, Deadline, and Logical Supervisions into one GlobalSupervision status (see also Figure 12.11).

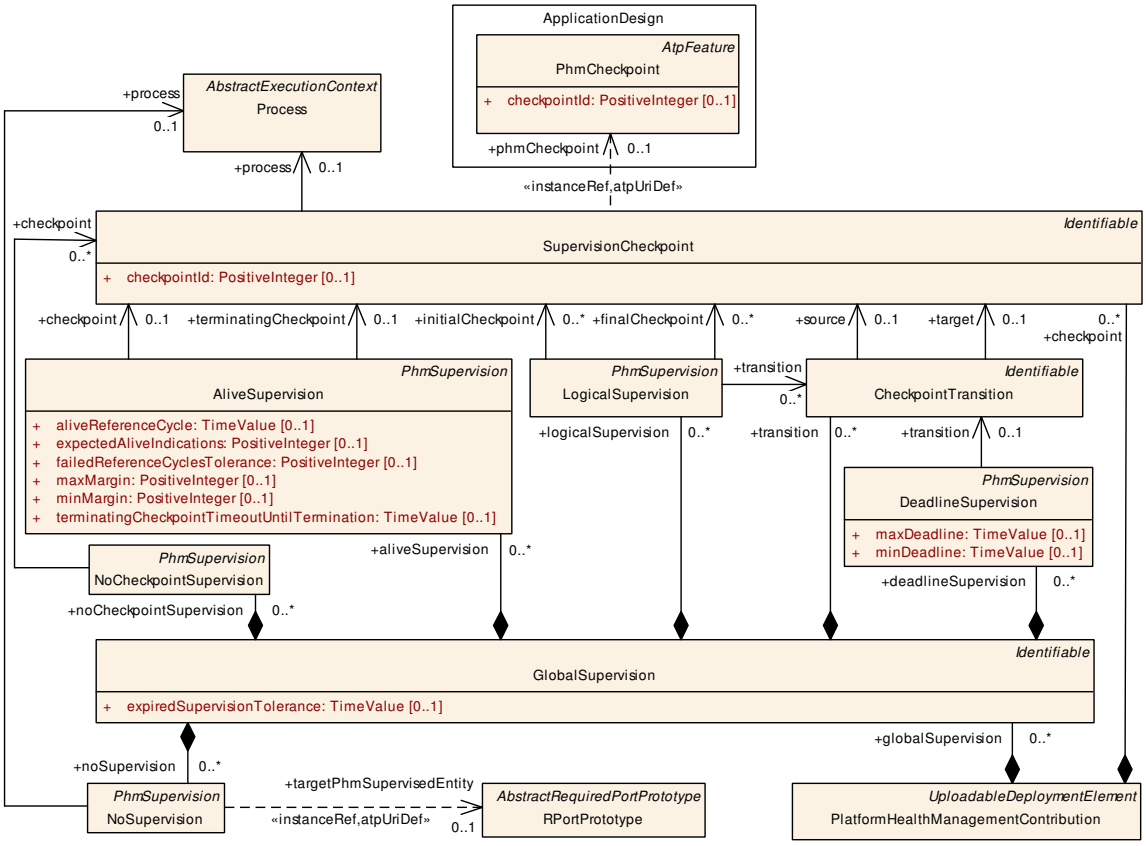


Figure 12.11: Modeling of GlobalSupervision

[constr_3623] Restriction for SupervisionCheckpoints in the context of a GlobalSupervision

Imposition time: IT_Mani

⌈All SupervisionCheckpoints belonging to the same PhmSupervisedEntityInterface instance (SupervisionCheckpoints with identical PhmCheckpointInExecutableInstanceRef.contextRootSwComponentPrototype, contextComponentPrototype, contextRPortPrototype, and

process references) shall only be referenced by PhmSupervisions which are aggregated by the same GlobalSupervision.⌋

[constr_3624] At least one Supervision defined in the context of a GlobalSupervision

Imposition time: IT_Mani

⌈At least one AliveSupervision, LogicalSupervision, or DeadlineSupervision shall be defined in the scope of a GlobalSupervision.⌋

[constr_3823] Existence of primitive attribute GlobalSupervision.expiredSupervisionTolerance

Imposition time: IT_Mani

⌈For each GlobalSupervision the attribute expiredSupervisionTolerance shall exist.⌋

[TPS_MANI_03514] Expiration tolerance for GlobalSupervision ⌈The attribute GlobalSupervision.expiredSupervisionTolerance defines the time which a GlobalSupervision is maintained in the state *expired* before it is considered *stopped*.⌋

[TPS_MANI_03512] Applicability of global supervision without Process context ⌈The aggregated AliveSupervisions, LogicalSupervisions, and DeadlineSupervisions contributing to a specific GlobalSupervision may refer to SupervisionCheckpoints where each SupervisionCheckpoint may refer to a different Process.⌋

With [TPS_MANI_03512] the GlobalSupervision and the aggregation AliveSupervisions, LogicalSupervisions, and DeadlineSupervisions can be used to establish three use-cases:

- compose the status of one Executable instance (Process) if all aggregated PhmSupervisions are defined with the same Process context.
- compose the status of several Executable instances (Processes) of a FunctionGroup if the aggregated PhmSupervisions are defined with (partially) different Process contexts within a FunctionGroup context.
- compose the status of all Executable instances (Processes) of a FunctionGroup if the aggregated PhmSupervisions are defined with (all) different Process contexts within a FunctionGroup context.

[constr_3625] DeadlineSupervision referencing CheckpointTransition in the context of a GlobalSupervision

Imposition time: IT_Mani

⌈DeadlineSupervision aggregated in a GlobalSupervision shall only refer to a CheckpointTransition which is aggregated by the same GlobalSupervision.⌋

[constr_3626] `LogicalSupervision` referencing `CheckpointTransition` in the context of a `GlobalSupervision`

Imposition time: `IT_Mani`

⌈`LogicalSupervision` aggregated in a `GlobalSupervision` shall only refer to `CheckpointTransition`s that are aggregated by the same `GlobalSupervision`.⌋

Class	**GlobalSupervision**			
Note	This element defines a collection of AliveSupervisions, DeadlineSupervisions, and LogicalSupervisions in order to provide an aggregated supervision state. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	PlatformHealthManagementContribution.globalSupervision			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
alive Supervision	AliveSupervision	*	aggr	Collection of AliveSupervisions in the context of this GlobalSupervision. **Tags:** atp.Splitkey=aliveSupervision, alive Supervision.shortName
deadline Supervision	DeadlineSupervision	*	aggr	Collection of DeadlineSupervisions in the context of this GlobalSupervision. **Tags:** atp.Splitkey=deadlineSupervision, deadline Supervision.shortName
expired Supervision Tolerance	TimeValue	0..1	attr	Defines the acceptable amount of time with EXPIRED supervision status before the GlobalSupervision is considered STOPPED.
logical Supervision	LogicalSupervision	*	aggr	Collection of LogicalSupervisions in the context of this GlobalSupervision. **Tags:** atp.Splitkey=logicalSupervision, logical Supervision.shortName
noCheckpoint Supervision	NoCheckpoint Supervision	*	aggr	Definition of No Checkpoint Supervision. **Tags:** atp.Splitkey=noCheckpointSupervision, no CheckpointSupervision.shortName
noSupervision	NoSupervision	*	aggr	Collection of NoSupervisions in the context of this Global Supervision. **Tags:** atp.Splitkey=noSupervision, noSupervision.short Name
supervision Mode	SupervisionMode	*	aggr	Collection of SupervisionModes in the context of this GlobalSupervision. **Tags:** atp.Splitkey=supervisionMode, supervision Mode.shortName
transition	CheckpointTransition	*	aggr	Collection of CheckpointTransitions in the context of this GlobalSupervision. **Tags:** atp.Splitkey=transition, transition.shortName

Table 12.21: GlobalSupervision

12.3.3.2 Supervision Mode

Expected execution (timing or sequence) of the Software can change based on certain conditions. Hence, the value of the Supervision attributes might have to be changed

based on conditions as well. SupervisionMode defines the Supervision attributes which change based on the referenced modeCondition.

The Execution Management may use Function Groups and Function Group States to define the startup conditions of application software. The behavior of the application software can change based on the Function Group State. Hence the monitoring shall also be configurable based on Function Group State changes.

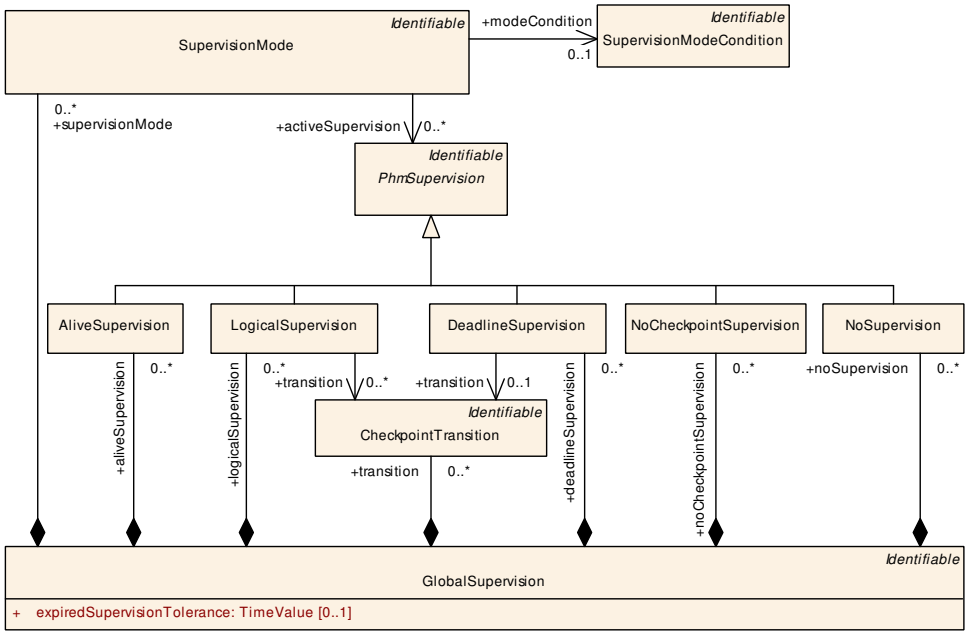


Figure 12.12: Mode dependent supervision

A SupervisionMode does define a set of PhmSupervisions in the role activeSupervision which shall be active while this SupervisionMode is active. This modeling allows for one specific PhmSupervision to be defined as active in several SupervisionModes.

Class	**SupervisionMode**			
Note	This element defines a SupervisionMode. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	GlobalSupervision.supervisionMode			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
active Supervision	PhmSupervision	*	ref	The reference defines which PhmSupervisions shall be active in this specific SupervisionMode. **Tags:** atp.Splitkey=activeSupervision
modeCondition	SupervisionMode Condition	0..1	ref	Reference to SupervisionModeCondition (Condition under which the configuration made under this SupervisionMode are to be applied). **Tags:** atp.Splitkey=modeCondition

Table 12.22: SupervisionMode

[constr_3640] Existence of SupervisionMode.modeCondition

Imposition time: IT_Mani

⌈For each SupervisionMode the attribute modeCondition shall exist.⌋

[TPS_MANI_03513] Collection of supervisions into a GlobalSupervision ⌈The PhmSupervisions referenced from a SupervisionMode in the role activeSupervision are taken into account to evaluate the supervision status of the GlobalSupervision in that particular SupervisionMode.⌋

[TPS_MANI_03633] Semantics of several supervisionMode references for one supervision ⌈If a PhmSupervision is referenced in the role activeSupervision then this PhmSupervision will be active while the referencing SupervisionMode is active.⌋

12.3.3.3 Supervision Mode condition

Each SupervisionMode refers to exactly one SupervisionModeCondition in the role modeCondition. The SupervisionModeCondition defines an collection of one or several stateReferences which are combined to provide a single logical state.

The reference to Function Groups and their Function Group States is essential for the definition of SupervisionModeConditions as the supervised Processes are started and stopped according to changes in Function Group States.

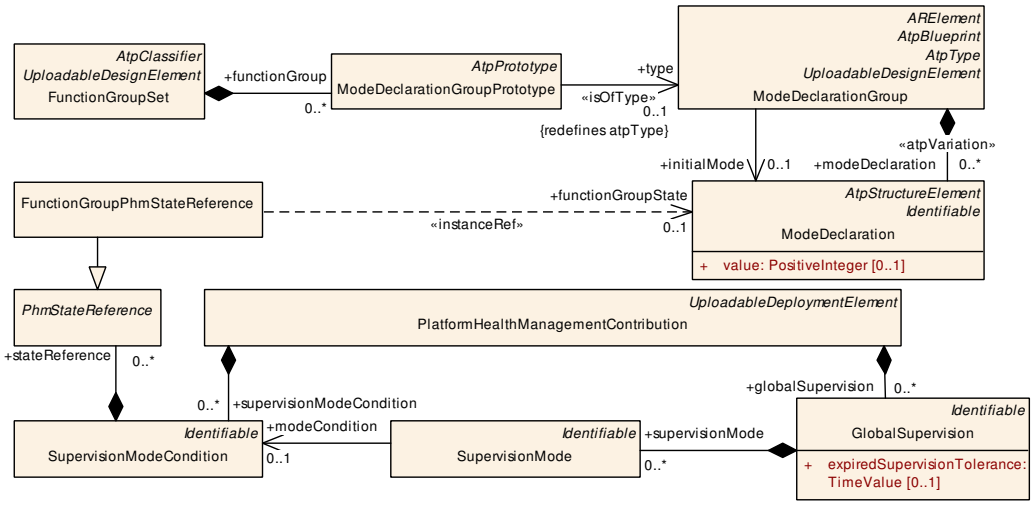


Figure 12.13: Modeling of SupervisionMode

Class	FunctionGroupPhmStateReference			
Note	Function Group state dependency. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *PhmStateReference*			
Aggregated by	SupervisionModeCondition.stateReference			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
functionGroup State	ModeDeclaration	0..1	iref	This represent the applicable functionGroupState. **Tags:** atp.Splitkey=functionGroupState, functionGroup State.contextModeDeclarationGroupPrototype, function GroupState.targetModeDeclaration **InstanceRef implemented by:** FunctionGroupStateIn FunctionGroupSetInstanceRef

Table 12.23: FunctionGroupPhmStateReference

Class	SupervisionModeCondition			
Note	This element defines a SupervisionModeCondition in the context of platform health management contribution. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	PlatformHealthManagementContribution.supervisionModeCondition			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
stateReference	PhmStateReference	*	aggr	Collection of stateReferences. **Tags:** atp.Splitkey=stateReference

Table 12.24: SupervisionModeCondition

[constr_3627] Existence of SupervisionModeCondition.stateReference

Imposition time: IT_Mani

⌈At the time of target-configuration of SupervisionModeCondition, at least one aggregation of PhmStateReference in the role SupervisionModeCondition.stateReference shall exist.⌋

[constr_3628] Reference to Function Group State from a SupervisionModeCondition

Imposition time: IT_Mani

⌈If a Function Group State is referenced by a SupervisionModeCondition in the scope of one GlobalSupervision, then that same Function Group State shall NOT be referenced by any other SupervisionModeCondition in the scope of the same GlobalSupervision.⌋

Rationale for [constr_3628]: It is guaranteed that only one of the SupervisionModes is active at any point in time. The SupervisionModeConditions referenced under a GlobalSupervision in the context of SupervisionMode shall be mutually exclusive.

E.g. Following SupervisionModeConditions referenced under the same GlobalSupervision in the context of SupervisionMode is not allowed:

- SupervisionModeCondition_1:
  (FunctionGroup_X in State ‘a’) or (FunctionGroup_X in State ‘b’)
- SupervisionModeCondition_2:
  (FunctionGroup_X in State ‘a’) or (FunctionGroup_X in State ‘c’)

In the above example, (FunctionGroup_X in State ‘a’) is common between the 2 SupervisionModeConditions.

[constr_3629] Identical Function Group in the scope of a GlobalSupervision

Imposition time: IT_Mani

⌈Within the context of one GlobalSupervision, all SupervisionMode.modeCondition shall only aggregate FunctionGroupPhmStateReferences in the role stateReference where the reference FunctionGroupPhmStateReference.functionGroupState.contextModeDeclarationGroupPrototype refers to the identical ModeDeclarationGroupPrototype (that implements the Function Group, as far as state management is concerned).⌋

That is, within the context of one GlobalSupervision, all SupervisionMode.modeCondition.stateReference shall reference states(modes) of the same Function Group.

[constr_3630] GlobalSupervision and Process relation

Imposition time: IT_Mani

⌈Within the context of one GlobalSupervision, all aggregated PhmSupervisions shall refer to SupervisionCheckpoints where the referenced Process aggregates a stateDependentStartupConfig that in turn refers to a functionGroupState where the contextModeDeclarationGroupPrototype refers to the identical ModeDeclarationGroupPrototype (that implements the Function Group, as far as state management is concerned).⌋

That is, all PhmSupervisions aggregated in a GlobalSupervision are referencing SupervisionCheckpoints corresponding to Processes that are configured to be executed in state/s of the same Function Group.

[constr_3631] Global supervision restricted to one Function Group

Imposition time: IT_Mani

⌈The Function Group (ModeDeclarationGroupPrototype) referenced in [constr_3629] and [constr_3630] shall be identical for one particular GlobalSupervision.⌋

[constr_3632] Supervision of a Supervised Entity Instance in the scope of a Function Group State

Imposition time: IT_Mani

⌈A Supervised Entity Instance shall be configured with checkpoint supervision (all SupervisionCheckpoints of the Supervised Entity Instance are covered by AliveSupervision, DeadlineSupervision, LogicalSupervision, NoCheckpointSupervision) or NoSupervision in all Function Group States in which the corresponding Process is configured to be executed.⌋

Rationale for [constr_3632]: The configuration of supervision for a specific Supervised Entity Instance is not missed unintentionally in a Function Group State.

[constr_3632] requires the Supervised Entity Instance to be part of the supervision definition in all Function Group States in which the corresponding Process is configured to be executed.

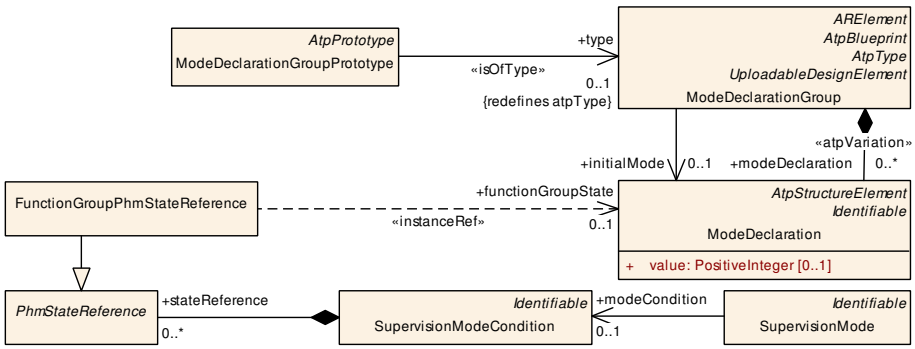


Figure 12.14: Modeling of FunctionGroupPhmStateReference

If there is no supervision required in a specific Function Group State (because the Supervised Entity Instance does not need to be supervised in that specific Function Group State), then NoSupervision can be configured for that Supervised Entity Instance.

If a Process survives a Function Group State change running (because the Process is defined to be running in two consecutive Function Group States), then the configuration of the activeSupervision for the Supervised Entity Instances corresponding to the Process is identical for the SupervisionModes that corresponds to the affected Function Group States.

This relations can not be formally expressed (because the possible Function Group State transitions are not formalized) and therefore this restriction can not be defined as a constraint.

A possible mitigation strategy is to define different Process arguments and therefore force a re-start of the Process, even if the Function Group States have been switched consecutively.

12.3.3.4 NoSupervision definition

In the scope of a GlobalSupervision, a NoSupervision can be defined for a Supervised Entity Instance as a whole. A NoSupervision can be configured to indicate that this Supervised Entity Instance shall not be monitored by the Phm in a specific SupervisionMode.

The Supervised Entity Instance is identified using the NoSupervision.targetPhmSupervisedEntity instance ref and NoSupervision.process reference.

Class	**NoSupervision**			
Note	Defines explicitly that NO supervision shall be applied for a specific Supervised Entity instance. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhmSupervision*, *Referrable*			
Aggregated by	GlobalSupervision.noSupervision			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
process	Process	0..1	ref	Reference to the Process this NoSupervision applies to. **Tags:** atp.Splitkey=process
targetPhm SupervisedEntity	RPortPrototype	0..1	iref	Instance reference to the RPortPrototype which represents the Supervised Entity instance. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=targetPhmSupervisedEntity, targetPhmSupervisedEntity.contextRootSwComponentPrototype, targetPhmSupervisedEntity.contextComponentPrototype, targetPhmSupervisedEntity.targetRPortPrototype **InstanceRef implemented by:** RPortPrototypeInExecutableInstanceRef

Table 12.25: NoSupervision

[constr_3674] Existence of NoSupervision.targetPhmSupervisedEntity

Imposition time: IT_Mani

⌈For each NoSupervision the attribute targetPhmSupervisedEntity shall exist.⌋

[constr_3675] Existence of NoSupervision.process

Imposition time: IT_Mani

⌈For each NoSupervision the attribute process shall exist.⌋

[constr_3676] Exclusive usage of NoSupervision

Imposition time: IT_Mani

⌈For a Supervised Entity Instance, in a given SupervisionMode, either a checkpoint supervision (AliveSupervision, DeadlineSupervision, LogicalSupervision, and/or NoCheckpointSupervision referencing all SupervisionCheckpoints corresponding to the Supervised Entity Instance) or NoSupervision shall be configured, but not both.⌋

The NoSupervision approach disables the supervision for all the SupervisionCheckpoints of a Supervised Entity Instance.

12.3.3.5 AliveSupervision definition

In the scope of a GlobalSupervision an AliveSupervision can be defined for a specific SupervisionCheckpoint. AliveSupervision can be used to define in which timing boundaries one specific checkpoint shall be monitored.

[TPS_MANI_03508] Definition of an AliveSupervision for a SupervisionCheckpoint ⌈An AliveSupervision definition provides attributes to configure the supervision of the referenced SupervisionCheckpoint.

- aliveReferenceCycle defines the time base used to monitor the reporting of this specific SupervisionCheckpoint
- expectedAliveIndications defines the number of indications which shall be observed during the time period defined by aliveReferenceCycle
- minMargin and maxMargin define the acceptable deviation from the expectedAliveIndications within the time period defined by aliveReferenceCycle

⌋

[TPS_MANI_03575] Definition of no minimum alive supervision ⌈If the value AliveSupervision.minMargin equals 0, this defines that no minimum alive supervision shall be performed.⌋

[TPS_MANI_03576] Definition of no maximum alive supervision ⌈If the value AliveSupervision.maxMargin equals INF, this defines that no maximum alive supervision shall be performed.⌋

[constr_3539] Only one AliveSupervision per SupervisionCheckpoint

Imposition time: IT_Mani

⌈A SupervisionCheckpoint shall only be referenced up to once by an AliveSupervision in the role checkpoint in the context of an identical SupervisionMode.⌋

[TPS_MANI_03515] Failure tolerance for AliveSupervision ⌈The attribute AliveSupervision.failedReferenceCyclesTolerance defines the acceptable amount of AliveSupervision.aliveReferenceCycles with incorrect/failed AliveSupervision.⌋

Class	AliveSupervision
Note	Defines an AliveSupervision for one checkpoint. This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhmSupervision*, *Referrable*
Aggregated by	GlobalSupervision.aliveSupervision

▽

△

Class	**AliveSupervision**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
aliveReference Cycle	TimeValue	0..1	attr	Time period at which the Alive Supervision mechanism compares the amount of received Alive Indications for the SupervisionCheckpoint against the expectedAlive Indications.
checkpoint	SupervisionCheckpoint	0..1	ref	Reference to a checkpoint in the context of Alive Supervision. **Tags:** atp.Splitkey=checkpoint
expectedAlive Indications	PositiveInteger	0..1	attr	Defines the amount of expected Alive Indications of the SupervisionCheckpoint within the aliveReferenceCycle.
failedReference Cycles Tolerance	PositiveInteger	0..1	attr	This attribute defines the acceptable amount of alive ReferenceCycles with incorrect/failed AliveSupervision.
maxMargin	PositiveInteger	0..1	attr	Defines the amount of Alive Indications of the Supervision Checkpoint that are acceptable to be additional to the expectedAliveIndications within the aliveReferenceCycle.
minMargin	PositiveInteger	0..1	attr	Defines the amount of Alive Indications of the Supervision Checkpoint that are acceptable to be missing to the expectedAliveIndications within the aliveReferenceCycle.
terminating Checkpoint	SupervisionCheckpoint	0..1	ref	Reference to the SupervisionCheckpoint which is defined as the terminating checkpoint of this AliveSupervision. **Tags:** atp.Splitkey=terminatingCheckpoint
terminating Checkpoint TimeoutUntil Termination	TimeValue	0..1	attr	Defines the time a process shall terminate after it has announced its start of termination by reporting terminatingCheckpoint.

Table 12.26: AliveSupervision

[constr_3633] Mandatory attributes of `AliveSupervision`

Imposition time: `IT_Mani`

⌈The following attributes of `AliveSupervision` shall be defined:

- `aliveReferenceCycle`
- `checkpoint`
- `expectedAliveIndications`
- `failedReferenceCyclesTolerance`
- `minMargin`
- `maxMargin`

⌋

12.3.3.5.1 `AliveSupervision` of self-terminating processes

If a `Process` is defined to be self-terminating (according to [TPS_MANI_01334]) and that `Process` has an `AliveSupervision` defined, then it is essential for that `Process` to indicate to the `Phm` when that `Process` is about to self terminate.

The indication is done using a dedicated SupervisionCheckpoint and is associated with the AliveSupervision. In this case the Phm can stop the AliveSupervision in a defined way when the application reports that AliveSupervision.terminatingCheckpoint has been reached.

[constr_3709] AliveSupervision.terminatingCheckpoint required for self terminating Processes

Imposition time: IT_Mani

⌈Only if a Process

- aggregates a StateDependentStartupConfig, and that StateDependentStartupConfig has the attribute StateDependentStartupConfig.terminationBehavior set to the value TerminationBehaviorEnum.processIsSelfTerminating, and
- the StateDependentStartupConfig.functionGroupState is identical to the respective GlobalSupervision.supervisionMode.modeCondition.stateReference.functionGroupState and
- there exists an AliveSupervision which refers to a SupervisionCheckpoint in the role checkpoint, and that SupervisionCheckpoint refers to that Process, then

there shall exist an AliveSupervision.terminatingCheckpoint reference from the AliveSupervision.⌋

That is, if an AliveSupervision is defined for a Process in a SupervisionMode that corresponds to a Function Group State in which the Process is configured to be self-terminating, then and only then AliveSupervision.terminatingCheckpoint shall be defined.

[constr_3710] Process referenced by AliveSupervision.terminatingCheckpoint

Imposition time: IT_Mani

⌈The SupervisionCheckpoint that is referenced in the role AliveSupervision.terminatingCheckpoint shall refer to the same Process as the SupervisionCheckpoint that is referenced by the AliveSupervision.checkpoint.⌋

[constr_3711] AliveSupervision.terminatingCheckpointTimeoutUntilTermination

Imposition time: IT_Mani

⌈If an AliveSupervision has the reference AliveSupervision.terminatingCheckpoint defined, then the attribute AliveSupervision.terminatingCheckpointTimeoutUntilTermination shall be defined.⌋

12.3.3.6 CheckpointTransition definition

For the definition of further supervision strategies the need to first define possible CheckpointTransitions between SupervisionCheckpoints arises. Since the application software design does not provide any transition definition between checkpoints, it is essential to define possible CheckpointTransitions.

The definition of CheckpointTransitions is done in the scope of the GlobalSupervision and can be used by the LogicalSupervision and DeadlineSupervision of the same GlobalSupervision.

[TPS_MANI_03509] Definition of a CheckpointTransition ⌈A CheckpointTransition defines one possible transition from the source SupervisionCheckpoint to the target SupervisionCheckpoint.⌋

[constr_3634] Multiplicity of CheckpointTransition.source and CheckpointTransition.target

Imposition time: IT_Mani

⌈Each CheckpointTransition shall define exactly one source reference and one target reference.⌋

Class	**CheckpointTransition**			
Note	Defines one transition between two checkpoints. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	GlobalSupervision.transition			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
source	SupervisionCheckpoint	0..1	ref	Reference to the source checkpoint for this transition. **Tags:** atp.Splitkey=source
target	SupervisionCheckpoint	0..1	ref	Reference to the target checkpoint for this transition. **Tags:** atp.Splitkey=target

Table 12.27: CheckpointTransition

Any CheckpointTransition may be used in either a LogicalSupervision, a DeadlineSupervision, or both. This is illustrated in Figure 12.15: The example shows 5 SupervisionCheckpoints and several CheckpointTransitions.

The CheckpointTransitions *t23* and *t34* are only used for the definition of the LogicalSupervision between the related SupervisionCheckpoints.

The CheckpointTransitions *t12* and *t45* are used for the definition of the LogicalSupervision between the related SupervisionCheckpoints as well as for the definition the DeadlineSupervisions on these CheckpointTransitions.

The CheckpointTransition *t24* is only used for the definition the DeadlineSupervision between the CheckpointTransitions. From a LogicalSupervision perspective it is not allowed to transition between *cp2* and *cp4* directly, but the total time for the transition chain *cp2* to *cp4* via *cp3* is set to take at most 8ms.

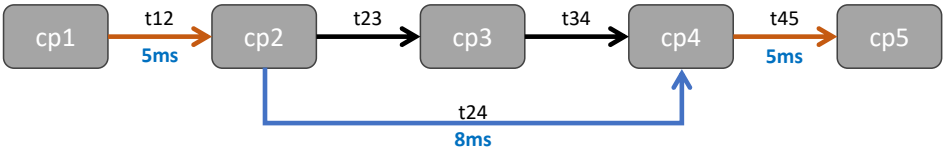


Figure 12.15: Usage of `CheckpointTransition` in `LogicalSupervision` and `DeadlineSupervision`

12.3.3.7 `LogicalSupervision` definition

The `LogicalSupervision` defines a supervision graph of allowed `CheckpointTransition`s which is monitored by the Platform Health Management without any timing considerations, just the order of reported checkpoints is considered for the monitoring.

[constr_3540] `SupervisionCheckpoint` in supervision graph

Imposition time: `IT_Mani`

⌈Each `SupervisionCheckpoint` shall only be part of one supervision graph in the context of an identical `SupervisionMode`.⌋

When a `SupervisionCheckpoint` belonging to the supervision graph is reported to the Platform Health Management where there is no `CheckpointTransition` defined from the last reported `SupervisionCheckpoint` as `source` to the current reported `SupervisionCheckpoint` as `target`, this situation violates the `LogicalSupervision`.

[TPS_MANI_03510] Definition of `LogicalSupervision` ⌈A `LogicalSupervision` defines relations between `SupervisionCheckpoint`s which form a directed graph from one or more `initialCheckpoint` `SupervisionCheckpoint`s through a set of `CheckpointTransition`s defined by collection of `transition`s to one or more `finalCheckpoint` `SupervisionCheckpoint`s.⌋

Class	**LogicalSupervision**			
Note	Defines a LogicalSupervision graph consisting of transitions, initial- and final checkpoints. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhmSupervision*, *Referrable*			
Aggregated by	GlobalSupervision.logicalSupervision			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
finalCheckpoint	SupervisionCheckpoint	*	ref	Reference to the final Checkpoint(s) for this Logical Supervision. **Tags:** atp.Splitkey=finalCheckpoint xml.sequenceOffset=20

▽

△

Class	**LogicalSupervision**			
initialCheckpoint	SupervisionCheckpoint	*	ref	Reference to the initial Checkpoint(s) for this Logical Supervision. **Tags:** atp.Splitkey=initialCheckpoint xml.sequenceOffset=10
transition	CheckpointTransition	*	ref	Reference to the transitions for this LogicalSupervision. **Tags:** atp.Splitkey=transition xml.sequenceOffset=30

Table 12.28: LogicalSupervision

[constr_1736] Multiplicity of the reference in the role `LogicalSupervision.initialCheckpoint`

Imposition time: `IT_Mani`

⌈At the time of target-configuration of a `LogicalSupervision`, at least one reference to meta-class `SupervisionCheckpoint` in the role `initialCheckpoint` shall exist.⌋

[constr_1737] Multiplicity of the reference in the role `LogicalSupervision.finalCheckpoint`

Imposition time: `IT_Mani`

⌈At the time of target-configuration of a `LogicalSupervision`, at least one reference to meta-class `SupervisionCheckpoint` in the role `finalCheckpoint` shall exist.⌋

[constr_1740] Multiplicity of the reference in the role `LogicalSupervision.transition`

Imposition time: `IT_Mani`

⌈At the time of target-configuration of a `LogicalSupervision`, at least one reference to meta-class `CheckpointTransition` in the role `LogicalSupervision.transition` shall exist.⌋

12.3.3.8 `DeadlineSupervision` definition

The `DeadlineSupervision` defines timing attributes for one specific `CheckpointTransition`.

[TPS_MANI_03511] Definition of a `DeadlineSupervision` ⌈A `DeadlineSupervision` defines timing attributes which are monitored by the Platform Health Management for one specific `CheckpointTransition`.⌋

[constr_3635] Mandatory attributes of `DeadlineSupervision`

Imposition time: `IT_Mani`

⌈The following attributes of `DeadlineSupervision` shall be defined:

- transition
- minDeadline
- maxDeadline

⌋

Class	DeadlineSupervision			
Note	Defines an DeadlineSupervision for one transition. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhmSupervision*, *Referrable*			
Aggregated by	GlobalSupervision.deadlineSupervision			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
maxDeadline	TimeValue	0..1	attr	Defines the longest time span before which the deadline is considered to be met for transition.
minDeadline	TimeValue	0..1	attr	Defines the shortest time span after which the deadline is considered to be met for transition.
transition	CheckpointTransition	0..1	ref	Reference to the transition in the context of a Deadline Supervision. **Tags:** atp.Splitkey=transition

Table 12.29: DeadlineSupervision

[TPS_MANI_03573] Definition of no minimum deadline supervision ⌈If the value DeadlineSupervision.minDeadline equals 0, this defines that no minimum deadline supervision shall be performed.⌋

[TPS_MANI_03574] Definition of no maximum deadline supervision ⌈If the value DeadlineSupervision.maxDeadline equals INF, this defines that no maximum deadline supervision shall be performed.⌋

12.3.3.9 NoCheckpointSupervision definition

The NoCheckpointSupervision is used to define a set of SupervisionCheckpoints for which no Alive, Deadline and Logical supervision shall be performed. It is a means to define a complete coverage of the configuration with respect to the possibly reported SupervisionCheckpoints.

If a set of SupervisionCheckpoints shall not be considered by the Phm for Supervision (Alive, Deadline and Logical) evaluation, then these SupervisionCheckpoints can be referenced from the NoCheckpointSupervision in the role NoCheckpointSupervision.checkpoint.

[TPS_MANI_03670] Definition of meta-class NoCheckpointSupervision ⌈A NoCheckpointSupervision defines explicitly that no supervision (Alive, Deadline and Logical) shall be applied to the SupervisionCheckpoints which are referenced in the role NoCheckpointSupervision.checkpoint.⌋

[constr_3712] Exclusive usage of NoCheckpointSupervision

Imposition time: IT_Mani

⌈If a SupervisionCheckpoint is referenced by a NoCheckpointSupervision in the role checkpoint, then that SupervisionCheckpoint shall not be referenced by any other checkpoint supervision (AliveSupervision, DeadlineSupervision (via CheckpointTransition), LogicalSupervision (also via or CheckpointTransition)) in the scope of one SupervisionMode.⌋

Class	**NoCheckpointSupervision**			
Note	Defines explicitly that NO supervision shall be applied for a set of SupervisionCheckpoints. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhmSupervision*, *Referrable*			
Aggregated by	GlobalSupervision.noCheckpointSupervision			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checkpoint	SupervisionCheckpoint	*	ref	Reference to the set of SupervisionCheckpoints which shall not be considered for any kind of supervision. **Tags:** atp.Splitkey=checkpoint

Table 12.30: NoCheckpointSupervision

12.3.4 Supervision Recovery Notification Target-Configuration

The only purpose of meta-classes HealthChannel and HealthChannelSupervision is to **provide a mapping** between a RecoveryNotification and a GlobalSupervision, as shown in Figure 12.16. Beside this limited functionality, HealthChannel and HealthChannelSupervision carry **no further semantics**.

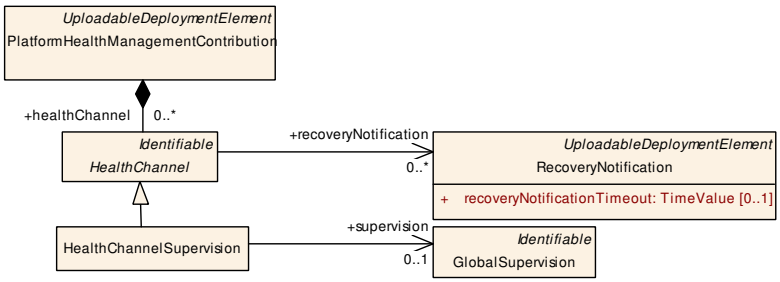


Figure 12.16: Modeling of HealthChannelSupervision

[constr_3730] Upper multiplicity of reference in the role HealthChannel.recoveryNotification

Imposition time: IT_Mani

⌈In the context of HealthChannel, the reference in the role recoveryNotification shall exist at most once.⌋

Class	**HealthChannel** (abstract)			
Note	This element defines the source of a health channel. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	HealthChannelSupervision			
Aggregated by	PlatformHealthManagementContribution.healthChannel			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
recovery Notification	RecoveryNotification	*	ref	Defines the RecoveryNotification for this HealthChannel. **Tags:** atp.Splitkey=recoveryNotification

Table 12.31: HealthChannel

[TPS_MANI_03672] Definition of `RecoveryNotification` for an `HealthChannel` ⌈The reference `HealthChannel.recoveryNotification` defines which `RecoveryNotification` shall be triggered if the `HealthChannel` is violated.⌋

[TPS_MANI_03673] `RecoveryNotification` referenced by several `HealthChannel.recoveryNotification`s ⌈One `RecoveryNotification` may be referenced by several `HealthChannel.recoveryNotification`s defining that the violation of any of the referencing `HealthChannel`s will trigger that `RecoveryNotification`.⌋

The `HealthChannelSupervision` is used to take the status of a `GlobalSupervision` trigger the `RecoveryNotification` if a violation is detected by the Phm.

[TPS_MANI_03516] Status for `HealthChannelSupervision` ⌈The status of the `GlobalSupervision` which is referenced in the role `supervision` is taken as the trigger for the `RecoveryNotification`.⌋

Class	**HealthChannelSupervision**			
Note	This element defines a health channel representing the status of a PhmSupervision. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *HealthChannel*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	PlatformHealthManagementContribution.healthChannel			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
supervision	GlobalSupervision	0..1	ref	Reference to the GlobalSupervision as source for the health channel. **Tags:** atp.Splitkey=supervision

Table 12.32: HealthChannelSupervision

12.3.5 Recovery Notification

If Phm detects a Supervision violation then the associated `RecoveryNotification` at the `HealthChannel` is activated. This general setup is illustrated in Figure 4.28. Via the `RecoveryNotificationToPPortPrototypeMapping` a

RecoveryNotification is mapped to a dedicated PPortPrototype in the context of a dedicated Process implementing the State Management functionality.

[constr_3612] Multiplicity of references recoveryNotification, recoveryAction, and process at RecoveryNotificationToPPortPrototypeMapping

Imposition time: IT_Mani

⌈The references recoveryNotification, recoveryAction, and process shall be defined for each RecoveryNotificationToPPortPrototypeMapping.⌋

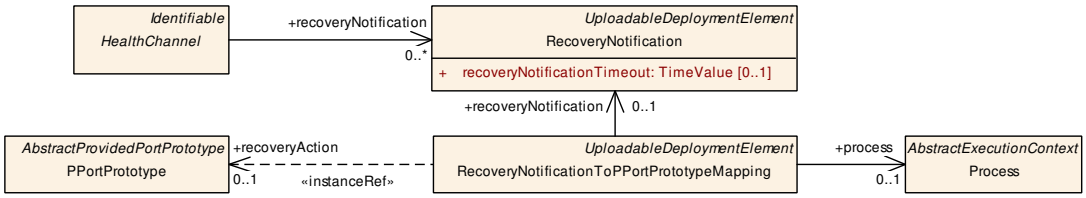


Figure 12.17: Modeling of RecoveryNotification

[constr_3613] Reference to a PhmSupervisionRecoveryNotificationInterface in the context of a HealthChannelSupervision

Imposition time: IT_Mani

⌈If the RecoveryNotification is aggregated by a HealthChannelSupervision then the RecoveryNotificationToPPortPrototypeMapping shall refer to a PPortPrototype in the role recoveryAction typed by PhmSupervisionRecoveryNotificationInterface.⌋

Class	**RecoveryNotification**			
Note	This meta-class represents a PHM action that can trigger a recovery operation inside a piece of State Management software. **Tags:** atp.recommendedPackage=RecoveryNotifications This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
recovery Notification Timeout	TimeValue	0..1	attr	The maximum acceptable amount of time (in seconds), Platform Health Management waits for an acknowledgement by State Management after sending the notification.

Table 12.33: RecoveryNotification

Class	**RecoveryNotificationToPPortPrototypeMapping**
Note	This meta-class represents the ability to associate a RecoveryNotification to a PPortPrototype while also being able to identify the respective Process in which the actual recovery executes. **Tags:** atp.recommendedPackage=RecoveryNotificationMappings This Class is only used by the AUTOSAR Adaptive Platform.

▽

△

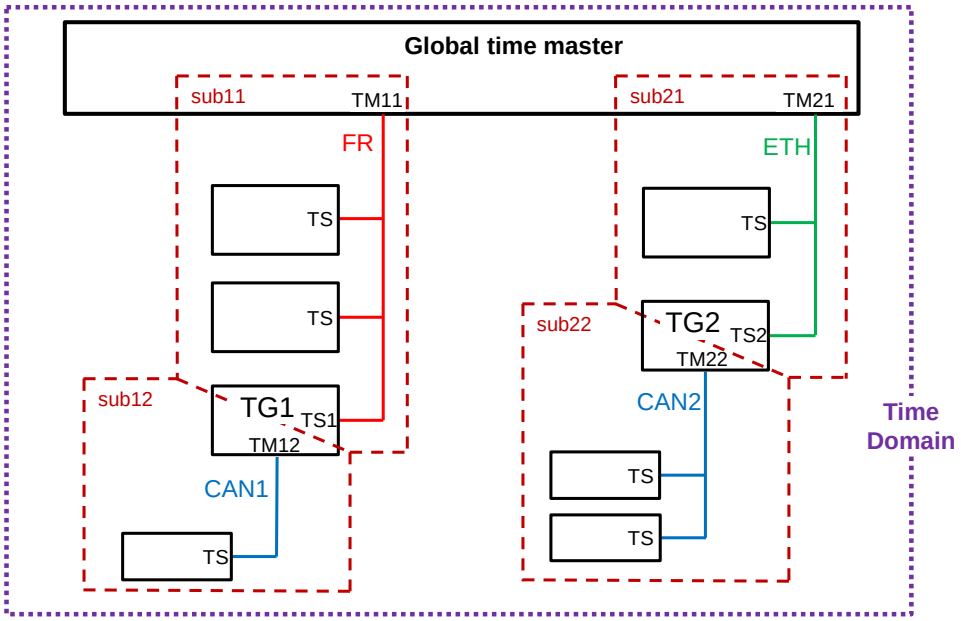
Class	RecoveryNotificationToPPortPrototypeMapping			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, Packageable Element, Referrable, UploadableDeploymentElement, UploadablePackageElement			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
process	Process	0..1	ref	Reference to the process which represents the State Management instance that the recovery notification shall be applied to. **Tags:** atp.Splitkey=process
recoveryAction	PPortPrototype	0..1	iref	This reference identifies the PortPrototype to be addressed as part of a PHM recovery. **Tags:** atp.Splitkey=recoveryAction, recovery Action.contextRootSwComponentPrototype, recovery Action.contextComponentPrototype, recovery Action.targetPPortPrototype **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
recovery Notification	RecoveryNotification	0..1	ref	This reference identifies the applicable Recovery Notification to be mapped. **Tags:** atp.Splitkey=recoveryNotification

Table 12.34: RecoveryNotificationToPPortPrototypeMapping

12.4 Time Synchronization Target-Configuration

12.4.1 Overview

This chapter explains the configuration of the Time Synchronization functional cluster.



TS: GlobalTime slave
TM: GlobalTime master
TG: GlobalTime gateway

Figure 12.18: Example setup of Synchronized Global Time in AUTOSAR

An adaptive AUTOSAR application can utilize several (synchronized) Time-Base Resources which are provided by the Time Synchronization functional cluster, see [17, SWS Time Synchronization].

The intended interaction of an adaptive AUTOSAR application with Time Synchronization is described in Chapter 4.8.

Since an adaptive `Machine` is usually collaborating with other `Machine`s (adaptive) and ECUs (classic), special focus has been put on the vehicle-wide definition of synchronized time.

For a detailed specification, please refer to the *Global Time Synchronization* chapter in the [4, TPS System Template].

CAUTION

The constraints marked with the imposition time `IT_SysDes` defined in chapter [4, TPS System Template] 9 *"Global Time Synchronization"* are valid for this specification as well.

Figure 12.18 provides an example system view on time domains and their transportation over diverse networks. In the scope of the *AUTOSAR adaptive platform* the focus is put on the Ethernet interaction with the rest of the system.

12.4.2 Time Synchronization Functional Cluster Configuration

The representation of the Time Synchronization functional cluster(see [17, SWS Time Synchronization]) within one specific `Machine` is defined by the `TimeSyncModuleInstantiation`.

The `Machine` has the ability to define a set of `moduleInstantiation`s, where a specialization can be the `TimeSyncModuleInstantiation`.

Class	**TimeSyncModuleInstantiation**			
Note	This meta-class defines the attributes for the Time Synchronization configuration on a specific machine. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
timeBase	TimeBaseResource	*	aggr	This aggregation defines the configured Time Bases for Time Synchronization. **Tags:** atp.Splitkey=timeBase, timeBase.shortName

Table 12.35: TimeSyncModuleInstantiation

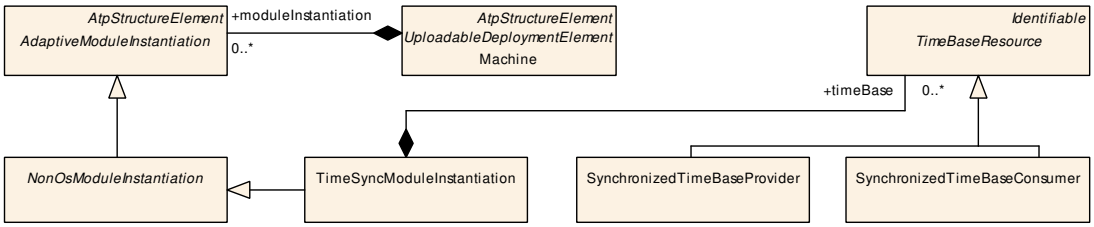


Figure 12.19: Modeling of `TimeSyncModuleInstantiation`

12.4.3 Time Base

The `TimeSyncModuleInstantiation` represents the actual instance of the Time Synchronization functional cluster executed on a specific `Machine`. In the scope of the `TimeSyncModuleInstantiation` the Time-Base Resources are defined.

[TPS_MANI_03539] Definition of Time-Base Resources ⌈The meta-class `TimeSyncModuleInstantiation` has the ability to define a set of Time-Base Resources of kind `TimeBaseResource` in the role `timeBase`.⌋

There are several sub types of `TimeBaseResource` which will be explained in the following sections.

Class	***TimeBaseResource*** (abstract)			
Note	This meta-class represents the attributes of one Time Base Resource for Time Synchronization. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	SynchronizedTimeBaseConsumer, SynchronizedTimeBaseProvider			
Aggregated by	TimeSyncModuleInstantiation.timeBase			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 12.36: TimeBaseResource

12.4.3.1 Synchronized Time Base

When configuring a synchronized time base many configuration aspects are already provided by the definition of the `GlobalTimeDomain` and are specified in the [4, TPS System TemplateS] and associated to `MachineDesign`.

As for the configuration of the `TimeSyncModuleInstantiation` the usage of the `SynchronizedTimeBaseProvider` respectively `SynchronizedTimeBaseConsumer` defines the interaction with the `GlobalTimeDomain`.

[TPS_MANI_03541] Definition of `SynchronizedTimeBaseConsumer` ⌈The meta-class `SynchronizedTimeBaseConsumer` defines a Time Base which is synchronized with a time coming from the network. With the reference `SynchronizedTimeBaseConsumer.networkTimeConsumer` to a `GlobalTimeSlave` the relation to the system model is established.⌋

[TPS_MANI_03542] Definition of `SynchronizedTimeBaseProvider` ⌈The meta-class `SynchronizedTimeBaseProvider` defines a Time Base which is propagated to a time on the network. With the reference `SynchronizedTimeBaseProvider.networkTimeProvider` to a `GlobalTimeMaster` the relation to the system model is established.⌋

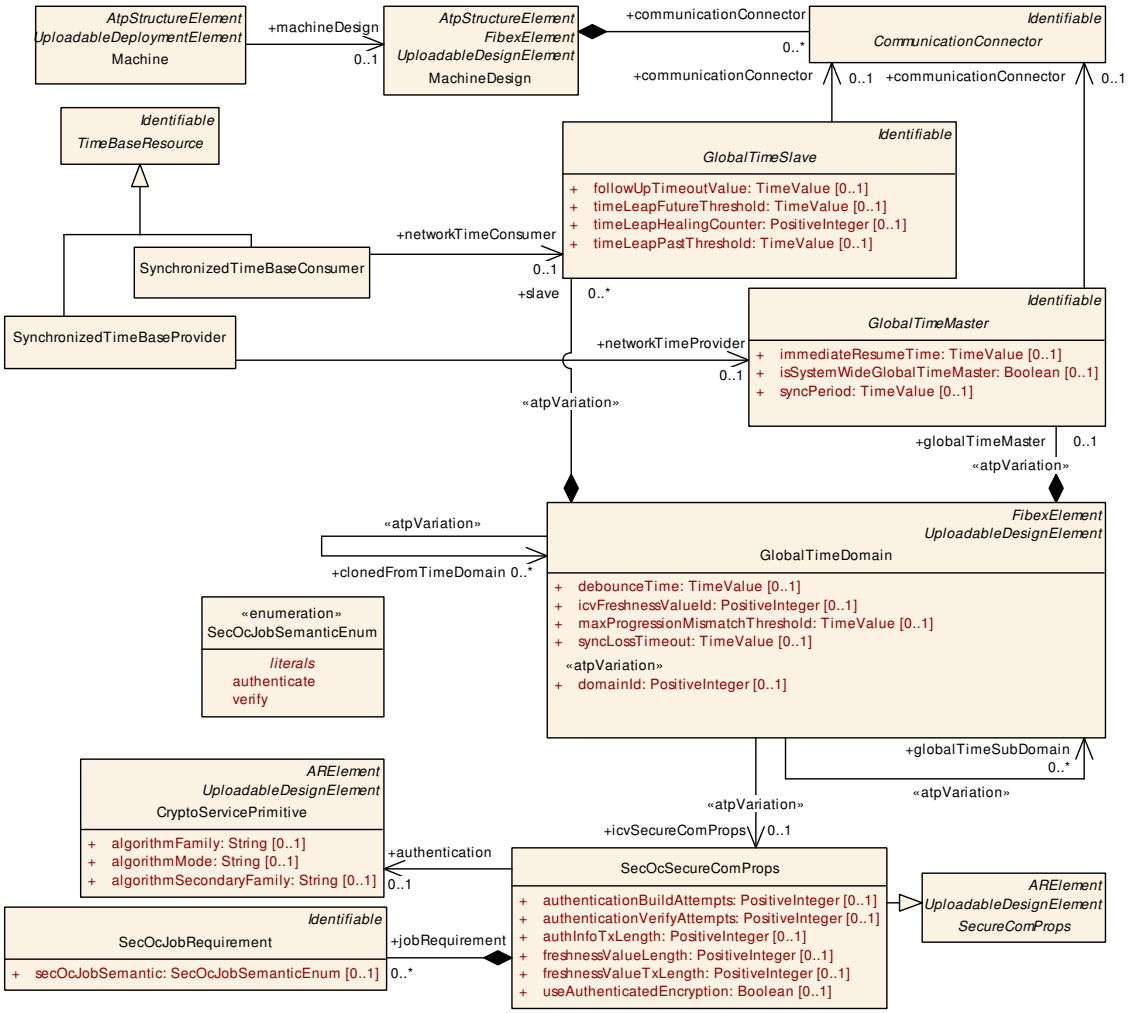


Figure 12.20: Modeling of synchronized time bases

It is possible to define that a `GlobalTimeDomain` is secured for the transport using the Integrity Check Value (ICV). This requires a configuration of the properties used for the calculation and verification of the ICV.

For a detailed specification please refer to the Global Time Synchronization chapter in the [4, TPS System Template].

[TPS_MANI_03543] Definition of time sync correction attributes ⌈`GlobalTimeCorrectionProps` aggregated by the `GlobalTimeDomain` define the attributes for time correction. The attributes defined in [TPS_SYST_02436] for the `GlobalTimeMaster` are applicable for the `SynchronizedTimeBaseProvider`. The attributes defined in [TPS_SYST_02437] for the `GlobalTimeSlave` are applicable for the `SynchronizedTimeBaseConsumer`.⌋

Class	**SynchronizedTimeBaseConsumer**			
Note	This meta-class represents a Synchronized Time Base Consumer. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimeBaseResource*			
Aggregated by	TimeSyncModuleInstantiation.timeBase			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
networkTime Consumer	GlobalTimeSlave	0..1	ref	This reference defines the GlobalTime Consumer which is synchronized with this Time Base. **Tags:** atp.Splitkey=networkTimeConsumer

Table 12.37: SynchronizedTimeBaseConsumer

[constr_10184] Multiplicity of the reference in the role `SynchronizedTimeBaseConsumer.networkTimeConsumer`

Imposition time: `IT_Mani`

⌈For each `SynchronizedTimeBaseConsumer`, the reference in the role `networkTimeConsumer` shall exist.⌋

Class	**SynchronizedTimeBaseProvider**			
Note	This meta-class represents a Synchronized Time Base Provider. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimeBaseResource*			
Aggregated by	TimeSyncModuleInstantiation.timeBase			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
networkTime Provider	GlobalTimeMaster	0..1	ref	This reference defines the GlobalTime Provider which is synchronized with this Time Base. **Tags:** atp.Splitkey=networkTimeProvider

Table 12.38: SynchronizedTimeBaseProvider

[constr_10185] Multiplicity of the reference in the role `SynchronizedTimeBaseProvider.networkTimeProvider`

Imposition time: `IT_Mani`

⌈For each `SynchronizedTimeBaseProvider`, the reference in the role `networkTimeProvider` shall exist.⌋

12.4.3.2 Persistent Time Base Value Storage

The Time Synchronization functional cluster allows to define a storage for a SynchronizedTimeBaseProvider time base. During shutdown the value of that specific time base is stored in a PersistencyDeploymentElement (either PersistencyKeyValuePair or PersistencyFile) and during startup the value is read from persistency and restored at the time base.

[TPS_MANI_03632] Semantics of TimeBaseProviderToPersistencyMapping ⌈The TimeBaseProviderToPersistencyMapping defines that the referenced SynchronizedTimeBaseProvider (TimeBaseProviderToPersistencyMapping.timeBaseProvider) time value shall be stored to persistency during shutdown and restored from persistency during startup (TimeBaseProviderToPersistencyMapping.persistencyDeploymentElement).⌋

It is also possible to define a cycle in which the time base value shall be stored periodically using the attribute TimeBaseProviderToPersistencyMapping.cyclicBackupInterval.

[TPS_MANI_03671] Periodic storage of time base value ⌈The attribute TimeBaseProviderToPersistencyMapping.cyclicBackupInterval defines an interval in which the time base value shall be stored periodically to the TimeBaseProviderToPersistencyMapping.persistencyDeploymentElement.⌋

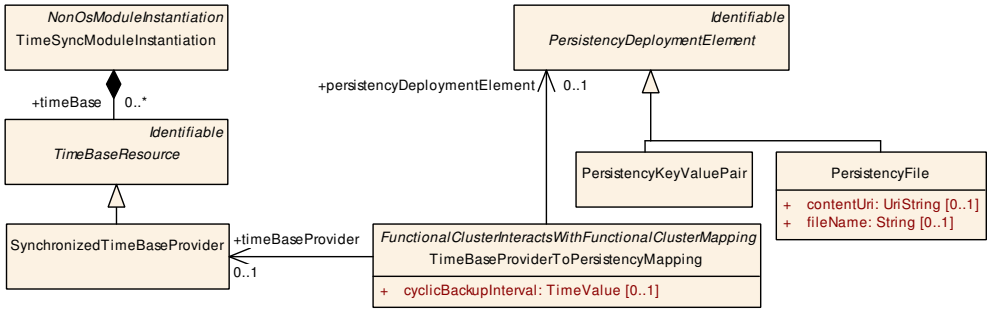


Figure 12.21: Modeling of TimeBaseProviderToPersistencyMapping

Class	**TimeBaseProviderToPersistencyMapping**			
Note	This meta-class represents the ability to define a mapping between a TimeBaseProvider and a PersistencyDeploymentElement for the purpose of storing and retrieving the time value. **Tags:** atp.recommendedPackage=FCInteractions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cyclicBackupInterval	TimeValue	0..1	attr	Time interval in seconds to store the time base value periodically to persistence.

△

Class	TimeBaseProviderToPersistencyMapping			
persistency Deployment Element	PersistencyDeployment Element	0..1	ref	This reference represents the PersistencyDeployment Element where the time value shall be stored in and retrieved from. **Tags:** atp.Splitkey=persistencyDeploymentElement
timeBase Provider	SynchronizedTimeBase Provider	0..1	ref	This reference represents the mapped TimeBase Provider. **Tags:** atp.Splitkey=timeBaseProvider

Table 12.39: TimeBaseProviderToPersistencyMapping

[constr_3619] Mandatory references of TimeBaseProviderToPersistency-Mapping

Imposition time: IT_Mani

⌈The references TimeBaseProviderToPersistencyMapping.persistency-DeploymentElement and TimeBaseProviderToPersistencyMapping.time-BaseProvider shall exist.⌋

12.4.4 Time Base to Port Prototype Mapping

The TimeBaseResource definition of Chapter 12.4.3 and the RPortPrototype typed by a sub-class of AbstractSynchronizedTimeBaseInterface of Chapter 4.8 have to be mapped to each other in order to define the binding of application software to the platform foundation software implementing the time synchronization.

The TimeSyncPortPrototypeToTimeBaseMapping takes the Process into account so that every instantiation of an Executable (and the resulting instantiation of all the RPortPrototypes typed by a sub-class of AbstractSynchronizedTime-BaseInterface) can be mapped individually to TimeBaseResources.

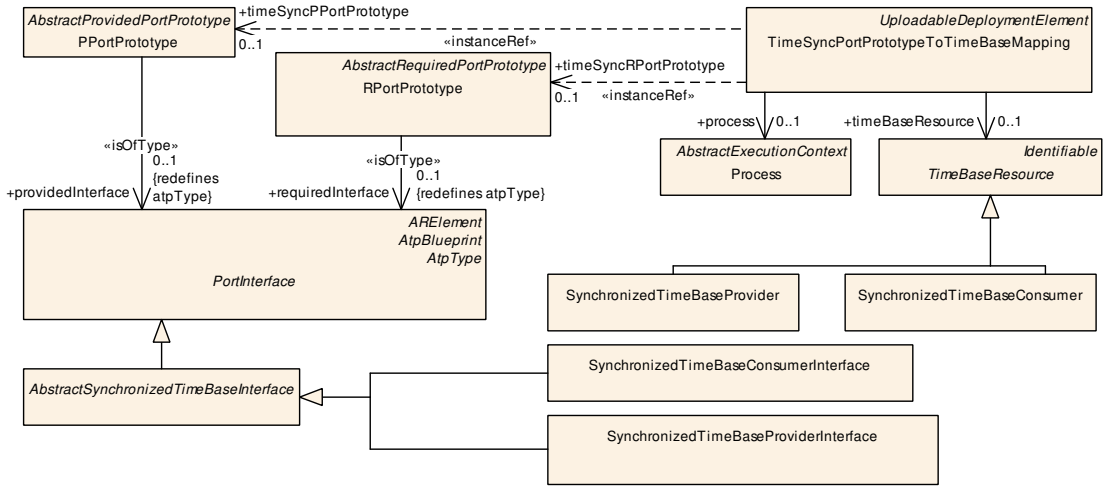


Figure 12.22: Modeling of TimeSyncPortPrototypeToTimeBaseMapping

[TPS_MANI_03548] Definition of TimeSyncPortPrototypeToTimeBaseMapping ⌈A TimeSyncPortPrototypeToTimeBaseMapping is used to define a mapping between a TimeBaseResource and a RPortPrototype typed by a sub-class of AbstractSynchronizedTimeBaseInterface in the context of a Process.⌋

Class	**TimeSyncPortPrototypeToTimeBaseMapping**			
Note	This meta-class provides the ability to map a PortPrototype typed by a AbstractSynchronizedTimeBase Interface to a TimeBaseResource in the context of a Process. **Tags:** atp.recommendedPackage=TimeSyncPortPrototypeToTimeBaseMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
process	Process	0..1	ref	Reference to the context Process this mapping applies to. **Tags:** atp.Splitkey=process
timeBase Resource	TimeBaseResource	0..1	ref	Reference to the mapped TimeBaseResource. **Tags:** atp.Splitkey=timeBaseResource
timeSyncPPort Prototype	PPortPrototype	0..1	iref	Instance reference to the mapped PPortPrototype typed by a AbstractSynchronizedTimeBaseInterface. **Tags:** atp.Splitkey=timeSyncPPortPrototype, timeSync PPortPrototype.contextRootSwComponentPrototype, timeSyncPPortPrototype.contextComponentPrototype, timeSyncPPortPrototype.targetPPortPrototype **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
timeSyncRPort Prototype	RPortPrototype	0..1	iref	Instance reference to the mapped RPortPrototype typed by a AbstractSynchronizedTimeBaseInterface. **Tags:** atp.Splitkey=timeSyncRPortPrototype, timeSync RPortPrototype.contextRootSwComponentPrototype, timeSyncRPortPrototype.contextComponentPrototype, timeSyncRPortPrototype.targetRPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 12.40: TimeSyncPortPrototypeToTimeBaseMapping

The example shown in Figure 12.23 illustrates the mapping of RPortPrototypes typed by one of the sub-classes of AbstractSynchronizedTimeBaseInterface to actually configured TimeBaseResources at the Time Sync Management.

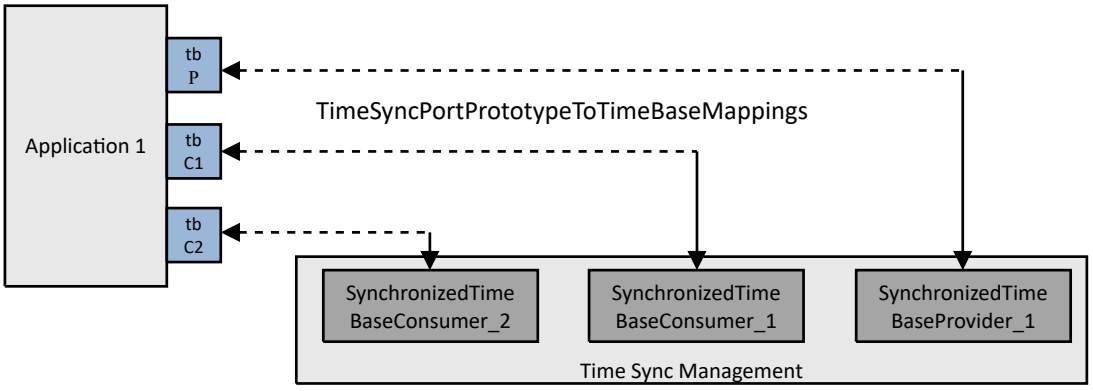


Figure 12.23: Example PortPrototype to TimeBase mapping

12.5 DoIP Configuration

Disclaimer

As of AUTOSAR R26-11, the canonical approach for the target configuration of the diagnostics functional cluster is by usage of the APMC model documented in the “Configuration” chapter of [24, SWS Diagnostics]. The description of target configuration via the M2 model (as described in this chapter) is still provided, but may be discontinued at a later point in time.

[TPS_MANI_03164] `Machine`-specific configuration settings for DoIP
⌈The `Machine`-specific configuration settings for DoIP are collected in `DoIpInstantiation`.⌋

Class	**DoIpInstantiation**			
Note	This meta-class defines the attributes for the DoIP configuration on a specific machine. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
doIpDesign	DoIpFunctionalClusterDesign	0..1	ref	Reference to the DoIP Design that this DoIpInstantiation implements. **Tags:** atp.Splitkey=doIpDesign
gid	PositiveUnlimitedInteger	0..1	attr	Configured GID (Group ID) used for VehicleIdentification Request. If configured, take this value (and set "Further action required" byte to 0x00="No further action required"), if not configured use ServiceInterface DoIPGroupIdentification to retrieve GID and 'further action required' values.

Table 12.41: DoIpInstantiation

[constr_9430] Existence of `DoIpInstantiation.doIpDesign`

Imposition time: `IT_Mani`

⌈For each `DoIpInstantiation`, the reference `doIpDesign` shall exist.⌋

Each `DoIpInstantiation` element contains a reference to the corresponding `MachineDesign` element that defines conditions that the `DoIpInstantiation` needs to implement.

[constr_3425] Restriction of `DoIpInstantiation`s on a `Machine`

Imposition time: `IT_Mani`

⌈Each `Machine` shall aggregate at most one `DoIpInstantiation` in the role `moduleInstantiation`.⌋

[constr_3496] Supported value range for attribute `DoIpInstantiation.gid`

Imposition time: `IT_Mani`

⌈The supported value range of attribute `DoIpInstantiation.gid` is limited to the interval [0..281474976710655].⌋

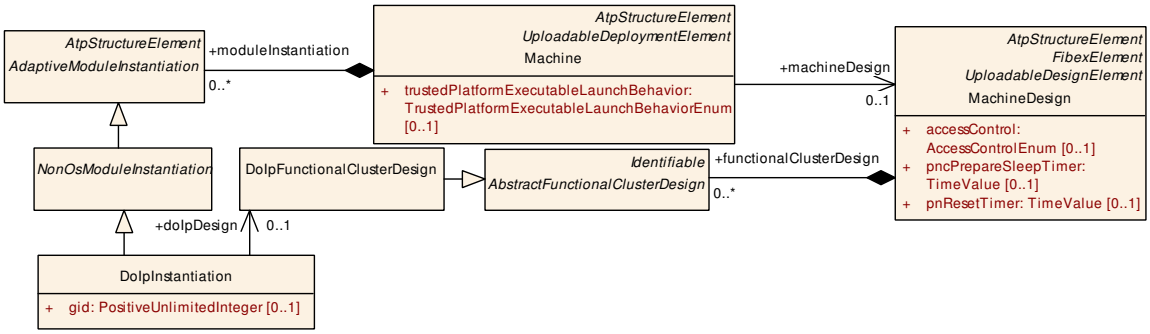


Figure 12.24: DoIP configuration

12.6 Log and Trace Module Configuration

The Log and Trace functionality in AUTOSAR supports the monitoring of applications and provides means to forward logging information onto the communication bus, the console, to the file system, or to the ARTI trace API. Forwarding to ARTI trace API is only valid for modeled messages.

The logging information is put into a standardized delivery and presentation format that is described in more detail in the [36, PRS Log and Trace protocol].The format contains meta-data that identifies for example the application that produces the logging information.

This chapter describes settings that are available in the Machine Manifest to configure the logging framework and defines the relation between the application design and the target-configuration model.

The modeling elements described in this chapter use elements from the Log And Trace configuration that defines an AUTOSAR platform independent format for description of log and trace messages and the source that produces these log and trace messages.

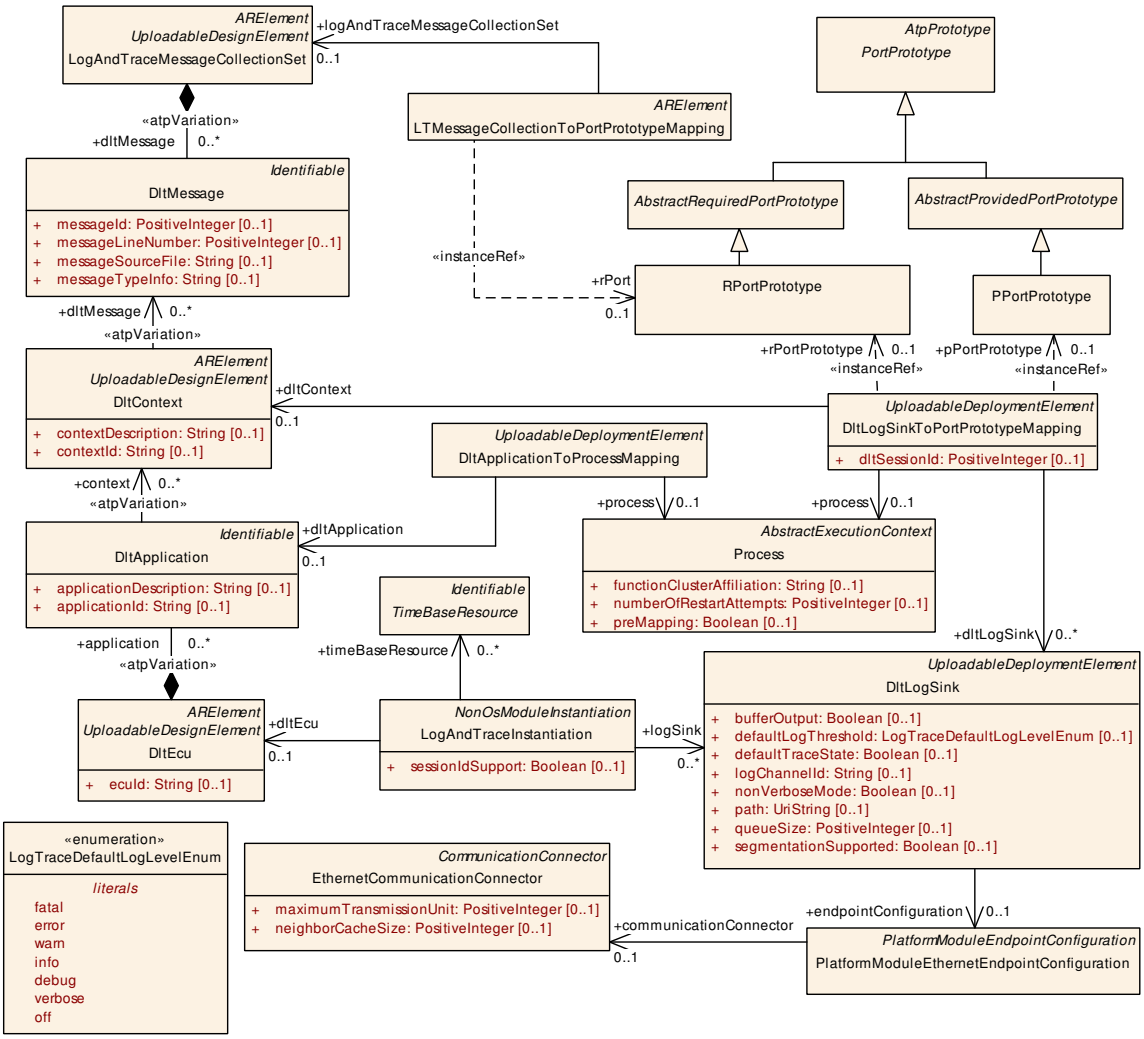


Figure 12.25: Log and Trace module configuration

12.6.1 Log and Trace Target-Configuration

[TPS_MANI_03162] **`Machine`-specific configuration settings for the Log and Trace functional cluster** ⌈The `Machine`-specific configuration settings for the Log and Trace functional cluster are collected in `LogAndTraceInstantiation`.⌋

[TPS_MANI_03282] **Assignment of a Dlt Ecu Identifier to a `LogAndTraceInstantiation`** ⌈The `Machine` is represented in the Log And Trace configuration by the `DltEcu` that is referenced from the `LogAndTraceInstantiation` in the role `dltEcu`. The referenced `DltEcu` defines the `ecuId` that is transported in the standard header of the log and trace message.⌋

[constr_3729] Upper multiplicity of reference in the role LogAndTraceInstantiation.timeBaseResource

Imposition time: IT_Mani

⌈In the context of LogAndTraceInstantiation, the reference in the role timeBaseResource shall exist at most once.⌋

Class	**LogAndTraceInstantiation**			
Note	This meta-class defines the attributes for the Log&Trace configuration on a specific machine. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dltEcu	DltEcu	0..1	ref	Reference to the Ecu representation in the Log And Trace configuration. **Tags:** atp.Splitkey=dltEcu
logSink	DltLogSink	*	ref	Reference to output sinks for log or trace messages that are produced on the Machine. **Tags:** atp.Splitkey=logSink
sessionId Support	Boolean	0..1	attr	This attribute defines whether the sessionId is used or not.
timeBase Resource	TimeBaseResource	*	ref	This reference is used to describe to which time base the Log and Trace module has access. From the Time Base Resource the Log and Trace module gets the needed information to generate the time stamp. **Tags:** atp.Splitkey=timeBaseResource

Table 12.42: LogAndTraceInstantiation

[constr_5275] Existence of LogAndTraceInstantiation.dltEcu

Imposition time: IT_Mani

⌈For each LogAndTraceInstantiation the reference to DltEcu in the role dltEcu shall exist.⌋

[constr_5276] Existence of LogAndTraceInstantiation.logSink

Imposition time: IT_Mani

⌈Each LogAndTraceInstantiation shall reference at least one DltLogSink in the role logSink.⌋

[TPS_MANI_03274] Configuration of log and trace sinks ⌈The output sinks for log or trace messages that are produced on the Machine are defined by DltLogSinks that are referenced in the role logSink from the LogAndTraceInstantiation.⌋

Class	**DltLogSink**			
Note	The meta-class defines the output sink for DltLogMessages **Tags:** atp.recommendedPackage=DltLogSinks This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bufferOutput	Boolean	0..1	attr	This attribute defines whether a buffer is used in case that the output sink is the console.
defaultLog Threshold	LogTraceDefaultLog LevelEnum	0..1	attr	This attribute allows to set a log level Threshold for Log Level filtering.
defaultTrace State	Boolean	0..1	attr	This attributes defines the default trace status.
endpoint Configuration	PlatformModule EthernetEndpoint Configuration	0..1	ref	Network configuration (Protocol, Port, IP Address) for transmission of dlt messages on a specific VLAN. **Tags:** atp.Splitkey=endpointConfiguration
logChannelId	String	0..1	attr	This attribute identifies the LogChannel for usage within the Log And Trace protocol.
nonVerbose Mode	Boolean	0..1	attr	This attribute defines whether this DltLogSink supports non-Verbose Dlt messages. If disabled only verbose mode messages shall be used.
path	UriString	0..1	attr	This attribute defines the path to the file that is used as output sink.
queueSize	PositiveInteger	0..1	attr	Length of the queue (in which messages can be stored before processing) in the unit "Log message".
segmentation Supported	Boolean	0..1	attr	If enabled, segmentation will be used for DLT messages that are larger than EthernetCommunication Connector.maximumTransmissionUnit referenced via Dlt LogSink.endpointConfiguration.

Table 12.43: DltLogSink

Enumeration	**LogTraceDefaultLogLevelEnum**
Note	This enum defines available log&trace log levels that may be used to define the severity level of a log message.
Aggregated by	DltLogChannel.logTraceDefaultLogThreshold, DltLogSink.defaultLogThreshold
Literal	***Description***
debug	Detailed information for programmers **Tags:** atp.EnumerationLiteralIndex=4
error	Error with impact to correct functionality **Tags:** atp.EnumerationLiteralIndex=1
fatal	Fatal error **Tags:** atp.EnumerationLiteralIndex=0
info	High level information **Tags:** atp.EnumerationLiteralIndex=3
off	logging is turned off **Tags:** atp.EnumerationLiteralIndex=6
verbose	Verbose debug message **Tags:** atp.EnumerationLiteralIndex=5
warn	Warning if correct behavior cannot be ensured **Tags:** atp.EnumerationLiteralIndex=2

Table 12.44: LogTraceDefaultLogLevelEnum

[TPS_MANI_03283] Standardized values for attribute `DltLogSink.category`
⌈AUTOSAR reserves the following values for attribute `DltLogSink.category`:

- `DLT_LOGSINK_REMOTE`
- `DLT_LOGSINK_DLT`
- `DLT_LOGSINK_FILE`
- `DLT_LOGSINK_CONSOLE`
- `DLT_LOGSINK_ARTI`

⌋

[TPS_MANI_02384] `DltLogSink` with `category DLT_LOGSINK_REMOTE` ⌈`DltLogSink` with `category` `DLT_LOGSINK_REMOTE` provides means to forward logging information to a Dlt log channel defined by `logChannelId`.⌋

[TPS_MANI_02385] `DltLogSink` with `category DLT_LOGSINK_DLT` ⌈`DltLogSink` with `category` `DLT_LOGSINK_DLT` provides means to forward logging information on a specific VLAN configured by `endpointConfiguration` via a Dlt log channel defined by `logChannelId`.⌋

[TPS_MANI_02386] `DltLogSink` with `category DLT_LOGSINK_FILE` ⌈`DltLogSink` with `category` `DLT_LOGSINK_FILE` provides means to forward logging information to a file in `path` of a file system.⌋

[TPS_MANI_02387] **DltLogSink with category DLT_LOGSINK_CONSOLE** ⌈DltLogSink with category DLT_LOGSINK_CONSOLE provides means to forward logging information to the console.⌋

[TPS_MANI_03325] **DltLogSink with category DLT_LOGSINK_ARTI** ⌈DltLogSink with category DLT_LOGSINK_ARTI provides means to forward all modeled messages without filtering to ARTI trace API.⌋

[TPS_MANI_03163] Network configuration for Log and Trace messages ⌈The output channel on Ethernet for Log and Trace messages is configured with the PlatformModuleEthernetEndpointConfiguration that is referenced by the DltLogSink in the role endpointConfiguration. The attributes tcpPort and udpPort are used to configure the Transport Protocol (Udp or Tcp) and the used Port number. The IP Address is configured in the NetworkEndpoint that is referenced by the PlatformModuleEthernetEndpointConfiguration via the EthernetCommunicationConnector.⌋

[constr_5277] Applicable values of DltLogSink.category vs. DltLogSink attributes

Imposition time: IT_Mani

⌈

Category	Applicable to ...					
	DltLogSink.logChannelId	DltLogSink.endpointConfiguration	DltLogSink.path	DltLogSink.bufferOutput	DltLogSink.nonVerboseMode	DltLogSink.segmentationSupported
DLT_LOGSINK_REMOTE	x					
DLT_LOGSINK_DLT	x	x			x	x
DLT_LOGSINK_FILE			x			
DLT_LOGSINK_CONSOLE				x		
DLT_LOGSINK_ARTI						

⌋

[constr_5278] **DltLogSink with category DLT_LOGSINK_REMOTE is only allowed to be referenced by DltLogSinkToPortPrototypeMapping**

Imposition time: IT_Mani

⌈DltLogSink with category DLT_LOGSINK_REMOTE shall not be referenced by LogAndTraceInstantiation in the role logSink.⌋

[constr_5279] `DltLogSink` with `category` `DLT_LOGSINK_DLT` is only allowed to be referenced by `LogAndTraceInstantiation`

Imposition time: IT_Mani

⌈`DltLogSink` with `category` `DLT_LOGSINK_DLT` shall not be referenced by `DltLogSinkToPortPrototypeMapping` in the role `dltLogSink`.⌋

[constr_5281] Existence of `DltLogSink.defaultTraceState`

Imposition time: IT_Mani

⌈For each `DltLogSink`, attribute `defaultTraceState` shall exist.⌋

12.6.2 Relation between Design and Target-Configuration

The relation between the `DltLogSink` and applications that are monitored and are producing the logging information is created with the `DltLogSinkToPortPrototypeMapping`.

[TPS_MANI_02388] Semantics of `DltLogSinkToPortPrototypeMapping`
⌈Meta-class `DltLogSinkToPortPrototypeMapping` has the ability to map a specific `PortPrototype` referenced in the role `rPortPrototype` or `pPortPrototype` to a `DltLogSink` referenced in the role `dltLogSink`. The `dltContext` reference to the `DltContext` that is defined in the Log And Trace configuration defines the relationship to `DltMessage`s that the application produces.

The mapping also comprises a reference to meta-class `Process` in order to accommodate for the fact that identical combinations of `DltLogSink` and `PortPrototype` may or may not apply for a given `Process` that represents the enclosing `Executable` at runtime.⌋

[constr_5282] Existence of `DltLogSinkToPortPrototypeMapping.process`

Imposition time: IT_Mani

⌈Each `DltLogSinkToPortPrototypeMapping` shall reference a `Process` in the role `process`.⌋

[constr_5283] Existence of `DltLogSinkToPortPrototypeMapping.dltLogSink`

Imposition time: IT_Mani

⌈Each `DltLogSinkToPortPrototypeMapping` shall reference at least one `DltLogSink` in the role `dltLogSink`.⌋

[constr_5284] Existence of DltLogSinkToPortPrototypeMapping.dltContext

Imposition time: IT_Mani

⌈Each DltLogSinkToPortPrototypeMapping shall reference a DltContext in the role dltContext.⌋

[constr_5285] Existence of PortPrototype references in DltLogSinkToPortPrototypeMapping

Imposition time: IT_Mani

⌈Each DltLogSinkToPortPrototypeMapping shall reference exactly one PortPrototype in the role rPortPrototype or pPortPrototype.⌋

Class	**DltLogSinkToPortPrototypeMapping**			
Note	This meta-class maps a PortPrototype to an output sink of a log and trace message. **Tags:** atp.recommendedPackage=DltLogSinkToPortPrototypeMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dltContext	DltContext	0..1	ref	Assignement of the DltContext that groups log and trace messages that will be transmitted to the DltLogSink. **Tags:** atp.Splitkey=dltContext
dltLogSink	DltLogSink	*	ref	Reference to the output sink to which the log or trace message will be transmitted, **Tags:** atp.Splitkey=dltLogSink
dltSessionId	PositiveInteger	0..1	attr	This attribute allows distinguishing log/trace messages from different instances of the same SW-C.
pPortPrototype	PPortPrototype	0..1	iref	Reference to PPortPrototype that is mapped to the DltLog Sink. **Tags:** atp.Splitkey=pPortPrototype, pPort Prototype.contextRootSwComponentPrototype, pPort Prototype.contextComponentPrototype, pPort Prototype.targetPPortPrototype **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference represents the process required as context for the mapping. **Tags:** atp.Splitkey=process
rPortPrototype	RPortPrototype	0..1	iref	Reference to RPortPrototype that is mapped to a DltLog Sink **Tags:** atp.Splitkey=rPortPrototype, rPortPrototype.context RootSwComponentPrototype, rPortPrototype.context ComponentPrototype, rPortPrototype.targetRPort Prototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 12.45: DltLogSinkToPortPrototypeMapping

[constr_5286] Restriction applicable for DltLogSinkToPortPrototypeMapping.rPortPrototype

Imposition time: IT_Mani

⌈The reference DltLogSinkToPortPrototypeMapping.rPortPrototype shall only be used for a RPortPrototype typed by a LogAndTraceInterface or by a ServiceInterface.⌋

[constr_5287] Restriction applicable for DltLogSinkToPortPrototypeMapping.pPortPrototype

Imposition time: IT_Mani

⌈The reference DltLogSinkToPortPrototypeMapping.pPortPrototype shall only be used for a PPortPrototype typed by a ServiceInterface.⌋

With the possibility to map a DltLogSink to a PortPrototype typed by a ServiceInterface it is possible to describe the ara::com communication that is monitored by Log And Trace.

[TPS_MANI_03286] Assignment of DltApplication to a Process ⌈Metaclass DltApplicationToProcessMapping has the ability to map a specific DltApplication that defines the applicationId used in the Log And Trace Protocol to a Process.⌋

[constr_5292] Assigned dltSessionId shall be consistent for the same PortPrototype

Imposition time: IT_Mani

⌈If several DltLogSinkToPortPrototypeMappings are referencing the same PortPrototype in the role rPortPrototype or pPortPrototype then the value for the dltSessionId in all these DltLogSinkToPortPrototypeMappings shall be the same.⌋

Class	**DltApplicationToProcessMapping**			
Note	This element assigns a DltApplicationId to a Process. **Tags:** atp.recommendedPackage=DltApplicationToProcessMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dltApplication	DltApplication	0..1	ref	Reference to a DltApplication that defines the application Id **Tags:** atp.Splitkey=dltApplication
process	Process	0..1	ref	Reference to the process that is assigned to a Log And Trace applicationId. **Tags:** atp.Splitkey=process

Table 12.46: DltApplicationToProcessMapping

[constr_5288] Existence of `process` reference in `DltApplicationToProcessMapping`

Imposition time: `IT_Mani`

⌈Each `DltApplicationToProcessMapping` shall reference a `Process` in the role `process`.⌋

[constr_5289] Existence of `dltApplication` reference in `DltApplicationToProcessMapping`

Imposition time: `IT_Mani`

⌈Each `DltApplicationToProcessMapping` shall reference a `DltApplication` in the role `dltApplication`.⌋

12.7 Network Management configuration

Disclaimer

As of AUTOSAR R26-11, the canonical approach for the target configuration of the network management functional cluster is by usage of the APMC model documented in [37, SWS Network Management] Chapter 10.1. The description of target configuration via the M2 model (as described in this chapter) is still provided, but may be discontinued at a later point in time.

12.7.1 UDP Network Management

[TPS_MANI_03166] `Machine`-specific configuration settings for NM module ⌈The `Machine`-specific configuration settings for Nm are collected in `NmInstantiation`.⌋

[TPS_MANI_03226] Collection of partialNetworks and VLANs in `NmNetworkHandle` ⌈The `NmNetworkHandle` element is used to describe a collection of `partialNetwork`s and `vlan`s that can be controlled collectively by the State Management.⌋

The `UdpNmCluster` with all included `UdpNmNode`s is described in the `System` design model. With the reference `NmNode.machine` the relation between the `System` design model and the `NmInstantiation` on a `Machine` is established.

Typically, the `System` design model is provided by an OEM that defines the network configuration and provides all configuration settings that are relevant for a network management cluster to an integrator.

The NM configuration options that will typically be set by an Integrator are collected in the `NmInstantiation` element. The Machine Manifest delivery to configure UdpNm consists of both, the `NmInstantiation` settings together with the `UdpNmCluster` and `UdpNmNode` settings.

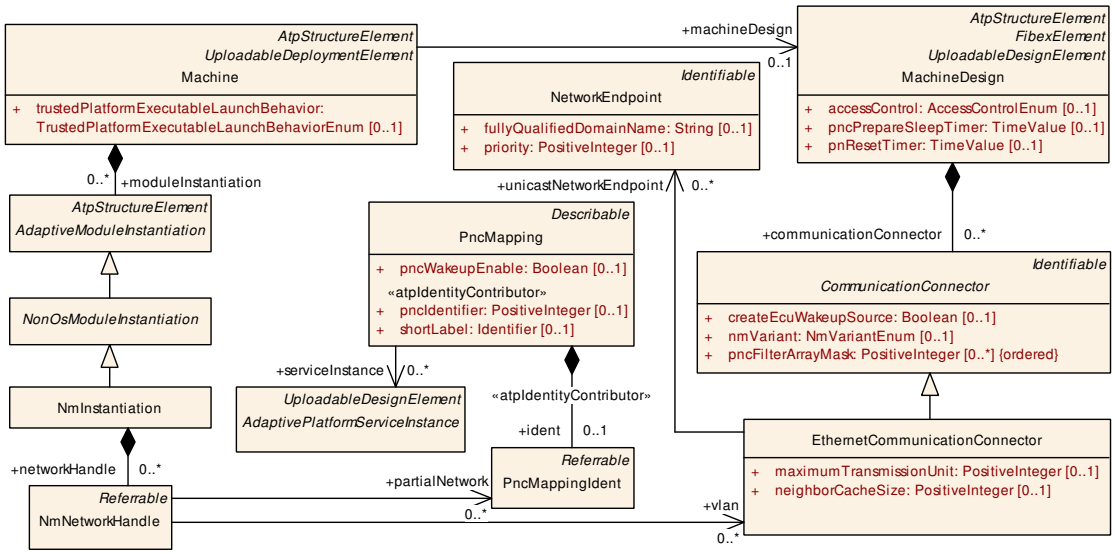


Figure 12.26: Network configuration for Nm

The `NmConfig` element is a wrapper that contains all network management specific configuration settings in the `System` model.

Class	**NmInstantiation**			
Note	This meta-class defines the attributes for the Nm configuration on a specific machine. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
networkHandle	NmNetworkHandle	*	aggr	Supported NmNetworkHandles used to control Partial Network Clusters/VLANs. **Tags:** atp.Splitkey=networkHandle, networkHandle.shortName

Table 12.47: NmInstantiation

Class	**NmNetworkHandle**			
Note	Group of partialNetworks and/or VLANs that can be controlled collectively. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Referrable*			
Aggregated by	NmInstantiation.networkHandle			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
partialNetwork	PncMappingIdent	*	ref	Reference to a Partial Network that is included in the Nm NetworkHandle. **Tags:** atp.Splitkey=partialNetwork
vlan	EthernetCommunicationConnector	*	ref	Reference to a VLAN that is included in the NmNetwork Handle. **Tags:** atp.Splitkey=vlan

Table 12.48: NmNetworkHandle

Class	PncMappingIdent			
Note	This meta-class is created to add the ability to become the target of a reference to the non-Referrable PncMapping.			
Base	*ARObject*, *Referrable*			
Aggregated by	PncMapping.ident			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 12.49: PncMappingIdent

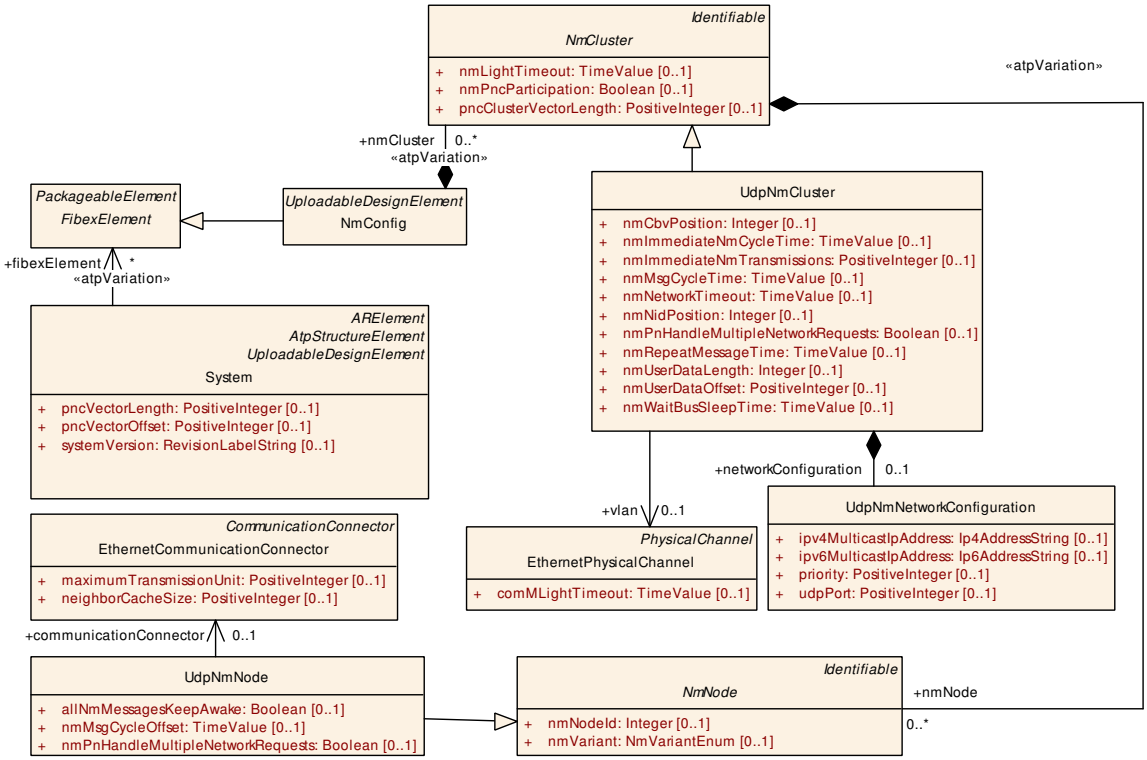


Figure 12.27: System Design for network management

AUTOSAR Adaptive Network Management is based on periodic NM messages, which are received by all `UdpNmNode`s in the `UdpNmCluster` via multicast. Reception of NM packets indicates that sending `UdpNmNode`s want to keep the `UdpNmCluster` awake.

If any node is ready to go to sleep mode, it stops sending NM messages, but as long as NM packets from other `UdpNmNode`s are received, it postpones transition to sleep mode.

[TPS_MANI_03167] Network configuration for Nm ⌈The UDP multicast connection over which Network Management messages are transported is configured with the `UdpNmNetworkConfiguration` that is aggregated by the `UdpNmCluster` in the role `networkConfiguration`. The attribute `udpPort` is used to configure the port number over which the Nm message is transmitted and received. The IP Address is configured either by `ipv4MulticastIpAddress` or `ipv6MulticastIpAddress`.⌋

[TPS_MANI_03279] Priority of Nm messages ⌈The attribute `priority` in the `UdpNmNetworkConfiguration` can be used to assign a VLAN priority to Network

Management messages that are transmitted over the Socket defined by the Udp-NmNetworkConfiguration.udpPort and ipv4MulticastIpAddress or ipv6MulticastIpAddress.⌋

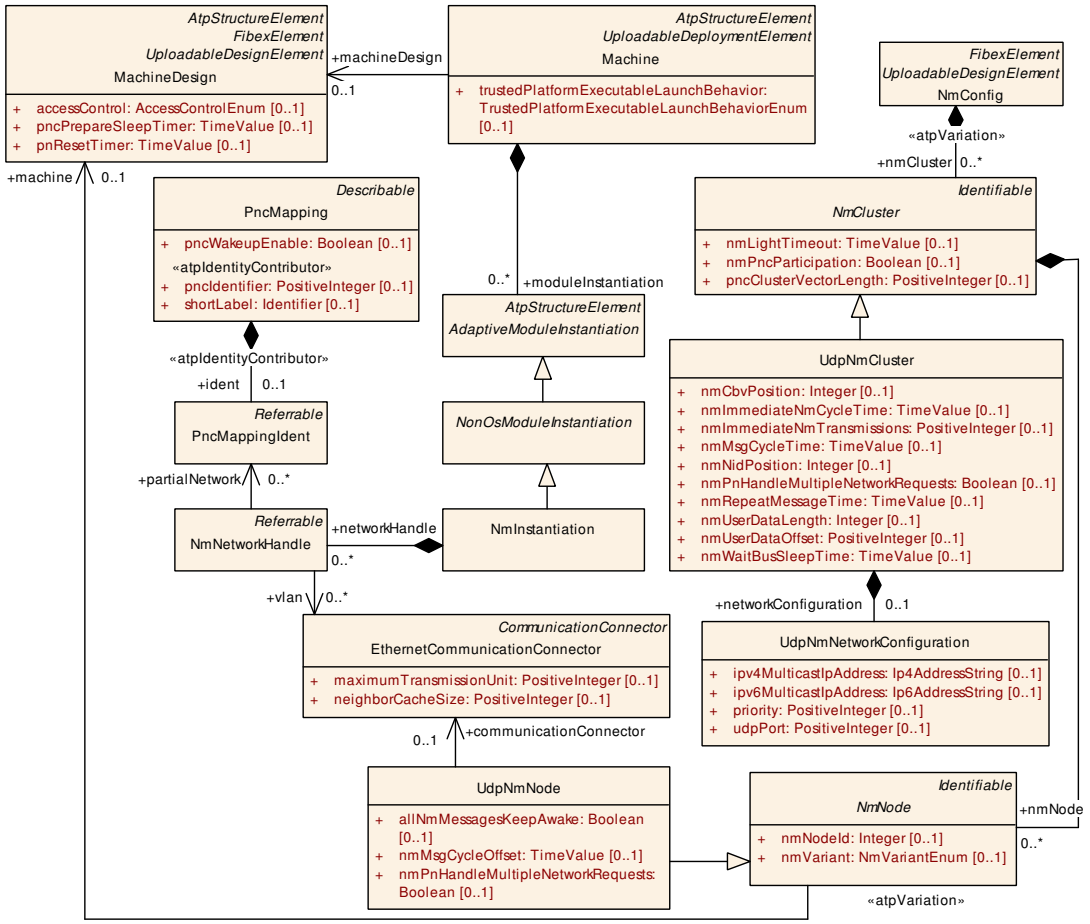


Figure 12.28: NM Cluster configuration

[constr_3419] Allowed usage of UdpNmNetworkConfiguration attributes

Imposition time: IT_Mani

⌈The UdpNmNetworkConfiguration that is aggregated by UdpNmCluster in the role networkConfiguration shall have either

- ipv4MulticastIpAddress or
- ipv6MulticastIpAddress.

⌋

Class	**UdpNmNetworkConfiguration**
Note	This meta-class defines the attributes for the configuration of a UDP port and UDP multicast IP address of the Nm communication on a VLAN. This Class is only used by the AUTOSAR Adaptive Platform.

△

Class	UdpNmNetworkConfiguration			
Base	*ARObject*			
Aggregated by	UdpNmCluster.networkConfiguration			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ipv4MulticastIp Address	Ip4AddressString	0..1	attr	Multicast IPv4 Address to which the message will be transmitted.
ipv6MulticastIp Address	Ip6AddressString	0..1	attr	Multicast IPv6 Address to which the message will be transmitted
priority	PositiveInteger	0..1	attr	This attribute defines the VLAN frame priority for messages on the Socket defined by the udpPort and the multicast IP address. Values from 0 (best effort) to 7 (highest) are allowed.
udpPort	PositiveInteger	0..1	attr	This attribute allows to configure a udp port number that is used for reception and transmission of UdpNm messages.

Table 12.50: UdpNmNetworkConfiguration

Class	NmConfig			
Note	Contains the all configuration elements for AUTOSAR Nm. **Tags:** atp.recommendedPackage=NmConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
nmCluster	NmCluster	*	aggr	Collection of NM Clusters atpVariation: Derived, because cluster can be variable. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=nmCluster, nmCluster.shortName, nm Cluster.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table 12.51: NmConfig

Class	***NmCluster*** (abstract)			
Note	Set of NM nodes coordinated with use of the NM algorithm.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	CanNmCluster, FlexrayNmCluster, UdpNmCluster			
Aggregated by	NmConfig.nmCluster			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
communication Cluster	CommunicationCluster	0..1	ref	Association to a CommunicationCluster in the topology description. **Tags:** atp.Splitkey=communicationCluster
nmLightTimeout	TimeValue	0..1	attr	Defines the timeout (in seconds) after COMM_FULL_ COMMUNICATION sub-state COMM_FULL_COM_ READY_SLEEP is left.

▽

△

Class	***NmCluster*** (abstract)			
nmNode	NmNode	*	aggr	Collection of NmNodes of the NmCluster. atpVariation: Derived, because NmNode can be variable. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=nmNode, nmNode.shortName, nm Node.variationPoint.shortLabel vh.latestBindingTime=postBuild
nmPnc Participation	Boolean	0..1	attr	Defines whether this NmCluster contributes to the partial network mechanism.
pncCluster VectorLength	PositiveInteger	0..1	attr	Optionally defines the length of the PNC Request Bit Vector per CommunicationCluster (and VLAN in case of UdpNm). If not defined then System.pncVectorLength applies. Should only make the PNC Request Bit Vector shorter (or same length as defined in System.pncVectorLength).

Table 12.52: NmCluster

Class	**UdpNmCluster**			
Note	Udp specific NmCluster attributes			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *NmCluster*, *Referrable*			
Aggregated by	NmConfig.nmCluster			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
network Configuration	UdpNmNetwork Configuration	0..1	aggr	Configuration of a UDP port and UDP multicast IP address of the Nm communication on a VLAN. **Tags:** atp.Splitkey=networkConfiguration This Attribute is only used by the AUTOSAR Adaptive Platform.
nmCbvPosition	Integer	0..1	attr	Defines the position of the control bit vector within the Nm Pdu (Byte position). If this attribute is not configured, the Control Bit Vector is not used.
nmImmediate NmCycleTime	TimeValue	0..1	attr	Defines the immediate NmPdu cycle time in seconds which is used for nmImmediateNmTransmissions NmPdu transmissions. This attribute is only valid if nmImmediate NmTransmissions is greater one.
nmImmediate Nm Transmissions	PositiveInteger	0..1	attr	Defines the number of immediate NmPdus which shall be transmitted. If the value is zero no immediate NmPdus are transmitted. The cycle time of immediate NmPdus is defined by nmImmediateNmCycleTime.
nmMsgCycle Time	TimeValue	0..1	attr	Period of a NmPdu in seconds. It determines the periodic rate in the periodic transmission mode with bus load reduction and is the basis for transmit scheduling in the periodic transmission mode without bus load reduction.
nmNetwork Timeout	TimeValue	0..1	attr	Network Timeout for NmPdus in seconds. It denotes the time how long the UdpNm shall stay in the Network Mode before transition into Prepare Bus-Sleep Mode shall take place.
nmNidPosition	Integer	0..1	attr	Defines the byte position of the source node identifier within the NmPdu. If this attribute is not configured, the Node Identification is not used.
nmPnHandle MultipleNetwork Requests	Boolean	0..1	attr	Defines if Nm performs an additional transition from Network Mode to Repeat Message State (true) or not (false).

▽

△

Class	**UdpNmCluster**			
nmRepeat MessageTime	TimeValue	0..1	attr	Timeout for Repeat Message State in seconds. Defines the time how long the NM shall stay in the Repeat Message State.
nmUserData Length	Integer	0..1	attr	Defines the length in bytes of the user data contained in the Nm message. User data excludes the PNC Request Bit Vector. This Attribute is only used by the AUTOSAR Adaptive Platform.
nmUserData Offset	PositiveInteger	0..1	attr	Specifies the offset (in bytes) of the user data information in the NM message. User data excludes the PNC Request Bit Vector. This Attribute is only used by the AUTOSAR Adaptive Platform.
nmWaitBus SleepTime	TimeValue	0..1	attr	Timeout for bus calm down phase in seconds. It denotes the time how long the CanNm shall stay in the Prepare Bus-Sleep Mode before transition into Bus-Sleep Mode shall take place.
vlan	EthernetPhysical Channel	0..1	ref	Reference to the vlan (represented by the Ethernet PhysicalChannel) this UdpNmCluster shall apply to. **Tags:** atp.Splitkey=vlan

Table 12.53: UdpNmCluster

Class	***NmNode*** (abstract)			
Note	The linking of NmEcus to NmClusters is realized via the NmNodes.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	CanNmNode, FlexrayNmNode, UdpNmNode			
Aggregated by	*NmCluster*.nmNode			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
machine	MachineDesign	0..1	ref	Reference to the machine that contains the NmNode. **Tags:** atp.Splitkey=machine This Attribute is only used by the AUTOSAR Adaptive Platform.
nmNodeId	Integer	0..1	attr	Node identifier of local NmNode. Shall be unique in the NmCluster.
nmVariant	NmVariantEnum	0..1	attr	Defines the functionality of Network Management. **Tags:** atp.Status=obsolete

Table 12.54: NmNode

Class	**UdpNmNode**			
Note	Udp specific NM Node attributes.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *NmNode*, *Referrable*			
Aggregated by	*NmCluster*.nmNode			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**UdpNmNode**			
allNmMessages KeepAwake	Boolean	0..1	attr	Specifies if Nm drops irrelevant NM PDUs. false: Only NM PDUs with a Partial Network Information Bit (PNI) = true and containing a Partial Network request for this ECU trigger the standard RX indication handling and thus keep the ECU awake true: Every NM PDU triggers the standard RX indication handling and keeps the ECU awake
communication Connector	EthernetCommunication Connector	0..1	ref	Reference to the CommunicationConnector that represents the UdpNmNode in the topology description. **Tags:** atp.Splitkey=communicationConnector This Attribute is only used by the AUTOSAR Adaptive Platform.
nmMsgCycle Offset	TimeValue	0..1	attr	Node specific time offset in the periodic transmission node. It determines the start delay of the transmission. Specified in seconds.
nmPnHandle MultipleNetwork Requests	Boolean	0..1	attr	Specifies if NM performs an additional transition from Network Mode to Repeat Message State (true) or not (false). This Attribute is only used by the AUTOSAR Adaptive Platform.

Table 12.55: UdpNmNode

CAUTION

The constraints marked with the imposition time `IT_SysDes` defined in chapter [4, TPS System Template] 6.9.3 *"UDP Network Management"* are valid for this specification as well.

12.7.2 Relation between Application Software and Network Handles

A `PPortPrototype` typed by a `NetworkManagementPortInterface` can make requests towards a status change of a network handle. But it is necessary to indicate which network handle is associated with which `PPortPrototype`, This relation can be indicated by means of the `NetworkHandlePortMapping`.

[constr_10441] Restriction for `NetworkHandlePortMapping.pPortPrototype-InExecutable`

Imposition time: `IT_Mani`

⌈A `PPortPrototype` referenced in the role `NetworkHandlePortMapping.pPort-PrototypeInExecutable` shall only be typed by a `NetworkManagementPort-Interface`.⌋

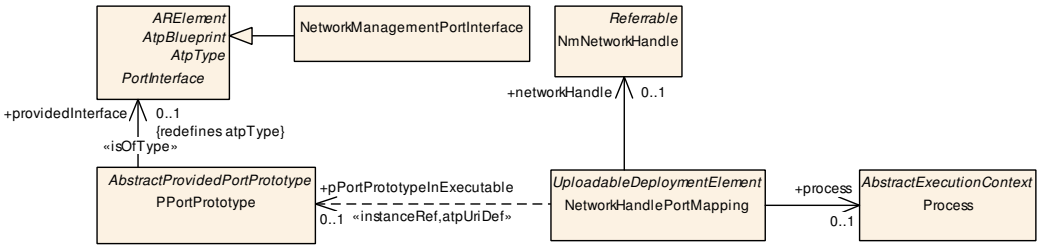


Figure 12.29: Modeling of the mapping between application software and network handle

Class	**NetworkHandlePortMapping**			
Note	This class is used to associate a PortPrototype with a network handle in order to control the network handle from the PortPrototype **Tags:** atp.recommendedPackage=NetworkHandleMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
networkHandle	NmNetworkHandle	0..1	ref	This reference identifies the applicable network handle. **Tags:** atp.Splitkey=networkHandle
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This reference identifies the applicable PortPrototype in the context of an enclosing Executable. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=pPortPrototypeInExecutable, pPort PrototypeInExecutable.contextRootSwComponent Prototype, pPortPrototypeInExecutable.context ComponentPrototype, pPortPrototypeInExecutable.target PPortPrototype **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference identifies the process applicable for the state management **Tags:** atp.Splitkey=process

Table 12.56: NetworkHandlePortMapping

12.7.3 Relation between Function Group states and Network Handles

[TPS_MANI_01343] Relation between `Function Group State`s and meta-class `NmNetworkHandle` ⌈An essential part of the network management on the *AUTOSAR adaptive platform* is the understanding which `Function Group State` requires network access. This relation can be expressed by means of meta-class `NmHandleToFunctionGroupStateMapping`.⌋

The relation between `Function Group State`s and network handles is in principle bi-directional, i.e. the following use cases exist for the creation of an `NmHandleToFunctionGroupStateMapping`:

- The application software wants to indicate which `Function Group State`s require network access. This means that attribute `mappingDirection` of the

first NmHandleToFunctionGroupStateMapping shall be set to NmHandle-MappingDirectionEnum.functionGroupStateToNmHandle.

- The application software wants to indicate which Function Group State shall be set if the network goes down. This means that attribute mappingDirection shall be set to NmHandleMappingDirectionEnum.nmHandleInactiveTo-FunctionGroupState.
- The application software wants to indicate which Function Group State shall be set if the network goes up. This means that attribute mapping-Direction shall be set to NmHandleMappingDirectionEnum.nmHandle-ActiveToFunctionGroupState.

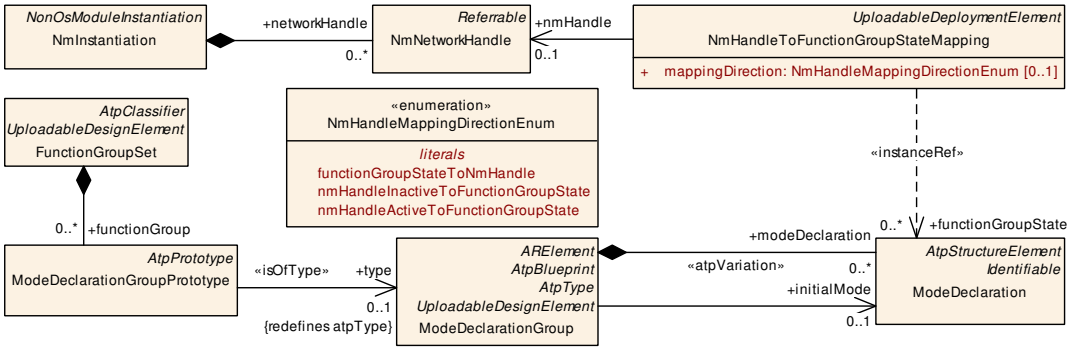


Figure 12.30: Mapping between NmNetworkHandle and Function Group States

Class	**NmHandleToFunctionGroupStateMapping**			
Note	This meta-class represents the ability to create a mapping between an NmNetworkHandle and a collection of function group states. This way, the impact of function groups on the network management can be specified. **Tags:** atp.recommendedPackage=NmHandleToFunctionGroupStateMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
functionGroup State	ModeDeclaration	*	iref	This reference identifies the collection of function group states in the context of the mapping. **Tags:** atp.Splitkey=functionGroupState, functionGroup State.contextModeDeclarationGroupPrototype, function GroupState.targetModeDeclaration **InstanceRef implemented by:** FunctionGroupStateIn FunctionGroupSetInstanceRef
mapping Direction	NmHandleMapping DirectionEnum	0..1	attr	This attribute describes the direction of the mapping.
nmHandle	NmNetworkHandle	0..1	ref	This reference identifies the applicable NmNetwork Handle in the context of the mapping. **Tags:** atp.Splitkey=nmHandle

Table 12.57: NmHandleToFunctionGroupStateMapping

Enumeration	**NmHandleMappingDirectionEnum**
Note	This enumeration provides direction values for the mapping of NM handles to function group states. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	NmHandleToFunctionGroupStateMapping.mappingDirection
Literal	***Description***
functionGroupState ToNmHandle	The purpose of the mapping is to indicate which function group state requires network access. **Tags:** atp.EnumerationLiteralIndex=0
nmHandleActiveTo FunctionGroupState	The purpose of the mapping is to indicate that the function group shall be switched to a given state if the network handle becomes active. **Tags:** atp.EnumerationLiteralIndex=2
nmHandleInactive ToFunctionGroup State	The purpose of the mapping is to indicate that the function group shall be switched to a given state if the network handle becomes inactive. **Tags:** atp.EnumerationLiteralIndex=1

Table 12.58: NmHandleMappingDirectionEnum

[TPS_MANI_01365] Semantics of attribute `NmHandleToFunctionGroupStateMapping.mappingDirection` set to `functionGroupStateToNmHandle` ⌈If the attribute `NmHandleToFunctionGroupStateMapping.mappingDirection` is set to the value `NmHandleMappingDirectionEnum.functionGroupStateToNmHandle`, then the semantics of the `NmHandleToFunctionGroupStateMapping` is that the referenced `nmHandle` is requested to become active if the `Function Group` to which the referenced `functionGroupState` belongs to is switched to the referenced `Function Group State`, formalized by meta-class `ModeDeclaration`.⌋

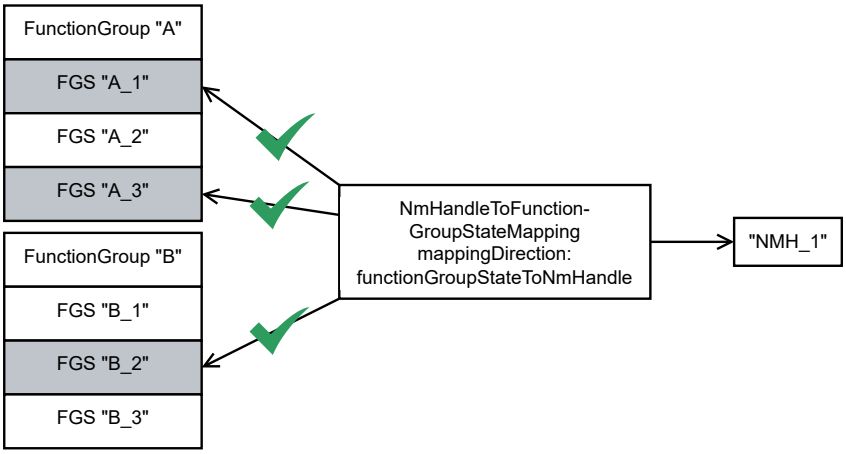


Figure 12.31: Mapping from `Function Group` to NM handle

The multiplicity of the reference `NmHandleToFunctionGroupStateMapping.functionGroupState` allows for the creation of a "shortcut" such that several `Function Group State`s can simultaneously be indicated as requiring communication in the context of a single `NmHandleToFunctionGroupStateMapping`, as depicted in Figure 12.31.

[TPS_MANI_01366] Semantics of attribute NmHandleToFunctionGroupStateMapping.mappingDirection set to nmHandleInactiveToFunctionGroupState ⌈If the attribute NmHandleToFunctionGroupStateMapping.mappingDirection is set to the value NmHandleMappingDirectionEnum.nmHandleInactiveToFunctionGroupState, then the semantics of the NmHandleToFunctionGroupStateMapping is that if the referenced nmHandle becomes **inactive**, the Function Group to which the referenced functionGroupState belongs to is switched to the referenced Function Group State, formalized by meta-class ModeDeclaration.⌋

[TPS_MANI_01367] Semantics of attribute NmHandleToFunctionGroupStateMapping.mappingDirection set to nmHandleActiveToFunctionGroupState ⌈If the attribute NmHandleToFunctionGroupStateMapping.mappingDirection is set to the value NmHandleMappingDirectionEnum.nmHandleActiveToFunctionGroupState, then the semantics of the NmHandleToFunctionGroupStateMapping is that if the referenced nmHandle becomes **active**, the function group to which the referenced functionGroupState belongs to is switched to the referenced Function Group State, formalized by meta-class ModeDeclaration.⌋

The mapping from NM Handle to Function Group State can only identify one Function Group State per Function Group because it is impossible to activate two Function Group States of the same Function Group at the same time.

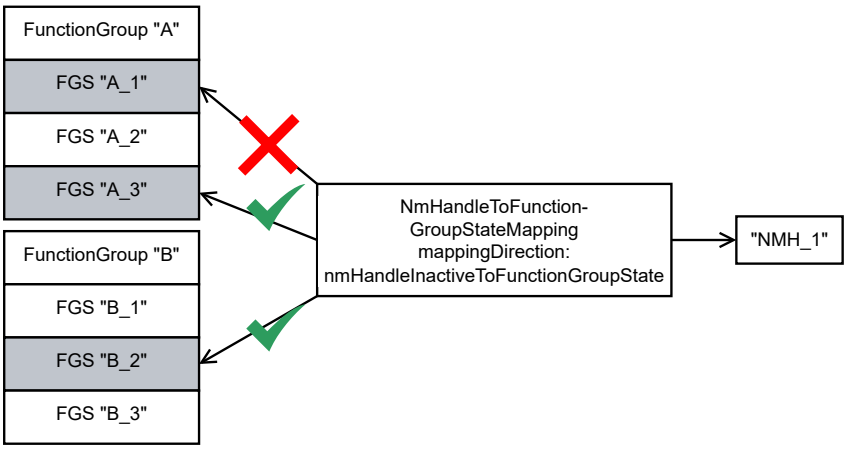


Figure 12.32: Mapping from NM handle to Function Group

Nevertheless, if an NM handle becomes active/inactive, it is very likely that this event has an impact on several SoftwareClusters. And because Function Groups cannot be shared among SoftwareClusters, it is necessary to be able to establish a 1:n communication pattern for this case under a precondition described by [constr_10101] and sketched by Figure 12.32.

[constr_10101] Attribute NmHandleToFunctionGroupStateMapping.mappingDirection is set to nmHandleActiveToFunctionGroupState or nmHandleInactiveToFunctionGroupState

Imposition time: IT_Mani

⌈If the value of attribute NmHandleToFunctionGroupStateMapping.mappingDirection is set to the value NmHandleMappingDirectionEnum.nmHandleActiveToFunctionGroupState or NmHandleMappingDirectionEnum.nmHandleInactiveToFunctionGroupState, then the reference NmHandleToFunctionGroupStateMapping.functionGroupState shall not refer to two (or more) ModeDeclarations **of the same** ModeDeclarationGroup.⌋

12.7.3.1 UdpNm configuration constraints

The Classic Platform and the Adaptive Platform use the same model for configuration of UdpNm. Some Classic Platform features like the NmCoordinator and UdpNmClusterCoupling are not supported on the *AUTOSAR adaptive platform*.

But the [4, TPS System Template] contains a more detailed description of UdpNmClusters and UdpNmNodes and defines modeling constraints that are also valid for the Adaptive Platform.

CAUTION

The constraints marked with the imposition time IT_SysDes defined in chapter [4, TPS System Template] 6.9.3 *"UDP Network Management"* are valid for this specification as well.

In addition, the following Adaptive Platform specific constraints are valid:

[constr_5227] Mandatory elements of UdpNmCluster

Imposition time: IT_Mani

⌈The following attributes shall always be defined for the UdpNmCluster:

- nmMsgCycleTime
- nmNetworkTimeout
- nmRepeatMessageTime
- nmWaitBusSleepTime
- communicationCluster

⌋

[constr_5228] Partial Networking timing constraint

Imposition time: IT_Mani

⌈For Partial Networking the following timing constraints shall be ensured: (MachineDesign.pnResetTimer + MachineDesign.pncPrepareSleepTimer) < UdpNmCluster.nmNetworkTimeout.⌋

[constr_10098] Relation of MachineDesign.pnResetTimer and UdpNmCluster.nmMsgCycleTime

Imposition time: IT_Mani

⌈For the configuration of the partial networking timing, the following condition shall be ensured: MachineDesign.pnResetTimer > UdpNmCluster.nmMsgCycleTime.⌋

12.7.4 CAN XL Network Management

The network management for CAN XL is modeled in a very similar way as the UDP NM. Please find details in Figure 12.33.

Although the CAN XL is entirely used to tunnel Ethernet communication, it still needs to be able to participate in the regular network management of the CAN cluster.

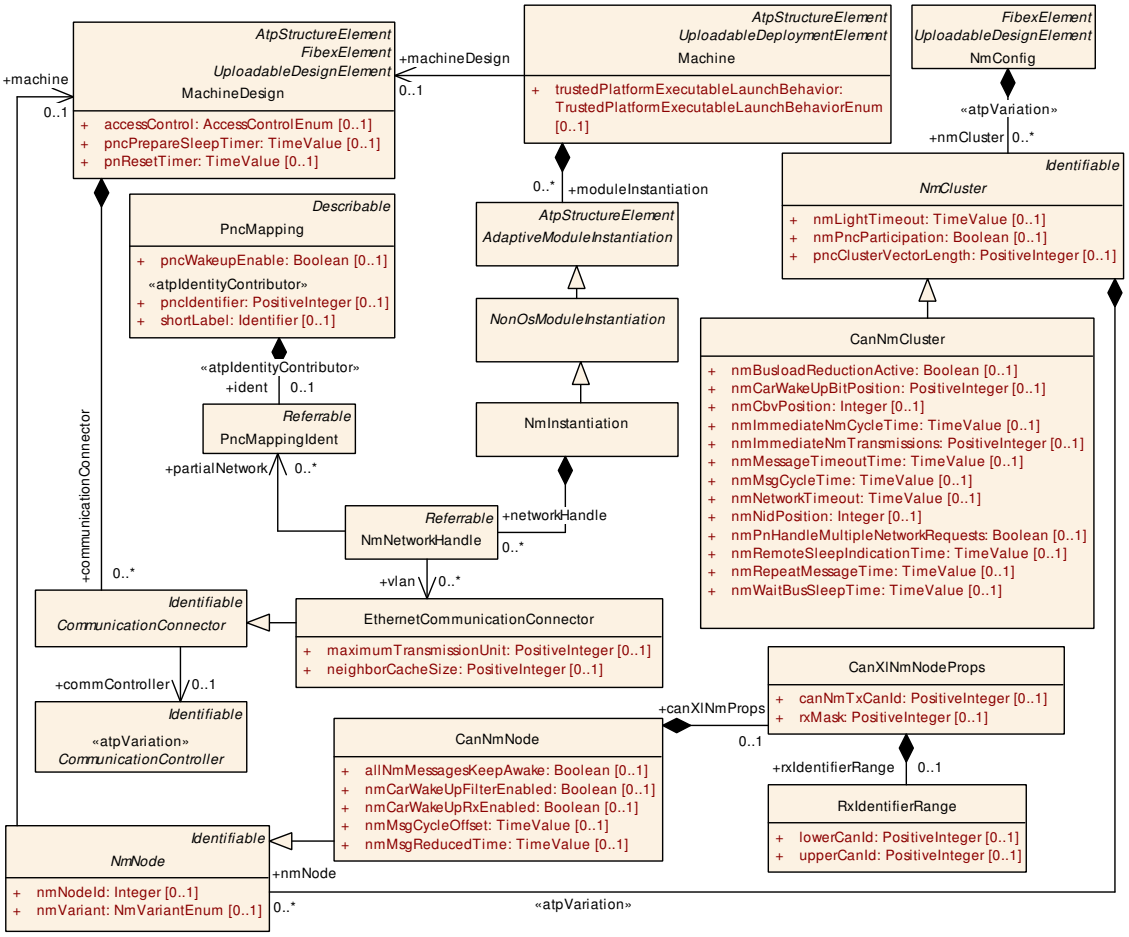


Figure 12.33: NM configuration for CAN XL

For this purpose, a relation between a EthernetCommunicationConnector and a CanNmNode that represents the local Machine in the CAN NM networking needs to be created.

The CAN ID used to transmit NM frames is configured in CanNmNode.canXlNmProps.canNmTxCanId.

The CAN IDs of received NM frames are configured by the respective attributes of the CanNmNodes connected to the same CanNmCluster.

Class	**CanNmNode**			
Note	CAN specific NM Node attributes.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *NmNode*, *Referrable*			
Aggregated by	*NmCluster*.nmNode			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
allNmMessages KeepAwake	Boolean	0..1	attr	Specifies if Nm drops irrelevant NM PDUs. false: Only NM PDUs with a Partial Network Information Bit (PNI) = true and containing a Partial Network request for this ECU trigger the standard RX indication handling and thus keep the ECU awake true: Every NM PDU triggers the standard RX indication handling and keeps the ECU awake
canXlNmProps	CanXlNmNodeProps	0..1	aggr	This aggregation contributes the CAN IDs for the execution of CAN NM on the adaptive platform. **Tags:** atp.Splitkey=canXlNmProps This Attribute is only used by the AUTOSAR Adaptive Platform.
nmCarWakeUp FilterEnabled	Boolean	0..1	attr	If this attribute is set to true the CareWakeUp filtering is supported.
nmCarWakeUp RxEnabled	Boolean	0..1	attr	If set to true this attribute enables the support of CarWake Up bit evaluation in received NmPdus.
nmMsgCycle Offset	TimeValue	0..1	attr	Node specific time offset in the periodic transmission node. It determines the start delay of the transmission. Specified in seconds.
nmMsg ReducedTime	TimeValue	0..1	attr	Node specific bus cycle time in the periodic transmission mode with bus load reduction. Specified in seconds.

Table 12.59: CanNmNode

Class	**CanXlNmNodeProps**			
Note	Definition of CAN attributes in the context of CanXL usage on the adaptive platform. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	CanNmNode.canXlNmProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
canNmTxCanId	PositiveInteger	0..1	attr	This attribute is used to define the identifier the CAN NM frame shall use on the CAN network.
rxIdentifier Range	RxIdentifierRange	0..1	aggr	Optional definition of a CanId range. **Tags:** atp.Splitkey=rxIdentifierRange

▽

△

Class	CanXlNmNodeProps			
rxMask	PositiveInteger	0..1	attr	Identifier mask which denotes the relevant bits in the CAN Identifier. Together with the identifier, this parameter defines a CAN identifier range.

Table 12.60: CanXlNmNodeProps

Class	RxIdentifierRange			
Note	Optional definition of a CanId range to reduce the effort of specifying every possible FrameTriggering within the defined Id range during reception. All frames received within a range are mapped to the same Pdu that is passed to a upper layer module (e.g. Nm, CDD, PduR).			
Base	*ARObject*			
Aggregated by	CanFrameTriggering.rxIdentifierRange, CanXlNmNodeProps.rxIdentifierRange, IEEE1722TpAcfCanPart.canIdentifierRange			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
lowerCanId	PositiveInteger	0..1	attr	This attribute can be used together with the upperCanId attribute to define a range of CanIds.
upperCanId	PositiveInteger	0..1	attr	This attribute can be used together with the lowerCanId attribute to define a range of CanIds.

Table 12.61: RxIdentifierRange

Class	CanNmCluster			
Note	Can specific NmCluster attributes			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *NmCluster*, *Referrable*			
Aggregated by	NmConfig.nmCluster			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
nmBusloadReductionActive	Boolean	0..1	attr	It determines if bus load reduction for the respective Can Nm channel is active or not.
nmCarWakeUpBitPosition	PositiveInteger	0..1	attr	Specifies the bit position of the CarWakeUp within the Nm Pdu.
nmCarWakeUpFilterNodeId	PositiveInteger	0..1	attr	Source node identifier for CarWakeUp filtering.
nmCbvPosition	Integer	0..1	attr	Defines the position of the control bit vector within the Nm Pdu (Byte position). If this attribute is not configured, the Control Bit Vector is not used.
nmImmediateNmCycleTime	TimeValue	0..1	attr	Defines the immediate NmPdu cycle time in seconds which is used for nmImmediateNmTransmissions NmPdu transmissions. This parameter is only valid if CanNm ImmediateNmTransmissions is greater one.
nmImmediateNmTransmissions	PositiveInteger	0..1	attr	Defines the number of immediate NmPdus which shall be transmitted. If the value is zero no immediate NmPdus are transmitted. The cycle time of immediate NmPdus is defined by nmImmediateNmCycleTime.
nmMessageTimeoutTime	TimeValue	0..1	attr	Timeout of an NmPdu in seconds. It determines how long the NM shall wait with notification of transmission failure while communication errors occur on the bus.
nmMsgCycleTime	TimeValue	0..1	attr	Period of a NmPdu in seconds. It determines the periodic rate in the periodic transmission mode with bus load reduction and is the basis for transmit scheduling in the periodic transmission mode without bus load reduction.

▽

△

Class	**CanNmCluster**			
nmNetwork Timeout	TimeValue	0..1	attr	Network Timeout for NmPdus in seconds It denotes the time how long the CanNm shall stay in the Network Mode before transition into Prepare Bus-Sleep Mode shall take place.
nmNidPosition	Integer	0..1	attr	Defines the byte position of the source node identifier within the NmPdu. If this attribute is not configured, the Node Identification is not used.
nmPnHandle MultipleNetwork Requests	Boolean	0..1	attr	Defines if Nm performs an additional transition from Network Mode to Repeat Message State (true) or not (false).
nmRemote SleepIndication Time	TimeValue	0..1	attr	Timeout for Remote Sleep Indication in seconds. It defines the time how long it shall take to recognize that all other nodes are ready to sleep.
nmRepeat MessageTime	TimeValue	0..1	attr	Timeout for Repeat Message State in seconds. Defines the time how long the NM shall stay in the Repeat Message State.
nmWaitBus SleepTime	TimeValue	0..1	attr	Timeout for bus calm down phase in seconds. It denotes the time how long the CanNm shall stay in the Prepare Bus-Sleep Mode before transition into Bus-Sleep Mode shall take place.

Table 12.62: CanNmCluster

12.8 Update and Configuration Management

Disclaimer

As of AUTOSAR R25-11, the canonical approach for the target configuration of the update and configuration management functional cluster is by usage of the APMC model documented in [38, SWS Specification of Update and Configuration Management] Chapter 10.1. The description of target configuration via the M2 model (as described in this chapter) is still provided, but may be discontinued at a later point in time.

12.8.1 Overview

[TPS_MANI_01226] Machine-specific configuration settings for the UCM module
⌈The Machine-specific configuration settings for UCM are collected in abstract meta-class `UcmModuleInstantiation` and its two concrete specializations:

- `UcmMasterModuleInstantiation` represents the configuration of a *UCM Master*.
- `UcmSubordinateModuleInstantiation` represents the configuration of a *UCM Subordinate*.

⌋

Please find more information about meta-class UcmMasterModuleInstantiation in Chapter 12.8.3 and about UcmSubordinateModuleInstantiation in Chapter 12.8.4.

Class	***UcmModuleInstantiation*** (abstract)			
Note	This meta-class represents the ability to define the target-configuration of a UCM instantiation. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Subclasses	UcmMasterModuleInstantiation, UcmSubordinateModuleInstantiation			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
identifier	String	0..1	attr	This represents the identification of a UCM.
maxBlockSize	PositiveInteger	0..1	attr	This attribute denotes the maximum block size (unit: bytes) used in the UCM implementation.
version	StrongRevisionLabel String	0..1	attr	This attribute defines the software version of the UCM on this platform. Note that the definition of the version is required if the ability of the SoftwarePackage to require a minimum version of the UCM is utilized.

Table 12.63: UcmModuleInstantiation

[constr_10428] Existence of attribute UcmModuleInstantiation.identifier in subclasses

Imposition time: IT_Mani

⌈The attribute identifier shall **not** exist in a UcmMasterModuleInstantiation.⌋

[constr_10200] Multiplicity of attribute UcmModuleInstantiation.identifier

Imposition time: IT_Mani

⌈For each UcmModuleInstantiation, the attribute identifier shall exist.⌋

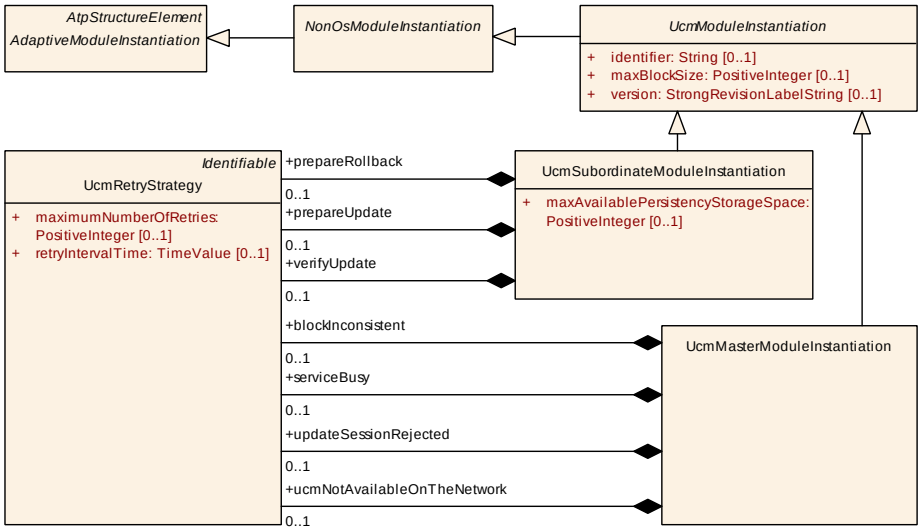


Figure 12.34: Modeling of UcmModuleInstantiation

[TPS_MANI_01227] Semantics of attribute `UcmModuleInstantiation.identifier` ⌈Attribute `UcmModuleInstantiation.identifier` shall be used to identify a specific UCM on a specific `Machine` during a service discovery run by a master UCM or VUM.⌋

The usage of attribute `UcmModuleInstantiation.identifier` is documented in Figure 12.35. The master UCM or VUM acts as a client in a service discovery that is configured to search for *any* server.

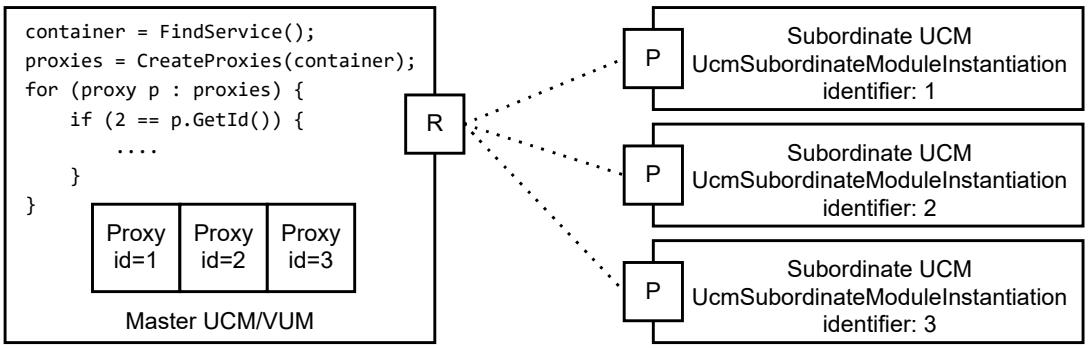


Figure 12.35: Identification of subordinate UCM modules by the master UCM

The individual UCMs offer their service and then the master UCM as the client calls a specific `method` in the server's `ServiceInterface` to reveal the `identifier` of each server.

In the case of this example there are three slave UCMs with `identifier` set to 1, 2, and 3. The master UCM instantiates a proxy for each of the slave UCMs such that the value of the respective `identifier` can be retrieved from the proxy in order to be able to communicate with a specific slave UCM.

The master UCM or VUM can then instantiate proxies for each service offer and programmatically access the respective server going forward.

[constr_1691] `UcmModuleInstantiation.identifier` shall be unique

Imposition time: `IT_Mani`

⌈The value of attribute `UcmModuleInstantiation.identifier` shall be unique for each `Machine` in a given vehicle.⌋

12.8.2 Retry Strategy

[TPS_MANI_01375] Semantics of meta-class `UcmRetryStrategy` ⌈Both the `UcmMasterModuleInstantiation` and the `UcmSubordinateModuleInstantiation` have the ability to retry certain activities if these activities don't succeed. The configuration of such retry cycles is done by means of meta-class `UcmRetryStrategy`.⌋

Class	UcmRetryStrategy			
Note	This meta-class describes the configuration of the retry strategy for a sub-class of UcmModule Implementation. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	UcmMasterModuleInstantiation.blockInconsistent, UcmMasterModuleInstantiation.serviceBusy, UcmMasterModuleInstantiation.ucmNotAvailableOnTheNetwork, UcmMasterModuleInstantiation.updateSessionRejected, UcmSubordinateModuleInstantiation.prepareRollback, UcmSubordinateModuleInstantiation.prepareUpdate, UcmSubordinateModuleInstantiation.verifyUpdate			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
maximumNumberOfRetries	PositiveInteger	0..1	attr	This attribute defines the maximum number of time the UCM module instantiation shall attempt a retry.
retryIntervalTime	TimeValue	0..1	attr	This attribute defines the time (in seconds) between two retry attempts.

Table 12.64: UcmRetryStrategy

[constr_10105] Existence of `UcmRetryStrategy.maximumNumberOfRetries`

Imposition time: `IT_Mani`

⌈For each `UcmRetryStrategy`, attribute `maximumNumberOfRetries` shall exist.⌋

[constr_10106] Existence of `UcmRetryStrategy.retryIntervalTime`

Imposition time: `IT_Mani`

⌈For each `UcmRetryStrategy`, attribute `retryIntervalTime` shall exist.⌋

12.8.3 UCM Master Module Instantiation

[TPS_MANI_01376] Semantics of meta-class `UcmMasterModuleInstantiation`

⌈Meta-class `UcmMasterModuleInstantiation` is used to configure the deployment of a *UCM Master* and the various retry strategies associated with the *UCM Master*.⌋

Class	UcmMasterModuleInstantiation			
Note	This meta-class represents the ability to define the target-configuration of a UCM Master instantiation. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*, *UcmModuleInstantiation*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
blockInconsistent	UcmRetryStrategy	0..1	aggr	This attribute defines the retry strategy of the UCM Master for the case that the block is inconsistent. **Tags:** atp.Splitkey=blockInconsistent, blockInconsistent.shortName

▽

△

Class	**UcmMasterModuleInstantiation**			
serviceBusy	UcmRetryStrategy	0..1	aggr	This attribute defines the retry strategy of the UCM Master for the case that the service is busy. **Tags:** atp.Splitkey=serviceBusy, serviceBusy.shortName
ucmNot AvailableOnThe Network	UcmRetryStrategy	0..1	aggr	This attribute defines the retry strategy of the UCMMaster for the case that one (or more) UCM is not available on the network. **Tags:** atp.Splitkey=ucmNotAvailableOnTheNetwork, ucm NotAvailableOnTheNetwork.shortName
updateSession Rejected	UcmRetryStrategy	0..1	aggr	This attribute defines the retry strategy of the UcmMaster for the case that the update session is rejected. **Tags:** atp.Splitkey=updateSessionRejected, update SessionRejected.shortName

Table 12.65: UcmMasterModuleInstantiation

[constr_10107] Existence of the attribute `UcmMasterModuleInstantiation.blockInconsistent`

Imposition time: `IT_Mani`

⌈The attribute `UcmMasterModuleInstantiation.blockInconsistent` shall exist.⌋

[constr_10108] Existence of the attribute `UcmMasterModuleInstantiation.serviceBusy`

Imposition time: `IT_Mani`

⌈The attribute `UcmMasterModuleInstantiation.serviceBusy` shall exist.⌋

[constr_10109] Existence of the attribute `UcmMasterModuleInstantiation.updateSessionRejected`

Imposition time: `IT_Mani`

⌈The attribute `UcmMasterModuleInstantiation.updateSessionRejected` shall exist.⌋

[constr_10578] Existence of the attribute `UcmMasterModuleInstantiation.ucmNotAvailableOnTheNetwork`

Imposition time: `IT_Mani`

⌈The attribute `UcmMasterModuleInstantiation.ucmNotAvailableOnTheNetwork` shall exist.⌋

12.8.4 UCM Subordinate Module Instantiation

[TPS_MANI_01377] Semantics of the meta-class `UcmSubordinateModuleInstantiation` ⌈Meta-class `UcmSubordinateModuleInstantiation` is used to configure the deployment of a *UCM Subordinate* and the various retry strategies associated with the *UCM Subordinate*.⌋

Class	**UcmSubordinateModuleInstantiation**			
Note	This meta-class represents the ability to define the target-configuration of a UCM Subordinate instantiation. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*, *UcmModuleInstantiation*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
maxAvailable Persistency StorageSpace	PositiveInteger	0..1	attr	This attribute names the maximum amount of space available for persistent data handled by the Persistency of installed packages. The UCM needs to figure out from traversing the minimum storage requirement from existing PersistencyDeployments whether specific packages can be installed from the perspective of available storage space. Note that the minimum storage requirement of PersistencyDeployment needs to include space for the handling of the storage, which shall be calculated by the tooling that creates the manifest information inside the package.
prepareRollback	UcmRetryStrategy	0..1	aggr	This attribute identifies the configuration of prepare rollback retries initiated by the Ucm Subordinate. **Tags:** atp.Splitkey=prepareRollback, prepare Rollback.shortName
prepareUpdate	UcmRetryStrategy	0..1	aggr	This attribute identifies the configuration of prepare update retries initiated by the Ucm Subordinate. **Tags:** atp.Splitkey=prepareUpdate, prepareUpdate.short Name
verifyUpdate	UcmRetryStrategy	0..1	aggr	This attribute identifies the configuration of verify update retries initiated by the Ucm Subordinate. **Tags:** atp.Splitkey=verifyUpdate, verifyUpdate.shortName

Table 12.66: UcmSubordinateModuleInstantiation

[constr_10374] Existence of the aggregation in the role UcmSubordinateModuleInstantiation.verifyUpdate

Imposition time: IT_Mani

⌈The aggregation in the role UcmSubordinateModuleInstantiation.verifyUpdate shall exist.⌋

[constr_10375] Existence of the aggregation in the role UcmSubordinateModuleInstantiation.prepareUpdate

Imposition time: IT_Mani

⌈The aggregation in the role UcmSubordinateModuleInstantiation.prepareUpdate shall exist.⌋

[constr_10376] Existence of the aggregation in the role UcmSubordinateModuleInstantiation.prepareRollback

Imposition time: IT_Mani

⌈The aggregation in the role UcmSubordinateModuleInstantiation.prepareRollback shall exist.⌋

[constr_10508] Existence of the aggregation in the role UcmSubordinateModuleInstantiation.maxAvailablePersistencyStorageSpace

Imposition time: IT_Mani

⌈The aggregation in the role UcmSubordinateModuleInstantiation.maxAvailablePersistencyStorageSpace shall exist.⌋

[constr_10110] Existence of UcmSubordinateModuleInstantiation on a Machine

Imposition time: IT_Mani

⌈For each Machine, a ProcessToMachineMapping shall exist that refers in the role machine to the Machine and in the role nonOsModuleInstantiation to a UcmSubordinateModuleInstantiation.⌋

12.9 CM configuration

The definition of the target-configuration for the *Identity and Access Manager* represents the creation of actual grants, as opposed to the definition of grants on design level.

One important aspect of the modeling on target-configuration level is that it is not intended to include a large portion of the design model. The goal is to keep the target-configuration part as self-contained as possible.

While this approach represents a significant benefit for the size of target-configuration models it also creates some sort of disconnect between design and target-configuration. In other words, the connection of the modeling of a specific Grant to the respective intent in the design model is not immediately obvious.

To mitigate this issue, AUTOSAR introduced the GrantDesign that in turn allows for the identification of the corresponding intent modeling. When loading the design model and target-configuration model together into a suitable tool it would still be possible to run an analysis in terms of completeness of the overall IAM configuration.

The enforcement of access restrictions is not mandatory for a Machine. Therefore, the existence of a Grant by itself is not sufficient to activate the IAM mechanisms.

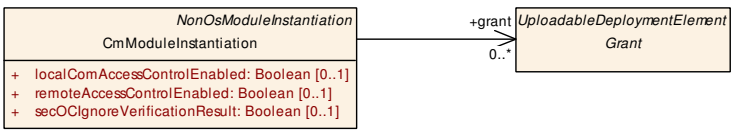


Figure 12.36: Modeling of the CmModuleInstantiation

Class	CmModuleInstantiation			
Note	This meta-class represents the ability to define a definition of a Communication Management instantiation. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
grant	Grant	*	ref	This reference identifies the applicable Grants for this Cm ModuleInstantiation. **Tags:** atp.Splitkey=grant
localCom AccessControl Enabled	Boolean	0..1	attr	This switch activates the policy enforcement in Communication Management on local applications.
remoteAccess ControlEnabled	Boolean	0..1	attr	This switch activates the check of the remote subject.
secOCIgnore Verification Result	Boolean	0..1	attr	The result of the authentication process (e.g. MAC Verify) is ignored after the first try and the SecOC proceeds like the result was a success. The calculation of the authenticator is still done, only its result will be ignored. • true: enabled (verification result is ignored). • false: disabled (verification result is NOT ignored).

Table 12.67: CmModuleInstantiation

[constr_1695] Semantics of a `Grant` depends on the existence of `CmModuleInstantiation`

Imposition time: `IT_Mani`

⌈The existence of `Grant`s shall only be enforced if in the context of the enclosing `Machine` an `CmModuleInstantiation` has been defined and is referencing the `Grant`.⌋

[constr_3677] `ComGrant`s referencing DDS Service Instances

Imposition time: `IT_Mani`

⌈`ComGrant`s associated with `DdsProvidedServiceInstance`s or `DdsRequiredServiceInstance`s via the `serviceInstance` attribute shall not be referenced by `CmModuleInstantiation.grant`.⌋

[constr_3677] exists because access control in the DDS Network Binding is delegated to DDS Security.

Class	**Grant** (abstract)
Note	This meta-class serves as the abstract base class for defining specific Grants This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*
Subclasses	*ComGrant*, ComOfferServiceGrant
Aggregated by	ARPackage.element

▽

△

Class	Grant (abstract)			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 12.68: Grant

Class	ComGrant (abstract)			
Note	This meta-class serves as the abstract base class for defining specific ComGrants This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	ComEventGrant, ComFieldGrant, ComMethodGrant, ComTriggerGrant			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
remoteSubject	AbstractIamRemote Subject	*	ref	This optional reference defines the remoteSubject that is allowed to access the defined Object via the Grant. **Tags:** atp.Splitkey=remoteSubject
serviceInstance	AdaptivePlatform ServiceInstance	0..1	ref	This reference identifies the applicable AdaptivePlatform ServiceInstance for which the grant applies. **Tags:** atp.Splitkey=serviceInstance

Table 12.69: ComGrant

[constr_10201] Multiplicity of the reference in the role `ComGrant.serviceInstance`

Imposition time: `IT_Mani`

⌈For each `ComGrant`, the reference in the role `serviceInstance` shall exist.⌋

Further information about DDS Security can be found in Chapter 13.5.3.

12.9.1 Com Grant Target-Configuration

12.9.1.1 Com Field Grant Target-Configuration

[TPS_MANI_01237] Semantics of meta-class `ComFieldGrant` ⌈Meta-class `ComFieldGrant` shall be used to award access to a given `field` (identified by means of the reference to meta-class `ServiceFieldDeployment` in the role `serviceDeployment`) in the context of a given `AdaptivePlatformServiceInstance` referenced in the role `serviceInstance`.⌋

In other words, if a given `AdaptivePlatformServiceInstance` and the respective `ServiceFieldDeployment` are not referenced from a `ComFieldGrant` and an `CmModuleInstantiation` exists **then this specific communication shall be suppressed**.

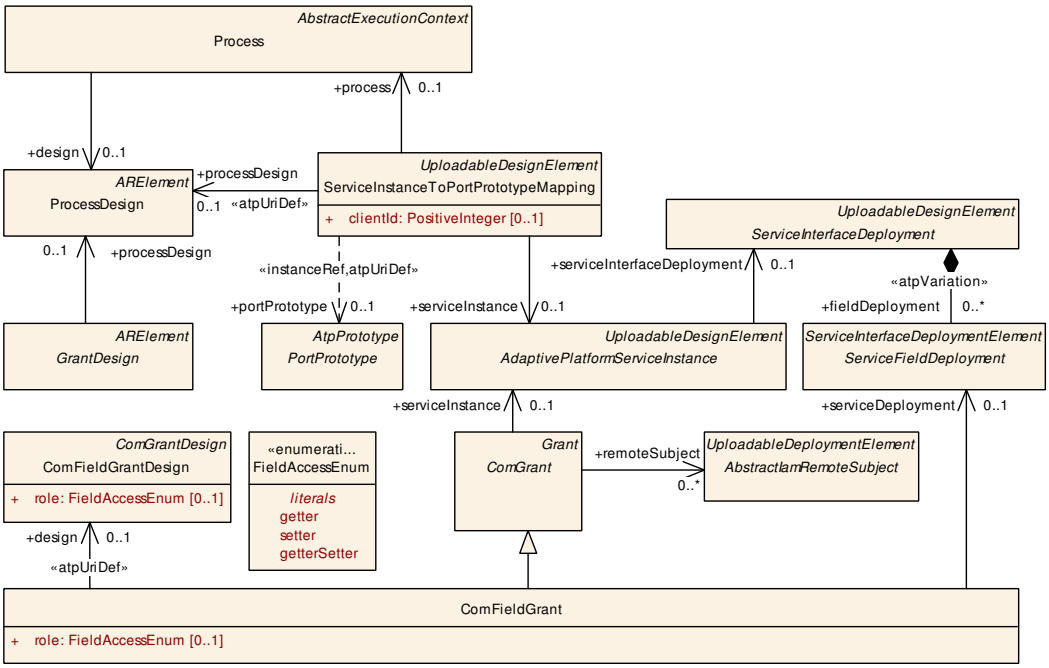


Figure 12.37: Modeling of the `ComFieldGrant`

Class	**ComFieldGrant**			
Note	This meta-class represents the ability to grant access to a ServiceInterface.field. **Tags:** atp.recommendedPackage=Grants This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrant*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ComFieldGrantDesign	0..1	ref	This reference identifies the ComFieldGrantDesign that the enclosing ComFieldGrant was created from. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=design
role	FieldAccessEnum	0..1	attr	This attribute provides the ability to further specify the access to the ServiceInterface.field.
service Deployment	ServiceField Deployment	0..1	ref	This reference identifies the applicable deployment within the context of an AdaptivePlatformServiceInstance for which the grant applies. **Tags:** atp.Splitkey=serviceDeployment

Table 12.70: ComFieldGrant

[constr_10202] Multiplicity of attribute `ComFieldGrant.role`

Imposition time: `IT_Mani`

⌈For each `ComFieldGrant`, the attribute `role` shall exist.⌋

[constr_10203] Multiplicity of the reference in the role ComFieldGrant.serviceDeployment

Imposition time: IT_Mani

⌈For each ComFieldGrant, the reference in the role serviceDeployment shall exist.⌋

12.9.1.2 Com Method Grant Target-Configuration

[TPS_MANI_01238] Semantics of meta-class ComMethodGrant ⌈Meta-class ComMethodGrant shall be used to clear the call of a given method (identified by means of the reference to meta-class ServiceMethodDeployment in the role serviceDeployment) in the context of a given AdaptivePlatformServiceInstance referenced in the role serviceInstance.⌋

In other words, if a given AdaptivePlatformServiceInstance and the respective ServiceMethodDeployment are not referenced from a ComMethodGrant and an CmModuleInstantiation exists **then this specific communication shall be suppressed**.

Class	**ComMethodGrant**			
Note	This meta-class represents the ability to grant access to a ServiceInterface.method. **Tags:** atp.recommendedPackage=Grants This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrant*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ComMethodGrant Design	0..1	ref	This reference identifies the ComMethodGrantDesign that the enclosing ComMethodGrant was created from. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=design
service Deployment	ServiceMethod Deployment	0..1	ref	This reference identifies the applicable deployment within the context of an AdaptivePlatformServiceInstance for which the grant applies. **Tags:** atp.Splitkey=serviceDeployment

Table 12.71: ComMethodGrant

[constr_10204] Multiplicity of the reference in the role ComMethodGrant.serviceDeployment

Imposition time: IT_Mani

⌈For each ComMethodGrant, the reference in the role serviceDeployment shall exist.⌋

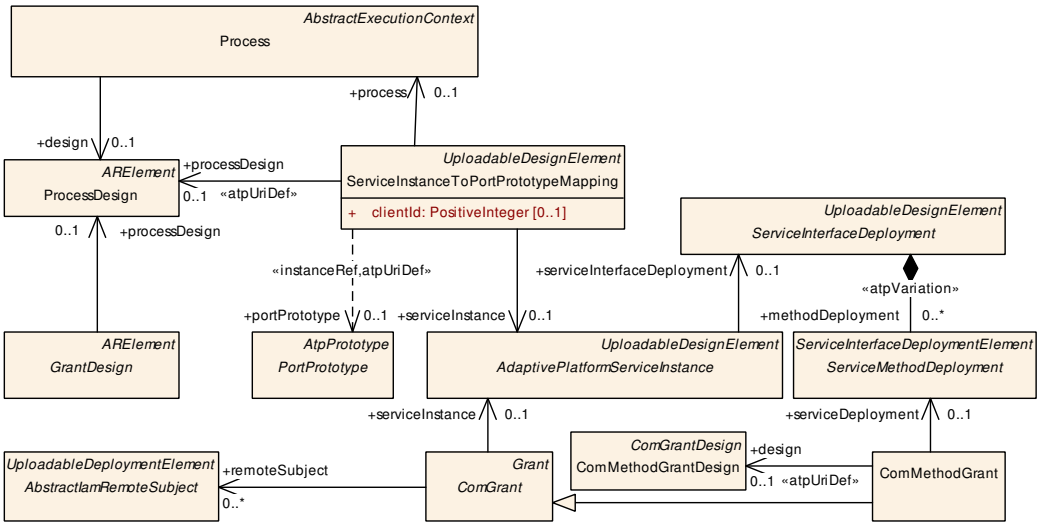


Figure 12.38: Modeling of the `ComMethodGrant`

12.9.1.3 Com Event Grant Target Configuration

[TPS_MANI_01239] Semantics of meta-class `ComEventGrant` ⌈Meta-class `ComEventGrant` shall be used to award access to a given `event` (identified by means of the reference to meta-class `ServiceEventDeployment` in the role `serviceDeployment`) in the context of a given `AdaptivePlatformServiceInstance` referenced in the role `serviceInstance`.⌋

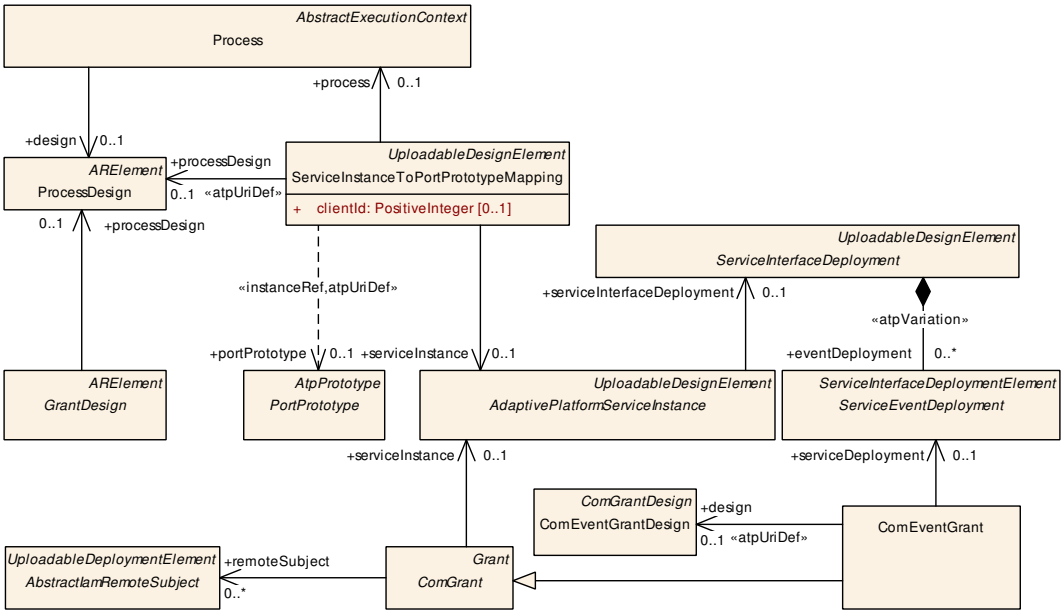


Figure 12.39: Modeling of the `ComEventGrant`

In other words, if a given `AdaptivePlatformServiceInstance` and the respective `ServiceEventDeployment` are not referenced from a `ComEventGrant` and an

CmModuleInstantiation exists **then this specific communication shall be suppressed**.

[constr_5366] Allowed target of ComEventGrant.serviceDeployment reference

Imposition time: IT_Mani

⌈ComEventGrant is allowed to reference only a ServiceEventDeployment in the role serviceDeployment that in turn references a VariableDataPrototype in the role event.⌋

Class	**ComEventGrant**			
Note	This meta-class represents the ability to grant access to a ServiceInterface.event. **Tags:** atp.recommendedPackage=Grants This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrant*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ComEventGrantDesign	0..1	ref	This reference identifies the ComEventGrantDesign that the enclosing ComEventGrant was created from. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=design
service Deployment	ServiceEvent Deployment	0..1	ref	This reference identifies the applicable deployment within the context of an AdaptivePlatformServiceInstance for which the grant applies. **Tags:** atp.Splitkey=serviceDeployment

Table 12.72: ComEventGrant

[constr_10205] Multiplicity of the reference in the role ComEventGrant.serviceDeployment

Imposition time: IT_Mani

⌈For each ComEventGrant, the reference in the role serviceDeployment shall exist.⌋

12.9.1.4 Com Trigger Grant Target-Configuration

[TPS_MANI_03322] Semantics of meta-class ComTriggerGrant ⌈Meta-class ComTriggerGrant shall be used to award access to a given trigger (identified by means of the reference to meta-class ServiceEventDeployment in the role serviceDeployment) in the context of a given AdaptivePlatformServiceInstance referenced in the role serviceInstance.⌋

In other words, if a given AdaptivePlatformServiceInstance and the respective ServiceEventDeployment for a Trigger are not referenced from a ComTriggerGrant and an CmModuleInstantiation exists then this specific communication shall be suppressed.

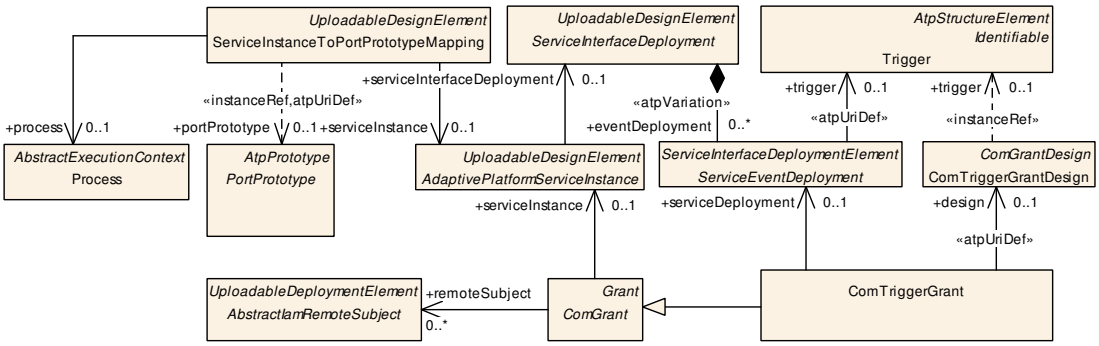


Figure 12.40: Modeling of the `ComTriggerGrant`

Class	**ComTriggerGrant**			
Note	This meta-class represents the ability to grant access to a ServiceInterface.trigger **Tags:** atp.recommendedPackage=Grants This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrant*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ComTriggerGrant Design	0..1	ref	This reference identifies the ComTriggerGrantDesign that the enclosing ComTriggerGrant was created from **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=design
service Deployment	ServiceEvent Deployment	0..1	ref	This reference identifies the applicable deployment within the context of an AdaptivePlatformServiceInstance for which the grant applies. **Tags:** atp.Splitkey=serviceDeployment

Table 12.73: ComTriggerGrant

[constr_5367] Allowed target of `ComTriggerGrant.serviceDeployment` reference

Imposition time: `IT_Mani`

⌈`ComTriggerGrant` is allowed to reference only a `ServiceEventDeployment` in the role `serviceDeployment` that in turn references a `Trigger` in the role `trigger`.⌋

[constr_5368] Existence of the reference in the role `ComTriggerGrant.serviceDeployment`

Imposition time: `IT_Mani`

⌈For each `ComTriggerGrant`, the reference in the role `serviceDeployment` shall exist.⌋

12.9.1.5 Com Offer Service Grant Target-Configuration

The enforcement of service discovery rights is modeled by means of meta-classes ComOfferServiceGrant.

There is no need for a dedicated grant for service discovery on the client side because the existence of one of the dedicated grants to method, field, event, or trigger is taken as the implicit intent to also do service discovery.

[TPS_MANI_01240] Semantics of meta-class ComOfferServiceGrant ⌈Meta-class ComOfferServiceGrant shall be used to award the right to offer the referenced AdaptivePlatformServiceInstance.⌋

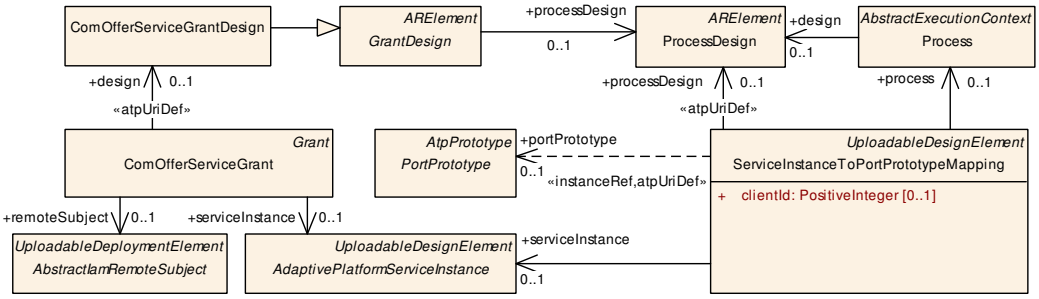


Figure 12.41: Modeling of the ComOfferServiceGrant

Class	**ComOfferServiceGrant**			
Note	This meta-class represents the ability to grant the offering of a service. **Tags:** atp.recommendedPackage=Grants This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ComOfferServiceGrant Design	0..1	ref	This reference identifies the ComOfferServiceGrant Design that the enclosing ComOfferServiceGrant was created from. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=design
remoteSubject	AbstractIamRemote Subject	0..1	ref	This optional reference defines the remoteSubject that is allowed to access the defined Object via the Grant. **Tags:** atp.Splitkey=remoteSubject
serviceInstance	AdaptivePlatform ServiceInstance	0..1	ref	This reference identifies the AdaptivePlatformService Instances for which the grant applies. **Tags:** atp.Splitkey=serviceInstance

Table 12.74: ComOfferServiceGrant

[constr_10206] Multiplicity of the reference in the role ComOfferServiceGrant.serviceInstance

Imposition time: IT_Mani

⌈For each ComOfferServiceGrant, the reference in the role serviceInstance shall exist.⌋

12.9.2 Remote Access Control

The overview of the remote access control was already introduced in Chapter 3.5.2. For the modeling of AbstractIamRemoteSubjects the same approach is used in the Design and in the Target-Configuration. If AbstractIamRemoteSubjects were defined during the design phase, they can be taken over into the target-configuration.

This chapter defines how ComGrants with a defined remoteSubjects shall be interpreted.

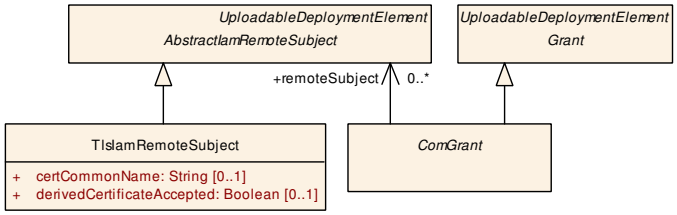


Figure 12.42: Modeling of the relation of ComGrants to the corresponding RemoteSubjects

[TPS_MANI_03245] Definition of ComMethodGrant.remoteSubjects on server side ⌈If the ComMethodGrant references a ProvidedApServiceInstance in the role serviceInstance then the ComMethodGrant.remoteSubject reference defines the remoteSubjects that are allowed to call the defined method that is referenced by the ComMethodGrant in the role serviceDeployment.⌋

[TPS_MANI_03246] Definition of ComMethodGrant.remoteSubjects on client side ⌈If the ComMethodGrant references a RequiredApServiceInstance in the role serviceInstance then the ComMethodGrant.remoteSubject reference defines the remoteSubjects to which a ServiceMethodDeployment.method call is allowed to be sent.⌋

[TPS_MANI_03247] Definition of ComEventGrant.remoteSubjects on provider side ⌈If the ComEventGrant references a ProvidedApServiceInstance in the role serviceInstance then the ComEventGrant.remoteSubject reference defines the remoteSubjects to which an ServiceEventDeployment.event is allowed to be sent.⌋

[TPS_MANI_03248] Definition of ComEventGrant.remoteSubjects on receiver side ⌈If the ComEventGrant references a RequiredApServiceInstance in the

role serviceInstance then the ComEventGrant.remoteSubject reference defines the remoteSubjects from which an ServiceEventDeployment.event is allowed to be received.⌋

[TPS_MANI_03249] Definition of ComFieldGrant.remoteSubjects on provider side ⌈If the ComFieldGrant references a ProvidedApServiceInstance in the role serviceInstance then the ComFieldGrant.remoteSubject reference defines the remoteSubjects that are allowed to access the Field that is referenced by the ServiceFieldDeployment that in turn is referenced by the ComFieldGrant in the role serviceDeployment. This allows the following communication between the local Machine and the AbstractIamRemoteSubject:

- the Field notifier is allowed to be sent to remoteSubjects,
- if ComFieldGrant.role equals setter or getterSetter, then the setter call is allowed to be received from the remoteSubjects,
- if ComFieldGrant.role equals setter or getterSetter, then the setter return is allowed to be sent to remoteSubjects,
- if ComFieldGrant.role equals getter or getterSetter, then the getter call is allowed to be received from the remoteSubjects,
- if ComFieldGrant.role equals getter or getterSetter, then the getter return is allowed to be sent to remoteSubjects.

⌋

[TPS_MANI_03250] Definition of ComFieldGrant.remoteSubjects on client side ⌈If the ComFieldGrant references a RequiredApServiceInstance in the role serviceInstance then the ComFieldGrant.remoteSubject reference defines the remoteSubjects that are allowed to provide the Field for access that is referenced by the ServiceFieldDeployment that in turn is referenced by the ComFieldGrant in the role serviceDeployment. This will allow the following communication between the local Machine and the AbstractIamRemoteSubject:

- the Field notifier is allowed to be received from remoteSubjects,
- if ComFieldGrant.role equals setter or getterSetter, then the setter call is allowed to be sent to the remoteSubjects,
- if ComFieldGrant.role equals setter or getterSetter, then the setter return is allowed to be received from remoteSubjects,
- if ComFieldGrant.role equals getter or getterSetter, the getter call is allowed to be sent to the remoteSubjects,
- if ComFieldGrant.role equals getter or getterSetter, the getter return is allowed to be received from remoteSubjects.

⌋

A ComGrant with the remoteSubject reference defines that a remote access control for this ComGrant will be performed. Such a ComGrant with the remoteSubject reference does not enforce any Machine local access restrictions.

To enforce local access restrictions for the same ServiceInterface element of the same AdaptivePlatformServiceInstance an additional ComGrant needs to be defined that points to the same serviceInstance and to the same ComFieldGrant.serviceDeployment or ComEventGrant.serviceDeployment or ComMethodGrant.serviceDeployment but does not contain the remoteSubject reference.

12.10 Crypto Target-Configuration

Disclaimer

As of AUTOSAR R26-11, the canonical approach for the target configuration of the crypto functional cluster is by usage of the APMC model documented in [18, SWS Cryptography] Chapter 10.1. The description of target configuration via the M2 model (as described in this chapter) is still provided, but may be discontinued at a later point in time.

This chapter explains the configuration of the Crypto functional cluster and the interaction of application Software with the [18, SWS Cryptography].

[TPS_MANI_03260] Semantics of meta-class CryptoModuleInstantiation ⌈The representation of the Crypto functional cluster [18, SWS Cryptography] within one specific Machine is defined by the CryptoModuleInstantiation.⌋

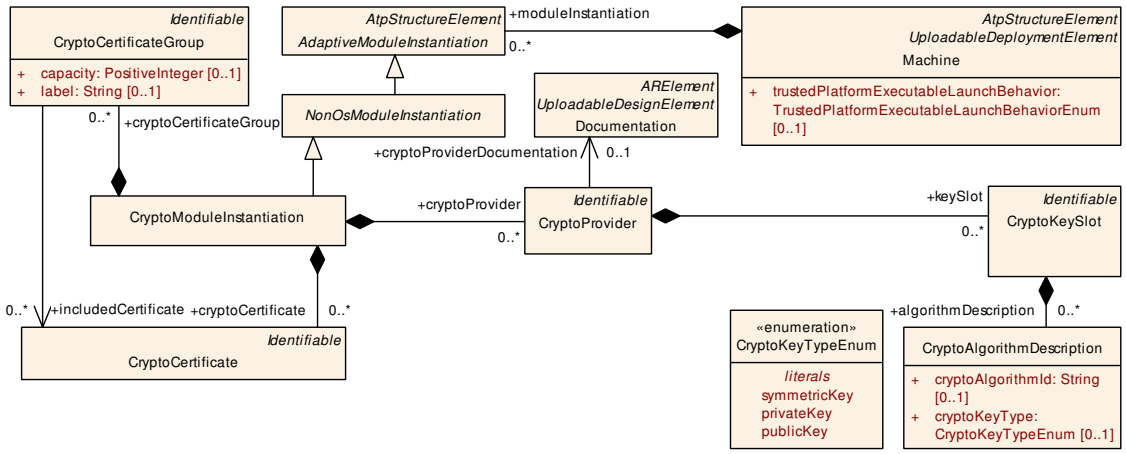


Figure 12.43: CryptoModuleInstantiation Overview

Class	CryptoModuleInstantiation			
Note	This meta-class defines the configuration for the Crypto stack on a specific machine. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
crypto Certificate	CryptoCertificate	*	aggr	List of CryptoCertificates managed in the CryptoStack **Tags:** atp.Splitkey=cryptoCertificate, crypto Certificate.shortName
crypto Certificate Group	CryptoCertificateGroup	*	aggr	List of CryptoCertificateGroups managed in the Crypto Stack. **Tags:** atp.Splitkey=cryptoCertificateGroup, cryptoCertificate Group.shortName atp.Status=candidate
cryptoProvider	CryptoProvider	*	aggr	List of CryptoProviders provided by the CryptoStack **Tags:** atp.Splitkey=cryptoProvider, cryptoProvider.short Name

Table 12.75: CryptoModuleInstantiation

12.10.1 Crypto Provider

[TPS_MANI_03261] Support of `CryptoProvider`s ⌈The Crypto functional cluster is able to support multiple `CryptoProvider`s. Each `CryptoProvider` implements a software- or a hardware-based cryptographic library.⌋

Class	CryptoProvider			
Note	CryptoProvider implements cryptographic primitives (algorithms) supported by the stack. Implementation of this component may be software or hardware based (HSM/TPM). This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	CryptoModuleInstantiation.cryptoProvider			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
cryptoProvider Documentation	Documentation	0..1	ref	Documentation of the CryptoProvider that describes the implemented cryptographic primitives. **Tags:** atp.Splitkey=cryptoProviderDocumentation
keySlot	CryptoKeySlot	*	aggr	This aggregation represents the key slots that are allocated by the CryptoProvider. **Tags:** atp.Splitkey=keySlot, keySlot.shortName

Table 12.76: CryptoProvider

Each `CryptoProvider` can be described in more detail with the `cryptoProvider-Documentation` that uses the `Documentation` element to provide means for a more detailed description.

[TPS_MANI_03262] Semantics of `CryptoProviderToPortPrototypeMapping` ⌈Meta-class `CryptoProviderToPortPrototypeMapping` has the ability to map

a specific PortPrototype referenced in the role portPrototype to a CryptoProvider referenced in the role cryptoProvider.

The mapping also comprises a reference to meta-class process in order to accommodate for the fact that identical combinations of cryptoProvider and portPrototype may or may not apply for a given Process that represents the enclosing Executable at runtime.⌋

Class	CryptoProviderToPortPrototypeMapping			
Note	This meta-class represents the ability to define a mapping between a CryptoProvider on deployment level to a given PortPrototype that is typed by a CryptoProviderInterface. **Tags:** atp.recommendedPackage=CryptoProviderToPortPrototypeMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
cryptoProvider	CryptoProvider	0..1	ref	This reference represents the mapped cryptoProvider. **Tags:** atp.Splitkey=cryptoProvider
portPrototype	RPortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **Tags:** atp.Splitkey=portPrototype, portPrototype.context RootSwComponentPrototype, portPrototype.context ComponentPrototype, portPrototype.targetRPort Prototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference represents the process required as context for the mapping. **Tags:** atp.Splitkey=process

Table 12.77: CryptoProviderToPortPrototypeMapping

[constr_10207] Multiplicity of the reference in the role CryptoProviderToPortPrototypeMapping.cryptoProvider

Imposition time: IT_Mani

⌈For each CryptoProviderToPortPrototypeMapping, the reference in the role cryptoProvider shall exist.⌋

[constr_10208] Multiplicity of the reference in the role CryptoProviderToPortPrototypeMapping.process

Imposition time: IT_Mani

⌈For each CryptoProviderToPortPrototypeMapping, the reference in the role process shall exist.⌋

[constr_5240] Restriction applicable for `CryptoProviderToPortPrototype-Mapping.portPrototype`

Status: DRAFT
Imposition time: `IT_Mani`

⌈The reference `CryptoProviderToPortPrototypeMapping.portPrototype` shall only be used for an `RPortPrototype` typed by a `CryptoProvider-Interface`.⌋

The application developer uses the `ara::core::InstanceSpecifier` as local identifier in the API call that represents the path to the modeled `RPortPrototype`. The Integrator maps the `RPortPrototype` in the target-configuration model with the `CryptoProviderToPortPrototypeMapping` to the concrete `CryptoProvider`.

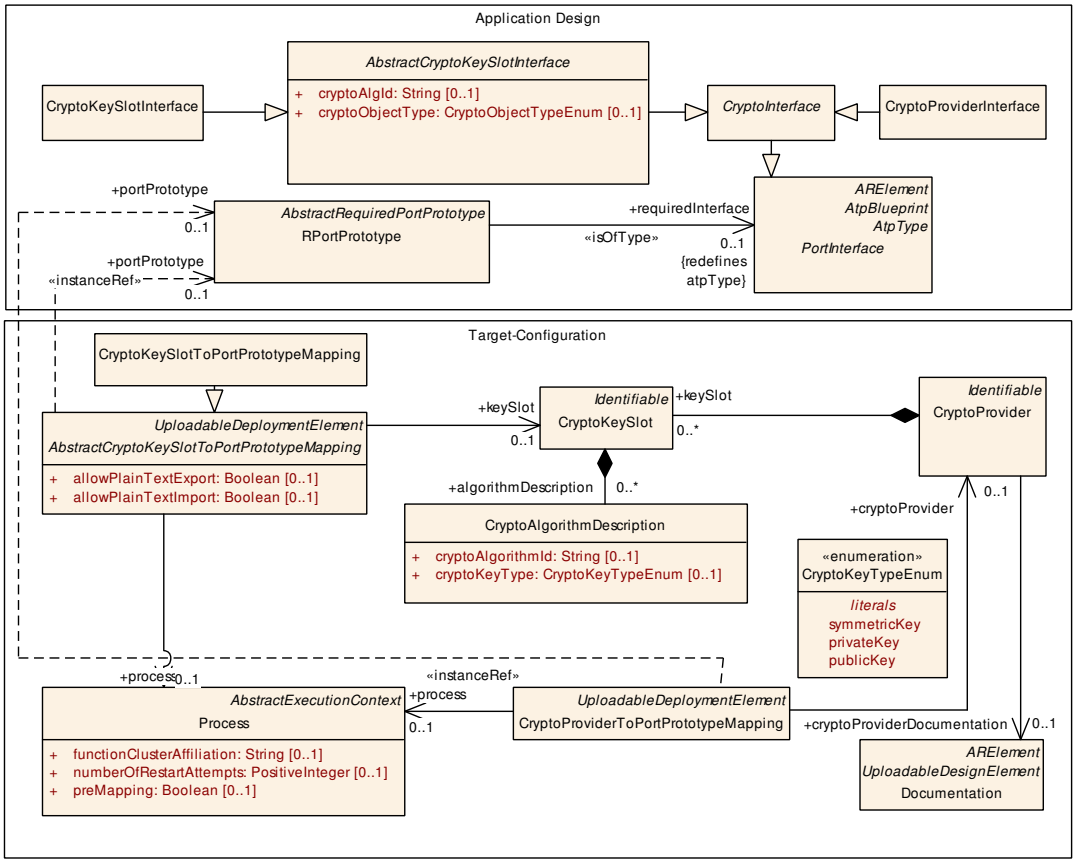


Figure 12.44: Mapping of CryptoProviders and CryptoKeySlots to RPortPrototypes

12.10.2 Crypto Key Slot

The Key Storage Provider is responsible for the storage of different type of key material. The crypto objects that are stored by the Key Storage Provider are represented as `CryptoKeySlot`s in the model.

[TPS_MANI_03263] Assignment of CryptoKeySlots to CryptoProviders
⌈Crypto objects that are used by the CryptoProvider are described by the CryptoKeySlots that are aggregated by the CryptoProvider in the role keySlot.⌋

Class	**CryptoKeySlot**			
Note	This meta-class represents the ability to define a concrete key to be used for a crypto operation. **Tags:** atp.ManifestKind=MachineManifest This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	CryptoProvider.keySlot			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
algorithm Description	CryptoAlgorithm Description	*	aggr	This aggregation contains the collection of crypto algorithm descriptions that can be used in the context of the enclosing crypto key slot. **Tags:** atp.Splitkey=algorithmDescription atp.Status=candidate
cryptoKeySlot Design	CryptoKeySlotDesign	0..1	ref	This reference identifies the CryptoKeySlotDesign from which the referencing CryptoKeySlot was derived. **Tags:** atp.Splitkey=cryptoKeySlotDesign

Table 12.78: CryptoKeySlot

Class	**CryptoAlgorithmDescription**			
Note	This meta-class is used to provide properties of the crypto algorithm. **Tags:** atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	CryptoKeySlot.algorithmDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cryptoAlgorithm Id	String	0..1	attr	This attribute describes the identity of the crypto algorithm associated with the key stored in this key slot
cryptoKeyType	CryptoKeyTypeEnum	0..1	attr	This attribute describes the type of the cryptographic key to be stored in this key slot.

Table 12.79: CryptoAlgorithmDescription

[constr_10690] Existence of attribute CryptoAlgorithmDescription.cryptoAlgorithmId

Imposition time: IT_Mani

⌈For each CryptoAlgorithmDescription, attribute cryptoAlgorithmId shall exist.⌋

[constr_10691] Existence of CryptoAlgorithmDescription.cryptoKeyType

Imposition time: IT_Mani

⌈For each CryptoAlgorithmDescription, attribute cryptoKeyType shall exist.⌋

Enumeration	**CryptoKeyTypeEnum**
Note	This enumeration defines the type of a cryptographic key. **Tags:** atp.Status=candidate This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	CryptoAlgorithmDescription.cryptoKeyType
Literal	***Description***
privateKey	This denotes a private key. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=candidate
publicKey	This denotes a public key. **Tags:** atp.EnumerationLiteralIndex=2 atp.Status=candidate
symmetricKey	This denotes a symmetric key. **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=candidate

Table 12.80: CryptoKeyTypeEnum

The application developer uses the `ara::core::InstanceSpecifier` as local identifier in the API call that represents the path to the modeled `RPortPrototype`. The Integrator maps the `RPortPrototype` in the target-configuration model with the `CryptoKeySlotToPortPrototypeMapping` resp. `CryptoKeySlotToClient-PortPrototypeMapping` to the concrete `CryptoKeySlot` that is stored by the Key Storage Provider.

The information from the target-configuration model is therefore used to replace the local identifier from the Application Design by the concrete `CryptoKeySlot`.

12.10.2.1 Crypto Key Slot to Port Prototype Mapping

[TPS_MANI_03264] Semantics of `CryptoKeySlotToPortPrototypeMapping`

Status: DRAFT

⌈Meta-class `CryptoKeySlotToPortPrototypeMapping` has the ability to map a specific `PortPrototype` typed by a `CryptoKeySlotInterface` referenced in the role `portPrototype` to a `CryptoKeySlot` referenced in the role `keySlot`.

The mapping also comprises a reference to meta-class `process` in order to accommodate for the fact that identical combinations of `keySlot` and `portPrototype` may or may not apply for a given `Process` that represents the enclosing `Executable` at runtime.

The usage of a `CryptoKeySlotToPortPrototypeMapping` indicates that the mapped `CryptoKeySlot` can be accessed for **reading and writing**.⌋

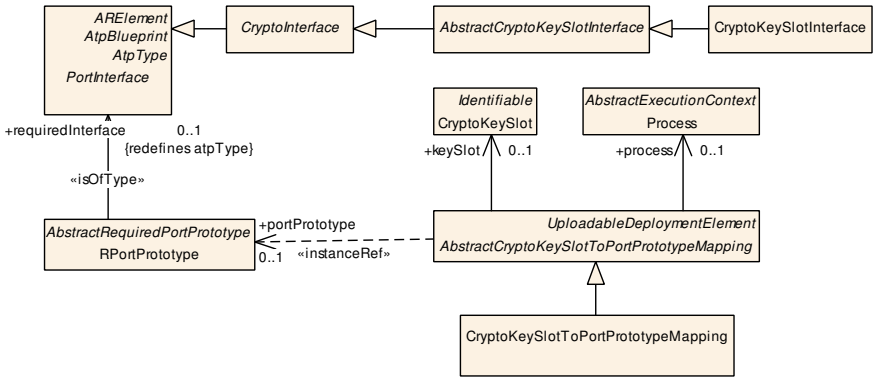


Figure 12.45: Modeling of `CryptoKeySlotToPortPrototypeMapping`

Class	***AbstractCryptoKeySlotToPortPrototypeMapping*** (abstract)			
Note	This abstract meta-class acts as a base class for crypto key slot mappings. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	CryptoKeySlotToClientPortPrototypeMapping, CryptoKeySlotToPortPrototypeMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allowPlainText Export	Boolean	0..1	attr	There are two conditions for a key to become exportable in plain text: 1. this attribute needs to be set to true. 2. the API call for using the key provides an argument that explicitly supports the usage of the API for plain text export.
allowPlainText Import	Boolean	0..1	attr	This attribute describes whether the key slot content is allowed to be imported in plaintext format or not by the application referenced in the role process. Plaintext import is only possible if supported by the cryptographic module and by the CryptoProvider implementation.
keySlot	CryptoKeySlot	0..1	ref	This reference represents the mapped CryptoKeySlot. **Tags:** atp.Splitkey=keySlot
portPrototype	RPortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **Tags:** atp.Splitkey=portPrototype, portPrototype.context RootSwComponentPrototype, portPrototype.context ComponentPrototype, portPrototype.targetRPort Prototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference represents the process required as context for the mapping. **Tags:** atp.Splitkey=process

Table 12.81: AbstractCryptoKeySlotToPortPrototypeMapping

[constr_10209] Existence of the reference in the role AbstractCryptoKeySlotToPortPrototypeMapping.keySlot

Imposition time: IT_Mani

⌈For each AbstractCryptoKeySlotToPortPrototypeMapping, the reference in the role keySlot shall exist.⌋

[constr_10210] Existence of the reference in the role AbstractCryptoKeySlotToPortPrototypeMapping.process

Imposition time: IT_Mani

⌈For each AbstractCryptoKeySlotToPortPrototypeMapping, the reference in the role process shall exist.⌋

Class	**CryptoKeySlotToPortPrototypeMapping**			
Note	This meta-class represents the ability to define a mapping between a CryptoKeySlot on target-configuration level to a given PortPrototype that is typed by a CryptoKeySlotInterface. **Tags:** atp.recommendedPackage=CryptoKeySlotMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractCryptoKeySlotToPortPrototypeMapping*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 12.82: CryptoKeySlotToPortPrototypeMapping

[constr_10754] Restriction applicable for CryptoKeySlotToPortPrototypeMapping.portPrototype

Status: DRAFT

Imposition time: IT_Mani

⌈The reference CryptoKeySlotToPortPrototypeMapping.portPrototype shall only be used for an RPortPrototype typed by a CryptoKeySlotInterface.⌋

12.10.2.2 Crypto Key Slot to Client Port Prototype Mapping

[TPS_MANI_01471] Semantics of meta-class CryptoKeySlotToClientPortPrototypeMapping

Status: DRAFT

⌈Meta-class CryptoKeySlotToClientPortPrototypeMapping has the ability to map a specific PortPrototype typed by a CryptoKeySlotClientInterface referenced in the role portPrototype to a CryptoKeySlot referenced in the role keySlot.

The mapping also comprises a reference to meta-class `process` in order to accommodate for the fact that identical combinations of `keySlot` and `portPrototype` may or may not apply for a given `Process` that represents the enclosing `Executable` at runtime.

The usage of a `CryptoKeySlotToClientPortPrototypeMapping` indicates that the mapped `CryptoKeySlot` can be accessed for **reading only**.⌋

[constr_10746] Existence of primitive attribute `CryptoKeySlotToClientPortPrototypeMapping.allowPlainTextImport`

Status: DRAFT

Imposition time: `IT_Mani`

⌈For each `CryptoKeySlotToClientPortPrototypeMapping`, the attribute `allowPlainTextImport` shall either not exist or be set to the value `false`.⌋

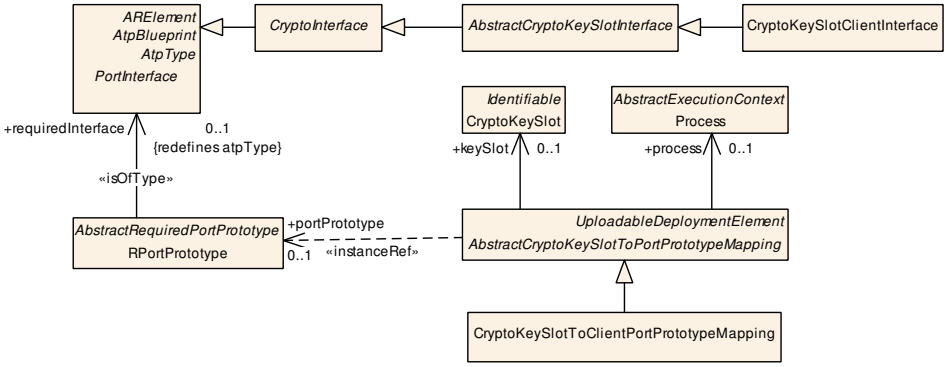


Figure 12.46: Modeling of `CryptoKeySlotToClientPortPrototypeMapping`

Class	**CryptoKeySlotToClientPortPrototypeMapping**			
Note	This meta-class represents the ability to define a mapping between a CryptoKeySlot on deployment level to a given PortPrototype that is typed by a CryptoKeyClientSlotInterface (this means only read is supported). **Tags:** atp.recommendedPackage=CryptoKeySlotMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractCryptoKeySlotToPortPrototypeMapping*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 12.83: CryptoKeySlotToClientPortPrototypeMapping

[constr_10755] Restriction applicable for `CryptoKeySlotToClientPortPrototypeMapping.portPrototype`

Status: DRAFT
Imposition time: `IT_Mani`

⌈The reference `CryptoKeySlotToClientPortPrototypeMapping.portPrototype` shall only be used for an `RPortPrototype` typed by a `CryptoKeySlotClientInterface`.⌋

12.10.3 Crypto Certificate

[TPS_MANI_03265] Support of `CryptoCertificate`s ⌈Certificates stored by the Certificate Management Provider that is available in the Crypto functional cluster are modeled as `CryptoCertificate`s.⌋

Class	**CryptoCertificate**			
Note	This meta-class represents the ability to model a cryptographic certificate. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	CryptoModuleInstantiation.cryptoCertificate			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 12.84: CryptoCertificate

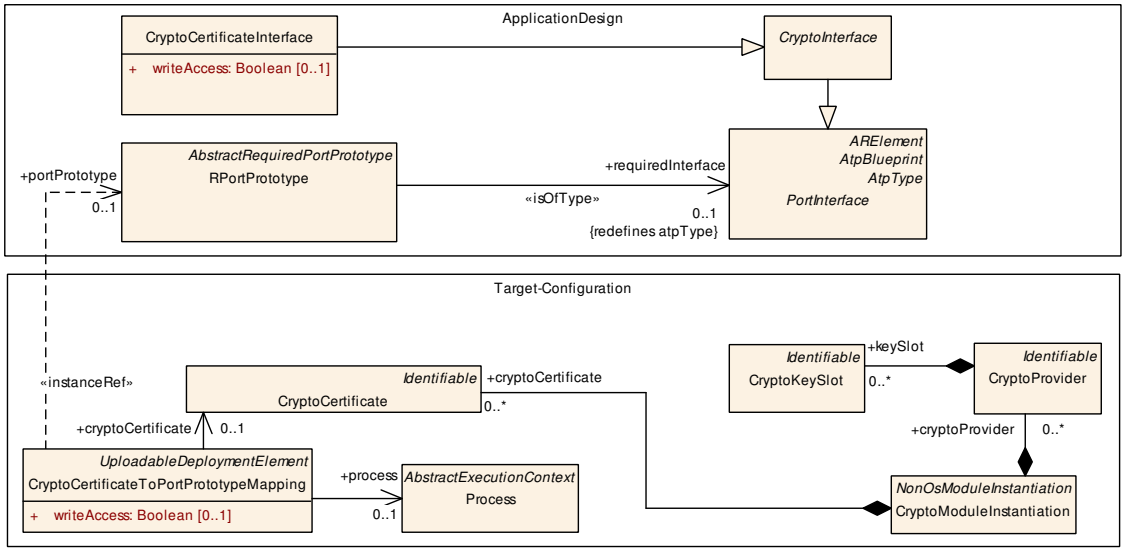


Figure 12.47: Mapping of CryptoCertificates to RPortPrototypes

[TPS_MANI_03267] Semantics of `CryptoCertificateToPortPrototypeMapping` ⌈Meta-class `CryptoCertificateToPortPrototypeMapping` has the ability to map a specific `PortPrototype` referenced in the role `portPrototype` to a `CryptoCertificate` referenced in the role `cryptoCertificate`.

The mapping also comprises a reference to meta-class process in order to accommodate for the fact that identical combinations of cryptoCertificate and portPrototype may or may not apply for a given Process that represents the enclosing Executable at runtime.⌋

Class	CryptoCertificateToPortPrototypeMapping			
Note	This meta-class represents the ability to define a mapping between a CryptoCertificate on target-configuration level to a given PortPrototype that is typed by a CryptoCertificateInterface. **Tags:** atp.recommendedPackage=CryptoCertificateToPortPrototypeMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
crypto Certificate	CryptoCertificate	0..1	ref	This reference represents the mapped cryptoCertificate. **Tags:** atp.Splitkey=cryptoCertificate
portPrototype	RPortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **Tags:** atp.Splitkey=portPrototype, portPrototype.context RootSwComponentPrototype, portPrototype.context ComponentPrototype, portPrototype.targetRPort Prototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference represents the process required as context for the mapping. **Tags:** atp.Splitkey=process
writeAccess	Boolean	0..1	attr	This attribute defines whether the application has write-access to the CryptoCertificate (true) or only read-access (false).

Table 12.85: CryptoCertificateToPortPrototypeMapping

[constr_5242] Restriction applicable for the reference CryptoCertificateToPortPrototypeMapping.portPrototype

Status: DRAFT
Imposition time: IT_Mani

⌈The reference in the role CryptoCertificateToPortPrototypeMapping.portPrototype shall only be used for an RPortPrototype typed by a CryptoCertificateInterface.⌋

The application developer uses the ara::core::InstanceSpecifier as local identifier in the API call that represents the path to the modeled RPortPrototype. The Integrator maps the RPortPrototype in the target-configuration model with the CryptoCertificateToPortPrototypeMapping to the concrete CryptoCertificate that is stored by the Certificate Management Provider.

The information from the target-configuration model is therefore used to replace the local identifier from the Application Design by the concrete CryptoCertificate.

12.10.4 Crypto Certificate Group

[TPS_MANI_01485] Support of CryptoCertificateGroups ⌈It is possible to model a CryptoCertificateGroup that represents a storage allocation for crypto certificates that can **dynamically** be associated with the CryptoCertificate-Group.

Optionally, it is also possible to include CryptoCertificates into a Crypto-CertificateGroup by reference in the role includedCertificate.⌋

The modeling of the CryptoCertificateGroup and its relations is depicted in Figure 12.43.

The modeling of the CryptoCertificateGroup tries to accommodate two different perspectives. On the one hand, the CryptoCertificateGroup was created to allow for access to cryptographic certificates with a collection-semantics.

The identity of the individual cryptographic certificates in the CryptoCertificate-Group is not relevant for users of the CryptoCertificateGroup.

On the other hand, there is the second perspective from the management point of view. Certificates may need to be replaced and for this purpose need to have an identity.

In order to include a CryptoCertificate in a CryptoCertificateGroup and – at the same time – be able to access the CryptoCertificate by identity it is necessary to include the CryptoCertificate by reference into the CryptoCertificate-Group.

The existence of the reference in the role CryptoCertificateGroup.included-Certificate is entirely optional.

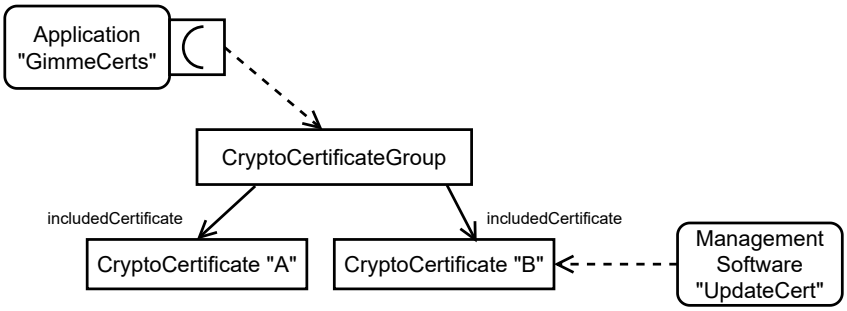


Figure 12.48: Different perspectives on the CryptoCertificateGroup

Class	**CryptoCertificateGroup**
Note	This meta-class represents the ability to model a group of cryptographic certificates. **Tags:** atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	**CryptoCertificateGroup**			
Aggregated by	CryptoModuleInstantiation.cryptoCertificateGroup			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
capacity	PositiveInteger	0..1	attr	Capacity of the certificate group in bytes of DER serialized certificates. **Tags:** atp.Status=candidate
included Certificate	CryptoCertificate	*	ref	This optional collection of references to individual Crypto CErtificates can be used to include some Crypto Certificates "by identity" into the CryptoCertificateGroup. **Tags:** atp.Splitkey=includedCertificate atp.Status=candidate
label	String	0..1	attr	Label for application-level identification of certificate group. **Tags:** atp.Status=candidate

Table 12.86: CryptoCertificateGroup

[TPS_MANI_01517] Semantics of meta-class `CryptoCertificateGroupToPortPrototypeMapping` ⌈The meta-class `CryptoCertificateGroupToPortPrototypeMapping` has the ability to map a specific `PortPrototype` referenced in the role `portPrototype` to a `CryptoCertificateGroup` referenced in the role `certificateGroup`.

The mapping also comprises a reference to meta-class `process` in order to accommodate for the fact that identical combinations of `certificateGroup` and `portPrototype` may or may not apply for a given `Process` that represents the enclosing `Executable` at runtime.⌋

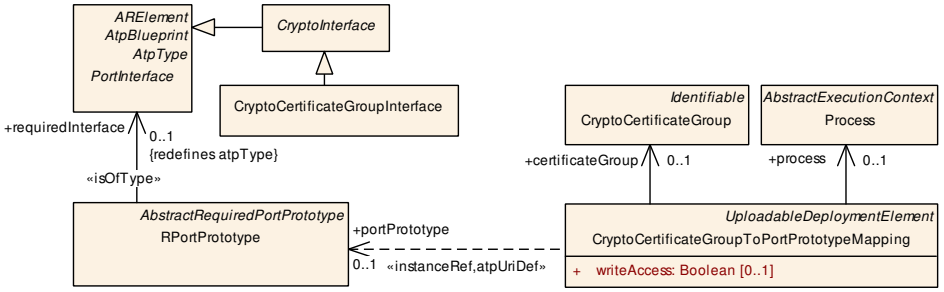


Figure 12.49: Modeling of `CryptoCertificateGroupToPortPrototypeMapping`

Class	**CryptoCertificateGroupToPortPrototypeMapping**
Note	This meta-class represents the ability to define a mapping between a CryptoCertificateGroup on target-configuration level to a given PortPrototype that is typed by a CryptoCertificateGroupInterface. **Tags:** atp.Status=candidate atp.recommendedPackage=CryptoMappipngs This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*

▽

△

Class	CryptoCertificateGroupToPortPrototypeMapping			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
certificateGroup	CryptoCertificateGroup	0..1	ref	This reference identifies the refernce group mapped to the PortPrototype. **Tags:** atp.Splitkey=certificateGroup atp.Status=candidate
portPrototype	RPortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=portPrototype, portPrototype.contextRootSwComponentPrototype, portPrototype.contextComponentPrototype, portPrototype.targetRPortPrototype atp.Status=candidate **InstanceRef implemented by:** RPortPrototypeInExecutableInstanceRef
process	Process	0..1	ref	This reference represents the process required as context for the mapping.This reference represents the process required as context for the mapping. **Tags:** atp.Splitkey=process atp.Status=candidate
writeAccess	Boolean	0..1	attr	This attribute defines whether the application has write-access to the CryptoCertificateGroup (true) or only read-access (false).

Table 12.87: CryptoCertificateGroupToPortPrototypeMapping

[constr_10736] Restriction applicable for CryptoCertificateGroupToPortPrototypeMapping.portPrototype

Status: DRAFT
Imposition time: IT_Mani

⌈The reference CryptoCertificateGroupToPortPrototypeMapping.portPrototype shall only be used for an RPortPrototype typed by a CryptoCertificateGroupInterface.⌋

[constr_10737] Existence of reference in the role CryptoCertificateGroupToPortPrototypeMapping.process

Imposition time: IT_Mani

⌈For each CryptoCertificateGroupToPortPrototypeMapping, the reference in the role process shall exist.⌋

[constr_10738] Existence of reference in the role CryptoCertificateGroupToPortPrototypeMapping.certificateGroup

Imposition time: IT_Mani

⌈For each CryptoCertificateGroupToPortPrototypeMapping, the reference in the role certificateGroup shall exist.⌋

[constr_10739] Existence of instanceRef in the role CryptoCertificateGroupToPortPrototypeMapping.portPrototype

Imposition time: IT_Mani

⌈For each CryptoCertificateGroupToPortPrototypeMapping, the instanceRef in the role portPrototype shall exist.⌋

12.10.5 Relation to Crypto System Design

Thanks to the existence of the crypto system design (which is described in Chapter 7.5), it is possible to trace the existence of a CryptoKeySlot up to the design level.

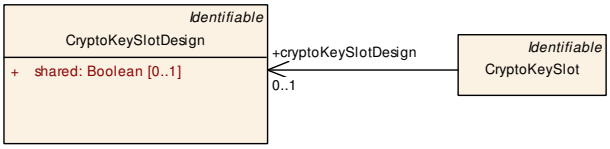


Figure 12.50: CryptoKeySlot traces to CryptoKeySlotDesign

[TPS_MANI_01472] Semantics of reference CryptoKeySlot.cryptoKeySlotDesign ⌈The reference in the role CryptoKeySlot.cryptoKeySlotDesign carries no actual semantics; it exists only as a trace from the CryptoKeySlot to a CryptoKeySlotDesign, indicating that the CryptoKeySlot was created out of the CryptoKeySlotDesign.⌋

12.11 IdsM Target-Configuration

Disclaimer

As of AUTOSAR R26-11, the canonical approach for the target configuration of the intrusion detection system manager (IdsM) is by usage of the APMC model documented in [39, SWS IntrusionDetectionSystemManager] Chapter 10.1. The description of target configuration via the M2 model (as described in this chapter) is still provided, but may be discontinued at a later point in time.

12.11.1 IdsM Instantiation

The definition of the target-configuration for the Intrusion Detection System Manager (IdsM) is modeled by means of the meta-class IdsmModuleInstantiation

Class	**IdsmModuleInstantiation**			
Note	This meta-class defines the attributes for the IdsM configuration on a specific machine. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *IdsPlatformInstantiation*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
idsmInstance	IdsmInstance	0..1	ref	This reference identifies the IdsmInstance that is associated with the enclosing IdsmModuleInstantiation. **Tags:** atp.Splitkey=idsmInstance
queueSize	PositiveInteger	0..1	attr	Length of the queue in which IDS events can be stored before processing, if immediate processing is not possible.
reportable SecurityEvent	SecurityEventMapping	*	ref	Collection of reportable instances of security events. **Tags:** atp.Splitkey=reportableSecurityEvent

Table 12.88: IdsmModuleInstantiation

[constr_10021] Existence of `IdsmModuleInstantiation`

Imposition time: `IT_Mani`

⌈On each `Machine`, only one instance of the `Intrusion Detection System Manager` (modeled by `IdsmModuleInstantiation`) shall exist.⌋

This instance manages all the reported `SEv`s created by `SWCL`s or `Function Clusters` on this `Adaptive Machine`.

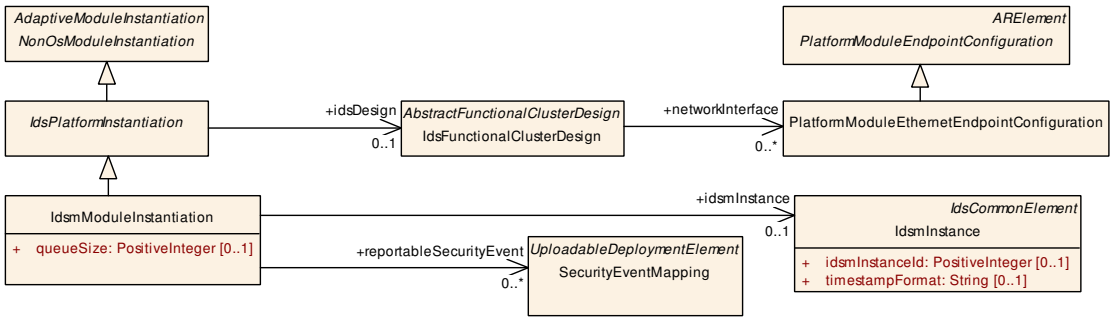


Figure 12.51: Specification of the `IdsmModuleInstantiation`

[TPS_MANI_03345] Semantics of `IdsmModuleInstantiation.reportableSecurityEvent`

Status: DRAFT

⌈The definition of `SEv`s that are managed by the `IdsmModuleInstantiation` is defined by the `IdsmModuleInstantiation.reportableSecurityEvent` reference.⌋

Meta-class `IdsmModuleInstantiation` is derived from the meta-class `IdsPlatformInstantiation` which acts as an abstract base class for all platform modules that contribute to the implementation of the Intrusion Detection System.

Class	***IdsPlatformInstantiation*** (abstract)			
Note	This meta-class acts as an abstract base class for platform modules that implement the intrusion detection system. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Subclasses	IdsmModuleInstantiation			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
idsDesign	IdsFunctionalClusterDesign	0..1	ref	Reference to the Ids Design that this IdsPlatformInstantiation implements. **Tags:** atp.Splitkey=idsDesign
idsTimeBase	TimeBaseResource	*	ref	This reference identifies the applicable time base resource. **Tags:** atp.Splitkey=idsTimeBase

Table 12.89: IdsPlatformInstantiation

[constr_9433] Existence of reference IdsPlatformInstantiation.idsDesign

Imposition time: IT_Mani

⌈For each IdsPlatformInstantiation, the reference idsDesign shall exist.⌋

Each IdsPlatformInstantiation element contains a reference to the corresponding MachineDesign element that defines conditions that the IdsPlatformInstantiation needs to implement.

[advisory_01017] Existence of the IdsFunctionalClusterDesign.networkInterface if IdsmModuleInstantiation.reportableSecurityEvent exists

Imposition time: IT_Mani

⌈If the reference in the role IdsmModuleInstantiation.reportableSecurityEvent exists at least once, then the IdsmModuleInstantiation should also define a reference to a IdsFunctionalClusterDesign that in turn defines a IdsFunctionalClusterDesign.networkInterface.⌋

[constr_10517] Upper multiplicity of the reference in the role IdsPlatformInstantiation.idsTimeBase

Imposition time: IT_Mani

⌈In the context of IdsPlatformInstantiation, the reference in the role idsTimeBase shall exist at most once.⌋

One of the tasks of an IdsM is to obtain a custom time stamp that corresponds to reported security events under certain circumstances (which are not relevant for the sake of discussing the modeling).

AUTOSAR supports two ways to obtain a time stamp for the purpose of *Intrusion Detection System Management*, but only one of these ways shall be used at any given time. This restriction is formalized in [constr_10501].

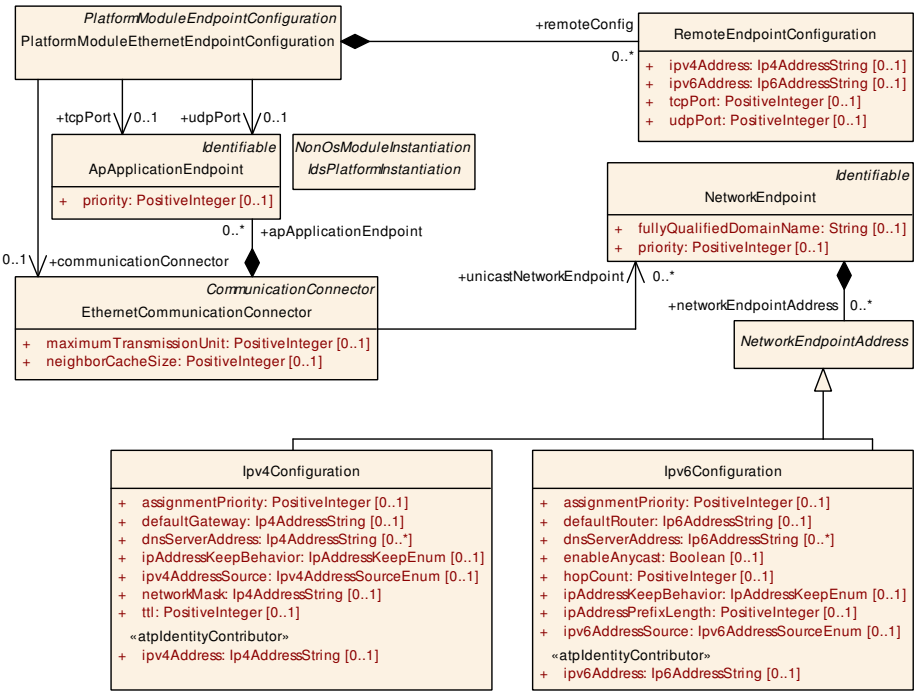


Figure 12.52: `PlatformModuleEthernetEndpointConfiguration` for transmission of security events

[constr_10501] Access to time stamps for *Intrusion Detection System Management*

Imposition time: `IT_Mani`

⌈Access to time stamps shall be configured in at most **one of two** possible ways:

- by reference in the role `idsTimeBase` from `IdsPlatformInstantiation` to `TimeBaseResource` or
- by creating a `IdsmTimestampProviderMapping` that refers in the role `pPortPrototypeInExecutable` to a `PPortPrototype` defined by a piece of application software that represents a time-stamp provider.

⌋

The first approach mentioned in [constr_10501] is documented in Chapter 12.11.2, while the second approach to obtain time-stamps mentioned in [constr_10501] is described in Chapter 12.11.3, resp. [TPS_MANI_01440].

To be clear, the difference between the two approaches for obtaining a time stamp is that the first approach is implemented entirely on the level of the interaction of functional clusters, whereas the second approach involves the execution of application-level software.

12.11.2 Time Stamps for Security Events via Time-Base Resource

In general, there are two possible ways to obtain time stamps, as described in Chapter 12.11.1. The approach described in this section is based on the creation of a reference to a `TimeBaseResource`.

Time stamps can be obtained from concrete sub-classes of the abstract base-class `TimeBaseResource`.

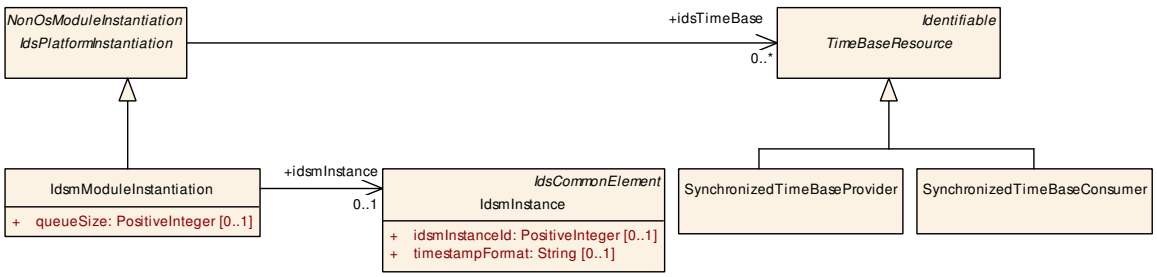


Figure 12.53: Creating an association between a `IdsmModuleInstantiation` and a `TimeBaseResource`

12.11.3 Mapping to Timestamp Provider

In general, there are two possible ways to obtain time stamps, as described in Chapter 12.11.1. The approach described in this section is based on the creation of a `IdsmTimestampProviderMapping`.

[TPS_MANI_01440] Using the `IdsmTimestampProviderMapping` for the provision of timestamps in the context of the *Intrusion Detection System Management*
⌈Meta-class `IdsmTimestampProviderMapping` creates an association between

- a `PPortPrototype` defined in the context of an `Executable` that implements application software to provide timestamps for the *Intrusion Detection System Management* and
- a `Process` in which the `Executable` is configured to run and
- the `IdsPlatformInstantiation` in which the *Intrusion Detection System Management* is implemented.

⌋

[constr_10491] Type of `PPortPrototype` referenced in the role `IdsmTimestampProviderMapping.pPortPrototypeInExecutable`

Imposition time: `IT_Mani`

⌈A `PPortPrototype` referenced in the role `IdsmTimestampProviderMapping.pPortPrototypeInExecutable` shall only be typed by a `IdsmTimestampProviderInterface`.⌋

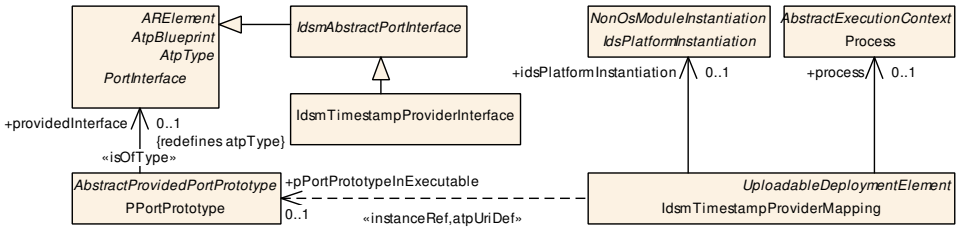


Figure 12.54: Mapping to timestamp provider

[constr_10492] Restriction for a PPortPrototype typed by a IdsmTimestamp-ProviderInterface

Imposition time: IT_Mani

⌈A PPortPrototype that is typed by a IdsmTimestampProviderInterface shall only be referenced in the role pPortPrototypeInExecutable by a Idsm-TimestampProviderMapping.⌋

Class	**IdsmTimestampProviderMapping**			
Note	This meta-class represents the ability to define a mapping between an IdsMInstance and a Process on target-configuration level to a given PortPrototype that is typed by a IdsmTimestampProviderInterface. **Tags:** atp.recommendedPackage=IdsmProviderMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
idsPlatform Instantiation	IdsPlatformInstantiation	0..1	ref	This represents the IdsM functional cluster. **Tags:** atp.Splitkey=idsPlatformInstantiation
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This reference identifies the mapped PortPrototype in the application software. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=pPortPrototypeInExecutable, pPort PrototypeInExecutable.contextRootSwComponent Prototype, pPortPrototypeInExecutable.context ComponentPrototype, pPortPrototypeInExecutable.target PPortPrototype **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference identifies the process in which the application runs. **Tags:** atp.Splitkey=process

Table 12.90: IdsmTimestampProviderMapping

[constr_10493] Existence of IdsmTimestampProviderMapping.idsPlatform-Instantiation

Imposition time: IT_Mani

⌈For each IdsmTimestampProviderMapping, the reference to an IdsPlatform-Instantiation in the role idsPlatformInstantiation shall exist.⌋

[constr_10494] Existence of instance reference in the role `IdsmTimestampProviderMapping.pPortPrototypeInExecutable`

Imposition time: IT_Mani

⌈For each IdsmTimestampProviderMapping, the instance reference to a PPortPrototype in the role pPortPrototypeInExecutable shall exist.⌋

[constr_10495] Existence of `IdsmTimestampProviderMapping.process`

Imposition time: IT_Mani

⌈For each IdsmTimestampProviderMapping, the reference to a Process in the role process shall exist.⌋

12.11.4 Mapping to Context Provider

[TPS_MANI_01441] Using the `IdsmContextProviderMapping` for the provision of context in the context of the *Intrusion Detection System Management* ⌈Metaclass IdsmContextProviderMapping creates an association between

- a PPortPrototype defined in the context of an Executable that implements application software to provide context for the *Intrusion Detection System Management* and
- a Process in which the Executable is configured to run and
- the IdsPlatformInstantiation (as the abstract base class IdsM module instantiations) in which the *Intrusion Detection System Management* is implemented.

⌋

[constr_10496] Type of `PPortPrototype` referenced in the role `IdsmContextProviderMapping.pPortPrototypeInExecutable`

Imposition time: IT_Mani

⌈A PPortPrototype referenced in the role IdsmContextProviderMapping.pPortPrototypeInExecutable shall only be typed by a IdsmContextProviderInterface.⌋

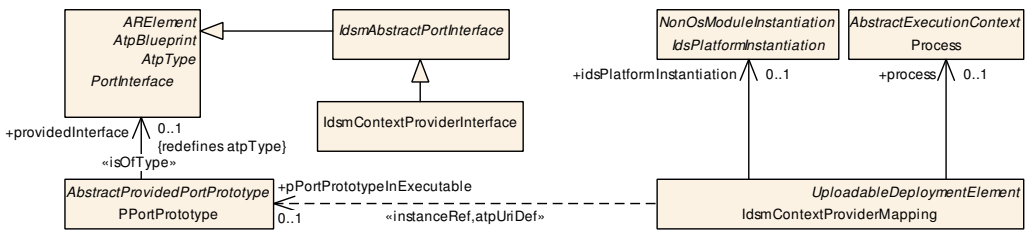


Figure 12.55: Mapping to context provider

[constr_10497] PPortPrototype typed by a IdsmContextProviderInterface

Imposition time: IT_Mani

⌈A PPortPrototype that is typed by a IdsmContextProviderInterface shall only be referenced in the role pPortPrototypeInExecutable by a IdsmContext-ProviderMapping.⌋

Class	**IdsmContextProviderMapping**			
Note	This meta-class represents the ability to define a mapping between an IdsMInstance and a Process on target-configuration level to a given PortPrototype that is typed by a IdsmContextProviderInterface. **Tags:** atp.recommendedPackage=IdsmProviderMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
idsPlatform Instantiation	IdsPlatformInstantiation	0..1	ref	This represents the IdsM functional cluster. **Tags:** atp.Splitkey=idsPlatformInstantiation
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This reference identifies the mapped PortPrototype in the application software. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=pPortPrototypeInExecutable, pPort PrototypeInExecutable.contextRootSwComponent Prototype, pPortPrototypeInExecutable.context ComponentPrototype, pPortPrototypeInExecutable.target PPortPrototype **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference identifies the process in which the application runs. **Tags:** atp.Splitkey=process

Table 12.91: IdsmContextProviderMapping

[constr_10498] Existence of IdsmContextProviderMapping.idsPlatform-Instantiation

Imposition time: IT_Mani

⌈For each IdsmContextProviderMapping, the reference to an IdsPlatform-Instantiation in the role idsPlatformInstantiation shall exist.⌋

[constr_10499] Existence of instance reference in the role IdsmContext-ProviderMapping.pPortPrototypeInExecutable

Imposition time: IT_Mani

⌈For each IdsmContextProviderMapping, the instance reference to a PPort-Prototype in the role pPortPrototypeInExecutable shall exist.⌋

[constr_10500] Existence ofIdsmContextProviderMapping.process

Imposition time: IT_Mani

⌈For each IdsmContextProviderMapping, the reference to a Process in the role process shall exist.⌋

12.11.5 Mapping to Reporting Mode Provider

[TPS_MANI_01455] Using the IdsmReportingModeProviderMapping for the setting and getting the reporting mode in the context of the *Intrusion Detection System Management* ⌈Meta-class IdsmReportingModeProviderMapping creates an association between

- an RPortPrototype defined in the context of an Executable that implements application software to set and get the reporting mode for the *Intrusion Detection System Management* and
- a Process in which the Executable is configured to run and
- the IdsPlatformInstantiation (as the abstract base class IdsM module instantiations) in which the *Intrusion Detection System Management* is implemented.

⌋

[constr_10568] Type of RPortPrototype referenced in the role IdsmReportingModeProviderMapping.rPortPrototypeInExecutable

Imposition time: IT_Mani

⌈An RPortPrototype referenced in the role IdsmReportingModeProviderMapping.rPortPrototypeInExecutable shall only be typed by a IdsmReportingModeProviderInterface.⌋

[constr_10569] RPortPrototype typed by an IdsmReportingModeProviderInterface

Imposition time: IT_Mani

⌈An RPortPrototype that is typed by a IdsmReportingModeProviderInterface shall only be referenced in the role rPortPrototypeInExecutable by a IdsmReportingModeProviderMapping.⌋

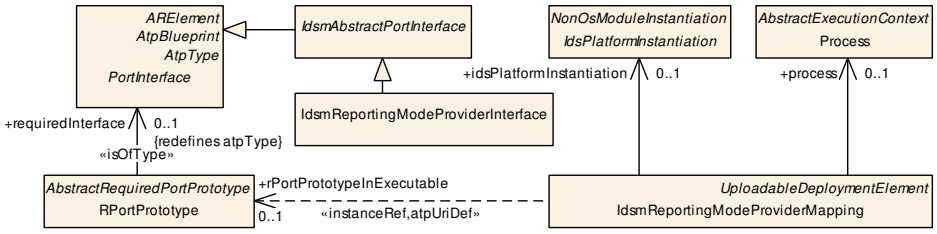


Figure 12.56: Mapping to reporting mode provider

Class	IdsmReportingModeProviderMapping			
Note	This meta-class represents the ability to define a mapping between an IdsMInstance and a Process on target-configuration level to a given PortPrototype that is typed by a IdsmReportingModeProvider Interface. **Tags:** atp.recommendedPackage=IdsmProviderMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
idsPlatform Instantiation	IdsPlatformInstantiation	0..1	ref	This represents the IdsM functional cluster. **Tags:** atp.Splitkey=idsPlatformInstantiation
process	Process	0..1	ref	This reference identifies the process in which the application runs **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This reference identifies the mapped RPortPrototype in the application software. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=rPortPrototypeInExecutable, rPort PrototypeInExecutable.contextRootSwComponent Prototype, rPortPrototypeInExecutable.context ComponentPrototype, rPortPrototypeInExecutable.target RPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 12.92: IdsmReportingModeProviderMapping

[constr_10570] Existence of `IdsmReportingModeProviderMapping.idsPlatformInstantiation`

Imposition time: `IT_Mani`

⌈For each `IdsmReportingModeProviderMapping`, the reference to a `IdsPlatformInstantiation` in the role `idsPlatformInstantiation` shall exist.⌋

[constr_10571] Existence of `IdsmReportingModeProviderMapping.rPortPrototypeInExecutable`

Imposition time: `IT_Mani`

⌈For each `IdsmReportingModeProviderMapping`, the reference to an `RPortPrototype` in the role `rPortPrototypeInExecutable` shall exist.⌋

[constr_10572] Existence of `IdsmReportingModeProviderMapping.process`

Imposition time: `IT_Mani`

⌈For each `IdsmReportingModeProviderMapping`, the reference to a `Process` in the role `process` shall exist.⌋

12.11.6 Mapping to Qualified Event Receiver

[TPS_MANI_01474] Using the `IdsmQualifiedEventReceiverMapping` for the receiving of qualified security events in the context of the *Intrusion Detection*

System Management ⌈Meta-class IdsmQualifiedEventReceiverMapping creates an association between

- an RPortPrototype, typed by an IdsmQualifiedEventReceiver-Interface, defined in the context of an Executable that implements application software to receive qualified security events and
- a Process in which the Executable is configured to run and
- the IdsPlatformInstantiation (as the abstract base class IdsM module instantiations) in which the *Intrusion Detection System Management* is implemented.

⌋

[constr_10623] Type of RPortPrototype referenced in the role IdsmQualified-EventReceiverMapping.rPortPrototypeInExecutable

Imposition time: IT_Mani

⌈An RPortPrototype referenced in the role IdsmQualifiedEventReceiver-Mapping.rPortPrototypeInExecutable shall only be typed by a Idsm-QualifiedEventReceiverInterface.⌋

[constr_10624] RPortPrototype typed by an IdsmQualifiedEventReceiver-Interface

Imposition time: IT_Mani

⌈An RPortPrototype that is typed by a IdsmReportingModeProvider-Interface shall only be referenced in the role rPortPrototypeInExecutable by a IdsmQualifiedEventReceiverMapping.⌋

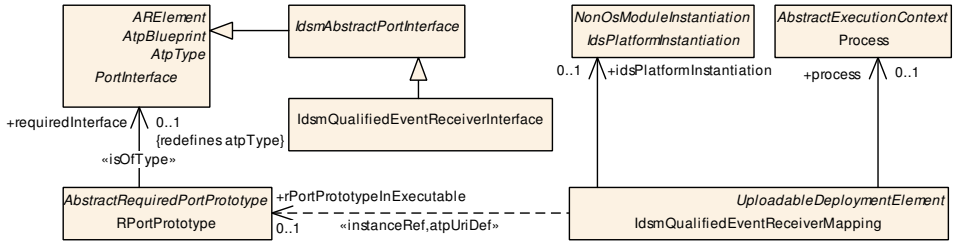


Figure 12.57: Mapping to qualified event receiver

Class	**IdsmQualifiedEventReceiverMapping**
Note	This meta-class represents the ability to define a mapping between an IdsM Module Instance and a Process on deployment level to a given PortPrototype that is typed by a IdsmQualifiedEventReceiver Interface. **Tags:** atp.recommendedPackage=IdsmReceiverMappings This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*

▽

△

Class	IdsmQualifiedEventReceiverMapping			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
idsPlatform Instantiation	IdsPlatformInstantiation	0..1	ref	This represents the IdsM functional cluster. **Tags:** atp.Splitkey=idsPlatformInstantiation
process	Process	0..1	ref	This reference identifies the process in which the application runs. **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This reference identifies the mapped RPortPrototype in the application software. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=rPortPrototypeInExecutable, rPort PrototypeInExecutable.contextRootSwComponent Prototype, rPortPrototypeInExecutable.context ComponentPrototype, rPortPrototypeInExecutable.target RPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 12.93: IdsmQualifiedEventReceiverMapping

[constr_10625] Existence of `IdsmQualifiedEventReceiverMapping.idsPlatformInstantiation`

Imposition time: `IT_Mani`

⌈For each `IdsmQualifiedEventReceiverMapping`, the reference to a `IdsPlatformInstantiation` in the role `idsPlatformInstantiation` shall exist.⌋

[constr_10626] Existence of `IdsmQualifiedEventReceiverMapping.rPortPrototypeInExecutable`

Imposition time: `IT_Mani`

⌈For each `IdsmQualifiedEventReceiverMapping`, the reference to an `RPortPrototype` in the role `rPortPrototypeInExecutable` shall exist.⌋

[constr_10627] Existence of `IdsmQualifiedEventReceiverMapping.process`

Imposition time: `IT_Mani`

⌈For each `IdsmQualifiedEventReceiverMapping`, the reference to a `Process` in the role `process` shall exist.⌋

12.11.7 Target-Configuration for Security Events

[TPS_MANI_01341] Security events that are actually reported by a local IdsM ⌈The security events that are actually reported by a local IdsM are represented by meta-class `SecurityEventMapping`, referenced by `IdsmModuleInstantiation` in the role `reportableSecurityEvent`.⌋

[TPS_MANI_01342] Semantics of `SecurityEventMapping` ⌈The semantics of meta-class `SecurityEventMapping` is to identify

- the PortPrototype in the context of an Executable from which the security event is reported,
- the Process that runs the Executable, and
- the SecurityEventDefinition that is subject to be reported.

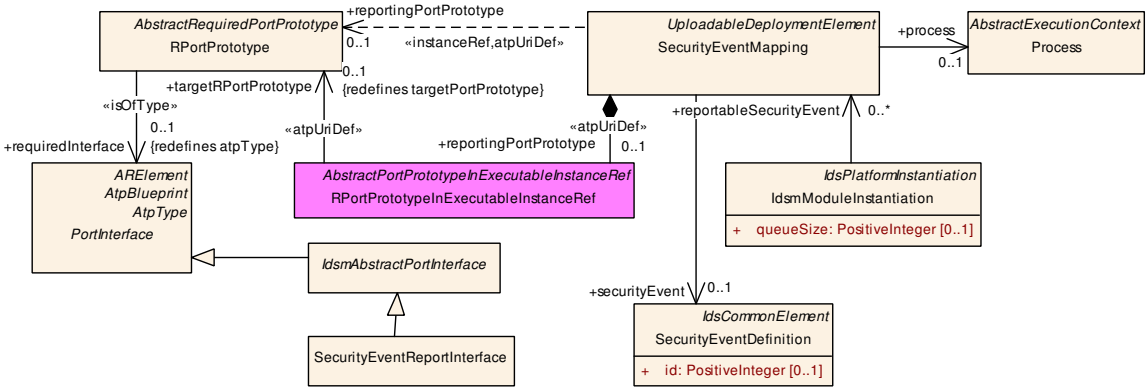


Figure 12.58: Modeling of between a SecurityEventMapping

Class	**SecurityEventMapping**			
Note	This meta-class represents a reportable instance of a security event. **Tags:** atp.recommendedPackage=SecurityEventMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
process	Process	0..1	ref	This reference identifies the process in which context the security event is reported. **Tags:** atp.Splitkey=process
reportingPort Prototype	RPortPrototype	0..1	iref	This instanceRef identifies the PortPrototype over which the security event is reported. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=reportingPortPrototype, reportingPort Prototype.contextRootSwComponentPrototype, reporting PortPrototype.contextComponentPrototype, reportingPort Prototype.targetRPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef
securityEvent	SecurityEventDefinition	0..1	ref	This reference identifies the corresponding SecurityEvent Definition. **Tags:** atp.Splitkey=securityEvent

Table 12.94: SecurityEventMapping

12.12 Service-oriented Vehicle Diagnostics

The definition of the target-configuration for the SOVD is modeled by means of the meta-class SovdModuleInstantiation.

[TPS_MANI_03346] Number of instances of SovdModuleInstantiation on a Machine ⌈On each Machine, several instances of the SOVD (modeled by SovdModuleInstantiation) may exist.⌋

As already described in Chapter 6.3.2, the Service-oriented Vehicle Diagnostics (SOVD) architecture foresees the existence of SOVD gateways that communicate with SOVD Clients outside the vehicle's network and with SOVD servers inside the vehicle's network. This is reflected in the target configuration in the formalization as SovdServerInstantiation and SovdGatewayInstantiation.

Each SovdModuleInstantiation element contains a reference to the corresponding MachineDesign element that defines conditions that the SovdModuleInstantiation needs to implement.

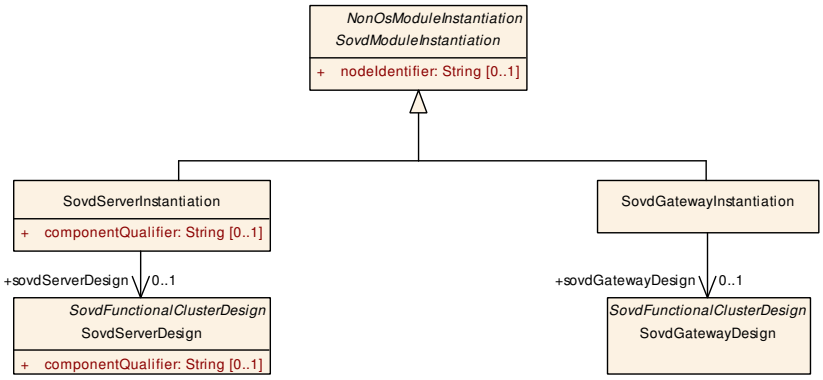


Figure 12.59: Modeling of subclasses of SovdModuleInstantiation

Class	***SovdModuleInstantiation*** (abstract)			
Note	This abstract meta-class serves as the base class for meta-classes that describe the configuration of an SOVD module. **Tags:** atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Subclasses	SovdGatewayInstantiation, SovdServerInstantiation			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
nodeIdentifier	String	0..1	attr	This attribute represents the local hostname of the SOVD server to be used during the execution of the DNS-SD protocol. **Tags:** atp.Status=candidate

Table 12.95: SovdModuleInstantiation

12.12.1 SOVD Server

[TPS_MANI_01531] Semantics of `SovdServerInstantiation` ⌈Meta-class `SovdServerInstantiation` represents that configuration of an SOVD server on a given target.⌋

Class	**SovdServerInstantiation**			
Note	This meta-class represents the configuration of an SOVD server. **Tags:** atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*, *SovdModuleInstantiation*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
component Qualifier	String	0..1	attr	This attributes described the component qualifier used to compose an SOVD query. **Tags:** atp.Status=candidate
sovdServer Design	SovdServerDesign	0..1	ref	Reference to the SovdServerDesign that this SovdServer Instantiation implements. **Tags:** atp.Splitkey=sovdServerDesign

Table 12.96: SovdServerInstantiation

[constr_9431] Existence of `SovdServerInstantiation.sovdServerDesign`

Imposition time: `IT_Mani`

⌈For each `SovdServerInstantiation`, the reference `sovdServerDesign` shall exist.⌋

12.12.2 SOVD Gateway

[constr_9432] Existence of `SovdGatewayInstantiation.sovdGatewayDesign`

Imposition time: `IT_Mani`

⌈For each `SovdGatewayInstantiation`, the reference `sovdGatewayDesign` shall exist.⌋

Class	**SovdGatewayInstantiation**			
Note	This meta-class represents the configuration of an SOVD gateway. **Tags:** atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*, *SovdModuleInstantiation*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**SovdGatewayInstantiation**			
sovdGateway Design	SovdGatewayDesign	0..1	ref	Reference to the SovdGatewayDesign that this Sovd GatewayInstantiation implements. **Tags:** atp.Splitkey=sovdGatewayDesign

Table 12.97: SovdGatewayInstantiation

12.13 Firewall Target-Configuration

AUTOSAR supports an Ethernet packet filtering firewall as an additional security layer. Such a firewall checks, for example, the source and destination IP address, protocols (UDP and TCP) and port addresses. If the address information matches then the packet is considered secured.

Two kinds of firewalls are supported:

1. Stateful packet filtering firewalls
2. Stateless packet filtering firewalls

The stateful firewall monitors the state of network connections and analyzes the complete context of traffic and data packets. The context involves the metadata of the packets (e.g. the ports and IP address of the endpoint and destination) and the state refers to the policy based on the connection state. Approved traffic requests will be added to an approved list and will be allowed to proceed to the network.

The stateless firewall supports a rule-based filtering of traffic based on the evaluation of IP packet parameters without knowledge of the connection status (e.g. TCP session). The filtering can either be allow list-based or block list-based.

Class	**AdaptiveFirewallModuleInstantiation**			
Note	This meta-class defines the attributes for the Firewall configuration on a specific machine. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
stateDep Firewall	StateDependentFirewall	*	ref	Firewall rules that are defined in a firewall state. **Tags:** atp.Splitkey=stateDepFirewall

Table 12.98: AdaptiveFirewallModuleInstantiation

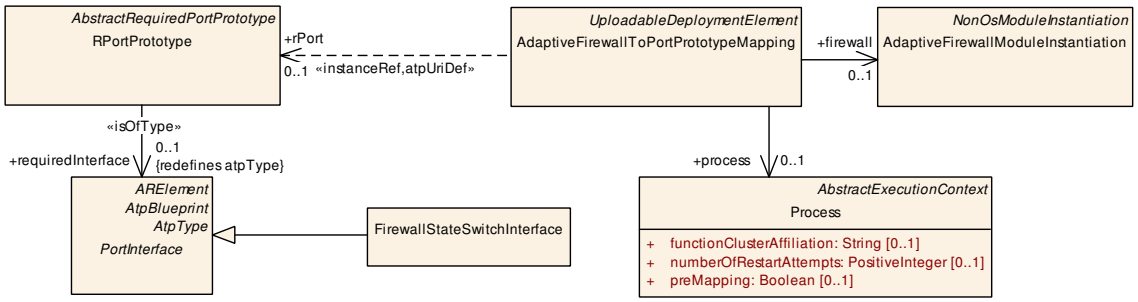


Figure 12.60: `AdaptiveFirewallModuleInstantiation` and the relationship to Application with a `RPortPrototype` typed by a `FirewallStateSwitchInterface`

[TPS_MANI_03296] Machine-specific configuration settings for Firewall ⌈The Machine-specific configuration settings for the Firewall module are collected in `AdaptiveFirewallModuleInstantiation`.⌋

Class	**AdaptiveFirewallToPortPrototypeMapping**			
Note	This meta-class maps the AdaptiveFirewall moduleInstantiation to the RPortPrototype that is typed by a FirewallModeSwitchInterface. **Tags:** atp.recommendedPackage=AdaptiveFirewallToPortPrototypeMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
firewall	AdaptiveFirewallModule Instantiation	0..1	ref	Reference to the Firewall module **Tags:** atp.Splitkey=firewall
process	Process	0..1	ref	The referenced Process is supposed to define the Port Prototype typed by a FirewallStateSwitchInterface, referenced by the enclosing mapping class. If this reference does not exist, any Process using the respective API can switch the firewall state. **Tags:** atp.Splitkey=process
rPort	RPortPrototype	0..1	iref	Reference to RPortPrototype typed by a FirewallMode SwitchInterface **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=rPort, rPort.contextRootSw ComponentPrototype, rPort.contextComponentPrototype, rPort.targetRPortPrototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 12.99: AdaptiveFirewallToPortPrototypeMapping

[TPS_MANI_03297] Semantics of `AdaptiveFirewallToPortPrototypeMapping` ⌈Meta-class `AdaptiveFirewallToPortPrototypeMapping` has the ability to reference a specific `RPortPrototype` with the `rPort` reference that is typed by the `FirewallStateSwitchInterface` and to map it to the configured `AdaptiveFirewallModuleInstantiation` that is referenced with the `firewall` reference.⌋

[constr_5358] `AdaptiveFirewallToPortPrototypeMapping.rPort` restriction

Imposition time: `IT_Mani`

⌈The `AdaptiveFirewallToPortPrototypeMapping` is only allowed to reference a `RPortPrototype` in the role `rPort` that is typed by the `FirewallStateSwitchInterface`.⌋

Firewall functionality can be implemented on Host ECUs and on switches (as network firewalls). This is the reason why the `CouplingElement` (that defines a Switch) and the `Machine` (via the `AdaptiveFirewallModuleInstantiation`) both have an optional reference to the `StateDependentFirewall` that defines the entry point for the Firewall configuration.

The in-vehicle traffic may depend on the current state of the firewall state-machine. The firewall rules are adaptable to this and therefore the `StateDependentFirewall` is associated with a `state`.

[TPS_MANI_03298] Semantics of `StateDependentFirewall` ⌈If the `state` that is referenced by the `StateDependentFirewall` is active, then the firewall rule that is defined by the `StateDependentFirewall` shall be considered as active as well.⌋

[constr_5348] Mandatory `initialMode` in `ModeDeclarationGroup` that is referenced by `StateDependentFirewall`

Imposition time: `IT_Mani`

⌈The `ModeDeclarationGroup` that is referenced via a `ModeDeclaration` from `StateDependentFirewall` in the role `state` shall define an `initialMode`.⌋

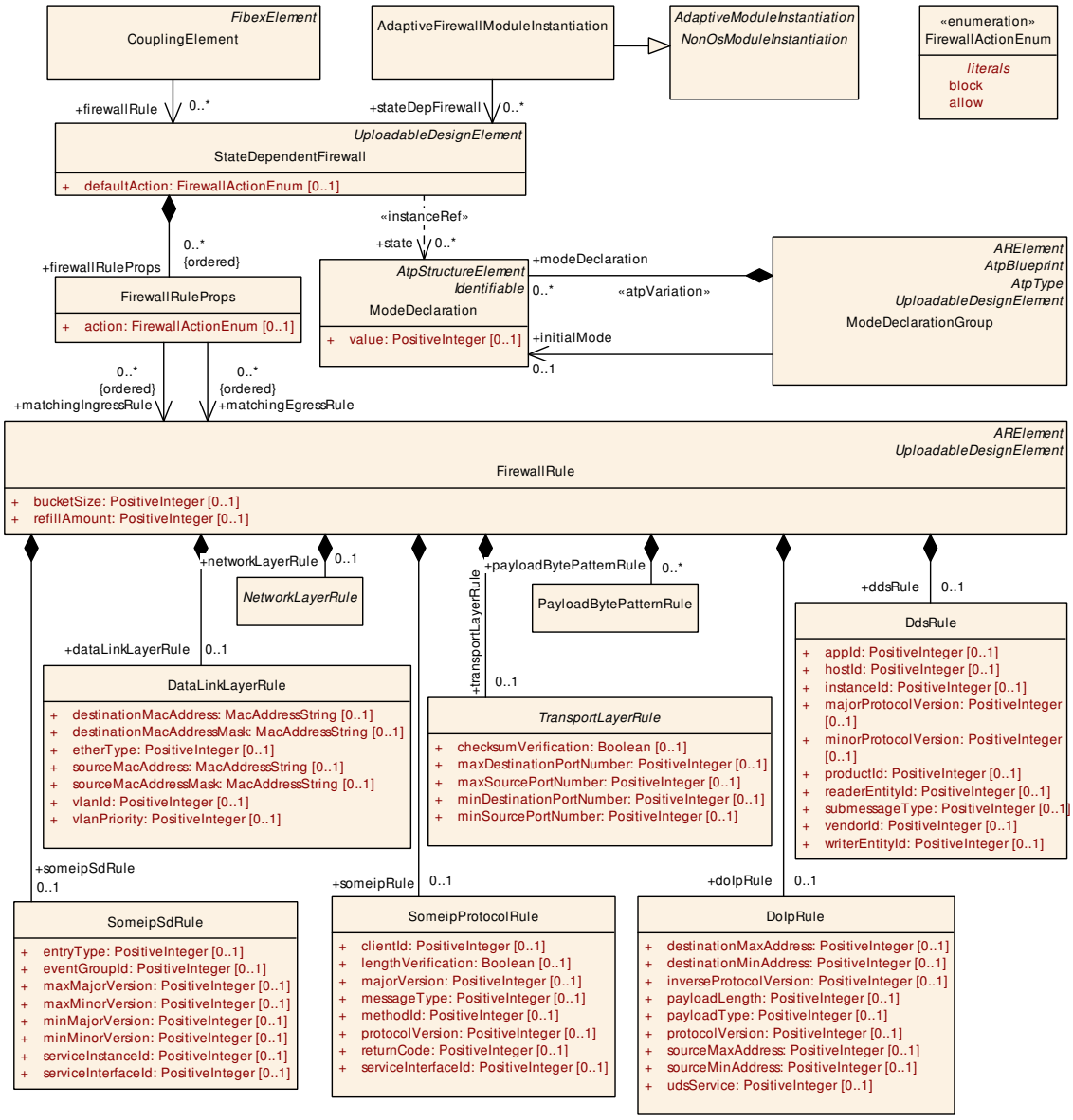


Figure 12.61: Modeling of the Firewall

This `initialMode` is used in the case that the system boots up and no FirewallMode is yet set by an application.

[constr_5349] Mandatory `defaultAction` in `StateDependentFirewall`

Imposition time: `IT_Mani`

⌈The `StateDependentFirewall` shall always define the attribute `defaultAction`.⌋

[TPS_MANI_03299] Semantics of meta-class `FirewallRuleProps` ⌈Each `StateDependentFirewall` defines `FirewallRuleProps` that in turn define a rule expression (`matchingIngressRule` and `matchingEgressRule`) and an `action` that is performed if the rule expression matches.⌋

[constr_5350] Mandatory `action` in `FirewallRuleProps`

Imposition time: `IT_Mani`

⌈The `FirewallRuleProps` shall always define the attribute `action`.⌋

[TPS_MANI_03300] Semantics of `FirewallRule` ⌈The `FirewallRule` defines an expression which matches network traffic against the conditions that are defined by the expression.⌋

[TPS_MANI_03301] Semantics of `FirewallRule.refillAmount` and `FirewallRule.bucketSize` ⌈Every firewall matching rule can be associated with a rate limit for limiting network traffic. In AUTOSAR, the rate limit is implemented by a leaky bucket algorithm. The attributes `FirewallRule.refillAmount` and `FirewallRule.bucketSize` are used to configure the rate limit algorithm.⌋

[constr_5351] `FirewallRule` is allowed to aggregate at most one protocol subelement

Imposition time: `IT_Mani`

⌈A `FirewallRule` is allowed to aggregate either:

- `someipSdRule`
- `someipRule`
- `doIpRule`

⌋

If a firewall rule needs to cover different Protocols (e.g. SOME/IP, DoIP) then several `FirewallRule`s need to be defined because of [constr_5351]. A `FirewallRule` is allowed to aggregate a `DataLinkLayerRule`, a `NetworkLayerRule` and a `TransportLayerRule` at the same time in addition to one of the sub-elements mentioned in [constr_5351].

Class	**StateDependentFirewall**			
Note	Firewall rules that are defined in a firewall state **Tags:** atp.recommendedPackage=StateDependentFirewallRules			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defaultAction	FirewallActionEnum	0..1	attr	This attribute defines a defaultAction in case that the VehicleMode is not yet set.
firewallRule Props (ordered)	FirewallRuleProps	*	aggr	Collection of firewall rules that apply in the vehicle mode

▽

△

Class	**StateDependentFirewall**			
state	ModeDeclaration	*	iref	Reference to firewall states in which the Firewall is active. If one of the referenced ModeDeclarations is the current firewall state then the firewall rule shall be considered as active. **Tags:** atp.Splitkey=state, state.contextModeDeclaration GroupPrototype, state.targetModeDeclaration **InstanceRef implemented by:** FirewallStateInFirewall StateSwitchInterfaceInstanceRef This Attribute is only used by the AUTOSAR Adaptive Platform.

Table 12.100: StateDependentFirewall

Class	**FirewallRule**			
Note	Firewall Rule that defines the control information in individual packets. **Tags:** atp.recommendedPackage=FirewallRules			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bucketSize	PositiveInteger	0..1	attr	This attribute defines the capacity of the queue for rate limitation (leaky-bucket Algorithm).
dataLinkLayer Rule	DataLinkLayerRule	0..1	aggr	Configuration of rules on the Data Link Layer **Tags:** atp.Splitkey=dataLinkLayerRule
ddsRule	DdsRule	0..1	aggr	Configuration of firewall rules for DDS. **Tags:** atp.Splitkey=ddsRule
doIpRule	DoIpRule	0..1	aggr	Configuration of firewall rules for DoIP messages **Tags:** atp.Splitkey=doIpRule
networkLayer Rule	NetworkLayerRule	0..1	aggr	Configuration of rules on the Network Layer **Tags:** atp.Splitkey=networkLayerRule
payloadByte PatternRule	PayloadBytePattern Rule	*	aggr	Configuration of generic firewall rules **Tags:** atp.Splitkey=payloadBytePatternRule
refillAmount	PositiveInteger	0..1	attr	This attribute defines the output rate that describes how many packets leave the queue per second (leaky-bucket Algorithm).
someipRule	SomeipProtocolRule	0..1	aggr	Configuration of firewall rules for SOME/IP messages **Tags:** atp.Splitkey=someipRule
someipSdRule	SomeipSdRule	0..1	aggr	Configuration of firewall rules for SOME/IP Service Discovery messages **Tags:** atp.Splitkey=someipSdRule
transportLayer Rule	TransportLayerRule	0..1	aggr	Configuration of rules on the Transport Layer **Tags:** atp.Splitkey=transportLayerRule

Table 12.101: FirewallRule

Class	**FirewallRuleProps**			
Note	Firewall rule that is defined by an action that is performed if the referenced pattern matches.			
Base	*ARObject*			
Aggregated by	StateDependentFirewall.firewallRuleProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
action	FirewallActionEnum	0..1	attr	Action that is performed by the firewall if the matching Rule is fulfilled.
matchingEgress Rule (ordered)	FirewallRule	*	ref	This element defines an egress rule expression against which the network traffic is matched.
matching IngressRule (ordered)	FirewallRule	*	ref	This element defines an ingress rule expression against which the network traffic is matched.

Table 12.102: FirewallRuleProps

Enumeration	**FirewallActionEnum**
Note	List of actions that the Firewall is able to perform.
Aggregated by	FirewallRuleProps.action, StateDependentFirewall.defaultAction
Literal	***Description***
allow	Firewall allows the communication **Tags:** atp.EnumerationLiteralIndex=1
block	Firewall blocks the communication **Tags:** atp.EnumerationLiteralIndex=0

Table 12.103: FirewallActionEnum

12.13.1 Firewall Rules on Datalink Layer

[TPS_MANI_03302] Firewall rules on Data Link Layer ⌈The meta-class `DataLinkLayerRule` is used to define firewall rules on the Data Link Layer (second layer of the seven-layer OSI model).⌋

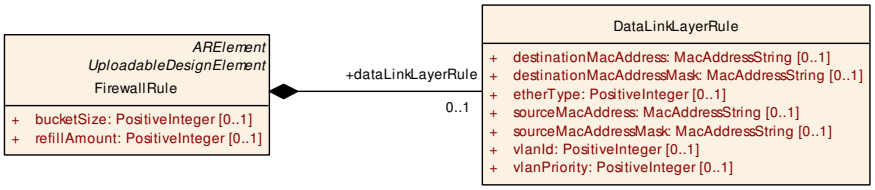


Figure 12.62: Modeling of Firewall rules on Data Link Layer

[TPS_MANI_03303] Filtering of packets with a single MAC address and MAC address range ⌈The `sourceMacAddressMask` is specifying which bits in the `sourceMacAddress` are static to define a source MAC Address or source MAC Address Range. For each bit position in the MAC mask a 1 indicates that the corresponding address bit is significant and a 0 indicates that the address bit is ignored. `destinationMacAddressMask` and `destinationMacAddress` shall be used in the same way to define a destination MAC Address or destination MAC Address Range.⌋

If, for example, only the first four octets of a MAC address are relevant, a sourceMacAddressMask of ff:ff:ff:ff:00:00 is used. A sourceMacAddressMask of ff:ff:ff:ff:ff:ff defines that all address bits are relevant and is used to define a single MAC address.

Class	**DataLinkLayerRule**			
Note	Configuration of filter rules on the DataLink layer			
Base	*ARObject*			
Aggregated by	FirewallRule.dataLinkLayerRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destinationMac Address	MacAddressString	0..1	attr	Filter to match packets with the destination MAC address.
destinationMac AddressMask	MacAddressString	0..1	attr	Filter to match packets with the destination MAC address range. The destinationMacAddress with the destination MacAddressMask defines the MAC address range.
etherType	PositiveInteger	0..1	attr	Filter to match packets based on the EtherType field in the Ethernet frame. The EtherType is used to indicate which protocol is encapsulated in the payload of the frame.
sourceMac Address	MacAddressString	0..1	attr	Filter to match packets with the source MAC address.
sourceMac AddressMask	MacAddressString	0..1	attr	Filter to match packets with the source MAC address range. The sourceMacAddress with the sourceMac AddressMask defines the MAC address range.
vlanId	PositiveInteger	0..1	attr	Filter of packets with a specific VlanId.
vlanPriority	PositiveInteger	0..1	attr	Filter of packets with a specific Vlan priority.

Table 12.104: DataLinkLayerRule

12.13.2 Firewall Rules on Network Layer

[TPS_MANI_03304] Firewall rules on Network Layer ⌈The meta-class NetworkLayerRule is used to define firewall rules on the Network Layer (third layer of the seven-layer OSI model).⌋

[TPS_MANI_03305] Filtering of packets with a single IP address and IP address range ⌈The Ipv4Rule.sourceNetworkMask (or Ipv6Rule.sourceNetworkMask) is specifying which bits in the Ipv4Rule.sourceIpAddress (or Ipv6Rule.sourceIpAddress) are static to define a source IP Address or source IP Address Range. destinationIpAddress and destinationNetworkMask shall be used in the same way to define a destination IP Address or destination IP Address Range.⌋

If, for example, only the first three octets of a IP address are relevant, a sourceNetworkMask of 255.255.255.0 is used. The octet value “255” in the mask defines that the corresponding numbers in the IP address are static and never change.

In this example, the first three octets are fixed and the fourth octet is 0. This means that on the fourth octet, all individual numbers from 1-254 are valid.

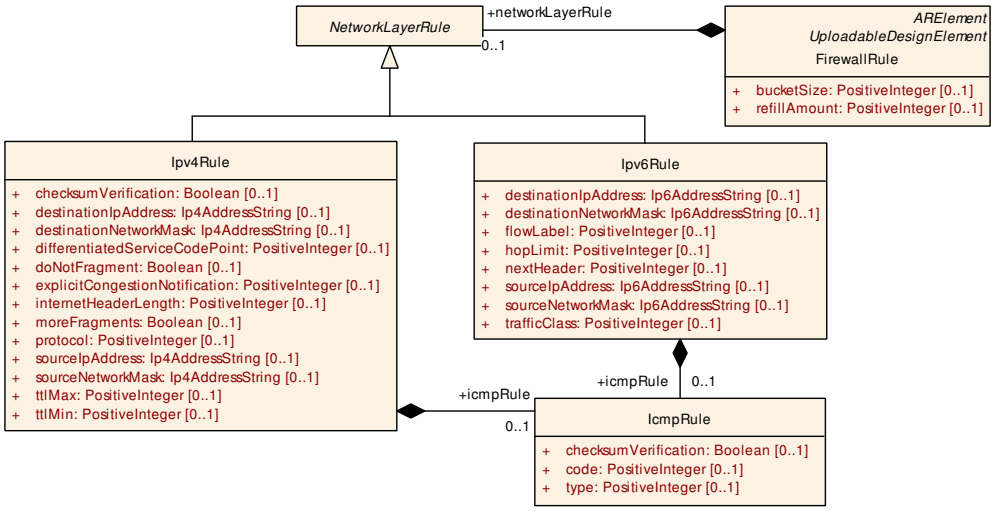


Figure 12.63: Modeling of Firewall rules on Network Layer

Class	***NetworkLayerRule*** (abstract)			
Note	Configuration of filter rules on the Network layer			
Base	*ARObject*			
Subclasses	Ipv4Rule, Ipv6Rule			
Aggregated by	FirewallRule.networkLayerRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 12.105: NetworkLayerRule

Class	**Ipv4Rule**			
Note	Configuration of filter rules on IPv4 level.			
Base	*ARObject*, *NetworkLayerRule*			
Aggregated by	FirewallRule.networkLayerRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checksum Verification	Boolean	0..1	attr	Defines whether a Ipv4 header checksum verification is performed or not.
destinationIp Address	Ip4AddressString	0..1	attr	Filter to match packets with the destination IPv4 address.
destination NetworkMask	Ip4AddressString	0..1	attr	Filter to match packets with the destination IPv4 address range. The destinationIpAddress with the destination NetworkMask defines the IP address range.
differentiated ServiceCode Point	PositiveInteger	0..1	attr	Filter to match packets with a DSCP value.
doNotFragment	Boolean	0..1	attr	Filter to match packets that have the doNotFragment bit in the Header set.
explicit Congestion Notification	PositiveInteger	0..1	attr	Filter to match packets with a ECN code point.
icmpRule	IcmpRule	0..1	aggr	Configuration of filter rules for ICMP (Internet Control Message Protocol). **Tags:** atp.Splitkey=icmpRule

△

Class	**Ipv4Rule**			
internetHeader Length	PositiveInteger	0..1	attr	Filter to match packets with a minimum ipv4 header length.
moreFragments	Boolean	0..1	attr	Filter to match packets that have the moreFragments flag in the Header set.
protocol	PositiveInteger	0..1	attr	Filter to match packets with a IP protocol number .
sourceIp Address	Ip4AddressString	0..1	attr	Filter to match packets with the source IPv4 address.
sourceNetwork Mask	Ip4AddressString	0..1	attr	Filter to match packets with the source IPv4 address range. The sourceIpAddress with the sourceNetwork Mask defines the IP address range.
ttlMax	PositiveInteger	0..1	attr	Filter to match packets with a maximum ttl value (TimeTo Live defines the lifetime of data on the network).
ttlMin	PositiveInteger	0..1	attr	Filter to match packets with a minimum ttl value (TimeTo Live defines the lifetime of data on the network).

Table 12.106: Ipv4Rule

Class	**Ipv6Rule**			
Note	Configuration of filter rules on IPv6 level.			
Base	*ARObject*, *NetworkLayerRule*			
Aggregated by	FirewallRule.networkLayerRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destinationIp Address	Ip6AddressString	0..1	attr	Filter to match packets with the destination IPv6 address.
destination NetworkMask	Ip6AddressString	0..1	attr	Filter to match packets with the destination IPv6 address range. The destinationIpAddress with the destination NetworkMask defines the MAC address range.
flowLabel	PositiveInteger	0..1	attr	Filter to match packets with a defined flow label.
hopLimit	PositiveInteger	0..1	attr	Filter to match packets with a minimum hop limit.
icmpRule	IcmpRule	0..1	aggr	Configuration of filter rules for ICMP (Internet Control Message Protocol). **Tags:** atp.Splitkey=icmpRule
nextHeader	PositiveInteger	0..1	attr	Filter to match packets with a defined type of an extension header.
sourceIp Address	Ip6AddressString	0..1	attr	Filter to match packets with the source IPv6 address.
sourceNetwork Mask	Ip6AddressString	0..1	attr	Filter to match packets with the source IPv6 address range. The sourceIpAddress with the sourceNetwork Mask defines the IP address range.
trafficClass	PositiveInteger	0..1	attr	Filter to match packets with a defined traffic class or priority.

Table 12.107: Ipv6Rule

Class	**IcmpRule**			
Note	Configuration of filter rules for ICMP (Internet Control Message Protocol).			
Base	*ARObject*			
Aggregated by	Ipv4Rule.icmpRule, Ipv6Rule.icmpRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	IcmpRule			
checksum Verification	Boolean	0..1	attr	Defines whether a Icmp header checksum verification is performed or not.
code	PositiveInteger	0..1	attr	Filter to match packets with the Icmp code.
type	PositiveInteger	0..1	attr	Filter to match packets with the Icmp type.

Table 12.108: IcmpRule

12.13.3 Firewall Rules on Transport Layer

[TPS_MANI_03306] Firewall rules on the Transport Layer ⌈The meta-class `TransportLayerRule` is used to define firewall rules on the Transport Layer (fourth layer of the seven-layer OSI model).⌋

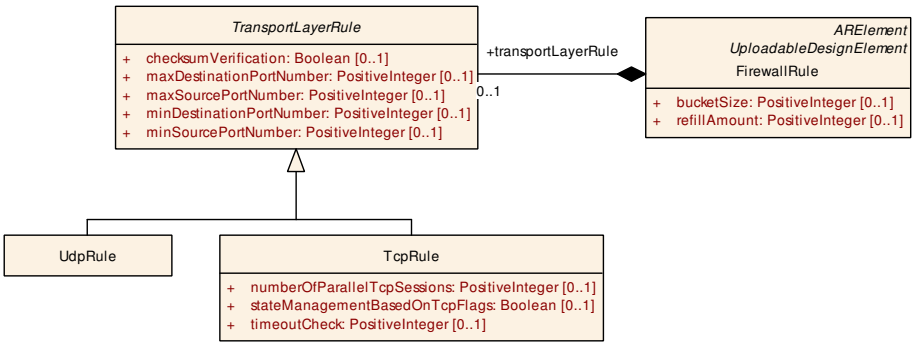


Figure 12.64: Modeling of Firewall rules on Transport Layer

Class	***TransportLayerRule*** (abstract)			
Note	Configuration of filter rules on Transport Layer level.			
Base	*ARObject*			
Subclasses	TcpRule, UdpRule			
Aggregated by	FirewallRule.transportLayerRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checksum Verification	Boolean	0..1	attr	Defines whether checksum verification is performed or not.
maxDestination PortNumber	PositiveInteger	0..1	attr	Filter to match packets with the maximum destination UDP/TCP port number.
maxSourcePort Number	PositiveInteger	0..1	attr	Filter to match packets with the maximum source UDP/ TCP port number.
minDestination PortNumber	PositiveInteger	0..1	attr	Filter to match packets with the minimum destination UDP/TCP port number.
minSourcePort Number	PositiveInteger	0..1	attr	Filter to match packets with the minimum source UDP/ TCP port number.

Table 12.109: TransportLayerRule

Class	UdpRule			
Note	Configuration of UDP filter rules.			
Base	*ARObject*, *TransportLayerRule*			
Aggregated by	FirewallRule.transportLayerRule			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 12.110: UdpRule

Class	TcpRule			
Note	Configuration of TCP filter rules.			
Base	*ARObject*, *TransportLayerRule*			
Aggregated by	FirewallRule.transportLayerRule			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
numberOf ParallelTcp Sessions	PositiveInteger	0..1	attr	This attribute defines the maximal number of TCP Sessions that are allowed to be established.
state Management BasedOnTcp Flags	Boolean	0..1	attr	This attribute defines whether the StateManagement is based on TCP flags or not.
timeoutCheck	PositiveInteger	0..1	attr	This attribute defines the TCP Session timeout in seconds

Table 12.111: TcpRule

12.13.4 Firewall Rules via Byte Pattern

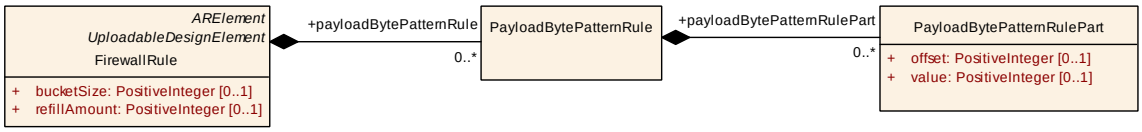


Figure 12.65: Modeling of Firewall rules via byte pattern

[TPS_MANI_03307] Byte-Pattern Firewall rule ⌈The meta-class `PayloadBytePatternRule` is used to define a generic firewall rule that specifies individually the value for each byte of the Ethernet frame. Each `PayloadBytePatternRulePart` that is aggregated by `PayloadBytePatternRule` defines one byte in the Ethernet frame. The `offset` defines the position in the Ethernet Frame and the `value` defines the byteValue.⌋

Class	PayloadBytePatternRule			
Note	Configuration of a generic firewall rule that defines the individual bytes of a message that shall match.			
Base	*ARObject*			
Aggregated by	FirewallRule.payloadBytePatternRule			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
payloadByte PatternRulePart	PayloadBytePattern RulePart	*	aggr	Configuration of bytes in the message, **Tags:** atp.Splitkey=payloadBytePatternRulePart

Table 12.112: PayloadBytePatternRule

Class	**PayloadBytePatternRulePart**			
Note	Configuration of one byte in the datagram,			
Base	*ARObject*			
Aggregated by	PayloadBytePatternRule.payloadBytePatternRulePart			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
offset	PositiveInteger	0..1	attr	This attribute defines the byte offset in the datagram (start byte of the Ethernet frame, i.e. offset 0 corresponds to the first byte of the destination MAC address).
value	PositiveInteger	0..1	attr	This attribute defines the byteValue (0..255) in the datagram.

Table 12.113: PayloadBytePatternRulePart

12.13.5 Firewall Deep Packet Inspection of SOME/IP SD and SOME/IP Messages

[TPS_MANI_03308] Deep inspection of SOME/IP SD messages ⌈The meta-class `SomeipSdRule` is used to define a firewall rule for SOME/IP SD messages.⌋

[TPS_MANI_03309] Deep inspection of SOME/IP messages ⌈The meta-class `SomeipProtocolRule` is used to define a firewall rule for SOME/IP messages.⌋

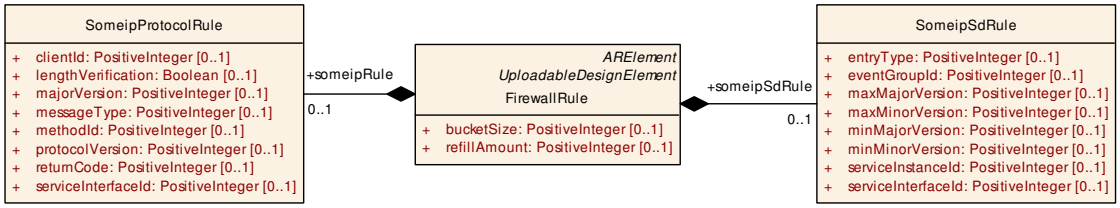


Figure 12.66: Modeling of SOME/IP Firewall rules

Class	**SomeipProtocolRule**			
Note	Configuration of SOME/IP firewall rules			
Base	*ARObject*			
Aggregated by	FirewallRule.someipRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientId	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the clientId in the SOME/IP header matches.
length Verification	Boolean	0..1	attr	Defines whether length verification is performed or not.
majorVersion	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the majorVersion in the SOME/IP header matches.
messageType	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the messageType in the SOME/IP header matches.
methodId	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the methodId in the SOME/IP header matches.
protocolVersion	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the protocol Version in the SOME/IP header matches.

▽

△

Class	**SomeipProtocolRule**			
returnCode	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the returnCode in the SOME/IP header matches.
serviceInterface Id	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the service InterfaceId in the SOME/IP header matches.

Table 12.114: SomeipProtocolRule

Class	**SomeipSdRule**			
Note	Configuration of SOME/IP Service Discovery firewall rules			
Base	*ARObject*			
Aggregated by	FirewallRule.someipSdRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
entryType	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the entryType in the SOME/IP header matches.
eventGroupId	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the eventGroup Id in the SOME/IP header matches.
maxMajor Version	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the Major Version in the SOME/IP header is smaller or equal than maxMajorVersion.
maxMinor Version	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the Minor Version in the SOME/IP header is smaller or equal than maxMinorVersion.
minMajor Version	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the Major Version in the SOME/IP header is greater or equal than minMajorVersion.
minMinor Version	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the Minor Version in the SOME/IP header is greater or equal than minMinorVersion.
serviceInstance Id	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the service InstanceId in the SOME/IP header matches.
serviceInterface Id	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the service InterfaceId in the SOME/IP header matches.

Table 12.115: SomeipSdRule

12.13.6 Firewall Deep Packet Inspection of DoIP Messages

[TPS_MANI_03310] Deep inspection of DoIP messages ⌈The meta-class `DoIpRule` is used to define a firewall rule for DoIP messages.⌋

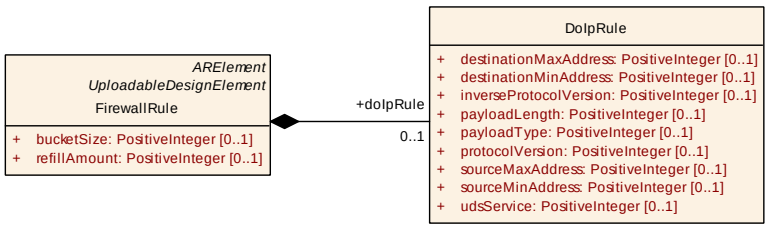


Figure 12.67: Modeling of DoIP Firewall rules

Class	**DoIpRule**			
Note	Configuration of a generic firewall rule			
Base	*ARObject*			
Aggregated by	FirewallRule.doIpRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destinationMax Address	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the destination Address is smaller or equal than destinationMaxAddress.
destinationMin Address	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the destination Address is greater or equal than destinationMinAddress.
inverseProtocol Version	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the inverseprotocolVersion in the DoIP header matches.
payloadLength	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the payload Length in the DoIP header matches.
payloadType	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the payloadType in the DoIP header matches.
protocolVersion	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the protocol Version in the DoIP header matches.
sourceMax Address	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the source Address is smaller or equal than sourceMaxAddress.
sourceMin Address	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the source Address is greater or equal than sourceMinAddress..
udsService	PositiveInteger	0..1	attr	Filter to match DoIP messages that contain the uds Service.

Table 12.116: DoIpRule

12.13.7 Firewall Deep Packet Inspection of DDS Messages

[TPS_MANI_03311] Deep inspection of DDS messages

Status: DRAFT

⌈The meta-class `DdsRule` is used to define a firewall rule for DDS messages.⌋

[constr_5352] `DdsRule.submessageType` value restriction

Status: DRAFT
Imposition time: `IT_Mani`

⌈The value of `DdsRule.submessageType` is restricted to the following values:

- 0x01 (PAD)
- 0x06 (ACKNACK)
- 0x07 (HEARTBEAT)
- 0x08 (GAP)
- 0x09 (INFO_TS)
- 0x0c (INFO_SRC)
- 0x0d (INFO_REPLY_IP4)
- 0x0e (INFO_DST)

- 0x0f (INFO_REPLY)
- 0x12 (NACK_FRAG)
- 0x13 (HEARTBEAT_FRAG)
- 0x15 (DATA)
- 0x16 (DATA_FRAG)

⌋

[constr_5353] `DdsRule.readerEntityId` and `DdsRule.writerEntityId` value restriction

Status: DRAFT
Imposition time: `IT_Mani`

⌈The value of `DdsRule.readerEntityId` and `DdsRule.writerEntityId` is only allowed to be set if the value of `DdsRule.submessageType` is set to one of the following values:

- 0x06 (ACKNACK)
- 0x07 (HEARTBEAT)
- 0x08 (GAP)
- 0x15 (DATA)

⌋

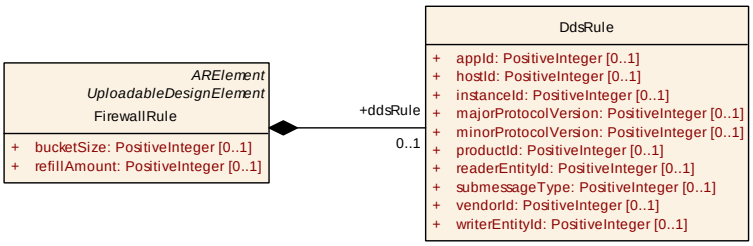


Figure 12.68: Modeling of DDS Firewall rules

Class	**DdsRule**			
Note	Configuration of a DDS firewall rule			
Base	*ARObject*			
Aggregated by	FirewallRule.ddsRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
appId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the appId in the DDSI-RTPS header and the INFO_DST (0x0E) submessage matches.
hostId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the hostId in the DDSI-RTPS header and the INFO_DST (0x0E) submessage matches.

▽

△

Class	DdsRule			
instanceId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the instanceId in the DDSI-RTPS header and the INFO_DST (0x0E) submessage matches.
majorProtocol Version	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the major ProtocolVersion in the DDSI-RTPS header matches.
minorProtocol Version	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the minor ProtocolVersion in the DDSI-RTPS header matches.
productId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the productId in the DDSI-RTPS header matches.
readerEntityId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the readerEntity ID in a DDSI-RTPS submessage matches
submessage Type	PositiveInteger	0..1	attr	Defines the allowed submessage type in the DDSI-RTPS message
vendorId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the vendorId in the DDSI-RTPS header matches.
writerEntityId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the writerEntity ID in a DDSI-RTPS submessage matches

Table 12.117: DdsRule

12.14 State Management Target-Configuration

12.14.1 Overview

On the *AUTOSAR adaptive platform*, state management can be implemented as a mixture of generated and manually written code. The generated code is created based on a model that is described in this chapter.

Another aspect of this hybrid nature of the state management is that it sometimes acts like an application and sometimes it acts like a functional cluster.

The entry point for the target-configuration of the state management functional cluster on the AUTOSAR adaptive platform is represented by meta-class `StateManagementModuleInstantiation`, a sub-class of `NonOsModuleInstantiation`.

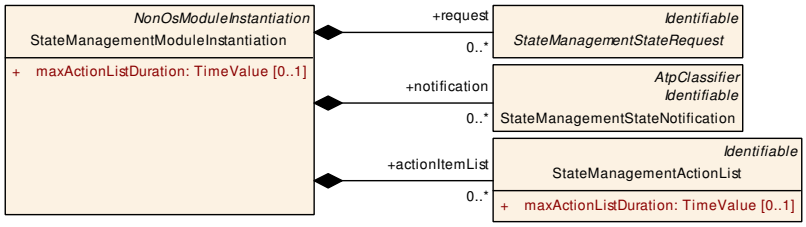


Figure 12.69: Modeling of the `StateManagementModuleInstantiation`

Class	**StateManagementModuleInstantiation**			
Note	This meta-class represents the target-configuration-level configuration of the state management on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
actionItemList	StateManagement ActionList	*	aggr	This represents the collection of action item lists defined in the context of the enclosing state management module. **Tags:** atp.Splitkey=actionItemList, actionItemList.shortName atp.Status=draft
functionGroup ErrorMapping	FunctionGroupError Mapping	*	aggr	This aggregation continas the collection of FunctionGroup ErrorMappings in the context of the enclosing State ManagementModuleInstantiation. **Tags:** atp.Splitkey=functionGroupErrorMapping atp.Status=draft
maxActionList Duration	TimeValue	0..1	attr	This attribute defines a global value for the maximum duration in which any enclosed StateManagementAction List shall finish execution. the value in this attribute will be superseded by the definition of attribute maxActionList Duration in the context of a specific StateManagement ActionList **Tags:** atp.Status=draft
nmNetwork HandleError Mapping	NmNetworkHandleError Mapping	*	aggr	This aggregation contains the collection of `NmNetworkHandleErrorMapping`s in the context of the enclosing `StateManagementModuleInstantiation`. **Tags:** atp.Splitkey=nmNetworkHandleErrorMapping atp.Status=draft
notification	StateManagementState Notification	*	aggr	This aggregation represents the state switch notifications handled by the state manager. **Tags:** atp.Splitkey=notification, notification.shortName atp.Status=draft
request	StateManagementState Request	*	aggr	This aggregation represents the state requests handled by the state manager. **Tags:** atp.Splitkey=request, request.shortName atp.Status=draft

Table 12.118: StateManagementModuleInstantiation

[constr_10641] Existence of reference in the role `StateManagementModuleInstantiation.maxActionListDuration`

Status: DRAFT
Imposition time: `IT_Mani`

⌈For each `StateManagementModuleInstantiation`, the attribute `maxActionListDuration` shall exist.⌋

The job of the State Management on the AUTOSAR adaptive platform is to control two kinds of states:

- the state of a Function Group, modeled as a ModeDeclarationGroup-Prototype owned by a FunctionGroupSet, that is referenced by a SoftwareCluster in the role claimedFunctionGroup.

  These states are only indirectly controlled, because the actual state switch for Function Groups is in the responsibility of the Execution Management.

- the state of an "internal" state machine, owned by StateManagementState-Notifications.

12.14.2 Requests to State Management

[TPS_MANI_01391] Semantics of StateManagementStateRequest

Status: DRAFT

⌈Meta-class StateManagementStateRequest that is aggregated directly at the StateManagementModuleInstantiation represents the ability to configure requests to State Management.⌋

Meta-class StateManagementStateRequest is modeled as an abstract class so that the concrete sub-classes that correspond to different flavors of requests (which are also respected for the definition of applicable PortInterfaces, see Chapter 4.5) can be individually considered.

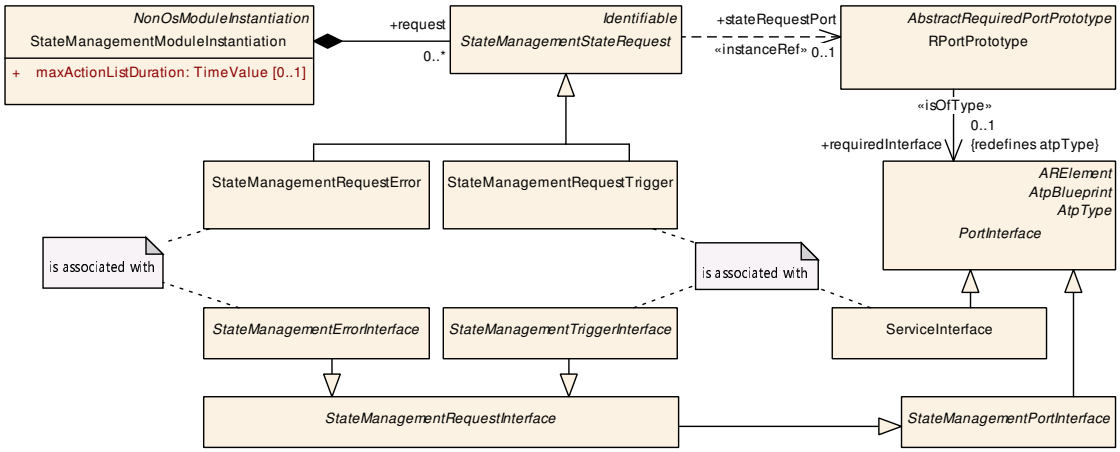


Figure 12.70: Modeling of StateManagementStateRequest

In particular, the State Management accepts requests for state changes in two different flavors:

- "triggers" (i.e. request for a state change from select functional clusters), modeled by mean of meta-class StateManagementRequestTrigger, see Chapter 12.14.2.1.

- "errors" (i.e. error report from a functional cluster), modeled by means of meta-class StateManagementRequestError, see Chapter 12.14.2.2.

As depicted in Figure 12.70, StateManagementStateRequest also references an RPortPrototype as the source of inputs from outside the State Management.

Figure 12.71 also shows that the configuration of State Management on the AUTOSAR adaptive platform is modeled as a hybrid of target-configuration-level StateManagementModuleInstantiation and the definition of an Executable that is executed as a Process.

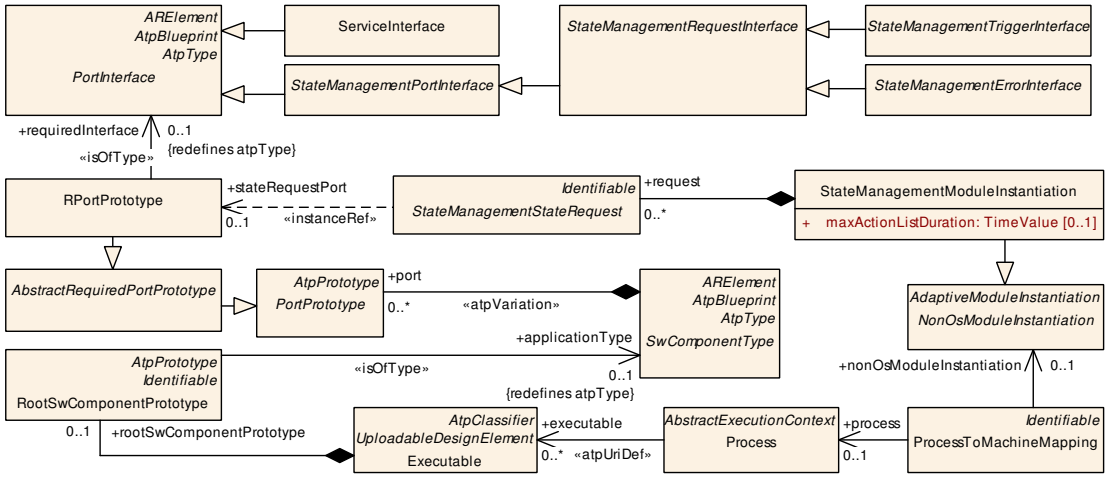


Figure 12.71: Modeling of StateManagementStateRequest including the association with the modeling of Executable

The connection between the StateManagementModuleInstantiation and the Executable/Process is created by ProcessToMachineMapping that references both the StateManagementModuleInstantiation and the Process, as depicted in Figure 12.1.

Class	***StateManagementStateRequest*** (abstract)			
Note	This abstract class serves as the base class for state requests on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	StateManagementRequestError, StateManagementRequestTrigger			
Aggregated by	StateManagementModuleInstantiation.request			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
stateRequestPort	RPortPrototype	0..1	iref	This represents the RPortPrototype in the application software that is issuing the request for state change. **Tags:** atp.Splitkey=stateRequestPort, stateRequestPort.contextRootSwComponentPrototype, stateRequestPort.contextComponentPrototype, stateRequestPort.targetRPortPrototype atp.Status=draft **InstanceRef implemented by:** RPortPrototypeInExecutableInstanceRef

Table 12.119: StateManagementStateRequest

[constr_10390] Existence of reference `StateManagementStateRequest.stateRequestPort`

Status: DRAFT
Imposition time: IT_Mani

⌈Unless the StateManagementStateRequest is referenced from a NmInteractsWithSmMapping, the reference in the role stateRequestPort shall exist.⌋

[constr_10511] Reference `StateManagementStateRequest.stateRequestPort` shall not exist if `StateManagementStateRequest` is referenced from `NmInteractsWithSmMapping`

Status: DRAFT
Imposition time: IT_Mani

⌈The reference StateManagementStateRequest.stateRequestPort shall not exist if StateManagementStateRequest is referenced from NmInteractsWithSmMapping.⌋

12.14.2.1 Trigger Requests to State Management

Trigger requests to State Management are configured by means of meta-class StateManagementRequestTrigger.

As depicted by Figure 12.70, the existence of a StateManagementRequestTrigger is associated with either a subclass of StateManagementTriggerInterface or a ServiceInterface. This relation is formalized in [constr_10384].

[constr_10384] `PortInterface` used for trigger state requests

Status: DRAFT
Imposition time: IT_Mani

⌈Each RPortPrototype that is referenced by a StateManagementRequestTrigger shall be typed by ether

- a subclass of StateManagementTriggerInterface or
- a ServiceInterface.

⌋

Class	**StateManagementRequestTrigger**			
Note	This meta-class has the ability to configure a trigger request to the state management. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementStateRequest*			
Aggregated by	StateManagementModuleInstantiation.request			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**StateManagementRequestTrigger**			
rule	StateManagement RequestRule	*	aggr	This aggregation represents the collection of rules applicable for the trigger request. **Tags:** atp.Splitkey=rule atp.Status=draft

Table 12.120: StateManagementRequestTrigger

12.14.2.2 Error Requests to State Management

Error submissions to State Management are configured by means of meta-class `StateManagementRequestError`, see Figure 12.70.

As depicted by Figure 12.70, the existence of a `StateManagementRequestTrigger` is associated with a subclass of `StateManagementTriggerInterface`. This relation is formalized in [constr_10385].

[constr_10385] `PortInterface` used for error state requests

Status: DRAFT

Imposition time: `IT_Mani`

⌈Each `RPortPrototype` that is referenced by a `StateManagementRequestError` shall be typed by subclass of `StateManagementErrorInterface`.⌋

Class	**StateManagementRequestError**			
Note	This meta-class has the ability to configure the submission of an error to the state management. **Tags:** atp.Status=draft atp.recommendedPackage=StateManagementRequests This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementStateRequest*			
Aggregated by	StateManagementModuleInstantiation.request			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
rule	StateManagement RequestRule	*	aggr	This aggregation represents the collection of rules applicable for the error request. **Tags:** atp.Splitkey=rule atp.Status=draft

Table 12.121: StateManagementRequestError

12.14.3 Notifications from State Management

[TPS_MANI_01392] Semantics of `StateManagementStateNotification`

Status: DRAFT

⌈Notifications are used to make state users aware of state changes. Meta-class `StateManagementStateNotification` is aggregated at `StateManagementModuleInstantiation` and in turn aggregates `ModeDeclarationGroupPrototype` and therefore one `StateManagementStateNotification` becomes the **owner of one internal state machine**.⌋

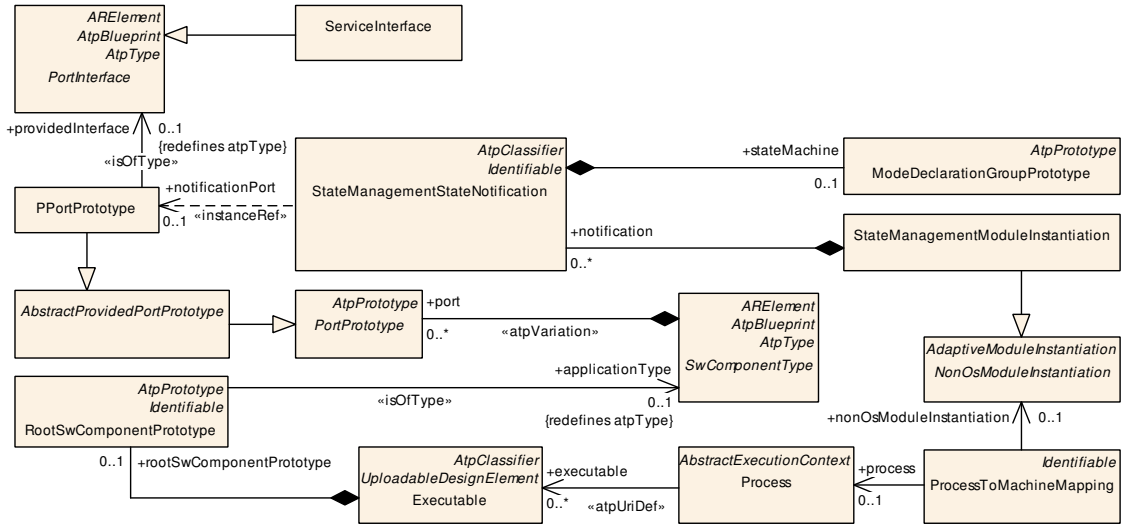


Figure 12.72: Modeling of `StateManagementStateNotification`

It is positively supported that several `StateManagementStateNotification` aggregate a `ModeDeclarationGroupPrototype` typed by the identical `ModeDeclarationGroup` as their internal state machine. This way, several instances of the same “type” of state machine are created.

A `StateManagementModuleInstantiation` references a `PPortPrototype` (typed by a `ServiceInterface`) that shall be taken to provide the actual state notification.

[constr_10388] Restriction for a `PortInterface` used for state switch notifications

Status: DRAFT
Imposition time: `IT_Mani`

⌈Each `PPortPrototype` that is referenced by a `StateManagementStateNotification` shall be typed by a `ServiceInterface`.⌋

[constr_10391] Existence of attribute `StateManagementStateNotification.notificationPort`

Status: DRAFT
Imposition time: `IT_Mani`

⌈For each `StateManagementStateNotification`, the aggregation in the role `notificationPort` shall exist.⌋

The `StateManagementStateNotification.stateMachine` modeled as a `ModeDeclarationGroupPrototype` that is typed by a `ModeDeclarationGroup` shall have a defined `ModeDeclarationGroup.initialMode` according to [constr_1973] [2, TPS Software Component Template] that is also valid for this specification.

Figure 12.72 also depicts the relation between the modeled `Executable` that implements the State Management functionality and the target-configuration aspects represented by the `StateManagementModuleInstantiation`.

Class	**StateManagementStateNotification**			
Note	This meta-class represents the ability to formalize state notifications on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpClassifier*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	StateManagementModuleInstantiation.notification			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
notificationPort	PPortPrototype	0..1	iref	This instanceRef identifies the PPortPrototype over which the notification is to be conveyed. **Tags:** atp.Splitkey=notificationPort, notificationPort.contextRoot SwComponentPrototype, notificationPort.context ComponentPrototype, notificationPort.targetPPort Prototype atp.Status=draft **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
stateMachine	ModeDeclarationGroup Prototype	0..1	aggr	This aggregation represents the existence of an actual state machine. **Tags:** atp.Splitkey=stateMachine, stateMachine.shortName atp.Status=draft

Table 12.122: StateManagementStateNotification

12.14.4 Rules

12.14.4.1 Overview

[TPS_MANI_01384] Semantics of `StateManagementRequestRule`

Status: DRAFT

⌈The purpose of `StateManagementRequestRule` is to evaluate an incoming request against a given comparison value.

If the evaluation of the formula owned by the StateManagementRequestRule yields true, then a state machine (modeled as a ModeDeclarationGroup-Prototype) is forwarded from the current state (represented by meta-class Mode-Declaration) to the nextState, as specified by the definition of the specific sub-class of StateManagementRequestRule.⌋



Figure 12.73: Modeling of StateManagementRequestRule

Class	**StateManagementRequestRule**			
Note	This meta-class represents a rule for deciding about a state change. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	StateManagementRequestError.rule, StateManagementRequestTrigger.rule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
formula	StateManagement CompareFormula	0..1	aggr	This aggregation represents the definition of the formula for the StateManagementRequestRule **Tags:** atp.Splitkey=formula atp.Status=draft
nextState	ModeDeclaration	0..1	iref	This reference identifies the state to be switched to if the condition is fulfilled. **Tags:** atp.Splitkey=nextState, nextState.contextMode DeclarationGroupPrototype, nextState.targetMode Declaration atp.Status=draft **InstanceRef implemented by:** ModeDeclarationInState ManagementStateNotificationInstanceRef

Table 12.123: StateManagementRequestRule

[constr_10392] Existence of attribute StateManagementRequestRule.formula

Status: DRAFT
Imposition time: IT_Mani

⌈For each StateManagementRequestRule, the aggregation in the role formula shall exist.⌋

[constr_10393] Existence of reference in the role StateManagementRequest-Rule.nextState

Status: DRAFT
Imposition time: IT_Mani

⌈For each StateManagementRequestRule, the reference in the role nextState shall exist.⌋

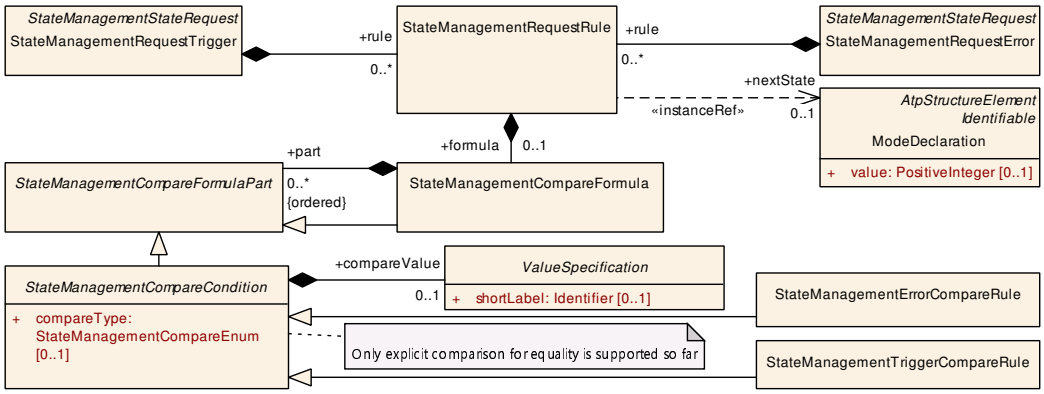


Figure 12.74: Relation between `StateManagementStateRequest` and `StateManagementRequestRule`

The general approach for rule formulation is very generic, although the State Management for the time being has only little requirements for rule evaluation: one value is checked against a reference value. But since the complexity of rule formulation might grow over time, the model is already prepared for future extensions.

The meta-model supports the formulation of hierarchical comparisons such that state changes could be evaluated against not only a single input, but also – where applicable – collection of inputs.

The execution of state changes is subject to the evaluation of rules that are owned by the respective request classes. This relation is depicted in Figure 12.74.

The states that are involved in the evaluation of State Management rules are entirely "internal" states, `Function Group State`s are not affected and shall[2] not be used for rule evaluation.

Class	**StateManagementCompareFormula**			
Note	A StateManagementCompareFormula embodies the computation instruction that is to be evaluated at runtime to determine if the aggregating request rule yields true or false. The formula itself consists of parts which are combined by the logical operations specified by StateManagementCompareFormula .op. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *StateManagementCompareFormulaPart*			
Aggregated by	StateManagementCompareFormula.part, StateManagementRequestRule.formula			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
part (ordered)	StateManagement CompareFormulaPart	*	aggr	This aggregation represents the collection of formula parts that can be combined by logical operators. **Tags:** atp.Status=draft

Table 12.124: StateManagementCompareFormula

[2]The nature of the underlying `instanceRef` modeling would unfortunately not be able to prohibit a brute-force referencing of function groups states.

Class	**StateManagementCompareFormulaPart** (abstract)			
Note	A StateManagementCompareFormulaPart can either be a atomic condition, e.g. a StateManagement TriggerCompareRule or StateManagementErrorCompareRule, or a StateManagementCompareFormula again, which allows arbitrary nesting. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Subclasses	*StateManagementCompareCondition*, StateManagementCompareFormula			
Aggregated by	StateManagementCompareFormula.part			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 12.125: StateManagementCompareFormulaPart

Class	**StateManagementCompareCondition** (abstract)			
Note	StateManagementCompareConditions are atomic conditions. They are based on the idea of a comparison at runtime of some variable data with something constant. The type of the comparison (==, !=, <, <=, ...) is specified in StateManagementCompareCondition.compareType. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *StateManagementCompareFormulaPart*			
Subclasses	StateManagementErrorCompareRule, StateManagementTriggerCompareRule			
Aggregated by	StateManagementCompareFormula.part			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
compareType	StateManagement CompareEnum	0..1	attr	This attributes represents the concrete type of the comparison. **Tags:** atp.Status=draft
compareValue	ValueSpecification	0..1	aggr	This aggregation represents the reference value against which the value obtained from request shall be compared to. **Tags:** atp.Status=draft

Table 12.126: StateManagementCompareCondition

[constr_10394] Existence of attribute `StateManagementCompareCondition.compareType`

Status: DRAFT
Imposition time: `IT_Mani`

⌈For each `StateManagementCompareCondition`, the aggregation in the role `compareType` shall exist.⌋

[constr_10395] Existence of attribute `StateManagementCompareCondition.compareValue`

Status: DRAFT
Imposition time: `IT_Mani`

⌈For each `StateManagementCompareCondition`, the aggregation in the role `compareValue` shall exist.⌋

Enumeration	**StateManagementCompareEnum**
Note	Enumeration for the type of a comparison of values usually expressed by the following operators: ==, !=, <, <=, >, >= **Tags:** atp.Status=draft This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	*StateManagementCompareCondition*.compareType
Literal	***Description***
isEqual	Equal **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=draft

Table 12.127: StateManagementCompareEnum

12.14.4.2 Trigger Rules

The modeling of the rules for trigger requests is depicted in Figure 12.74 and Figure 12.75.

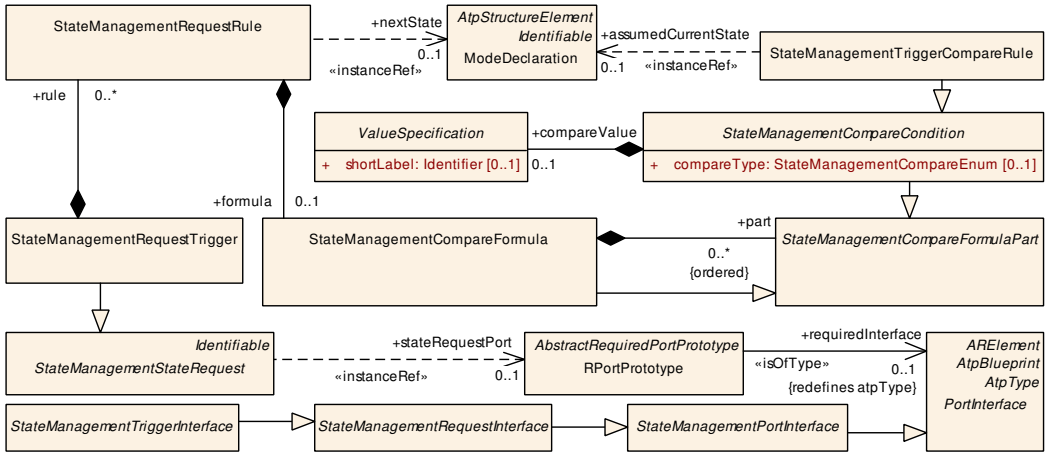


Figure 12.75: Modeling of `StateManagementRequestRule` for the "trigger" case

As depicted in Figure 12.75, the definition of the trigger request rule involves two `instanceRef`s to `ModeDeclarationGroup`, as further explained in

[TPS_MANI_01835] Semantics of `StateManagementRequestTrigger.rule`

Status: DRAFT

⌈The semantics of `StateManagementRequestTrigger.rule` is mainly defined by the existence of two `instanceRef`s:

- The reference in the role `StateManagementRequestTrigger.rule.nextState` denotes the state that shall be switched to if the rule evaluates to `true`.
- The additional reference in the role `StateManagementTriggerCompareRule.assumedCurrentState` represents an input to the rule evaluation. In other words, the `rule.formula` evaluates to `true` if

– the input value obtained from the PortPrototype referenced in the role StateManagementRequestTrigger.stateRequestPort typed by either a sub-class of StateManagementTriggerInterface or a ServiceInterface is equal the compare value **and**

– the reference StateManagementTriggerCompareRule.assumedCurrentState identifies the current state.

⌋

Class	**StateManagementTriggerCompareRule**			
Note	This meta-class represents the configuration of a compare rule for the processing of a trigger request. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *StateManagementCompareCondition*, *StateManagementCompareFormulaPart*			
Aggregated by	StateManagementCompareFormula.part			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
assumed CurrentState	ModeDeclaration	0..1	iref	This reference denotes the assumed current state for the given compare rule for trigger values. **Tags:** atp.Status=draft **InstanceRef implemented by:** ModeDeclarationInStateManagementStateNotificationInstanceRef

Table 12.128: StateManagementTriggerCompareRule

[constr_10396] Existence of reference in the role StateManagementTriggerCompareRule.assumedCurrentState

Status: DRAFT
Imposition time: IT_Mani

⌈For each StateManagementTriggerCompareRule, the reference in the role assumedCurrentState shall exist.⌋

12.14.4.3 Error Rules

The modeling of the rules for error submissions is depicted in Figure 12.74 and Figure 12.76.

[TPS_MANI_01836] Semantics of StateManagementRequestError.rule

Status: DRAFT

⌈The semantics of the rule is that the affected state machine shall switch to the nextState if the formula evaluates to true.⌋

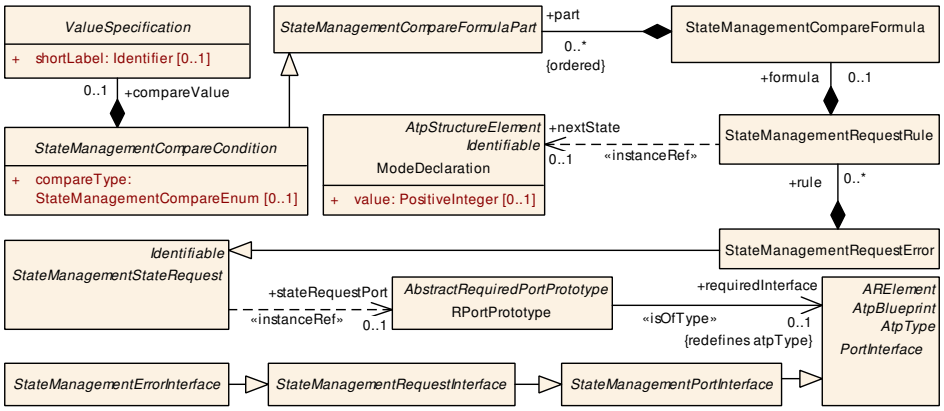


Figure 12.76: Modeling of `StateManagementRequestRule` for the "error" case

Class	**StateManagementErrorCompareRule**			
Note	This meta-class represents the configuration of a compare rule for the processing of an error submission. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *StateManagementCompareCondition*, *StateManagementCompareFormulaPart*			
Aggregated by	StateManagementCompareFormula.part			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 12.129: StateManagementErrorCompareRule

12.14.5 Action Items

[TPS_MANI_01387] Semantics of meta-class `StateManagementActionList`

Status: DRAFT

⌈Whenever a state machine switches to a different state, one or more actions shall be executed in response.

This functionality is modeled by means of the existence of a `StateManagementActionList` that aggregates abstract class `StateManagementActionItem`s as an **ordered** collection, i.e. the order of the execution of individual `StateManagementActionItem`s is significant.

The `StateManagementActionList` refers to a `ModeDeclaration` (an "internal" state, as opposed to a `Function Group State`) as the representation of the state to which the state machine was switched to.⌋

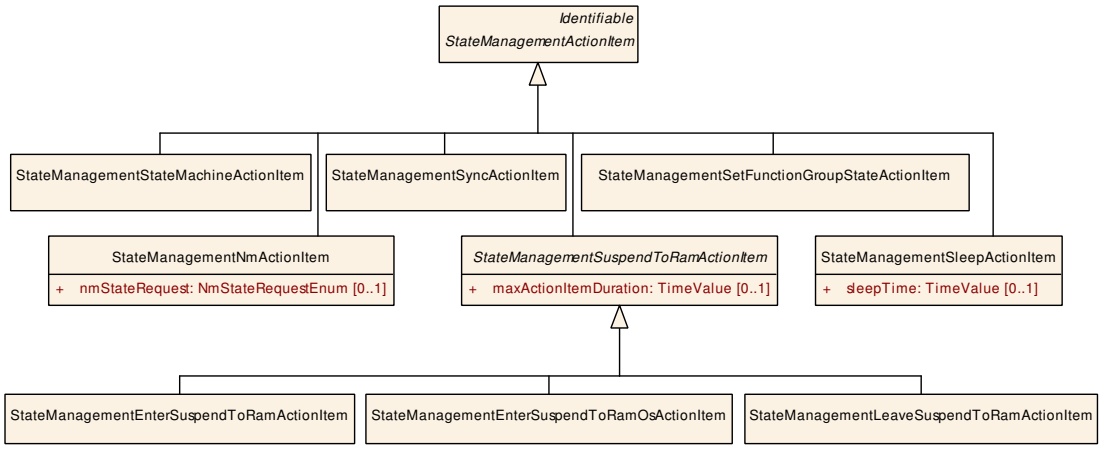


Figure 12.77: Modeling of `StateManagementActionItem`

Class	**StateManagementActionList**			
Note	This meta-class represents the ability to define an action list that is associated with a state of a state machine. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	StateManagementModuleInstantiation.actionItemList			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
actionItem (ordered)	StateManagement ActionItem	*	aggr	This represents the collection of action items in the context of the action item list. **Tags:** atp.Status=draft
actionList Processing FailedError	ApApplicationError	0..1	ref	This reference identifies the error code for the case that the enclosing action list fails to process successfully. This reference is only relevant for state management agents. **Tags:** atp.Splitkey=actionListProcessingFailedError atp.Status=draft
affectedState	ModeDeclaration	0..1	iref	This reference identifies the state for which the referencing action list applies. **Tags:** atp.Splitkey=affectedState, affectedState.contextMode DeclarationGroupPrototype, affectedState.targetMode Declaration atp.Status=draft **InstanceRef implemented by:** ModeDeclarationInState ManagementStateNotificationInstanceRef
maxActionList Duration	TimeValue	0..1	attr	This attribute defines the maximum duration in which the enclosing StateManagementActionList shall finish execution **Tags:** atp.Status=draft
maxDuration ExceededError	ApApplicationError	0..1	ref	This reference identifies the ApApplicationError that shall be triggered if the configured maximum duration of the execution of the action item list is exceeded. **Tags:** atp.Splitkey=maxDurationExceededError atp.Status=draft

Table 12.130: StateManagementActionList

[constr_10405] Existence of reference in the role StateManagementActionList.affectedState

Status: DRAFT
Imposition time: IT_Mani

⌈For each StateManagementActionList, the reference in the role affectedState shall exist.⌋

Class	***StateManagementActionItem*** (abstract)			
Note	This meta-class represents an action item that is executed in response to a state change. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	StateManagementNmActionItem, StateManagementSetFunctionGroupStateActionItem, StateManagementSleepActionItem, StateManagementStateMachineActionItem, *StateManagementSuspendToRamActionItem*, StateManagementSyncActionItem			
Aggregated by	StateManagementActionList.actionItem			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 12.131: StateManagementActionItem

12.14.5.1 State Machine Action Item

[TPS_MANI_01388] Semantics of the meta-class StateManagementStateMachineActionItem

Status: DRAFT

⌈A StateManagementStateMachineActionItem can be used to start or stop a state machine in the context of State Management.

This activity is typically needed to interact with subordinate State Management software that comes with a specific SoftwareCluster and is responsible for the State Management of the enclosing SoftwareCluster.⌋

[TPS_MANI_01436] Relation of StateManagementStateMachineActionItem to internal state machine instances

Status: DRAFT

⌈The modeling of the relation between a StateManagementStateMachineActionItem and the affected state machines (represented by ModeDeclarationGroupPrototype) assumes that the internal state machines are located in the context of the same Executable/Process as the StateManagementStateMachineActionItem.

In other words, an inter-process communication is not foreseen for starting and stopping internal state machines.⌋

As mentioned by [TPS_MANI_01388], the purpose of the StateManagementStateMachineActionItem is to either start or stop a given state machine. There is therefore no valid use case for both references in the role startAgent and stopAgent to exist at the same time.

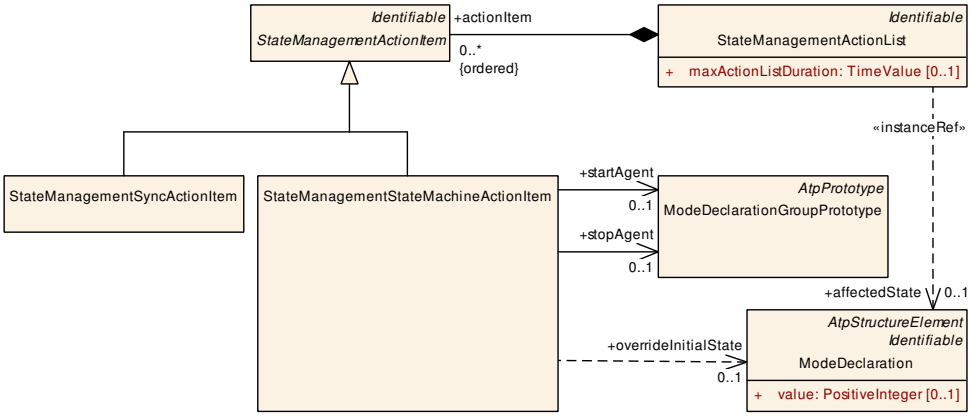


Figure 12.78: Modeling of StateManagementStateMachineActionItem

[constr_10386] Existence of references StateManagementStateMachineActionItem.startAgent and stopAgent

Status: DRAFT
Imposition time: IT_Mani

⌈For each StateManagementStateMachineActionItem, **at most one** of the two references

- startAgent
- stopAgent

shall exist.⌋

The roles startAgent and stopAgent can only be applied to a StateManagementStateNotification.stateMachine where attribute category is set to the value STATE_MANAGEMENT_AGENT and never to a StateManagementStateNotification.stateMachine where attribute category is set to the value STATE_MANAGEMENT_CONTROLLER. This restriction is formalized in [constr_10666].

[constr_10666] Restriction for StateManagementStateMachineActionItem

Status: DRAFT
Imposition time: IT_Mani

⌈No StateManagementStateMachineActionItem shall refer (in the roles startAgent and stopAgent) to any StateManagementStateNotification.stateMachine where attribute category is set to the value STATE_MANAGEMENT_CONTROLLER.⌋

[TPS_MANI_01437] Standardized values of attribute StateManagementStateNotification.stateMachine.category

Status: DRAFT

⌈The following values of attribute StateManagementStateNotification.stateMachine.category are standardized:

STATE_MANAGEMENT_CONTROLLER the aggregated StateManagementStateNotification.stateMachine represents the state machine that starts all other state machines.

STATE_MANAGEMENT_AGENT the aggregated StateManagementStateNotification.stateMachine represents a state machine that is started by a "controller".

⌋

[constr_10489] Existence of StateManagementModuleInstantiation in the context of a Machine

Status: DRAFT
Imposition time: IT_Mani

⌈Each Machine shall only aggregate at most one StateManagementModuleInstantiation in the role moduleInstantiation.⌋

[TPS_MANI_01438] Existence of StateManagementStateMachineActionItem.overrideInitialState ⌈If the instance reference in the role StateManagementStateMachineActionItem.overrideInitialState exists, then the references

- StateManagementStateMachineActionItem.startAgent.type.initialMode
- StateManagementStateMachineActionItem.stopAgent.type.initialMode

shall be ignored.⌋

[constr_10633] Existence of references StateManagementStateMachineActionItem.overrideInitialState and stopAgent

Status: DRAFT
Imposition time: IT_Mani

⌈If the reference in the role StateManagementStateMachineActionItem.stopAgent exists, then the instance reference in the role overrideInitialState shall **not** exist.⌋

Class	**StateManagementStateMachineActionItem**			
Note	This meta-class represents a state management action item to start or stop a state machine. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementActionItem*			
Aggregated by	StateManagementActionList.actionItem			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
overrideInitial State	ModeDeclaration	0..1	iref	The referenced ModeDeclaration shall be considered the initial state of the context ModeDeclarationGroup Prototype and the corresponding reference Mode DeclarationGroup.initialMode shall be ignored. **Tags:** atp.Splitkey=overrideInitialState, overrideInitial State.contextModeDeclarationGroupPrototype, override InitialState.targetModeDeclaration atp.Status=draft **InstanceRef implemented by:** ModeDeclarationInState ManagementStateNotificationInstanceRef
startAgent	ModeDeclarationGroup Prototype	0..1	ref	This reference identifies the state machine that shall be started when the enclosing action list item is executed. **Tags:** atp.Splitkey=startAgent atp.Status=draft
startAgentError	ApApplicationError	0..1	ref	This reference identifies the error that shall be raised if the staring of an agent failed. **Tags:** atp.Splitkey=startAgentError atp.Status=draft
stopAgent	ModeDeclarationGroup Prototype	0..1	ref	This reference identifies the state machine that shall be stopped when the enclosing action list item is executed. **Tags:** atp.Splitkey=stopAgent atp.Status=draft
stopAgentError	ApApplicationError	0..1	ref	This reference identifies the error that shall be raised if the stopping of an agent failed. **Tags:** atp.Splitkey=stopAgentError atp.Status=draft

Table 12.132: StateManagementStateMachineActionItem

12.14.5.2 Sync Action Item

[TPS_MANI_01389] Semantics of meta-class `StateManagementSyncAction-Item`

Status: DRAFT

⌈A `StateManagementSyncActionItem` can be used to introduce a synchronization point at which the workflow waits for the completion of actions that could (in theory) be executed in parallel.

In other words, any `StateManagementActionItem` positioned after the `State-ManagementSyncActionItem` can only be executed if the `StateManagement-ActionItem` placed before the `StateManagementSyncActionItem` are finished.⌋

Class	StateManagementSyncActionItem			
Note	This meta-class represents a state management action item to synchronize state machines. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementActionItem*			
Aggregated by	StateManagementActionList.actionItem			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 12.133: StateManagementSyncActionItem

12.14.5.3 Function Group Action Item

[TPS_MANI_01390] Semantics of meta-class `StateManagementSetFunctionGroupStateActionItem`

Status: DRAFT

⌈The purpose of `StateManagementSetFunctionGroupStateActionItem` is to enable the switch of the state machine representing a specific `Function Group` to a new `Function Group State`, identified by means of the reference in the role `StateManagementSetFunctionGroupStateActionItem.setFunctionGroupState`.⌋

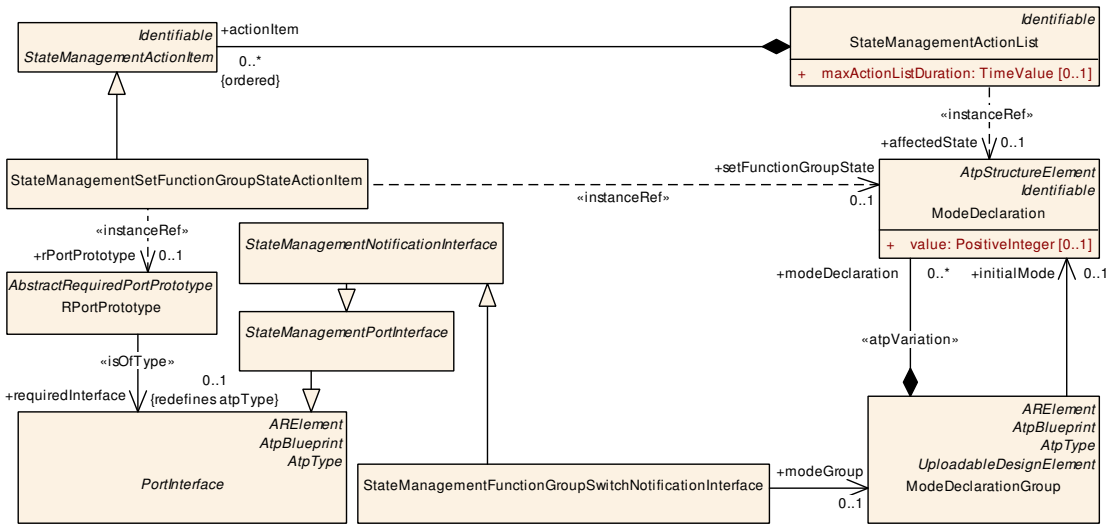


Figure 12.79: Modeling of `StateManagementSetFunctionGroupStateActionItem`

Each `StateManagementSetFunctionGroupStateActionItem` references

- a `ModeDeclaration` that represents a `Function Group State` and
- a `PPortPrototype` over which the switch to the new `Function Group State` shall be communicated.

The ModeDeclarationGroup that is used in the context of the definition of the Function Group is identical[3] to the definition of the StateManagementFunctionGroupSwitchNotificationInterface.modeGroup. This consequence is the basis for the definition of [constr_10387].

[constr_10387] Consistency of StateManagementSetFunctionGroupStateActionItem.rPortPrototype and StateManagementSetFunctionGroupStateActionItem.setFunctionGroupState

Status: DRAFT
Imposition time: IT_Mani

⌈For each StateManagementSetFunctionGroupStateActionItem, the ModeDeclarationGroup used to type the ModeDeclaration that is referenced in the role setFunctionGroupState shall be identical to the ModeDeclarationGroup referenced in the role modeGroup from the StateManagementFunctionGroupSwitchNotificationInterface that is used to type the PPortPrototype that is referenced in the role rPortPrototype from the affected StateManagementSetFunctionGroupStateActionItem.⌋

The relation described in [constr_10387] is depicted in Figure 12.79.

Class	**StateManagementSetFunctionGroupStateActionItem**			
Note	This meta-class represents a state management action item to set a specific state in a specific function group. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementActionItem*			
Aggregated by	StateManagementActionList.actionItem			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
rPortPrototype	RPortPrototype	0..1	iref	This reference identifies the PortPrototype over which the function group state switch shall be communicated. **Tags:** atp.Splitkey=rPortPrototype, rPortPrototype.contextRoot SwComponentPrototype, rPortPrototype.context ComponentPrototype, rPortPrototype.targetRPort Prototype atp.Status=draft **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef
setFunction GroupState	ModeDeclaration	0..1	iref	This reference identifies the function group step that shall become active after the action step terminates. **Tags:** atp.Splitkey=setFunctionGroupState, setFunction GroupState.contextModeDeclarationGroupPrototype, set FunctionGroupState.targetModeDeclaration **InstanceRef implemented by:** FunctionGroupStateIn FunctionGroupSetInstanceRef

Table 12.134: StateManagementSetFunctionGroupStateActionItem

[3]Technically, it would be sufficient if the two affected ModeDeclarationGroups were compatible. But the utilization of the concept of compatibility doesn't (yet) have the same foothold on the *AUTOSAR adaptive platform* as it has on the *AUTOSAR classic platform*.

[constr_10397] Existence of reference in the role StateManagementSetFunctionGroupStateActionItem.rPortPrototype

Status: DRAFT
Imposition time: IT_Mani

⌈For each StateManagementSetFunctionGroupStateActionItem, the reference in the role rPortPrototype shall exist.⌋

[constr_10398] Existence of reference in the role StateManagementSetFunctionGroupStateActionItem.setFunctionGroupState

Status: DRAFT
Imposition time: IT_Mani

⌈For each StateManagementSetFunctionGroupStateActionItem, the reference in the role setFunctionGroupState shall exist.⌋

12.14.5.4 Network Management Action Item

[TPS_MANI_01443] Semantics of meta-class StateManagementNmActionItem

Status: DRAFT

⌈A StateManagementNmActionItem can be used to request a communication state of the network management to become active. At the moment, the following communication states are supported:

- NmStateRequestEnum.noCom: network management switches communication off.
- NmStateRequestEnum.fullCom: network management switches communication on.

⌋

The switch is applied at the granularity of the so-called *network management handle*, formalized as NmNetworkHandle. The relation of the StateManagementNmActionItem to the NmNetworkHandle is defined by meta-class SmInteractsWithNmMapping.

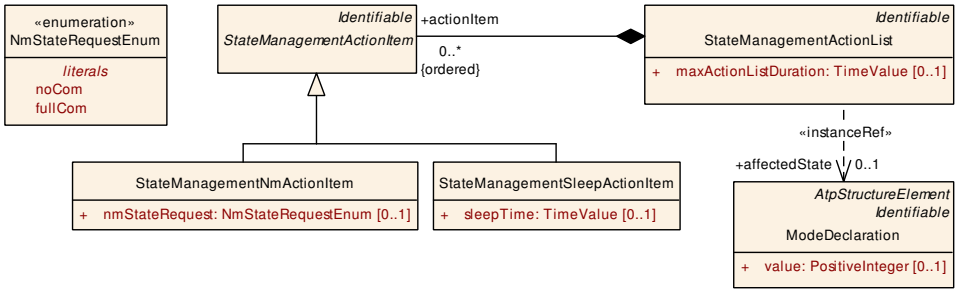


Figure 12.80: Modeling of StateManagementNmActionItem

[constr_10779] Reference to NM Network Handle from NmNetworkHandleError-Mapping

Status: DRAFT
Imposition time: IT_Mani

⌈Each NM network handle modeled as a NmNetworkHandle aggregated by a NmInstantiation shall only be referenced by a single NmNetworkHandleErrorMapping.⌋

The modeling of NmNetworkHandleErrorMapping is depicted in Figure 12.84.

Class	**StateManagementNmActionItem**			
Note	This meta-class represents a state management action item to interact with the network management. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementActionItem*			
Aggregated by	StateManagementActionList.actionItem			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
nmState Request	NmStateRequestEnum	0..1	attr	This attribute defines the target network management state that is requested by state management. **Tags:** atp.Status=draft

Table 12.135: StateManagementNmActionItem

Enumeration	**NmStateRequestEnum**
Note	This enumeration defines the description of states that can be requested from the network management. **Tags:** atp.Status=draft This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	StateManagementNmActionItem.nmStateRequest
Literal	***Description***
fullCom	This literal represents that case that full communication should be possible. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=draft
noCom	This literal represents that case that no communication should be possible. **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=draft

Table 12.136: NmStateRequestEnum

Class	**NmNetworkHandleErrorMapping**
Note	This meta-class is used to associate an error code to a network handle. The error code shall be used if any network management activity on the network handle fails. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARObject*, *Describable*
Aggregated by	StateManagementModuleInstantiation.nmNetworkHandleErrorMapping

▽

△

Class	NmNetworkHandleErrorMapping			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
nmNetwork Handle	NmNetworkHandle	0..1	ref	This reference identifies the NM network handle that is affected by an error raised if a request to transition the state of the NM network handle failed. **Tags:** atp.Splitkey=nmNetworkHandle atp.Status=draft
nmNetwork Management Error	ApApplicationError	0..1	ref	This reference identifies the `ApApplicationError` to be raised if a request to transition an NM state failed. **Tags:** atp.Splitkey=nmNetworkManagementError atp.Status=draft

Table 12.137: NmNetworkHandleErrorMapping

12.14.5.5 Sleep Action Item

[TPS_MANI_01445] Semantics of meta-class `StateManagementSleepAction-Item`

Status: DRAFT

⌈A `StateManagementSleepActionItem` can be used to make the execution of a `StateManagementActionList` pause for some time.

The purpose is to give external code time for completion until the execution of the `StateManagementActionList` commences.⌋

[TPS_MANI_01444] Semantics of attribute `StateManagementSleepAction-Item.sleepTime`

Status: DRAFT

⌈Attribute `StateManagementSleepActionItem.sleepTime` can be used to define a time interval, after which the action represented by the `StateManagementSleep-ActionItem` is expected to be complete. The action is considered failed should the timeout expire without completion.⌋

[constr_10512] Restriction for the value of `StateManagementSleepAction-Item.sleepTime`

Status: DRAFT
Imposition time: `IT_Mani`

⌈For each `StateManagementSleepActionItem`, the value of attribute `sleepTime` shall only have a value greater than 0.⌋

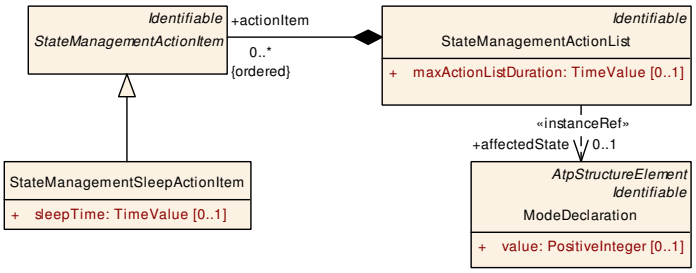


Figure 12.81: Modeling of `StateManagementSleepActionItem`

Class	**StateManagementSleepActionItem**			
Note	This action item can be used to universally implement afterrun. One specific use case for afterrun comes up in the context of network management. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementActionItem*			
Aggregated by	StateManagementActionList.actionItem			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sleepTime	TimeValue	0..1	attr	This attribute represents the amount of time that the execution of the StateManagementActionItemList is supposed to go to sleep. **Tags:** atp.Status=draft

Table 12.138: StateManagementSleepActionItem

12.14.5.6 Suspend to RAM Action Item

[TPS_MANI_01508] Semantics of meta-class `StateManagementSuspendToRam-ActionItem`

Status: DRAFT

⌈A `StateManagementSuspendToRamActionItem` can be used to enter or leave a `suspend-to-RAM` state.⌋

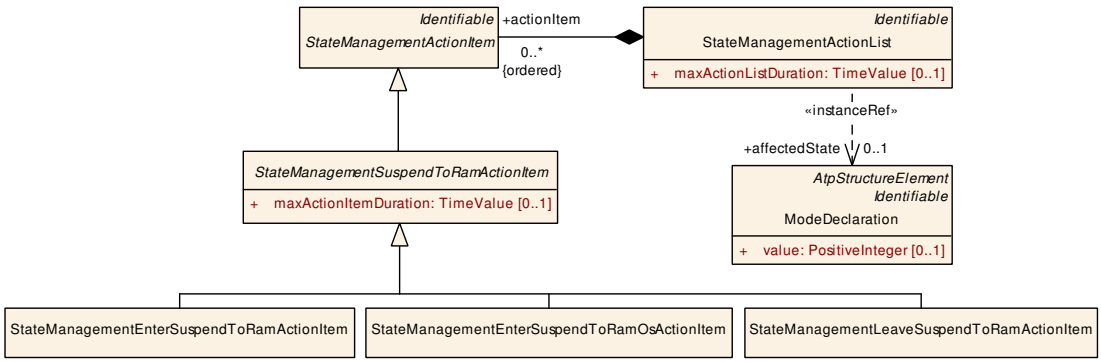


Figure 12.82: Modeling of `StateManagementSuspendToRamActionItem`

Class	**StateManagementSuspendToRamActionItem** (abstract)			
Note	This meta-class serves as an abstract base class for all suspend-to-RAM-related subclasses. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementActionItem*			
Subclasses	StateManagementEnterSuspendToRamActionItem, StateManagementEnterSuspendToRamOsAction Item, StateManagementLeaveSuspendToRamActionItem			
Aggregated by	StateManagementActionList.actionItem			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
maxActionItem Duration	TimeValue	0..1	attr	This attribute denotes the amount of time after which the execution of the action item is considered failed. **Tags:** atp.Status=draft

Table 12.139: StateManagementSuspendToRamActionItem

[constr_10720] Restriction for the value of StateManagementSuspendToRam-ActionItem.maxActionItemDuration

Status: DRAFT
Imposition time: IT_Mani

⌈For each StateManagementSuspendToRamActionItem, the value of attribute maxActionItemDuration shall exist and only have a value greater than 0.⌋

Class	**StateManagementEnterSuspendToRamOsActionItem**			
Note	This meta-class represents a state management action item to trigger the OS to enter the Suspend2Ram state. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementActionItem*, *State ManagementSuspendToRamActionItem*			
Aggregated by	StateManagementActionList.actionItem			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 12.140: StateManagementEnterSuspendToRamOsActionItem

Class	**StateManagementEnterSuspendToRamActionItem**			
Note	This meta-class represents a state management action item to trigger S2R-aware application to enter the Suspend2Ram state. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementActionItem*, *State ManagementSuspendToRamActionItem*			
Aggregated by	StateManagementActionList.actionItem			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**StateManagementEnterSuspendToRamActionItem**			
enterSuspend ToRamError	ApApplicationError	0..1	ref	This reference identifies the error code for the case that entering suspend-to-RAM fails. **Tags:** atp.Splitkey=enterSuspendToRamError atp.Status=draft

Table 12.141: StateManagementEnterSuspendToRamActionItem

Class	**StateManagementLeaveSuspendToRamActionItem**			
Note	This meta-class represents a state management action item to trigger S2R-aware application to leave the Suspend2Ram state. **Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementActionItem*, *StateManagementSuspendToRamActionItem*			
Aggregated by	StateManagementActionList.actionItem			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
leaveSuspend ToRamError	ApApplicationError	0..1	ref	This reference identifies the error code for the case that leaving suspend-to-RAM fails. **Tags:** atp.Splitkey=leaveSuspendToRamError atp.Status=draft

Table 12.142: StateManagementLeaveSuspendToRamActionItem

12.14.6 State Client

Switching of function group states is done by means of an `RPortPrototype` typed by the `StateClientInterface`.

The state client does not know which concrete function group (instance) is changed; it knows only which function group state shall be set (modeled as `ModeDeclaration` in the context of a `ModeDeclarationGroup`, entirely on type level).

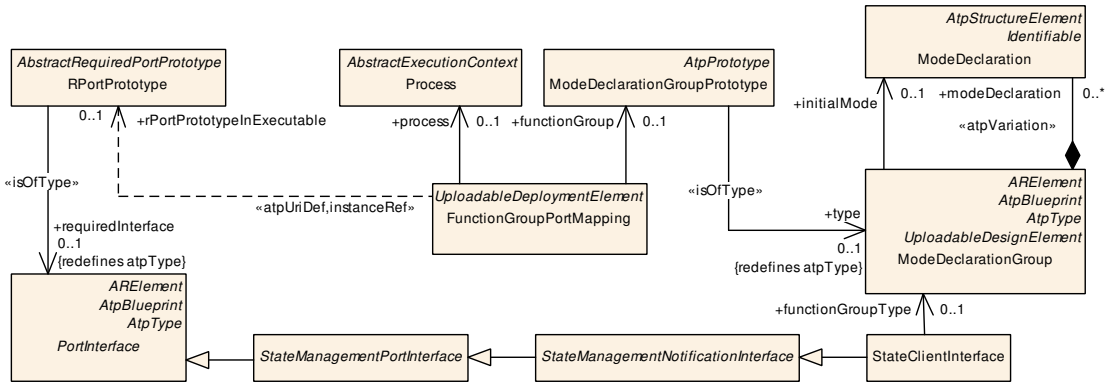


Figure 12.83: Modeling of the configuration for switching a function group state

The relation to the actual function group on instance level is created by the existence of the FunctionGroupPortMapping, which identifies the actual function group, represented by a ModeDeclarationGroupPrototype aggregated by a FunctionGroupSet.

Class	**FunctionGroupPortMapping**			
Note	This class is used to associate a PortPrototype typed by a StateClientInterface with the actual function group to which the state changes communicated over the PortPrototype shall apply. **Tags:** atp.Status=draft atp.recommendedPackage=FunctionGroupPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
functionGroup	ModeDeclarationGroup Prototype	0..1	ref	This reference identifies the applicable function group for which the state change shall be executed. **Tags:** atp.Splitkey=functionGroup atp.Status=draft
process	Process	0..1	ref	This reference identifies the Process of the state client **Tags:** atp.Splitkey=process atp.Status=draft
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This reference identifies the applicable PortPrototype for the function group state change. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=rPortPrototypeInExecutable, rPortPrototype InExecutable.contextRootSwComponentPrototype, rPort PrototypeInExecutable.contextComponentPrototype, r PortPrototypeInExecutable.targetRPortPrototype atp.Status=draft **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 12.143: FunctionGroupPortMapping

[constr_10617] Existence of the reference in the role FunctionGroupPortMapping.functionGroup

Status: DRAFT
Imposition time: IT_Mani

⌈For each FunctionGroupPortMapping, the reference in the role functionGroup shall exist.⌋

[constr_10618] Existence of the reference in the role FunctionGroupPortMapping.process

Status: DRAFT
Imposition time: IT_Mani

⌈For each FunctionGroupPortMapping, the reference in the role process shall exist.⌋

[constr_10619] Existence of the reference in the role `FunctionGroupPortMapping.rPortPrototypeInExecutable`

Status: DRAFT
Imposition time: IT_Mani

⌈For each FunctionGroupPortMapping, the reference in the role rPortPrototypeInExecutable shall exist.⌋

[constr_10620] Reference to function group from `FunctionGroupPortMapping`

Status: DRAFT
Imposition time: IT_Mani

⌈Each function group modeled as a ModeDeclarationGroupPrototype aggregated by a FunctionGroupSet shall only be referenced by a single FunctionGroupPortMapping.⌋

[constr_10621] Same `ModeDeclarationGroup` for `StateClientInterface` and `FunctionGroupPortMapping.functionGroup`

Status: DRAFT
Imposition time: IT_Mani

⌈For each FunctionGroupPortMapping, the ModeDeclarationGroup referenced in the role FunctionGroupPortMapping.rPortPrototypeInExecutable.requiredInterface.functionGroupType shall be identical to the FunctionGroupPortMapping.functionGroup.type.⌋

[constr_10622] Existence of `FunctionGroupPortMapping` depending on the role `rPortPrototypeInExecutable` and `process`

Status: DRAFT
Imposition time: IT_Mani

⌈Each combination of RPortPrototype and Process may be referenced by **at most one** FunctionGroupPortMapping in the roles rPortPrototypeInExecutable and process.⌋

12.14.7 Error Handling inside State Management

The processing of errors inside state management is based on the configuration of references to ApApplicationError from various corners of the state management configuration.

[TPS_MANI_01478] Supported error handling scenarios in the context of state management ⌈The following error scenarios are supported in the context of state management:

- Error while setting up and starting a state management agent, independent of the initial state: StateManagementStateMachineActionItem.startAgentError.

- Error while stopping a state management agent: StateManagementStateMachineActionItem.stopAgentError.
- Error while requesting to switch a function group state: FunctionGroupErrorMapping.functionGroupTransitionRequestFailedError.
- Executing an action item list takes too long: StateManagementActionList.maxDurationExceededError.
- Error while processing of action list fails: StateManagementActionList.actionListProcessingFailedError.
- Error while launching a Process: StateDependentStartupConfig.processExecutionError.
- Error while trying to enter suspend-to-RAM: StateManagementEnterSuspendToRamActionItem.enterSuspendToRamError.
- Error while trying to leave suspend-to-RAM: StateManagementLeaveSuspendToRamActionItem.leaveSuspendToRamError.
- Error while trying to change the state of a NM network handle: NmNetworkHandleErrorMapping.nmNetworkManagementError.

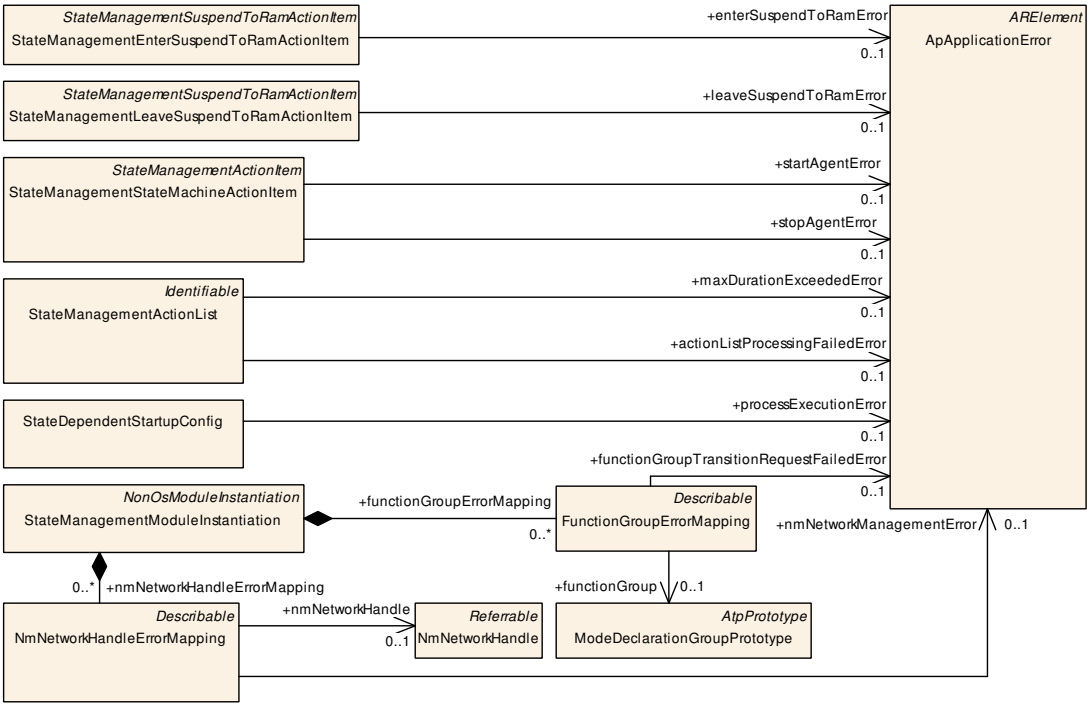


Figure 12.84: Modeling of the configuration for error handling

The error scenarios listed in [TPS_MANI_01478] are also depicted in Figure 12.84.

The modeling of most of the error scenarios is quite straight-forward. The notable exception affects the switching of function group states[4].

12.14.7.1 Request Function Group State

There is no intention to support the assignment of an individual error code for each function group state if the request of a function group state goes wrong. Instead, a single error code is assigned to the entire function group, independent of the state request that may have caused a problem.

For this purpose, meta-class `FunctionGroupErrorMapping` allows for the association of a function group to a **single** error code in the context of a `StateManagementModuleInstantiation`.

Class	**FunctionGroupErrorMapping**			
Note	This meta-class is used to associate an error code to an entire function group. The error code shall be used if any function group state change in the referenced function group fails.			
Base	*ARObject*, *Describable*			
Aggregated by	StateManagementModuleInstantiation.functionGroupErrorMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
functionGroup	ModeDeclarationGroup Prototype	0..1	ref	This reference identifies the function group that is affected by an error raised if a request to transition a function group failed. **Tags:** atp.Splitkey=functionGroup atp.Status=draft This Attribute is only used by the AUTOSAR Adaptive Platform.
functionGroup Transition RequestFailed Error	ApApplicationError	0..1	ref	This reference identifies the ApApplicationError to be raised if a request to transition a function group failed. **Tags:** atp.Splitkey=functionGroupTransitionRequestFailedError atp.Status=draft This Attribute is only used by the AUTOSAR Adaptive Platform.

Table 12.144: FunctionGroupErrorMapping

12.15 Suspend to RAM

The control over the `suspend-to-RAM` functionality implemented in state management, which consequently represents a " hub" in the communication with various pieces of application software, taking over a "satellite" role.

Requests sent out by the hub are either confirmed or denied by the satellites. The communication pattern is sketched in Figure 12.85.

[4]This scenario requires additional modeling effort that is explained in Chapter 12.14.7.1.

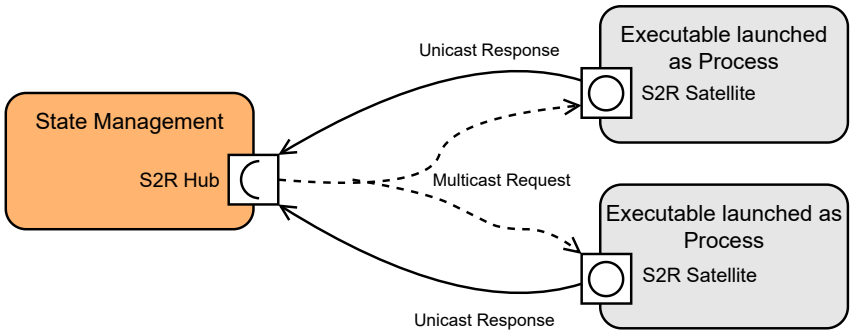


Figure 12.85: Interaction pattern for `suspend-to-RAM` functionality

12.15.1 Suspend to RAM Module Instantiation

On the *AUTOSAR adaptive platform*, support for `suspend-to-RAM` functionality is implemented by state management and the application software, as sketched in Figure 12.85.

The existence of the `SuspendToRamModuleInstantiation` is mainly bound to the necessity to provide identity and access management in the context of `suspend-to-RAM`.

Class	**SuspendToRamModuleInstantiation**			
Note	This meta-class represents the ability to define the target-configuration of a suspend-to-RAM module instantiation. **Tags:** atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 12.145: SuspendToRamModuleInstantiation

The connection between a `SuspendToRamModuleInstantiation` and `PortPrototype`(s) in the application software is created by two different mapping classes:

- `SuspendToRamSatelliteMapping` associates the `SuspendToRamModuleInstantiation` to a `PPortPrototype` that is typed by a `SuspendToRamSatelliteInterface`. This aspect is described in Chapter 12.15.2.
- `SuspendToRamHubMapping` associates the `SuspendToRamModuleInstantiation` to a `PPortPrototype` that is typed by a `SuspendToRamHubInterface`. This aspect is described in Chapter 12.15.3.

12.15.2 Suspend to RAM Module Satellite Mapping

[TPS_MANI_01510] Mapping of `SuspendToRamModuleInstantiation` to a `PortPrototype` in the `suspend-to-RAM` satellite role

Status: DRAFT

⌈The relation between a SuspendToRamModuleInstantiation and one PPort-Prototype in the suspend-to-RAM satellite role is created by using the Suspend-ToRamSatelliteMapping that refers to

- a SuspendToRamModuleInstantiation in the role module-Instantiation,
- a Process in the role process, and
- a PPortPrototype typed by a SuspendToRamSatelliteInterface in the role pPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the SuspendToRam-ModuleInstantiation. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype on the application layer will use at run-time.

⌋

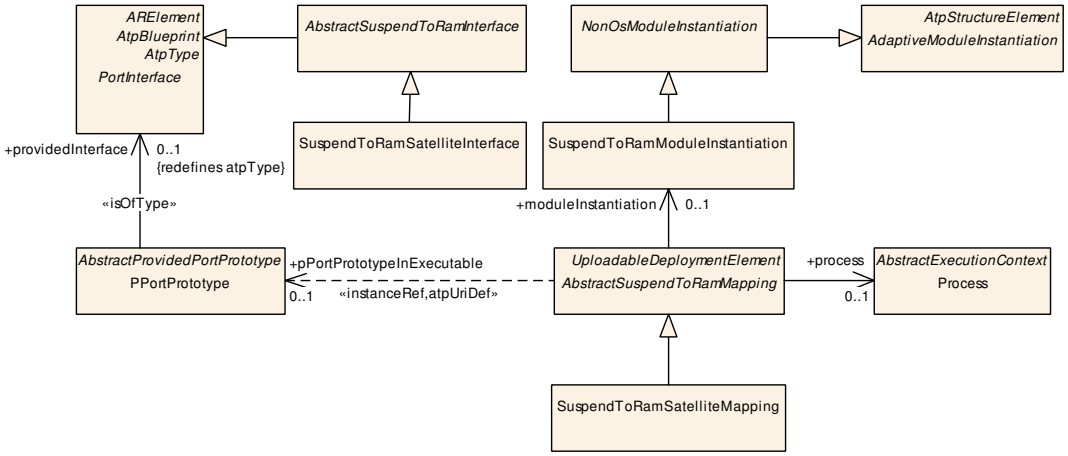


Figure 12.86: Modeling of the `SuspendToRamSatelliteMapping`

[constr_10721] Target of `AbstractSuspendToRamMapping.pPortPrototype-InExecutable`

Status: DRAFT
Imposition time: IT_Mani

⌈The target of the instanceRef in the role AbstractSuspendToRamMapping.pPortPrototypeInExecutable shall only be a PPortPrototype that is typed by a SuspendToRamSatelliteInterface.⌋

[constr_10722] Restriction for PPortPrototype that is typed by a SuspendToRamSatelliteInterface

Status: DRAFT
Imposition time: IT_Mani

⌈A PPortPrototype that is typed by a SuspendToRamSatelliteInterface shall only be referenced by a SuspendToRamSatelliteMapping in the role pPortPrototypeInExecutable.⌋

Class	**SuspendToRamSatelliteMapping**			
Note	This mapping associates a suspend-to-RAM satellite with the applicable module instantiation. **Tags:** atp.Status=candidate atp.recommendedPackage=SuspendToRamMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractSuspendToRamMapping*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *Uploadable PackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 12.146: SuspendToRamSatelliteMapping

Class	***AbstractSuspendToRamMapping*** (abstract)			
Note	This meta-class acts as an abstract base class for suspend-to-RAM-related mappings. **Tags:** atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	SuspendToRamHubMapping, SuspendToRamSatelliteMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
module Instantiation	SuspendToRamModule Instantiation	0..1	ref	This reference identifies the applicable suspend-to-RAM module instantiation. **Tags:** atp.Splitkey=moduleInstantiation atp.Status=candidate
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This reference identifies the suspend-to-RAM-related port. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=pPortPrototypeInExecutable, pPortPrototype InExecutable.contextRootSwComponentPrototype, pPort PrototypeInExecutable.contextComponentPrototype, p PortPrototypeInExecutable.targetPPortPrototype atp.Status=candidate **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef

▽

△

Class	***AbstractSuspendToRamMapping*** (abstract)			
process	Process	0..1	ref	This reference identifies the Process in which the Executable referenced in the role pPortPrototypeIn Executable is executed. **Tags:** atp.Splitkey=process atp.Status=candidate

Table 12.147: AbstractSuspendToRamMapping

[constr_10723] Existence of reference SuspendToRamSatelliteMapping.moduleInstantiation

Status: DRAFT
Imposition time: IT_Mani

⌈For each concrete sub-class of SuspendToRamSatelliteMapping, reference moduleInstantiation shall exist.⌋

[constr_10724] Existence of reference SuspendToRamSatelliteMapping.pPortPrototypeInExecutable

Status: DRAFT
Imposition time: IT_Mani

⌈For each concrete sub-class of SuspendToRamSatelliteMapping, reference pPortPrototypeInExecutable shall exist.⌋

[constr_10725] Existence of reference SuspendToRamSatelliteMapping.process

Status: DRAFT
Imposition time: IT_Mani

⌈For each concrete sub-class of SuspendToRamSatelliteMapping, reference process shall exist.⌋

12.15.3 Suspend to RAM Module Hub Mapping

[TPS_MANI_01512] Mapping of SuspendToRamModuleInstantiation to a PortPrototype in the suspend-to-RAM hub role

Status: DRAFT

⌈The relation between a SuspendToRamModuleInstantiation and one PPortPrototype in the suspend-to-RAM hub role is created by using the SuspendToRamHubMapping that refers to

- a SuspendToRamModuleInstantiation as moduleInstantiation,
- a Process in the role process, and

- a PPortPrototype typed by a SuspendToRamHubInterface in the role pPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the SuspendToRamModuleInstantiation. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype on the application layer will use at run-time.

⌋

[constr_10726] Target of AbstractSuspendToRamMapping.pPortPrototypeInExecutable

Status: DRAFT
Imposition time: IT_Mani

⌈The target of the instanceRef in the role AbstractSuspendToRamMapping.pPortPrototypeInExecutable shall only be a PPortPrototype that is typed by a SuspendToRamHubInterface.⌋

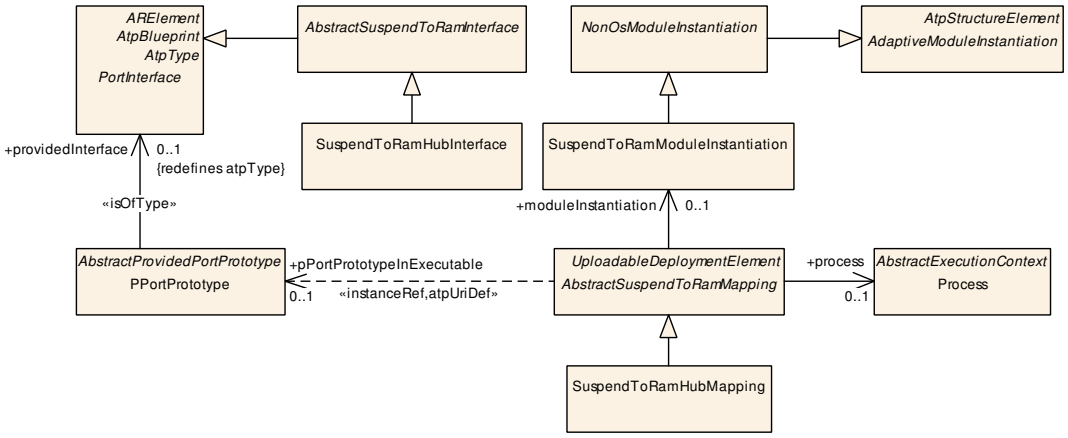


Figure 12.87: Modeling of the SuspendToRamHubMapping

[constr_10727] Restriction for PPortPrototype that is typed by a SuspendToRamHubInterface

Status: DRAFT
Imposition time: IT_Mani

⌈A PPortPrototype that is typed by a SuspendToRamHubInterface shall only be referenced by a SuspendToRamHubMapping in the role pPortPrototypeInExecutable.⌋

[constr_10728] Restriction for the reference in the role SuspendToRamHubMapping.moduleInstantiation

Status: DRAFT
Imposition time: IT_Mani

⌈Each SuspendToRamModuleInstantiation shall only be referenced from at most one SuspendToRamHubMapping in the role moduleInstantiation.⌋

Class	**SuspendToRamHubMapping**			
Note	This mapping associates a suspend-to-RAM hub with the applicable module instantiation. **Tags:** atp.Status=candidate atp.recommendedPackage=SuspendToRamMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractSuspendToRamMapping*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *Uploadable PackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 12.148: SuspendToRamHubMapping

[constr_10729] Existence of reference SuspendToRamHubMapping.moduleInstantiation

Status: DRAFT
Imposition time: IT_Mani

⌈For each concrete sub-class of SuspendToRamHubMapping, reference moduleInstantiation shall exist.⌋

[constr_10730] Existence of reference SuspendToRamHubMapping.pPortPrototypeInExecutable

Status: DRAFT
Imposition time: IT_Mani

⌈For each concrete sub-class of SuspendToRamHubMapping, reference pPortPrototypeInExecutable shall exist.⌋

[constr_10731] Existence of reference SuspendToRamHubMapping.process

Status: DRAFT
Imposition time: IT_Mani

⌈For each concrete sub-class of SuspendToRamHubMapping, reference process shall exist.⌋

12.16 Generic Diagnostic TransportInstantiation

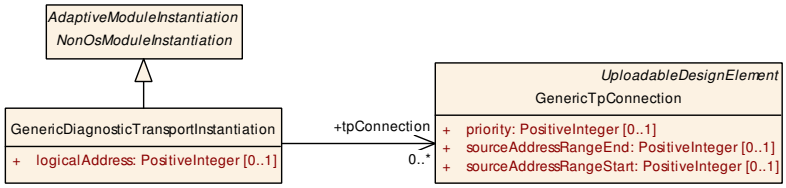


Figure 12.88: Modeling of the GenericDiagnosticTransportInstantiation

[TPS_MANI_03335] Semantics of meta-class GenericDiagnosticTransportInstantiation ⌈Meta-class GenericDiagnosticTransportInstantiation

represents the ability to configure a generic diagnostic transport protocol (other than DoIP) used to define the communication of diagnostic testers with the vehicle.⌋

The configuration of DoIp is described in Chapter 12.5.

Class	**GenericDiagnosticTransportInstantiation**			
Note	This meta-class defines a generic transport protocol (other than DoIP) for communication of diagnostic testers with a vehicle. Please note that the necessary configuration attributes can be defined via Special Data Groups. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
logicalAddress	PositiveInteger	0..1	attr	Contains the logical address of the receiver of a diagnostic message
tpConnection	GenericTpConnection	*	ref	Reference to a TpConnection that identifies the receiver(s) of a particular communication **Tags:** atp.Splitkey=tpConnection

Table 12.149: GenericDiagnosticTransportInstantiation

13 Service Instance Manifest

13.1 ApplicationEndpoint

[TPS_MANI_03280] Semantics of meta-class ApApplicationEndpoint ⌈The meta-class ApApplicationEndpoint defines a UDP or TCP Port on a EthernetCommunicationConnector and defines together with the NetworkEndpoint that is referenced by the same EthernetCommunicationConnector in the role unicastNetworkEndpoint a local endpoint of a network communication path, i.e. a Socket.⌋

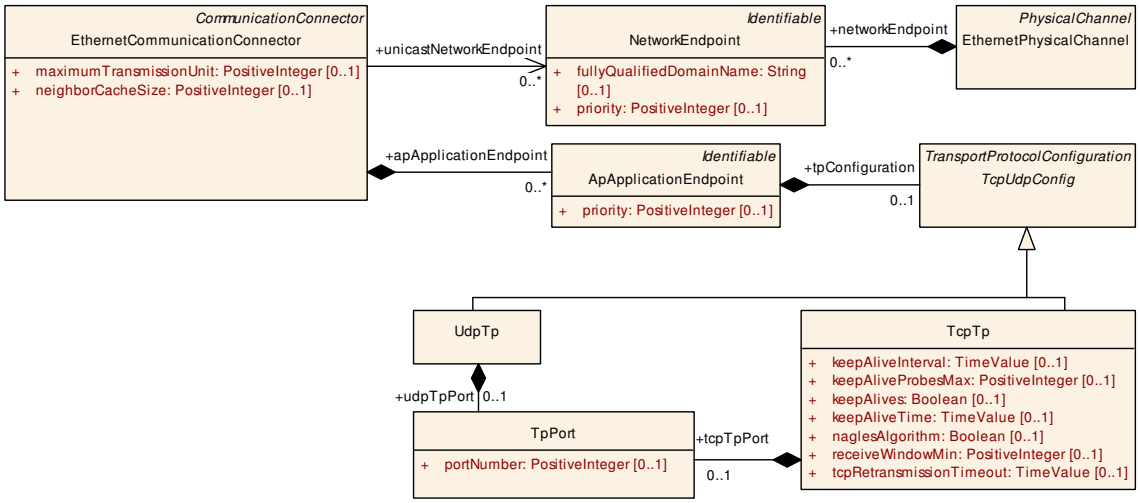


Figure 13.1: ApApplicationEndpoint definition

For the usable value range of portNumber, restrictions may be imposed on the members of an EthernetCluster. If all the participants of the EthernetCluster are fully statically configured with respect to their used portNumbers, no need for reserved portNumber ranges may exist.

But with the introduction of the AUTOSAR adaptive platform, more dynamic communication use-cases and protocols have been introduced. Some portNumber ranges might have a dedicated usage in these scenarios.

An operating system implementation might utilize a certain set of portNumbers, assuming these portNumbers are free to use in an ephemeral way. If now an AUTOSAR Service defines a Service Instance to use a portNumber that has already been allocated by the operating system (maybe already during boot), then the AUTOSAR Service Instance will not be able to be instantiated.

To avoid clashing portNumbers, special care has to be put into defining the usable portNumber ranges for AUTOSAR communication. The source of the restrictions on the usage of portNumbers is [40, IETF RFC 6335], especially section 8.1.2 “Variances for Specific Port Number Ranges”.

See also [advisory_02005] in [4, TPS System Template].

Class	**ApApplicationEndpoint**			
Note	An application endpoint is the endpoint on an Ecu in terms of application addressing (e.g. UDP or TCP Port). This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EthernetCommunicationConnector.apApplicationEndpoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
priority	PositiveInteger	0..1	attr	This attribute defines the VLAN frame priority where values from 0 (best effort) to 7 (highest) are allowed.
tpConfiguration	TcpUdpConfig	0..1	aggr	Configuration of the used transport protocol. **Tags:** atp.Splitkey=tpConfiguration

Table 13.1: ApApplicationEndpoint

Class	***TcpUdpConfig*** (abstract)			
Note	Tcp or Udp Transport Protocol Configuration.			
Base	*ARObject*, *TransportProtocolConfiguration*			
Subclasses	TcpTp, UdpTp			
Aggregated by	ApApplicationEndpoint.tpConfiguration, ApplicationEndpoint.tpConfiguration			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 13.2: TcpUdpConfig

Class	**UdpTp**			
Note	Content Model for UDP configuration.			
Base	*ARObject*, *TcpUdpConfig*, *TransportProtocolConfiguration*			
Aggregated by	ApApplicationEndpoint.tpConfiguration, ApplicationEndpoint.tpConfiguration			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
udpTpPort	TpPort	0..1	aggr	Udp Port configuration. **Tags:** atp.Splitkey=udpTpPort

Table 13.3: UdpTp

Class	**TcpTp**			
Note	Content Model for TCP configuration.			
Base	*ARObject*, *TcpUdpConfig*, *TransportProtocolConfiguration*			
Aggregated by	ApApplicationEndpoint.tpConfiguration, ApplicationEndpoint.tpConfiguration			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
keepAlive Interval	TimeValue	0..1	attr	Specifies the interval in seconds between subsequent keepalive probes.
keepAlive ProbesMax	PositiveInteger	0..1	attr	Maximum number of times that TCP retransmits an individual data segment before aborting the connection.
keepAlives	Boolean	0..1	attr	Indicates if Keep-Alive messages are sent.
keepAliveTime	TimeValue	0..1	attr	Specifies the time in seconds between the last data packet sent and the first keepalive probe.
naglesAlgorithm	Boolean	0..1	attr	Indicates if Nagle's Algorithm is used.
receiveWindow Min	PositiveInteger	0..1	attr	Minimum size of the TCP receive window in bytes.



△

Class	TcpTp			
tcp Retransmission Timeout	TimeValue	0..1	attr	Defines the timeout in seconds before an unacknowledged TCP segment is sent again. If the tcp RetransmissionTimeout is not defined or set to "INF", no TCP segments shall be re-transmitted.
tcpTpPort	TpPort	0..1	aggr	TCP Port configuration. **Tags:** atp.Splitkey=tcpTpPort

Table 13.4: TcpTp

[TPS_MANI_03281] Port specific TCP configuration settings ⌈The configuration settings in the `TcpTp` element of an `ApApplicationEndpoint` overwrite potential TCP settings that are available in `TcpProps` referenced by the `MachineDesign` that aggregates the `EthernetCommunicationConnector` that in turn contains the `ApApplicationEndpoint`.⌋

Class	TpPort			
Note	Dynamic or direct assignment of a PortNumber.			
Base	*ARObject*			
Aggregated by	TcpTp.tcpTpPort, UdpTp.udpTpPort			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
portNumber	PositiveInteger	0..1	attr	Port Number.

Table 13.5: TpPort

13.1.1 VLAN Priority

The `priority` is a 3-bit field which refers to the IEEE 802.1Q priority. It indicates the frame priority level. Values are from 0 (best effort) to 7 (highest). These values can be used to prioritize different classes of traffic (voice, video, data, etc.). The priority is contained in the Ethernet Header together with the `vlanIdentifier`.

The priority can be defined on different levels and can overwrite the `defaultPriority` in the context of a `CouplingPort` on the VLAN (higher numbers in the list indicate higher overwriting priority) `CouplingPort`:

1. `NetworkEndpoint` (for network traffic over the IP Address)
2. `ApApplicationEndpoint` (for TCP Port and/or UDP Port network traffic)
3. `ProvidedSomeipServiceInstance` (for network traffic resulting from the Service Instance)

An example of the definition of VLAN priority is shown in Chapter B.2.5.

The definition of the `priority` in the `SomeipSdServerServiceInstanceConfig` can be used to define priorities for SD messages like `Service Offer` and

SubscribeEventGroupAck that are resulting from ProvidedSomeipServiceInstances that are referencing the SomeipSdServerServiceInstanceConfig in the role sdServerConfig.

To prioritize SD messages that are sent from the service consumer to the service provider like FindService or SubscribeEventGroup, the priority in the SomeipSdClientServiceInstanceConfig can be used.

The SD messages are communicated over a UDP Port that is defined in SomeipServiceDiscovery.someipServiceDiscoveryPort and over an IP Multicast Address that is defined by SomeipServiceDiscovery.multicastSdIpAddress. In other words, the priority in the SomeipSdServerServiceInstanceConfig and SomeipSdClientServiceInstanceConfig can be used to prioritize network traffic that results from specific ServiceInstances on SD level.

13.2 Service Interface Deployment

The different meta-class specializations of ServiceInterfaceDeployment define a binding of a ServiceInterface to a middleware transport layer.

This chapter describes the usage of the ServiceInterfaceDeployment in different bindings that are supported by AUTOSAR.

[TPS_MANI_03036] ServiceInterface deployment to a middleware transport layer ⌈The ServiceInterfaceDeployment meta-class provides the ability to map a ServiceInterface to a middleware transport layer that is represented by a concrete class that is derived from the abstract ServiceInterfaceDeployment meta-class.⌋

The association between the ServiceInterfaceDeployment and the ServiceInterface implicitly also defines the relation between the technology specific version number of the service on the ServiceInterfaceDeployment and the service version of the ServiceInterface (defined by ServiceInterface.majorVersion and ServiceInterface.minorVersion)

[TPS_MANI_03617] Version mapping between ServiceInterface and ServiceInterfaceDeployment ⌈The contract version of a ServiceInterface (majorVersion, minorVersion) shall be mapped to a version of the ServiceInterfaceDeployment for each transport layer.

This version mapping may lead to different version numbers for different ServiceInterfaceDeployments that refer to the same ServiceInterface. This allows defining different version numbers, on the same network or on different networks (e.g. VLANs).⌋

Transport layer specific constraints on the uniqueness of protocol credentials still have to be respected, e.g. [constr_1723].

In Figure 13.3, the use-case of having one `PPortPrototype` typed by one `ServiceInterface` and having several `ProvidedSomeipServiceInstance`s on several VLANs is illustrated.

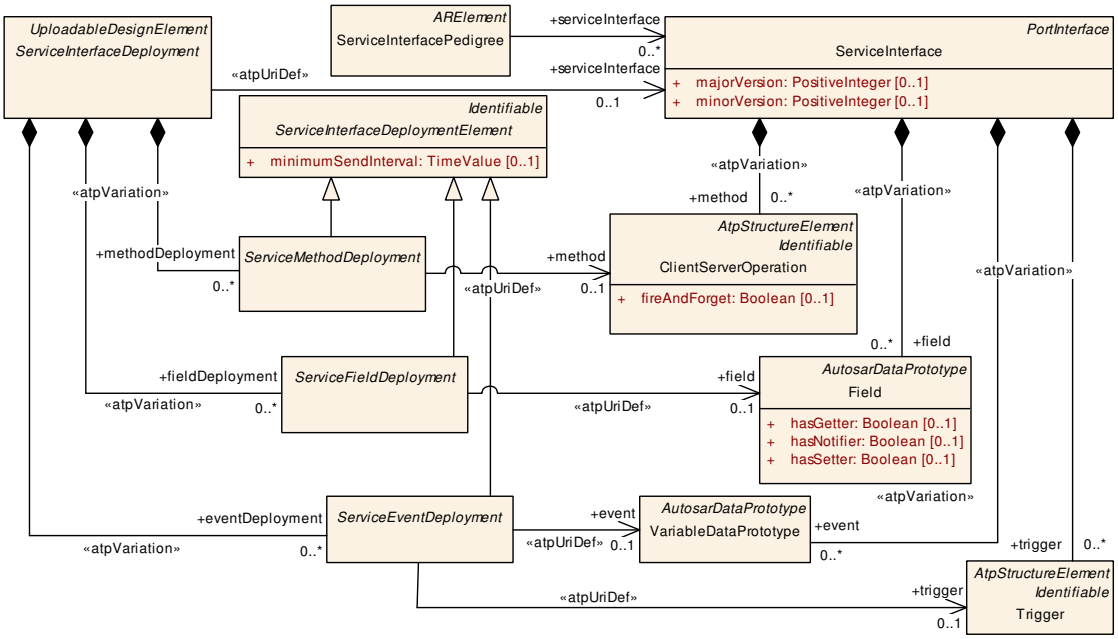


Figure 13.2: Deployment-related modeling of ServiceInterface

It is possible to have different `serviceInterfaceId`, `serviceInstanceId`, and `majorVersion` values on each individual VLAN.

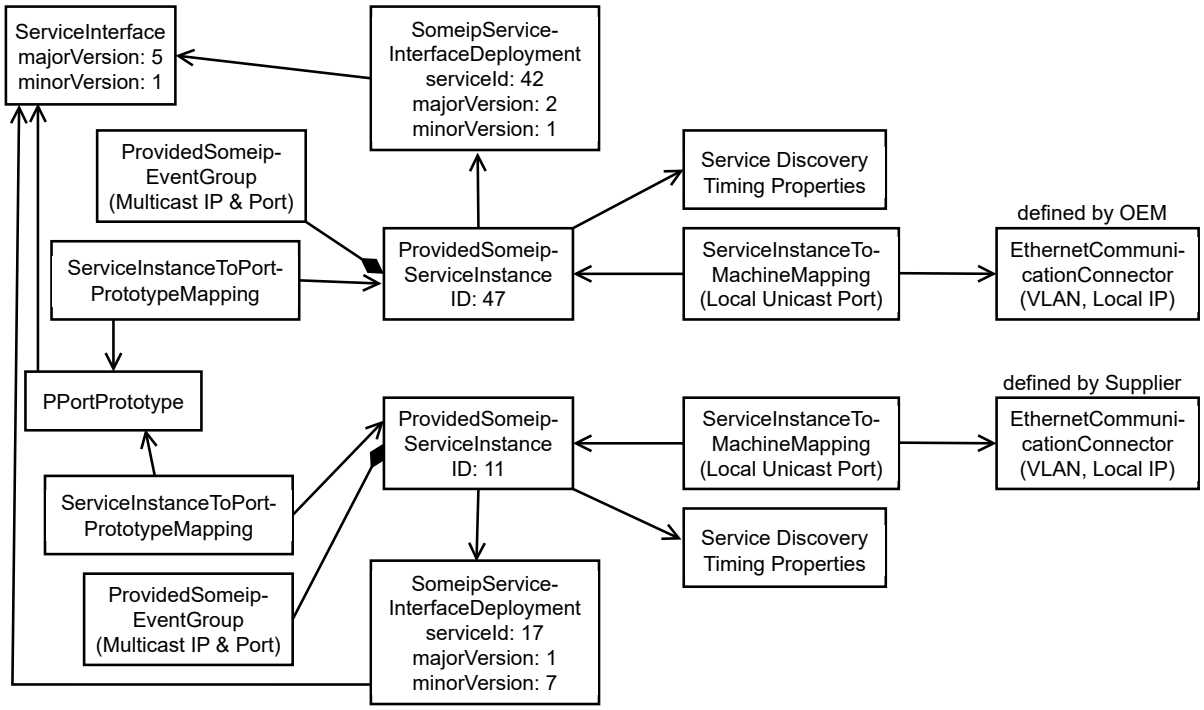


Figure 13.3: Example of 1 `PPortPrototype` mapped to 2 VLANs

But also the use-case of providing both service instances on the same VLAN would be supported, as long as their SOME/IP credentials are unique (see [constr_1723]).

On different VLANs, the service instance may also use the same SOME/IP credentials. Only if all service instance specific configuration settings (e.g. IP Multicast Address, SD Timing Properties) are exactly the same on both VLANs, the same `AdaptivePlatformServiceInstance` can be used to model the communication on both VLANs.

If, for example, the `serviceInterfaceId`, `serviceInstanceId`, and `majorVersion` values of a service instance on each VLAN are the same, but different IP Multicast Addresses are used on the different VLANs for the EventGroup transmission, then an individual VLAN specific modeling of `AdaptivePlatformServiceInstance` is necessary as shown in Figure 13.3.

Class	***ServiceInterfaceDeployment*** (abstract)			
Note	Middleware transport layer specific configuration settings for the ServiceInterface and all contained ServiceInterface elements. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Subclasses	DdsServiceInterfaceDeployment, SomeipServiceInterfaceDeployment, UserDefinedServiceInterface Deployment			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
event Deployment	ServiceEvent Deployment	*	aggr	Middleware transport layer specific configuration settings for an Event that is defined in the ServiceInterface. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=eventDeployment, eventDeployment.shortName, eventDeployment.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime
fieldDeployment	ServiceField Deployment	*	aggr	Middleware transport layer specific configuration settings for a Field that is defined in the ServiceInterface. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=fieldDeployment, fieldDeployment.shortName, fieldDeployment.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime
method Deployment	ServiceMethod Deployment	*	aggr	Middleware transport layer specific configuration settings for a method that is defined in the ServiceInterface. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=methodDeployment, methodDeployment.shortName, methodDeployment.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime
serviceInterface	ServiceInterface	0..1	ref	Reference to a ServiceInterface that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=serviceInterface

Table 13.6: ServiceInterfaceDeployment

Class	**ServiceInterfaceDeploymentElement** (abstract)			
Note	This meta-class provides shared attributes for concrete classes used in the context of the definition of the ServiceInterfaceDeployment. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*ServiceEventDeployment*, *ServiceFieldDeployment*, *ServiceMethodDeployment*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
minimumSend Interval	TimeValue	0..1	attr	Minimum time between two consecutive transmissions of a service instance element (event, field, trigger or method call). Unit: seconds

Table 13.7: ServiceInterfaceDeploymentElement

[constr_10655] Allowed range of values for attribute `ServiceInterfaceDeploymentElement.minimumSendInterval`

Imposition time: `IT_Mani`

⌈The allowed value of attribute `ServiceInterfaceDeploymentElement.minimumSendInterval` shall be within the interval [0.0 .. 10.0].⌋

A value of 0 for `ServiceInterfaceDeploymentElement.minimumSendInterval` means that **no minimum transmission time** for the corresponding `event`, `field`, `trigger` or `method` shall be enforced, i.e., the tx frequency is **not** limited by ara::com.

[TPS_MANI_03037] Purpose of `ServiceMethodDeployment` ⌈The `ServiceMethodDeployment` meta-class provides the ability to define middleware transport layer specific configuration settings relevant for a `method` that is defined in the context of a `ServiceInterface`.⌋

[constr_3300] Allowed `ServiceMethodDeployment.method` references

Imposition time: `IT_Mani`

⌈The `ClientServerOperation` that is referenced by `ServiceMethodDeployment` in the role `method` shall be defined in the context of a `ServiceInterface` that is referenced by the `ServiceInterfaceDeployment` in the role `serviceInterface` that contains the `ServiceMethodDeployment`.⌋

[TPS_MANI_03038] Purpose of `ServiceEventDeployment` ⌈The `ServiceEventDeployment` meta-class provides the ability to define middleware transport layer specific configuration settings relevant for an `event` or `trigger` that is defined in the context of a `ServiceInterface`.⌋

[constr_3301] Allowed `ServiceEventDeployment.event` references

Imposition time: `IT_Mani`

⌈The `VariableDataPrototype` that is referenced by `ServiceEventDeployment` in the role `event` shall be defined in the context of a `ServiceInterface` that is referenced by the `ServiceInterfaceDeployment` in the role `serviceInterface` that contains the `ServiceEventDeployment`.⌋

[constr_5316] Allowed `ServiceEventDeployment.trigger` references

Imposition time: IT_Mani

⌈The Trigger that is referenced by ServiceEventDeployment in the role trigger shall be defined in the context of a ServiceInterface that is referenced by the ServiceInterfaceDeployment in the role serviceInterface that contains the ServiceEventDeployment.⌋

[constr_5317] `ServiceEventDeployment` not allowed to reference an `event` and a `trigger` at the same time

Imposition time: IT_Mani

⌈The ServiceEventDeployment element shall reference either:

- a VariableDataPrototype in the role event or
- a Trigger in the role trigger,

but not both at the same time.⌋

[TPS_MANI_03039] Purpose of `ServiceFieldDeployment` ⌈The ServiceFieldDeployment meta-class provides the ability to define middleware transport layer specific configuration settings relevant for a field that is defined in the context of a ServiceInterface.⌋

[constr_3302] Allowed `ServiceFieldDeployment.field` references

Imposition time: IT_Mani

⌈The Field that is referenced by ServiceFieldDeployment in the role field shall be defined in the context of a ServiceInterface that is referenced by the ServiceInterfaceDeployment in the role serviceInterface that contains the ServiceFieldDeployment.⌋

A partial ServiceInterfaceDeployment that only covers certain parts of the corresponding ServiceInterfaceDeployment is not supported. This understanding is formalized in [constr_10029].

The coverage of elements of a ServiceInterfaceDeployment varies depending on the sub-class of ServiceInterfaceDeployment. This distinction is formalized in the existence of [constr_10029] and [constr_10629].

[constr_10029] `ServiceInterfaceDeployment` (except `DdsServiceInterfaceDeployment`) shall cover all elements of the corresponding `ServiceInterface`

Imposition time: IT_Mani

⌈With the exception of DdsServiceInterfaceDeployment, if a ServiceInterfaceDeployment references a ServiceInterface in the role serviceInterface, then all methods, fields, triggers, and events defined in the context

of the referenced ServiceInterface shall be referenced by respective methodDeployments, fieldDeployments, and eventDeployments owned by the referencing ServiceInterfaceDeployment.⌋

[constr_3715] Reference in the role SomeipEventGroup.event

Imposition time: IT_Mani

⌈In the context of a given SomeipServiceInterfaceDeployment, all aggregated SomeipEventDeployments shall be referenced at least once in the role event by SomeipEventGroups that in turn are aggregated at the same SomeipServiceInterfaceDeployment.⌋

Class	***ServiceMethodDeployment*** (abstract)			
Note	This abstract meta-class represents the ability to specify a deployment of a Method to a middleware transport layer. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceInterfaceDeploymentElement*			
Subclasses	SomeipMethodDeployment, UserDefinedMethodDeployment			
Aggregated by	*ServiceInterfaceDeployment*.methodDeployment			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
method	ClientServerOperation	0..1	ref	Reference to a method that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=method

Table 13.8: ServiceMethodDeployment

Class	***ServiceEventDeployment*** (abstract)			
Note	This abstract meta-class represents the ability to specify a deployment of an Event to a middleware transport layer. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceInterfaceDeploymentElement*			
Subclasses	DdsEventDeployment, SomeipEventDeployment, UserDefinedEventDeployment			
Aggregated by	*ServiceInterfaceDeployment*.eventDeployment			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
event	VariableDataPrototype	0..1	ref	Reference to an Event that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=event
trigger	Trigger	0..1	ref	Reference to a Trigger that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=trigger

Table 13.9: ServiceEventDeployment

Class	**ServiceFieldDeployment** (abstract)			
Note	This abstract meta-class represents the ability to specify a deployment of a Field to a middleware transport layer. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceInterfaceDeploymentElement*			
Subclasses	DdsFieldDeployment, SomeipFieldDeployment, UserDefinedFieldDeployment			
Aggregated by	*ServiceInterfaceDeployment*.fieldDeployment			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
field	Field	0..1	ref	Reference to a Field that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=field

Table 13.10: ServiceFieldDeployment

[constr_10251] Multiplicity of the reference in the role `ServiceFieldDeployment.field`

Imposition time: `IT_Mani`

⌈For each `ServiceFieldDeployment`, the reference in the role `field` shall exist.⌋

13.2.1 SOME/IP Service Interface Deployment

This chapter describes the SOME/IP deployment of a `ServiceInterface`.

[TPS_MANI_03040] SOME/IP ServiceInterface binding ⌈The `SomeipServiceInterfaceDeployment` meta-class provides the ability to bind a `ServiceInterface` to SOME/IP and to assign a SOME/IP Service identifier to the `ServiceInterface` with the `serviceInterfaceId` attribute.⌋

The idea behind the `SomeipServiceInterfaceDeployment` is the definition of a common configuration set that is shared between the server that provides the `ServiceInterface` and all clients that are consuming the `ServiceInterface`. So it contains all relevant SOME/IP settings used for identification of the `ServiceInterface` and its content in messages on the network.

[constr_3410] Value range of `SomeipServiceInterfaceDeployment.serviceInterfaceId`

Imposition time: `IT_Mani`

⌈The value of `serviceInterfaceId` shall be in the range of 0..65535.⌋

The SOME/IP MessageId is 32 bits long and contains a 16-bit `serviceInterfaceId`, a single bit that defines whether the message transports a method or an event and a 15-bit `eventId` or `methodId`.

See also [PRS_SOMEIPSD_00515] in [41, PRS SOME/IP Service Discovery Protocol] that defines special and reserved `serviceInterfaceId`s for SOME/IP and SOME/IP-SD.

[TPS_MANI_03041] Definition of SOME/IP EventGroups ⌈The SomeipService-InterfaceDeployment.eventGroup allows defining SOME/IP *EventGroups* that are included in the SOME/IP Service and provide a logical grouping of events and notification events used for publish/subscribe handling.⌋

[constr_3304] Value of attribute SomeipEventGroup.eventGroupId shall be unique

Imposition time: IT_Mani

⌈The value of attribute eventGroupId shall be unique in the context of the enclosing SomeipServiceInterfaceDeployment.⌋

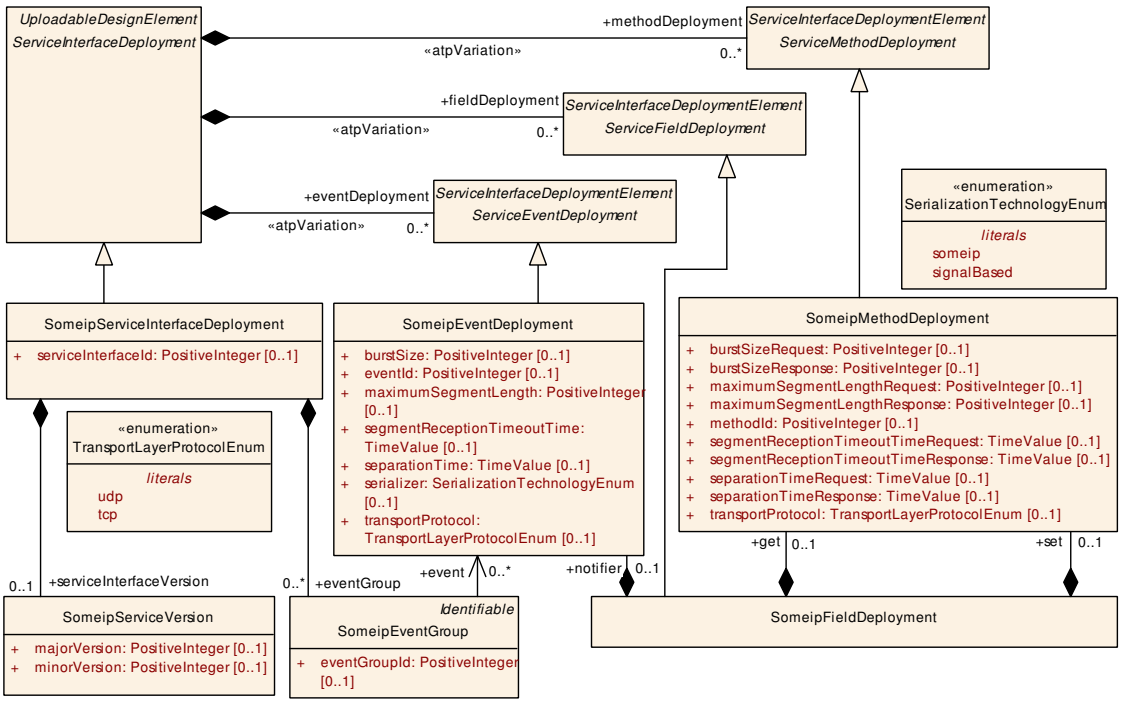


Figure 13.4: SOME/IP deployment of ServiceInterface

[TPS_MANI_03042] Definition of SOME/IP Service Version ⌈The Someip-ServiceInterfaceDeployment.serviceInterfaceVersion allows defining a major and a minor version for the SOME/IP Service.⌋

[constr_3557] Mandatory majorVersion at SomeipServiceInterface-Deployment.serviceInterfaceVersion

Imposition time: IT_Mani

⌈If the SomeipServiceVersion is aggregated at the SomeipServiceInterface-Deployment in the role serviceInterfaceVersion then the attribute Someip-ServiceVersion.majorVersion shall be defined.⌋

Class	**SomeipServiceInterfaceDeployment**			
Note	SOME/IP configuration settings for a ServiceInterface. **Tags:** atp.recommendedPackage=ServiceInterfaceDeployments This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *ServiceInterfaceDeployment*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventGroup	SomeipEventGroup	*	aggr	SOME/IP EventGroups that are defined within the SOME/IP ServiceClass. **Tags:** atp.Splitkey=eventGroup, eventGroup.shortName
serviceInterfaceId	PositiveInteger	0..1	attr	Unique Identifier that identifies the ServiceInterface in SOME/IP. This Identifier is sent as Service ID in SOME/IP Service Discovery messages.
serviceInterfaceVersion	SomeipServiceVersion	0..1	aggr	The SOME/IP major and minor Version of the Service. **Tags:** atp.Splitkey=serviceInterfaceVersion

Table 13.11: SomeipServiceInterfaceDeployment

[constr_10212] Multiplicity of attribute SomeipServiceInterfaceDeployment.serviceInterfaceId

Imposition time: IT_Mani

⌈For each SomeipServiceInterfaceDeployment, the attribute serviceInterfaceId shall exist.⌋

[constr_10213] Multiplicity of attribute SomeipServiceInterfaceDeployment.serviceInterfaceVersion

Imposition time: IT_Mani

⌈For each SomeipServiceInterfaceDeployment, the attribute serviceInterfaceVersion.⌋

The value of each SomeipEventDeployment.eventId that is used in the context of an enclosing SomeipServiceInterfaceDeployment is allowed to overlap with the value of each SomeipEventGroup.eventGroupId used in the same SomeipServiceInterfaceDeployment.

In other words, the range of valid numbers of SomeipEventDeployment.eventId is independent of the range of valid numbers used for SomeipEventGroup.eventGroupId and both attributes are allowed to use the same value in the context of the same SomeipServiceInterfaceDeployment.

For service-signal-translation according to [TPS_MANI_03667] special semantics applies as defined in [TPS_MANI_03666].

[TPS_MANI_03043] SOME/IP VariableDataPrototype binding ⌈The SomeipEventDeployment meta-class provides the ability to bind a VariableDataPrototype to SOME/IP and to assign a SOME/IP Event identifier to the event with the eventId attribute.⌋

Class	SomeipEventGroup			
Note	Grouping of events and notification events inside a ServiceInterface in order to allow subscriptions. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SomeipServiceInterfaceDeployment.eventGroup			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
event	SomeipEvent Deployment	*	ref	Reference to an event that is part of the EventGroup. **Tags:** atp.Splitkey=event
eventGroupId	PositiveInteger	0..1	attr	Unique Identifier that identifies the EventGroup in SOME/IP. This Identifier is sent as Eventgroup ID in SOME/IP Service Discovery messages.

Table 13.12: SomeipEventGroup

[constr_10214] Multiplicity of attribute SomeipEventGroup.eventGroupId

Imposition time: IT_Mani

⌈For each SomeipEventGroup, the attribute eventGroupId shall exist.⌋

[constr_3305] Value of attribute SomeipEventDeployment.eventId shall be unique

Imposition time: IT_Mani

⌈The value of eventId shall be unique in the context of the enclosing SomeipServiceInterfaceDeployment, unless SomeipEventDeployment.serializer is set to SerializationTechnologyEnum.signalBased.⌋

[constr_3408] Value range of SomeipEventDeployment.eventId

Imposition time: IT_Mani

⌈The value of eventId shall be in the range of 0..32767.⌋

The specification item [PRS_SOMEIPSD_00517] in [41, PRS SOME/IP Service Discovery Protocol] defines special and reserved EVENT-IDs for SOME/IP and SOME/IP-SD that result in the eventId values of 0 and 32767.

[TPS_MANI_03050] Usage of SomeipEventDeployment.transportProtocol

⌈The value of SomeipEventDeployment.transportProtocol defines over which Transport Layer Protocol the SomeipEventDeployment.event is provided.⌋

[constr_5156] SomeipEventDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances

Imposition time: IT_Mani

⌈If SomeipEventDeployment.transportProtocol is set to udp then each ProvidedSomeipServiceInstance that refers the SomeipServiceInterfaceDeployment in the role serviceInterfaceDeployment shall only be mapped to a MachineDesign with a SomeipServiceInstanceToMachineMapping with a configured udpPort.⌋

[constr_3308] SomeipEventDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances

Imposition time: IT_Mani

⌈If SomeipEventDeployment.transportProtocol is set to tcp then each ProvidedSomeipServiceInstance that refers the SomeipServiceInterfaceDeployment in the role serviceInterfaceDeployment shall only be mapped to a MachineDesign with a SomeipServiceInstanceToMachineMapping with a configured tcpPort.⌋

[TPS_MANI_03067] SOME/IP segmentation of udp SomeipEventDeployments
⌈If the maximumSegmentLength is set to a value and the data length is larger than maximumSegmentLength then SOME/IP shall segment the SomeipEventDeployment into several packets and transmit them over the network.

The sender shall wait for the separationTime between the transmissions of segments. On the reception side, SOME/IP re-assembles the received SOME/IP segments to the original SomeipEventDeployment.⌋

[constr_3351] SOME/IP segmentation is allowed for udp SomeipEventDeployments

Imposition time: IT_Mani

⌈Attribute SomeipEventDeployment.maximumSegmentLength shall only be used if the value of attribute SomeipEventDeployment.transportProtocol is set to udp.⌋

[TPS_MANI_03328] Reception Monitoring of SOME/IP segments ⌈The attribute SomeipEventDeployment.segmentReceptionTimeoutTime is used to monitor the successful reception of SOME/IP segments (conforms to SOMEIPTP_REASSEMBLY_TIMEOUT of [11, PRS SOME/IP Protocol]).⌋

[constr_9327] SOME/IP segment reception monitoring allowed for udp SomeipEventDeployments

Imposition time: IT_Mani

⌈Attribute SomeipEventDeployment.segmentReceptionTimeoutTime shall only be used if the value of attribute SomeipEventDeployment.transportProtocol is set to udp.⌋

As the SomeipServiceInterfaceDeployment is also used for the deployment of signal-based Pdus on Ethernet the attribute SomeipEventDeployment.serializer defines whether the someip or the signalBased serialization shall be used for a specific event.

[TPS_MANI_03615] SomeipEventDeployment.serializer equals someip ⌈If the attribute SomeipEventDeployment.serializer is not defined or is set to the value someip then the event shall be serialized/de-serialized using the SOME/IP serializer.⌋

[TPS_MANI_03591] **SomeipEventDeployment.serializer equals signal-Based** ⌈If the attribute SomeipEventDeployment.serializer is set to the value signalBased then the event shall be serialized/de-serialized using the signal-based approach.⌋

This aspect is described in Chapter 9.

Class	**SomeipEventDeployment**			
Note	SOME/IP configuration settings for an Event. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceEventDeployment*, *ServiceInterface DeploymentElement*			
Aggregated by	*ServiceInterfaceDeployment*.eventDeployment, SomeipFieldDeployment.notifier			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
burstSize	PositiveInteger	0..1	attr	Specifies the number of segments that shall be transmitted in a burst ignoring separationTime. SeparationTime will then only be applied between bursts. If not configured, SeparationTime will be applied between all frames.
eventId	PositiveInteger	0..1	attr	Unique Identifier within a ServiceInterface that identifies the Event in SOME/IP. This Identifier is sent as part of the Message ID in SOME/IP messages.
maximum SegmentLength	PositiveInteger	0..1	attr	This attribute describes the length in bytes of the SOME/IP segment. This includes 8 bytes for the Request ID, Protocol Version, Interface Version, Message Type and Return Code and 4 additional SOME/IP TP bytes. If this attribute is set to a value and the data length is larger than maximumSegmentLength then the corresponding SOME/IP message will be segmented into smaller parts that are transmitted over the network.
segment Reception TimeoutTime	TimeValue	0..1	attr	Timer to monitor the successful reception of segments (in seconds) in SOME/IP.
separationTime	TimeValue	0..1	attr	Sets the duration of the minimum time in seconds SOME/IP shall wait between the transmissions of segments.
serializer	SerializationTechnology Enum	0..1	attr	Defines which serialization technology shall be used.
transport Protocol	TransportLayerProtocol Enum	0..1	attr	This attribute defines over which Transport Layer Protocol this event is intended to be sent.

Table 13.13: SomeipEventDeployment

[constr_10215] Multiplicity of attribute SomeipEventDeployment.eventId

Imposition time: IT_Mani

⌈For each SomeipEventDeployment, the attribute eventId shall exist.⌋

[constr_10216] Multiplicity of attribute SomeipEventDeployment.transport-Protocol

Imposition time: IT_Mani

⌈For each SomeipEventDeployment, the attribute transportProtocol shall exist.⌋

Enumeration	**SerializationTechnologyEnum**
Note	This enumeration allows to choose a Serialization Technology. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	SomeipEventDeployment.serializer
Literal	***Description***
signalBased	Signal-Based serializer. **Tags:** atp.EnumerationLiteralIndex=1
someip	SOME/IP Serializer **Tags:** atp.EnumerationLiteralIndex=0

Table 13.14: SerializationTechnologyEnum

[TPS_MANI_03044] SOME/IP `ClientServerOperation` binding ⌈The `SomeipMethodDeployment` meta-class provides the ability to bind a `ClientServerOperation` to SOME/IP and to assign a SOME/IP Method identifier to the `method` with the `methodId` attribute.⌋

[constr_3306] Value of attribute `methodId` shall be unique per `SomeipServiceInterfaceDeployment`

Imposition time: `IT_Mani`

⌈The value of `methodId` shall be unique in the context of the enclosing `SomeipServiceInterfaceDeployment`.⌋

[constr_3409] Value range of `SomeipMethodDeployment.methodId`

Imposition time: `IT_Mani`

⌈The value of `methodId` shall be in the range of 0..32767.⌋

The specification item [PRS_SOMEIPSD_00517] in [41, PRS SOME/IP Service Discovery Protocol] defines special and reserved METHOD-IDs for SOME/IP and SOME/IP-SD that result in the `methodId` values of 0 and 32767.

[TPS_MANI_03051] Usage of `SomeipMethodDeployment.transportProtocol` ⌈The value of `SomeipMethodDeployment.transportProtocol` defines over which Transport Layer Protocol this method is provided.⌋

[constr_3309] `SomeipMethodDeployment.transportProtocol` setting to `udp` and the impact on `ProvidedSomeipServiceInstance`s

Imposition time: `IT_Mani`

⌈If `SomeipMethodDeployment.transportProtocol` is set to `udp` then each `ProvidedSomeipServiceInstance` that refers the `SomeipServiceInterfaceDeployment` in the role `serviceInterfaceDeployment` shall only be mapped to a `MachineDesign` with a `SomeipServiceInstanceToMachineMapping` with a configured `udpPort`.⌋

[constr_3310] SomeipMethodDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances

Imposition time: IT_Mani

⌈If SomeipMethodDeployment.transportProtocol is set to tcp then each ProvidedSomeipServiceInstance that refers the SomeipServiceInterface-Deployment in the role serviceInterfaceDeployment shall only be mapped to a MachineDesign with a SomeipServiceInstanceToMachineMapping with a configured tcpPort.⌋

[TPS_MANI_03068] SOME/IP segmentation of SomeipMethodDeployment Calls ⌈If the maximumSegmentLengthRequest is set to a value and the data length is larger than maximumSegmentLengthRequest then SOME/IP shall segment the SomeipMethodDeployment Call-Message into several packets and transmit them over the network.

The sender shall wait for the separationTimeRequest between the transmissions of segments. On the reception side, SOME/IP re-assembles the received SOME/IP segments to the original SomeipMethodDeployment Call-Message.⌋

[TPS_MANI_03069] SOME/IP segmentation of SomeipMethodDeployment Responses ⌈If the maximumSegmentLengthResponse is set to a value and the data length is larger than maximumSegmentLengthResponse then SOME/IP shall segment the SomeipMethodDeployment Response-Message into several packets and transmit them over the network.

The sender shall wait for the separationTimeResponse between the transmissions of segments. On the reception side, SOME/IP re-assembles the received SOME/IP segments to the original SomeipMethodDeployment Response-Message.⌋

[constr_3352] SOME/IP segmentation allowed for udp SomeipMethod-Deployments

Imposition time: IT_Mani

⌈SomeipMethodDeployment.maximumSegmentLengthRequest and SomeipMethodDeployment.maximumSegmentLengthResponse shall only be used if SomeipMethodDeployment.transportProtocol is set to udp.⌋

[TPS_MANI_03329] Reception Monitoring of SOME/IP segments of the Method Call Message ⌈The attribute SomeipMethodDeployment.segmentReception-TimeoutTimeRequest is used to monitor the successful reception of SOME/IP segments of the Method Call Message (conforms to SOMEIPTP_REASSEMBLY_TIMEOUT of [11, PRS SOME/IP Protocol]).⌋

[TPS_MANI_03330] Reception Monitoring of SOME/IP segments of the Method Response Message ⌈The attribute SomeipMethodDeployment.segment-ReceptionTimeoutTimeResponse is used to monitor the successful reception of SOME/IP segments of the Method Response Message (conforms to SOMEIPTP_REASSEMBLY_TIMEOUT of [11, PRS SOME/IP Protocol]).⌋

[constr_9328] Reception Monitoring of SOME/IP segments is allowed for udp SomeipMethodDeployments

Imposition time: IT_Mani

⌈SomeipMethodDeployment.segmentReceptionTimeoutTimeRequest and SomeipMethodDeployment.segmentReceptionTimeoutTimeResponse shall only be used if the value of attribute SomeipMethodDeployment.transportProtocol is set to udp.⌋

Class	**SomeipMethodDeployment**			
Note	SOME/IP configuration settings for a Method. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceInterfaceDeploymentElement*, *ServiceMethodDeployment*			
Aggregated by	*ServiceInterfaceDeployment*.methodDeployment, SomeipFieldDeployment.get, SomeipFieldDeployment.set			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
burstSizeRequest	PositiveInteger	0..1	attr	Specifies the number of segments for the Method Call that shall be transmitted in a burst ignoring separation Time. SeparationTime will then only be applied between bursts. If not configured, SeparationTime will be applied between all frames.
burstSizeResponse	PositiveInteger	0..1	attr	Specifies the number of segments for the Method Response that shall be transmitted in a burst ignoring separationTime. SeparationTime will then only be applied between bursts. If not configured, SeparationTime will be applied between all frames.
maximumSegmentLengthRequest	PositiveInteger	0..1	attr	This attribute describes the length in bytes of one SOME/IP segment into which the Method Call Message will be divided. This length field includes 8 bytes for the Request ID, Protocol Version, Interface Version, Message Type and Return Code and 4 additional SOME/IP TP bytes. If this attribute is set to a value and the data length is larger than maximumSegmentLengthRequest then the corresponding SOME/IP message will be segmented into smaller parts that are transmitted over the network.
maximumSegmentLengthResponse	PositiveInteger	0..1	attr	This attribute describes the length in bytes of one SOME/IP segment into which the Method Return Message will be divided. This length field includes 8 bytes for the Request ID, Protocol Version, Interface Version, Message Type and Return Code and 4 additional SOME/IP TP bytes. If this attribute is set to a value and the data length is larger than maximumSegmentLengthResponse then the corresponding SOME/IP message will be segmented into smaller parts that are transmitted over the network.
methodId	PositiveInteger	0..1	attr	Unique Identifier within a ServiceInterface that identifies the Method in SOME/IP. This Identifier is sent as part of the Message ID in SOME/IP messages.
segmentReceptionTimeoutTimeRequest	TimeValue	0..1	attr	Timer to monitor the successful reception of segments (in seconds) in SOME/IP for the Method Call.

▽

△

Class	**SomeipMethodDeployment**			
segment Reception TimeoutTime Response	TimeValue	0..1	attr	Timer to monitor the successful reception of segments (in seconds) in SOME/IP for the Method Response.
separationTime Request	TimeValue	0..1	attr	Sets the duration of the minimum time in seconds SOME/IP shall wait between the transmissions of segments into which the Method Call Message will be divided.
separationTime Response	TimeValue	0..1	attr	Sets the duration of the minimum time in seconds SOME/IP shall wait between the transmissions of segments into which the Method Return Message will be divided.
transport Protocol	TransportLayerProtocol Enum	0..1	attr	This attribute defines over which Transport Layer Protocol this method is intended to be sent.

Table 13.15: SomeipMethodDeployment

Class	**SomeipServiceInstanceToMachineMapping**			
Note	This meta-class allows to map SomeipServiceInstances to a CommunicationConnector of a Machine. In this step the network configuration (IP Address, Transport Protocol, Port Number) for the ServiceInstance is defined. **Tags:** atp.recommendedPackage=ServiceInstanceToMachineMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInstanceToMachineMapping*, *UploadableDesignElement*, *Uploadable PackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
remoteMulticast Config	SomeipRemote MulticastConfig	*	ref	This reference defines a remote multicast Address (IP Address, Port) that is used in a static configuration to setup the communication path between a service provider and service consumer. This reference shall ONLY be used if the remote address is determined from the configuration and not at runtime from the Service Discovery. **Tags:** atp.Splitkey=remoteMulticastConfig
remoteUnicast Config	SomeipRemoteUnicast Config	*	ref	In case that a static service connection is used and a single peer exists this element is used to statically configure the remote peer's address. **Tags:** atp.Splitkey=remoteUnicastConfig
tcpPort	ApApplicationEndpoint	0..1	ref	local TcpPort that will be used by the ServiceInstance for the communication. **Tags:** atp.Splitkey=tcpPort
udpCollection BufferSize Threshold	PositiveInteger	0..1	attr	Specifies the amount of data in bytes that shall be buffered for data transmission over the udp connection specified by this SomeipServiceInstanceToMachine Mapping. If this attribute is set to a value, then the data collection feature is enabled.
udpPort	ApApplicationEndpoint	0..1	ref	local UdpPort that will be used by the ServiceInstance for the communication. **Tags:** atp.Splitkey=udpPort

Table 13.16: SomeipServiceInstanceToMachineMapping

[TPS_MANI_03057] SOME/IP `Field` binding ⌈The `SomeipFieldDeployment` meta-class provides the ability to bind a `Field` to SOME/IP.

If the Field contains a notifier (hasNotifier = true), it is possible to assign a SOME/IP notifier identifier to the field by setting the value of attribute SomeipFieldDeployment.notifier.eventId.

If the Field contains a getter method (hasGetter = true), it is possible to assign a SOME/IP method identifier to the field by setting the value of attribute SomeipFieldDeployment.get.methodId.

If the Field contains a setter method (hasSetter = true), it is possible to assign a SOME/IP method identifier to the field by setting the value of attribute SomeipFieldDeployment.set.methodId⌋

Each methodId and each eventId of a SomeipFieldDeployment shall be unique in the context of a ServiceInterface as defined in [constr_3306] and [constr_3305].

Class	**SomeipFieldDeployment**			
Note	SOME/IP configuration settings for a Field. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceFieldDeployment*, *ServiceInterface DeploymentElement*			
Aggregated by	*ServiceInterfaceDeployment*.fieldDeployment			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
get	SomeipMethod Deployment	0..1	aggr	This aggregation represents the setting of the get method. **Tags:** atp.Splitkey=get, get.shortName
notifier	SomeipEvent Deployment	0..1	aggr	This aggregation represents the settings of the notifier. **Tags:** atp.Splitkey=notifier, notifier.shortName
set	SomeipMethod Deployment	0..1	aggr	This aggregation represents the settings of the set method **Tags:** atp.Splitkey=set, set.shortName

Table 13.17: SomeipFieldDeployment

[constr_3362] SomeipEventDeployments aggregated by a SomeipFieldDeployment

Imposition time: IT_Mani

⌈A SomeipEventDeployment that is aggregated by a SomeipFieldDeployment in the role notifier shall not reference a VariableDataPrototype in the role event.⌋

[constr_3363] SomeipMethodDeployments aggregated by a SomeipFieldDeployment

Imposition time: IT_Mani

⌈A SomeipMethodDeployment that is aggregated by a SomeipFieldDeployment in the role get or set shall not reference a ClientServerOperation in the role method.⌋

[TPS_MANI_03227] Usage of ephemeral ports ⌈Ephemeral ports are short-lived transport protocol ports that are allocated automatically by the communication middleware. If the port number in the `ApApplicationEndpoint` (`TpPort.portNumber`) is configured to 0, an *ephemeral* port shall be used. If the port number is configured to a value different from 0, exactly that value shall be used.⌋

13.2.2 DDS Service Interface Deployment

This chapter describes the [42, DDS] deployment of a `ServiceInterface`.

[TPS_MANI_03525] DDS ServiceInterface binding ⌈The `DdsServiceInterfaceDeployment` meta-class provides the ability to bind a `ServiceInterface` to DDS and to assign a DDS Service identifier to the `ServiceInterface` with the `serviceInterfaceId` attribute.⌋

[constr_3690] `DdsServiceInterfaceDeployment.serviceInterfaceId` value shall not conflict with topic-based service discovery

Imposition time: `IT_SysDes`

⌈The value "`discovery`" for `DdsServiceInterfaceDeployment.serviceInterfaceId` is reserved and shall not be used for modeled `DdsServiceInterfaceDeployment`s in the context of the *AUTOSAR adaptive platform*.⌋

[constr_3690] is defined according to [SWS_CM_90508] of [32, SWS Communication Management].

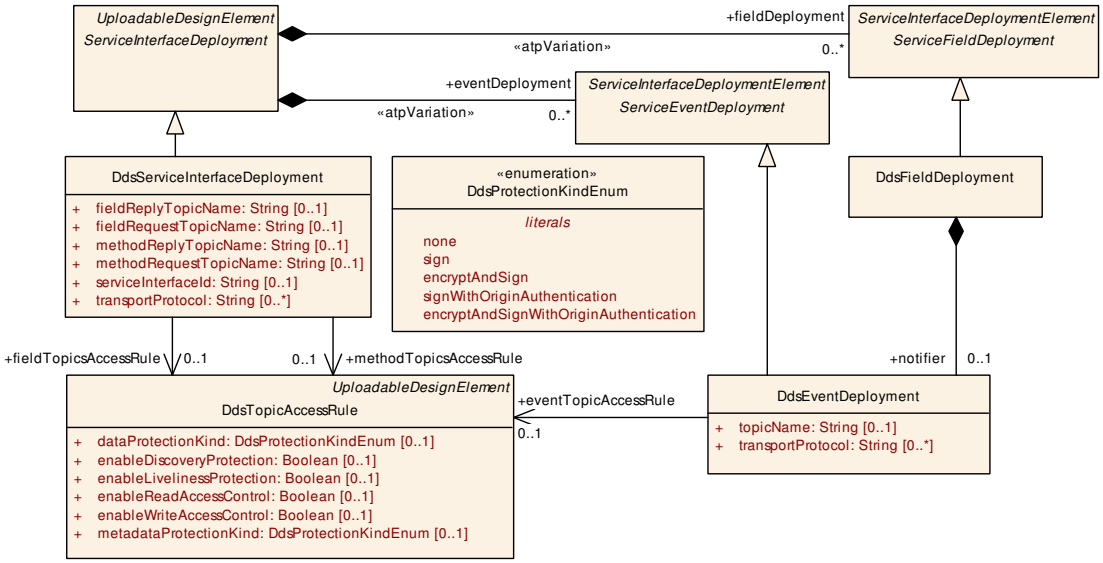


Figure 13.5: DDS deployment of ServiceInterface

[TPS_MANI_03556] DDS-RPC Service Binding ⌈The DdsServiceInterfaceDeployment meta-class provides the ability to configure the name of the DDS Request and Reply Topics associated with a DDS-RPC Service with the methodRequestTopicName and methodReplyTopicName attributes, respectively. DDS-RPC Services are the mechanisms specified in the OMG RPC over DDS specification (see [43, DDS RPC]) to handle method calls with DDS.

The methodRequestTopicName and methodReplyTopicName attributes are optional; if unspecified, they shall be configured as specified in the DDS-RPC specification.⌋

[TPS_MANI_03526] DDS VariableDataPrototype binding ⌈The DdsEventDeployment meta-class provides the ability to bind a VariableDataPrototype to DDS and to assign a DDS Topic to the event with the topicName attribute. Moreover, the meta-class provides the ability to configure the transportProtocols over which the VariableDataPrototype may be accessed.⌋

Class	**DdsServiceInterfaceDeployment**			
Note	DDS configuration settings for a ServiceInterface. **Tags:** atp.recommendedPackage=ServiceInterfaceDeployments This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *ServiceInterfaceDeployment*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
fieldReplyTopicName	String	0..1	attr	Name of the DDS Reply Topic associated with the Field.
fieldRequestTopicName	String	0..1	attr	Name of the DDS Request Topic associated with the Field.
fieldTopicsAccessRule	DdsTopicAccessRule	0..1	ref	DDS Security access rule applicable to the DDS Topics used for service interface field access methods (Get, Set). **Tags:** atp.Splitkey=fieldTopicsAccessRule
methodReplyTopicName	String	0..1	attr	Name of the DDS Reply Topic associated with the Method.
methodRequestTopicName	String	0..1	attr	Name of the DDS Request Topic associated with the Method.
methodTopicsAccessRule	DdsTopicAccessRule	0..1	ref	DDS Security access rule applicable to the DDS Topics used for service interface methods. **Tags:** atp.Splitkey=methodTopicsAccessRule
serviceInterfaceId	String	0..1	attr	Unique Identifier that identifies the ServiceInterface in DDS. This Identifier is encoded in the USER_DATA QoS of the DomainParticipant associated with the Service Instance and its value is propagated by DDS Discovery messages.
transportProtocol	String	*	attr	This attribute defines over which Transport Layer Protocol(s) this Method is intended to be sent.

Table 13.18: DdsServiceInterfaceDeployment

[constr_10217] Multiplicity of the attribute `DdsServiceInterfaceDeployment.serviceInterfaceId`

Imposition time: `IT_Mani`

⌈For each `DdsServiceInterfaceDeployment`, the primitive attribute `serviceInterfaceId` shall exist.⌋

Class	**DdsEventDeployment**			
Note	DDS configuration settings for an Event. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceEventDeployment*, *ServiceInterfaceDeploymentElement*			
Aggregated by	DdsFieldDeployment.notifier, *ServiceInterfaceDeployment*.eventDeployment			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventTopicAccessRule	DdsTopicAccessRule	0..1	ref	DDS Security access rule applicable to the DDS Topics used for the service interface event. **Tags:** atp.Splitkey=eventTopicAccessRule
topicName	String	0..1	attr	Name of the DDS Topic associated with the Event.
transportProtocol	String	*	attr	This attribute defines over which Transport Layer Protocol(s) this event is intended to be sent.

Table 13.19: DdsEventDeployment

[TPS_MANI_03557] DDS `ClientServerOperation` Binding ⌈There exists no concrete subclass of `ServiceMethodDeployment` to bind a `ClientServerOperation` to DDS. This binding is done with the `DdsServiceInterfaceDeployment` (see [TPS_MANI_03556]).⌋

[TPS_MANI_03558] DDS `Field` Binding ⌈The `DdsFieldDeployment` meta-class provides the ability to bind a `Field` to DDS.

To bind the `Field`'s notification event the `notifier` is used to define a DDS Topic. To assign the get/set `Field` methods the `fieldRequestTopicName` and `fieldReplyTopicName` are used to define a DDS Topic.

The `fieldRequestTopicName` and `fieldReplyTopicName` attributes are optional; if unspecified, they shall be configured as specified in the DDS-RPC specification.⌋

[TPS_MANI_03622] DDS Transport Protocols are up to the stack implementer ⌈Underlying transports below the RTPS protocol are not part of the DDS OMG standard (QoS APIs of [42, DDS]) or (XML schema [44, DDS XML]). It is up to each DDS implementation vendor to decide which transports are supported and how those are expressed in APIs and XML.⌋

[constr_3563] Mandatory topic name values

Imposition time: IT_Mani

⌈The attributes methodRequestTopicName, methodReplyTopicName, fieldRequestTopicName, fieldReplyTopicName, topicName shall specify string values, each of them unique within the service interface.⌋

Class	**DdsFieldDeployment**			
Note	DDS configuration settings for a Field. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceFieldDeployment*, *ServiceInterfaceDeploymentElement*			
Aggregated by	*ServiceInterfaceDeployment*.fieldDeployment			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
notifier	DdsEventDeployment	0..1	aggr	This aggregation represents the settings of the notifier. **Tags:** atp.Splitkey=notifier, notifier.shortName

Table 13.20: DdsFieldDeployment

[TPS_MANI_03662] Configuration of Topic access rules

Status: DRAFT

⌈The DdsTopicAccessRule meta-class defines rules generally applicable to DDS Topics defined by Service Interface deployments, determining the protection mechanisms applicable for discovery, liveliness, access control, metadata and user data.

The enableReadAccessControl and enableWriteAccessControl attributes are not meant to interfere with or duplicate ComGrant-based access rules. These are merely additional mechanisms for integrators to fully disable access control for a specific topic at Service Interface Deployment level, if that is desirable for a specific deployment scenario.⌋

Class	**DdsTopicAccessRule**			
Note	DDS Topic access rule definition. **Tags:** atp.recommendedPackage=DdsTopicAccessRules This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataProtectionKind	DdsProtectionKindEnum	0..1	attr	Defines the data protection policy applicable to metadata related to the DDS Topic(s).
enableDiscoveryProtection	Boolean	0..1	attr	Defines whether discovery protection mechanisms should apply to the DDS Topic(s).
enableLivelinessProtection	Boolean	0..1	attr	Defines whether liveliness protection mechanisms should apply to the DDS Topic(s).

▽

△

Class	**DdsTopicAccessRule**			
enableRead AccessControl	Boolean	0..1	attr	Defines whether read access control mechanisms should apply to the DDS Topic(s).
enableWrite AccessControl	Boolean	0..1	attr	Defines whether write access control mechanisms should apply to the DDS Topic(s).
metadata ProtectionKind	DdsProtectionKind Enum	0..1	attr	Defines the data protection policy applicable to metadata related to the DDS Topic(s).

Table 13.21: DdsTopicAccessRule

[constr_3680] Existence of attributes for DdsTopicAccessRule

Imposition time: IT_Mani

⌈The following attributes of DdsTopicAccessRule shall exist:

- enableDiscoveryProtection
- enableLivelinessProtection
- enableReadAccessControl
- enableWriteAccessControl
- metadataProtectionKind
- dataProtectionKind

⌋

[constr_3681] Supported values of DdsTopicAccessRule.dataProtectionKind

Imposition time: IT_Mani

⌈Only values none, sign, or encryptAndSign from DdsProtectionKindEnum shall be used when setting DdsTopicAccessRule.dataProtectionKind.⌋

[constr_3683] Attributes referencing DdsTopicAccessRule

Imposition time: IT_Mani

⌈DdsServiceInterfaceDeployment.fieldTopicsAccessRule, DdsServiceInterfaceDeployment.methodTopicsAccessRule, and DdsEventDeployment.eventTopicAccessRule shall be set if the Service Interface Deployment is to be used by Service Instances relying on DDS Security (meaning DdsServiceInstanceToMachineMapping.secureComPropsForDds is defined).⌋

[constr_10629] DdsServiceInterfaceDeployment shall cover all applicable elements of the corresponding ServiceInterface

Imposition time: IT_Mani

⌈If a DdsServiceInterfaceDeployment references a ServiceInterface in the role serviceInterface, then all fields, triggers, and events defined in

the context of the referenced ServiceInterface shall be referenced by respective fieldDeployments and eventDeployments owned by the referencing DdsServiceInterfaceDeployment.⌋

13.2.3 User Defined Service Interface Deployment

This chapter describes a user defined deployment of a ServiceInterface to a middleware technology that is not standardized by AUTOSAR. Such UserDefinedServiceInterfaceDeployment can, for example, also be used to describe a machine local IPC communication.

[TPS_MANI_03045] User-Defined ServiceInterface binding ⌈The UserDefinedServiceInterfaceDeployment meta-class provides the ability to bind a ServiceInterface that is referenced in the role serviceInterface to a middleware technology that is not standardized by AUTOSAR.⌋

UserDefinedServiceInterfaceDeployment is Identifiable and therefore it is able to describe special data (sdg) which is not represented by the standard model.

Class	**UserDefinedServiceInterfaceDeployment**			
Note	UserDefined configuration settings for a ServiceInterface. **Tags:** atp.recommendedPackage=ServiceInterfaceDeployments This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *ServiceInterfaceDeployment*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 13.22: UserDefinedServiceInterfaceDeployment

[TPS_MANI_01165] Standardized value of UserDefinedServiceInterfaceDeployment.category ⌈The AUTOSAR Standard reserves the following value for attribute UserDefinedServiceInterfaceDeployment.category:

- SERVICE_INTERFACE_DEPLOYMENT_IPC

It is possible to use a custom, non-standardized value for the attribute UserDefinedServiceInterfaceDeployment.category but this option comes with the obligation to use a value that is guaranteed not to clash with possible future extensions of the collection of standardized values.⌋

IPC communication may or may not require configuration settings that are nevertheless not standardized by AUTOSAR. The best support that the AUTOSAR standard can deliver is the provision of meta-classes that can be taken as the basis to define configuration settings by means of the definition of Sdg.

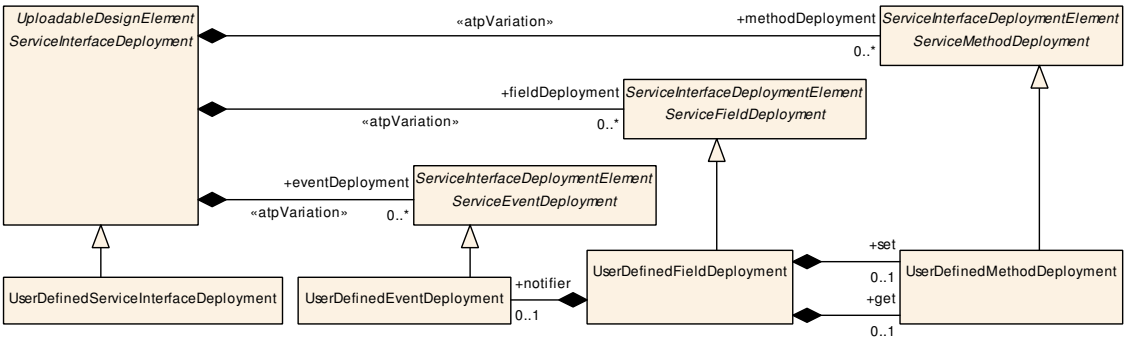


Figure 13.6: User defined deployment of ServiceInterface

[constr_1570] Restriction for `UserDefinedServiceInterfaceDeployment` of `category SERVICE_INTERFACE_DEPLOYMENT_IPC`

Imposition time: `IT_Mani`

⌈An `AdaptivePlatformServiceInstance` that references a `UserDefinedServiceInterfaceDeployment` where attribute `category` is set to `SERVICE_INTERFACE_DEPLOYMENT_IPC` shall **only** be referenced by a `UserDefinedServiceInstanceToMachineMapping` in the role `serviceInstance` that in turn references a `UserDefinedCommunicationConnector`.⌋

Rationale for [constr_1570]: for a local IPC binding it is sometimes necessary to define properties of the IPC system. For this purpose, the `UserDefinedCommunicationConnector` mapped to an `AdaptivePlatformServiceInstance` can be used to define global properties (e.g. for service discovery) of a given “IPC-Domain”.

In other words, each defined `UserDefinedCommunicationConnector` may represent such an “IPC-Domain” that requires a dedicated configuration on the basis of the definition of `Sdg`s.

Class	**UserDefinedServiceInstanceToMachineMapping**			
Note	This meta-class allows to map UserDefinedServiceInstances to a CommunicationConnector of a Machine. **Tags:** atp.recommendedPackage=ServiceInstanceToMachineMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *ServiceInstanceToMachineMapping*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 13.23: UserDefinedServiceInstanceToMachineMapping

[TPS_MANI_03046] User defined `VariableDataPrototype` binding

⌈The `UserDefinedEventDeployment` meta-class provides the ability to bind a `VariableDataPrototype` that is referenced in the role `event` to a middleware technology that is not standardized by AUTOSAR.⌋

UserDefinedEventDeployment is Identifiable and therefore it is able to describe special data (sdg) which is not represented by the standard model.

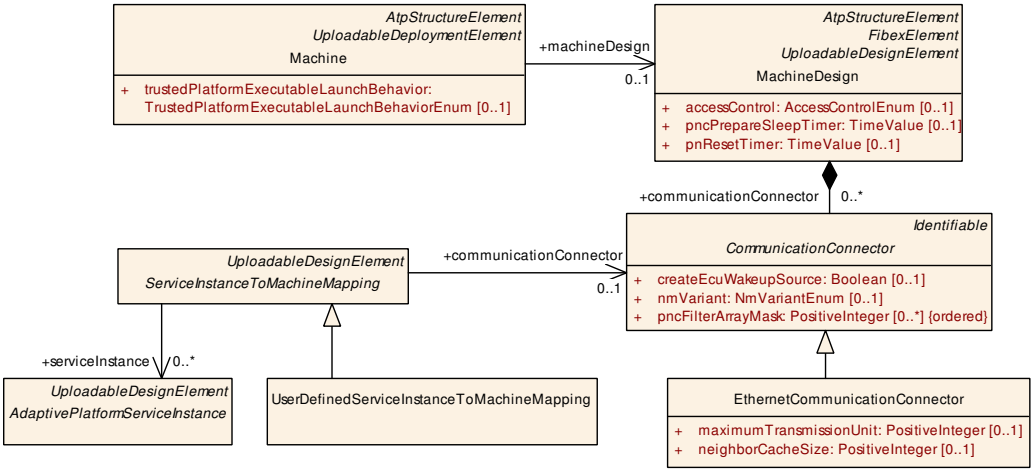


Figure 13.7: Modeling of the UserDefinedServiceInstanceToMachineMapping

Class	**UserDefinedCommunicationConnector**			
Note	This element allows the modeling of arbitrary Communication Connectors.			
Base	*ARObject*, *CommunicationConnector*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcuInstance.connector, MachineDesign.communicationConnector			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 13.24: UserDefinedCommunicationConnector

Class	**UserDefinedEventDeployment**			
Note	UserDefined configuration settings for an Event. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceEventDeployment*, *ServiceInterfaceDeploymentElement*			
Aggregated by	*ServiceInterfaceDeployment*.eventDeployment, UserDefinedFieldDeployment.notifier			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 13.25: UserDefinedEventDeployment

[TPS_MANI_03047] User defined ClientServerOperation binding ⌈The UserDefinedMethodDeployment meta-class provides the ability to bind a ClientServerOperation that is referenced in the role method to a middleware technology that is not standardized by AUTOSAR.⌋

UserDefinedMethodDeployment is Identifiable and therefore it is able to describe special data (sdg) which is not represented by the standard model.

Class	UserDefinedMethodDeployment			
Note	UserDefined configuration settings for a Method. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceInterfaceDeploymentElement*, *ServiceMethodDeployment*			
Aggregated by	*ServiceInterfaceDeployment*.methodDeployment, UserDefinedFieldDeployment.get, UserDefinedFieldDeployment.set			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 13.26: UserDefinedMethodDeployment

[TPS_MANI_03048] User defined `Field` binding ⌈The `UserDefinedFieldDeployment` meta-class provides the ability to bind a `Field` that is referenced in the role `field` to a middleware technology that is not standardized by AUTOSAR.⌋

`UserDefinedFieldDeployment` is `Identifiable` and therefore it is able to describe special data (sdg) which is not represented by the standard model.

Class	UserDefinedFieldDeployment			
Note	UserDefined configuration settings for a Field. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceFieldDeployment*, *ServiceInterfaceDeploymentElement*			
Aggregated by	*ServiceInterfaceDeployment*.fieldDeployment			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
get	UserDefinedMethodDeployment	0..1	aggr	This aggregation represents the settings of the get method **Tags:** atp.Splitkey=get, get.shortName
notifier	UserDefinedEventDeployment	0..1	aggr	This aggregation represents the settings of the notifier. **Tags:** atp.Splitkey=notifier, notifier.shortName
set	UserDefinedMethodDeployment	0..1	aggr	This aggregation represents the settings of the set method **Tags:** atp.Splitkey=set, set.shortName

Table 13.27: UserDefinedFieldDeployment

[constr_3417] `UserDefinedEventDeployment`s aggregated by a `UserDefinedFieldDeployment`

Imposition time: `IT_Mani`

⌈A `UserDefinedEventDeployment` that is aggregated by a `UserDefinedFieldDeployment` in the role `notifier` shall not reference a `VariableDataPrototype` in the role `event`.⌋

[constr_3418] UserDefinedMethodDeployments aggregated by a UserDefinedFieldDeployment

Imposition time: IT_Mani

⌈A UserDefinedMethodDeployment that is aggregated by a UserDefinedFieldDeployment in the role get or set shall not reference a ClientServerOperation in the role method.⌋

13.3 Service Instance Target-Configuration

An AdaptivePlatformServiceInstance makes the functionality of a ServiceInterface available on the *AUTOSAR adaptive platform*. Several AdaptivePlatformServiceInstances may be set up for the same ServiceInterface. They deliver the same functionality, but for different purposes and/or to different users.

The ProvidedApServiceInstance represents a provider that offers the functionality of a ServiceInterface with particular properties. Clients that are represented by the RequiredApServiceInstance observe offers and choose a provider with respect to service properties.

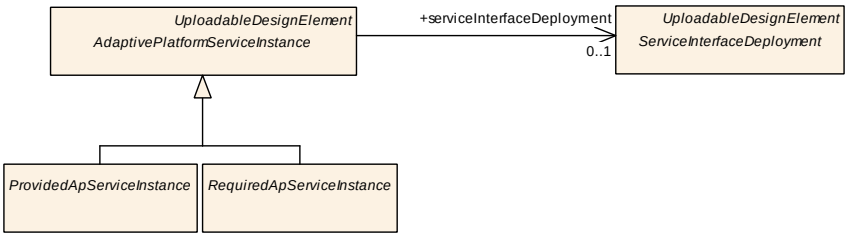


Figure 13.8: Modeling of the AdaptivePlatformServiceInstance

Note that the abstract meta-class AdaptivePlatformServiceInstance is derived from ARElement. This means that all meta-classes derived from AdaptivePlatformServiceInstance can be declared on the M1 level as part of an ARPackage and thus can be used in several Manifest descriptions.

Class	***AdaptivePlatformServiceInstance*** (abstract)			
Note	This meta-class represents the ability to describe the existence and configuration of a service instance in an abstract way. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Subclasses	*ProvidedApServiceInstance*, *RequiredApServiceInstance*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	***AdaptivePlatformServiceInstance*** (abstract)			
e2eEvent ProtectionProps	End2EndEvent ProtectionProps	*	aggr	This aggregation allows to protect an event or a field notifier that is defined inside of the ServiceInterface that is referenced by the ServiceInstance in the role service Interface. **Tags:** atp.Splitkey=e2eEventProtectionProps, e2eEvent ProtectionProps.shortName
e2eMethod ProtectionProps	End2EndMethod ProtectionProps	*	aggr	This aggregation allows to protect a method or a field getter or a field setter that is defined inside of the Service Interface that is referenced by the ServiceInstance in the role serviceInterface **Tags:** atp.Splitkey=e2eMethodProtectionProps, e2e MethodProtectionProps.shortName
secureCom Config	ServiceInterface ElementSecureCom Config	*	aggr	Configuration settings to secure the communication of ServiceInterface elements. **Tags:** atp.Splitkey=secureComConfig, secureCom Config.shortName
serviceInterface Deployment	ServiceInterface Deployment	0..1	ref	Reference to a ServiceInterfaceDeployment that identifies the ServiceInterface that is represented by the Service Instance. **Tags:** atp.Splitkey=serviceInterfaceDeployment

Table 13.28: AdaptivePlatformServiceInstance

Class	***RequiredApServiceInstance*** (abstract)			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in an abstract way. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *Uploadable PackageElement*			
Subclasses	DdsRequiredServiceInstance, RequiredSomeipServiceInstance, RequiredUserDefinedServiceInstance			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 13.29: RequiredApServiceInstance

Class	***ProvidedApServiceInstance*** (abstract)			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in an abstract way. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *Uploadable PackageElement*			
Subclasses	DdsProvidedServiceInstance, ProvidedSomeipServiceInstance, ProvidedUserDefinedServiceInstance			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 13.30: ProvidedApServiceInstance

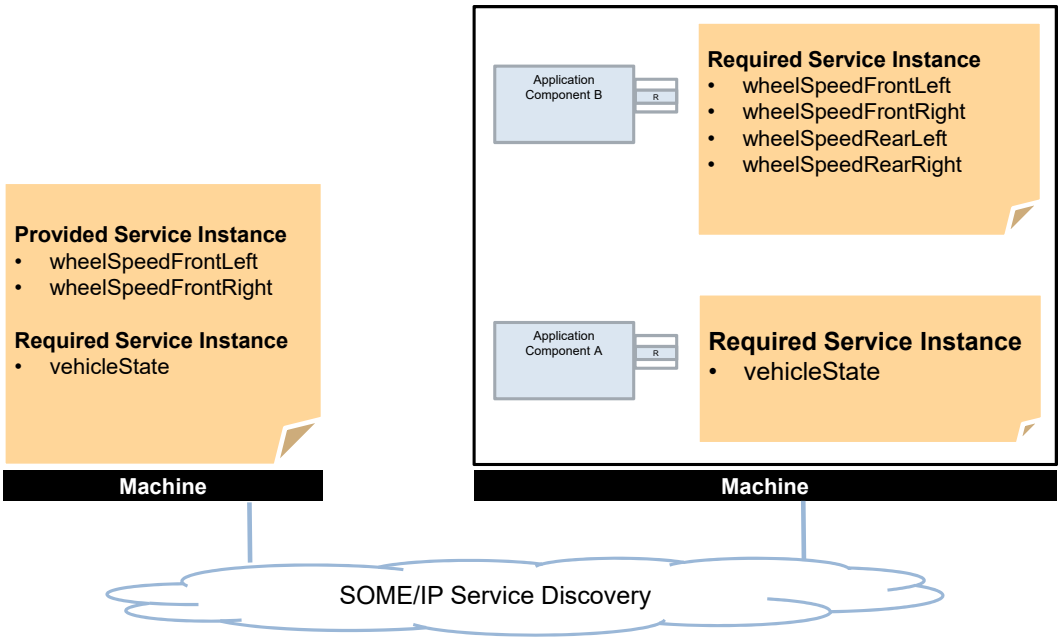


Figure 13.9: Different approaches for ServiceInstanceMapping

There are two alternative ways to relate an AdaptivePlatformServiceInstance with a MachineDesign as described in [TPS_MANI_03000] and [TPS_MANI_03001]. Figure Figure 13.9 shows both approaches in an example.

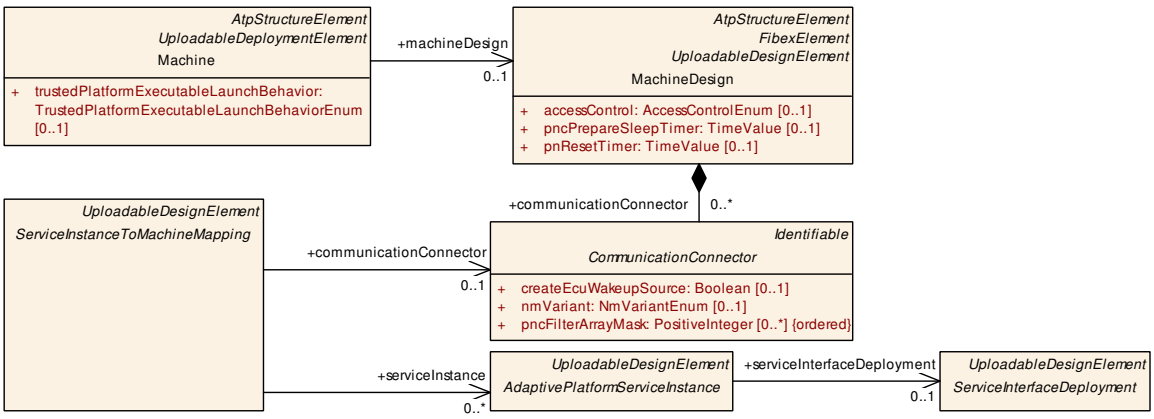


Figure 13.10: ServiceInstanceToMachineMapping

[TPS_MANI_03001] Mapping of AdaptivePlatformServiceInstance to a MachineDesign ⌈ServiceInstanceToMachineMapping is used to assign one or several AdaptivePlatformServiceInstances to (via a Communication-Connector) a MachineDesign. This allows defining a "black box" machine view without any assumption on the application software but with all necessary information to configure the communication (e.g. SOME/IP).⌋

Class	***ServiceInstanceToMachineMapping*** (abstract)			
Note	This meta-class represents the ability to map one or several AdaptivePlatformServiceInstances to a CommunicationConnector of a Machine. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Subclasses	DdsServiceInstanceToMachineMapping, SomeipServiceInstanceToMachineMapping, UserDefined ServiceInstanceToMachineMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Connector	Communication Connector	0..1	ref	Reference to the Machine to which the ServiceInstance is mapped. **Tags:** atp.Splitkey=communicationConnector
secOcCom PropsFor Multicast	SecOcSecureCom Props	*	ref	Reference to communication security configuration settings that are valid for the udp multicast endpoint (Port + Multicast IP Address) defined by the ServiceInstanceTo MachineMapping. **Tags:** atp.Splitkey=secOcComPropsForMulticast
secureCom PropsForTcp	SecureComProps	0..1	ref	Reference to communication security configuration settings that are valid for the tcp unicast endpoint (Tcp Port + Unicast IP Address) defined by the Service InstanceToMachineMapping. **Tags:** atp.Splitkey=secureComPropsForTcp
secureCom PropsForUdp	SecureComProps	0..1	ref	Reference to communication security configuration settings that are valid for the udp unicast endpoint (Udp Port + Unicast IP Address) defined by the Service InstanceToMachineMapping. **Tags:** atp.Splitkey=secureComPropsForUdp
serviceInstance	AdaptivePlatform ServiceInstance	*	ref	Reference to a ServiceInstance that is mapped to the Machine. **Tags:** atp.Splitkey=serviceInstance

Table 13.31: ServiceInstanceToMachineMapping

[constr_5155] `SomeipServiceInstanceToMachineMapping` only supports a single Address Family

Imposition time: `IT_Mani`

⌈A `SomeipServiceInstanceToMachineMapping` shall only support a single Address Family, i.e. either IPv4 or IPv6. If IPv4 is defined for IP unicast communication according to [constr_3288] then the `SomeipProvidedEventGroup`s in `ProvidedSomeipServiceInstance`s that are referenced by the `SomeipServiceInstanceToMachineMapping` shall only define an `ipv4MulticastIpAddress`.

If IPv6 is defined for IP unicast communication according to [constr_3288] then the `SomeipProvidedEventGroup`s in `ProvidedSomeipServiceInstance`s that are referenced by the `SomeipServiceInstanceToMachineMapping` shall only define an `ipv6MulticastIpAddress`.⌋

[TPS_MANI_03292] Semantics of the role `SomeipServiceInstanceToMachineMapping.tcpPort` ⌈The `SomeipServiceInstanceToMachineMapping.tcpPort` defines a TCP-Port number that will be used in the following way:

- In the case of a ProvidedSomeipServiceInstance, the value of attribute tcpPort will be transmitted in the SD-Offer message to the client and this port number will be used in the following way:
  - Method communication: over this port the ProvidedSomeipServiceInstance will receive the method call messages from the clients and over the same port the ProvidedSomeipServiceInstance will send the responses to the service consumers.
  - Event communication: over this port the ProvidedSomeipServiceInstance will transmit the event messages to the subscribed client in IP-Unicast case.
- In the case of a RequiredSomeipServiceInstance, the value of attribute tcpPort will be transmitted in the SD-SubscribeEventGroup message to the server and this port number will be used in the following way:
  - Method communication: over this port the RequiredSomeipServiceInstance will send the method call messages to the ProvidedSomeipServiceInstance and over the same port the RequiredSomeipServiceInstance will receive the responses from the service provider.
  - Event communication: over this port the RequiredSomeipServiceInstance will receive the event messages in IP-Unicast case.

⌋

[TPS_MANI_03293] Semantics of the role SomeipServiceInstanceToMachineMapping.udpPort ⌈The SomeipServiceInstanceToMachineMapping.udpPort defines a UDP-Port number that will be used in the following way:

- In the case of a ProvidedSomeipServiceInstance, the value of attribute udpPort will be transmitted in the SD-Offer message to the client and this port number will be used in the following way:
  - Method communication: over this port the ProvidedSomeipServiceInstance will receive the method call messages from the clients and over the same port the ProvidedSomeipServiceInstance will send the responses to the service consumers.
  - Event communication: over this port the ProvidedSomeipServiceInstance will transmit the event messages to the subscribed client in IP-Unicast case.
- In the case of a RequiredSomeipServiceInstance, the value of attribute udpPort will be transmitted in the SD-SubscribeEventGroup message to the server and this port number will be used in the following way:

– Method communication: over this port the RequiredSomeipServiceInstance will send the method call messages to the ProvidedSomeipServiceInstance and over the same port the RequiredSomeipServiceInstance will receive the responses from the service provider.
– Event communication: over this port the RequiredSomeipServiceInstance will receive the event messages in IP-Unicast case.

⌋

[constr_3487] TCP endpoint can only serve provided or required service instances exclusively

Imposition time: IT_Mani

⌈ServiceInstanceToMachineMapping is not allowed to refer to a ProvidedApServiceInstance and at the same time a RequiredApServiceInstance in the role serviceInstance if

- the ServiceInterfaceDeployment that is referenced by the ProvidedApServiceInstance in the role serviceInterfaceDeployment and
- the ServiceInterfaceDeployment that is referenced by the RequiredApServiceInstance in the role serviceInterfaceDeployment

both contain defined tcp content that is described by the transportProtocol attribute in the deployment elements of SOME/IP or DDS.

In other words, a TCP endpoint can only serve provided or required service instances exclusively.⌋

The reason for [constr_3487] is that the POSIX Socket API does not support the binding of several TCP sockets onto the same tuple <local IP address, local port>. But this would be necessary if a service is provided and consumed over the same TCP Endpoint.

[TPS_MANI_03000] Mapping of AdaptivePlatformServiceInstance to PortPrototypes ⌈ServiceInstanceToPortPrototypeMapping is used to assign an AdaptivePlatformServiceInstance to a PortPrototype of a SwComponentType. This allows defining how specific PortPrototypes of a Software Component are represented in the middleware in terms of the service configuration.⌋

In other words, the “outside” appearance of a PortPrototype from the middleware point of view is the AdaptivePlatformServiceInstance, or the concrete subclasses RequiredApServiceInstance and ProvidedApServiceInstance.

Meta-classes ProvidedApServiceInstance and RequiredApServiceInstance are abstract and this allows for using specific derived classes that fit the underlying middleware (e.g. SOME/IP). The following sub-chapters will detail the supported specializations.

[TPS_MANI_01316] Existence of the reference in the role ServiceInstanceToPortPrototypeMapping.processDesign ⌈The reference ServiceInstanceToPortPrototypeMapping.processDesign shall only be used in a design-level modeling scenario where a pre-assignment of a given ServiceInstanceToPortPrototypeMapping to a specific ProcessDesign is intended.

By this means it is possible to express that one Executable is foreseen to be executed in multiple instances and it is also possible to assign service instances to each of the foreseen ProcessDesigns that represent instances of Executable at design time.⌋

Class	**ServiceInstanceToPortPrototypeMapping**			
Note	This meta-class represents the ability to assign a transport layer dependent ServiceInstance to a Port Prototype. With this mapping it is possible to define how specific PortPrototypes are represented in the middleware in terms of service configuration. **Tags:** atp.recommendedPackage=ServiceInstanceToPortPrototypeMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientId	PositiveInteger	0..1	attr	This attribute defines the client identifier for a user of a Service Interface (i.e. RPortPrototype or RequriredAp ServiceInstance)
portPrototype	PortPrototype	0..1	iref	Reference to a specific PortPrototype that represents the ServiceInstance. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=portPrototype, portPrototype.context RootSwComponentPrototype, portPrototype.context ComponentPrototype, portPrototype.targetPortPrototype **InstanceRef implemented by:** PortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	Reference to the Process in which the enclosing Service InstanceToPortPrototypeMapping is executed. **Tags:** atp.Splitkey=process
processDesign	ProcessDesign	0..1	ref	Reference to the ProcessDesign in which the Executable that contains the SoftwareComponent and the referenced PortPrototype is executed. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=processDesign
serviceInstance	AdaptivePlatform ServiceInstance	0..1	ref	Reference to a ServiceInstance that is represented in the Software Component by the mapped group of Port Prototypes. **Tags:** atp.Splitkey=serviceInstance

Table 13.32: ServiceInstanceToPortPrototypeMapping

[TPS_MANI_01317] Existence of the reference in the role ServiceInstanceToPortPrototypeMapping.process ⌈The reference ServiceInstanceToPortPrototypeMapping.process shall only be used in a deployment-level modeling where the integration of a SoftwareCluster is created. The reference has the role to identify the actual Process used in the execution manifest.

This reference overwrites a potentially existing reference to a ProcessDesign in the context of the enclosing ServiceInstanceToPortPrototypeMapping.⌋

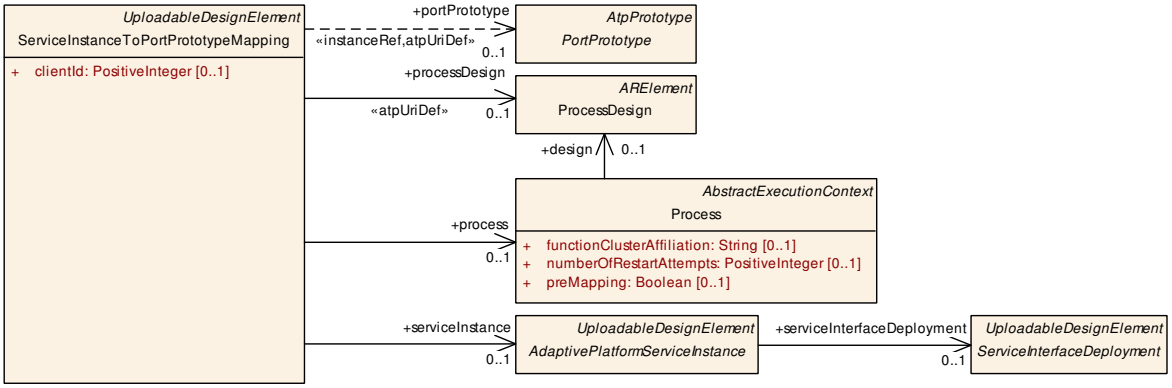


Figure 13.11: Modeling of the ServiceInstanceToPortPrototypeMapping

If both references in the roles ServiceInstanceToPortPrototypeMapping.processDesign and process exist, the latter has higher significance because it is created by an integrator who may overrule design decisions based on superior knowledge about the context.

In such a case it is acceptable that the Process references a different ProcessDesign than the one referenced in the role ServiceInstanceToPortPrototypeMapping.processDesign. This is just the result of superior knowledge of the integrator over the designer.

13.3.1 Handling of Client ID

[TPS_MANI_03349] Semantics of the attribute ServiceInstanceToPortPrototypeMapping.clientId ⌈The attribute ServiceInstanceToPortPrototypeMapping.clientId defines the client identifier for a user of a ServiceInterface and allows differentiating calls to the same method from multiple clients that are located on the same Machine.

The number that is used as clientId may be restricted by the defined ClientIdRange on the MachineDesign (see [TPS_MANI_03347] for more details).⌋

[constr_9435] ServiceInstanceToPortPrototypeMapping.clientId is only applicable to RPortPrototypes and RequiredApServiceInstances

Imposition time: IT_Mani

⌈The attribute ServiceInstanceToPortPrototypeMapping.clientId shall only be defined if the ServiceInstanceToPortPrototypeMapping references a RPortPrototype in the role portPrototype and a RequiredApServiceInstance in the role serviceInstance.⌋

[constr_9436] Restriction of allowed values for `ServiceInstanceToPortPrototypeMapping.clientId` in case that a `ClientIdRange` is defined

Imposition time: IT_Mani

⌈If

- a RequiredApServiceInstance is mapped to a MachineDesign via a ServiceInstanceToMachineMapping and
- the aggregation MachineDesign.clientIdRange exists and
- the RequiredApServiceInstance is referenced in the role serviceInstance by a ServiceInstanceToPortPrototypeMapping for which the attribute clientId exists,

then the value of the attribute ServiceInstanceToPortPrototypeMapping.clientId shall be within the range specified by the MachineDesign.clientIdRange.⌋

[constr_9437] `ServiceInstanceToPortPrototypeMapping.clientId` shall be unique in the context of a Machine

Imposition time: IT_Mani

⌈The value of the attribute clientId defined in the ServiceInstanceToPortPrototypeMapping that maps a RPortPrototype to a RequiredApServiceInstance shall be unique in the context of the MachineDesign to which the RequiredApServiceInstance is mapped via the ServiceInstanceToMachineMapping.⌋

[constr_9438] Partial configuration of `clientId`s for `ServiceInterface` users that are located on the same `Machine` is not allowed

Imposition time: IT_Mani

⌈If one ServiceInstanceToPortPrototypeMapping that maps an RPortPrototype to a RequiredApServiceInstance defines a clientId and

- this RequiredApServiceInstance is mapped to a MachineDesign via ServiceInstanceToMachineMapping and
- this RequiredApServiceInstance references a ServiceInterfaceDeployment in the role serviceInterfaceDeployment,

then all other ServiceInstanceToPortPrototypeMappings that map RequiredApServiceInstances that in turn

- reference the same ServiceInterfaceDeployment and
- are mapped via the ServiceInstanceToMachineMapping to the same MachineDesign

shall define a clientId as well.⌋

13.3.2 SOME/IP Service Instance Deployment

In the case of SOME/IP used as the middleware the derived meta-classes are `ProvidedSomeipServiceInstance` or `RequiredSomeipServiceInstance`. These meta-classes also carry attributes that apply to the service discovery on SOME/IP.

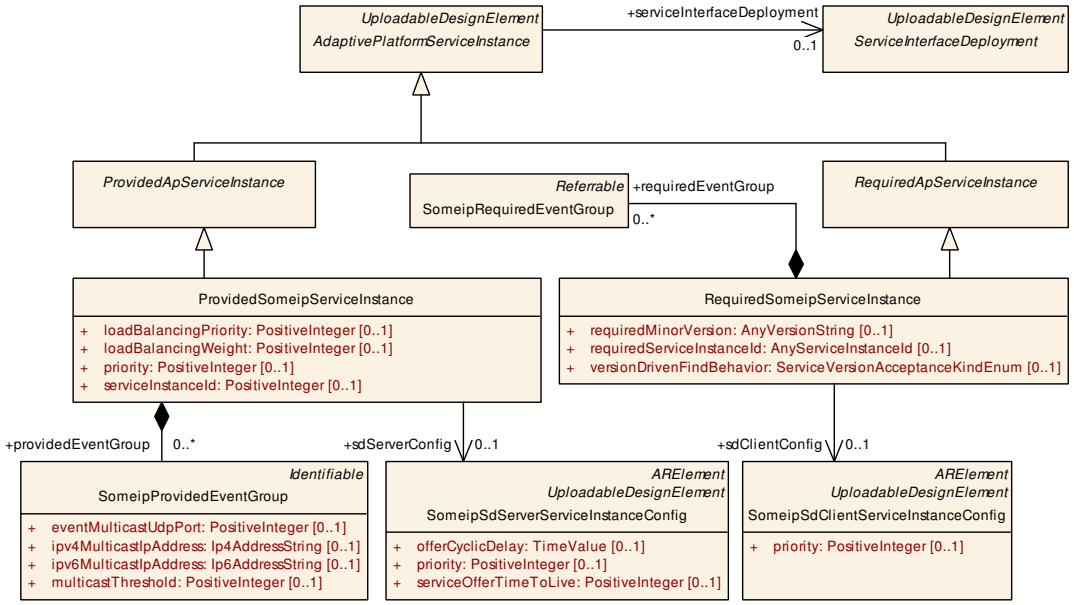


Figure 13.12: SOME/IP Service Instances

Primitive	**AnyServiceInstanceId**
Note	This is a positive integer or the literal ALL (the value ANY is technically supported but deprecated) which can be denoted in decimal, octal and hexadecimal. The value is between 0 and 65535. **Tags:** xml.xsd.customType=ANY-SERVICE-INSTANCE-ID xml.xsd.pattern=[1-9][0-9]*|0[xX][0-9a-fA-F]+|0[0-7]*|0[bB][0-1]+|ANY|ALL xml.xsd.type=string

Table 13.33: AnyServiceInstanceId

13.3.2.1 Provided Service Instance

The `ProvidedSomeipServiceInstance` defines the `serviceInstanceId` for the Service Instance of the `SomeipServiceInterfaceDeployment` that is referenced with the `serviceInterfaceDeployment` reference.

It means that the Server on which the `ProvidedSomeipServiceInstance` is deployed offers the Service Instance over SOME/IP with the `serviceInstanceId` and `serviceInterfaceId`.

Class	**ProvidedSomeipServiceInstance**			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in a concrete implementation on top of SOME/IP. **Tags:** atp.recommendedPackage=ServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *ProvidedApServiceInstance*, *Referrable*, *Uploadable DesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
capability Record (ordered)	TagWithOptionalValue	*	aggr	A sequence of records to store arbitrary name/value pairs conveying additional information about the named service.
eventProps	SomeipEventProps	*	aggr	Configuration settings for individual events that are provided by the ServiceInstance. **Tags:** atp.Splitkey=eventProps
loadBalancing Priority	PositiveInteger	0..1	attr	This attribute is used to specify the priority in the load balancing option of SOME/IP that is added to the Offer Service. When a client searches for all service instances of a service, the client shall choose the service instance with highest priority if one is defined.
loadBalancing Weight	PositiveInteger	0..1	attr	This attribute is used to specify the weight in the load balancing option of SOME/IP that is added to the Offer Service. When a client searches for all service instances of a service, the client shall choose the service instance with highest priority if one is defined. If several service instances exist with the highest priority the service instance shall be chosen based on the weights of the service instances.
method ResponseProps	SomeipMethodProps	*	aggr	Configuration settings for individual methods that are provided by the ServiceInstance. **Tags:** atp.Splitkey=methodResponseProps
priority	PositiveInteger	0..1	attr	This attribute defines the VLAN frame priority for SOME/IP messages that are resulting from this ProvidedSomeip ServiceInstance (Method and Event communication). Values from 0 (best effort) to 7 (highest) are allowed.
providedEvent Group	SomeipProvidedEvent Group	*	aggr	List of EventGroups that are provided by the Service Instance. **Tags:** atp.Splitkey=providedEventGroup, providedEvent Group.shortName
sdServerConfig	SomeipSdServer ServiceInstanceConfig	0..1	ref	Server specific configuration settings relevant for the SOME/IP service discovery. **Tags:** atp.Splitkey=sdServerConfig
serviceInstance Id	PositiveInteger	0..1	attr	Identification number that is used by SOME/IP service discovery to identify the instance of the service. The value 65535 for service instance id is reserved and should not be used.

Table 13.34: ProvidedSomeipServiceInstance

[constr_10218] Multiplicity of reference in the role ProvidedSomeipServiceInstance.sdServerConfig

Imposition time: IT_Mani

⌈For each ProvidedSomeipServiceInstance, the reference in the role sdServerConfig shall exist.⌋

[constr_10219] Multiplicity of attribute ProvidedSomeipServiceInstance.serviceInstanceId

Imposition time: IT_Mani

⌈For each ProvidedSomeipServiceInstance, the attribute serviceInstanceId shall exist.⌋

[constr_3287] Mandatory information of a ProvidedSomeipServiceInstance

Imposition time: IT_Mani

⌈The ProvidedSomeipServiceInstance shall always define the attribute serviceInstanceId.⌋

[constr_1770] Restriction for the value of attribute ProvidedSomeipServiceInstance.serviceInstanceId

Imposition time: IT_Mani

⌈For each ProvidedSomeipServiceInstance.serviceInstanceId, the value shall be in the range 0..65534.⌋

Rationale for [constr_1770]: on protocol level, the value 65535 represents the “ALL” semantics that can only be used in find messages.

In addition to the service identification properties a SOME/IP offer message contains so-called endpoint options that define how the service instance is reachable by clients.

[constr_5338] ProvidedSomeipServiceInstance shall offer all SomeipEventGroups for subscription

Imposition time: IT_Mani

⌈In the scope of a ProvidedSomeipServiceInstance, SomeipProvidedEventGroups shall be defined such that

- every aggregated ProvidedSomeipServiceInstance.providedEventGroup references a SomeipEventGroup in the context of the SomeipServiceInterfaceDeployment referenced from the enclosing ProvidedSomeipServiceInstance in the role serviceInterfaceDeployment
- each SomeipEventGroup defined in the scope of the SomeipServiceInterfaceDeployment referenced from the enclosing ProvidedSomeipServiceInstance in the role serviceInterfaceDeployment shall be referenced from exactly one SomeipProvidedEventGroup aggregated in the role providedEventGroup in the scope of the enclosing ProvidedSomeipServiceInstance.

⌋

In other words each ProvidedSomeipServiceInstance shall offer all SomeipEventGroups for subscription that are defined in the SomeipServiceInterfaceDeployment and there shall not be two or more SomeipProvidedEventGroups in the ProvidedSomeipServiceInstance that are referencing to the same SomeipEventGroup in the SomeipServiceInterfaceDeployment.

[constr_5339] Restriction for SomeipEventGroups of a SomeipServiceInterfaceDeployment shall be referenced at most once from a RequiredSomeipServiceInstance that instantiates the SomeipServiceInterfaceDeployment

Imposition time: IT_Mani

⌈Each SomeipEventGroup that is defined in a SomeipServiceInterfaceDeployment shall be referenced at most once from a SomeipRequiredEventGroup that is aggregated by the RequiredSomeipServiceInstance that is referencing the SomeipServiceInterfaceDeployment in the role serviceInterfaceDeployment.⌋

In other words there shall not be two or more SomeipRequiredEventGroups in the RequiredSomeipServiceInstance that are referencing to the same SomeipEventGroup in the SomeipServiceInterfaceDeployment.

[TPS_MANI_03168] Configuration of the SOME/IP load balancing option ⌈The SOME/IP load balancing option is configurable per ProvidedSomeipServiceInstance with the two attributes loadBalancingPriority and loadBalancingWeight.⌋

The SOME/IP load balancing option is used to prioritize different ProvidedSomeipServiceInstances that point to the same SomeipServiceInterfaceDeployment, so that a client chooses the service instance based on these settings. This option is attached to SOME/IP Offer Service entries.

[constr_3415] Value range of attribute ProvidedSomeipServiceInstance.loadBalancingPriority

Imposition time: IT_Mani

⌈The value of attribute ProvidedSomeipServiceInstance.loadBalancingPriority shall be in the range of 0..65535.⌋

According to SOME/IP, a lower value means higher priority.

[constr_3416] Value range of attribute ProvidedSomeipServiceInstance.loadBalancingWeight

Imposition time: IT_Mani

⌈The value of attribute ProvidedSomeipServiceInstance.loadBalancingWeight shall be in the range of 0..65535.⌋

According to SOME/IP, a higher value means a higher probability of being chosen.

[constr_1723] `ProvidedSomeipServiceInstance` shall be unique in respect of `serviceInstanceId`, `serviceInterfaceId` and `majorVersion` on a VLAN

Imposition time: `IT_Mani`

⌈On a VLAN, each `ProvidedSomeipServiceInstance` shall have a different `serviceInstanceId`, `serviceInterfaceId` and `majorVersion` value combination.

In other words, no two `ProvidedSomeipServiceInstance`s shall have the same `serviceInstanceId`, `serviceInterfaceId` and `majorVersion` value combination during runtime on the same VLAN.⌋

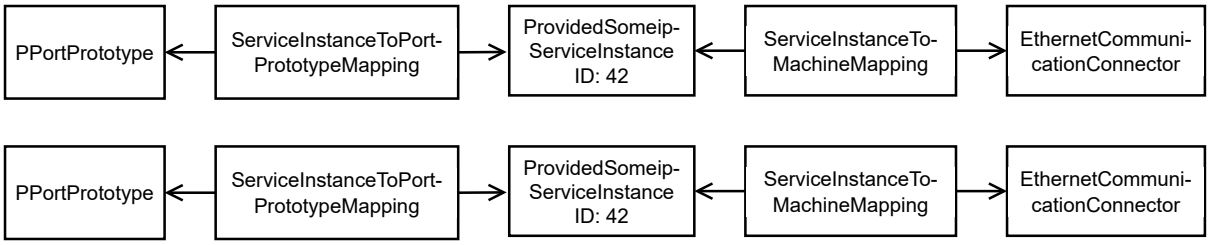


Figure 13.13: Scenario in which two `ProvidedSomeipServiceInstance`s with the same credentials are provided on two VLANs

Figure 13.13 shows that different SOME/IP `ServiceInstance`s with the same `serviceInstanceId`, `serviceInterfaceId` and `majorVersion` are provided on different VLANs. This is a valid setup according to [constr_1723].

In the following example where the same `ProvidedSomeipServiceInstance` is mapped to different `PPortPrototype`s and is provided on different VLANs the specification item [TPS_MANI_03236] applies. This means that only one of the `PPortPrototype`s is active at runtime at the same time and offers the `ServiceInstance` on two different VLANs.

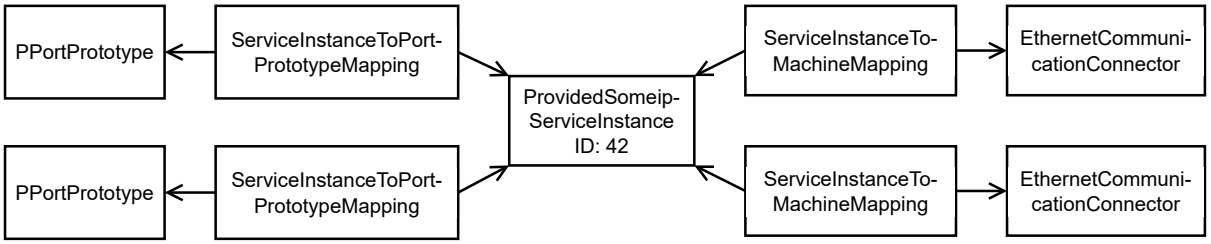


Figure 13.14: Scenario in which a `ProvidedSomeipServiceInstance` is provided on two VLANs and is mapped to different `PPortPrototype`s from which latest at runtime only one is operational

[TPS_MANI_03236] Mapping of `ProvidedSomeipServiceInstance` to different `PPortPrototype`s ⌈If the same `ProvidedSomeipServiceInstance` is mapped

by several ServiceInstanceToPortPrototypeMappings to different PPortPrototypes, it shall be ensured (latest at runtime) that only one of these mapped PPortPrototypes is actually operational at any given point in time.⌋

Two ProvidedSomeipServiceInstance elements with the same credentials according to [constr_1723] may exist that both are mapped to different PPortPrototypes and the ProvidedSomeipServiceInstances are mapped by ServiceInstanceToMachineMappings to the same VLAN. At runtime only one of these PPortPrototypes shall be operational at any given point in time.

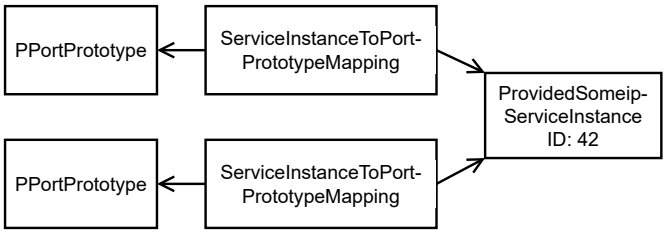


Figure 13.15: Static setup in which a ProvidedSomeipServiceInstance is mapped to two PPortPrototypes from which latest at runtime only one is operational

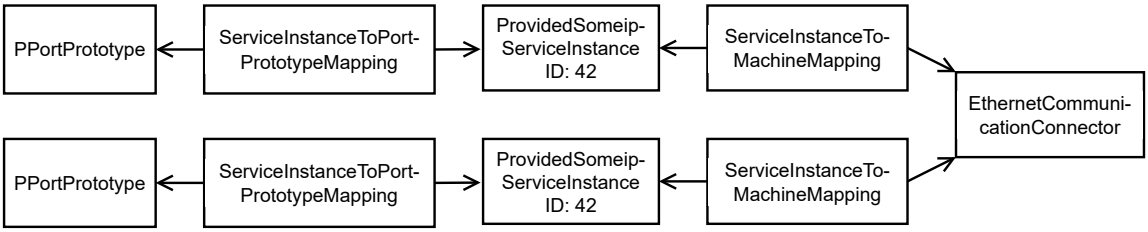


Figure 13.16: Static setup in which ProvidedSomeipServiceInstances with the same credentials provided on the same VLAN are mapped to two different PPortPrototypes from which latest at runtime only one is operational

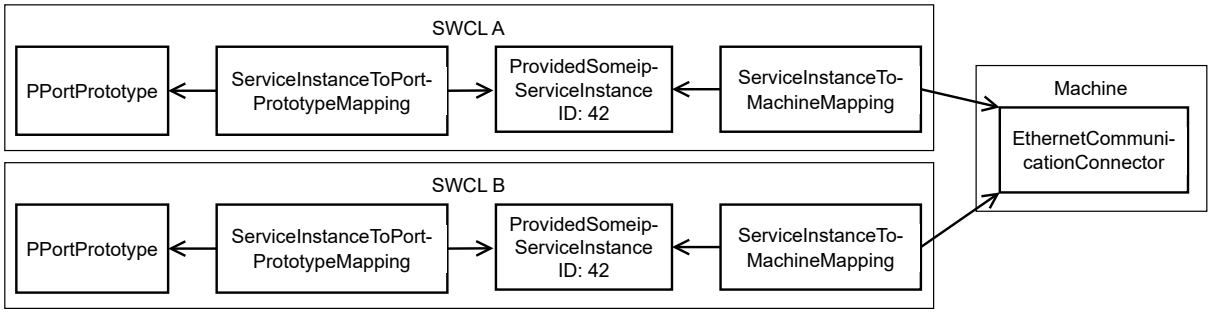


Figure 13.17: SoftwareCluster scenario in which ProvidedSomeipServiceInstances with the same credentials provided on the same VLAN exist

Such a scenario may be created by the installation of two separate SoftwareClusters as shown in the following figure. It is not possible to check such a setup since the SoftwareClusters may be developed, integrated, and deployed independently of each other.

CAUTION

The constraints marked with the imposition time IT_SysDes defined in chapter [4, TPS System Template] 6.7.4.5.3 *"SOME/IP Service Discovery Server Configuration"* are valid for this specification as well.

13.3.2.1.1 IP Configuration

In SOME/IP the Offer service entry references IPv4 or IPv6 Endpoint options to indicate to the client where the server accepts the method calls and where the server sends the event messages.

Such an Endpoint contains the IP address of the sender. The IP address configuration is described in this chapter.

[TPS_MANI_03002] IP configuration for a ProvidedSomeipServiceInstance ⌈A ProvidedSomeipServiceInstance can be mapped to a CommunicationConnector of a MachineDesign with the SomeipServiceInstanceToMachineMapping.

With this mapping an assignment of the ProvidedSomeipServiceInstance to a unicast IP Address is established since the EthernetCommunicationConnector refers to a NetworkEndpoint in the role unicastNetworkEndpoint.⌋

[TPS_MANI_03003] ProvidedSomeipServiceInstance Fanout ⌈It is allowed to map the same ProvidedSomeipServiceInstance to different CommunicationConnectors of a MachineDesign. In such a case, several SomeipServiceInstanceToMachineMappings shall be defined.

This allows for offering the same ProvidedSomeipServiceInstance on different VLANs or even on different CommunicationClusters.⌋

[constr_3538] Only one ServiceInstanceToMachineMapping per technology and CommunicationConnector

Imposition time: IT_Mani

⌈Each AdaptivePlatformServiceInstance shall only be referenced up to once by a specific ServiceInstanceToMachineMapping subclass in the role serviceInstance where the ServiceInstanceToMachineMapping refer to the same CommunicationConnector.⌋

In other words, it is not allowed to define for the same service instance two ServiceInstanceToMachineMapping of the same kind (e.g. SomeipServiceInstanceToMachineMapping) which refer to the same CommunicationConnector.

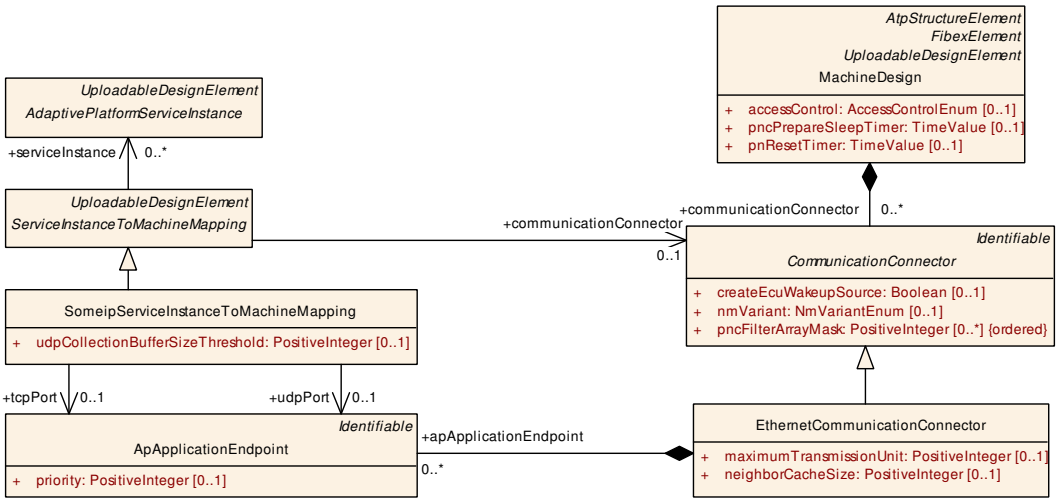


Figure 13.18: `SomeipServiceInstanceToMachineMapping` with TP and IP configuration

[TPS_MANI_03554] Several `SomeipServiceInstanceToMachineMapping`s with equal settings ⌈If

- one `SomeipServiceInstanceToMachineMapping` refers to several service instances in the role `serviceInstance`
- several `SomeipServiceInstanceToMachineMapping`s with equal settings refer to several service instances in the role `serviceInstance`
- the combination of the two above applies

then for all the referenced service instances the same network connection (i.e. Ethernet socket) will be used.⌋

[constr_5052] `ProvidedSomeipServiceInstance`s of the same serviceInterface on one Machine

Imposition time: `IT_Mani`

⌈`ProvidedSomeipServiceInstance`s that are referring to the same `SomeipServiceInterfaceDeployment` element or to several `SomeipServiceInterfaceDeployment`s that all have the same SOME/IP credentials (`SomeipServiceInterfaceDeployment.serviceInterfaceId` and `SomeipServiceInterfaceDeployment.serviceInterfaceVersion.majorVersion`) shall not be mapped to the same combination of:

- IP address that is assigned by the `SomeipServiceInstanceToMachineMapping` with the reference to the `EthernetCommunicationConnector` that in turn references the `NetworkEndpoint` and
- UDP Port or TCP Port number that are defined by the `SomeipServiceInstanceToMachineMapping.udpPort` and `SomeipServiceInstanceToMachineMapping.tcpPort` references to the `ApApplicationEndpoint`.

⌋

The reason for this restriction is that the Instance IDs are only used for Service Discovery but are not contained in the SOME/IP header. So if for example two ProvidedSomeipServiceInstances of the same ServiceInterface are provided on the same machine and a client wants to call a method of one of these ProvidedSomeipServiceInstances the only possibility for the client to distinguish the ProvidedSomeipServiceInstances is the port number over which the individual ProvidedSomeipServiceInstances are provided.

[TPS_MANI_03555] Mix of SomeipServiceInstanceToMachineMapping and signal-based communication ⌈SomeipServiceInstanceToMachineMapping defines service instance communication on a specific Ethernet socket and the same socket may also be used for signal-based communication at the same time.⌋

The signal-based communication is described in Chapter 9.

Via the definition of respective ISignalTriggering and PduTriggering for signal-based communication, the same values for Ethernet address and port may be defined as used at the SomeipServiceInstanceToMachineMapping.

Class	«atpVariation» ***CommunicationCluster*** (abstract)			
Note	The CommunicationCluster is the main element to describe the topological connection of communicating ECUs. A cluster describes the ensemble of ECUs, which are linked by a communication medium of arbitrary topology (bus, star, ring, ...). The nodes within the cluster share the same communication protocol, which may be event-triggered, time-triggered or a combination of both. A CommunicationCluster aggregates one or more physical channels. **Tags:** vh.latestBindingTime=postBuild			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Subclasses	*AbstractCanCluster*, EthernetCluster, FlexrayCluster, LinCluster, UserDefinedCluster			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
baudrate	PositiveUnlimitedInteger	0..1	attr	Channels speed in bits/s.
comMLight Timeout	TimeValue	0..1	attr	Defines the timeout (in seconds) after COMM_FULL_COMMUNICATION sub-state COMM_FULL_COM_READY_SLEEP is left in the ComM channel state machine.
physical Channel	PhysicalChannel	*	aggr	This relationship defines which channel element belongs to which cluster. A channel shall be assigned to exactly one cluster, whereas a cluster may have one or more channels. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=physicalChannel, physicalChannel.shortName, physicalChannel.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime
protocolName	String	0..1	attr	The name of the protocol used.
protocolVersion	String	0..1	attr	The version of the protocol used.

Table 13.35: CommunicationCluster

Class	***CommunicationConnector*** (abstract)			
Note	The connection between the referencing ECU and the referenced channel via the referenced controller. Connectors are used to describe the bus interfaces of the ECUs and to specify the sending/receiving behavior. Each CommunicationConnector has a reference to exactly one communicationController. Note: Several CommunicationConnectors can be assigned to one PhysicalChannel in the scope of one ECU Instance.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractCanCommunicationConnector*, EthernetCommunicationConnector, FlexrayCommunication Connector, UserDefinedCommunicationConnector			
Aggregated by	EcuInstance.connector, MachineDesign.communicationConnector			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
commController	Communication Controller	0..1	ref	Reference to the communication controller. The CommunicationConnector and referenced CommunicationController shall be aggregated by the same ECUInstance. The communicationController can be referenced by several CommunicationConnector elements. This is important for the FlexRay Bus. FlexRay communicates via two physical channels. But only one controller in an ECU is responsible for both channels. Thus, two connectors (for channel A and for channel B) shall reference to the same controller. **Tags:** atp.Splitkey=commController
createEcu WakeupSource	Boolean	0..1	attr	If this parameter is available and set to true then a channel wakeup source shall be created for the Physical Channel referencing this CommunicationConnector.
explicitWakeup Channel	PhysicalChannel	*	ref	Defines a restriction which PhysicalChannels shall be woken up if this CommunicationConnector is the wakeup source. If not defined, then no restriction applies. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=explicitWakeupChannel, explicitWakeup Channel.physicalChannel, explicitWakeup Channel.variationPoint.shortLabel vh.latestBindingTime=postBuild
explicitWakeup Pnc	PncMappingIdent	*	ref	Defines a restriction which PNCs shall be woken up if this CommunicationConnector is the wakeup source. If not defined, then no restriction applies. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=explicitWakeupPnc, explicitWakeupPnc.pnc MappingIdent, explicitWakeupPnc.variationPoint.short Label vh.latestBindingTime=postBuild
nmVariant	NmVariantEnum	0..1	attr	Defines the functionality of Network Management.
pncFilterArray Mask (ordered)	PositiveInteger	*	attr	Bit mask for NM-Pdu Payload used to configure the NM filter mask for the Network Management.

Table 13.36: CommunicationConnector

Class	**EthernetCommunicationConnector**			
Note	Ethernet specific attributes to the CommunicationConnector.			
Base	*ARObject*, *CommunicationConnector*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcuInstance.connector, MachineDesign.communicationConnector			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	EthernetCommunicationConnector			
apApplication Endpoint	ApApplicationEndpoint	*	aggr	Collection of Application Addresses that are used on the CommunicationConnector. **Tags:** atp.Splitkey=apApplicationEndpoint, apApplication Endpoint.shortName This Attribute is only used by the AUTOSAR Adaptive Platform.
canXlProps	CanXlProps	*	ref	If the Ethernet frames handled by this Ethernet CommunicationConnector are tunneled through CAN XL, then this reference shall refer the CanXlProps which contains the specific configuration parameters of the CAN XL controller of the physical CAN XL connection to be used for tunneling. **Tags:** atp.Splitkey=canXlProps This Attribute is only used by the AUTOSAR Adaptive Platform.
maximum Transmission Unit	PositiveInteger	0..1	attr	This attribute specifies the maximum transmission unit in bytes.
neighborCache Size	PositiveInteger	0..1	attr	This attribute specifies the size of neighbor cache or ARP table in units of entries.
unicastNetwork Endpoint	NetworkEndpoint	*	ref	Network Endpoint that defines the IPAddress of the machine. **Tags:** atp.Splitkey=unicastNetworkEndpoint This Attribute is only used by the AUTOSAR Adaptive Platform.

Table 13.37: EthernetCommunicationConnector

[constr_3721] Existence of reference in the role `EthernetCommunicationConnector.unicastNetworkEndpoint`

Imposition time: `IT_Mani`

⌈In the context of `EthernetCommunicationConnector`, the reference in the role `unicastNetworkEndpoint` shall exist at most once.⌋

[constr_3722] Existence of reference in the role `EthernetCommunicationConnector.canXlProps`

Imposition time: `IT_Mani`

⌈In the context of `EthernetCommunicationConnector`, the reference in the role `canXlProps` shall exist at most once.⌋

[constr_3288] IP configuration restriction for `unicastNetworkEndpoint`s

Imposition time: `IT_Mani`

⌈Any `NetworkEndpoint` that is referenced by a `EthernetCommunicationConnector` in the role `unicastNetworkEndpoint` shall have either

- one `Ipv4Configuration` or
- one `Ipv6Configuration`

as `networkEndpointAddress` that is defined in the unicast IP range according to the rules defined in [TPS_MANI_03005] and [TPS_MANI_03006].⌋

In SOME/IP, a server that offers a ProvidedSomeipServiceInstance is able to send events and notification events to an IP-Multicast address.

As the server needs to inform the client to which Multicast IP address the event messages will be sent, the *Subscribe Eventgroup Acknowledgement Entry* contains a reference to an IPv4 Multicast Option and/or an IPv6 Multicast Option.

[TPS_MANI_03004] IPv4 Multicast event destination address ⌈Meta-class SomeipProvidedEventGroup defines the multicast IPv4 address to which the events and notification events of the SomeipProvidedEventGroup are sent to with the attribute ipv4MulticastIpAddress.⌋

[TPS_MANI_03061] IPv6 Multicast event destination address ⌈Meta-class SomeipProvidedEventGroup defines the multicast IPv6 address to which the events and notification events of the SomeipProvidedEventGroup are sent to with the attribute ipv6MulticastIpAddress.⌋

[TPS_MANI_03005] IPv4 Multicast address range

Status: DRAFT

⌈The IPv4 addresses reserved for multicast communication are in the range 224.0.0.0 through 239.255.255.255. Addresses between 0.0.0.0 and 223.255.255.255 are reserved for unicast communication.⌋

The endpoint 224.0.0.251:5353 is reserved by [28, IANA] [29] for the utilization of mDNS (only relevant if SOVD is used).

[TPS_MANI_03006] IPv6 Multicast address range ⌈IPv6 multicast addresses are distinguished from unicast addresses by the value of the high-order octet of the addresses: a value of 0xFF (binary 11111111) identifies an address as an address reserved for multicast communication; any other value identifies an address as a unicast address.⌋

The endpoint [FF02::FB]:5353 is reserved by [30, IANA] [29] for the utilization of mDNS (only relevant if SOVD is used).

Class	***NetworkEndpointAddress*** (abstract)			
Note	To build a valid network endpoint address there has to be either one MAC multicast group reference or an ipv4 configuration or an ipv6 configuration.			
Base	*ARObject*			
Subclasses	Ipv4Configuration, Ipv6Configuration, MacMulticastConfiguration			
Aggregated by	NetworkEndpoint.networkEndpointAddress			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 13.38: NetworkEndpointAddress

13.3.2.1.2 TP Configuration

The IPv4 or IPv6 Endpoint option that is referenced in the SOME/IP Offer message contains besides the IP address the transport layer protocol (e.g. UDP or TCP), and the port number of the sender.

With the SomeipServiceInstanceToMachineMapping the Transport Layer configuration attributes are assigned to the ProvidedSomeipServiceInstance.

The same element contains the Transport Layer configuration attributes for the IPv4/IPv6 Multicast Option that may be used in the SOME/IP SubscribeEventGroupAck message.

[TPS_MANI_03007] Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance ⌈The reference SomeipServiceInstanceToMachineMapping.udpPort to the ApApplicationEndpoint defines the Transport Protocol for a UDP communication.

This setting is used in an IPv4 or IPv6 Endpoint Option that is referenced by an OfferService entry.⌋

[TPS_MANI_03008] Tcp Transport Protocol Configuration for ProvidedSomeipServiceInstance ⌈The reference SomeipServiceInstanceToMachineMapping.tcpPort to the ApApplicationEndpoint defines the Transport Protocol for a TCP communication.

This setting is used in an IPv4 or IPv6 Endpoint Option that is referenced by an OfferService entry.⌋

[TPS_MANI_03009] Tcp and Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance ⌈It is allowed to set tcpPort and udpPort in the same SomeipServiceInstanceToMachineMapping.

Such a setting shall be used to indicate that one UDP endpoint and one TCP endpoint are referenced in the OfferService entry. It means that the Server provides the ProvidedSomeipServiceInstance over both Transport Protocols.⌋

If a Tcp and Udp Transport Protocol Configuration is defined for a ProvidedSomeipServiceInstance as described in [TPS_MANI_03009] then the SOME/IP ServiceInterfaceDeployment settings decide which content of the ProvidedSomeipServiceInstance is transported over udp and which content is transported over tcp.

This is described in [TPS_MANI_03050] and [TPS_MANI_03051].

[TPS_MANI_03010] Udp Transport Protocol Configuration in the case of IP-Multicast ⌈The SomeipProvidedEventGroup.eventMulticastUdpPort defines the Transport Protocol Port Number for a UDP event communication if IP-Multicast is used.

This setting is used in an IPv4 or IPv6 Multicast Option that is referenced by a SubscribeEventGroupAck Service entry.⌋

[constr_3290] Transport Protocol attributes defined for a ProvidedSomeipServiceInstance

Imposition time: IT_Mani

⌈Each SomeipServiceInstanceToMachineMapping that is defined for a ProvidedSomeipServiceInstance shall define either

- a udpPort or
- a tcpPort or
- a udpPort and a tcpPort.

⌋

[TPS_MANI_03157] Enabling of data collection for UDP data transmission ⌈The setting of the attribute SomeipServiceInstanceToMachineMapping.udpCollectionBufferSizeThreshold to a value enables the data collection for data transmission over the udpPort and unicastNetworkEndpoint defined on the EthernetCommunicationConnector that is referenced by the SomeipServiceInstanceToMachineMapping. In this case all event and method messages that are configured for data collection will be collected in the buffer until a transmission trigger arrives and the data transmission starts.⌋

For configuration of transmission triggers please see [TPS_MANI_03158] and [TPS_MANI_03159].

13.3.2.1.3 Service Discovery Server Configuration

The multicast messages of the SOME/IP Service Discovery come with the risk of overflowing Machines with too many messages. Therefore, the Service Discovery can be configured with a suitable message sending behavior.

For every ProvidedSomeipServiceInstance on a Server different phases are existing:

- Down
- Available
  - Initial Wait Phase
  - Repetition Phase
  - Main Phase

[TPS_MANI_03011] Server Timing configuration for a ProvidedSomeipServiceInstance ⌈The Server Timing is configurable with SomeipSdServerServiceInstanceConfig that is referenced in the role sdServerConfig by the ProvidedSomeipServiceInstance for which the Timing is valid.⌋

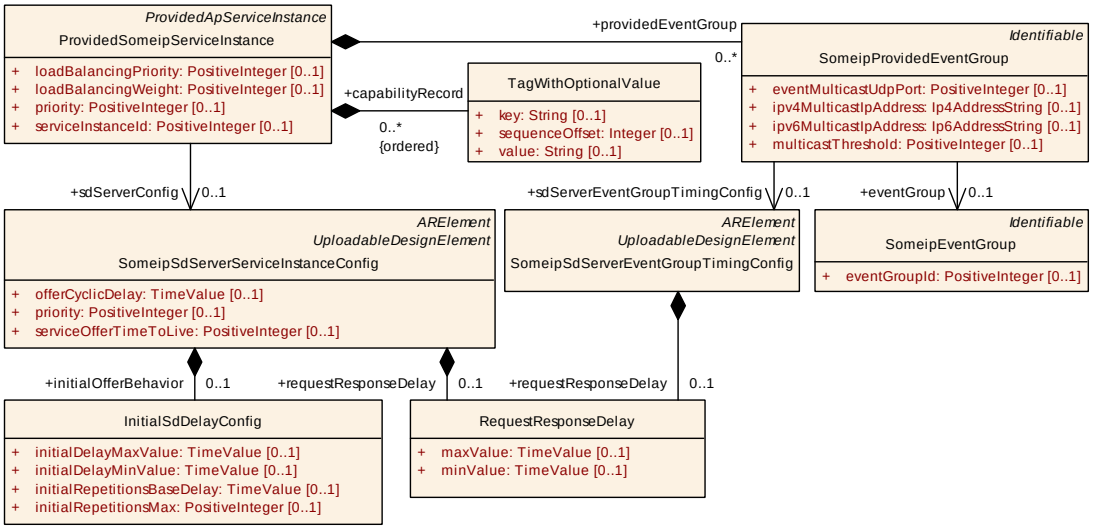


Figure 13.19: SOME/IP Service Discovery Server configuration settings

The number of SomeipSdServerServiceInstanceConfig elements determines how many timers shall actually be used by the middleware to keep the randomized times. Via the reference ProvidedSomeipServiceInstance.sdServerConfig each ProvidedSomeipServiceInstance defines to which timer it is assigned.

If several ProvidedSomeipServiceInstances share the same timer then the expiration of that timer will lead to a combined sending of service discovery messages.

Note that it is possible to define several SomeipSdServerServiceInstanceConfig elements with identical timing specification values in order to request several timer handling in the middleware.

[TPS_MANI_03230] Sharing timers for ProvidedSomeipServiceInstance ⌈If several ProvidedSomeipServiceInstances point to the same SomeipSdServerServiceInstanceConfig in the role sdServerConfig then all of these ProvidedSomeipServiceInstances will share the same timers for their timing behavior. This will lead to combining several service discovery entries in one service discovery message.⌋

Class	**SomeipSdServerServiceInstanceConfig**
Note	Server specific settings that are relevant for the configuration of SOME/IP Service-Discovery. **Tags:** atp.recommendedPackage=SomeipSdTimingConfigs
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*

▽

△

Class	**SomeipSdServerServiceInstanceConfig**			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initialOffer Behavior	InitialSdDelayConfig	0..1	aggr	Controls offer behavior of the server. **Tags:** atp.Splitkey=initialOfferBehavior
offerCyclicDelay	TimeValue	0..1	attr	Optional attribute to define cyclic offers. Cyclic offer is active, if the delay is set (in seconds) and greater then 0.
priority	PositiveInteger	0..1	attr	This attribute defines the VLAN frame priority for Service Discovery messages that result from ProvidedSomeip ServiceInstances that are referencing the SomeipSd ServerServiceInstanceConfig (OfferService, StopOffer Service, SubscribeEventGroupAck). Values from 0 (best effort) to 7 (highest) are allowed.
request ResponseDelay	RequestResponseDelay	0..1	aggr	Maximum/Minimum allowable response delay to entries received by multicast in seconds. The Service Discovery shall delay answers to entries that were transported in a multicast SOME/IP-SD message (e.g. FindService). **Tags:** atp.Splitkey=requestResponseDelay
serviceOffer TimeToLive	PositiveInteger	0..1	attr	Defines the time in seconds the service offer is valid.

Table 13.39: SomeipSdServerServiceInstanceConfig

[TPS_MANI_03012] Initial Wait Phase configuration for a `ProvidedSomeipServiceInstance` ⌈The Initial Wait Phase for a `ProvidedSomeipServiceInstance` is configured with the `initialOfferBehavior` and the two attributes `initialDelayMinValue` and `initialDelayMaxValue`.

When a calculated random timer based on these min and max values expires the first `OfferService` entry will be sent out.⌋

When the calculated random timer expires, the Repetition Phase will be entered.

[TPS_MANI_03013] Repetition Phase configuration for a `ProvidedSomeipServiceInstance` ⌈The Repetition Phase for a `ProvidedSomeipServiceInstance` is configured with the `initialOfferBehavior` and the two attributes `initialRepetitionsMax` and `initialRepetitionsBaseDelay`.⌋

If the Repetition Phase is entered the Service Discovery waits for the `initialRepetitionsBaseDelay` and then sends an `OfferService` entry. If the amount of sent `OfferService` entries reaches `initialRepetitionsMax`, the Main Phase will be entered.

[TPS_MANI_03014] Main Phase configuration for a `ProvidedSomeipServiceInstance` ⌈The Main Phase for a `ProvidedSomeipServiceInstance` is configured with the `offerCyclicDelay` attribute of `SomeipSdServerServiceInstanceConfig`.

The `OfferService` entry will be sent cyclically with an interval that is defined by the value of attribute `offerCyclicDelay`, if the value is greater than 0.⌋

Class	InitialSdDelayConfig			
Note	This element is used to configure the offer behavior of the server and the find behavior on the client.			
Base	*ARObject*			
Aggregated by	SomeipSdClientServiceInstanceConfig.initialFindBehavior, SomeipSdServerServiceInstanceConfig.initialOfferBehavior			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
initialDelayMaxValue	TimeValue	0..1	attr	Max Value in seconds to delay randomly the first offer (if aggregated in role initialOfferBehavior by SomeipSdServerServiceInstanceConfig) or the transmission of a find message (if aggregated in role initialFindBehavior by SomeipSdClientServiceInstanceConfig).
initialDelayMinValue	TimeValue	0..1	attr	Min Value in seconds to delay randomly the first offer (if aggregated in role initialOfferBehavior by SomeipSdServerServiceInstanceConfig) or the transmission of a find message (if aggregated in role initialFindBehavior by SomeipSdClientServiceInstanceConfig).
initialRepetitionsBaseDelay	TimeValue	0..1	attr	The base delay for offer repetitions (if aggregated in role initialOfferBehavior by SomeipSdServerServiceInstanceConfig) or find repetitions (if aggregated in role initialFindBehavior by SomeipSdClientServiceInstanceConfig). Successive find messages have an exponential back off delay.
initialRepetitionsMax	PositiveInteger	0..1	attr	Describes the maximum amount of offer repetitions (if aggregated in role initialOfferBehavior by SomeipSdServerServiceInstanceConfig) or the maximum amount of find repetitions (if aggregated in role initialFindBehavior by SomeipSdClientServiceInstanceConfig).

Table 13.40: InitialSdDelayConfig

[TPS_MANI_03015] TTL for Offer Service Entries ⌈The lifetime of a `ProvidedSomeipServiceInstance` is configurable with the `serviceOfferTimeToLive` attribute of `SomeipSdServerServiceInstanceConfig`.

If the time that is configured by `serviceOfferTimeToLive` expires, the `ProvidedSomeipServiceInstance` is no longer offered.

If the `offerCyclicDelay` attribute of `SomeipSdServerServiceInstanceConfig` is not configured, or the configured value of `offerCyclicDelay` is 0, the configured value of `serviceOfferTimeToLive` shall be ignored, and the infinite value 0xFFFFFF (24 bits) shall be used instead.⌋

The [4, TPS System Template] defines [constr_5382] and [constr_5383] that are also valid for `SomeipSdServerServiceInstanceConfig`s that are referenced by a `ProvidedSomeipServiceInstance`.

[TPS_MANI_03016] Servers `RequestResponseDelay` for received `FindService` entries ⌈The Server will delay the `OfferService` answer to a received multicast `FindService` entry by the configured `SomeipSdServerServiceInstanceConfig.requestResponseDelay`.

The actual delay will be randomly chosen between the `maxValue` and `minValue`.⌋

Class	**RequestResponseDelay**			
Note	Time to wait before answering the query.			
Base	*ARObject*			
Aggregated by	SomeipSdClientEventGroupTimingConfig.requestResponseDelay, SomeipSdServerEventGroupTimingConfig.requestResponseDelay, SomeipSdServerServiceInstanceConfig.requestResponseDelay			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
maxValue	TimeValue	0..1	attr	Maximum allowable response delay to entries received by multicast in seconds.
minValue	TimeValue	0..1	attr	Minimum allowable response delay to entries received by multicast in seconds.

Table 13.41: RequestResponseDelay

Figure 13.20 shows an example of the different SOME/IP phases on the Server side.

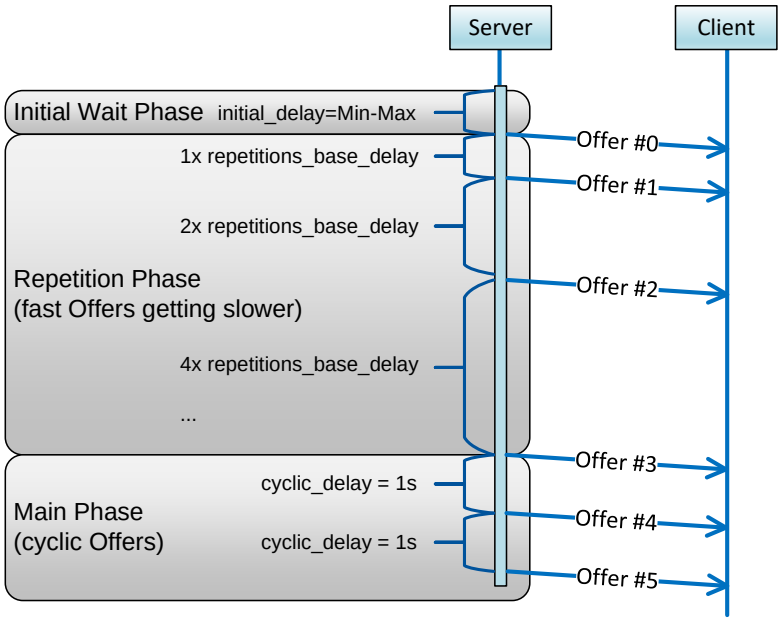


Figure 13.20: SOME/IP Server Timing example

SOME/IP allows for the specification of additional information about the `ProvidedSomeipServiceInstance` with the Capability Record that allows transporting arbitrary configuration strings (key/value pairs). This allows encoding additional information like the name of a service or its configuration.

[TPS_MANI_03017] Server Capability Records ⌈A Capability Record (key/value pair) on the Server side is configurable with the `capabilityRecord` and the two attributes `key` and `value`.⌋

Class	TagWithOptionalValue			
Note	A tagged value is a combination of a tag (key) and a value that gives supplementary information that is attached to a model element. Please note that keys without a value are allowed.			
Base	*ARObject*			
Aggregated by	*AbstractServiceInstance*.capabilityRecord, Machine.environmentVariable, ProvidedSomeipService Instance.capabilityRecord, RequiredSomeipServiceInstance.capabilityRecord, Startup Config.environmentVariable, StateDependentStartupConfig.environmentVariable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
key	String	0..1	attr	Defines a key.
sequenceOffset	Integer	0..1	attr	The sequenceOffset attribute supports the use case where TagWithOptionalValue is aggregated as splitable. If multiple aggregations define the same value of attribute key then the order in which the value collection is merged might be significant. As an example consider the modeling of the $PATH environment variable by means of a meta class TagWithOptionalValue. The sequenceOffset describes the relative position of each contribution in the concatenated value. The contributions are sorted in increasing integer order.
value	String	0..1	attr	Defines the corresponding value.

Table 13.42: TagWithOptionalValue

[constr_10037] Existence of attribute `TagWithOptionalValue.sequenceOffset` in the context of attribute `capabilityRecord` owned by `ProvidedSomeipServiceInstance`, `RequiredSomeipServiceInstance`, or `AbstractServiceInstance`

Imposition time: `IT_Mani`

⌈For all `capabilityRecord` modeled in the context of `ProvidedSomeipServiceInstance`, `RequiredSomeipServiceInstance`, or `AbstractServiceInstance`, attribute `TagWithOptionalValue.sequenceOffset` shall not exist.⌋

13.3.2.1.4 Provided Event Group

The `ProvidedSomeipServiceInstance` aggregates a `SomeipProvidedEventGroup` in the role `providedEventGroup` that allows defining service instance specific configuration settings for a `SomeipEventGroup`.

Class	SomeipProvidedEventGroup			
Note	The meta-class represents the ability to configure ServiceInstance related communication settings on the provided side for each EventGroup separately. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ProvidedSomeipServiceInstance.providedEventGroup			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**SomeipProvidedEventGroup**			
eventGroup	SomeipEventGroup	0..1	ref	Reference to the SomeipEventGroup in the System Manifest for which the ServiceInstance related Event Group settings are valid. **Tags:** atp.Splitkey=eventGroup
eventMulticast UdpPort	PositiveInteger	0..1	attr	UdpPort configuration that is used for Event communication in the IP-Multicast case. During SOME/IP Service Discovery: Send in the SD-SubscribeEventGroupAck Message to client (answer to SD-SubscribeEventGroup). Event: This is the destination-port where the server sends the multicast event messages if the multicastThreshold is exceeded.
ipv4MulticastIp Address	Ip4AddressString	0..1	attr	Multicast IPv4 Address that is transmitted in the Event GroupSubscribeAck message.
ipv6MulticastIp Address	Ip6AddressString	0..1	attr	Multicast IPv6 Address that is transmitted in the Event GroupSubscribeAck message.
multicast Threshold	PositiveInteger	0..1	attr	Specifies the number of subscribed clients that trigger the server to change the transmission of events to multicast. Example: If configured to 0 only unicast will be used. If configured to 1 the first client will be already served by multicast. If configured to 2 the first client will be served with unicast and as soon as the 2nd client arrives both will be served by multicast. This does not influence the handling of initial events, which are served using unicast only.
sdServerEvent GroupTiming Config	SomeipSdServerEvent GroupTimingConfig	0..1	ref	Server Timing configuration settings that are EventGroup specific. **Tags:** atp.Splitkey=sdServerEventGroupTimingConfig

Table 13.43: SomeipProvidedEventGroup

[constr_10220] Multiplicity of the primitive attribute `SomeipProvidedEventGroup.multicastThreshold`

Imposition time: `IT_Mani`

⌈For each `SomeipProvidedEventGroup`, the attribute `multicastThreshold` shall exist.⌋

[TPS_MANI_03018] Usage of `SomeipProvidedEventGroup.multicastThreshold` ⌈The switching between IP-Unicast and IP-Multicast is guided by the server with the `SomeipProvidedEventGroup.multicastThreshold` attribute and by the number of subscribed clients to the `SomeipProvidedEventGroup`.

The Server will change the transmission of events to Multicast if the `multicastThreshold` of the corresponding `SomeipProvidedEventGroup` is reached by the number of subscribed clients. If the number of subscribed clients is smaller than the configured `multicastThreshold`, the transmission of events takes place via unicast communication.⌋

The following example shows the effect of the `multicastThreshold` in relation to the number of subscribed clients to the transmission of the SOME/IP event to the unicast or multicast destination address:

- If multicastThreshold is configured to 0, only the unicast IP address and the port will be used as destination address.
- If multicastThreshold is configured to 1, the first client will be served by multicast.
- If multicastThreshold is configured to 2, the first client will be served with unicast and as soon as the second client arrives both will be served by multicast, etc.

[constr_5332] Mandatory multicast endpoint if multicastThreshold is different from 0

Imposition time: IT_Mani

⌈If SomeipProvidedEventGroup.multicastThreshold is configured to a value different from 0, then

- SomeipProvidedEventGroup.eventMulticastUdpPort

and either

- SomeipProvidedEventGroup.ipv4MulticastIpAddress or
- SomeipProvidedEventGroup.ipv6MulticastIpAddress

⌋

[constr_5333] No multicast if TCP is used

Imposition time: IT_Mani

⌈If a SomeipProvidedEventGroup references only SomeipEventDeployments that have the attribute transportProtocol set to tcp (via SomeipProvided-EventGroup.eventGroup.event) then this SomeipProvidedEventGroup shall not have a SomeipProvidedEventGroup.multicastThreshold attribute or shall have the SomeipProvidedEventGroup.multicastThreshold set to 0.⌋

[constr_5392] Assignment of the same event to several SomeipEventGroups on the same VLAN is forbidden if one of these SomeipEventGroups has the multicastThreshold set to a value greater than 0

Imposition time: IT_Mani

⌈The same SomeipEventDeployment (Event) shall not be referenced by several SomeipEventGroups if these SomeipEventGroups

- are referenced by different SomeipProvidedEventGroups in the context of one ServiceInstanceToMachineMapping.communicationConnector (VLAN) and
- one or more of these SomeipProvidedEventGroups has the multicast-Threshold set to a value>0

except for the case that all these SomeipProvidedEventGroups have the multicastThreshold set to the value 1.⌋

[constr_5392] is introduced to forbid scenarios in which the Service Provider is forced to transmit the same event over IP Unicast and IP Multicast in cases where the same event is assigned to several SomeipProvidedEventGroups and the multicastThreshold is reached for one SomeipProvidedEventGroup but not for the other ones.

[TPS_MANI_03020] Servers RequestResponseDelay for received SubscribeEventGroup entries ⌈The Server will delay the SubscribeEventGroupAck answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SomeipSdClientEventGroupTimingConfig.requestResponseDelay.

The actual delay will be randomly chosen between the maxValue and minValue.⌋

Class	**SomeipSdServerEventGroupTimingConfig**			
Note	EventGroup specific timing configuration settings. **Tags:** atp.recommendedPackage=SomeipSdTimingConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
request ResponseDelay	RequestResponseDelay	0..1	aggr	The Service Discovery shall delay answers to unicast messages triggered by multicast messages (e.g. Subscribe Eventgroup after Offer Service). **Tags:** atp.Splitkey=requestResponseDelay

Table 13.44: SomeipSdServerEventGroupTimingConfig

13.3.2.1.5 ProvidedSomeipServiceInstance related event and method properties

[TPS_MANI_03154] ProvidedSomeipServiceInstance related configuration settings for events ⌈The class SomeipEventProps that is aggregated by the ProvidedSomeipServiceInstance in the role eventProps allows for specifying ProvidedSomeipServiceInstance related configuration settings for events that are defined in the SomeipServiceInterfaceDeployment referenced by the ProvidedSomeipServiceInstance in the role serviceInterfaceDeployment.⌋

[TPS_MANI_03155] ProvidedSomeipServiceInstance related configuration settings for methods ⌈The class SomeipMethodProps that is aggregated by the ProvidedSomeipServiceInstance in the role methodResponseProps allows for specifying ProvidedSomeipServiceInstance related configuration settings for a method response message.

The method is defined in the SomeipServiceInterfaceDeployment referenced by the ProvidedSomeipServiceInstance in the role serviceInterfaceDeployment.⌋

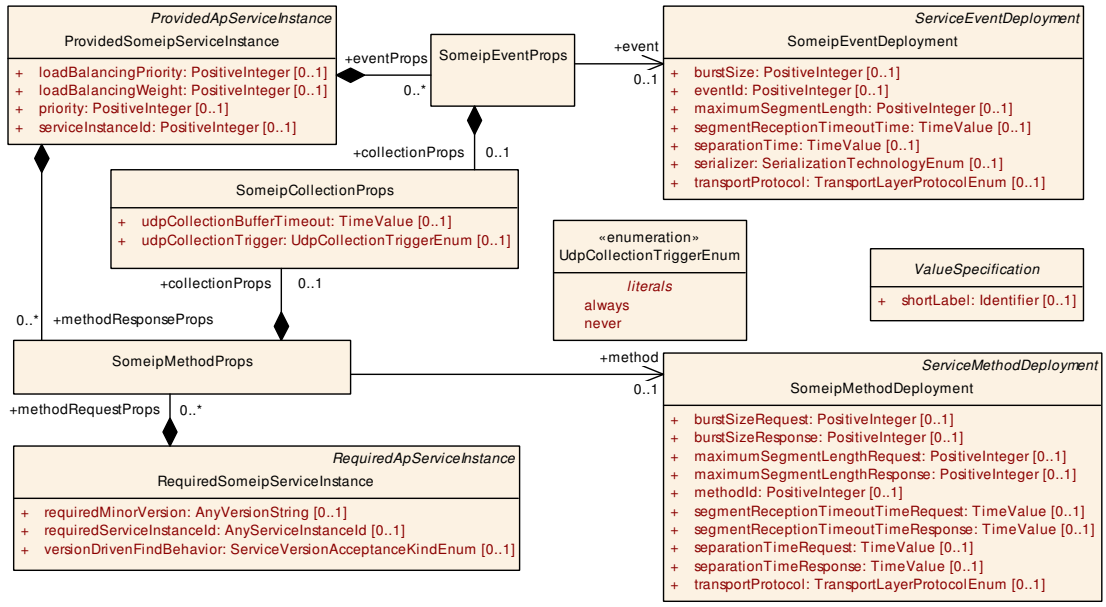


Figure 13.21: `ProvidedSomeipServiceInstance` related event and method properties

Class	**SomeipEventProps**			
Note	This meta-class allows to set configuration options for an event in the provided service instance. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	ProvidedSomeipServiceInstance.eventProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
collectionProps	SomeipCollectionProps	0..1	aggr	Collection of timing attributes configurable for an event that is provided by a Service Instance. **Tags:** atp.Splitkey=collectionProps
event	SomeipEvent Deployment	0..1	ref	Reference to the event for which the SomeipEventProps are applicable. **Tags:** atp.Splitkey=event

Table 13.45: SomeipEventProps

Class	**SomeipMethodProps**			
Note	This meta-class allows to set configuration options for a method in the service instance. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	ProvidedSomeipServiceInstance.methodResponseProps, RequiredSomeipServiceInstance.method RequestProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
collectionProps	SomeipCollectionProps	0..1	aggr	Collection of timing attributes configurable for a method that is provided or requested by a Service Instance. **Tags:** atp.Splitkey=collectionProps
method	SomeipMethod Deployment	0..1	ref	Reference to the method for which the SomeipMethod Props are applicable. **Tags:** atp.Splitkey=method

Table 13.46: SomeipMethodProps

Class	**SomeipCollectionProps**			
Note	Collection of attributes that are configurable for an event that is provided by a ServiceInstance or for a method that is provided or requested by a ServiceInstance. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	SomeipEventProps.collectionProps, SomeipMethodProps.collectionProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
udpCollection BufferTimeout	TimeValue	0..1	attr	Maximum time, an outgoing message (event, method call or method response) may be delayed, due to data collection.
udpCollection Trigger	UdpCollectionTrigger Enum	0..1	attr	Defines whether the ServiceInterface element (event or method) contributes to the triggering of the udp data transmission if data collection is enabled.

Table 13.47: SomeipCollectionProps

Enumeration	**UdpCollectionTriggerEnum**
Note	Defines whether the ServiceInterface element (event or method) contributes to the triggering of the udp data transmission if data collection is enabled. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	SomeipCollectionProps.udpCollectionTrigger
Literal	***Description***
always	ServiceInterface element will trigger the transmission of the data. **Tags:** atp.EnumerationLiteralIndex=0
never	ServiceInterface element will be buffered and will not trigger the transmission of the data. **Tags:** atp.EnumerationLiteralIndex=1

Table 13.48: UdpCollectionTriggerEnum

[TPS_MANI_03158] Configuration of a data collection on a `ProvidedSomeipServiceInstance` for transmission over udp ⌈The attributes `udpCollectionBufferTimeout` and `udpCollectionTrigger` support the configuration of a data collection of several messages for transmission over Udp.

In the `ProvidedSomeipServiceInstance` all `method` responses and `event`s for which the `udpCollectionTrigger` is set to `never` will be collected in a buffer until a trigger arrives that starts the data transmission.

The following trigger options are supported:

- a message needs to be transmitted for which the `udpCollectionTrigger` is set to `always`.
- the `udpCollectionBufferTimeout` is reached for a message.
- the buffer size defined by the attribute `udpCollectionBufferSizeThreshold` is reached.

⌋

13.3.2.2 Required Service Instance

[TPS_MANI_03059] **RequiredSomeipServiceInstance.requiredService-InstanceId** ⌈The RequiredSomeipServiceInstance defines the required-ServiceInstanceId of a SomeipServiceInterfaceDeployment that the client searches.

The client may search for a specific requiredServiceInstanceId or for ALL requiredServiceInstanceId of the serviceInterfaceDeployment.⌋

[constr_10070] **Value of RequiredSomeipServiceInstance.required-ServiceInstanceId**

Imposition time: IT_Mani

⌈For each RequiredSomeipServiceInstance.requiredServiceInstanceId, the value shall be in the range 0..65534 or ALL.⌋

Class	**RequiredSomeipServiceInstance**			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in a concrete implementation on top of SOME/IP. **Tags:** atp.recommendedPackage=ServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *RequiredApServiceInstance*, *Uploadable DesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
blocklisted Version	SomeipServiceVersion	*	aggr	Collection of blocklisted versions. **Tags:** atp.Splitkey=blocklistedVersion
capability Record (ordered)	TagWithOptionalValue	*	aggr	A sequence of records to store arbitrary name/value pairs conveying additional information about the named service.
methodRequest Props	SomeipMethodProps	*	aggr	Configuration settings for individual methods that are requested by the ServiceInstance. **Tags:** atp.Splitkey=methodRequestProps
requiredEvent Group	SomeipRequiredEvent Group	*	aggr	List of EventGroups that are used by the RequiredService Instance. **Tags:** atp.Splitkey=requiredEventGroup, requiredEvent Group.shortName
requiredMinor Version	AnyVersionString	0..1	attr	This attribute is used to configure for which minor version of the SomeIp ServiceInterface the Service Discovery will search. Value can be set to a number that represents the Minor Version of the searched service or to ANY.
requiredService InstanceId	AnyServiceInstanceId	0..1	attr	This attribute represents the ability to describe the required service instance ID.
sdClientConfig	SomeipSdClientService InstanceConfig	0..1	ref	Client specific configuration settings relevant for the SOME/IP service discovery. **Tags:** atp.Splitkey=sdClientConfig
versionDriven FindBehavior	ServiceVersion AcceptanceKindEnum	0..1	attr	Defines the service discovery find behavior.

Table 13.49: RequiredSomeipServiceInstance

[constr_10221] Multiplicity of reference in the role RequiredSomeipService-Instance.sdClientConfig

Imposition time: IT_Mani

⌈For each RequiredSomeipServiceInstance, the reference in the role sdClientConfig shall exist.⌋

Class	**SomeipServiceVersion**			
Note	This meta-class represents the ability to describe a version of a SOME/IP Service.			
Base	*ARObject*			
Aggregated by	ConsumedServiceInstance.blocklistedVersion, RequiredSomeipServiceInstance.blocklistedVersion, SomeipServiceInterfaceDeployment.serviceInterfaceVersion			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
majorVersion	PositiveInteger	0..1	attr	Major Version of the ServiceInterface. **Tags:** xml.sequenceOffset=10
minorVersion	PositiveInteger	0..1	attr	Minor Version of the ServiceInterface. **Tags:** xml.sequenceOffset=20

Table 13.50: SomeipServiceVersion

Enumeration	**ServiceVersionAcceptanceKindEnum**
Note	Defined the possible acceptance kinds for required service instances.
Aggregated by	ConsumedServiceInstance.versionDrivenFindBehavior, RequiredSomeipServiceInstance.version DrivenFindBehavior
Literal	***Description***
exactOrAnyMinor Version	Search for ANY or specific minor version service instance and select either ALL returned service instances (in case of ANY) or exactly the specific minor version service instances defined in required MinorVersion. **Tags:** atp.EnumerationLiteralIndex=0
minimumMinor Version	Search for ANY minor version service instance and select only those service instances which have an equal or greater minor version than given in requiredMinorVersion. **Tags:** atp.EnumerationLiteralIndex=1

Table 13.51: ServiceVersionAcceptanceKindEnum

[TPS_MANI_03021] Requirements on the searched minor version from the client's point of view ⌈The meta-class RequiredSomeipServiceInstance is able to make further specifications regarding the version of the service from the client's point of view.

For this purpose, the attribute RequiredSomeipServiceInstance.requiredMinorVersion exists and provides the ability to define the required minor version (SomeipServiceVersion.minorVersion).⌋

The major version that the client searches for is already defined by the SomeipServiceVersion.majorVersion in the SomeipServiceInterfaceDeployment. It is therefore not possible to search for ANY major version, so the client always looks for a specific major version.

The minor version that may be defined by SomeipServiceVersion.minorVersion in the SomeipServiceInterfaceDeployment is irrelevant for the client and the service search and shall be ignored.

[TPS_MANI_03619] SOME/IP Service search for requiredMinorVersion ⌈A RequiredSomeipServiceInstance is searching for a SOME/IP Service Instance requiredMinorVersion:

- if versionDrivenFindBehavior = exactOrAnyMinorVersion: Service minor version that matches the value set in requiredMinorVersion or *ANY* minor version of the Service Instance if the requiredMinorVersion is set to *ANY*
- if versionDrivenFindBehavior = minimumMinorVersion: Service minor version that matches at least the value set in requiredMinorVersion or is higher

⌋

[constr_3561] Restriction to minimumMinorVersion and RequiredSomeipServiceInstance.requiredMinorVersion value

Imposition time: IT_Mani

⌈The RequiredSomeipServiceInstance.requiredMinorVersion shall not have the value *ANY* if versionDrivenFindBehavior = minimumMinorVersion.⌋

[TPS_MANI_03618] Usage of attribute RequiredSomeipServiceInstance.blocklistedVersion ⌈A service connection of a RequiredSomeipServiceInstance to a ProvidedSomeipServiceInstance is not considered for service discovery if the SomeipServiceVersion.minorVersion of the enclosing SomeipServiceInterfaceDeployment that is referenced by the ProvidedSomeipServiceInstance exists in the collection of SomeipServiceVersions aggregated at the RequiredSomeipServiceInstance in the role blocklistedVersion.⌋

A typical scenario for using a blocklist may be: For a certain RequiredSomeipServiceInstance a certain compatible provider service version inside a system may not work which may have been identified after the design phase.

In order to keep the system running this certain provider version will not be considered in the service search if it has been block-listed. Therefore, the RequiredSomeipServiceInstance may connect only to ProvidedSomeipServiceInstances that fulfill the search criteria and are not block-listed.

[constr_3558] RequiredSomeipServiceInstance.blocklistedVersion is restricted to the usage of minorVersion

Imposition time: IT_Mani

⌈The majorVersion attribute shall not be used in the SomeipServiceVersion that is aggregated by the RequiredSomeipServiceInstance in the role blocklistedVersion.⌋

[constr_5115] Search for a specific SOME/IP ServiceInstance and for all SOME/IP ServiceInstances over the same `RPortPrototype`

Imposition time: `IT_Mani`

⌈A `RequiredSomeipServiceInstance` that configures the search for a specific `ServiceInstance` on SOME/IP (with concrete `requiredServiceInstanceId`) and a `RequiredSomeipServiceInstance` that configures the search for ALL `ServiceInstance`s on SOME/IP (with `requiredServiceInstanceId` = ALL) that are mapped using `ServiceInstanceToMachineMapping` to the same `EthernetCommunicationConnector` (and therefore are searching for SOME/IP `ServiceInstance`s on the same VLAN) are not allowed to be mapped by `ServiceInstanceToPortPrototypeMapping`s to the same `RPortPrototype`.⌋

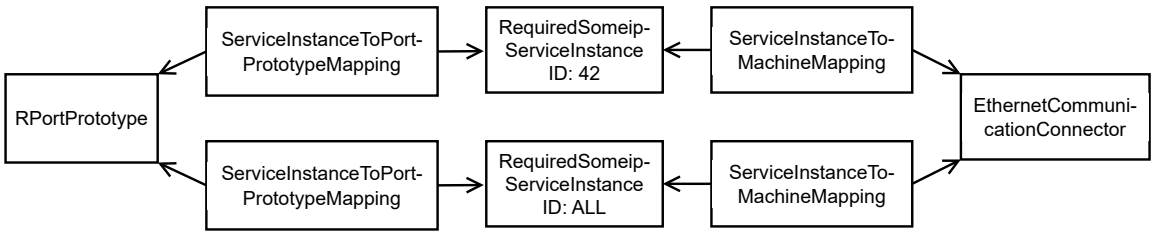


Figure 13.22: Depiction of not-allowed scenario

The depicted scenario in Figure 13.22 is covered by [constr_5115].

In other words, the middleware is allowed to search for a concrete SOME/IP `ServiceInstance` on one VLAN and for ALL SOME/IP `ServiceInstance`s on a different VLAN via a single `RPortPrototype`. But the middleware is only able to search for either ALL SOME/IP `ServiceInstance`s or for one concrete SOME/IP `ServiceInstance` on the same VLAN via a single `RPortPrototype`.

The depicted scenario in Figure 13.23 is covered by [constr_5115].

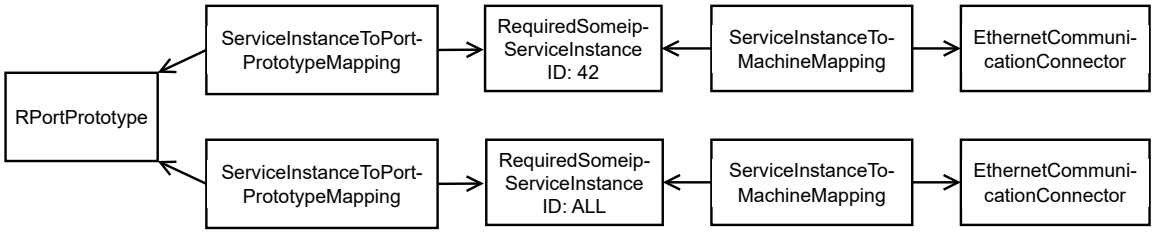


Figure 13.23: Depiction of allowed scenario

13.3.2.2.1 IP Configuration

In SOME/IP, the `SubscribeEventGroup` entry references IPv4 or IPv6 Endpoint options to indicate to the server where the client wants to receive the events of the `SomeipEventGroup`. Such an Endpoint contains the IP address of the client.

[TPS_MANI_03022] Context of meta-class RequiredSomeipServiceInstance ⌈A RequiredSomeipServiceInstance can be mapped to a CommunicationConnector of a MachineDesign with the SomeipServiceInstanceToMachineMapping.

With this mapping an assignment of the RequiredSomeipServiceInstance to a unicast IP Address is established since the EthernetCommunicationConnector refers to a NetworkEndpoint in the role unicastNetworkEndpoint. The unicastNetworkEndpoint defines the local IP address of the client.⌋

13.3.2.2.2 TP Configuration

The IPv4 or IPv6 Endpoint option that is referenced in the SOME/IP SubscribeEventGroup message contains besides the IP address the transport layer protocol (e.g. UDP or TCP), and the port number of the client.

With the SomeipServiceInstanceToMachineMapping the Transport Layer configuration attributes are assigned to the RequiredSomeipServiceInstance.

The Transport Layer (TCP/UDP) configuration attributes for the SubscribeEventGroup entry are directly available in the SomeipServiceInstanceToMachineMapping element.

The SomeipServiceInstanceToMachineMapping defines also the source-port where the client sends the method call messages to the server and the destination-port where the client receives the method responses from the server.

The modeling of the SomeipServiceInstanceToMachineMapping with TP and IP configuration is depicted in Figure D.1.

[TPS_MANI_03023] Udp Transport Protocol Configuration for RequiredSomeipServiceInstance ⌈The SomeipServiceInstanceToMachineMapping.udpPort defines the Transport Protocol for a UDP communication if the server provides ServiceInterface content over UDP and the client wants to use it.⌋

[TPS_MANI_03024] Tcp Transport Protocol Configuration for RequiredSomeipServiceInstance ⌈The SomeipServiceInstanceToMachineMapping.tcpPort defines the Transport Protocol for a TCP communication if the server provides ServiceInterface content over TCP and the client wants to use it.⌋

[TPS_MANI_03049] Tcp and Udp Transport Protocol Configuration for RequiredSomeipServiceInstance ⌈It is allowed to set tcpPort and udpPort in the same SomeipServiceInstanceToMachineMapping. Such a setting shall be used if the server provides ServiceInterface content over Udp and Tcp and the client wants to use it.⌋

[TPS_MANI_03237] Transport Protocol attributes defined for a RequiredSomeipServiceInstance ⌈Each SomeipServiceInstanceToMachineMapping that is defined for a RequiredSomeipServiceInstance is allowed to have:

- a configured udpPort or
- a configured tcpPort or
- a configured udpPort and a tcpPort or
- no configured udpPort and tcpPort.

⌋

A RequiredSomeipServiceInstance that is mapped by a SomeipServiceInstanceToMachineMapping that does not contain a udpPort and tcpPort is allowed to receive events over IP Multicast only.

In this case, it is not required for a client to have a unicast socket prepared if the server will always use the multicast transport. In such a case, the SubscribeEventGroup SOME/IP SD message that is sent from the ServiceConsumer to the ServiceProvider will not contain any Unicast Endpoint options.

In addition, the corresponding ServiceInterface is not allowed to have any methods defined since the Request/Response communication pattern is restricted to IP Unicast only. It is also not allowed to have any Fields defined, since field getters and setters are methods, and field notifiers shall always send their initial value using Unicast (see [PRS_SOMEIPSD_00487]).

An additional prerequisite for such a setup is that the ProvidedSomeipServiceInstance is configured for IP multicast transmission only. In other words, the value of attribute multicastThreshold for all SomeipProvidedEventGroups needs to be set to 1.

[constr_5161] RequiredSomeipServiceInstance that is mapped by a SomeipServiceInstanceToMachineMapping without a configured tcpPort and udpPort

Imposition time: IT_Mani

⌈A RequiredSomeipServiceInstance that is mapped to an EthernetCommunicationConnector by a SomeipServiceInstanceToMachineMapping that has no references in the roles

- udpPort or
- tcpPort

is not allowed to reference a SomeipServiceInterfaceDeployment that includes SomeipMethodDeployments and/or ServiceFieldDeployments.⌋

If a Tcp and Udp Transport Protocol Configuration is defined for a RequiredSomeipServiceInstance as described in [TPS_MANI_03049] then the SOME/IP ServiceInterfaceDeployment settings decide which content of the ProvidedSomeipServiceInstance is transported over udp and which content is transported over tcp. This is described in [TPS_MANI_03050] and [TPS_MANI_03051].

13.3.2.2.3 Service Discovery Client Configuration

Service Discovery phases on the Client side allow minimizing the number of Service Discovery messages and allow a fast synchronization upon ECU start.

For every RequiredSomeipServiceInstance on a Client different phases are existing:

- Down
- Requested
  - Initial Wait Phase
  - Repetition Phase
  - Main Phase

[TPS_MANI_03025] Client Timing configuration for a RequiredSomeipServiceInstance ⌈The Client Timing is configurable with SomeipSdClientServiceInstanceConfig that is referenced in the role sdClientConfig by the RequiredSomeipServiceInstance for which the Timing is valid.⌋

The number of SomeipSdClientServiceInstanceConfig elements determines how many timers shall actually be used by the middleware to keep the randomized times. Via the reference RequiredSomeipServiceInstance.sdClientConfig, each RequiredSomeipServiceInstance defines to which timer it is assigned.

Note that it is possible to define several SomeipSdClientServiceInstanceConfig elements with identical timing specification values in order to request several timer handling in the middleware.

If several RequiredSomeipServiceInstance share the same timer then the expiration of that timer will lead to a combined sending of service discovery messages.

[TPS_MANI_03231] Sharing timers for RequiredSomeipServiceInstance ⌈If several RequiredSomeipServiceInstances point to the same SomeipSdClientServiceInstanceConfig in the role sdClientConfig then all of these RequiredSomeipServiceInstances will share the same timers for their timing behavior. This will lead to combining several service discovery entries in one service discovery message.⌋

Class	**SomeipSdClientServiceInstanceConfig**			
Note	Client specific settings that are relevant for the configuration of SOME/IP Service-Discovery. **Tags:** atp.recommendedPackage=SomeipSdTimingConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initialFind Behavior	InitialSdDelayConfig	0..1	aggr	Controls initial find behavior of clients. **Tags:** atp.Splitkey=initialFindBehavior
priority	PositiveInteger	0..1	attr	This attribute defines the VLAN frame priority for Service Discovery messages that result from RequiredSomeip ServiceInstances that are referncing this SomeipSdClient ServiceInstanceConfig (Find, SubscribeEventGroup, Stop SubscribeEventgroup). Values from 0 (best effort) to 7 (highest) are allowed.

Table 13.52: SomeipSdClientServiceInstanceConfig

[TPS_MANI_03026] Initial Wait Phase configuration for a `RequiredSomeip-ServiceInstance` ⌈The Initial Wait Phase for a `RequiredSomeipService-Instance` is configured with the `initialFindBehavior` and the two attributes `initialDelayMinValue` and `initialDelayMaxValue`.

If a calculated random timer based on these min and max values expires the first `FindService` entry will be sent out.⌋

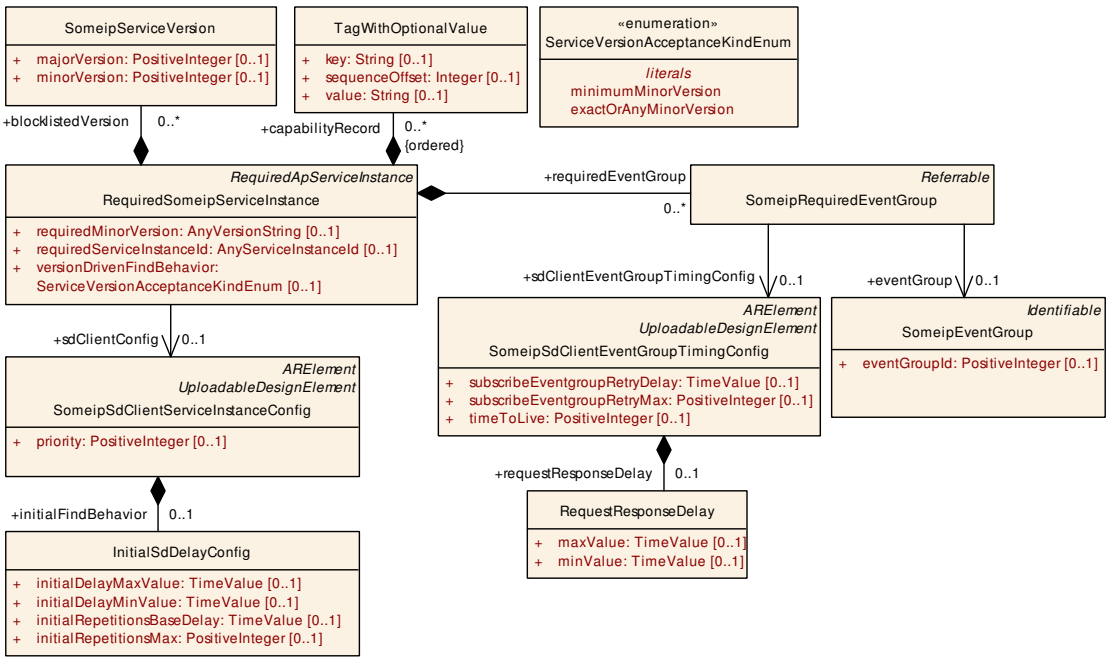


Figure 13.24: SOME/IP Service Discovery Client configuration settings

When the calculated random timer expires and no `OfferService` is received the Repetition Phase will be entered.

[TPS_MANI_03027] Repetition Phase configuration for a `RequiredSomeipServiceInstance` ⌈The Repetition Phase for a `RequiredSomeipServiceInstance` is configured with the `initialFindBehavior` and the two attributes `initialRepetitionsMax` and `initialRepetitionsBaseDelay`.⌋

If the Repetition Phase is entered, the Service Discovery waits for the `initialRepetitionsBaseDelay` and sends an `FindService` entry.

If the amount of sent `FindService` entries reaches `initialRepetitionsMax` and no `OfferService` is received the Main Phase will be entered. In the Main Phase no further `FindService` entries are sent by the client.

SOME/IP allows for specifying additional information about the `RequiredSomeipServiceInstance` with the Capability Record that allows transporting arbitrary configuration strings (key/value pairs).

This allows encoding additional information like the name of a service or its configuration.

Figure 13.25 shows an example of the different SOME/IP phases on the Client side.

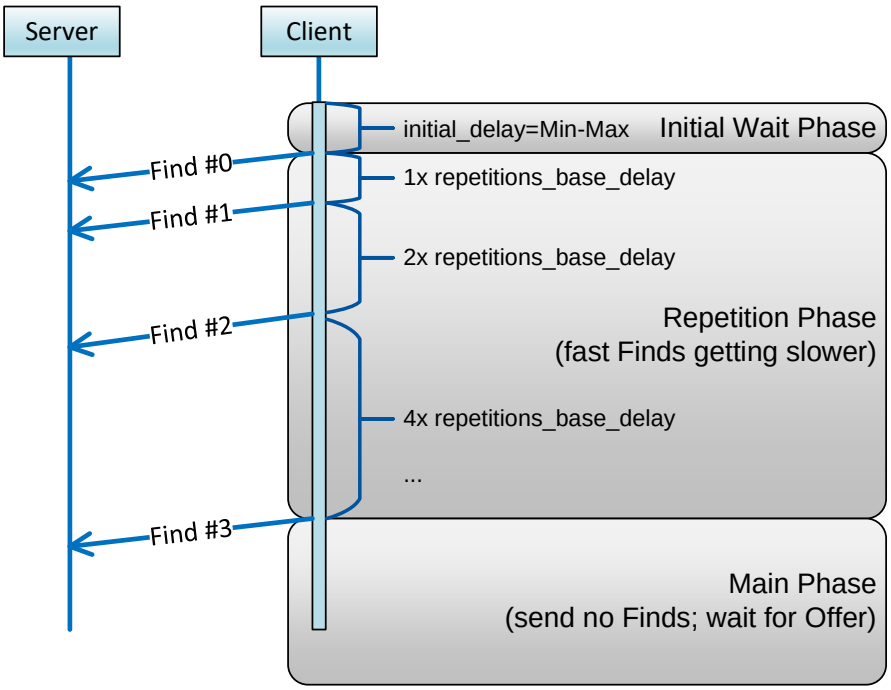


Figure 13.25: SOME/IP Client Timing example

[TPS_MANI_03029] Client Capability Records ⌈A Capability Record (key/value pair) on the Client side is configurable with the `capabilityRecord` and the two attributes `key` and `value`.⌋

13.3.2.2.4 Required Event Group

The RequiredSomeipServiceInstance aggregates a SomeipRequiredEvent-Group in the role requiredEventGroup that allows defining service instance specific configuration settings for a SomeipEventGroup.

Class	SomeipRequiredEventGroup			
Note	The meta-class represents the ability to configure ServiceInstance related communication settings on the required side for each EventGroup separately. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Referrable*			
Aggregated by	RequiredSomeipServiceInstance.requiredEventGroup			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
eventGroup	SomeipEventGroup	0..1	ref	Reference to the SomeipEventGroup in the System Manifest for which the ServiceInstance related Event Group settings are valid. **Tags:** atp.Splitkey=eventGroup
sdClientEvent GroupTiming Config	SomeipSdClientEvent GroupTimingConfig	0..1	ref	Client Timing configuration settings that are EventGroup specific. **Tags:** atp.Splitkey=sdClientEventGroupTimingConfig

Table 13.53: SomeipRequiredEventGroup

[constr_10222] Multiplicity of the reference in the role SomeipRequiredEvent-Group.sdClientEventGroupTimingConfig

Imposition time: IT_Mani

⌈For each SomeipRequiredEventGroup, the reference in the role sdClient-EventGroupTimingConfig shall exist.⌋

Class	SomeipSdClientEventGroupTimingConfig			
Note	This meta-class is used to specify configuration related to service discovery in the context of an event group on SOME/IP. **Tags:** atp.recommendedPackage=SomeipSdTimingConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
request ResponseDelay	RequestResponseDelay	0..1	aggr	The Service Discovery shall delay answers to unicast messages triggered by multicast messages (e.g. Subscribe Eventgroup after Offer Service). **Tags:** atp.Splitkey=requestResponseDelay
subscribe Eventgroup RetryDelay	TimeValue	0..1	attr	This attribute defines the interval in seconds to re-trigger a subscription to a Eventgroup, if a retry to subscribe to a Eventgroup is configured (subscribeEventgroupRetryMax > 0).
subscribe Eventgroup RetryMax	PositiveInteger	0..1	attr	This attribute define the maximum counts of retries to subscribe to an Eventgroup. If the value is set to 0 no retry shall be done. If the value is set to 255 the retry shall be done as along as the Eventgroup is requested and no SubscribeEventGroupAck was received.

▽

△

Class	SomeipSdClientEventGroupTimingConfig			
timeToLive	PositiveInteger	0..1	attr	Defines the time in seconds the subscription of this event is expected by the client. this value is sent from the client to the server in the SD-subscribeEvent message.

Table 13.54: SomeipSdClientEventGroupTimingConfig

[TPS_MANI_03030] **`SomeipSdClientEventGroupTimingConfig.timeToLive` for `SubscribeEventGroup` Entries** ⌈The lifetime of an event subscription is configurable with the `timeToLive` attribute of `SomeipSdClientEventGroupTimingConfig`.

If the time that is configured by `timeToLive` expires, the event subscription is canceled.⌋

[TPS_MANI_03031] **Clients `RequestResponseDelay` for received `ServiceOffer` entries** ⌈The Client will delay the `SubscribeEventGroup` answer to a received `ServiceOffer` message by the configured `SomeipSdClientEventGroupTimingConfig.requestResponseDelay`.

The actual delay will be randomly chosen between the `maxValue` and `minValue`.⌋

13.3.2.2.5 `RequiredSomeipServiceInstance` related method call properties

[TPS_MANI_03156] **`RequiredSomeipServiceInstance` related configuration settings for `methods`** ⌈The class `SomeipMethodProps` that is aggregated by the `RequiredSomeipServiceInstance` in the role `methodRequestProps` allows specifying `RequiredSomeipServiceInstance` related configuration settings for a `method` request message. The `method` is defined in the `SomeipServiceInterfaceDeployment` referenced by the `RequiredSomeipServiceInstance` in the role `serviceInterfaceDeployment`.⌋

[TPS_MANI_03159] **Configuration of a data collection on a `RequiredSomeipServiceInstance` for transmission over UDP** ⌈The attributes `udpCollectionBufferTimeout` and `udpCollectionTrigger` support the configuration of a data collection of several messages for transmission over UDP. In the `RequiredSomeipServiceInstance` all `method` requests for which the `udpCollectionTrigger` is set to `never` will be collected in a buffer until a trigger arrives that starts the data transmission.

The following trigger options are supported:

- a message needs to be transmitted for which the `udpCollectionTrigger` is set to `always`.
- the `udpCollectionBufferTimeout` is reached for a message.

- the buffer size defined by the attribute udpCollectionBufferSize-Threshold is reached.

⌋

The modeling of the RequiredSomeipServiceInstance-related event-properties and method-properties is depicted in Figure 13.21.

13.3.2.3 Provided Service Instance with static remote peers

[TPS_MANI_03312] Static configuration of remote peer addresses for a ProvidedSomeipServiceInstance ⌈AUTOSAR supports a static configuration of remote peers for a ProvidedSomeipServiceInstance that is mapped with a SomeipServiceInstanceToMachineMapping to an ApApplicationEndpoint with the remoteUnicastConfig.⌋

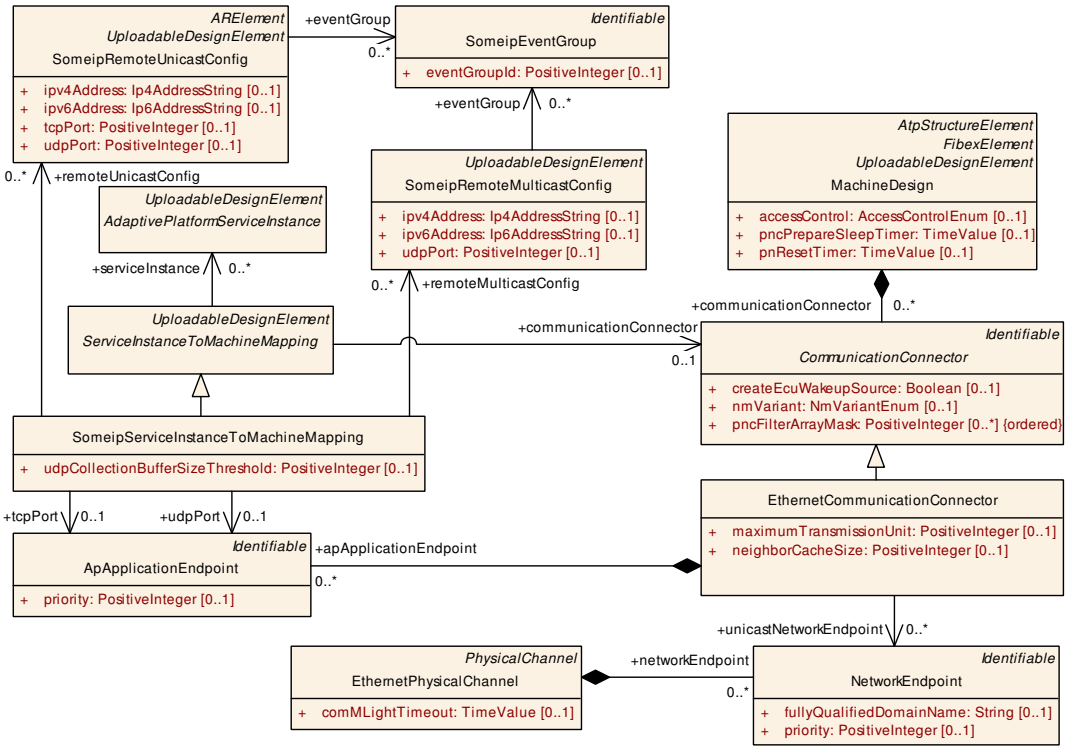


Figure 13.26: SomeipServiceInstanceToMachineMapping with static configured remote peers

If such SomeipRemoteUnicastConfig is referenced by the SomeipService-InstanceToMachineMapping then the Service Discovery mechanism is not used and the address of the remote peers is retrieved from the configuration.

This allows establishing a communication connection between the local address (defined by the ApApplicationEndpoint and the NetworkEndpoint) and the configured remote address that is defined by the referenced SomeipRemoteUnicastConfig.

A collection of remote peers can be defined if the SomeipServiceInstanceToMachineMapping references a ProvidedSomeipServiceInstance in the role serviceInstance.

In this case a communication connection to each of the configured remote peer addresses will be established. The ProvidedSomeipServiceInstance will transmit event and field notifications to the address that is defined in the SomeipRemoteUnicastConfig.

Class	**SomeipRemoteUnicastConfig**			
Note	This meta-class is used to statically configure the remote peer's unicast address in case that a static service connection is used and only a single remote peer exists. **Tags:** atp.recommendedPackage=SomeipRemoteUnicastConfigs This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventGroup	SomeipEventGroup	*	ref	Reference to the SomeipEventGroups this Someip RemoteUnicatConfig applies to. **Tags:** atp.Splitkey=eventGroup
ipv4Address	Ip4AddressString	0..1	attr	This attribute defines the IPv4 address of the remote peer to allow a static service connection between Service Provider and Service Consumer.
ipv6Address	Ip6AddressString	0..1	attr	This attribute defines the IPv6 address of the remote peer to allow a static service connection between Service Provider and Service Consumer.
tcpPort	PositiveInteger	0..1	attr	This attribute defines the tcpPort of the remote peer to allow a static service connection between Service Provider and Service Consumer.
udpPort	PositiveInteger	0..1	attr	This attribute defines the udpPort of the remote peer to allow a static service connection between Service Provider and Service Consumer.

Table 13.55: SomeipRemoteUnicastConfig

[constr_5355] SomeipServiceInstanceToMachineMapping with configured remote peer addresses shall not mix ProvidedSomeipServiceInstances and RequiredSomeipServiceInstances

Imposition time: IT_Mani

⌈A SomeipServiceInstanceToMachineMapping that contains a reference to a SomeipRemoteUnicastConfig with the remoteUnicastConfig shall not reference:

- ProvidedSomeipServiceInstances in the role serviceInstance and
- RequiredSomeipServiceInstances in the role serviceInstance at the same time.

⌋

[TPS_MANI_03313] Semantics of `SomeipRemoteUnicastConfig.eventGroup` reference ⌈For each remote peer that is specified with the `SomeipRemoteUnicastConfig` it is possible to statically define the `SomeipEventGroup`s that the remote peer is interested in with the `SomeipRemoteUnicastConfig.eventGroup` reference.⌋

With the `SomeipRemoteUnicastConfig.eventGroup` information each service provider has the knowledge from the configuration to which service consumers which event notifications need to be transmitted.

An example for the static configuration of a `ProvidedSomeipServiceInstance` with remote peers can be found in Chapter B.2.6.1.

13.3.2.4 Required Service Instance with static remote peers

[TPS_MANI_03314] Static configuration of a remote peer address for a `RequiredSomeipServiceInstance` ⌈AUTOSAR supports a static configuration of a remote peer for a `RequiredSomeipServiceInstance` that is mapped with a `SomeipServiceInstanceToMachineMapping` to an `ApApplicationEndpoint` with the `remoteUnicastConfig`.⌋

With the static configuration of a remote peer the `RequiredSomeipServiceInstance` knows the address of the `ProvidedSomeipServiceInstance` and is able to call methods of the provided service.

[constr_5356] `RequiredSomeipServiceInstance` is allowed to have only a single statically configured remote peer as service provider

Imposition time: `IT_Mani`

⌈A `SomeipServiceInstanceToMachineMapping` that contains references to a `RequiredSomeipServiceInstance` with the `serviceInstance` is allowed to reference only a single `SomeipRemoteUnicastConfig` in the role `remoteUnicastConfig`.⌋

Since the service provider may transport event notification via IP Multicast, the service consumer needs to know the IP Multicast groups it needs to join:

[TPS_MANI_03315] Semantics of `SomeipRemoteMulticastConfig` ⌈It is possible to statically define the IP Multicast addresses that the `RequiredSomeipServiceInstance`s shall listen to with the `SomeipRemoteMulticastConfig` element that is referenced in the `remoteMulticastConfig` role by the `SomeipServiceInstanceToMachineMapping` that references the `RequiredSomeipServiceInstance`s in the role `serviceInstance`.⌋

Class	**SomeipRemoteMulticastConfig**			
Note	This meta-class is used to statically configure the remote peer's multicast address. **Tags:** atp.recommendedPackage=RemoteMulticastConfigs This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventGroup	SomeipEventGroup	*	ref	Reference to the SomeipEventGroups this Someip RemoteMulticastConfig applies to. **Tags:** atp.Splitkey=eventGroup
ipv4Address	Ip4AddressString	0..1	attr	This attribute defines the multicast IPv4 address to allow a static service connection between Service Provider and Service Consumers.
ipv6Address	Ip6AddressString	0..1	attr	This attribute defines the multicast IPv6 address to allow a static service connection between Service Provider and Service Consumers.
udpPort	PositiveInteger	0..1	attr	This attribute defines the udpPort used for the multicast communication.

Table 13.56: SomeipRemoteMulticastConfig

[TPS_MANI_03316] Semantics of `SomeipRemoteMulticastConfig.eventGroup` ⌈For each Multicast group that is specified with the `SomeipRemoteMulticastConfig`, it is possible to statically define the `eventGroup`s that include events that may be transmitted from the configured remote peer to the configured IP Multicast address.⌋

[constr_5357] `SomeipRemoteMulticastConfig` shall only be used on required side

Imposition time: `IT_Mani`

⌈Only a `SomeipServiceInstanceToMachineMapping` that contains references to one or several `RequiredSomeipServiceInstance`s with the `serviceInstance` role is allowed to reference one or several `SomeipRemoteMulticastConfig`s in the role `remoteUnicastConfig`.⌋

An example for the static configuration of a `RequiredSomeipServiceInstance` with remote peers can be found in Chapter B.2.6.2.

13.3.3 DDS Service Instance Deployment

In the case of DDS used as the transport layer the derived meta-classes are `DdsProvidedServiceInstance` or `DdsRequiredServiceInstance`. These meta-classes also carry attributes that apply for the service discovery on DDS.

Two discovery protocols are supported by the DDS Network binding:

- Service Instance announcement via purpose-specific formatting of the Domain Participant USER_DATA QoS policy

- Service Instance announcement through a purpose-specific discovery Topic.

For details on these two protocols and their implications, see section 7.7.3 “DDS Network Binding” in [32, SWS Communication Management].

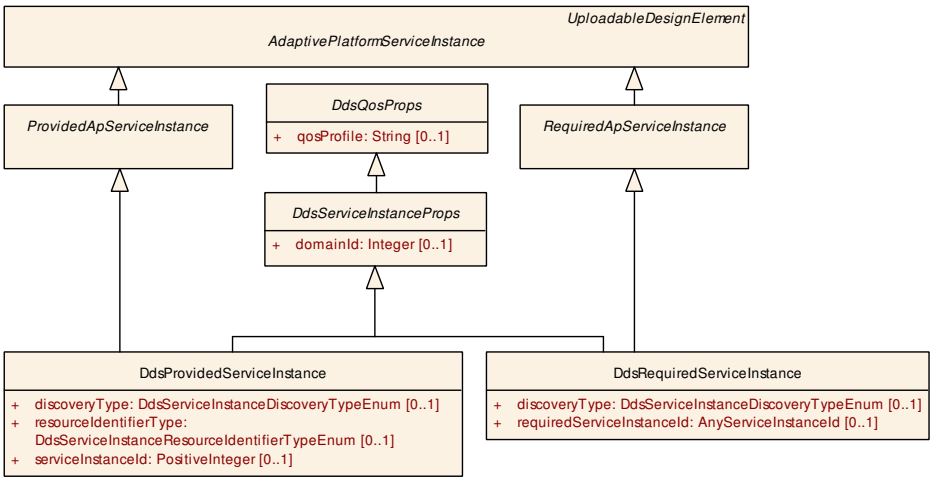


Figure 13.27: DDS Service Instances

Class	***DdsQosProps*** (abstract)			
Note	QoS configuration properties for the DDS entities associated with an event, method, or field provided by or requested from a Service Instance using DDS as the underlying network binding. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Subclasses	DdsEventQosProps, DdsFieldQosProps, *DdsServiceInstanceProps*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
qosProfile	String	0..1	attr	Identifies a group of QoS Policies that apply to the DDS entities associated with the event, method, field, or the service instance.

Table 13.57: DdsQosProps

Class	***DdsServiceInstanceProps*** (abstract)			
Note	Common configuration properties for the DDS entities provided by or requested from a Service Instance using DDS as the underlying network binding. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *DdsQosProps*			
Subclasses	DdsProvidedServiceInstance, DdsRequiredServiceInstance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
domainId	Integer	0..1	attr	This attribute identifies the DDS Domain the Service Instance shall join.

Table 13.58: DdsServiceInstanceProps

[constr_10223] Multiplicity of attribute `DdsServiceInstanceProps.domainId`

Imposition time: `IT_Mani`

⌈For each `DdsServiceInstanceProps`, the attribute `domainId` shall exist.⌋

13.3.3.1 Provided DDS Service Instance

[TPS_MANI_03527] Definition of DdsProvidedServiceInstance ⌈The DdsProvidedServiceInstance configures the Service to join a DDS Domain with the domainId attribute, and to instantiate the underlying DDS entities according to a QoS profile with the qosProfile attribute, using the discovery protocol defined by discoveryType and resource identification scheme defined by the resourceIdentifierType attribute.

Moreover, it assigns an Instance ID to the Service for deployment with the serviceInstanceId attribute.⌋

[TPS_MANI_03650] Definition of DdsServiceInstanceResourceIdentifierTypeEnum ⌈The resource identification schemes enumerated by DdsServiceInstanceResourceIdentifierTypeEnum in DdsProvidedServiceInstance.resourceIdentifierType attribute define the various combinations of DDS features through which Provided Service Interface Instances can advertise, be bound, and communicate on a given DDS domain, as defined by section 7.7.3 “DDS Network Binding” of [32, SWS Communication Management].⌋

[constr_3528] Value range of DdsServiceInstanceProps.domainId

Imposition time: IT_Mani

⌈The value of attribute DdsServiceInstanceProps.domainId at DdsProvidedServiceInstance and domainId at DdsRequiredServiceInstance shall be in the range of a signed 32-bit integer.⌋

[constr_3529] Value range of DdsProvidedServiceInstance.serviceInstanceId

Imposition time: IT_Mani

⌈The value of attribute DdsProvidedServiceInstance.serviceInstanceId shall be in the range of 0..65535.⌋

[constr_3541] qosProfile mandatory for DdsProvidedServiceInstance

Imposition time: IT_Mani

⌈The attribute qosProfile shall be defined for every DdsProvidedServiceInstance.⌋

[constr_3645] discoveryType mandatory for DdsProvidedServiceInstance

Imposition time: IT_Mani

⌈The attribute discoveryType shall be defined for every DdsProvidedServiceInstance.⌋

[constr_3646] `resourceIdentifierType` mandatory for `DdsProvidedServiceInstance`

Imposition time: `IT_Mani`

⌈The attribute `resourceIdentifierType` shall be defined for every `DdsProvidedServiceInstance`.⌋

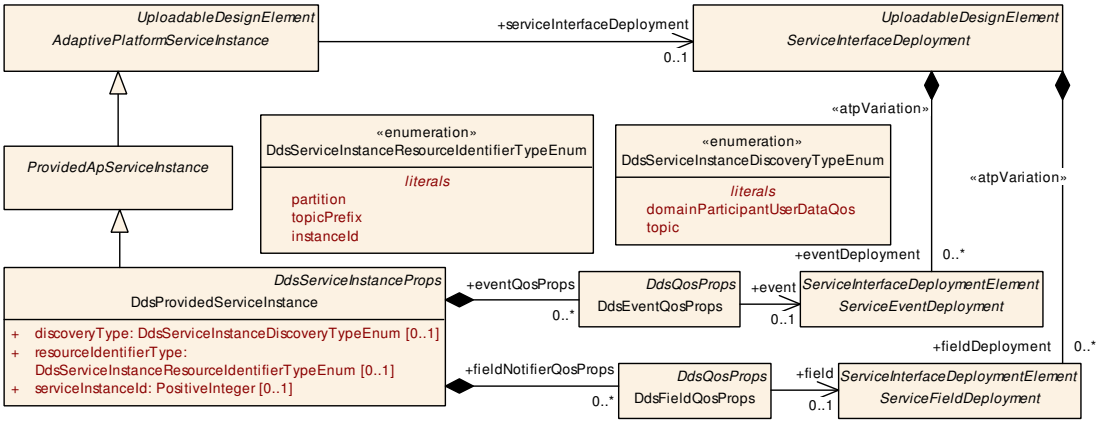


Figure 13.28: Provided DDS Service Instances

[constr_3647] `resourceIdentifierType` value for USER_DATA QoS-based discovery

Imposition time: `IT_Mani`

⌈If the value of `discoveryType` is `domainParticipantUserDataQos`, for a given `DdsProvidedServiceInstance`, the only valid value for attribute `resourceIdentifierType` is `partition`.⌋

[constr_3564] Consistency between DDS Service Interface Deployment and Provided DDS Service Instance

Imposition time: `IT_Mani`

⌈Transport attributes `DdsServiceInterfaceDeployment.transportProtocol` and `DdsEventDeployment.transportProtocol` shall be consistent with DDS profiles generated and selected by the `DdsQosProps` component of `DdsProvidedServiceInstance`, `DdsFieldQosProps`, and `DdsEventQosProps`.⌋

Class	**DdsProvidedServiceInstance**
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in a concrete implementation on top of DDS. **Tags:** atp.recommendedPackage=ServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *DdsQosProps*, *DdsServiceInstanceProps*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *ProvidedApServiceInstance*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*
Aggregated by	ARPackage.element

△

Class	**DdsProvidedServiceInstance**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
discoveryType	DdsServiceInstance DiscoveryTypeEnum	0..1	attr	Discovery protocol.
eventQosProps	DdsEventQosProps	*	aggr	List of configuration properties for the Events that are provided by the Service Instance. **Tags:** atp.Splitkey=eventQosProps
fieldNotifierQos Props	DdsFieldQosProps	*	aggr	List of configuration properties for Field notifiers that are provided by the Service Instance. **Tags:** atp.Splitkey=fieldNotifierQosProps
resource IdentifierType	DdsServiceInstance ResourceIdentifierType Enum	0..1	attr	Type of resource identification scheme.
serviceInstance Id	PositiveInteger	0..1	attr	Identification number that is used by DDS to identify DomainParticipants associated with an instance of the service.

Table 13.59: DdsProvidedServiceInstance

Enumeration	**DdsServiceInstanceDiscoveryTypeEnum**
Note	Supported discovery schemes for DDS Service Instances. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	DdsProvidedServiceInstance.discoveryType, DdsRequiredServiceInstance.discoveryType
Literal	***Description***
domainParticipant UserDataQos	The USER_DATA QoS policy is used to advertise and discover available Service Instances hosted by each Domain Participant. **Tags:** atp.EnumerationLiteralIndex=0
topic	A purpose-specific Topic is used to convey availability of Service Instances and how to bind against them. **Tags:** atp.EnumerationLiteralIndex=1

Table 13.60: DdsServiceInstanceDiscoveryTypeEnum

Enumeration	**DdsServiceInstanceResourceIdentifierTypeEnum**
Note	Supported Resource Identification schemes for DDS Service Instances. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	DdsProvidedServiceInstance.resourceIdentifierType
Literal	***Description***
instanceId	In-band instance identification fields are used to discriminate samples related to specific Service Instances sharing the same DDS Topics • Partitions: - • Topics: ara.com://services/<InterfaceId>/<Major>.<<Minor>/<TopicName> **Tags:** atp.EnumerationLiteralIndex=2
partition	The DDS PARTITION QoS policy is used to isolate DDS Topics related to specific Service Instances • Partitions: ara.com://services/<InterfaceId>/<InstanceId> • Topics: ara.com://services/<InterfaceId>/<Major>.<Minor>/<TopicName> **Tags:** atp.EnumerationLiteralIndex=0

▽

△

Enumeration	**DdsServiceInstanceResourceIdentifierTypeEnum**
topicPrefix	Unique prefixes are assigned to DDS Topics related to specific Service Instances • Partitions: - • Topics: ara.com://services/<InterfaceId>/<InstanceId>/<TopicName> **Tags:** atp.EnumerationLiteralIndex=1

Table 13.61: DdsServiceInstanceResourceIdentifierTypeEnum

[TPS_MANI_03528] Definition of DdsProvidedServiceInstance.eventQosProps ⌈The DdsProvidedServiceInstance.eventQosProps configures the DDS entities associated with the event according to a QoS Profile specified with the qosProfile attribute.⌋

[TPS_MANI_03531] qosProfile of DdsProvidedServiceInstance.eventQosProps is optional ⌈The attribute qosProfile of DdsProvidedServiceInstance.eventQosProps is optional; if qosProfile is not defined, the underlying DDS entities shall be configured according to the qosProfile attribute of the parent DdsProvidedServiceInstance.⌋

[TPS_MANI_03561] Definition of DdsProvidedServiceInstance.fieldNotifierQosProps ⌈The DdsProvidedServiceInstance.fieldNotifierQosProps configures the DDS entities associated with the field according to a QoS Profile specified with the qosProfile attribute.⌋

Class	**DdsEventQosProps**			
Note	Configuration properties of the Event using DDS as the underlying network binding. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *DdsQosProps*			
Aggregated by	DdsProvidedServiceInstance.eventQosProps, DdsRequiredServiceInstance.eventQosProps			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
event	ServiceEvent Deployment	0..1	ref	Reference to an event that is provided. **Tags:** atp.Splitkey=event

Table 13.62: DdsEventQosProps

[constr_10224] Multiplicity of reference in the role DdsEventQosProps.event

Imposition time: IT_Mani

⌈For each DdsEventQosProps, the reference in the role event shall exist.⌋

[TPS_MANI_03562] qosProfile of DdsProvidedServiceInstance.fieldNotifierQosProps is optional ⌈The attribute qosProfile of DdsProvidedServiceInstance.fieldNotifierQosProps is optional; if qosProfile is not defined, the underlying DDS entities shall be configured according to the qosProfile attribute of the parent DdsProvidedServiceInstance.⌋

Class	**DdsFieldQosProps**			
Note	Configuration properties of the Field interaction when using DDS as the underlying network binding. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *DdsQosProps*			
Aggregated by	DdsProvidedServiceInstance.fieldNotifierQosProps, DdsRequiredServiceInstance.fieldNotifierQosProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
field	ServiceField Deployment	0..1	ref	Reference to the field. **Tags:** atp.Splitkey=field

Table 13.63: DdsFieldQosProps

[constr_10225] Multiplicity of reference in the role `DdsFieldQosProps.field`

Imposition time: `IT_Mani`

⌈For each `DdsFieldQosProps`, the reference in the role `field` shall exist.⌋

13.3.3.2 Required DDS Service Instance

[TPS_MANI_03529] Definition of `DdsRequiredServiceInstance` ⌈The `DdsRequiredServiceInstance` configures the Client to join a DDS Domain with the `domainId` attribute, and to instantiate the underlying DDS entities according to a QoS Profile with the `qosProfile` attribute, using the discovery protocol defined by `discoveryType`. Optionally, the `requiredServiceInstanceId` attribute allows a Client to search for a specific Instance ID of the serviceInterface.⌋

[constr_3542] `qosProfile` mandatory for `DdsRequiredServiceInstance`

Imposition time: `IT_Mani`

⌈The attribute `qosProfile` shall be defined for every `DdsRequiredServiceInstance`.⌋

[constr_3648] `discoveryType` mandatory for `DdsRequiredServiceInstance`

Imposition time: `IT_Mani`

⌈The attribute `discoveryType` shall be defined for every `DdsRequiredServiceInstance`.⌋

[constr_3565] Consistency between DDS Service Interface Deployment and Required DDS Service Instance

Imposition time: `IT_Mani`

⌈Transport attributes `DdsServiceInterfaceDeployment.transportProtocol` and `DdsEventDeployment.transportProtocol` shall be consistent with DDS profiles generated and selected by the `DdsQosProps` component of `DdsRequiredServiceInstance`, `DdsFieldQosProps`, and `DdsEventQosProps`.⌋

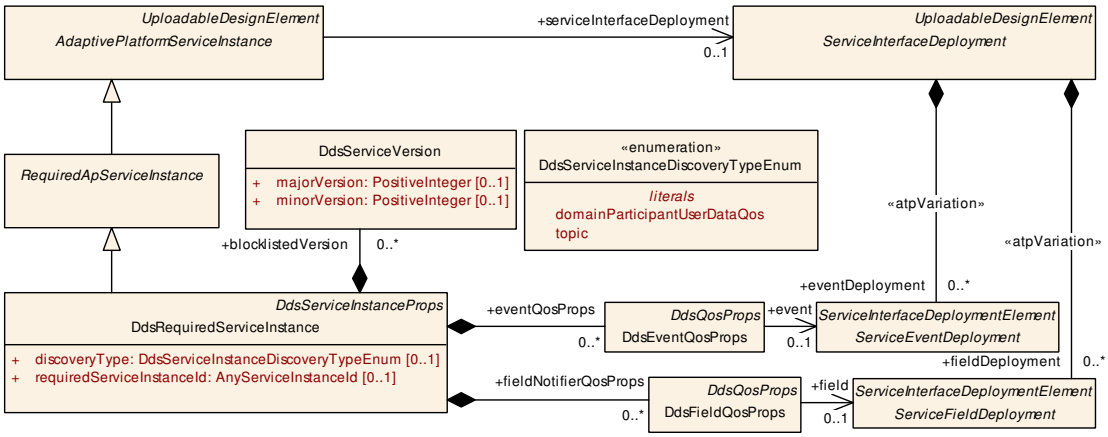


Figure 13.29: Required DDS Service Instances

Class	**DdsRequiredServiceInstance**			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in a concrete implementation on top of DDS. **Tags:** atp.recommendedPackage=ServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *DdsQosProps*, *DdsServiceInstanceProps*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *RequiredApServiceInstance*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
blocklisted Version	DdsServiceVersion	*	aggr	Collection of blocklisted versions. **Tags:** atp.Splitkey=blocklistedVersion
discoveryType	DdsServiceInstance DiscoveryTypeEnum	0..1	attr	Discovery protocol.
eventQosProps	DdsEventQosProps	*	aggr	List of configuration properties for the Events that are required by the Service Instance. **Tags:** atp.Splitkey=eventQosProps
fieldNotifierQos Props	DdsFieldQosProps	*	aggr	List of configuration properties for Field notifiers that are required by the Service Instance. **Tags:** atp.Splitkey=fieldNotifierQosProps
requiredService InstanceId	AnyServiceInstanceId	0..1	attr	This attribute represents the ability to describe the required service instance ID.

Table 13.64: DdsRequiredServiceInstance

Class	**DdsServiceVersion**			
Note	Definition of a Dds service version. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	DdsRequiredServiceInstance.blocklistedVersion			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
majorVersion	PositiveInteger	0..1	attr	Service major version. **Tags:** xml.sequenceOffset=10
minorVersion	PositiveInteger	0..1	attr	Service minor version. **Tags:** xml.sequenceOffset=20

Table 13.65: DdsServiceVersion

[TPS_MANI_03530] Definition of DdsRequiredServiceInstance.eventQosProps ⌈The DdsRequiredServiceInstance.eventQosProps configures the DDS entities responsible for subscribing to an event according to a QoS Profile specified with the qosProfile attribute.⌋

[TPS_MANI_03532] qosProfile of DdsRequiredServiceInstance.eventQosProps is optional ⌈The attribute qosProfile of DdsRequiredServiceInstance.eventQosProps is optional; if qosProfile is not defined, the underlying DDS entities shall be configured according to the qosProfile attribute of the parent DdsRequiredServiceInstance.⌋

[TPS_MANI_03567] Definition of DdsRequiredServiceInstance.fieldNotifierQosProps ⌈The DdsRequiredServiceInstance.fieldNotifierQosProps configures the DDS entities associated with the field according to a QoS Profile specified with the qosProfile attribute.⌋

[TPS_MANI_03568] qosProfile of DdsRequiredServiceInstance.fieldNotifierQosProps is optional ⌈The attribute qosProfile of DdsRequiredServiceInstance.fieldNotifierQosProps is optional; if qosProfile is not defined, the underlying DDS entities shall be configured according to the qosProfile attribute of the parent DdsRequiredServiceInstance.⌋

13.3.3.3 DDS Service Instance to Machine mapping

The DdsServiceInstanceToMachineMapping defines on which network / VLAN the DDS communication shall be deployed.

[TPS_MANI_03533] Meta-class DdsServiceInstanceToMachineMapping ⌈The DdsServiceInstanceToMachineMapping defines for a specific serviceInstance (either DdsProvidedServiceInstance or DdsRequiredServiceInstance) on which network the communication shall be done using the reference communicationConnector to CommunicationConnector.⌋

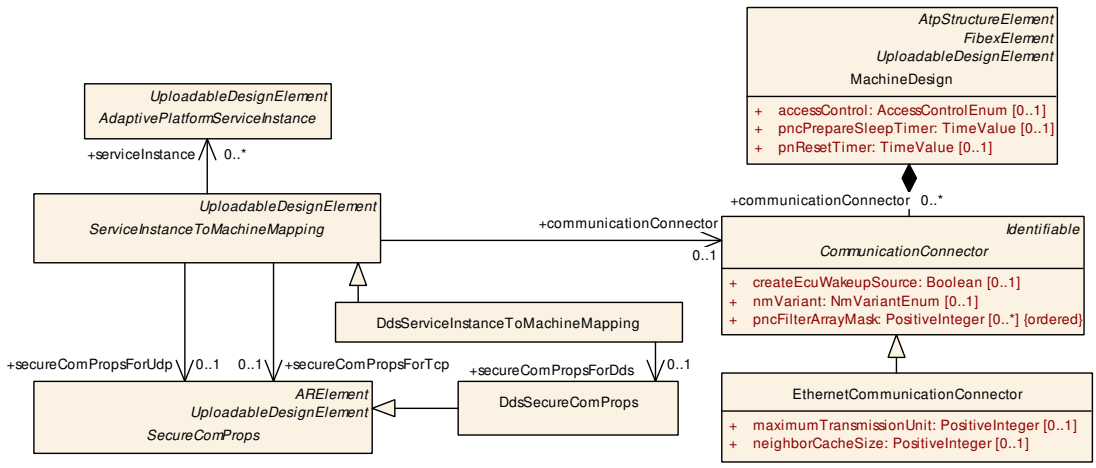


Figure 13.30: DDS Service Instance to Machine mapping

[constr_3684] Mutual exclusivity of Secure Communication Properties

Imposition time: IT_Mani

⌈The attributes ServiceInstanceToMachineMapping.secOcComPropsForMulticast and DdsServiceInstanceToMachineMapping.secureComPropsForDds are mutually exclusive, meaning zero or just one of them shall be set depending on whether no security, SecOC, or DDS Security is chosen as data-level security (optionally) above transport-level security.⌋

Class	**DdsServiceInstanceToMachineMapping**			
Note	This meta-class allows to map DdsServiceInstances to a CommunicationConnector of a Machine. **Tags:** atp.recommendedPackage=ServiceInstanceToMachineMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInstanceToMachineMapping*, *UploadableDesignElement*, *Uploadable PackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
secureCom PropsForDds	DdsSecureComProps	0..1	ref	Reference to SecureComProps applicable to the service instance. **Tags:** atp.Splitkey=secureComPropsForDds

Table 13.66: DdsServiceInstanceToMachineMapping

13.3.4 User Defined Service Instance Deployment

[TPS_MANI_03032] Description of middleware technologies not standardized by AUTOSAR ⌈The elements ProvidedUserDefinedServiceInstance and RequiredUserDefinedServiceInstance can be used to describe alternative middleware technologies that are not standardized by AUTOSAR.⌋

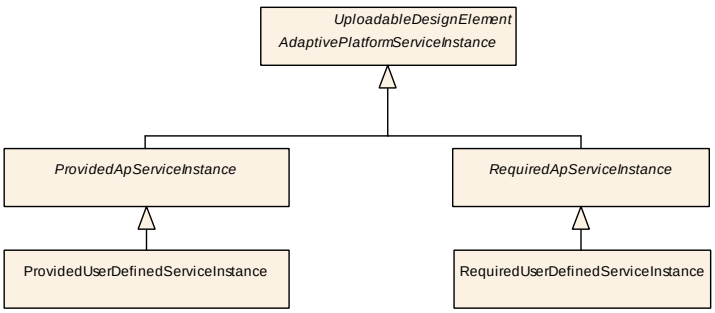


Figure 13.31: User Defined Service Instance Deployment

Both elements ProvidedUserDefinedServiceInstance and RequiredUserDefinedServiceInstance are Identifiable and therefore are able to describe special data (sdg) which is not represented by the standard model.

Class	ProvidedUserDefinedServiceInstance			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in a concrete implementation that is not standardized by AUTOSAR. **Tags:** atp.recommendedPackage=ServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *ProvidedApServiceInstance*, *Referrable*, *Uploadable DesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 13.67: ProvidedUserDefinedServiceInstance

Class	RequiredUserDefinedServiceInstance			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in a concrete implementation that is not standardized by AUTOSAR. **Tags:** atp.recommendedPackage=ServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *RequiredApServiceInstance*, *Uploadable DesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 13.68: RequiredUserDefinedServiceInstance

13.3.5 Service Deployment Relations

The meta-model is set up in a way that it is in general possible to create a relation between different sub-classes of

- `ServiceInterfaceDeployment`
- `AdaptivePlatformServiceInstance`
- `ServiceInstanceToMachineMapping`

For example, from the formal point of view, it would be possible to let a `UserDefinedServiceInstanceToMachineMapping` reference a `DdsProvided-ServiceInstance` that in turn references a `SomeipServiceInterface-Deployment`. However, it is questionable whether such a combination makes sense in a real-world model.

Consequently, the possible combinations of the three mentioned meta-classes need to be regulated by a constraint.

[constr_3641] Allowed combinations of ServiceInterfaceDeployment, AdaptivePlatformServiceInstance, ServiceInstanceToMachineMapping

Imposition time: IT_Mani

⌈

	DdsProvidedServiceInstance	ProvidedSomeipServiceInstance	ProvidedUserDefinedServiceInstance	DdsRequiredServiceInstance	RequiredSomeipServiceInstance	RequiredUserDefinedServiceInstance
DdsServiceInterfaceDeployment	Yes	No	Yes	Yes	No	Yes
SomeipServiceInterfaceDeployment	No	Yes	Yes	No	Yes	Yes
UserDefinedServiceInterfaceDeployment	No	No	Yes	No	No	Yes
DdsServiceInstanceToMachineMapping	Yes	No	No	Yes	No	No
SomeipServiceInstanceToMachineMapping	No	Yes	No	No	Yes	No
UserDefinedServiceInstanceToMachineMapping	No	Yes	Yes	No	Yes	Yes

⌋

13.4 End-to-End Protection

AUTOSAR supports the protection of events, methods, Field notifiers, Field get methods and Field set methods with E2E Profiles that are defined in the E2E Communication Protection Library, see [45, SWS E2E Library].

[TPS_MANI_03127] Usage of End2EndEventProtectionProps ⌈The End2EndEventProtectionProps element is used to define event specific E2E configuration settings in the context of an AdaptivePlatformServiceInstance.⌋

The E2E protection of a field notifier is possible with the End2EndEventProtectionProps.event reference, since each specific ServiceFieldDeployment element aggregates a ServiceEventDeployment in the role notifier.

If such an aggregated ServiceEventDeployment is referenced with the End2EndEventProtectionProps.event reference the E2E protection settings are valid for the notifier that is embedded by the ServiceFieldDeployment.

Since the End2EndEventProtectionProps element is aggregated by the abstract AdaptivePlatformServiceInstance, it can be used to describe the End-to-End

protection on specific derived classes like ProvidedSomeipServiceInstance or RequiredSomeipServiceInstance that fit the underlying middleware.

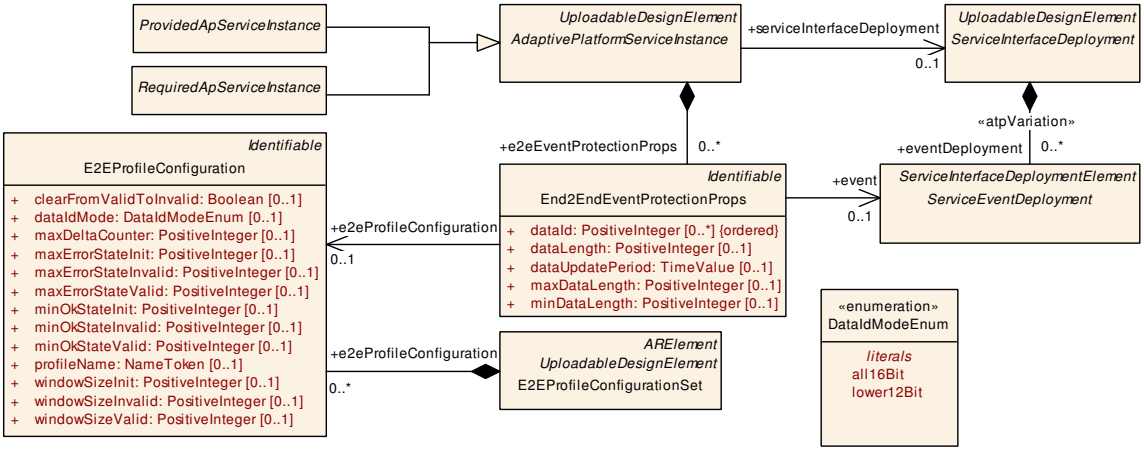


Figure 13.32: E2E EventProtection

With this approach, it is possible to define different End-to-End protection settings for different used transport layer mechanisms in the case of Multi-Binding.

[TPS_MANI_03228] Usage of End2EndMethodProtectionProps ⌈The End2End-MethodProtectionProps element is used to define method specific E2E configuration settings in the context of an AdaptivePlatformServiceInstance.⌋

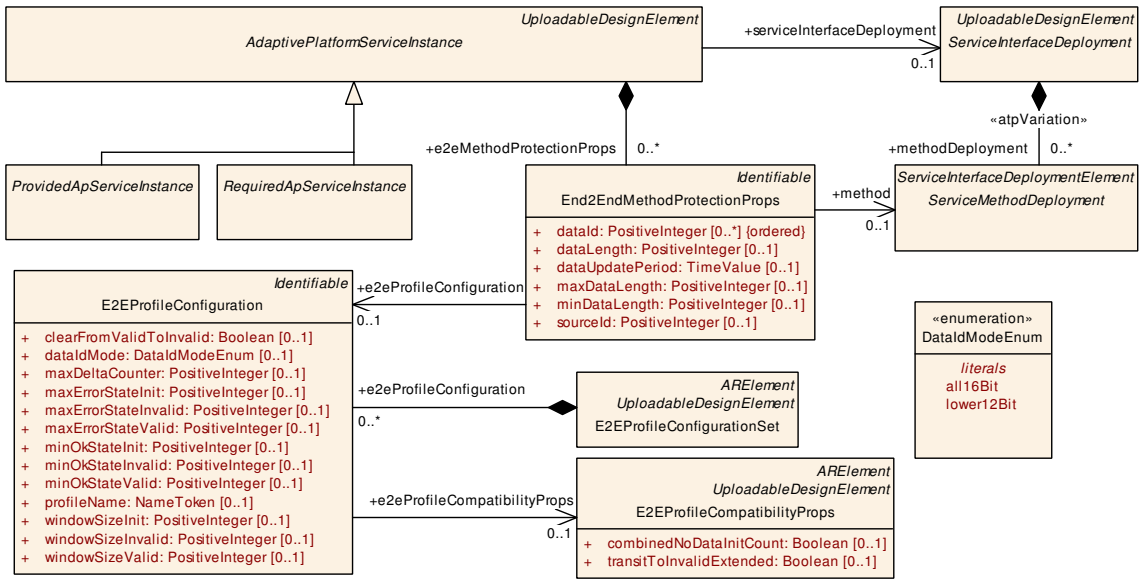


Figure 13.33: E2E MethodProtection

The E2E protection of field get and set methods is possible with the End2End-MethodProtectionProps.method reference, since each specific ServiceField-Deployment element is allowed to aggregate a ServiceMethodDeployment in the role get and/or set.

If such an aggregated ServiceMethodDeployment is referenced with the End2EndMethodProtectionProps.method reference the E2E protection settings are valid for the get or set method that is embedded by the ServiceFieldDeployment.

[TPS_MANI_03129] E2E profile ⌈The E2E profile is defined by E2EProfileConfiguration.profileName.⌋

[TPS_MANI_03130] Standardized values of the attribute E2EProfileConfiguration.profileName ⌈The E2EProfileConfiguration.profileName that is referenced by an End2EndEventProtectionProps or by an End2EndMethodProtectionProps can have the following values that are standardized by AUTOSAR:

- PROFILE_04
- PROFILE_05
- PROFILE_06
- PROFILE_07
- PROFILE_08
- PROFILE_11
- PROFILE_22
- PROFILE_04m
- PROFILE_07m
- PROFILE_44
- PROFILE_08m
- PROFILE_44m

⌋

[TPS_MANI_03131] Usage of non-standardized values of attribute E2EProfileConfiguration.profileName ⌈The values for the profileName of E2EProfileConfiguration mentioned in [TPS_MANI_03130] are standardized and reserved for being used in the way the AUTOSAR standard foresees.

PROFILE_01 and PROFILE_02 are also reserved by AUTOSAR but excluded for usage on the *adaptive platform*. In addition, it is possible to use non-standardized values for the profileName.⌋

[TPS_MANI_03128] Usage of same `End2EndEventProtectionProps.dataId` in the case of Multi-Binding ⌈In the case of Multi-Binding, i.e. if different `AdaptivePlatformServiceInstance`s exist that are mapped by `ServiceInstanceToPortPrototypeMapping` to the same `PortPrototype`, the different `AdaptivePlatformServiceInstance`s may contain the same `dataId` for the same `event`.⌋

In other words, if a `PortPrototype` contains two transport layer bindings, e.g. a `ProvidedSomeipServiceInstance` and a `ProvidedUserDefinedServiceInstance` representing an IPC communication, then an `event` is allowed to be protected with the same `dataId` in both `AdaptivePlatformServiceInstance`s because the two `AdaptivePlatformServiceInstance`s effectively represent the identical piece of data.

[TPS_MANI_03229] Usage of same `End2EndMethodProtectionProps.dataId` in the case of Multi-Binding ⌈In the case of Multi-Binding, i.e. if different `AdaptivePlatformServiceInstance`s exist that are mapped by `ServiceInstanceToPortPrototypeMapping` to the same `PortPrototype`, the different `AdaptivePlatformServiceInstance`s may contain the same `dataId` for the same `method`.⌋

In other words, if a `PortPrototype` contains two transport layer bindings, e.g. a `ProvidedSomeipServiceInstance` and a `ProvidedUserDefinedServiceInstance` representing an IPC communication, then a `method` is allowed to be protected with the same `dataId` in both `AdaptivePlatformServiceInstance`s.

[TPS_MANI_03252] Usage of same`End2EndMethodProtectionProps.sourceId` in the case of Multi-Binding ⌈In the case of Multi-Binding, i.e. if different `AdaptivePlatformServiceInstance`s exist that are mapped by `ServiceInstanceToPortPrototypeMapping` to the same `PortPrototype`, the different `AdaptivePlatformServiceInstance`s may contain the same `sourceId` (for the same and even for a different `method`).⌋

In other words, if a `PortPrototype` contains two transport layer bindings, e.g. a `ProvidedSomeipServiceInstance` and a `ProvidedUserDefinedServiceInstance` representing an IPC communication, then a single as well as different `method`s are allowed to be protected with the same `sourceId` in both `AdaptivePlatformServiceInstance`s.

Class	**End2EndEventProtectionProps**			
Note	This element allows to protect an event or a field notifier with an E2E profile. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AdaptivePlatformServiceInstance*.e2eEventProtectionProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**End2EndEventProtectionProps**			
dataId (ordered)	PositiveInteger	*	attr	This represents a unique numerical identifier for the referenced event or field notifier that is included in the CRC calculation. Note: ID is used for protection against masquerading. The details concerning the maximum number of values (this information is specific for each E2E profile) applicable for this attribute are controlled by a semantic constraint that depends on the category of the EndToEnd Protection.
dataLength	PositiveInteger	0..1	attr	Length of payload including E2E header in bits.
dataUpdate Period	TimeValue	0..1	attr	This attribute describes the period in which the applications are assumed to process E2E-protected messages. The middleware does not use this attribute at all.
e2eProfile Configuration	E2EProfileConfiguration	0..1	ref	Reference to E2E profile configuration settings that are valid to protect the referenced event or field notifier. **Tags:** atp.Splitkey=e2eProfileConfiguration
event	ServiceEvent Deployment	0..1	ref	Reference to an event that is protected by the E2E profile. **Tags:** atp.Splitkey=event
maxDataLength	PositiveInteger	0..1	attr	Maximum length of payload including E2E header in bits.
minDataLength	PositiveInteger	0..1	attr	Minimum length of payload including E2E header in bits.

Table 13.69: End2EndEventProtectionProps

Class	**End2EndMethodProtectionProps**			
Note	This element allows to protect a method, a field setter or a field getter with an E2E profile. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AdaptivePlatformServiceInstance*.e2eMethodProtectionProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataId (ordered)	PositiveInteger	*	attr	This represents a numerical identifier that is included in the CRC calculation. This dataId is used for call and response. Note: ID is used for protection against masquerading. The details concerning the maximum number of values (this information is specific for each E2E profile) applicable for this attribute are controlled by a semantic constraint that depends on the category of the EndToEnd Protection.
dataLength	PositiveInteger	0..1	attr	Length of payload including E2E header in bits.
dataUpdate Period	TimeValue	0..1	attr	This attribute describes the period in which the applications are assumed to process E2E-protected messages. The middleware does not use this attribute at all.
e2eProfile Configuration	E2EProfileConfiguration	0..1	ref	Reference to E2E profile configuration settings that are valid to protect the referenced method, field getter or field setter. **Tags:** atp.Splitkey=e2eProfileConfiguration
maxDataLength	PositiveInteger	0..1	attr	Maximum length of payload including E2E header in bits.
method	ServiceMethod Deployment	0..1	ref	Reference to a method, a field getter or a field setter that is protected by the E2E profile. **Tags:** atp.Splitkey=method

▽

△

Class	End2EndMethodProtectionProps			
minDataLength	PositiveInteger	0..1	attr	Minimum length of payload including E2E header in bits.
sourceId	PositiveInteger	0..1	attr	This represents a unique numerical identifier identifying the source of a certain transmission. In case of C/S communication, this ID uniquely identifies the client. Note: ID is used for protection against masquerading. The details concerning the maximum number of values (this information is specific for each E2E profile) applicable for this attribute are controlled by a semantic constraint that depends on the category of the EndToEnd Protection.

Table 13.70: End2EndMethodProtectionProps

Class	E2EProfileConfigurationSet			
Note	This meta-class represents the ability to aggregate a collection of E2EProfileConfigurations. **Tags:** atp.recommendedPackage=E2EProfileConfigurationSets This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
e2eProfile Configuration	E2EProfileConfiguration	*	aggr	This represents the collection of E2EProfileConfigurations aggregated at the E2EProfileConfigurationSet. **Tags:** atp.Splitkey=e2eProfileConfiguration, e2eProfile Configuration.shortName

Table 13.71: E2EProfileConfigurationSet

Class	E2EProfileConfiguration			
Note	This element holds E2E profile specific configuration settings. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	E2EProfileConfigurationSet.e2eProfileConfiguration			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
clearFromValid ToInvalid	Boolean	0..1	attr	Clear monitoring window on transition from state Valid to state Invalid.
dataIdMode	DataIdModeEnum	0..1	attr	This attribute describes the inclusion mode that is used to include the implicit Data ID in the one-byte CRC.
e2eProfile Compatibility Props	E2EProfileCompatibility Props	0..1	ref	Reference to additional settings for the E2E state machine. **Tags:** atp.Splitkey=e2eProfileCompatibilityProps
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.

▽

△

Class	**E2EProfileConfiguration**			
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.
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.
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.
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.
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.
profileName	NameToken	0..1	attr	Definition of the E2E profile.
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 13.72: E2EProfileConfiguration

[constr_10226] Multiplicity of attribute `E2EProfileConfiguration.profileName`

Imposition time: `IT_Mani`

⌈For each `E2EProfileConfiguration`, the attribute `profileName` shall exist.⌋

Enumeration	**DataIdModeEnum**
Note	Supported inclusion modes to include the implicit two-byte Data ID in the one-byte CRC.
Aggregated by	E2EProfileConfiguration.dataIdMode, EndToEndTransformationDescription.dataIdMode
Literal	***Description***
all16Bit	Two bytes are included in the CRC (double ID configuration). **Tags:** atp.EnumerationLiteralIndex=0
lower12Bit	The low byte is included in the implicit CRC calculation, the low nibble of the high byte is transmitted along with the data (i.e. it is explicitly included), the high nibble of the high byte is not used. This is applicable for the IDs up to 12 bits. **Tags:** atp.EnumerationLiteralIndex=2

Table 13.73: DataIdModeEnum

Class	**E2EProfileCompatibilityProps**
Note	This meta-class collects settings for configuration of the E2E state machine. **Tags:** atp.recommendedPackage=E2EProfileCompatibilityPropsCollection
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*
Aggregated by	ARPackage.element

▽

△

Class	**E2EProfileCompatibilityProps**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
combinedNo DataInitCount	Boolean	0..1	attr	E2E State machine behavior concerning counting of detected counter errors and missing messages in states NODATA and INIT • value = 0 (false) or not defined: counting of detected counter errors and missing messages in states NODATA and INIT are counted per state separated (Autosar R23-11 or former behavior) • value = 1 (true): counting of detected counter errors and missing messages in states NODATA and INIT are counted in total
transitToInvalid Extended	Boolean	0..1	attr	E2E State machine behavior concerning transition from NODATA/INIT to INVALID value=0 (false): no direct transition from NODATA to INVALID, no transition from INIT to INVALID due to counter-related faults (Autosar R19-11 or former behavior) value=1 (true): direct transition from NODATA to INVALID covered, transition from INIT to INVALID due to counter-related faults covered (state machine extended)

Table 13.74: E2EProfileCompatibilityProps

It is possible to overwrite the E2E state machine configuration settings that are defined in `E2EProfileConfiguration` at the `RPortPrototype` of a `SwComponentType` with settings available in the `ReceiverComSpec` as described in [TPS_MANI_03132]. With this approach, it is possible to define individual E2E settings for different receivers of the event, or field notifiers.

Likewise, it is possible to overwrite the E2E state machine configuration settings that are defined in `E2EProfileConfiguration` at the `RPortPrototype` of a `SwComponentType` with settings available in the `ClientComSpec` as described in [TPS_MANI_01324].

With this approach, it is possible to define individual E2E settings for different callers of a `method`.

Finally, it is possible to overwrite the E2E state machine configuration settings that are defined in `E2EProfileConfiguration` at the `PPortPrototype` of a `SwComponentType` with settings available in the `ServerComSpec` as described in [TPS_MANI_01325].

In `PROFILE_22`, the `dataId` is defined as a list of 16 dataId values, where a different value is transmitted depending on the counter value.

The Classic Platform attributes `counterOffset`, `crcOffset` and `dataIdNibble-Offset` are not configurable on the *adaptive platform* and are set to fixed values by the AUTOSAR Standard.

[constr_3493] Applicable attributes for standardized E2E Profiles

Imposition time: IT_Mani

⌈

E2E Attributes	Root Element			Attribute Existence per Profile												
	End2EndEventProtectionProps	End2EndMethodProtectionProps	E2EProfileConfiguration	PROFILE_04	PROFILE_05	PROFILE_06	PROFILE_07	PROFILE_08	PROFILE_11	PROFILE_22	PROFILE_04m	PROFILE_07m	PROFILE_44	PROFILE_08m	PROFILE_44m
dataId	x	x		1	1	1	1	1	1	n	1	1	1	1	1
dataLength	x	x			x				x	x					
minDataLength	x	x		x		x	x	x			x	x	x	x	x
maxDataLength	x	x		x		x	x	x			x	x	x	x	x
dataUpdatePeriod	x	x		x	x	x	x	x	x	x	x	x	x	x	x
sourceId		x									x	x		x	x
dataIdMode			x						x						
maxDeltaCounter			x	x	x	x	x	x	x	x	x	x	x	x	x
maxErrorStateInit			x	x	x	x	x	x	x	x	x	x	x	x	x
maxErrorStateInvalid			x	x	x	x	x	x	x	x	x	x	x	x	x
maxErrorStateValid			x	x	x	x	x	x	x	x	x	x	x	x	x
minOkStateInit			x	x	x	x	x	x	x	x	x	x	x	x	x
minOkStateInvalid			x	x	x	x	x	x	x	x	x	x	x	x	x
minOkStateValid			x	x	x	x	x	x	x	x	x	x	x	x	x
windowSizeValid			x	x	x	x	x	x	x	x	x	x	x	x	x
windowSizeInvalid			x	x	x	x	x	x	x	x	x	x	x	x	x
windowSizeInit			x	x	x	x	x	x	x	x	x	x	x	x	x
clearFromValidToInvalid			x	x	x	x	x	x	x	x	x	x	x	x	x

⌋

[constr_5230] The attribute E2EProfileCompatibilityProps.transitToInvalidExtended shall exist for each E2EProfileConfiguration

Imposition time: IT_Mani

⌈For each E2EProfileConfiguration, a reference to meta-class E2EProfileCompatibilityProps in the role e2eProfileCompatibilityProps shall exist and the referenced E2EProfileCompatibilityProps shall define a value for the attribute transitToInvalidExtended.⌋

[constr_5250] Protection of AdaptivePlatformServiceInstances of the same ServiceInterfaceDeployment

Imposition time: IT_Mani

⌈If several AdaptivePlatformServiceInstances exist that are referencing the same ServiceInterfaceDeployment and these AdaptivePlatformServiceInstances contain aggregated End2EndMethodProtectionProps and/or End2EndEventProtectionProps then the E2EProfileConfigurations that are referenced by the End2EndMethodProtectionProps and End2EndEventProtectionProps shall have the same profileName defined.⌋

In other words it is not allowed to protect different AdaptivePlatformServiceInstances of the same ServiceInterfaceDeployment with different E2E Profiles.

The End2EndMethodProtectionProps and/or End2EndEventProtectionProps in the different AdaptivePlatformServiceInstances are allowed to reference different E2EProfileConfigurations since the same E2E Profile may be configured with different E2E settings.

13.5 Secure Communication

AUTOSAR supports different protocols that provide communication security over a network. To configure the secured communication of ServiceInterface elements between a ProvidedApServiceInstance and a RequiredApServiceInstance the ServiceInterfaceElementSecureComConfig meta-class is defined.

[TPS_MANI_03133] Usage of ServiceInterfaceElementSecureComConfig
⌈The ServiceInterfaceElementSecureComConfig element is used to define ServiceInterface element specific secure communication configuration settings in the context of an AdaptivePlatformServiceInstance.⌋

The modeling allows protecting selected elements of a ServiceInterface, like particular events or methods.

Since the ServiceInterfaceElementSecureComConfig meta-class is aggregated by the abstract AdaptivePlatformServiceInstance it can be used to configure the secure communication on specific derived classes like ProvidedSomeipServiceInstance or RequiredSomeipServiceInstance that fit the underlying middleware.

With this approach, it is possible to define different communication security protections for different used transport layer mechanisms in the case of Multi-Binding.

[constr_3391] `ServiceInterfaceElementSecureComConfig` references to `ServiceInterfaceDeployment` elements

Imposition time: `IT_Mani`

⌈`ServiceInterfaceElementSecureComConfig` element shall be defined for exactly one `ServiceInterface` element and shall therefore contain only a single reference to an element defined in the scope of a `ServiceInterfaceDeployment`.⌋

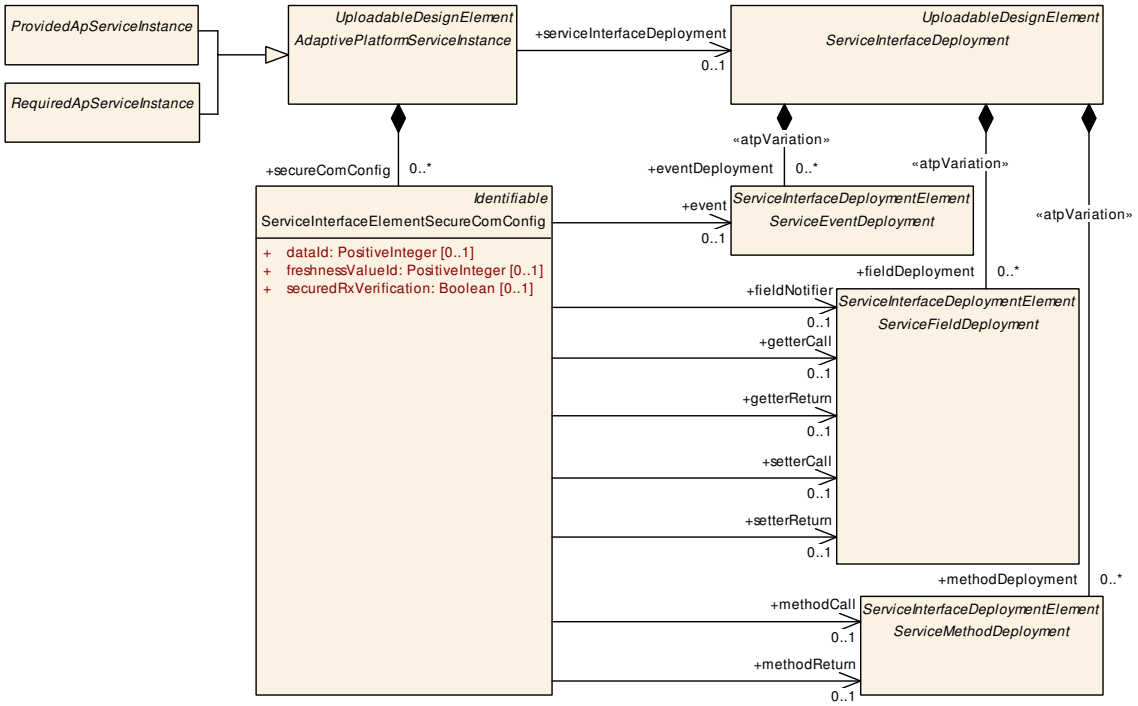


Figure 13.34: Secure Communication

Class	**ServiceInterfaceElementSecureComConfig**			
Note	This element allows to secure the communication of the referenced ServiceInterface element. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AdaptivePlatformServiceInstance*.secureComConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataId	PositiveInteger	0..1	attr	This attribute defines a unique numerical identifier for the referenced ServiceInterface element.
event	ServiceEvent Deployment	0..1	ref	Reference to an event that is protected by a security protocol. **Tags:** atp.Splitkey=event
fieldNotifier	ServiceField Deployment	0..1	ref	Reference to a field notifier that is protected by a security protocol. **Tags:** atp.Splitkey=fieldNotifier
freshnessValue Id	PositiveInteger	0..1	attr	This attribute defines the Id of the Freshness Value.

▽

△

Class	ServiceInterfaceElementSecureComConfig			
getterCall	ServiceField Deployment	0..1	ref	Reference to a field getter call message that is protected by a security protocol. **Tags:** atp.Splitkey=getterCall
getterReturn	ServiceField Deployment	0..1	ref	Reference to a field getter return message that is protected by a security protocol. **Tags:** atp.Splitkey=getterReturn
methodCall	ServiceMethod Deployment	0..1	ref	Reference to a method call message that is protected by a security protocol. **Tags:** atp.Splitkey=methodCall
methodReturn	ServiceMethod Deployment	0..1	ref	Reference to a method return message that is protected by a security protocol. **Tags:** atp.Splitkey=methodReturn
securedRx Verification	Boolean	0..1	attr	This attribute defines whether the ServiceInterface element shall verify its security credentials during reception.
setterCall	ServiceField Deployment	0..1	ref	Reference to a field setter call message that is protected by a security protocol. **Tags:** atp.Splitkey=setterCall
setterReturn	ServiceField Deployment	0..1	ref	Reference to a field setter return message that is protected by a security protocol. **Tags:** atp.Splitkey=setterReturn

Table 13.75: ServiceInterfaceElementSecureComConfig

The attributes in the `ServiceInterfaceElementSecureComConfig` meta-class define security configuration settings that are specific for the referenced `ServiceInterface` element in the context of an `AdaptivePlatformServiceInstance`. The used security protocol is defined in the `ServiceInstanceToMachineMapping`.

[TPS_MANI_03199] Endpoint protection by `SecureComProps` ⌈The `ServiceInstanceToMachineMapping` allows assigning a security protocol configuration settings that are defined in the referenced `SecureComProps` meta-class to protect endpoints that are defined by the Transport Protocol, Port and IP Address on which one or several `AdaptivePlatformServiceInstance`s are provided or consumed.⌋

[TPS_MANI_03200] `SecureComProps` for udp, tcp and multicast communication ⌈The `ServiceInstanceToMachineMapping` allows assigning a security protocol configuration settings for:

- udp communication if the `ServiceInstanceToMachineMapping` refers to the `SecureComProps` in the role `secureComPropsForUdp`
- tcp communication if the `ServiceInstanceToMachineMapping` refers to the `SecureComProps` in the role `secureComPropsForTcp`
- multicast communication if the `ServiceInstanceToMachineMapping` refers to the `SecOcSecureComProps` in the role `secOcComPropsForMulticast`

⌋

With this modeling approach, it is possible to configure different security protocol settings for the communication over TCP, UDP and multicast. For example, it is allowed to use different settings to protect the TCP communication of a AdaptivePlatformServiceInstance via TLS and UDP communication via DTLS.

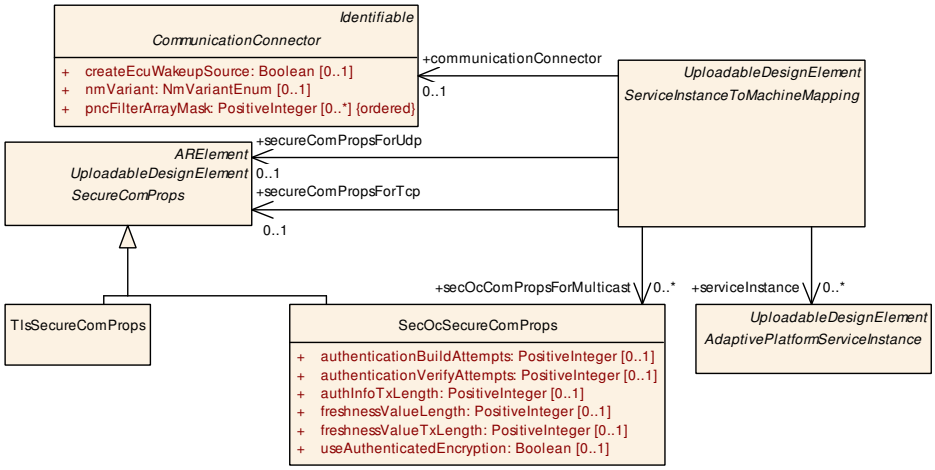


Figure 13.35: Security protocol configuration

Protection of IP multicast traffic is only supported by SecOC; therefore, the ServiceInstanceToMachineMapping refers directly to the SecOcSecureComProps in the secOcComPropsForMulticast role.

Class	**SecureComProps** (abstract)			
Note	This meta-class defines a communication security protocol and its configuration settings.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Subclasses	DdsSecureComProps, SecOcSecureComProps, TlsSecureComProps			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 13.76: SecureComProps

[advisory_03003] Secure communication should not be lost by switching to multicast

Imposition time: IT_Mani

⌈If a ProvidedSomeipServiceInstance that is referenced by a ServiceInstanceToMachineMapping provides at least one SomeipProvidedEventGroup where multicastThreshold is not equal to 0 and containing at least one udp event then the ProvidedSomeipServiceInstance should either:

- not reference SecureComProps in the role secureComPropsForUdp or secureComPropsForTcp or secOcComPropsForMulticast or
- reference SecOcSecureComProps in the role secureComPropsForUdp and secOcComPropsForMulticast

to make sure that the secure communication properties are consistent for the unicast and multicast communication.⌋

TLS is a point-to-point security protocol that does not support multicast. A switch between secure communication with TLS for IP unicast and SecOC for IP multicast is not supported by AUTOSAR Classic Platform.

13.5.1 Secure Communication over TLS

The configuration of the Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) protocols is supported with the `TlsSecureComProps` meta-class, which is a specialization of `SecureComProps`.

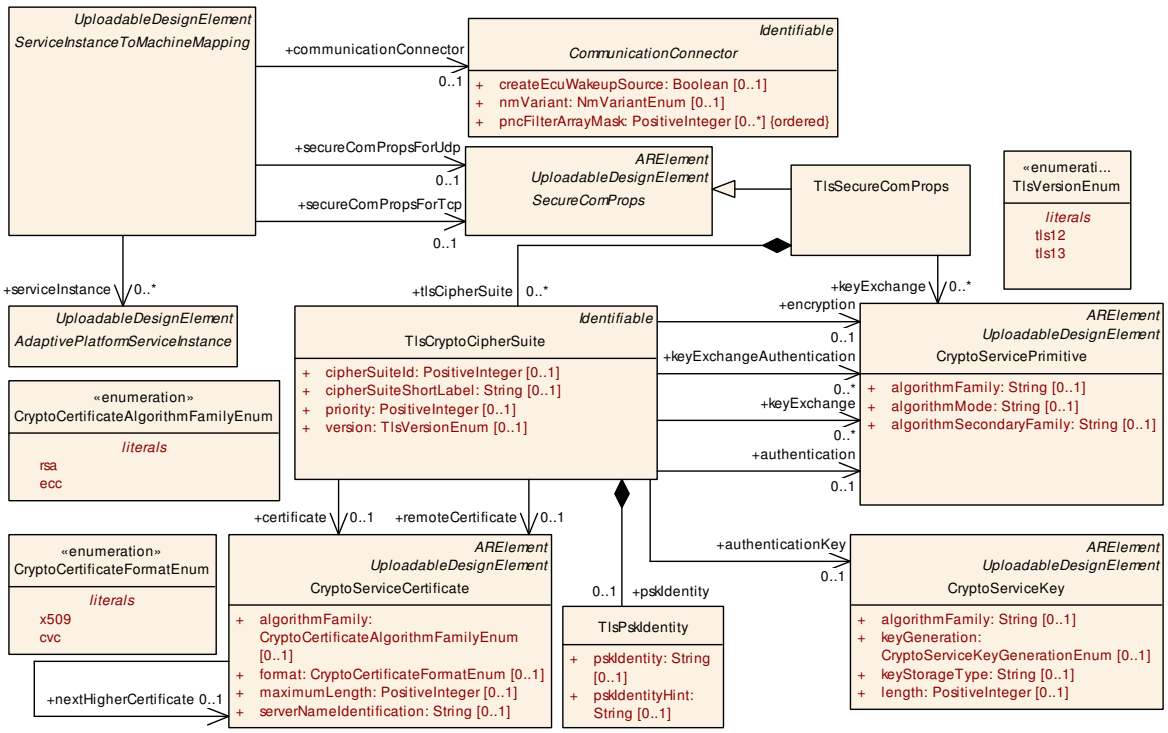


Figure 13.36: Secure Communication over TLS

It is a common use case that only one end of a TLS-based connection is actually modeled in an AUTOSAR model. It is therefore important that the modeling does not rely on or imply knowledge about both ends of such a TLS-based connection.

An AUTOSAR model that only describes one end of the communication is required to work, independently of the availability of a formal modeling of the other end.

Class	**TlsSecureComProps**			
Note	Configuration of the Transport Layer Security protocol (TLS). **Tags:** atp.recommendedPackage=SecureComProps This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *SecureComProps*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
keyExchange	CryptoServicePrimitive	*	ref	This reference identifies the shared (i.e. applicable for each of the aggregated cipher suites) crypto service primitive for the execution of key exchange during the handshake phase. **Tags:** atp.Splitkey=keyExchange
tlsCipherSuite	TlsCryptoCipherSuite	*	aggr	Collection of supported cipher suites that are used to negotiate the security settings for a network connection defined by the ServiceInstanceToMachineMapping. **Tags:** atp.Splitkey=tlsCipherSuite, tlsCipherSuite.shortName

Table 13.77: TlsSecureComProps

Enumeration	**CryptoServiceKeyGenerationEnum**
Note	This enumeration shall be taken to express the handling of a crypto key in terms of whether it is obtained from e.g. a diagnostic tester or whether it is created by derivation from a master key.
Aggregated by	CryptoServiceKey.keyGeneration
Literal	***Description***
keyDerivation	This means that the crypto key is created by derivation from a master key. **Tags:** atp.EnumerationLiteralIndex=0
keyStorage	This means that the crypto key is obtained from an external entity, e.g. a diagnostic tester. **Tags:** atp.EnumerationLiteralIndex=1

Table 13.78: CryptoServiceKeyGenerationEnum

TLS is composed of the TLS Record Protocol and the TLS Handshake Protocol. The Record Protocol provides connection security and encrypts and authenticates packets. The record layer functions can be called at any time after the handshake process is finished, when there is a need to receive or send data.

The Handshake Protocol allows the server and client to authenticate each other and to negotiate encryption algorithms and cryptographic keys before any data is exchanged.

In order to establish a cryptographically secure data channel, the communication partners in the form of `ServiceInstanceToMachineMapping`s shall agree on cipher suites and on keys that will be used to encrypt the data.

The client sends a list of supported cipher suites to the server. The server decides on a cipher suite from the list provided by the client, and continues with the handshake.

The server and client roles cannot be swapped while the connection exists, i.e. a *server* remains the *server* for the full amount of time the connection exists.

[TPS_MANI_03213] Semantics of meta-class `TlsSecureComProps` ⌈As a subclass of `SecureComProps`, meta-class `TlsSecureComProps` has the ability to collect the TLS-related configuration aspects from either the perspective of the client or the server.

In the case of TLS, the collection boils down to the aggregation of meta-class `TlsCryptoCipherSuite` in the role `tlsCipherSuite` plus the ability (by means of the role `keyExchange`) to define handshake properties that are shared for each of the aggregated `tlsCipherSuite`.⌋

[constr_5047] Supported values of `TlsSecureComProps.category`

Imposition time: `IT_Mani`

⌈The only supported values of attribute `TlsSecureComProps.category` are:

- **`TLS_SERVER`**: the `TlsSecureComProps` assumes the role of the *server* in the TLS connection.
- **`TLS_CLIENT`**: the `TlsSecureComProps` assumes the role of the *client* in the TLS connection.

⌋

[TPS_MANI_03134] Configuration of supported TLS ciphersuites ⌈The creation of a TLS connection requires the usage of a suite of cryptographic operations in specific roles, also known as a *cipher suite*.

Meta-class `TlsCryptoCipherSuite` represents a given cipher suite for a TLS connection. `TlsCryptoCipherSuite` references meta-class `CryptoServicePrimitive` in three dedicated roles that represent the steps of the creation of a TLS connection.

More specifically, the cryptographic operations for setting up a TLS connection involve the following steps:

- **Key exchange**: these `CryptoServicePrimitive`s may be used for the handshake phase of the TLS connection. Different alternatives exist for executing this phase and therefore the multiplicity of this reference is 0..*.
- **Authentication** of communication partners during the operational phase of the TLS connection. For this purpose a single `CryptoServicePrimitive` is used on each end of the communication.
- **Encryption** of content exchanged between the communication partners that have established the TLS connection. For this purpose a single `CryptoServicePrimitive` is used on each end of the communication.

⌋

If a certificate-based cipher suite is used, the `TlsCryptoCipherSuite.authenticationKey` references the private key used for authentication. For PSK

cipher suites, the `TlsCryptoCipherSuite.authenticationKey` references the pre-shared (symmetric) key used for authentication.

Class	**TlsCryptoCipherSuite**			
Note	This meta-class represents a cipher suite for describing cryptographic operations in the context of establishing a connection of ApplicationEndpoints that is protected by TLS.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	TlsCryptoServiceMapping.tlsCipherSuite, TlsSecureComProps.tlsCipherSuite			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authentication	CryptoServicePrimitive	0..1	ref	This reference identifies the crypto service primitive for the generation and verification of MACs. **Tags:** atp.Splitkey=authentication
authentication Key	CryptoServiceKey	0..1	ref	This reference identifies the private key or pre-shared key used for authentication. **Tags:** atp.Splitkey=authenticationKey
certificate	CryptoService Certificate	0..1	ref	This reference identifies the applicable local certificate. **Tags:** atp.Splitkey=certificate
cipherSuiteId	PositiveInteger	0..1	attr	Identification of the CipherSuite according to the IANA assignments list.
cipherSuite ShortLabel	String	0..1	attr	Name of the CipherSuite according to the IANA assignments list.
ellipticCurve	CryptoEllipticCurve Props	*	ref	This references point to the properties of elliptic curves. **Tags:** atp.Splitkey=ellipticCurve
encryption	CryptoServicePrimitive	0..1	ref	This reference identifies the crypto service primitive for the execution of encryption. **Tags:** atp.Splitkey=encryption
keyExchange	CryptoServicePrimitive	*	ref	This reference identifies the individual (i.e. per cipher suite) crypto service primitive for the execution of key exchange during the handshake phase. **Tags:** atp.Splitkey=keyExchange
keyExchange Authentication	CryptoServicePrimitive	*	ref	This reference identifies the crypto service primitives for the generation and verification of signatures during the key exchange algorithm. **Tags:** atp.Splitkey=keyExchangeAuthentication
priority	PositiveInteger	0..1	attr	This attribute identifies the priority of the cipher suite. Range: 1..65535. Lower values represent higher priorities.
props	TlsCryptoCipherSuite Props	0..1	aggr	The aggregated TlsCryptoCipherSuiteProps provide details for the TLS Cipher Suite. **Tags:** atp.Splitkey=props, props.shortName
pskIdentity	TlsPskIdentity	0..1	aggr	Pre-shared key identity shared during the handshake among the communication parties, to establish a TLS connection if the handshake is based on the existence of a pre-shared key. **Tags:** atp.Splitkey=pskIdentity
remote Certificate	CryptoService Certificate	0..1	ref	This reference identifies the applicable remote certificate. **Tags:** atp.Splitkey=remoteCertificate
signature Scheme	CryptoSignature Scheme	*	ref	This reference points to the properties of a TLS Signature Scheme. **Tags:** atp.Splitkey=signatureScheme
version	TlsVersionEnum	0..1	attr	This attribute supports the definition of the applicable version of TLS.

Table 13.79: TlsCryptoCipherSuite

According to TLS, each TlsCryptoCipherSuite may support multiple PSKs or certificates. The model restricts the multiplicity of references from TlsCryptoCipherSuite to CryptoServiceCertificate and TlsPskIdentity to 0..1.

A TLS configuration with multiple PSKs or Certificates per cipher suite can be achieved by having multiple TlsCryptoCipherSuites that share the same set of parameters but reference different CryptoServiceCertificates or TlsPskIdentitys, respectively.

Class	**CryptoServicePrimitive**			
Note	This meta-class has the ability to represent a crypto primitive. **Tags:** atp.recommendedPackage=CryptoPrimitives			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
algorithmFamily	String	0..1	attr	This attribute represents a description of the family (e.g. AES) of crypto algorithm implemented by the crypto primitive.
algorithmMode	String	0..1	attr	This attribute represents a description of the mode of the crypto algorithm implemented by the crypto primitive.
algorithm SecondaryFamily	String	0..1	attr	This attribute represents a further description of the secondary family of crypto algorithm implemented by the crypto primitive. The secondary family is needed for the specification of the hash algorithm for a signature check, e.g. using RSA.

Table 13.80: CryptoServicePrimitive

Class	**CryptoEllipticCurveProps**			
Note	This meta-class provides attributes to specify the properties of elliptic curves. **Tags:** atp.recommendedPackage=CryptoEllipticCurveProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
namedCurveId	PositiveInteger	0..1	attr	Defines the value of one specific NamedCurve Id.

Table 13.81: CryptoEllipticCurveProps

Class	**TlsCryptoCipherSuiteProps**			
Note	This meta-class provides attributes to specify details of TLS Cipher Suites.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	TlsCryptoCipherSuite.props			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpTlsUse SecurityExtensionForce EncryptThenMac	Boolean	0..1	attr	Defines if the security extension according to IETF RFC 7366 shall be supported. This is useful for cipher suites using CBC mode.

Table 13.82: TlsCryptoCipherSuiteProps

Class	CryptoSignatureScheme			
Note	This meta-class provides attributes to specify the TLS Signature Scheme. **Tags:** atp.recommendedPackage=CryptoSignatureSchemas			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
signature SchemeId	PositiveInteger	0..1	attr	Defines the value of one specific TLS Signature Scheme.

Table 13.83: CryptoSignatureScheme

Enumeration	TlsVersionEnum
Note	This meta-class has the ability to identify a specific version of the transport-layer security (TLS) protocol.
Aggregated by	TlsCryptoCipherSuite.version
Literal	**Description**
tls12	TLS version 1.2 **Tags:** atp.EnumerationLiteralIndex=0
tls13	TLS version 1.3 **Tags:** atp.EnumerationLiteralIndex=2

Table 13.84: TlsVersionEnum

Class	CryptoServiceKey			
Note	This meta-class has the ability to represent a crypto key. **Tags:** atp.recommendedPackage=CryptoDevelopmentKeys			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
algorithmFamily	String	0..1	attr	This attribute represent the description of the family of the applicable crypto algorithm.
development Value	ValueSpecification	0..1	aggr	This aggregation represents the ability to assign a specific value to the crypto key as part of the system description. This value can then be taken for the development of the respective ECU. **Tags:** atp.Splitkey=developmentValue
keyGeneration	CryptoServiceKey GenerationEnum	0..1	attr	This attribute describes how a the specific cryptographic key is created.
keyStorageType	String	0..1	attr	This attribute describes where the enclosing cryptographic key shall be stored. AUTOSAR reserves specific values for this attributes but it is possible to insert custom values as well.
length	PositiveInteger	0..1	attr	This attribute describes the length of the cryptographic key in bits.

Table 13.85: CryptoServiceKey

[TPS_MANI_03214] Existence of `TlsCryptoCipherSuite.keyExchange` vs. `TlsSecureComProps.keyExchange` ⌈The role `TlsSecureComProps.keyExchange` has been introduced as an optimization.

It is assumed that the references for key exchange look very similar or even identical for many concrete TlsCryptoCipherSuites.

Adding these references in an identical form to several TlsCryptoCipherSuites does not make sense. Therefore, TlsSecureComProps allows defining these references as well with the intention to make them valid for all TlsSecureComProps.tlsCipherSuites.

A mixture of references in the role TlsCryptoCipherSuite.keyExchange and TlsSecureComProps.keyExchange is supported.⌋

[TPS_MANI_03215] Semantics of CryptoServiceCertificate ⌈Meta-class CryptoServiceCertificate represents a cryptographic certificate needed for the creation of a TLS connection between *server* and *client*.⌋

Class	**CryptoServiceCertificate**			
Note	This meta-class represents the ability to model a cryptographic certificate. **Tags:** atp.recommendedPackage=CryptoServiceCertificates			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
algorithmFamily	CryptoCertificate AlgorithmFamilyEnum	0..1	attr	This attribute represents a description of the family of crypto algorithm used to generate public key and signature of the cryptographic certificate.
format	CryptoCertificateFormat Enum	0..1	attr	This attribute can be used to provide information about the format used to create the certificate
maximum Length	PositiveInteger	0..1	attr	This attribute represents the ability to define the maximum length of the certificate in bytes.
nextHigher Certificate	CryptoService Certificate	0..1	ref	The reference identifies the next higher certificate in the certificate chain. **Tags:** atp.Splitkey=nextHigherCertificate
serverName Identification	String	0..1	attr	Server Name Indication (SNI) is needed if the IP address hosts multiple servers (on the same port), each of them using a different certificate. If the client sends the SNI to the Server in the client hello, the server looks the SNI up in its certificate list and uses the certificate identified by the SNI.

Table 13.86: CryptoServiceCertificate

Enumeration	**CryptoCertificateAlgorithmFamilyEnum**
Note	This meta-class defies possible cryptographic algorithm families used to create public keys and signatures within the certificate.
Aggregated by	CryptoServiceCertificate.algorithmFamily
Literal	***Description***
ecc	The cryptographic operations in the certificate are executed using elliptic curves (ecc) **Tags:** atp.EnumerationLiteralIndex=2
rsa	The cryptographic operations in the certificate are executed using the RSA approach. **Tags:** atp.EnumerationLiteralIndex=1

Table 13.87: CryptoCertificateAlgorithmFamilyEnum

Enumeration	**CryptoCertificateFormatEnum**
Note	This meta-class defines possible formats of cryptographic certificates.
Aggregated by	CryptoServiceCertificate.format
Literal	***Description***
cvc	The certificate has been created in Card Verifiable Certificate (CVC) format **Tags:** atp.EnumerationLiteralIndex=2
x509	The certificate is created in X.509 format. **Tags:** atp.EnumerationLiteralIndex=1

Table 13.88: CryptoCertificateFormatEnum

[constr_5048] Existence of `TlsCryptoCipherSuite.certificate` and `TlsCryptoCipherSuite.pskIdentity` in the *server* role

Imposition time: `IT_Mani`

⌈**Either**

- the reference to `CryptoServiceCertificate` in the role `TlsCryptoCipherSuite.certificate`
- the aggregation of `TlsPskIdentity` in the role `TlsCryptoCipherSuite.pskIdentity`

shall exist if the `TlsCryptoCipherSuite` is aggregated by `TlsSecureComProps` that has the attribute `category` set to the value `TLS_SERVER`.⌋

In other words, two different approaches are supported by TLS for the handling of key compromise: Pre-shared secret and certificate.

The server may optionally request a certificate from the *client*. If this option is not used then other documented approaches for completing the handshake phase is foreseen for the specific case.

[TPS_MANI_03216] Existence of `TlsCryptoCipherSuite.certificate` and `TlsCryptoCipherSuite.pskIdentity` in the *client* role ⌈The client (`TlsSecureComProps` has set the value of attribute `category` to `TLS_CLIENT`) has the following authentication options:

- the reference to `CryptoServiceCertificate` in the role `TlsCryptoCipherSuite.certificate` exists,
- the aggregation of `TlsPskIdentity` in the role `TlsCryptoCipherSuite.pskIdentity` exists,
- neither one nor the other exists. In this case the handshake is provided on the basis of the server certificate only.

⌋

In the pre-shared key approach the client indicates which key to use by including a `pskIdentity` in the ClientKeyExchange message. To help the client in selecting

which identity to use, the server can provide a pskIdentityHint in the ServerKeyExchange message.

The pre-shared key used for client authentication is identified by the reference TlsCryptoCipherSuite.authenticationKey.

The usage of pskIdentityHints is restricted to TLS 1.2.

[TPS_MANI_03137] ServiceInterfaceElementSecureComConfig is not relevant in the case of TLS communication ⌈The element ServiceInterfaceElementSecureComConfig is not relevant in the case of TLS communication.⌋

[constr_3485] UDP endpoint using DTLS SERVER role can only serve provided service instances

Imposition time: IT_Mani

⌈A ServiceInstanceToMachineMapping that refers to TlsSecureComProps in the role secureComPropsForUdp is only allowed to reference ProvidedApServiceInstances in the role serviceInstance if the TlsSecureComProps has the category TLS_SERVER.⌋

[constr_3486] TCP endpoint using TLS SERVER role can only serve provided service instances

Imposition time: IT_Mani

⌈A ServiceInstanceToMachineMapping that refers to TlsSecureComProps in the role secureComPropsForTcp is only allowed to reference ProvidedApServiceInstances in the role serviceInstance if the TlsSecureComProps has the category TLS_SERVER.⌋

[constr_5260] UDP endpoint using DTLS CLIENT role can only serve required service instances

Imposition time: IT_Mani

⌈A ServiceInstanceToMachineMapping that refers to TlsSecureComProps in the role secureComPropsForUdp is only allowed to reference RequiredApServiceInstances in the role serviceInstance if the TlsSecureComProps has the category TLS_CLIENT.⌋

[constr_5261] TCP endpoint using TLS CLIENT role can only serve required service instances

Imposition time: IT_Mani

⌈A ServiceInstanceToMachineMapping that refers to TlsSecureComProps in the role secureComPropsForTcp is only allowed to reference RequiredApServiceInstances in the role serviceInstance if the TlsSecureComProps has the category TLS_CLIENT.⌋

The reason for [constr_3485], [constr_3486], [constr_5260], [constr_5261] is that the (D)TLS client needs to establish the (D)TLS connection and a TCP/UDP endpoint

that is described by the ServiceInstanceToMachineMapping can only take one role: (D)TLS client or (D)TLS server. If a ServiceInstanceToMachineMapping would act as (D)TLS client and would refer to a ProvidedApServiceInstance then this (D)TLS client would need to establish the (D)TLS connection. But in this case the (D)TLS client would not know to which remote service client a connection needs to be established since different RequiredApServiceInstances may directly call methods of the ProvidedApServiceInstance without any registration.

The same issue exists if the ServiceInstanceToMachineMapping acts as (D)TLS server and refers to RequiredApServiceInstances. The (D)TLS client needs to establish the (D)TLS connection before any messages are exchanged. But the remote service provider has no knowledge that this service consumer wants to call methods over a (D)TLS connection.

[constr_9350] All Provided Service Instances using the same UDP endpoint shall use the same DTLS SERVER configuration settings

Imposition time: IT_Mani

⌈If more than one SomeipServiceInstanceToMachineMapping is referencing a ProvidedSomeipServiceInstance and all these SomeipServiceInstanceToMachineMappings are referencing the same ApApplicationEndpoint in the role udpPort and the same EthernetCommunicationConnector in the role communicationConnector then all these SomeipServiceInstanceToMachineMappings shall reference the same TlsSecureComProps with category TLS_SERVER in the role secureComPropsForUdp.⌋

[constr_9351] All Provided Service Instances using the same TCP endpoint shall use the same TLS SERVER configuration settings

Imposition time: IT_Mani

⌈If more than one SomeipServiceInstanceToMachineMapping is referencing a ProvidedSomeipServiceInstance and all these SomeipServiceInstanceToMachineMappings are referencing the same ApApplicationEndpoint in the role tcpPort and the same EthernetCommunicationConnector in the role communicationConnector then all these SomeipServiceInstanceToMachineMappings shall reference the same TlsSecureComProps with category TLS_SERVER in the role secureComPropsForTcp.⌋

In other words, the mapping of secured and unsecured ProvidedSomeipServiceInstances to the same ApApplicationEndpoint and NetworkEndpoint combination is not allowed. If secured and unsecured ProvidedSomeipServiceInstances would exist on the same endpoint then TLS handshake data or unsecured data could be sent from a client to the server instance. The server would not be able to detect whether the first byte of the received data is a TLS handshake or a SOME/IP message.

13.5.2 Secure Communication over SecOC

AUTOSAR Secure Onboard Communication (SecOC) supports symmetric and asymmetric authentication approaches as well as Authenticated Encryption with Associated Data (AEAD). To configure the SecOC secure protection of a message by a MAC or Signature, the ServiceInterfaceElementSecureComConfig needs to be defined.

This element contains the configuration settings for the individual ServiceInterface elements. In addition, the ServiceInstanceToMachineMapping needs to point to SecOcSecureComProps to configure the endpoint protection that is defined by the Transport Protocol, Port and IP Address.

[constr_3392] ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are mandatory in the case of SecOC communication

Imposition time: IT_Mani

⌈The attributes ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are mandatory in the case of SecOC communication.⌋

[TPS_MANI_03664] ServiceInterfaceElementSecureComConfig.securedRxVerification ⌈The attribute ServiceInterfaceElementSecureComConfig.securedRxVerification defines whether a received SecOC message shall be verified with respect to the security credentials or not.

If the attribute ServiceInterfaceElementSecureComConfig.securedRxVerification is set to false, then security verification shall not be applied and the message payload shall be forwarded to the receivers.

If the attribute ServiceInterfaceElementSecureComConfig.securedRxVerification is set to true or is not defined, then security verification shall be applied.⌋

[constr_3691] Existence of ServiceInterfaceElementSecureComConfig.securedRxVerification

Imposition time: IT_Mani

⌈The attribute ServiceInterfaceElementSecureComConfig.securedRxVerification shall only be defined for a ServiceInterfaceElementSecureComConfig with the following definitions:

- The ServiceInterfaceElementSecureComConfig is aggregated by a RequiredApServiceInstance and defines at least one of the following roles:
  - ServiceInterfaceElementSecureComConfig.event
  - ServiceInterfaceElementSecureComConfig.fieldNotifier
  - ServiceInterfaceElementSecureComConfig.getterReturn

  - ServiceInterfaceElementSecureComConfig.setterReturn
  - ServiceInterfaceElementSecureComConfig.methodReturn
- The ServiceInterfaceElementSecureComConfig is aggregated by a ProvidedApServiceInstance and defines at least one of the following roles:
  - ServiceInterfaceElementSecureComConfig.getterCall
  - ServiceInterfaceElementSecureComConfig.setterCall
  - ServiceInterfaceElementSecureComConfig.methodCall

⌋

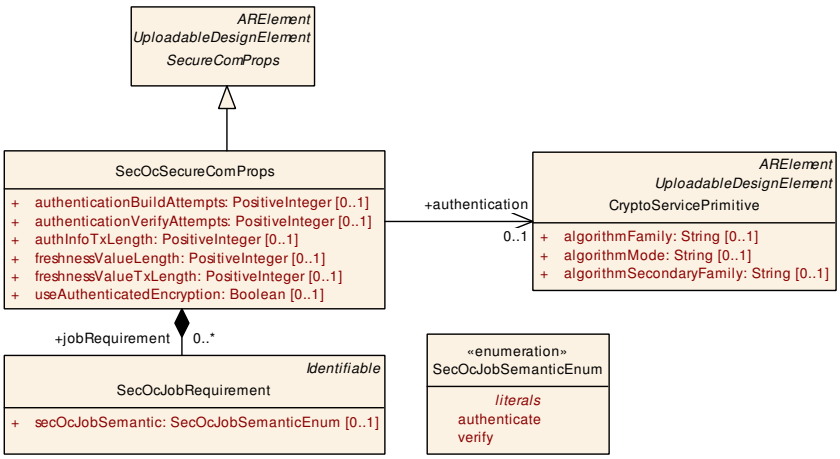


Figure 13.37: Secure Communication over SecOC

[TPS_MANI_03138] SecOC Security Profile ⌈The SecOC security profile is defined by SecOcSecureComProps.category.⌋

Class	**SecOcSecureComProps**			
Note	Configuration of AUTOSAR SecOC. **Tags:** atp.recommendedPackage=SecureComProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *SecureComProps*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authentication	CryptoServicePrimitive	0..1	ref	This reference defines the authentication or authenticated encryption algorithm used for MAC generation and verification or AEAD encryption and decryption. **Tags:** atp.Splitkey=authentication
authentication BuildAttempts	PositiveInteger	0..1	attr	This attribute defines the additional number of authentication build attempts that are to be carried out when the generation of the authentication information failed for a given message. If zero is set then only one authentication attempt is done.

△

Class	**SecOcSecureComProps**			
authentication VerifyAttempts	PositiveInteger	0..1	attr	This attribute defines the additional number of authentication attempts that are to be carried out when the generation of the authentication information failed for a given message. If zero is set than only one authentication attempt is done.
authInfoTx Length	PositiveInteger	0..1	attr	This attribute defines the length in bits of the authentication code to be included in the payload of the authenticated Message.
freshnessValue Length	PositiveInteger	0..1	attr	This attribute defines the complete length in bits of the Freshness Value.
freshnessValue TxLength	PositiveInteger	0..1	attr	This attribute defines the length in bits of the Freshness Value to be included in the payload of the secured message.
jobRequirement	SecOcJobRequirement	*	aggr	Collection of cryptographic job requirements. **Tags:** atp.Splitkey=jobRequirement, job Requirement.shortName
use Authenticated Encryption	Boolean	0..1	attr	A Boolean value that indicates if the payload shall be handled as confidential. If it set to TRUE, Authenticated Encryption with Associated Data (AEAD) shall be applied.

Table 13.89: SecOcSecureComProps

[TPS_MANI_03139] Standardized SecOC Security Profiles ⌈The SecOC security profile that is defined by `SecOcSecureComProps.category` can have the following values that are standardized by AUTOSAR: `PROFILE_01`, `PROFILE_02`, `PROFILE_03`, `PROFILE_04`.⌋

The attribute values for the predefined categories mentioned in [TPS_MANI_03139] are defined in [constr_3325] in [4, TPS System Template].

[constr_5347] Supported value range for attribute `SecOcSecureComProps.authenticationVerifyAttempts`

Imposition time: `IT_Mani`

⌈The supported value range of the primitive attribute `SecOcSecureComProps.authenticationVerifyAttempts` is limited to the interval [0..65535].⌋

[TPS_MANI_03140] Non-Standardized SecOC Security Profiles ⌈The values for the `SecOcSecureComProps.category` mentioned in [TPS_MANI_03139] are standardized and reserved for being used in the way the AUTOSAR standard foresees. In addition, it is possible to use non-standardized values for the `SecOcSecureComProps.category`.⌋

With the `SecOcJobRequirement`, the cryptographic routines that need to be supported can be selected. In the case of SecOC, it can be selected whether the symmetric and/or asymmetric authentication approach is needed.

Class	SecOcJobRequirement			
Note	Requirements for the cryptographic job that need to be executed.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SecOcSecureComProps.jobRequirement			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
secOcJob Semantic	SecOcJobSemantic Enum	0..1	attr	This attribute defines the cryptographic algorithm that needs to be supported.

Table 13.90: SecOcJobRequirement

[constr_10227] Multiplicity of attribute SecOcJobRequirement.secOcJobSemantic

Imposition time: IT_Mani

⌈For each SecOcJobRequirement, the attribute secOcJobSemantic shall exist.⌋

Enumeration	SecOcJobSemanticEnum
Note	List of cryptographic routines supported by SecOC.
Aggregated by	SecOcJobRequirement.secOcJobSemantic
Literal	**Description**
authenticate	Authentication algorithm for Authenticator generation/verification. **Tags:** atp.EnumerationLiteralIndex=0
verify	Asymmetric cryptographic algorithm to generate/verify a signature **Tags:** atp.EnumerationLiteralIndex=1

Table 13.91: SecOcJobSemanticEnum

CAUTION

The constraints marked with the imposition time IT_SysDes defined in chapter [4, TPS System Template] 6.3.2 *"SecuredIPdu"* are valid for this specification as well.

13.5.3 Secure Communication over DDS

The configuration of participant identities for the [46, DDS Security] plugins is supported by the DdsSecureComProps meta-class, which is a specialization of SecureComProps.

The following classes and constraints depict a structural specification of DDS Security deployment configuration. For functional details see section "Secure Communication" of [32, SWS Communication Management].

These modeling elements affect only DDS Security (as defined by [46, DDS Security]) deployments via DdsServiceInstanceToMachineMapping.secureComPropsForDds. For DDS Transport Security over TCP (TLS) and UDP (DTLS) configuration see ServiceInstanceToMachineMapping.secureComPropsForTcp and ServiceInstanceToMachineMapping.secureComPropsForUdp, respectively.

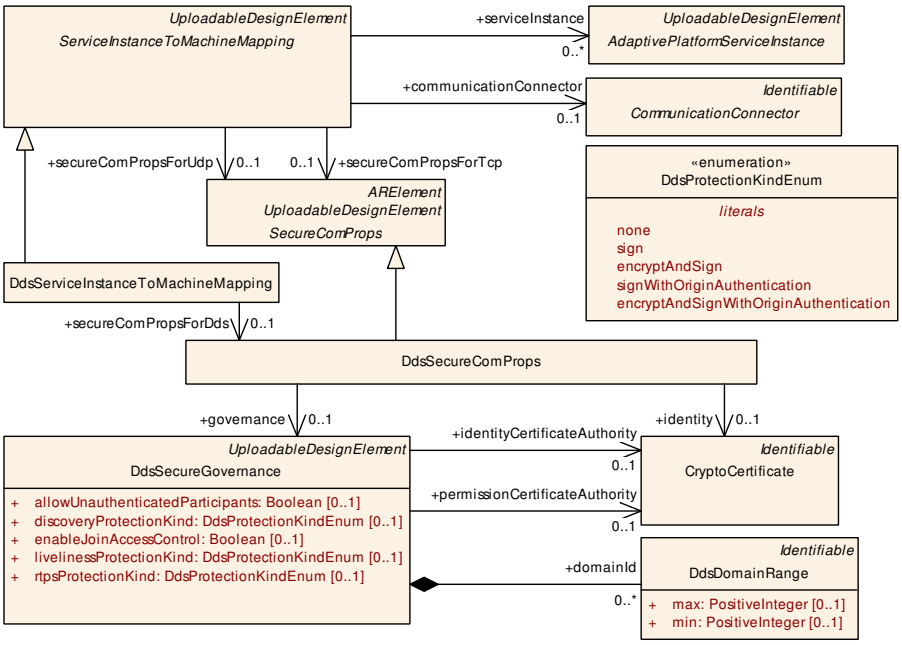


Figure 13.38: `DdsSecureComProps` at the `DdsServiceInstanceToMachineMapping`

Class	**DdsSecureComProps**			
Note	Identity and governance information of participants in case of DDS Security. **Tags:** atp.recommendedPackage=SecureComProps This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *SecureComProps*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
governance	DdsSecureGovernance	0..1	ref	This attribute defines general DDS Security communication properties applicable to the DDS domain(s) in which the subject operates. **Tags:** atp.Splitkey=governance atp.Status=candidate
identity	CryptoCertificate	0..1	ref	This attribute defines the cryptographic identity of the subject. **Tags:** atp.Splitkey=identity

Table 13.92: DdsSecureComProps

[constr_3678] Existence of attributes for `DdsSecureComProps`

Status: DRAFT
Imposition time: `IT_Mani`

⌈The following attributes of `DdsSecureComProps` shall exist:

- `identity`
- `governance`

⌋

[TPS_MANI_03661] Configuration of Governance in DDS Security

Status: DRAFT

⌈The `DdsSecureGovernance` meta-class defines domain-wide rules determining relevant certificate authorities (for participant identities and permission manifests), constraints related to how participants may join such domain(s) and protection levels associated with each kind of communication (none, sign, encrypt+sign, with or without origin authentication).⌋

Class	**DdsSecureGovernance**			
Note	Configuration of DDS Security for all applications joining a specific set of DDS Domains. **Tags:** atp.Status=candidate atp.recommendedPackage=DdsSecureGovernances This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allowUnauthenticated Participants	Boolean	0..1	attr	Defines whether unauthenticated participants can join this domain. **Tags:** atp.Status=candidate
discovery ProtectionKind	DdsProtectionKind Enum	0..1	attr	Defines the kind of cryptographic transformation to apply in DDS discovery communication. **Tags:** atp.Status=candidate
domainId	DdsDomainRange	*	aggr	Set of domains to be covered by this property set. **Tags:** atp.Splitkey=domainId, domainId.shortName atp.Status=candidate
enableJoin AccessControl	Boolean	0..1	attr	Defines whether access control is to be enforced upon joining this domain. **Tags:** atp.Status=candidate
identity Certificate Authority	CryptoCertificate	0..1	ref	Certificate representing the identity certificate authority applicable to the domain(s) specified by domainsIds. **Tags:** atp.Splitkey=identityCertificateAuthority atp.Status=candidate
liveliness ProtectionKind	DdsProtectionKind Enum	0..1	attr	Defines the kind of cryptographic transformation to apply in DDS liveliness communication. **Tags:** atp.Status=candidate
permission Certificate Authority	CryptoCertificate	0..1	ref	Certificate representing the permissions certificate authority applicable to the domain(s) specified by domainsIds. **Tags:** atp.Splitkey=permissionCertificateAuthority atp.Status=candidate
rtpsProtection Kind	DdsProtectionKind Enum	0..1	attr	Defines the kind of cryptographic transformation to apply to whole DDS RTPS. **Tags:** atp.Status=candidate

Table 13.93: DdsSecureGovernance

Class	**DdsDomainRange**			
Note	DDS Domain ID range. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DdsSecureGovernance.domainId			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
max	PositiveInteger	0..1	attr	Upper bound of the DdsDomainRange.
min	PositiveInteger	0..1	attr	Lower bound of the DdsDomainRange.

Table 13.94: DdsDomainRange

Enumeration	**DdsProtectionKindEnum**
Note	Supported cryptographic transformations (extended). This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	DdsSecureGovernance.discoveryProtectionKind, DdsSecureGovernance.livelinessProtectionKind, DdsSecureGovernance.rtpsProtectionKind, DdsTopicAccessRule.dataProtectionKind, DdsTopic AccessRule.metadataProtectionKind
Literal	***Description***
encryptAndSign	encryption and MAC transformations (in that precise order) are applied **Tags:** atp.EnumerationLiteralIndex=2
encryptAndSign WithOrigin Authentication	similar to "EncryptAndSign" but with additional authentication codes produced under different secret keys, which prevents receiving peers from impersonating a specific sender **Tags:** atp.EnumerationLiteralIndex=4
none	no transformation is applied **Tags:** atp.EnumerationLiteralIndex=0
sign	Message Authentication Code (MAC) is applied, no encryption **Tags:** atp.EnumerationLiteralIndex=1
signWithOrigin Authentication	similar to "sign" but with additional authentication codes produced under different secret keys, which prevents receiving peers from impersonating a specific sender **Tags:** atp.EnumerationLiteralIndex=3

Table 13.95: DdsProtectionKindEnum

[constr_3679] Existence of attributes for `DdsSecureGovernance`

Status: DRAFT
Imposition time: `IT_Mani`

⌈The following attributes of `DdsSecureGovernance` shall exist:

- at least one `domainId`
- `identityCertificateAuthority`
- `permissionCertificateAuthority`
- `allowUnauthenticatedParticipants`
- `enableJoinAccessControl`
- `discoveryProtectionKind`
- `livelinessProtectionKind`
- `rtpsProtectionKind`

⌋

[constr_3682] Values of DdsDomainRange.min and DdsDomainRange.max

Imposition time: IT_Mani

⌈The value of DdsDomainRange.min shall be less than or equal to the value of DdsDomainRange.max.⌋

14 Raw Data Stream Manifest

The Raw Data Stream Manifest has initially been introduced for the support of IP based (TCP/UDP) Ethernet communication.

Later, further protocols have been added and the original (IP based) meta-class naming might not be expressive enough to reflect the dedicated IP binding. The descriptions have been updated to clarify to which specific network binding the respective meta-classes contribute.

14.1 Raw Data Stream Deployment

[TPS_MANI_01285] Purpose of meta-class `RawDataStreamDeployment` ⌈Meta-class `RawDataStreamDeployment` has the ability to further qualify an existing `AbstractRawDataStreamInterface` on deployment level.⌋



Figure 14.1: Modeling of the `RawDataStreamDeployment`

Class	**RawDataStreamDeployment**			
Note	This meta-class represents the ability to model deployment-level information for a raw data stream **Tags:** atp.recommendedPackage=RawDataStreamDeployments This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
rawDataStream Interface	AbstractRawData StreamInterface	0..1	ref	This reference identifies the corresponding RawData StreamInterface, **Tags:** atp.Splitkey=rawDataStreamInterface

Table 14.1: RawDataStreamDeployment

14.2 Raw Data Stream Mapping

[TPS_MANI_01287] Semantics of `RawDataStreamMapping` ⌈On the target-configuration side, the access to a raw data stream requires the provision of actual transport for the raw data.

It is possible to implement the raw data transport on top of various technologies.

Therefore, abstract meta-class RawDataStreamMapping exists to provide the basic ability to map to an RPortPrototype and to a Process.

The mapping to a concrete transport technology is left to sub-classes of RawDataStreamMapping.

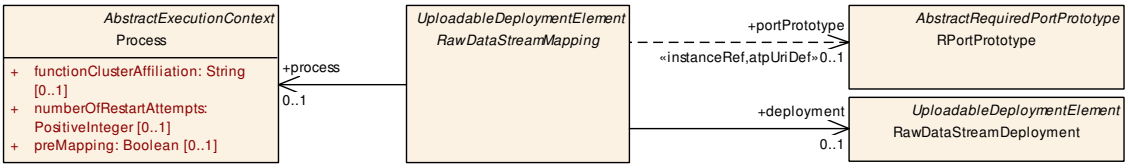


Figure 14.2: Modeling of the RawDataStreamMapping

Class	***RawDataStreamMapping*** (abstract)			
Note	This meta-class acts as an abstract base class for mapping raw data streams to the application software. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	*EthernetMacRawDataStreamMapping*, *EthernetRawDataStreamMapping*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
deployment	RawDataStream Deployment	0..1	ref	This reference identifies the applicable RawDataStream Deployment. **Tags:** atp.Splitkey=deployment
portPrototype	RPortPrototype	0..1	iref	Reference to a specific PortPrototype that represents the raw data stream to the application. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=portPrototype, portPrototype.context RootSwComponentPrototype, portPrototype.context ComponentPrototype, portPrototype.targetRPort Prototype **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	Reference to the Process in which the Executable that contains the SoftwareComponent and the referenced Port Prototype is executed. **Tags:** atp.Splitkey=process

Table 14.2: RawDataStreamMapping

14.3 Raw Data Streams over Ethernet using IP based Protocol

The configuration of raw data streams over Ethernet using an IP based protocol is done by means of the subclasses of meta-class AbstractRawDataStreamEthernetCredentials as well as meta-class EthernetRawDataStreamMapping.

The obvious requirement to the configuration of the Ethernet credentials is that both ends of the communication have to be configured sufficiently, such that a connection between the two ends can be established.

The modeling of the configuration of Ethernet credentials uses different concepts for the description of the **local** end vs. the description of the **remote** end, see [TPS_MANI_01354].

[TPS_MANI_01354] Rationale for the existence of meta-class `AbstractRawDataStreamEthernetCredentials` ⌈On the *AUTOSAR adaptive platform*, the configuration of Ethernet credentials is typically modeled by means of the combination of `NetworkEndpoint` and `ApApplicationEndpoint`.

This approach is also used for the unicast raw data stream communication, but only for the **local** configuration of unicast Ethernet credentials.

The credentials of the **remote** end (unicast and multicast) are configured in a simplified way by means of sub-classes of meta-class `AbstractRawDataStreamEthernetCredentials`.

This approach also supports the configuration of credentials for a remote end that is not running an AUTOSAR stack and that is therefore not contributing its credentials inside an AUTOSAR model in a more approachable way.⌋

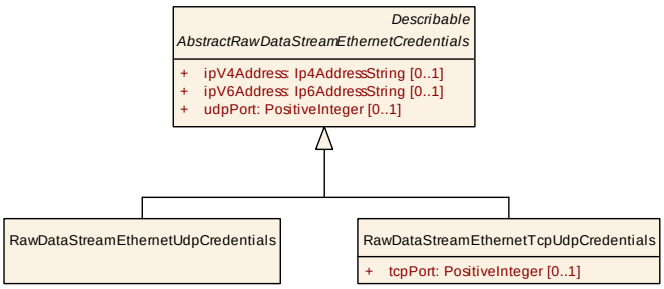


Figure 14.3: Modeling of the `AbstractRawDataStreamEthernetCredentials`

From the point of view of the application software, and in the case of raw data stream communication, both communication ends, i.e. the client side **and** the server side, are implemented using an `RPortPrototype`.

This (from an AUTOSAR perspective) seemingly counter-intuitive aspect is also depicted in Figure 14.4, i.e. `RawDataStreamMapping` only defines a reference to an `RPortPrototype`.

Meta-class `AbstractRawDataStreamEthernetCredentials` and its subclasses (see Figure 14.3) have the ability to define IP addresses (either V4 or V6) and transport protocol (UDP and/or TCP).

[constr_10076] Existence of `RawDataStreamEthernetUdpCredentials.udpPort`

Imposition time: `IT_Mani`

⌈In the context of `RawDataStreamEthernetUdpCredentials`, the attribute `udpPort` shall exist.⌋

Class	**AbstractRawDataStreamEthernetCredentials** (abstract)			
Note	This meta-class serves as an abstract base class for the configuration of network credentials. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Describable*			
Subclasses	RawDataStreamEthernetTcpUdpCredentials, RawDataStreamEthernetUdpCredentials			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ipV4Address	Ip4AddressString	0..1	attr	This attribute describes the IP V4 address of the remote server.
ipV6Address	Ip6AddressString	0..1	attr	This attribute describes the IP V6 address of the remote server.
udpPort	PositiveInteger	0..1	attr	This attribute represents the configuration of a UDP port number.

Table 14.3: AbstractRawDataStreamEthernetCredentials

Class	**RawDataStreamEthernetUdpCredentials**			
Note	This-meta-class represents the ability to create a configuration of network credentials for a raw data stream connection over UDP. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractRawDataStreamEthernetCredentials*, *Describable*			
Aggregated by	EthernetRawDataStreamRemoteClientConfig.multicastCredentials, EthernetRawDataStreamRemote ClientConfig.unicastUdpCredentials, EthernetRawDataStreamRemoteServerConfig.multicastCredentials			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 14.4: RawDataStreamEthernetUdpCredentials

Class	**RawDataStreamEthernetTcpUdpCredentials**			
Note	This-meta-class represents the ability to create a configuration of network credentials for a raw data stream connection over TCP and UDP (inherited from base class). This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractRawDataStreamEthernetCredentials*, *Describable*			
Aggregated by	EthernetRawDataStreamRemoteServerConfig.unicastCredentials			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
tcpPort	PositiveInteger	0..1	attr	This attribute represents the configuration of a TCP port number.

Table 14.5: RawDataStreamEthernetTcpUdpCredentials

[constr_10077] Existence of `ipV4Address` and `ipV6Address` within `AbstractRawDataStreamEthernetCredentials`

Imposition time: `IT_Mani`

⌈Within the context of a `AbstractRawDataStreamEthernetCredentials`, either the attribute `ipV4Address` or the attribute `ipV6Address` shall exist.⌋

[constr_10078] Existence of `RawDataStreamEthernetTcpUdpCredentials.tcpPort` and `udpPort`

Imposition time: `IT_Mani`

⌈In the context of a `RawDataStreamEthernetTcpUdpCredentials`, either the attribute `tcpPort` **or** `udpPort` shall exist.⌋

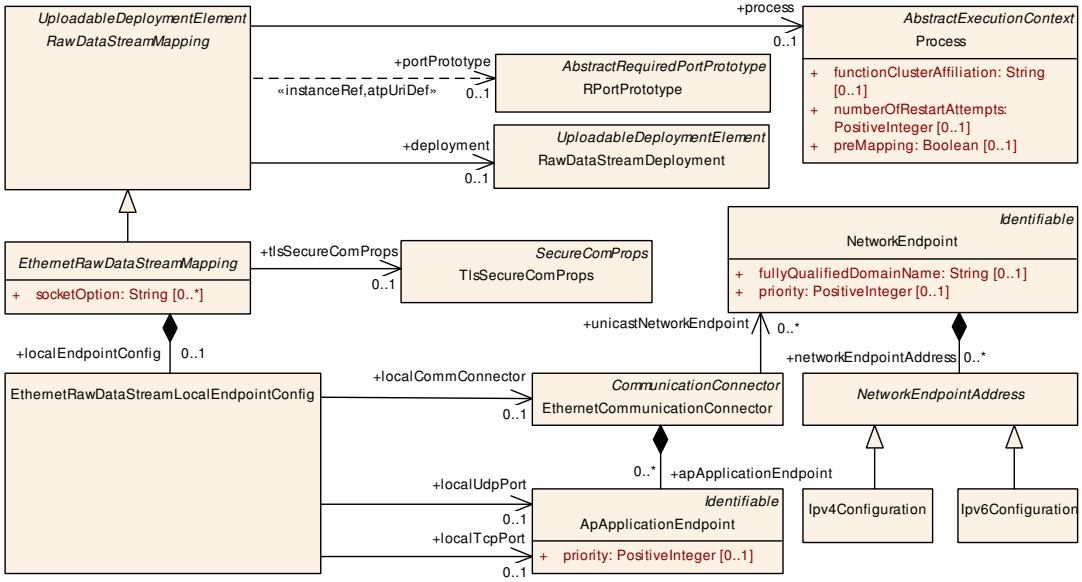


Figure 14.4: Modeling of the `EthernetRawDataStreamMapping`

[TPS_MANI_01355] Definition of local Ethernet credentials ⌈The definition of the **local** Ethernet credentials is done by means of the aggregation of meta-class `EthernetRawDataStreamLocalEndpointConfig` in the role `localEndpoint-Config`.

`EthernetRawDataStreamLocalEndpointConfig`, in turn, defines the following aggregations for the configuration of local Ethernet credentials:

- a reference to `EthernetCommunicationConnector` in the role `localComm-Connector`, that in turn refers to `NetworkEndpoint`.
- references to `ApApplicationEndpoint` in the roles
  - `EthernetRawDataStreamLocalEndpointConfig.localUdpPort`
  - `EthernetRawDataStreamLocalEndpointConfig.localTcpPort`

⌋

The configuration of the corresponding remote end depends on whether the remote end represents a server (see [TPS_MANI_01356] in Chapter 14.3.1) or a client (see [TPS_MANI_01357] in Chapter 14.3.2).

Class	***EthernetRawDataStreamMapping*** (abstract)
Note	This meta-class serves as the abstract bases class for the ability to map a PortPrototype to a Ethernet-based communication channel. This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*
Subclasses	EthernetRawDataStreamClientMapping, EthernetRawDataStreamServerMapping

▽

△

Class	***EthernetRawDataStreamMapping*** (abstract)			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
localEndpoint Config	EthernetRawData StreamLocalEndpoint Config	0..1	aggr	This aggregation is used to configure the credentials of the endpoint. **Tags:** atp.Splitkey=localEndpointConfig
socketOption	String	*	attr	This attribute represents the ability to specify non-formal socket options that might only be valid for specific platforms. AUTOSAR does not define a standardized meaning for the possible values of this attribute.
tlsSecureCom Props	TlsSecureComProps	0..1	ref	This reference provides the ability to define TLS-related properties for the enclosing SocketRawDataStream Mapping. **Tags:** atp.Splitkey=tlsSecureComProps

Table 14.6: EthernetRawDataStreamMapping

[constr_10079] Existence of `EthernetRawDataStreamMapping.localTcpPort` and `localUdpPort`

Imposition time: `IT_Mani`

⌈In the context of a `EthernetRawDataStreamMapping.localCommConnector`, only **one** attribute out of

- `localTcpPort`
- `localUdpPort`

shall exist.⌋

Class	**EthernetRawDataStreamLocalEndpointConfig**			
Note	This meta-class has the ability to act as a wrapper for the configuration of the remote endpoint in the context of a raw data stream mapping. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	*EthernetRawDataStreamMapping*.localEndpointConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
localComm Connector	EthernetCommunication Connector	0..1	ref	This attribute represents the CommunicationConnector taken for socket-based data communication. **Tags:** atp.Splitkey=localCommConnector
localTcpPort	ApApplicationEndpoint	0..1	ref	This aggregation represents the configuration of a local TCP port number. **Tags:** atp.Splitkey=localTcpPort
localUdpPort	ApApplicationEndpoint	0..1	ref	This aggregation represents the configuration of a local unicast UDP port number. **Tags:** atp.Splitkey=localUdpPort

Table 14.7: EthernetRawDataStreamLocalEndpointConfig

14.3.1 Ethernet Raw Data Stream Client Mapping

A raw data stream client may be prepared to receive data via unicast or multicast communication, but it shall only send requests via unicast to exactly one server.

[TPS_MANI_01356] Definition of remote server's Ethernet credentials ⌈The definition of the **remote** server's Ethernet credentials is done by means of attributes of meta-class `EthernetRawDataStreamRemoteServerConfig`:

multicast If multicast configuration shall be supported, then the aggregation in the role `EthernetRawDataStreamRemoteClientConfig.multicastCredentials` shall exist and the attribute `udpPort` shall be configured.

unicast If unicast configuration shall be supported, then the aggregation in the role `EthernetRawDataStreamRemoteClientConfig.unicastUdpCredentials` shall exist and either the attribute `udpPort` **or** attribute `tcpPort` shall be configured.

⌋

The mutual existence of attributes `udpPort` and `tcpPort` is also addressed by [constr_10078].

It is also possible for the client to use ephemeral ports for the communication. In this case, the `EthernetRawDataStreamLocalEndpointConfig.localTcpPort` – by convention – would be configured to value 0.

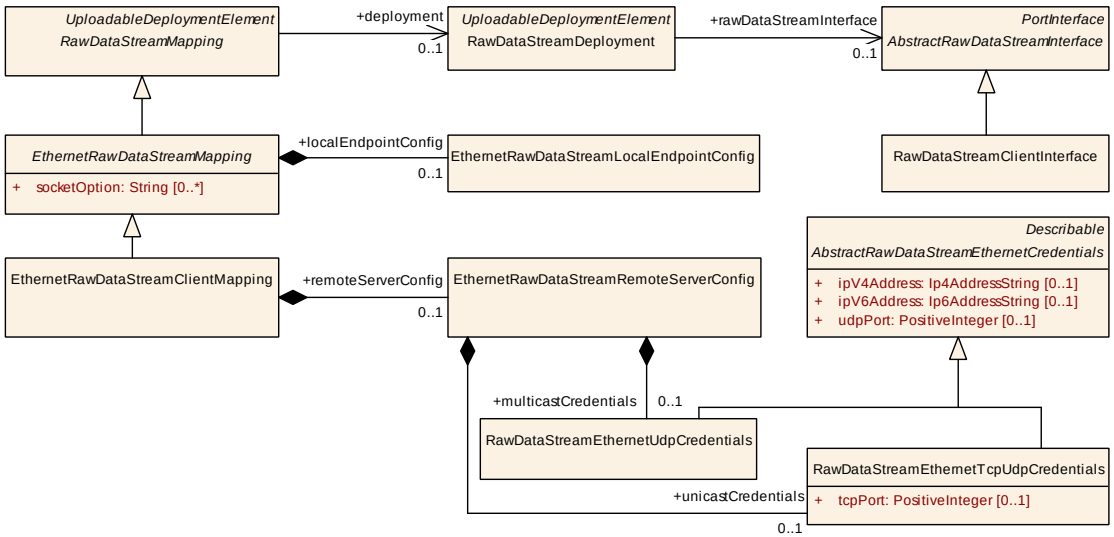


Figure 14.5: Modeling of the `EthernetRawDataStreamClientMapping`

Class	**EthernetRawDataStreamClientMapping**			
Note	This meta-class represents the ability to map a client PortPrototype to a Ethernet-based communication channel. **Tags:** atp.recommendedPackage=RawDataStreamingMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *EthernetRawDataStreamMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *Uploadable DeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
remoteServer Config	EthernetRawData StreamRemoteServer Config	0..1	aggr	This aggregation is used to configure the credentials of the remote server. **Tags:** atp.Splitkey=remoteServerConfig

Table 14.8: EthernetRawDataStreamClientMapping

Class	**EthernetRawDataStreamRemoteServerConfig**			
Note	This meta-class has the ability to act as a wrapper for the configuration of the remote server in the context of a raw data stream client mapping. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	EthernetRawDataStreamClientMapping.remoteServerConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
multicast Credentials	RawDataStream EthernetUdpCredentials	0..1	aggr	This aggregation represents the configuration of multicast credentials for communication with a remote raw data stream server. **Tags:** atp.Splitkey=multicastCredentials
unicast Credentials	RawDataStream EthernetTcpUdp Credentials	0..1	aggr	This meta-class represents the ability to map a server PortPrototype to a Ethernet-based communication channel. **Tags:** atp.Splitkey=unicastCredentials

Table 14.9: EthernetRawDataStreamRemoteServerConfig

14.3.2 Ethernet Raw Data Stream Server Mapping

A raw data stream server shall *either* communicate via unicast *or* via multicast, as expressed in [constr_10086]. A switch between the two approaches depending on the evaluation of conditions at runtime, as possible in the SOME/IP communication, is not supported for raw data streams.

[TPS_MANI_01357] Definition of remote client's Ethernet credentials ⌈The definition of the **remote** client's Ethernet credentials is done by means of attributes of meta-class `EthernetRawDataStreamRemoteClientConfig`:

multicast If multicast configuration shall be supported, then the aggregation in the role `EthernetRawDataStreamRemoteClientConfig.multicastCredentials` shall exist and the attribute `udpPort` shall be configured.

unicast If unicast configuration shall be supported, then the aggregation in the role `EthernetRawDataStreamRemoteClientConfig.unicastUdpCredentials` shall exist and attribute `udpPort` shall be configured.

⌋

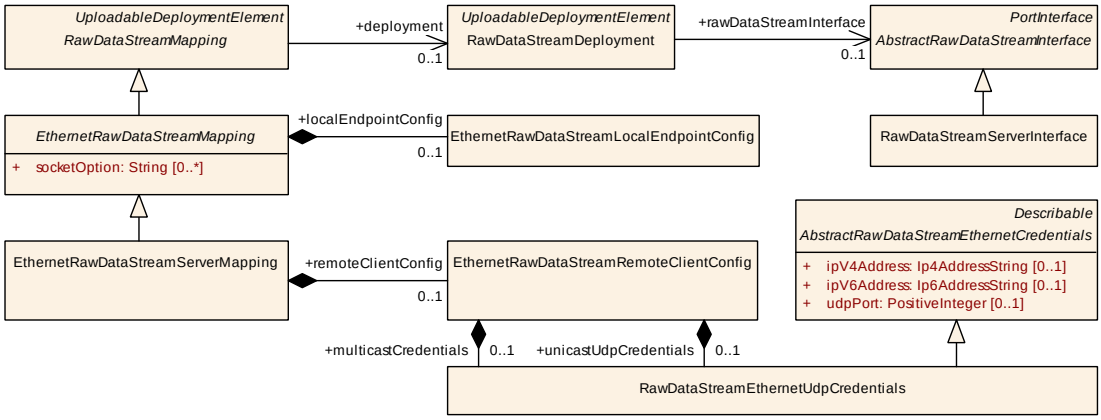


Figure 14.6: Modeling of the `EthernetRawDataStreamServerMapping`

A raw data stream server does not need to configure a TCP port of the client, because the information about the TCP port of the client is conveyed as part of the TCP protocol.

[constr_10086] Existence of `unicastUdpCredentials` and `multicastCredentials` in the context of a `EthernetRawDataStreamServerMapping`

Imposition time: `IT_Mani`

⌈In the context of a `EthernetRawDataStreamServerMapping`, only one aggregation out of

- `remoteClientConfig.multicastCredentials`
- `remoteClientConfig.unicastUdpCredentials`

shall exist.⌋

Class	**EthernetRawDataStreamServerMapping**			
Note	This meta-class represents the ability to map a server PortPrototype to a Ethernet-based communication channel. **Tags:** atp.recommendedPackage=RawDataStreamingMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *EthernetRawDataStreamMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *Uploadable DeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	EthernetRawDataStreamServerMapping			
remoteClient Config	EthernetRawData StreamRemoteClient Config	0..1	aggr	This aggregation is used to configure the credentials of the remote client. **Tags:** atp.Splitkey=remoteClientConfig

Table 14.10: EthernetRawDataStreamServerMapping

Class	EthernetRawDataStreamRemoteClientConfig			
Note	This meta-class has the ability to act as a wrapper for the configuration of the remote server in the context of a raw data stream client mapping. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	EthernetRawDataStreamServerMapping.remoteClientConfig			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
multicast Credentials	RawDataStream EthernetUdpCredentials	0..1	aggr	This aggregation represents the configuration of multicast credentials for communication with a remote raw data stream client. **Tags:** atp.Splitkey=multicastCredentials
unicastUdp Credentials	RawDataStream EthernetUdpCredentials	0..1	aggr	This aggregation represents the configuration of a remote raw data stream client that communicates via unicast over UDP. **Tags:** atp.Splitkey=unicastUdpCredentials

Table 14.11: EthernetRawDataStreamRemoteClientConfig

Examples for the configuration of raw data streams can be found in Chapter B.9.

14.4 Raw Data Streams over Ethernet using MAC Layer Protocol

The abstract class `EthernetMacRawDataStreamMapping` defines attributes applicable to MAC-based Ethernet communication.

[constr_3774] Existence of `EthernetMacRawDataStreamMapping.localCommConnector`

Status: DRAFT
Imposition time: `IT_Mani`

⌈The reference `EthernetMacRawDataStreamMapping.localCommConnector` shall exist.⌋

Class	*EthernetMacRawDataStreamMapping* (abstract)
Note	This meta-class serves as the abstract bases class for the ability to map a PortPrototype to an Ethernet-Mac-layer based communication. **Tags:** atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.

▽

△

Class	**EthernetMacRawDataStreamMapping** (abstract)			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	*IEEE1722RawDataStreamMapping*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
localComm Connector	EthernetCommunication Connector	0..1	ref	This reference represents the CommunicationConnector taken for Mac-based data communication. **Tags:** atp.Splitkey=localCommConnector atp.Status=candidate
socketOption	String	*	attr	This attribute represents the ability to specify non-formal socket options that might only be valid for specific platforms. AUTOSAR does not define a standardized meaning for the possible values of this attribute. **Tags:** atp.Status=candidate

Table 14.12: EthernetMacRawDataStreamMapping

14.4.1 Raw Data Streams over Ethernet using IEEE1722 Protocol

The [19, IEEE 1722] protocol is capable of transporting a variety of content: audio and video streams, clock reference streams, control data streams.

The

- `IEEE1722RawDataStreamConsumerMapping` and
- `IEEE1722RawDataStreamProducerMapping`

are used to bind IEEE1722 streams to `RPortPrototype`s typed by an

- `IEEE1722RawDataStreamConsumerInterface` and
- `IEEE1722RawDataStreamProducerInterface`,

respectively. The binding uses the `IEEE1722TpConnection` as defined in the [4, TPS System Template].

The actual stream Id is defined by the two attributes `IEEE1722TpConnection.macAddressStreamId` and `IEEE1722TpConnection.uniqueStreamId`. They are concatenated according to the [4, TPS System Template] as defined in [TPS_SYST_03098].

By referring to a specific `IEEE1722TpConnection`, the corresponding IEEE1722 stream is mapped to the `RPortPrototype`. Any messages sent (`IEEE1722RawDataStreamProducerMapping`) or received (`IEEE1722RawDataStreamConsumerMapping`) are handed through this `RPortPrototype`. Which parts of the messages are passed through the APIs on the `RPortPrototype` is defined in [20, SWS Raw Data Stream].

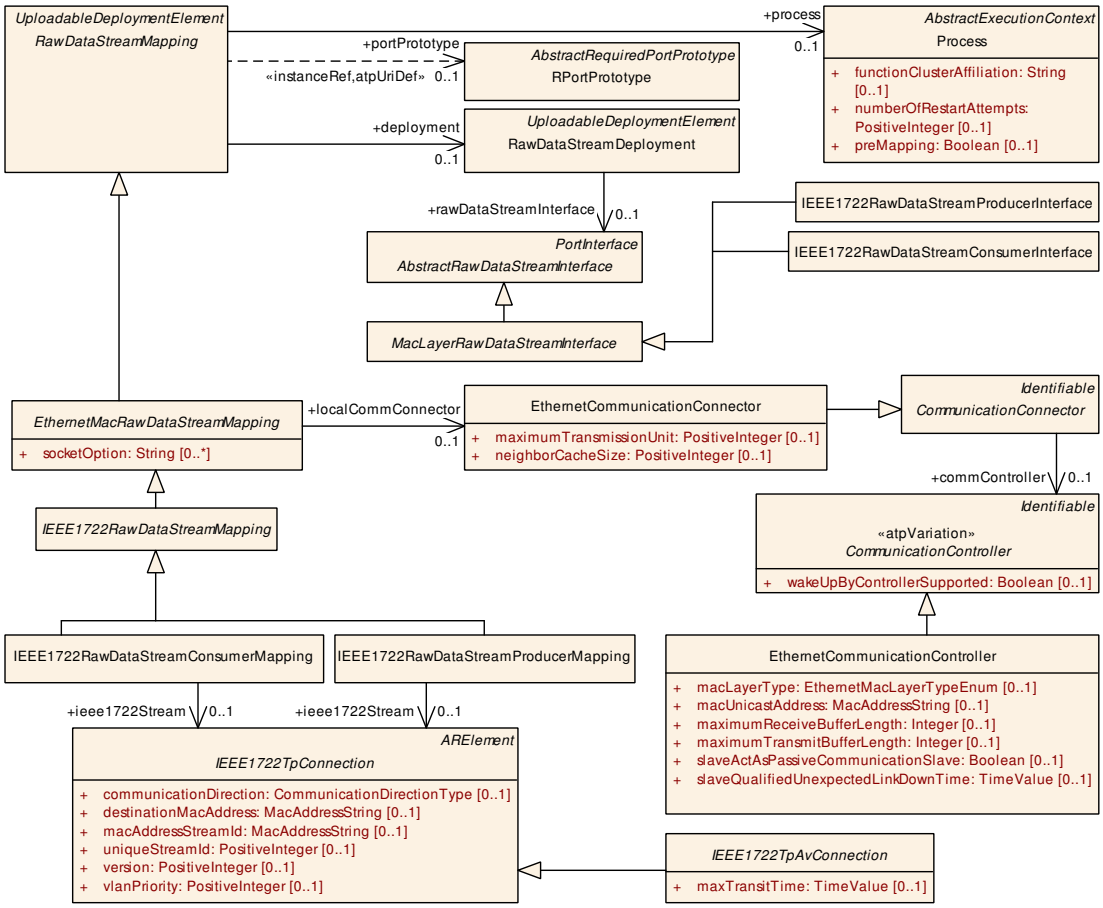


Figure 14.7: Modeling of the IEEE1722 Raw Data Stream mapping

[TPS_MANI_03675] Consuming an IEEE1722 stream

Status: DRAFT

⌈The `IEEE1722RawDataStreamConsumerMapping` defines that all received payload of the referenced `ieee1722Stream IEEE1722TpConnection` shall be consumed at the mapped `IEEE1722RawDataStreamConsumerInterface` (defined via `IEEE1722RawDataStreamConsumerMapping.deployment.RawDataStreamDeployment.rawDataStreamInterface`).⌋

[TPS_MANI_03676] Producing an IEEE1722 stream

Status: DRAFT

⌈The `IEEE1722RawDataStreamProducerMapping` defines that all payload provided to the mapped `IEEE1722RawDataStreamProducerInterface` (defined via `IEEE1722RawDataStreamProducerMapping.deployment.RawDataStreamDeployment.rawDataStreamInterface`) shall be sent to the `ieee1722Stream IEEE1722TpConnection`.⌋

Class	IEEE1722RawDataStreamConsumerMapping			
Note	This meta-class represents the ability to map a consumer PortPrototype to an Ethernet-Mac-layer IEEE1722 based communication. **Tags:** atp.Status=candidate atp.recommendedPackage=RawDataStreamingMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *EthernetMacRawDataStreamMapping*, *IEEE1722RawDataStreamMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ieee1722Stream	IEEE1722TpConnection	0..1	ref	Reference to the IEEE1722TpConnection. **Tags:** atp.Splitkey=ieee1722Stream atp.Status=candidate

Table 14.13: IEEE1722RawDataStreamConsumerMapping

[constr_3775] Existence of IEEE1722RawDataStreamConsumerMapping.ieee1722Stream

Status: DRAFT

Imposition time: IT_Mani

⌈The reference IEEE1722RawDataStreamConsumerMapping.ieee1722Stream shall exist.⌋

Class	IEEE1722RawDataStreamProducerMapping			
Note	This meta-class represents the ability to map a producer PortPrototype to an Ethernet-Mac-layer IEEE1722 based communication. **Tags:** atp.Status=candidate atp.recommendedPackage=RawDataStreamingMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *EthernetMacRawDataStreamMapping*, *IEEE1722RawDataStreamMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ieee1722Stream	IEEE1722TpConnection	0..1	ref	Reference to the IEEE1722TpConnection. **Tags:** atp.Splitkey=ieee1722Stream atp.Status=candidate

Table 14.14: IEEE1722RawDataStreamProducerMapping

[constr_3776] Existence of IEEE1722RawDataStreamProducerMapping.ieee1722Stream

Status: DRAFT

Imposition time: IT_Mani

⌈The reference IEEE1722RawDataStreamProducerMapping.ieee1722Stream shall exist.⌋

Class	***IEEE1722RawDataStreamMapping*** (abstract)			
Note	This meta-class serves as the abstract bases class for the ability to map a PortPrototype to an IEEE1722 based communication. **Tags:** atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *EthernetMacRawDataStreamMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *Uploadable DeploymentElement*, *UploadablePackageElement*			
Subclasses	IEEE1722AcfBusPartRawDataStreamConsumerMapping, IEEE1722AcfBusRawDataStreamConsumer Mapping, IEEE1722RawDataStreamConsumerMapping, IEEE1722RawDataStreamProducerMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 14.15: IEEE1722RawDataStreamMapping

14.4.2 Raw Data Streams for Control Messages over Ethernet using IEEE1722 ACF Protocol

The IEEE1722 ACF Protocol according to [19, IEEE 1722-2016] supports two kinds of message formats:

- Non-Time-Synchronous Control Format, where no presentation time is sent, and
- Time-Synchronous Control Format, where a presentation time is sent in the message.

The composition of the IEEE1722 ACF message is in control of the application producing the messages. Especially the accumulation of several CAN and/or LIN frames into one IEEE1722 ACF message has to be handled by the application.

Whether Non-Time-Synchronous Control Format or Time-Synchronous Control Format is actually used is defined by the attribute `IEEE1722TpAcfConnection.acfMaxTransitTime`.

[TPS_MANI_03677] Producing a Non-Time-Synchronous Control Format messages

Status: DRAFT

⌈If an `IEEE1722 ACF stream` message shall be sent and the attribute `IEEE1722TpAcfConnection.acfMaxTransitTime` is not defined, then the `IEEE1722 ACF stream` message shall be sent according to the “Non-Time-Synchronous Control Format” of [19, IEEE 1722-2016].⌋

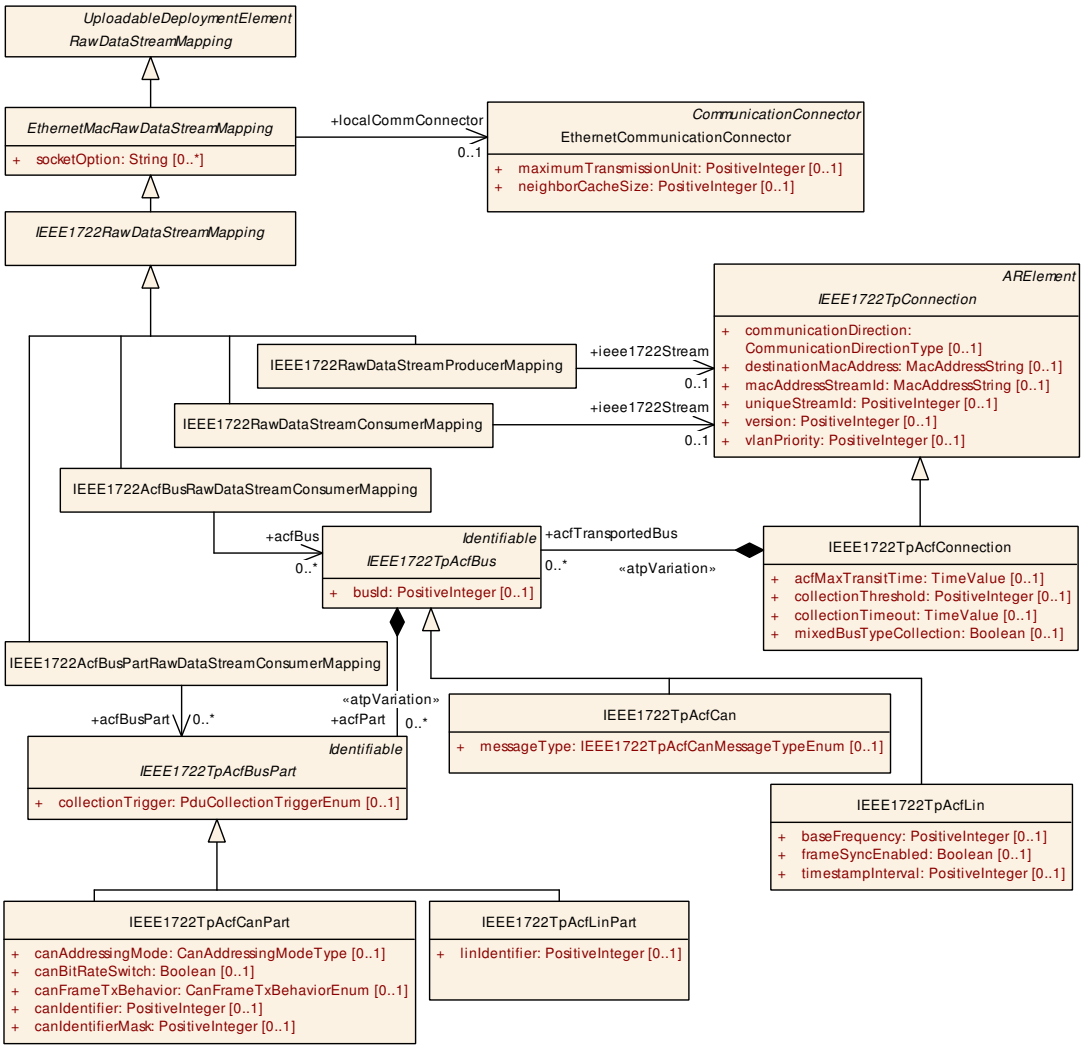


Figure 14.8: Modeling of the IEEE1722 Raw Data Stream ACF mapping

[TPS_MANI_03678] Producing a Time-Synchronous Control Format messages

Status: DRAFT

⌈If an `IEEE1722 ACF stream` message shall be sent and the attribute `IEEE1722TpAcfConnection.acfMaxTransitTime` is defined, then the `IEEE1722 ACF stream` message shall be sent according to the “Time-Synchronous Control Format” of [19, IEEE 1722-2016].

The value used for the presentation time shall be calculated by taking the current time from the `GlobalTimeDomain` referenced by `IEEE1722TpConnection.globalTimeDomain` and adding the value `IEEE1722TpAcfConnection.acfMaxTransitTime`.⌋

For the reception of IEEE1722 ACF messages there are several filters available to reduce the number of indicated CAN and/or LIN frames to the application.

[TPS_MANI_03679] Consuming all IEEE1722 ACF messages of an IEEE1722 ACF stream

Status: DRAFT

⌈If all CAN and/or LIN frames contained in a `IEEE1722 ACF stream` shall be consumed by the application, then an `IEEE1722RawDataStreamConsumerMapping` shall be defined to indicate that the whole `IEEE1722 ACF stream` shall be available to the application.⌋

[TPS_MANI_03680] Consuming a set of buses from an IEEE1722 ACF stream

Status: DRAFT

⌈If all frames from a set of CAN and/or LIN buses contained in an `IEEE1722 ACF stream` shall be consumed by the application, then an `IEEE1722AcfBusRawDataStreamConsumerMapping` shall be defined to indicate that all frames of the referenced `IEEE1722TpAcfBus`es (`IEEE1722AcfBusRawDataStreamConsumerMapping.acfBus`) shall be available to the application.⌋

[TPS_MANI_03681] Consuming a set of frames from an IEEE1722 ACF stream

Status: DRAFT

⌈If only a dedicated set of frames contained in an `IEEE1722 ACF stream` shall be consumed by the application, then an `IEEE1722AcfBusPartRawDataStreamConsumerMapping` shall be defined to indicate that only the frames (`IEEE1722TpAcfBusPart`s) referenced by `IEEE1722AcfBusPartRawDataStreamConsumerMapping.acfBusPart` shall be available to the application.⌋

Class	**IEEE1722AcfBusRawDataStreamConsumerMapping**			
Note	This meta-class represents the ability to map a consumer PortPrototype to an Ethernet-Mac-layer IEEE1722 ACF based communication for a whole bus (Can or Lin). **Tags:** atp.Status=candidate atp.recommendedPackage=RawDataStreamingMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *EthernetMacRawDataStreamMapping*, *IEEE1722RawDataStreamMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
acfBus	IEEE1722TpAcfBus	*	ref	Reference to the ACF buses which shall be handled by this mapping. **Tags:** atp.Splitkey=acfBus atp.Status=candidate

Table 14.16: IEEE1722AcfBusRawDataStreamConsumerMapping

[constr_3777] Existence of IEEE1722AcfBusRawDataStreamConsumerMapping.acfBus

Status: DRAFT
Imposition time: IT_Mani

⌈At least one reference IEEE1722AcfBusRawDataStreamConsumerMapping.acfBus shall exist.⌋

Class	**IEEE1722AcfBusPartRawDataStreamConsumerMapping**			
Note	This meta-class represents the ability to map a consumer PortPrototype to an Ethernet-Mac-layer IEEE1722 ACF based communication for a whole bus (Can or Lin). **Tags:** atp.Status=candidate atp.recommendedPackage=RawDataStreamingMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *EthernetMacRawDataStreamMapping*, *IEEE1722RawDataStreamMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
acfBusPart	IEEE1722TpAcfBusPart	*	ref	Reference to the ACF bus parts (Can or Lin frames) which shall be handled by this mapping. **Tags:** atp.Splitkey=acfBusPart atp.Status=candidate

Table 14.17: IEEE1722AcfBusPartRawDataStreamConsumerMapping

[constr_3778] Existence of IEEE1722AcfBusPartRawDataStreamConsumerMapping.acfBusPart

Status: DRAFT
Imposition time: IT_Mani

⌈At least one reference IEEE1722AcfBusPartRawDataStreamConsumerMapping.acfBusPart shall exist.⌋

15 Cross-Functional Cluster interaction

[TPS_MANI_03268] Semantics of meta-class `FunctionalClusterInteracts-WithFunctionalClusterMapping` ⌈Abstract meta-class `FunctionalCluster-InteractsWithFunctionalClusterMapping` provides an anchor for the specification of interaction between two functional clusters.

The identification of the interaction use case towards the functional cluster implementation is done by using an `InstanceSpecifier` of the concrete subclass of `FunctionalClusterInteractsWithFunctionalClusterMapping` that is used to define the concrete interaction.⌋

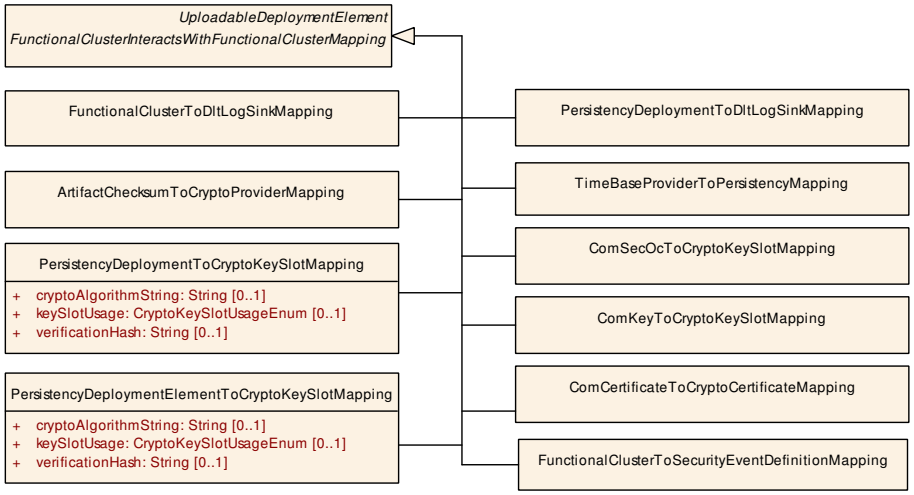


Figure 15.1: FunctionalClusterInteractsWithFunctionalClusterMapping Overview

Class	***FunctionalClusterInteractsWithFunctionalClusterMapping*** (abstract)			
Note	This meta-class identifies a relation between functional clusters on the adaptive platform such one functional cluster can call APIs of the other functional cluster. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	ArtifactChecksumToCryptoProviderMapping, ComCertificateToCryptoCertificateMapping, ComKeyToCryptoKeySlotMapping, ComSecOcToCryptoKeySlotMapping, FunctionalClusterInteractsWithDiagnosticEventMapping, FunctionalClusterInteractsWithPersistencyDeploymentMapping, FunctionalClusterToDltLogSinkMapping, FunctionalClusterToSecurityEventDefinitionMapping, IdsmModuleInstantiationToCryptoKeySlotMapping, NmInteractsWithSmMapping, PersistencyDeploymentElementToCryptoKeySlotMapping, PersistencyDeploymentToCryptoKeySlotMapping, PersistencyDeploymentToDltLogSinkMapping, SmInteractsWithNmMapping, TimeBaseProviderToPersistencyMapping, UcmToTimeBaseResourceMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 15.1: FunctionalClusterInteractsWithFunctionalClusterMapping

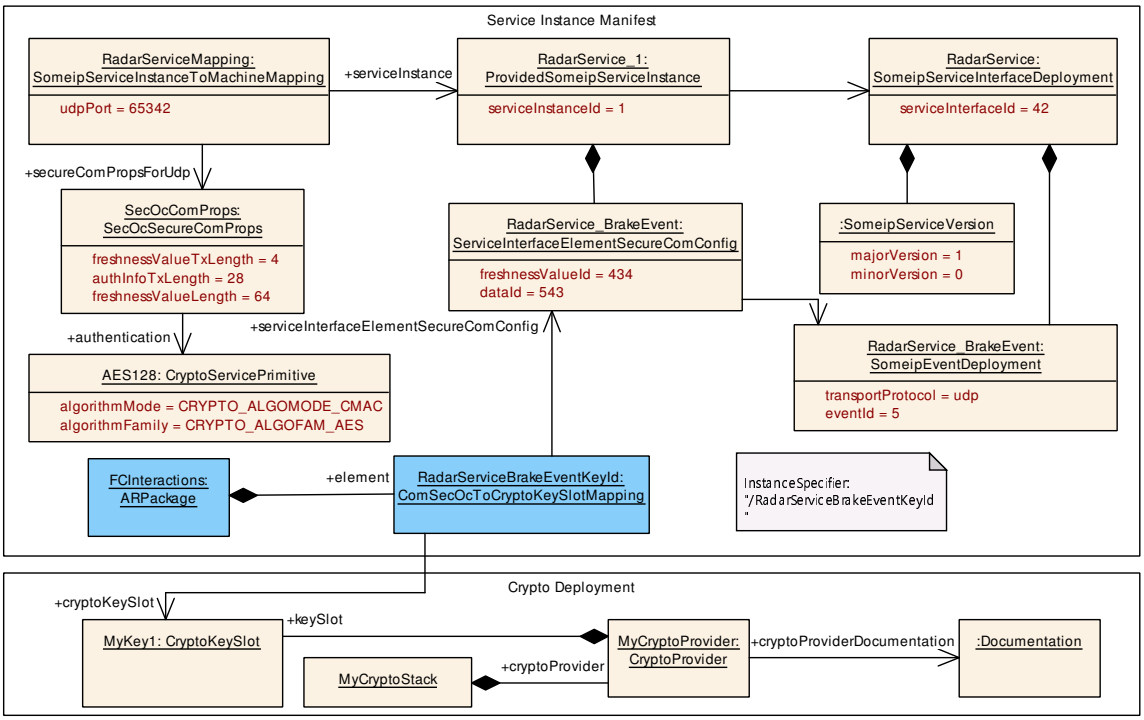


Figure 15.2: Example for the usage of `FunctionalClusterInteractsWithFunctionalClusterMapping`

In the case of an application, the model-path to an `PortPrototype` that is referencing a specific `PortInterface` defines the `InstanceSpecifier` that is used as identification towards the functional cluster implementation.

In the case of a FunctionalCluster interaction, the `PortPrototype` is not available. Therefore the path to a mapping element that is derived from `FunctionalClusterInteractsWithFunctionalClusterMapping` is used to define the `InstanceSpecifier` that is used in the API call.

The following figure shows the supported `FunctionalClusterInteractsWithFunctionalClusterMapping` subclasses that are available in the model.

The Figure 15.2 shows an example for the usage of `FunctionalClusterInteractsWithFunctionalClusterMapping`. In this example, the Service Instance Manifest describes a configuration for a `ProvidedSomeipServiceInstance` that contains an `Event` that is protected by SecOC.

Inside the Service Instance Manifest the “RadarServiceBrakeEventKeyId” represents the `FunctionalClusterInteractsWithFunctionalClusterMapping` that is located in the `ARPackage` “FCInteractions”.

These two elements in the manifest define the `InstanceSpecifier` “/RadarServiceBrakeEventKeyId” (see Figure 15.2) that is used in the Crypto API call from the Communication Management. This `InstanceSpecifier` is resolved to a concrete `CryptoKeySlot` with the information that is available in the Crypto target-configuration.

15.1 Com Certificate To Crypto Certificate Mapping

[TPS_MANI_03269] Semantics of ComCertificateToCryptoCertificate-Mapping ⌈The meta-class ComCertificateToCryptoCertificateMapping provides an anchor for the specification of interaction between eligible AdaptiveModuleInstantiations (see [TPS_MANI_01458]) and the CryptoModuleInstantiation and is used to map a CryptoServiceCertificate to a CryptoCertificate defined in the Crypto Stack.⌋

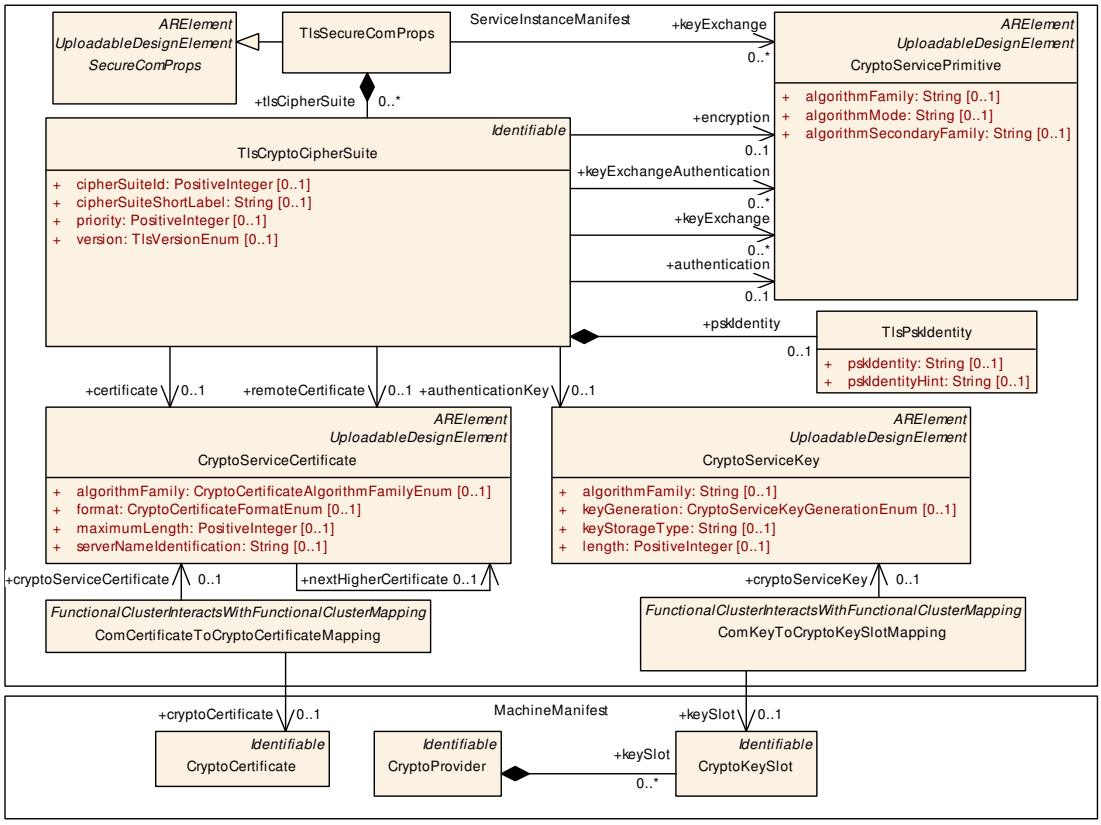


Figure 15.3: Mapping of TLS config elements to crypto objects in the crypto stack

Figure 15.3 shows the ComCertificateToCryptoCertificateMapping used to assign a CryptoServiceCertificate defined in the TLS configuration to a CryptoCertificate in the Crypto Stack.

[TPS_MANI_01458] Usage of the primitive attribute ComCertificateToCrypto-CertificateMapping.category ⌈The problem solved by ComCertificate-ToCryptoCertificateMapping is quite general and is **not** restricted to the CmModuleInstantiation.

The possibilities for reuse of ComCertificateToCryptoCertificateMapping for other AdaptiveModuleInstantiations are indicated by the value of the optional attribute category.

Standardized values of attribute category are:

- CM
- DOIP
- LOG_AND_TRACE
- SOVD

⌋

Class	**ComCertificateToCryptoCertificateMapping**			
Note	This meta-class maps the CryptoServiceCertificate defined in AdaptiveModuleInstantiation (see [TPS_MANI_01458]) to the Crypto Certificate defined in the Crypto Stack. **Tags:** atp.recommendedPackage=FCInteractions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
crypto Certificate	CryptoCertificate	0..1	ref	Reference to the CryptoCertifcate in the Crypto Stack. **Tags:** atp.Splitkey=cryptoCertificate
cryptoService Certificate	CryptoService Certificate	0..1	ref	Reference to the CryptoServiceCertificate in eligible AdaptiveModuleInstantiation (see [TPS_MANI_01458]). **Tags:** atp.Splitkey=cryptoServiceCertificate

Table 15.2: ComCertificateToCryptoCertificateMapping

15.2 Com Key To Crypto Key Slot Mapping

[TPS_MANI_03270] Semantics of `ComKeyToCryptoKeySlotMapping` ⌈The meta-class `ComKeyToCryptoKeySlotMapping` provides an anchor for the specification of interaction between eligible `AdaptiveModuleInstantiation`s (see [TPS_MANI_01459]) and the `CryptoModuleInstantiation` and is used to map a `CryptoServiceKey` to a `CryptoKeySlot` defined in the Crypto Stack.⌋

[TPS_MANI_01459] Usage of attribute `ComKeyToCryptoKeySlotMapping.category` ⌈The problem solved by `ComKeyToCryptoKeySlotMapping` is quite general and is **not** restricted to the `CmModuleInstantiation`.

The possibilities for reuse of `ComKeyToCryptoKeySlotMapping` for other `AdaptiveModuleInstantiation`s are indicated by the value of the optional attribute `category`.

Standardized values of attribute `category` are:

- CM
- DOIP
- LOG_AND_TRACE

- SOVD

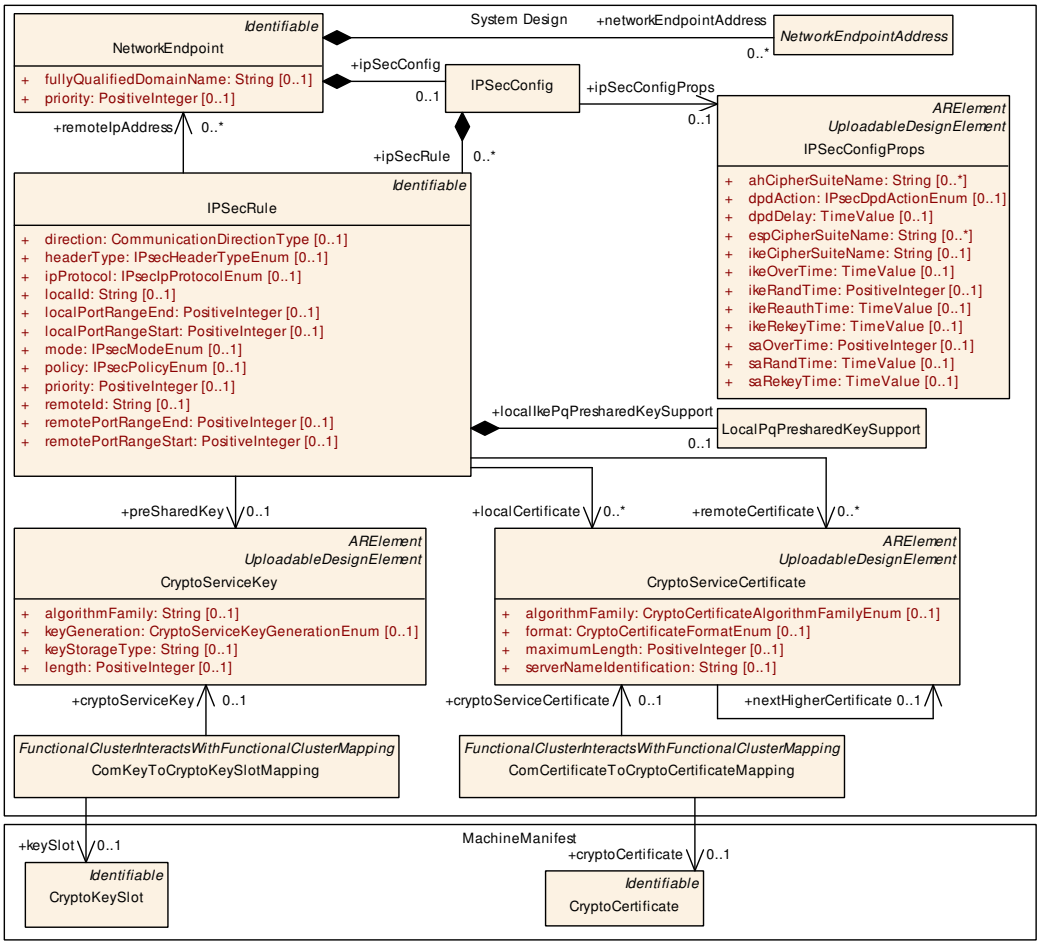


Figure 15.4: Mapping of IPsec config elements to crypto objects in the crypto stack

Class	**ComKeyToCryptoKeySlotMapping**			
Note	This meta-class maps the CryptoServiceKey defined in eligible AdaptiveModuleInstantiation (see [TPS_MANI_01459]) to the CryptoKeySlot defined in the Crypto Stack. **Tags:** atp.recommendedPackage=FCInteractions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cryptoService Key	CryptoServiceKey	0..1	ref	Reference to the CryptoServiceKey in eligible Adaptive ModuleInstantiation (see [TPS_MANI_01459]) **Tags:** atp.Splitkey=cryptoServiceKey
keySlot	CryptoKeySlot	0..1	ref	Reference to the CryptoKeySlot in the Crypto Stack. **Tags:** atp.Splitkey=keySlot

Table 15.3: ComKeyToCryptoKeySlotMapping

Figure 15.4 shows the ComKeyToCryptoKeySlotMapping used to assign a CryptoServiceKey defined in the IPsec configuration to CryptoKeySlot in the Crypto Stack.

15.3 Com SecOc To Crypto Key Slot Mapping

[TPS_MANI_03271] Semantics of ComSecOcToCryptoKeySlotMapping ⌈The meta-class ComSecOcToCryptoKeySlotMapping provides an anchor for the specification of interaction between the COM FunctionalCluster and the Crypto FunctionalCluster and is used to map one or several ServiceInterfaceElementSecureComConfig elements defined in the COM target configuration to a CryptoKeySlot defined in the Crypto Stack.⌋

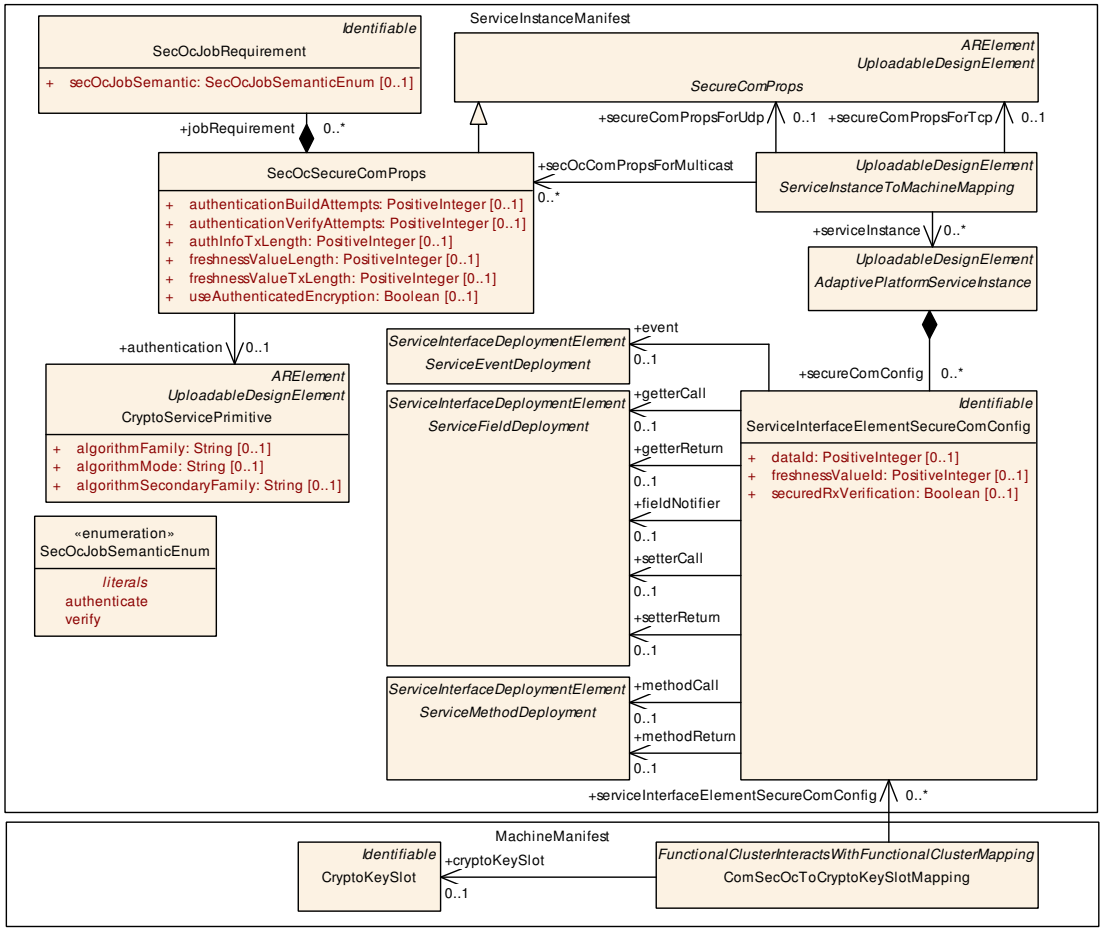


Figure 15.5: Mapping of SecOC config elements to crypto objects in the crypto stack

Figure 15.5 shows the ComSecOcToCryptoKeySlotMapping used to assign a ServiceInterfaceElementSecureComConfig defined in the SecOC configuration to CryptoKeySlot in the Crypto Stack.

Class	ComSecOcToCryptoKeySlotMapping			
Note	This meta-class maps one or several ServiceInterfaceElementSecureComConfig elements defined in the COM deployment to a CryptoKeySlot defined in the Crypto Stack. **Tags:** atp.recommendedPackage=FCInteractions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
cryptoKeySlot	CryptoKeySlot	0..1	ref	Reference to the CryptoKeySlot in the Crypto Stack. **Tags:** atp.Splitkey=cryptoKeySlot
serviceInterfaceElementSecureComConfig	ServiceInterfaceElementSecureComConfig	*	ref	Reference to one or several elements of a Service Interface defined in the deployment in the context of an AdaptivePlatformServiceInstance. **Tags:** atp.Splitkey=serviceInterfaceElementSecureComConfig

Table 15.4: ComSecOcToCryptoKeySlotMapping

15.4 Persistency Deployment To Crypto Key Slot Mapping

[TPS_MANI_03272] Semantics of PersistencyDeploymentToCryptoKeySlotMapping ⌈The meta-class PersistencyDeploymentToCryptoKeySlotMapping provides an anchor for the specification of interaction between the Persistency FunctionalCluster and the Crypto FunctionalCluster and is used to map a PersistencyDeployment defined in Persistency to a CryptoKeySlot defined in the Crypto Stack.⌋

Class	PersistencyDeploymentToCryptoKeySlotMapping			
Note	This meta-class represents the ability to define a mapping between the PersistencyDeployment and a CryptoKeySlot. **Tags:** atp.recommendedPackage=FCInteractions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
cryptoAlgorithm	String	0..1	attr	This attribute defines the cryptographic algorithm used for hashing, encryption, decryption, signature/MAC verification, or MAC generation.
cryptoKeySlot	CryptoKeySlot	0..1	ref	This reference represents the mapped CryptoKeySlot. **Tags:** atp.Splitkey=cryptoKeySlot
keySlotUsage	CryptoKeySlotUsageEnum	0..1	attr	This attribute defines the role of the keySlot assignment.

▽

△

Class	**PersistencyDeploymentToCryptoKeySlotMapping**			
persistency Deployment	PersistencyDeployment	0..1	ref	This reference represents the mapped Persistency Deployment. **Tags:** atp.Splitkey=persistencyDeployment
verificationHash	String	0..1	attr	This attribute defines the hash of the storage used in case of verification.

Table 15.5: PersistencyDeploymentToCryptoKeySlotMapping

[constr_10232] Multiplicity of reference in the role `PersistencyDeployment-ToCryptoKeySlotMapping.persistencyDeployment`

Imposition time: `IT_Mani`

⌈For each `PersistencyDeploymentToCryptoKeySlotMapping`, the reference in the role `persistencyDeployment` shall exist.⌋

Enumeration	**CryptoKeySlotUsageEnum**
Note	This enum defines the possible roles of the keySlotUsage. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	PersistencyDeploymentElementToCryptoKeySlotMapping.keySlotUsage, PersistencyDeploymentToCryptoKeySlotMapping.keySlotUsage
Literal	***Description***
encryption	Key slot usage for encryption **Tags:** atp.EnumerationLiteralIndex=1
verification	Key slot usage for verification **Tags:** atp.EnumerationLiteralIndex=0

Table 15.6: CryptoKeySlotUsageEnum

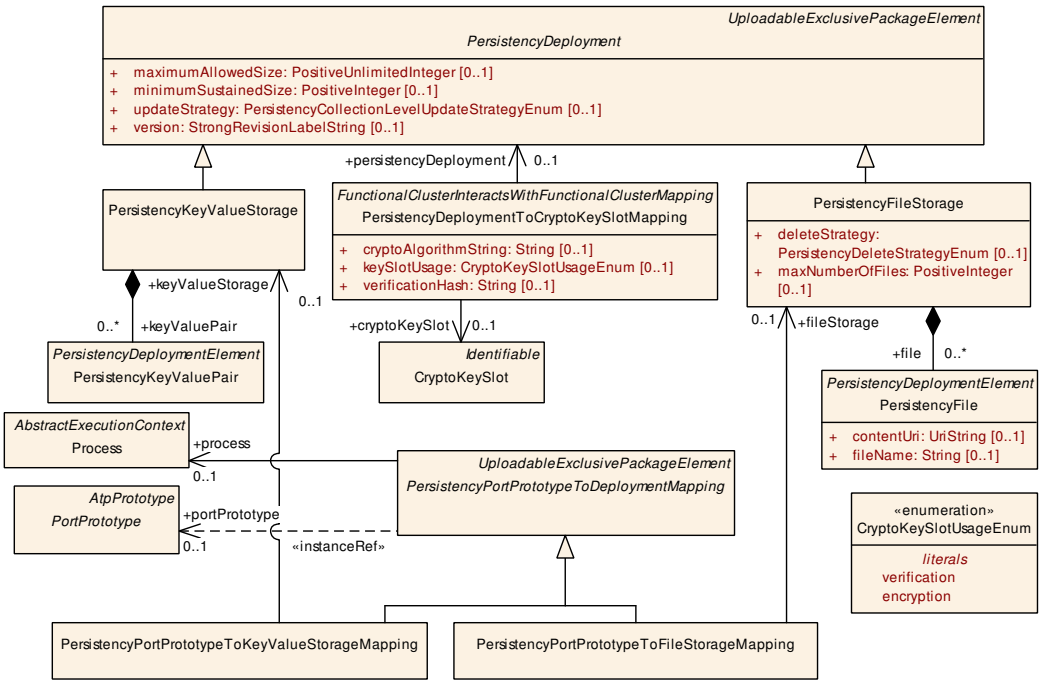


Figure 15.6: Mapping of PersistencyDeployment to crypto objects in the crypto stack

15.5 Persistency Deployment Element To Crypto Key Slot Mapping

[TPS_MANI_03273] Semantics of `PersistencyDeploymentElementToCryptoKeySlotMapping` ⌈The meta-class `PersistencyDeploymentElementToCryptoKeySlotMapping` provides an anchor for the specification of interaction between the Persistency FunctionalCluster and the Crypto FunctionalCluster and is used to map a `PersistencyDeploymentElement` defined in Persistency to a `CryptoKeySlot` defined in the Crypto Stack.⌋

Class	**PersistencyDeploymentElementToCryptoKeySlotMapping**			
Note	This meta-class represents the ability to define a mapping between the PersistencyDeploymentElement and a CryptoKeySlot. **Tags:** atp.recommendedPackage=FCInteractions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cryptoAlgorithm String	String	0..1	attr	This attribute defines the cryptographic algorithm used for hashing, encryption, decryption, signature/MAC verification, or MAC generation.
cryptoKeySlot	CryptoKeySlot	0..1	ref	This reference represents the mapped CryptoKeySlot. **Tags:** atp.Splitkey=cryptoKeySlot
keySlotUsage	CryptoKeySlotUsage Enum	0..1	attr	This attribute defines the role of the keySlot assignment.
persistency Deployment Element	PersistencyDeployment Element	0..1	ref	This reference represents the mapped Persistency Deployment. **Tags:** atp.Splitkey=persistencyDeploymentElement
verificationHash	String	0..1	attr	This attribute defines the hash of the storage used in case of verification.

Table 15.7: PersistencyDeploymentElementToCryptoKeySlotMapping

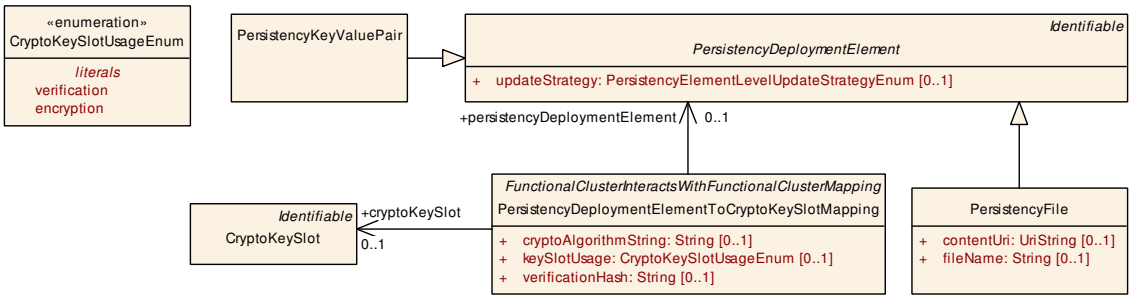


Figure 15.7: Mapping of PersistencyDeploymentElement to crypto objects in the crypto stack

15.6 Persistency Deployment To Dlt Log Sink Mapping

[TPS_MANI_03276] Semantics of meta-class `PersistencyDeploymentToDltLogSinkMapping` ⌈The meta-class `PersistencyDeploymentToDltLogSinkMapping` provides an anchor for the specification of interaction between the Persistency FunctionalCluster and Log & Trace and is used to map a `PersistencyDeployment` defined in Persistency to a `DltLogSink` defined in the `LogAndTraceInstantiation`.⌋

Class	**PersistencyDeploymentToDltLogSinkMapping**			
Note	This meta-class represents the ability to define a mapping between the PersistencyDeployment and a Dlt LogSink. **Tags:** atp.recommendedPackage=FCInteractions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dltContext	DltContext	0..1	ref	Reference to the DltContext to which this mapping belongs. **Tags:** atp.Splitkey=dltContext
logSink	DltLogSink	0..1	ref	Reference to the DltLogSink to which the Persistency Deployment is mapped **Tags:** atp.Splitkey=logSink
persistency Deployment	PersistencyDeployment	0..1	ref	This reference represents the mapped Persistency Deployment. **Tags:** atp.Splitkey=persistencyDeployment

Table 15.8: PersistencyDeploymentToDltLogSinkMapping

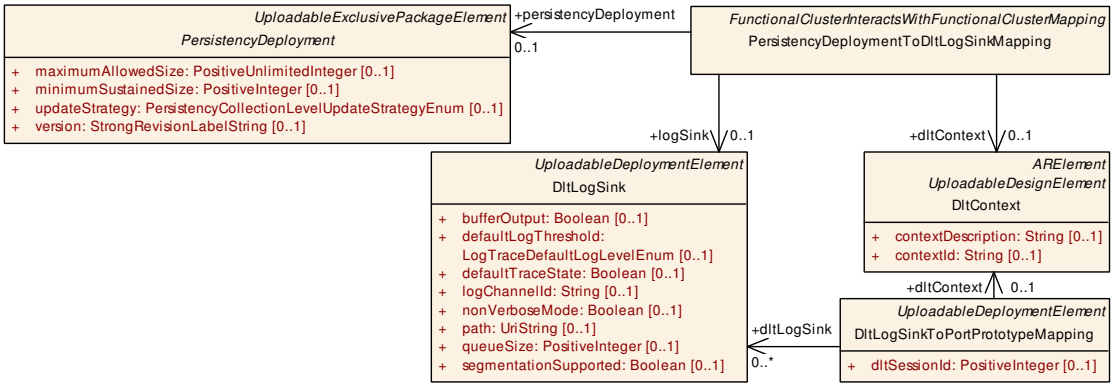


Figure 15.8: Mapping of PersistencyDeployment to DltLogSink

15.7 Functional Cluster To Dlt Log Sink Mapping

[TPS_MANI_01476] Semantics of meta-class `FunctionalClusterToDltLogSinkMapping` ⌈The meta-class `FunctionalClusterToDltLogSinkMapping` provides an anchor for the specification of interaction between a given functional cluster and Log & Trace and is used to map a functional cluster to a `DltLogSink` defined in the `LogAndTraceInstantiation`.⌋

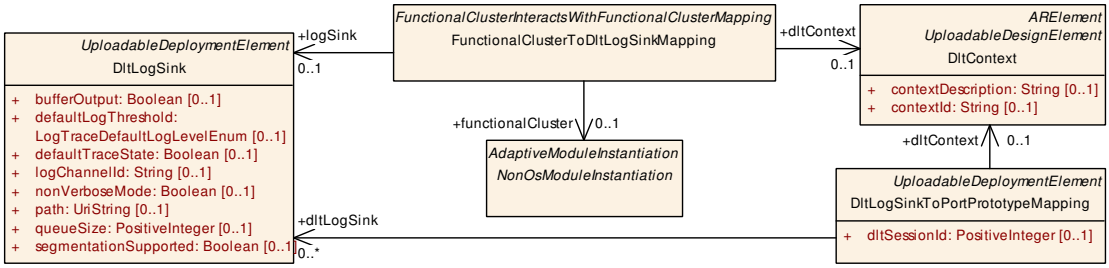


Figure 15.9: Mapping of functional cluster to DltLogSink

Class	**FunctionalClusterToDltLogSinkMapping**			
Note	This meta-class represents the ability to define a mapping between a functional cluster, represented by a NonOsModuleInstantiation, and a DltLogSink. **Tags:** atp.recommendedPackage=FCInteractions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dltContext	DltContext	0..1	ref	Reference to the DltContext to which this mapping belongs. **Tags:** atp.Splitkey=dltContext
functional Cluster	NonOsModule Instantiation	0..1	ref	This reefrence indentifies the functional cluster that wants to utilize logging. **Tags:** atp.Splitkey=functionalCluster
logSink	DltLogSink	0..1	ref	Reference to the DltLogSink to which the functional cluster is mapped. **Tags:** atp.Splitkey=logSink

Table 15.9: FunctionalClusterToDltLogSinkMapping

15.8 Ucm To Time Base Resource Mapping

[TPS_MANI_01374] Semantics of `UcmToTimeBaseResourceMapping` ⌈The meta-class `UcmToTimeBaseResourceMapping` provides an anchor for the specification of interaction between the Ucm Functional Cluster and Time Sync Functional Cluster and is used to identify the source of timestamp-information in the UCM.⌋

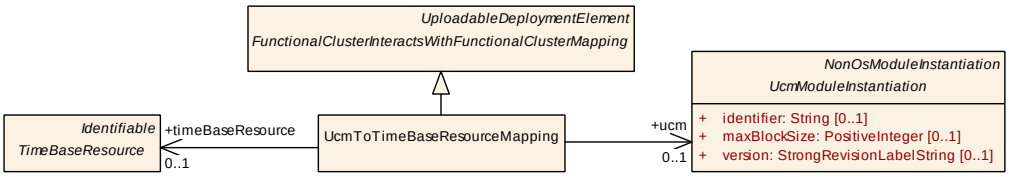


Figure 15.10: Mapping of the Ucm Functional Cluster to the Time Sync Functional Cluster

Class	**UcmToTimeBaseResourceMapping**			
Note	This meta-class maps the UCM Module Instantiation to the TimeSync Module Instantiation. **Tags:** atp.recommendedPackage=FCInteractions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
timeBase Resource	TimeBaseResource	0..1	ref	This reference identifies the relevant TimeBaseResource. **Tags:** atp.Splitkey=timeBaseResource
ucm	UcmModuleInstantiation	0..1	ref	This reference identifies the relevant UcmModule Instantiation. **Tags:** atp.Splitkey=ucm

Table 15.10: UcmToTimeBaseResourceMapping

15.9 Functional Cluster Interacts With Persistency Deployment Mapping

[TPS_MANI_01379] Semantics of meta-class `FunctionalClusterInteracts-WithPersistencyDeploymentMapping` ⌈Meta-class `FunctionalCluster-InteractsWithPersistencyDeploymentMapping` provides the ability to specify that (and how) a functional cluster that is implemented as a **library component** wants to access persistently stored data (formalized by meta-class `Persistency-Deployment`) on the target platform.⌋

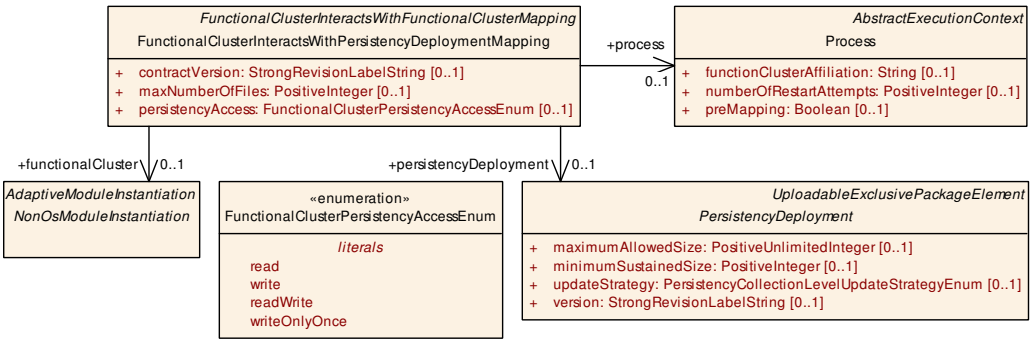


Figure 15.11: Mapping of any non-OS Functional Cluster to the Persistency Functional Cluster

Functional clusters are typically not dealing with modeled data types, and therefore a formalized relation of FunctionalClusterInteractsWithPersistency-DeploymentMapping to a subclass of CppImplementationDataType is not foreseen.

Class	**FunctionalClusterInteractsWithPersistencyDeploymentMapping**			
Note	This meta-class represents the ability to define a mapping between any functional cluster modeled as a subclass of NonOsModuleInstantiation and a PersistencyDeployment. **Tags:** atp.recommendedPackage=FCInteractions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contractVersion	StrongRevisionLabel String	0..1	attr	This attribute represents the contract version that is used to determine whether the Persistency configuration experienced structural changes and is also used for the check for data type compatibility.
functional Cluster	NonOsModule Instantiation	0..1	ref	This reference identifies the client functional cluster that wants to use persistency. **Tags:** atp.Splitkey=functionalCluster
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 FunctionalClusterInteractsWithPersistency DeploymentMapping.
persistency Access	FunctionalCluster PersistencyAccess Enum	0..1	attr	This attribute represents the definition of the persistency access of all kinds of persisted data at run-time in the context of the enclosing FunctionalClusterInteractsWith PersistencyDeploymentMapping.
persistency Deployment	PersistencyDeployment	0..1	ref	This reference identifies the applicable Persistency Deployment. **Tags:** atp.Splitkey=persistencyDeployment
process	Process	0..1	ref	"This reference identifies the applicable process. **Tags:** atp.Splitkey=process

Table 15.11: FunctionalClusterInteractsWithPersistencyDeploymentMapping

[constr_10484] Existence of attribute FunctionalClusterInteractsWith-PersistencyDeploymentMapping.contractVersion

Imposition time: IT_Mani

⌈In the context of meta-class FunctionalClusterInteractsWithPersistency-DeploymentMapping, the attribute in the role contractVersion shall exist.⌋

The number of values of FunctionalClusterPersistencyAccessEnum differs intentionally from the number of values of the related PersistencyAccessEnum, because of the existence of different use cases.

Enumeration	**FunctionalClusterPersistencyAccessEnum**
Note	This meta-class provides possible values about how functional clusters may use persistency with respect to the direction of access. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	FunctionalClusterInteractsWithPersistencyDeploymentMapping.persistencyAccess
Literal	***Description***
read	Functional Cluster wants to access persistency with a read semantics. **Tags:** atp.EnumerationLiteralIndex=0
readWrite	Functional Cluster wants to access persistency with a read and write semantics. **Tags:** atp.EnumerationLiteralIndex=2
write	Functional Cluster wants to access persistency with a write semantics. **Tags:** atp.EnumerationLiteralIndex=1
writeOnlyOnce	The elements of this storage are prevented from being changed after first initialization. **Tags:** atp.EnumerationLiteralIndex=3

Table 15.12: FunctionalClusterPersistencyAccessEnum

15.10 Adaptive Module Instantiation to Security Event Definition Mapping

[TPS_MANI_03324] Semantics of FunctionalClusterToSecurityEvent-DefinitionMapping

Status: DRAFT

⌈The meta-class FunctionalClusterToSecurityEventDefinitionMapping provides an anchor for the specification of interaction between a Functional Cluster that is modeled as a AdaptiveModuleInstantiation and the IDSM (Intrusion Detection System Manager).

FunctionalClusterToSecurityEventDefinitionMapping is used to map a security event (formalized by meta-class SecurityEventDefinition) and the Functional Cluster that reports this security event.⌋

[constr_5381] Modeling of Security Event reports by FunctionalCluster shall not be done via ProcessToMachineMapping

Imposition time: IT_Mani

⌈If a Process is mapped by the ProcessToMachineMapping and this ProcessToMachineMapping references a NonOsModuleInstantiation in the role nonOsModuleInstantiation, then this Process shall not be referenced by a SecurityEventMapping in the role process.⌋

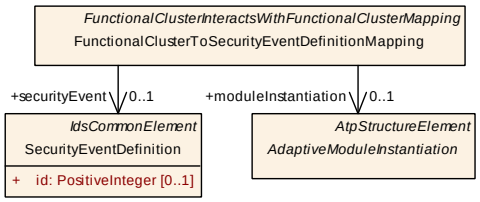


Figure 15.12: Mapping of the interaction between a Functional Cluster and the IdsM

Class	**FunctionalClusterToSecurityEventDefinitionMapping**			
Note	This meta-class represents the ability to define a mapping between the SecurityEventDefinition and the Module Instantiation that reports the security event. **Tags:** atp.recommendedPackage=FunctionalClusterToSecurityEventDefinitionMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
module Instantiation	AdaptiveModule Instantiation	0..1	ref	Reference to the AdaptiveModuleInstantiation that reports the SecurityEvent. **Tags:** atp.Splitkey=moduleInstantiation
securityEvent	SecurityEventDefinition	0..1	ref	Reference to the SecurityEvent that is reported by the AdaptiveModuleInstantiation. **Tags:** atp.Splitkey=securityEvent

Table 15.13: FunctionalClusterToSecurityEventDefinitionMapping

15.11 State Management interacts with Network Management and vice versa

The interaction between state management and network management is bi-directional, but the purpose of each direction is different enough that a dedicated modeling of the configuration of each direction makes sense.

15.11.1 State Management interacts with Network Management

State management interacts with network management to request the network management to switch to a specific communication state (see also Chapter 12.14.5.4).

In this interaction, state management acts like a functional cluster rather than an application and therefore the modeling of a subclass of meta-class `FunctionalClusterInteractsWithFunctionalClusterMapping` needs to be created: `SmInteractsWithNmMapping`.

[TPS_MANI_01446] Semantics of meta-class SmInteractsWithNmMapping

Status: DRAFT

⌈Meta-class SmInteractsWithNmMapping supports the interaction between state management and network management by mapping a StateManagementNm-ActionItem (which represents the state management) to an NmNetworkHandle (representing the network management).⌋

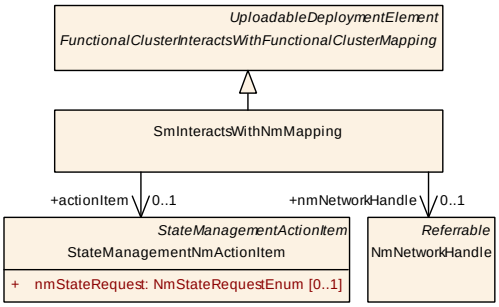


Figure 15.13: Mapping of the interaction between state management and network management

Class	**SmInteractsWithNmMapping**			
Note	This mapping represents an interaction from state management to network management. **Tags:** atp.Status=draft atp.recommendedPackage=FCInteractions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
actionItem	StateManagementNm ActionItem	0..1	ref	This reference identifies the action item with which the state management wants to interact with network management. **Tags:** atp.Splitkey=actionItem atp.Status=draft
nmNetwork Handle	NmNetworkHandle	0..1	ref	This reference identifies the network management handle that is affected by the interaction with the state management. **Tags:** atp.Splitkey=nmNetworkHandle atp.Status=draft

Table 15.14: SmInteractsWithNmMapping

[constr_10513] Existence of the reference in the role SmInteractsWithNm-Mapping.actionItem

Status: DRAFT
Imposition time: IT_Mani

⌈For each SmInteractsWithNmMapping, the reference in the role actionItem shall exist.⌋

[constr_10514] Existence of the reference in the role SmInteractsWithNmMapping.nmNetworkHandle

Status: DRAFT
Imposition time: IT_Mani

⌈For each SmInteractsWithNmMapping, the reference in the role nmNetworkHandle shall exist.⌋

15.11.2 Network Management interacts with State Management

Network management interacts with state management to notify the state management that a specific communication state (see also Chapter 12.14.5.4) has been switched to.

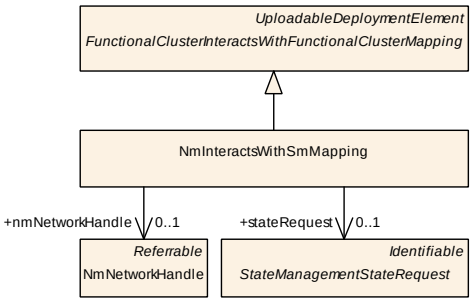


Figure 15.14: Mapping of the interaction between network management and state management

This may happen, for example, if the communication bus goes down unexpectedly and state management needs to react to this situation.

In this interaction, state management acts like a functional cluster rather than an application and therefore the modeling of a subclass of meta-class FunctionalClusterInteractsWithFunctionalClusterMapping needs to be created: NmInteractsWithSmMapping.

[TPS_MANI_01447] Semantics of meta-class NmInteractsWithSmMapping

Status: DRAFT

⌈Meta-class NmInteractsWithSmMapping supports the interaction between state management and network management by mapping an NmNetworkHandle (which represents the network management) to a StateManagementStateRequest (representing the state management).⌋

Class	**NmInteractsWithSmMapping**			
Note	This mapping represents an interaction from network management to state management. **Tags:** atp.Status=draft atp.recommendedPackage=FCInteractions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
nmNetwork Handle	NmNetworkHandle	0..1	ref	This reference identifies the network management handle that wants to interact with state management. **Tags:** atp.Splitkey=nmNetworkHandle atp.Status=draft
stateRequest	StateManagementState Request	0..1	ref	This reference identifies the state management state request that is involved in the interaction with the network management. **Tags:** atp.Splitkey=stateRequest atp.Status=draft

Table 15.15: NmInteractsWithSmMapping

[constr_10515] Existence of the reference in the role `NmInteractsWithSmMapping.stateRequest`

Status: DRAFT
Imposition time: `IT_Mani`

⌈For each `NmInteractsWithSmMapping`, the reference in the role `stateRequest` shall exist.⌋

[constr_10516] Existence of the reference in the role `NmInteractsWithSmMapping.nmNetworkHandle`

Status: DRAFT
Imposition time: `IT_Mani`

⌈For each `NmInteractsWithSmMapping`, the reference in the role `nmNetworkHandle` shall exist.⌋

15.12 Functional Cluster reports Production Error

[TPS_MANI_01448] Semantics of meta-class `FunctionalClusterInteractsWithDiagnosticEventMapping` ⌈The meta-class `FunctionalClusterInteractsWithDiagnosticEventMapping` provides an anchor for the specification of interaction between a functional cluster that is either represented as a `AdaptiveModuleInstantiation` or a `PersistencyDeployment` and the diagnostic management, specifically for the reporting of production errors from functional clusters.⌋

There is no sub-class of meta-class AdaptiveModuleInstantiation that represents the *persistency* functional cluster. This role is taken by PersistencyDeployment.

This is the reason why two references exist that are mutually exclusive (see [constr_10518]):

- persistencyDeployment: use for persistency.
- functionalCluster: use for everything else.

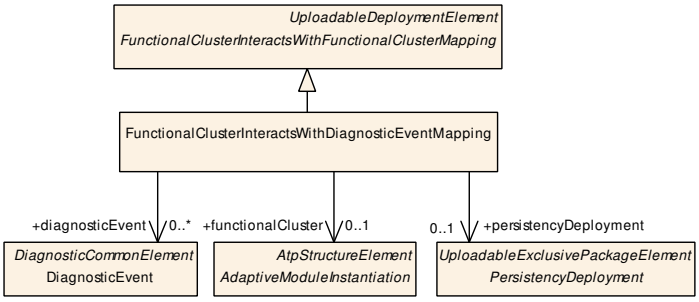


Figure 15.15: Functional Cluster reports production error

Class	**FunctionalClusterInteractsWithDiagnosticEventMapping**			
Note	This meta-class creates an association between a functional cluster and a diagnostic event that represents the diagnostic management by proxy. The purpose of this mapping is to support the reporting of production errors. **Tags:** atp.recommendedPackage=FCInteractions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnosticEvent	DiagnosticEvent	*	ref	This reference identifies the collection of diagnostic events that can be reported by the mapped functional cluster. **Tags:** atp.Splitkey=diagnosticEvent
functional Cluster	AdaptiveModule Instantiation	0..1	ref	This reference identifies the reporting functional cluster. **Tags:** atp.Splitkey=functionalCluster
persistency Deployment	PersistencyDeployment	0..1	ref	This reference identifies the reporting persistency deployment. **Tags:** atp.Splitkey=persistencyDeployment

Table 15.16: FunctionalClusterInteractsWithDiagnosticEventMapping

[constr_10518] Mutually exclusive existence of the roles FunctionalClusterInteractsWithDiagnosticEventMapping.persistencyDeployment vs. functionalCluster

Imposition time: IT_Mani

⌈For each FunctionalClusterInteractsWithDiagnosticEventMapping, **either** the reference in the role

- persistencyDeployment or
- functionalCluster

shall exist.⌋

[constr_10519] Existence of reference FunctionalClusterInteractsWithDiagnosticEventMapping.diagnosticEvent

Imposition time: IT_Mani

⌈For each FunctionalClusterInteractsWithDiagnosticEventMapping, the reference in the role diagnosticEvent shall exist.⌋

15.13 Idsm needs Crypto Key for Signature Generation

The Idsm (see [34, TPS Security Extract Template]) needs to cryptographically sign the reports about security events. For this purpose, it needs access to one cryptographic key, represented as CryptoKeySlot in the meta-model.

The relation between Idsm and the CryptoKeySlot is configured by means of the IdsmModuleInstantiationToCryptoKeySlotMapping.

[TPS_MANI_01523] Semantics of IdsmModuleInstantiationToCryptoKeySlotMapping ⌈The meta-class IdsmModuleInstantiationToCryptoKeySlotMapping provides an anchor for the Idsm (represented by meta-class IdsmModuleInstantiation, referenced in the role idsmModuleInstantiation) to access the content of a CryptoKeySlot (referenced in the role cryptoKeySlot), specifically for the signing of security event reporting.⌋

[TPS_MANI_01524] Semantics of the primitive attribute IdsmModuleInstantiationToCryptoKeySlotMapping.authenticatorFormat ⌈Attribute IdsmModuleInstantiationToCryptoKeySlotMapping.authenticatorFormat controls the format in which the signature used by the IdsM is stored. AUTOSAR standardizes the following values for this attribute:

RAW is used if the signature is supposed to be stored in raw format, see [SWS_CRYPT_60022] in [18, SWS Cryptography].

DER is used if the signature is supposed to be stored in the DER (distinguished encoding rules) format, see [SWS_CRYPT_04210] in [18, SWS Cryptography].

PEM is used if the signature is supposed to be stored in the PEM (privacy-enhanced electronic mail) format, see [SWS_CRYPT_04211] in [18, SWS Cryptography].

⌋

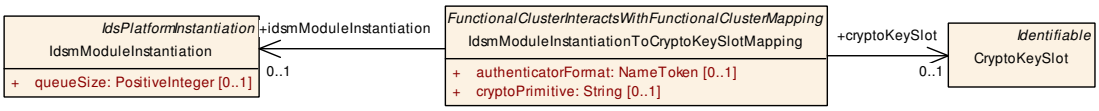


Figure 15.16: Idsm needs Crypto Key for Signature Generation

Class	**IdsmModuleInstantiationToCryptoKeySlotMapping**			
Note	This meta-class maps the cryptographic algorithm and key to be used to the the IdsM instantiation, for providing signature information in QSEv messages. **Tags:** atp.recommendedPackage=FCInteractions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authenticator Format	NameToken	0..1	attr	This attribute configures the authenticator format. The potential content of the attribute is limited by a constraint.
cryptoKeySlot	CryptoKeySlot	0..1	ref	This reference represents the mapped `CryptoKeySlot`. **Tags:** atp.Splitkey=cryptoKeySlot
cryptoPrimitive	String	0..1	attr	This attribute defines the cryptographic algorithm to be used for providing authentication information in QSEv messages. The content of this attribute shall comply to the "Cryptographic Primitives Naming Convention".
idsmModule Instantiation	IdsmModule Instantiation	0..1	ref	This reference identifies the applicable `IdsmModuleInstantiation`. **Tags:** atp.Splitkey=idsmModuleInstantiation atp.Status=candidate

Table 15.17: IdsmModuleInstantiationToCryptoKeySlotMapping

[constr_01523] Existence of the role `IdsmModuleInstantiationToCrypto-KeySlotMapping.idsmModuleInstantiation`

Imposition time: `IT_Mani`

⌈For each `IdsmModuleInstantiationToCryptoKeySlotMapping`, the reference in the role `idsmModuleInstantiation` shall exist.⌋

[constr_01524] Existence of the role `IdsmModuleInstantiationToCrypto-KeySlotMapping.cryptoKeySlot`

Imposition time: `IT_Mani`

⌈For each `IdsmModuleInstantiationToCryptoKeySlotMapping`, the reference in the role `cryptoKeySlot` shall exist.⌋

16 Software Distribution

16.1 Overview

One of the key features of the *AUTOSAR adaptive platform* is the ability to extend the software on a given ECU without having to re-flash the entire ECU. Instead, software packages are uploaded to the ECU where the content is taken care of by responsible platform modules.

This topic is relevant for the modeling because an uploadable software package consists not only of software itself but also of manifest content required to support the integration of the uploaded software with the existing platform instance.

As far as the meta-model is concerned, the discussion about manifests and which manifest content needs to go with which other model elements is independent of the file granularity. In other words, it would not make sense to formalize the uploadable software package on the basis of references to files that carry model elements.

Instead, the view on the manifest topic from the modeling point of view focuses on model elements that make up manifest content.

Therefore, the modeling of an uploadable software package allows for putting references to all the required model elements that, in their entirety, make up the manifest of the corresponding application software that is also going to end up in the uploadable software package.

From the formal point of view, such an uploadable software package is modeled as a so-called `SoftwareCluster`. This meta-class is the root element that in turn describes all the necessary content of an uploadable software package.

However, the software package is not created out of thin air. It is the result of a workflow that starts from the formulation of requirements on the content of a `SoftwareCluster`.

These requirements are formalized by means of meta-class `SoftwareClusterDesign`s. The relation between `SoftwareClusterDesign` and `SoftwareCluster` is depicted in Figure 16.1.

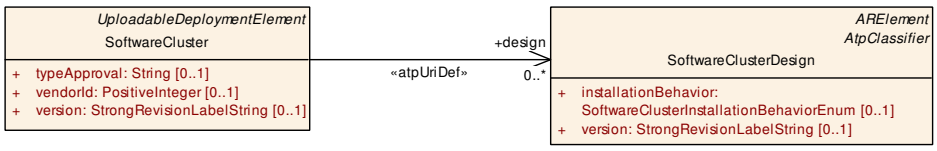


Figure 16.1: Relation of `SoftwareClusterDesign` to `SoftwareCluster`

16.2 Uploadable Package Element

[TPS_MANI_01109] Semantics of `UploadablePackageElement` ⌈To keep the complexity of the modeling of `SoftwareCluster` as low as possible, the abstract meta-class `UploadablePackageElement` has been created.

This allows for the referencing of model elements derived from `UploadablePackageElement` that need to be considered in an uploadable software package from within a `SoftwareCluster` with just the reference `containedPackageElement`.⌋

Meta-class `UploadablePackageElement` has two abstract sub-classes:

- one for model elements that are target-configuration-related (represented by meta-class `UploadableDeploymentElement`) and
- one for model elements that are design-related (represented by meta-class `UploadableDesignElement`), but still needed at configuration time on the target system as a dependency of the `UploadableDeploymentElement`s.

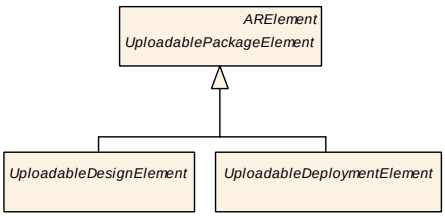


Figure 16.2: Overview of `UploadablePackageElement` and its direct sub-classes

Class	***UploadablePackageElement*** (abstract)			
Note	This meta-class acts as an abstract base class for all meta-classes that need to be added to an uploadable software package in order to complete the manifest content. This applies for both design-level and deployment-level meta-classes.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	ApmcDefinitionCollection, ApmcFunctionalClusterDef, ApmcFunctionalClusterValue, ApmcValueCollection, *UploadableDeploymentElement*, *UploadableDesignElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 16.1: UploadablePackageElement

[TPS_MANI_01413] Semantics of `UploadableDeploymentElement` ⌈As a refinement of the abstract base class `UploadablePackageElement`, meta-class `UploadableDeploymentElement` represents configuration information that is **related to the target-configuration phase** and which is expected to be **uploaded to the target** system and **needed for configuration** on the *AUTOSAR adaptive platform*.⌋

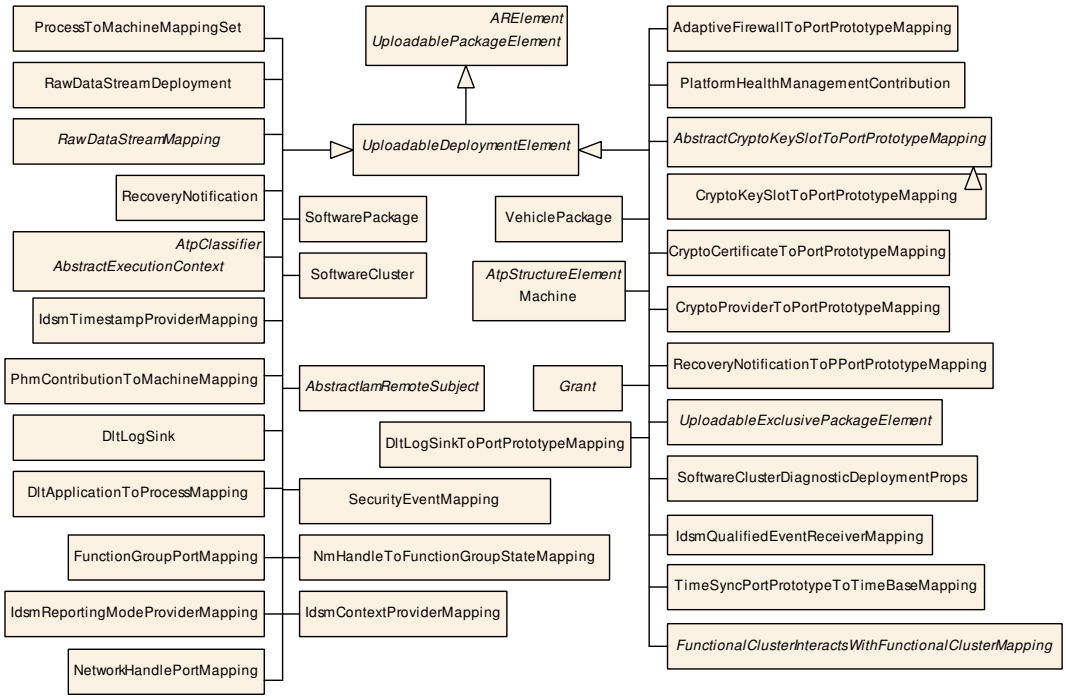


Figure 16.3: Overview of `UploadableDeploymentElement`

Class	***UploadableDeploymentElement*** (abstract)			
Note	This meta-class acts as an abstract base class for all deployment-level meta-classes that need to be added to an uploadable software package in order to complete the manifest content.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Subclasses	*AbstractCryptoKeySlotToPortPrototypeMapping*, *AbstractExecutionContext*, *AbstractIamRemoteSubject*, *AbstractProcessAttribute*, *AbstractSuspendToRamMapping*, AdaptiveFirewallToPortPrototypeMapping, CryptoCertificateGroupToPortPrototypeMapping, CryptoCertificateToPortPrototypeMapping, CryptoProviderToPortPrototypeMapping, DltApplicationToProcessMapping, DltLogSink, DltLogSinkToPortPrototypeMapping, FunctionGroupPortMapping, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Grant*, IdsmContextProviderMapping, IdsmQualifiedEventReceiverMapping, IdsmReportingModeProviderMapping, IdsmTimestampProviderMapping, Machine, NetworkHandlePortMapping, NmHandleToFunctionGroupStateMapping, PhmContributionToMachineMapping, PlatformHealthManagementContribution, ProcessToMachineMappingSet, RawDataStreamDeployment, *RawDataStreamMapping*, RecoveryNotification, RecoveryNotificationToPPortPrototypeMapping, SecurityEventMapping, SoftwareCluster, SoftwareClusterDiagnosticDeploymentProps, SoftwarePackage, StartupConfig, TimeSyncPortPrototypeToTimeBaseMapping, *UploadableExclusivePackageElement*, VehiclePackage			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 16.2: UploadableDeploymentElement

[TPS_MANI_01414] Semantics of `UploadableDesignElement` ⌈As a refinement of the abstract base class `UploadablePackageElement`, meta-class `UploadableDesignElement` represents configuration information that is **related to the design phase** and which is nevertheless expected to be **uploaded to the target** system and **needed for configuration** (as a dependency) on the *AUTOSAR adaptive platform*.⌋

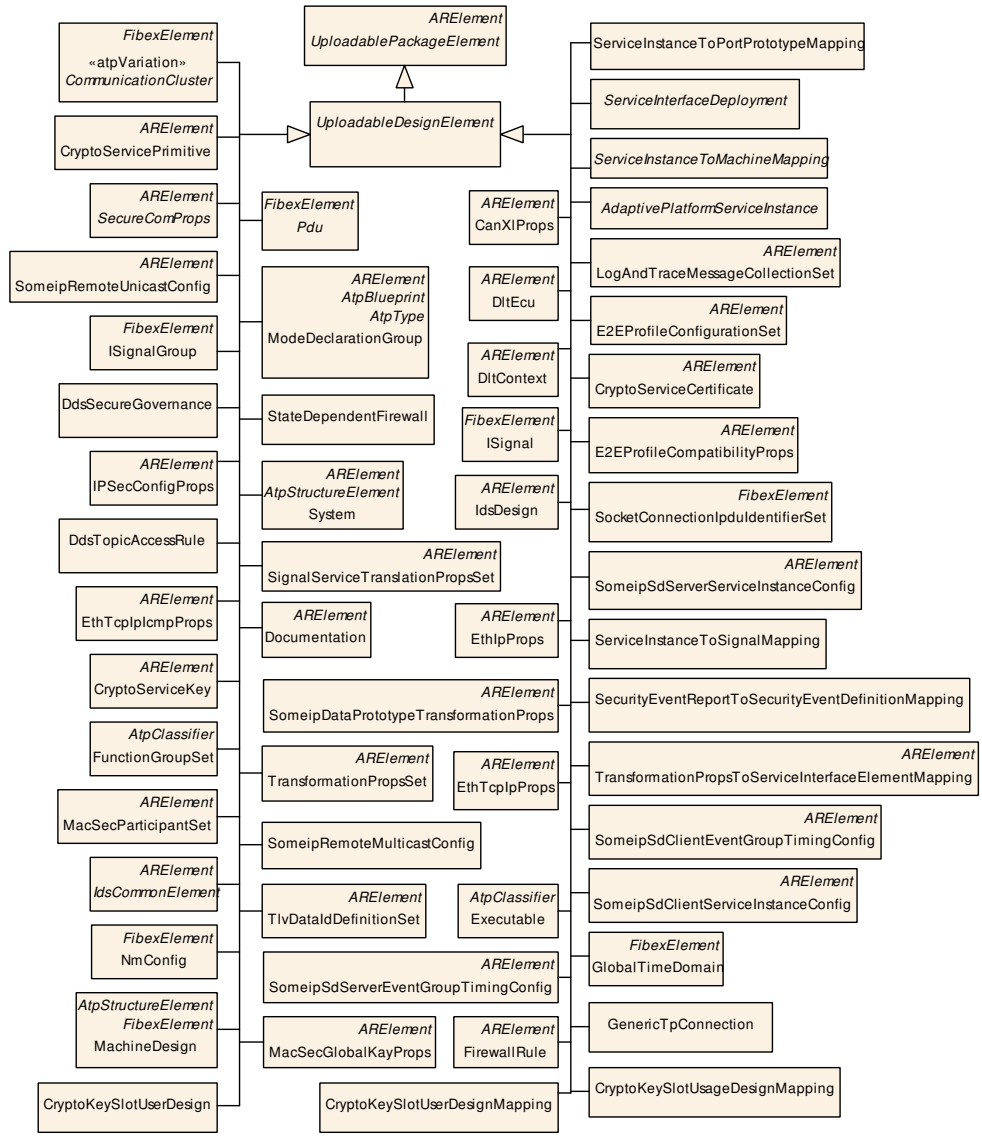


Figure 16.4: Overview of `UploadableDesignElement`

This approach to collecting elements is very similar in semantics to `System.fibexElement` or `DiagnosticContributionSet.element`.

Class	***UploadableDesignElement*** (abstract)
Note	This meta-class acts as an abstract base class for all design-level meta-classes that need to be added to an uploadable software package in order to complete the manifest content.
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*

△

Class	**UploadableDesignElement** (abstract)			
Subclasses	*AdaptivePlatformServiceInstance*, CanXlProps, *CommunicationCluster*, CryptoKeySlotUsageDesign Mapping, CryptoKeySlotUserDesign, CryptoKeySlotUserDesignMapping, CryptoServiceCertificate, CryptoServiceKey, CryptoServicePrimitive, DdsSecureGovernance, DdsTopicAccessRule, DltContext, DltEcu, Documentation, E2EProfileCompatibilityProps, E2EProfileConfigurationSet, EthIpProps, EthTcp IpIcmpProps, EthTcpIpProps, Executable, FirewallRule, FunctionGroupSet, GenericTpConnection, GlobalTimeDomain, IPSecConfigProps, ISignal, ISignalGroup, *IdsCommonElement*, IdsDesign, LogAnd TraceMessageCollectionSet, MacSecGlobalKayProps, MacSecParticipantSet, MachineDesign, Mode DeclarationGroup, NmConfig, *Pdu*, *SecureComProps*, SecurityEventReportToSecurityEventDefinition Mapping, *ServiceInstanceToMachineMapping*, ServiceInstanceToPortPrototypeMapping, Service InstanceToSignalMapping, *ServiceInterfaceDeployment*, SignalServiceTranslationPropsSet, Socket ConnectionIpduIdentifierSet, SomeipDataPrototypeTransformationProps, SomeipRemoteMulticast Config, SomeipRemoteUnicastConfig, SomeipSdClientEventGroupTimingConfig, SomeipSdClient ServiceInstanceConfig, SomeipSdServerEventGroupTimingConfig, SomeipSdServerServiceInstance Config, *SovdAbstractContributingEntity*, SovdSystemDesign, StateDependentFirewall, System, TlvData IdDefinitionSet, TransformationPropsSet, TransformationPropsToServiceInterfaceElementMapping			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 16.3: UploadableDesignElement

16.3 Software Cluster

16.3.1 Software Cluster General Modeling

[TPS_MANI_01110] **Semantics of `SoftwareCluster`** ⌈The existence of a `SoftwareCluster` represents an uploadable software package.⌋

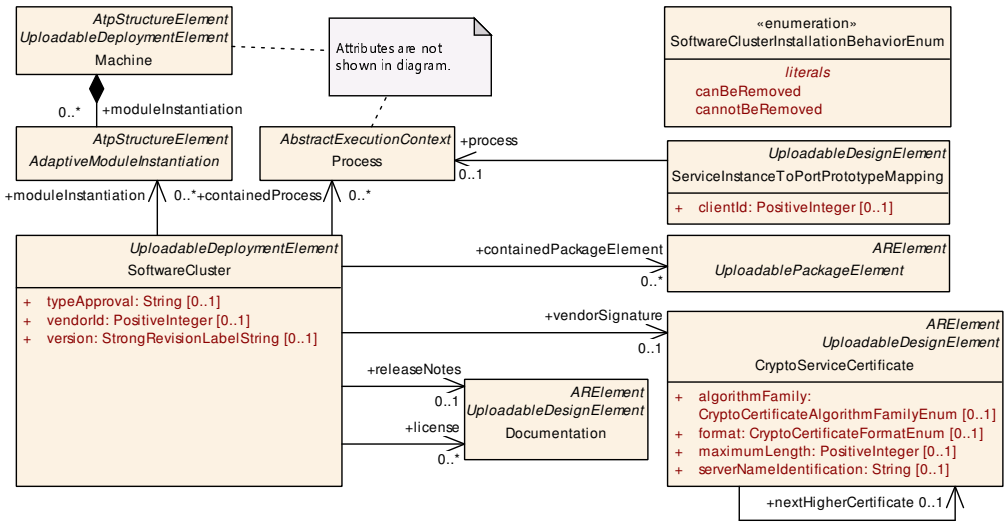


Figure 16.5: Modeling of `SoftwareCluster`

[TPS_MANI_01213] **Semantics of meta-class `StrongRevisionLabelString`** ⌈Meta-class `StrongRevisionLabelString` supports the specification of a version number for a `SoftwareCluster` that consists of four components ([constr_1747] applies):

- Major version
- Minor version
- Patch version
- Additional labels for pre-release version and build metadata

⌋

Class	**SoftwareCluster**			
Note	This meta-class represents the ability to define an uploadable software-package, i.e. the SoftwareCluster shall contain all software and configuration for a given purpose. **Tags:** atp.recommendedPackage=SoftwareClusters This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
artifact Checksum	ArtifactChecksum	*	aggr	This aggregation carries the checksums for artifacts contained in the enclosing SoftwareCluster. Please note that the value of these checksums is only applicable at the time of configuration. **Tags:** atp.Splitkey=artifactChecksum, artifact Checksum.shortName
artifactLocator	ArtifactLocator	*	aggr	This aggregation represents the artifact locations that are relevant in the context of the enclosing SoftwareCluster **Tags:** atp.Splitkey=artifactLocator, artifactLocator.short Name
claimed FunctionGroup	ModeDeclarationGroup Prototype	*	ref	Each SoftwareCluster can reserve the usage of a given functionGroup such that no other SoftwareCluster is allowed to use it **Tags:** atp.Splitkey=claimedFunctionGroup
contained ARElement	ARElement	*	ref	This reference represents the collection of model elements that cannot derive from UploadablePackage Element and that contribute to the completeness of the definition of the SoftwareCluster. **Tags:** atp.Splitkey=containedARElement
contained Package Element	UploadablePackage Element	*	ref	This reference identifies model elements that are required to complete the manifest content. **Tags:** atp.Splitkey=containedPackageElement
contained Process	Process	*	ref	This reference represent the processes contained in the enclosing SoftwareCluster. **Tags:** atp.Splitkey=containedProcess
design	SoftwareClusterDesign	*	ref	This reference represents the identification of all Software ClusterDesigns applicable for the enclosing Software Cluster. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=design
diagnostic Deployment Props	SoftwareCluster DiagnosticDeployment Props	0..1	ref	This reference identifies the applicable SoftwareCluster DiagnosticDeploymentProps that are applicable for the referencing SoftwareCluster. **Tags:** atp.Splitkey=diagnosticDeploymentProps

▽

△

Class	**SoftwareCluster**			
license	Documentation	*	ref	This attribute allows for the inclusion of the full text of a license of the enclosing SoftwareCluster. In many cases open source licenses require the inclusion of the full license text to any software that is released under the respective license. **Tags:** atp.Splitkey=license
module Instantiation	AdaptiveModule Instantiation	*	ref	This reference identifies AdaptiveModuleInstantiations that need to be included with the SoftwareCluster in order to establish infrastructure required for the installation of the SoftwareCluster. **Tags:** atp.Splitkey=moduleInstantiation
releaseNotes	Documentation	0..1	ref	This attribute allows for the explanations of changes since the previous version. The list of changes might require the creation of multiple paragraphs of test. **Tags:** atp.Splitkey=releaseNotes
typeApproval	String	0..1	attr	This attribute carries the homologation information that may be specific for a given country.
vendorId	PositiveInteger	0..1	attr	Vendor ID of this Implementation according to the AUTOSAR vendor list.
vendor Signature	CryptoService Certificate	0..1	ref	This reference identifies the certificate that represents the vendor's signature. **Tags:** atp.Splitkey=vendorSignature
version	StrongRevisionLabel String	0..1	attr	This attribute can be used to describe a version information for the enclosing SoftwareCluster.

Table 16.4: SoftwareCluster

[constr_10233] Multiplicity of the reference in the role `SoftwareCluster.vendorSignature`

Imposition time: `IT_Mani`

⌈For each `SoftwareCluster`, the reference in the role `vendorSignature` shall exist.⌋

[constr_10234] Multiplicity of attribute `SoftwareCluster.version`

Imposition time: `IT_Mani`

⌈For each `SoftwareCluster`, the attribute `version` shall exist.⌋

[constr_10235] Multiplicity of attribute `SoftwareCluster.vendorId`

Imposition time: `IT_Mani`

⌈For each `SoftwareCluster`, the attribute `vendorId` shall exist.⌋

The modeling of the relation `SoftwareCluster.claimedFunctionGroup` is modeled in Figure 16.6.

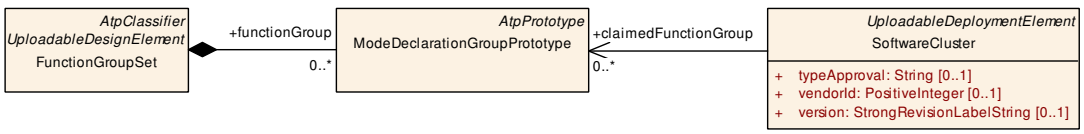


Figure 16.6: Modeling of the relation between `SoftwareCluster` and `Function Group`s

[constr_1747] Completeness of the `SoftwareCluster.version`

Imposition time: `IT_Mani`

⌈The `SoftwareCluster.version` shall contain all the following parts:

- Major version
- Minor version
- Patch version
- Additional labels for pre-release version and build metadata

⌋

Primitive	**StrongRevisionLabelString**
Note	This primitive represents a revision label which identifies an object under version control. It represents a pattern which requires three integer numbers separated by a dot, representing from left to right Major Version, MinorVersion, PatchVersion and additional labels for pre-release version and build metadata. Legal patterns are for example: 1.0.0-alpha+001 1.0.0+20130313144700 1.0.0-beta+exp.sha.5114f85 **Tags:** xml.xsd.customType=STRONG-REVISION-LABEL-STRING xml.xsd.pattern=(0|[1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)(-((0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*)(\.(0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*))*))?(\+([0-9a-zA-Z-]+(\.[0-9a-zA-Z-]+)*))? xml.xsd.type=string

Table 16.5: StrongRevisionLabelString

The build number does not necessarily have to be incremented consecutively between two builds. In some cases, the build number is created by hashing the build.

In such a case, it would not make sense to include the build number in a greater/less comparison, while a comparison for equality/inequality may well make sense. This aspect shall be taken into account when processing the values of attributes of type `StrongRevisionLabelString`.

Even if the AUTOSAR UCM is not used as the Update Management System on the Machine, the `PLATFORM_CORE` `SoftwareCluster` shall be defined for methodology reasons in the Manifest file. However, in such a case, it is allowed to define a single `PLATFORM_CORE` `SoftwareCluster` on the Machine that contains all software deployed on this machine (platform software and application software).

In addition to the standardized values, custom values can be defined for attribute `SoftwareCluster.category`.

In this case, however, it is important to use custom values that do not clash with future extensions of the standardized values. A good way to avoid a clash is to use specific prefixes or suffixes that identify a company or project name.

[constr_1788] Restriction to `SoftwareCluster` of `category` `PLATFORM_CORE`

Imposition time: `IT_Mani`

⌈On each `Machine`, exactly one `SoftwareCluster` of `category` `PLATFORM_CORE` shall be deployed.⌋

[TPS_MANI_01115] Specification of executable software within `SoftwareCluster` ⌈One of the most prominent contents of an uploadable software package is the reference to the executable software.

Within the definition of a `SoftwareCluster`, this reference is implicitly given by means of the reference `SoftwareCluster.containedProcess`.

The target of `SoftwareCluster.containedProcess` is a `Process` that represents an instance of the corresponding executable program (the software image), formalized as `Executable`⌋

The prominence of the dedicated reference to `Process` is amplified by the fact that it would have been technically possible to let `Process` inherit from `UploadablePackageElement` and thus include the referenced `Process`(es) in the bulk of references to other required model elements.

These references are formalized in two different forms. For technical reasons it is not possible to let all model elements that need to be immediately referenced by a `SoftwareCluster` inherit from `UploadablePackageElement`.

The main reason is that further model elements need to be referenced by a `SoftwareCluster` that are also used on the *AUTOSAR classic platform*.

In other words, it would be questionable to introduce the concept of an `UploadablePackageElement` into the scope of the *AUTOSAR classic platform* as an unwanted side effect of providing a definition of the `SoftwareCluster` on the *AUTOSAR classic platform*.

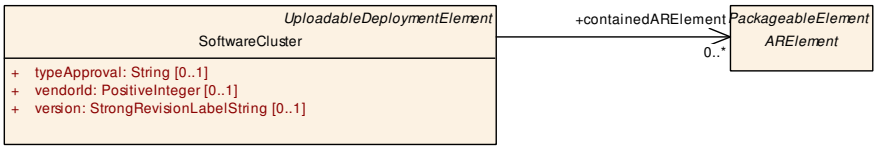


Figure 16.7: `SoftwareCluster` can reference `ARElement`

The scope of a single `SoftwareCluster` in terms of a relation to a `Machine` is that all software contained in one `SoftwareCluster` is supposed to be uploaded to one and only one `Machine`.

The definition of `SoftwareCluster` shall never include multiple `Machine`s. This is expressed in [constr_1536].

[constr_1536] Definition of `SoftwareCluster` applies for a single `Machine`

Imposition time: `IT_Mani`

⌈Within the scope of a `SoftwareCluster`, each `Process` referenced in the role `containedProcess` shall be mapped (e.g. by means of the existence of a `ProcessToMachineMapping`) to the same `Machine`.⌋

[TPS_MANI_01116] Reference to model elements included in an uploadable software package ⌈Beside the ability to explicitly reference a `Process` in the role

containedProcess it is possible to define the following references to required model elements:

- references to meta-classes derived from UploadablePackageElement are formalized by way of SoftwareCluster.containedPackageElement.
- references to meta-classes derived from ARElement are formalized by way of SoftwareCluster.containedARElement.

Technically, an UploadablePackageElement is also an ARElement, but it is still mandated to use the dedicated reference specifically for UploadablePackageElement.⌋

To exemplify the reference to UploadablePackageElement, Figure 16.5 contains a subclass of UploadablePackageElement: ServiceInstanceToPortPrototypeMapping.

The uploaded software needs to integrate with the communication stack, and ServiceInstanceToPortPrototypeMapping is a prominent model element for this purpose.

[TPS_MANI_01202] Semantics of reference in the role SoftwareCluster.moduleInstantiation ⌈By means of the reference SoftwareCluster.moduleInstantiation it is possible to express the need for updates of the platform infrastructure along with other resources referenced by the enclosing SoftwareCluster.⌋

[TPS_MANI_01218] Application of a cryptographic signature of SoftwareCluster ⌈A SoftwareCluster also shall be signed cryptographically. For this purpose, meta-class CryptoServiceCertificate is referenced in the role vendorSignature.⌋

[TPS_MANI_01219] License of software in included SoftwareCluster ⌈It is possible to refer to licenses for software included in a SoftwareCluster by means of a reference to meta-class Documentation in the role license.⌋

Class	**Documentation**			
Note	This meta-class represents the ability to handle a so called standalone documentation. Standalone means, that such a documentation is not embedded in another ARElement or identifiable object. The standalone documentation is an entity of its own which denotes its context by reference to other objects and instances. **Tags:** atp.recommendedPackage=Documentations			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
context	DocumentationContext	*	aggr	This is the context of the particular documentation. **Tags:** atp.Splitkey=context, context.shortName

▽

△

Class	**Documentation**			
documentation Content	PredefinedChapter	0..1	aggr	This is the content of the documentation related to the specified contexts. **Tags:** atp.Splitkey=documentationContent xml.sequenceOffset=200

Table 16.6: Documentation

Documentation is an ARElement that cannot be owned by a SoftwareCluster; the latter can only refer to it.

Once a given license is formalized by means of a Documentation, it is in general possible to refer to this formalization from within different SoftwareClusters.

[TPS_MANI_01220] Release notes of software in included SoftwareCluster ⌈It is possible to refer to release notes for software included in a SoftwareCluster by means of a reference to meta-class Documentation in the role releaseNotes.⌋

[constr_1566] Usage of SoftwareCluster.containedARElement

Imposition time: IT_Mani

⌈The reference SoftwareCluster.containedARElement shall not be used to refer to a SoftwareCluster or a SoftwareClusterDesign.⌋

The multiplicity of SoftwareCluster.design has been set to 0..* to support the following use cases:

- Both an OEM and a Tier-1 supplier want to provide a design for a SoftwareCluster, but the OEM does not allow the supplier to extend its design.

  The consequence is that the supplier would create its own SoftwareClusterDesign and let the SoftwareCluster refer to it. This means that the SoftwareCluster has two references in the role SoftwareCluster.design.

- An integrator decides to implement several SoftwareClusterDesigns with one SoftwareCluster.

16.3.2 Relevance of Software Cluster for Diagnostics

16.3.2.1 Diagnostic Props

[TPS_MANI_01349] Configuration of diagnostic-related properties of a SoftwareCluster ⌈The diagnostics-related properties of a SoftwareCluster are configured in the context of meta-class SoftwareClusterDiagnosticDeploymentProps, referenced in the role SoftwareCluster.diagnosticDeploymentProps.⌋

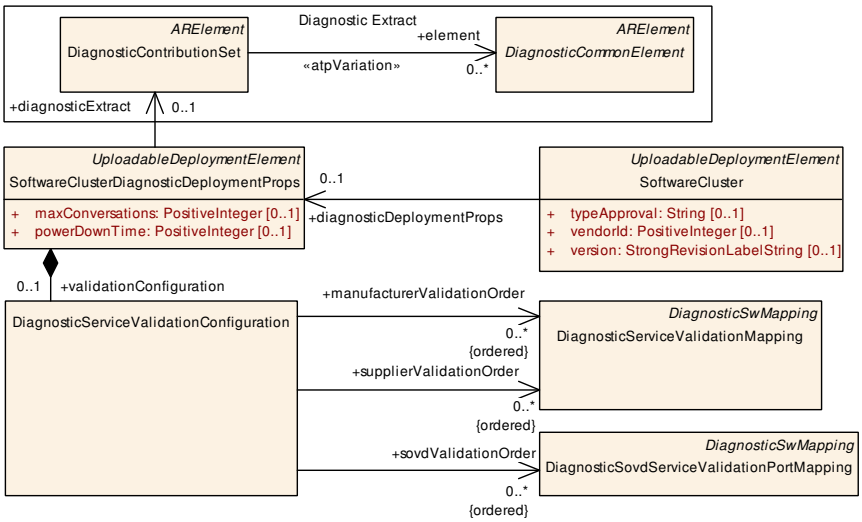


Figure 16.8: Modeling of diagnostic properties of a `SoftwareCluster`

[constr_10069] Existence of `SoftwareClusterDiagnosticDeployment-Props.powerDownTime`

Imposition time: `IT_Mani`

⌈The attribute `SoftwareClusterDiagnosticDeploymentProps.powerDown-Time` shall exist and have a value between 0 and 254 if the referenced `diagnostic-Extract` that in turn references in the role `element` a `DiagnosticEcuReset` where attribute `category` is set to the value `ENABLE_RAPID_POWER_SHUT_DOWN`.⌋

16.3.2.2 Relation to the Diagnostic Contribution Set

[TPS_MANI_01114] Relation of `DiagnosticContributionSet` to `Software-Cluster` ⌈In AUTOSAR, the formalization of the external behavior of the diagnostic stack is rooted in meta-class `DiagnosticContributionSet`.

On the *AUTOSAR classic platform* the scope of the “external behavior of the diagnostic stack” is represented by an entire ECU.

This relation changes on the *AUTOSAR adaptive platform* where each uploadable software package is shipped with the definition of the “external behavior of the diagnostic stack” **as far as the software in the scope of respective uploadable software package is concerned**.

To fully support the different approaches of *AUTOSAR classic platform* and *AUTOSAR adaptive platform* it is necessary to provide means for specifying a `Diagnostic-ContributionSet` for a given `SoftwareCluster`.

In particular, this relation is created by means of the reference `Software-Cluster.diagnosticDeploymentProps.diagnosticExtract`.⌋

The placement of the reference to the DiagnosticContributionSet at the SoftwareClusterDiagnosticDeploymentProps follows the pattern on the *AUTOSAR adaptive platform*, where target-configuration-related elements shall reference their corresponding design-level elements.

In addition, it is much easier to keep the definition of diagnostic target-configuration and diagnostic design in sync if the former has a direct reference to the latter.

This approach also works for all levels of granularity in the definition of diagnostics on the *AUTOSAR adaptive platform*:

- One SoftwareCluster represents a separately diagnosable entity with its own diagnostic address.

  In this case, the DiagnosticContributionSet referenced in the role SoftwareCluster.diagnosticDeploymentProps.diagnosticExtract is only referenced by this specific SoftwareCluster.

- All SoftwareClusters deployed to one Machine share the same SoftwareClusterDiagnosticDeploymentProps and therefore also the same DiagnosticContributionSet.

Class	**SoftwareClusterDiagnosticDeploymentProps**			
Note	This meta-class acts as the owner of all deployment-related diagnostic properties of a SoftwareCluster. **Tags:** atp.recommendedPackage=SoftwareClusterDiagnosticProps This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnostic Extract	DiagnosticContribution Set	0..1	ref	This reference identifies the DiagnosticContributionSet that is applicable for the referencing SoftwareCluster. **Tags:** atp.Splitkey=diagnosticExtract
max Conversations	PositiveInteger	0..1	attr	Maximum number of diagnostic Conversations supported by the Diagnostic Server Instance. This attribute has no relation to the definition of the maximum number of clients in DoIP context, configured by means of DoIpNetwork Configuration.maxTesterConnections.
powerDown Time	PositiveInteger	0..1	attr	This attribute indicates the minimum time of the stand-by sequence the server will remain in the power-down sequence. The unit is seconds.
validation Configuration	DiagnosticService ValidationConfiguration	0..1	aggr	This aggregation represents the ability to define the order of manufacturer and supplier validations in diagnostic management. **Tags:** atp.Splitkey=validationConfiguration

Table 16.7: SoftwareClusterDiagnosticDeploymentProps

16.3.2.3 Diagnostic Service Validation

The configuration of the diagnostic management is mostly done on design level by means of the diagnostic extract (see Chapter 4.10 and Chapter 5).

Some aspects, like the validation of service requests by manufacturer-specific or supplier-specific checks, can only be decided on target-configuration level and in the context of the enclosing SoftwareCluster, see Figure 16.8.

[TPS_MANI_01350] Semantics of meta-class DiagnosticServiceValidationConfiguration ⌈For the specific purpose of the validation prior to the execution of diagnostic services, meta-class DiagnosticServiceValidationConfiguration is defined.

The main purpose of DiagnosticServiceValidationConfiguration is to define the order in which manufacturer-specific (by means of the reference in the role manufacturerValidationOrder) and (separately, via the reference in the role supplierValidationOrder) supplier-specific checks are executed.⌋

[TPS_MANI_01449] Service validation in SOVD context ⌈In contrast to the service validation in a UDS context, there is no separate handling of manufacturer and supplier validation tasks in the context of SOVD.

The configuration of service validation in an SOVD context can therefore be implemented in just one ordered collection of references in the role DiagnosticServiceValidationConfiguration.sovdValidationOrder.⌋

Class	**DiagnosticServiceValidationConfiguration**			
Note	This meta-class has the ability to configure the order of manufacturer/supplier-checks. **Tags:** atp.recommendedPackage=DiagnosticValueConfigurations This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	SoftwareClusterDiagnosticDeploymentProps.validationConfiguration			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
manufacturer ValidationOrder (ordered)	DiagnosticService ValidationMapping	*	ref	This reference defines the order in which validations created by manufacturer are executed.
sovdValidation Order (ordered)	DiagnosticSovdService ValidationPortMapping	*	ref	This reference defines the order in which validations of SOVD requests are executed.
supplier ValidationOrder (ordered)	DiagnosticService ValidationMapping	*	ref	This reference defines the order in which validations created by supplier are executed.

Table 16.8: DiagnosticServiceValidationConfiguration

[constr_10065] Validity of DiagnosticServiceValidationConfiguration.manufacturerValidationOrder

Imposition time: IT_Mani

⌈Any DiagnosticServiceValidationConfiguration.manufacturerValidationOrder shall only refer to a DiagnosticServiceValidationMapping where attribute category has been set to MANUFACTURER_VALIDATION.⌋

[constr_10066] Validity of `DiagnosticServiceValidationConfiguration.supplierValidationOrder`

Imposition time: `IT_Mani`

⌈Any `DiagnosticServiceValidationConfiguration.supplierValidationOrder` shall only refer to a `DiagnosticServiceValidationMapping` where attribute `category` has been set to `SUPPLIER_VALIDATION`.⌋

16.3.3 Contract between Software Cluster and Software Cluster Design

As already mentioned in [TPS_MANI_01112], there is a **certain level of contractual obligation** in the relation between a `SoftwareClusterDesign` and a `SoftwareCluster` that references said `SoftwareClusterDesign` in the role `design`.

The reason for the existence of the contractual obligation is the definition of dependencies between `SoftwareClusterDesign`s, as described in Chapter 8.4. The definition of dependencies is usually based on some characteristics of the pair of `SoftwareClusterDesign` and `SoftwareCluster`.

If, for example, a `SoftwareClusterDesign` imposes a dependency on another `SoftwareClusterDesign`, then this relation could be based on the existence of a certain `ProcessDesign` that is referenced by the `SoftwareClusterDesign` that imposes the dependency.

But if the corresponding `SoftwareCluster` does not refer to a `Process` that corresponds to the `ProcessDesign` that gave rise to the existence of the `SoftwareClusterDependencyFormula`, then the dependency does not transfer to the `SoftwareCluster`.

On the other hand, if the dependency defined in the form of a `SoftwareClusterDependencyFormula` does not transfer, then the definition of the dependency on the level of `SoftwareClusterDesign` is worthless and at worst creates confusion.

However, the exact (formal) nature of the contract between `SoftwareCluster` and `SoftwareClusterDesign` is subject to future refinement.

16.3.4 References between Software Clusters

There are several strong use cases for referencing into different `SoftwareCluster`s, for example:

- Reference from a `RequiredApServiceInstance` to a `ProvidedApServiceInstance` defined in the context of a “host” `SoftwareCluster`.
- Reference to `CommunicationConnector`s defined on `Machine` level from within application `SoftwareCluster`s.

- Reference from a Process in one SoftwareCluster to an Executable in another SoftwareCluster with the semantics that the referencing Process is just another instance of the Executable.

To support such use cases, AUTOSAR provides the definition of dependencies among SoftwareClusterDesigns such that a SoftwareClusterDesign that contains a reference can define a dependency to another SoftwareClusterDesign that contains the referenced object.

[TPS_MANI_01329] Reference to model elements in different SoftwareClusters ⌈If a model element inside a given SoftwareCluster defines a reference to another model element and the referenced model element is contained in a different SoftwareCluster, then the SoftwareCluster that contains the referencing model element shall establish a dependency to the other SoftwareCluster on the level of the respective SoftwareClusterDesigns by means of the aggregation of SoftwareClusterDependencyFormula in the role SoftwareClusterDesign.dependsOn.⌋

[constr_1784] Restriction for the reference to UploadableExclusivePackageElement

Imposition time: IT_Mani

⌈A reference to an UploadableExclusivePackageElement shall not cross the boundary of the enclosing SoftwareCluster, i.e. the target UploadableExclusivePackageElement of such a reference shall not be located in a different SoftwareCluster than the owner of the reference.⌋

Class	***UploadableExclusivePackageElement*** (abstract)			
Note	This meta-class represents an abstract base class for an uploadable package element that is not supposed to be referenced from a different software cluster. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	*PersistencyDeployment*, *PersistencyPortPrototypeToDeploymentMapping*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 16.9: UploadableExclusivePackageElement

Regardless of whether a dependency relation is defined, [constr_1784] forbids a reference across SoftwareClusters to an UploadableExclusivePackageElement.

Referencing from one SoftwareCluster into another SoftwareCluster is only allowed if the design of the referenced SoftwareCluster is in the list of dependent SoftwareCluster.designs for the referencing SoftwareCluster.

This restriction is formalized in [constr_1785].

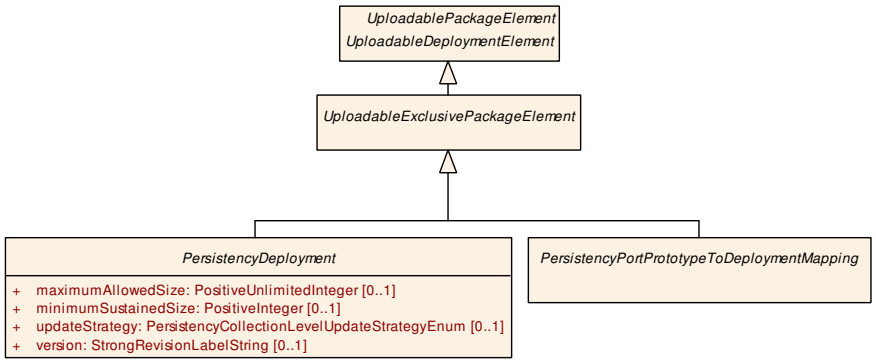


Figure 16.9: Modeling of the `UploadableExclusivePackageElement`

[constr_1785] Restriction regarding the reference into another `SoftwareCluster`

Imposition time: `IT_Mani`

⌈A reference from an element in one `SoftwareCluster` to an element located in another `SoftwareCluster` shall only exist if the `design` of the `SoftwareCluster` that owns the referenced element is referenced by a `SoftwareClusterDependencyCompareCondition` in the context of the mentioned `SoftwareClusterDependencyFormula` in the role `part.softwareClusterDesign`.

[constr_1784] applies.⌋

16.3.5 Software Cluster Artifact Checksum

[TPS_MANI_01345] Ability to attach checksums to `SoftwareCluster` ⌈Meta-class `SoftwareCluster` supports the collection of checksums for artifacts like binary executable files, libraries, or persistency files by means of the aggregation of meta-class `ArtifactChecksum` in the role `artifactChecksum`.⌋

Class	**ArtifactChecksum**			
Note	This meta-class provides the ability to associate a checksum with a given artifact identified by its URI. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SoftwareCluster.artifactChecksum			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checksumValue	String	0..1	attr	This attributes carries the serialized checksum of the corresponding artifact.
uri	UriString	0..1	attr	This attribute represents the URI of the artifact on which the checksum shall be computed.

Table 16.10: ArtifactChecksum

[TPS_MANI_01346] No formal definition of checksum algorithm ⌈The checksum algorithm used for computing the `ArtifactChecksum.checksumValue` is not formally defined.

A description of the algorithm shall be contained in the CryptoProvider.cryptoProviderDocumentation of the CryptoProvider that is associated with a specific ArtifactChecksum via the existence of ArtifactChecksumToCryptoProviderMapping.⌋

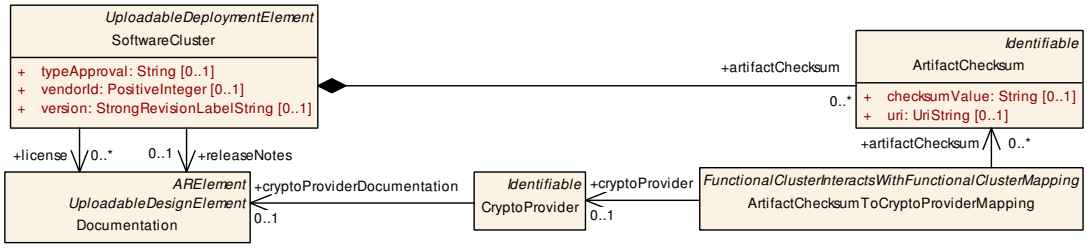


Figure 16.10: Modeling of the artifact checksum of a SoftwareCluster

Class	**ArtifactChecksumToCryptoProviderMapping**			
Note	This meta-class provides the ability to associate a CryptoProvider with a collection of the checksums computed for artifacts. **Tags:** atp.recommendedPackage=FCInteractions This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
artifact Checksum	ArtifactChecksum	*	ref	This reference identifies the collection of Artifact Checksums associated with the corresponding Crypto Provider. **Tags:** atp.Splitkey=artifactChecksum
cryptoProvider	CryptoProvider	0..1	ref	This reference identifies the applicable CryptoProvider. **Tags:** atp.Splitkey=cryptoProvider

Table 16.11: ArtifactChecksumToCryptoProviderMapping

16.3.6 Software Cluster Artifact Locator

[TPS_MANI_01381] Semantics of meta-class ArtifactLocator ⌈Meta-class ArtifactLocator represents a generic approach to configure a location of a referable model element in the role representedModelElement by means of attribute uri.⌋

The modeling of meta-class ArtifactLocator as a generic concept is depicted in Figure 16.11.

The association between an artifact and its location is a 1:1 relation. It is not allowed to associate a second (or more) location to a model element that is already the target of a reference in the role ArtifactLocator.representedModelElement.

[constr_10380] Target of ArtifactLocator.representedModelElement

Imposition time: IT_Mani

⌈For all ArtifactLocators aggregated by a given SoftwareCluster, the target of a reference in the role representedModelElement of one ArtifactLocator (aggregated by the enclosing SoftwareCluster) shall not be the target of another reference in the role representedModelElement from another ArtifactLocator aggregated by the same SoftwareCluster.⌋

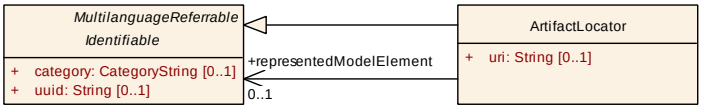


Figure 16.11: Modeling of the artifact location of a model element that belongs to a SoftwareCluster

Class	**ArtifactLocator**			
Note	This meta-class has the ability to define the location of an artifact that is represented by a model element, e.g. Executable. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SoftwareCluster.artifactLocator, SoftwarePackage.artifactLocator			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
represented ModelElement	Identifiable	0..1	ref	This reference identifies the model element that is represented by the artifact. **Tags:** atp.Splitkey=representedModelElement
uri	String	0..1	attr	This attribute describes the location of the artifact.

Table 16.12: ArtifactLocator

[constr_10381] Existence of attribute ArtifactLocator.uri

Imposition time: IT_Mani

⌈For each ArtifactLocator, the attribute uri shall exist.⌋

[constr_10382] Existence of attribute ArtifactLocator.representedModelElement

Imposition time: IT_Mani

⌈For each ArtifactLocator, the attribute representedModelElement shall exist.⌋

Meta-class ArtifactLocator is used in the context of the modeling of a SoftwareCluster to facilitate the configuration of model elements that belong to the SoftwareCluster and for which a representation in the software exists that can have a “location” (e.g. a file that contains executable code).

One example for the application of ArtifactLocator in the context of the modeling of a SoftwareCluster is the configuration of the location of binary executable code that corresponds to an Executable, which is claimed by a SoftwareCluster via

the indirection of SoftwareCluster.containedProcess that in turn references the Executable in the role executable.

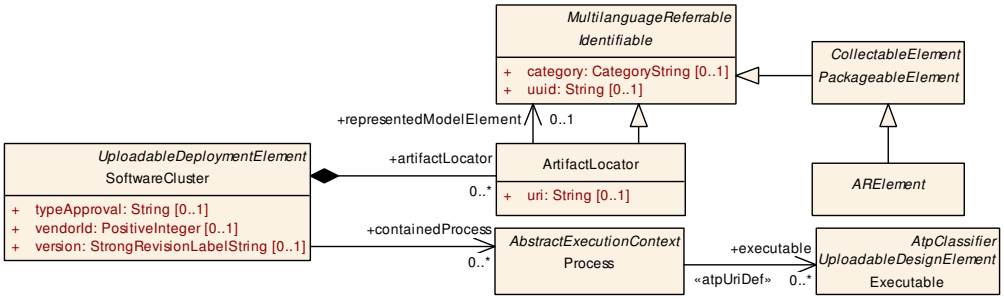


Figure 16.12: Modeling of the location of an Executable that (via a containedProcess) belongs to a SoftwareCluster

It is assumed that the entire code represented by an Executable is locatable in one place described by the attribute uri.

This relation is depicted in Figure 16.12. The ability to refer to an Executable in the role representedModelElement is inherited by Executable from its superclass Identifiable.

[TPS_MANI_01382] Location of artifact that contains executable code represented by Executable ⌈The location of the executable code that is represented by Executable can be identified by means of the aggregation SoftwareCluster.artifactLocator where attribute ArtifactLocator.representedModelElement.dest is set to the value EXECUTABLE.⌋

16.4 Software Package

The existence of the SoftwareCluster by itself is not sufficient for installation.

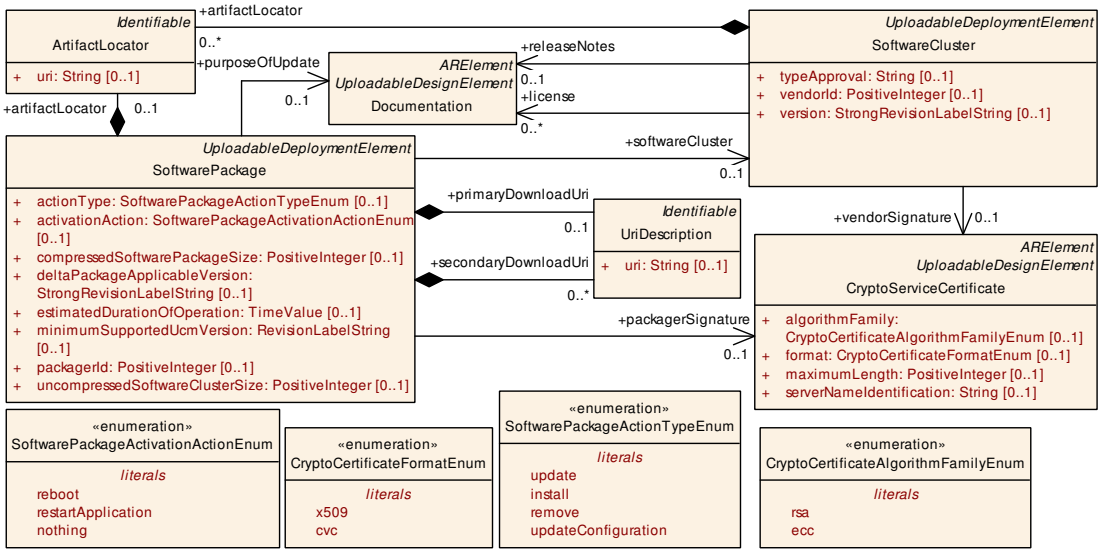


Figure 16.13: Modeling of SoftwarePackage

The SoftwareCluster gets wrapped into a so-called SoftwarePackage that comes with its own manifest format that is at least partly standardized.

The difference between the semantics of a SoftwareCluster and the semantics of SoftwarePackage is that a SoftwareCluster focuses on the structure of the software itself while the SoftwarePackage is created to handle the logistics aspect of the software installation.

[TPS_MANI_01221] Semantics of meta-class SoftwarePackage ⌈The purpose of meta-class SoftwarePackage is to cover the "logistics" aspect of the software installation procedure.⌋

Class	**SoftwarePackage**			
Note	This meta-class represents the ability to formalize the content of a software package. **Tags:** atp.recommendedPackage=SoftwarePackages This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
actionType	SoftwarePackageAction TypeEnum	0..1	attr	This attribute defines the action to be taken in the step of processing the enclosing SoftwarePackage.
activationAction	SoftwarePackage ActivationActionEnum	0..1	attr	This attribute governs the action to be taken after the installation of the SoftwareCluster completed.
artifactLocator	ArtifactLocator	0..1	aggr	This attribute identifies the software package at configuration time, out of the context of an AUTOSAR model. **Tags:** atp.Splitkey=artifactLocator, artifactLocator.shortName
compressed Software PackageSize	PositiveInteger	0..1	attr	This size represents the size of the compressed Software Package.
deltaPackage Applicable Version	StrongRevisionLabel String	0..1	attr	This attribute identifies the version of the included SoftwareCluster for which the enclosing SoftwarePackage can be used as a delta update
estimated DurationOf Operation	TimeValue	0..1	attr	This attribute provides an estimation about how long the operation of the SoftwarePackage is going to take for its transfer, processing and activation when updated standalone (not within an update campaign)
minimum SupportedUcm Version	RevisionLabelString	0..1	attr	This attribute identifies the minimum supported version of the UCM for this SoftwarePackage.
packagerId	PositiveInteger	0..1	attr	This attribute identifies Id of the organization that provides the packager generating the SoftwarePackage.
packager Signature	CryptoService Certificate	0..1	ref	This reference identifies the certificate that represents the packager's signature. **Tags:** atp.Splitkey=packagerSignature
primary DownloadUri	UriDescription	0..1	aggr	This attribute represents the description of the primary download location. **Tags:** atp.Splitkey=primaryDownloadUri, primaryDownloadUri.shortName



△

Class	**SoftwarePackage**			
purposeOf Update	Documentation	0..1	ref	The referenced Documentation is supposed to provide a description of the purpose of the update. **Tags:** atp.Splitkey=purposeOfUpdate
secondary DownloadUri	UriDescription	*	aggr	This attribute represents the description of the secondary download locations. **Tags:** atp.Splitkey=secondaryDownloadUri, secondary DownloadUri.shortName
softwareCluster	SoftwareCluster	0..1	ref	This reference identifies the SoftwareCluster that belongs to the SoftwarePackage. The nature of this relation is actually more like an aggregation than a reference. But the relation is still modelled as a reference because two ARElements cannot aggregate each other. **Tags:** atp.Splitkey=softwareCluster
uncompressed SoftwareCluster Size	PositiveInteger	0..1	attr	This attribute gives an indication about the storage that has to be available on the target.

Table 16.13: SoftwarePackage

[constr_10240] Multiplicity of attribute `SoftwarePackage.actionType`

Imposition time: `IT_Mani`

⌈For each `SoftwarePackage`, the attribute `actionType` shall exist.⌋

[constr_10242] Multiplicity of attribute `SoftwarePackage.minimumSupported-UcmVersion`

Imposition time: `IT_Mani`

⌈For each `SoftwarePackage`, the attribute `minimumSupportedUcmVersion` shall exist.⌋

[constr_10243] Multiplicity of attribute `SoftwarePackage.packagerId`

Imposition time: `IT_Mani`

⌈For each `SoftwarePackage`, the attribute `packagerId` shall exist.⌋

[constr_10244] Multiplicity of the reference in the role `SoftwarePackage.packagerSignature`

Imposition time: `IT_Mani`

⌈For each `SoftwarePackage`, the reference in the role `packagerSignature` shall exist.⌋

[constr_10245] Multiplicity of the reference in the role `SoftwarePackage.softwareCluster`

Imposition time: `IT_Mani`

⌈For each `SoftwarePackage`, the reference in the role `softwareCluster` shall exist.⌋

[constr_10246] Multiplicity of the attribute SoftwarePackage.uncompressed-SoftwareClusterSize

Imposition time: IT_Mani

⌈For each SoftwarePackage, the attribute uncompressedSoftwareCluster-Size shall exist.⌋

Furthermore, a SoftwareCluster is derived from base class Packageable-Element which is only aggregated by ARPackage. Subclasses of Packageable-Element – by convention – shall not be aggregated by any other meta-class.

[TPS_MANI_01222] Cryptographic signature of a SoftwarePackage ⌈A SoftwarePackage shall be signed cryptographically. For this purpose, meta-class CryptoServiceCertificate is referenced in the role packagerSignature.⌋

[TPS_MANI_01223] Semantics of attribute SoftwarePackage.packagerId ⌈Attribute SoftwarePackage.packagerId contains the value of the AUTOSAR vendor Id of the organization that created software tool that created the Software-Package.⌋

Primitive	**RevisionLabelString**
Note	This primitive represents an internal AUTOSAR revision label which identifies an engineering object. It represents a pattern which • supports three integers representing from left to right MajorVersion, MinorVersion, PatchVersion. • may add an application specific suffix separated by one of ".", "_", ";". Legal patterns are for example: • 4.0.0 • 4.0.0.1234565 • 4.0.0_vendor specific;13 • 4.0.0;12 **Tags:** xml.xsd.customType=REVISION-LABEL-STRING xml.xsd.pattern=[0-9]+\.[0-9]+\.[0-9]+([\._;].*)? xml.xsd.type=string

Table 16.14: RevisionLabelString

A UCM can only accept packages that are generated by a packaging tool developed by the same organization that also developed the UCM itself. The vendor of the SoftwareCluster contained in the SoftwarePackage can be different.

[TPS_MANI_01225] Actions taken during installation of a SoftwarePackage ⌈It is necessary to define the concrete activity that shall be taken to handle the Software-Package on the target machine. Possible actions are:

- Do a clean installation of a SoftwareCluster.
- Update a previously installed SoftwareCluster.
- Remove a SoftwareCluster.

These options are formalized by means of meta-class SoftwarePackageActionTypeEnum and attribute SoftwarePackage.actionType.⌋

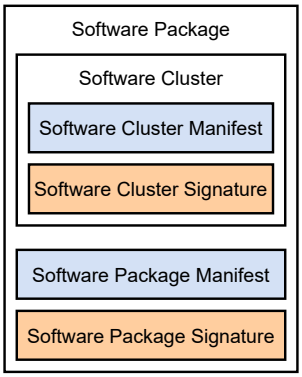


Figure 16.14: Conceptual relation of SoftwarePackage and SoftwareCluster

Enumeration	**SoftwarePackageActionTypeEnum**
Note	This enumeration provides a choice of possible actions for the handling of a software package. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	SoftwarePackage.actionType
Literal	***Description***
install	Do a clean installation of a SoftwareCluster. **Tags:** atp.EnumerationLiteralIndex=1
remove	Remove a SoftwareCluster. **Tags:** atp.EnumerationLiteralIndex=2
update	Update a previously installed SoftwareCluster. **Tags:** atp.EnumerationLiteralIndex=0
update Configuration	Execute a configuration update. **Tags:** atp.EnumerationLiteralIndex=3

Table 16.15: SoftwarePackageActionTypeEnum

[TPS_MANI_01344] Actions taken after installation of a SoftwarePackage ⌈After a SoftwarePackage has been installed on the target machine it is possible to execute one of the following actions:

- Reboot the target platform.
- Restart the installed SoftwareCluster.
- Do nothing.

These options are formalized by means of meta-class SoftwarePackageActivationActionEnum and attribute SoftwarePackage.activationAction.⌋

Enumeration	**SoftwarePackageActivationActionEnum**
Note	This enumeration provides a choice of possible actions to be executed on installing a Software Package to a target Machine. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	SoftwarePackage.activationAction
Literal	***Description***
nothing	The installation has no immediate consequences in terms of other software on the target. **Tags:** atp.EnumerationLiteralIndex=2
reboot	Reboot the whole Machine. **Tags:** atp.EnumerationLiteralIndex=0
restartApplication	Restart the application software on the target Machine. **Tags:** atp.EnumerationLiteralIndex=1

Table 16.16: SoftwarePackageActivationActionEnum

Class	**UriDescription**			
Note	This meta-class can be used to define a URI. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SoftwarePackage.primaryDownloadUri, SoftwarePackage.secondaryDownloadUri			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
uri	String	0..1	attr	This attribute represents the actual URI. There are intentionally no format restrictions for how the URI shall be created.

Table 16.17: UriDescription

[constr_10653] Existence of attribute `UriDescription.uri`

Imposition time: `IT_Mani`

⌈For each `UriDescription`, the attribute `uri` shall exist.⌋

16.5 Vehicle Package

16.5.1 Overview

The ability to handle `SoftwarePackage`s is the prerequisite for an important further step: the execution of an **update campaign that applies to the whole vehicle**. The basis for the update campaign is the definition of meta-class `VehiclePackage`.

[TPS_MANI_01290] `VehiclePackage` names affected UCMs ⌈Meta-class `VehiclePackage` has the ability to describe the set of UCMs that are affected by the update campaign by means of aggregating meta-class `UcmDescription` in the role `ucm`.⌋

[TPS_MANI_01291] Identification of an actual UCM in the context of an update campaign ⌈It is necessary to unambiguously identify the individual UCMs that are

affected in the update campaign. For this purpose, meta-class UcmDescription defines attribute identifier.

By means of the reference to UcmModuleInstantiation in the role ucmModuleInstantiation it is in addition possible to identify the actual UCMs (represented by a UcmModuleInstantiation) that are relevant for the update campaign.

Resolving the reference requires access to the manifest model of the target Machine.⌋

[constr_1731] Value of attribute UcmDescription.identifier in the scope of a VehiclePackage

Imposition time: IT_Mani

⌈Within the scope of any given VehiclePackage, no two UcmDescriptions shall define the same value of attribute identifier.⌋

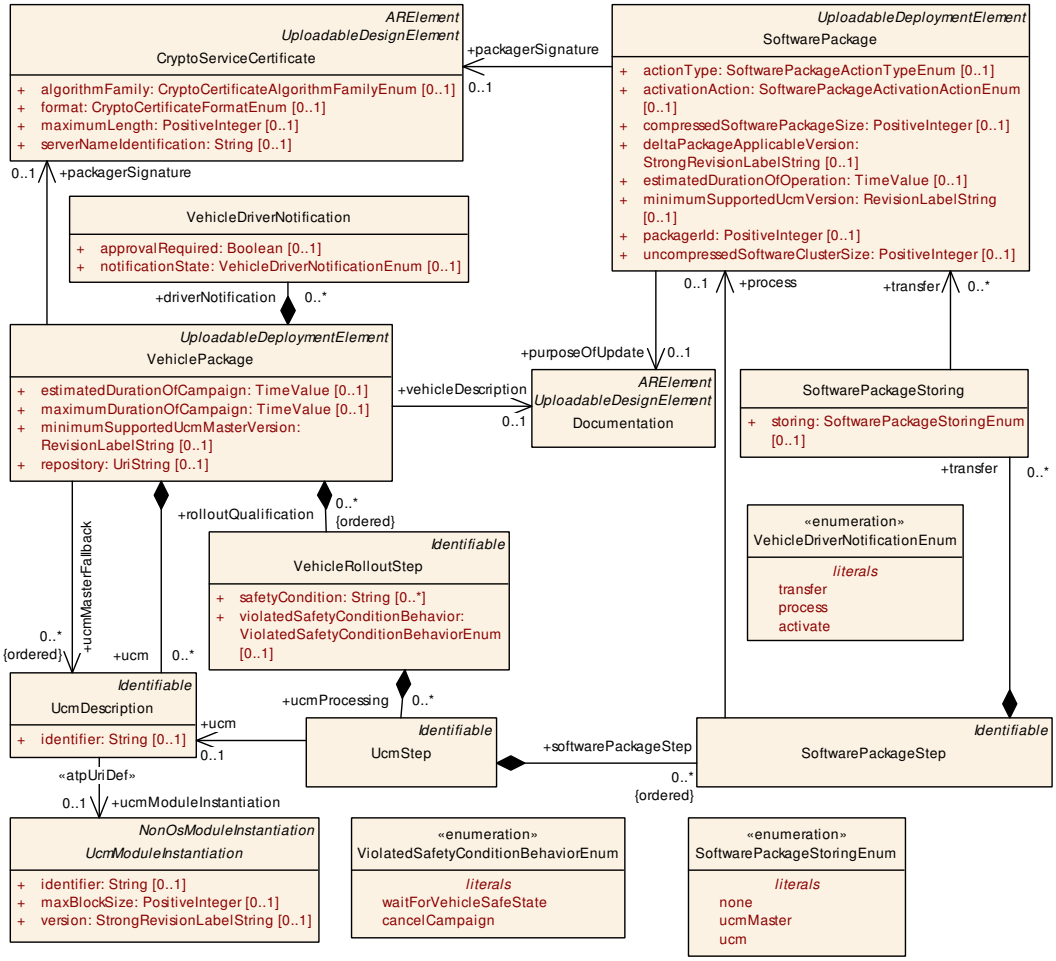


Figure 16.15: Modeling of VehiclePackage

[TPS_MANI_01292] Definition of fallback-order for UCM master ⌈The update campaign is executed under the management of one UCM that acts as a “master UCM”.

If this UCM goes down for some reason, VehiclePackage has the ability to define an **ordered** list of other candidates for becoming the "master UCM" by means of the reference to meta-class UcmDescription in the role ucmMasterFallback.⌋

[TPS_MANI_01294] Update campaign depends on driver's acceptance ⌈It is not possible to arbitrarily trigger the execution of an update campaign at any time. It is the prerogative of the vehicle driver to decide on the amount and consequences of the UCM activities with respect to an update campaign.

For this purpose VehiclePackage aggregates meta-class VehicleDriver-Notification in the role driverNotification.⌋

Class	**VehiclePackage**			
Note	This meta-class represents the ability to define a vehicle package for executing an update campaign. **Tags:** atp.recommendedPackage=VehiclePackages This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
driver Notification	VehicleDriver Notification	*	aggr	This aggregation provides the ability to configure the necessary driver notifications. **Tags:** atp.Splitkey=driverNotification
estimated DurationOf Campaign	TimeValue	0..1	attr	This attribute provides an estimation about how long the campaign based on the VehiclePackage is going to take.
maximum DurationOf Campaign	TimeValue	0..1	attr	Maximum time allowed for the campaign to be active until UCM Master automatically cancels the campaign.
minimum SupportedUcm MasterVersion	RevisionLabelString	0..1	attr	This attribute identifies the minimum supported version of the UCM Master for this VehiclePackage.
packager Signature	CryptoService Certificate	0..1	ref	This reference identifies the certificate that represents the packager's signature. **Tags:** atp.Splitkey=packagerSignature
repository	UriString	0..1	attr	This attribute identifies the repository where the Vehicle Package is stored.
rollout Qualification (ordered)	VehicleRolloutStep	*	aggr	This represents the rollout qualification.
ucm	UcmDescription	*	aggr	This aggregation represents the UcmDescriptions to be considered in the context of the VehiclePackage. **Tags:** atp.Splitkey=ucm, ucm.shortName
ucmMaster Fallback (ordered)	UcmDescription	*	ref	This reference lists the fallback order of Ucms that can take over the master role if the master goes down.
vehicle Description	Documentation	0..1	ref	This reference identifies the vehicle description. **Tags:** atp.Splitkey=vehicleDescription

Table 16.18: VehiclePackage

[constr_10247] Multiplicity of reference in the role VehiclePackage.packagerSignature

Imposition time: IT_Mani

⌈For each VehiclePackage, the reference in the role packagerSignature shall exist.⌋

Class	UcmDescription			
Note	This meta-class represents the ability to define an identifier for a given UCM. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	VehiclePackage.ucm			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
identifier	String	0..1	attr	This attribute represents the unique identification of the UcmIdentifier.
ucmModule Instantiation	UcmModuleInstantiation	0..1	ref	This reference identifies the applicable UcmModule Instantiation. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=ucmModuleInstantiation

Table 16.19: UcmDescription

[constr_10248] Multiplicity of attribute named UcmDescription.identifier

Imposition time: IT_Mani

⌈For each UcmDescription, the attribute named identifier shall exist.⌋

Class	VehicleDriverNotification			
Note	This meta-class provides the ability to configure a notification of the vehicle driver with respect to the update of vehicle software. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	VehiclePackage.driverNotification			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
approval Required	Boolean	0..1	attr	This attribute controls whether approval is required for the driver notification.
notificationState	VehicleDriver NotificationEnum	0..1	attr	This attribute is used to configure the notification state.

Table 16.20: VehicleDriverNotification

[constr_10249] Multiplicity of attribute named VehicleDriverNotification.approvalRequired

Imposition time: IT_Mani

⌈For each VehicleDriverNotification, the attribute named approvalRequired shall exist.⌋

[constr_10250] Multiplicity of attribute named `VehicleDriverNotification.notificationState`

Imposition time: `IT_Mani`

⌈For each `VehicleDriverNotification`, the attribute named `notificationState` shall exist.⌋

Enumeration	**VehicleDriverNotificationEnum**
Note	This meta-class provides available options for vehicle driver notification. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	VehicleDriverNotification.notificationState
Literal	***Description***
activate	Software package shall be activated. **Tags:** atp.EnumerationLiteralIndex=2
process	Processing of software package shall be executed **Tags:** atp.EnumerationLiteralIndex=1
transfer	Software shall be transferred to the vehicle. **Tags:** atp.EnumerationLiteralIndex=0

Table 16.21: VehicleDriverNotificationEnum

16.5.2 VehicleRolloutStep

[TPS_MANI_01295] Semantics of `VehicleRolloutStep` ⌈The purpose of an update campaign is to roll out the installation or update of `SoftwarePackage`s in the context of given UCMs. Each `VehicleRolloutStep` may apply to several UCMs at the same time.

The activation of the `SoftwarePackage`s processed in the context of the enclosing `VehiclePackage` is triggered as the last element of the `rolloutQualification` is processed.⌋

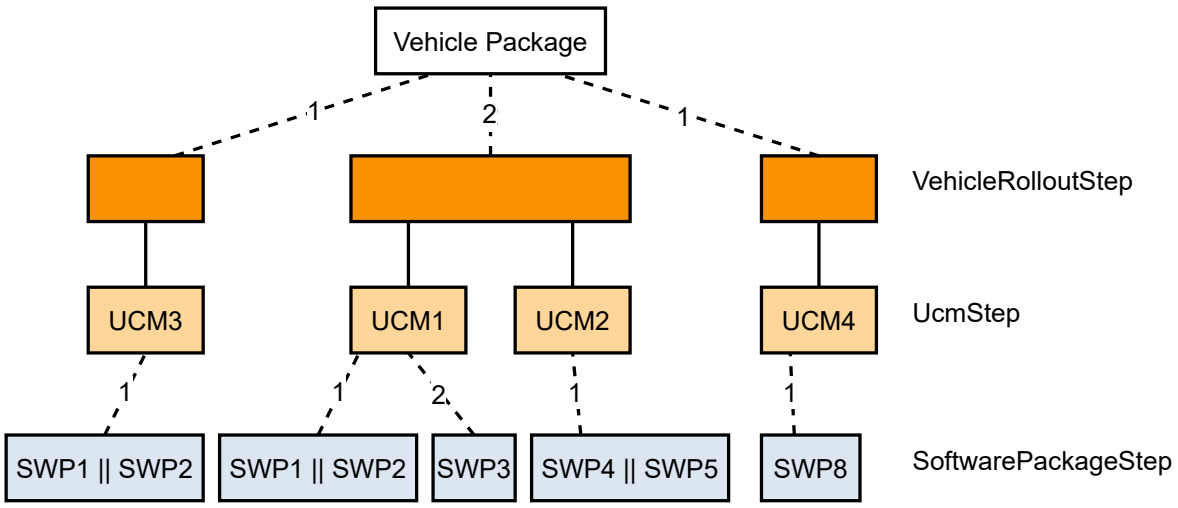


Figure 16.16: Conceptual view on an update campaign

Figure 16.16 takes a conceptual view on the structure of an update campaign and associates the relevant aspects of this view to meta-classes explained in this chapter. Associations that are labeled by a number indicate that an ordering is implied with the respective step.

For example, the execution of the update campaign happens in dedicated steps formalized as `VehicleRolloutStep`, as explained in [TPS_MANI_01295]. Each of the three steps sketched in the picture would be modeled as an individual `VehicleRolloutStep`.

The rollout action is formalized by `UcmStep.softwarePackageStep`. In other words, it is possible to specify a different `softwarePackageStep` for each individual UCM.

The individual `VehicleRolloutStep`s are executed in the order in which they are aggregated at the enclosing `VehiclePackage`. This aspect is explained in more detail in [TPS_MANI_01296].

[TPS_MANI_01296] Ordered execution of rollout steps in an update campaign ⌈The individual `VehicleRolloutStep`s defined in the context of a given `VehiclePackage` are executed in the defined order and therefore the aggregation of `VehicleRolloutStep` at `VehiclePackage` is `ordered`.⌋

Class	**VehicleRolloutStep**			
Note	This meta-class represents the ability to define a rollout-condition for a vehicle update campaign. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	VehiclePackage.rolloutQualification			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
safetyCondition	String	*	attr	This attribute represents a list of textual safety conditions (e.g.: close the driver window) that need to be fulfilled before the rollout step can proceed and need to be maintained while the campaign's rolloutQualification is executed.
ucmProcessing	UcmStep	*	aggr	This aggregation collects the UcmProcessingSteps that make up the rollout step. **Tags:** atp.Splitkey=ucmProcessing, ucmProcessing.shortName
violatedSafetyCondition Behavior	ViolatedSafetyCondition BehaviorEnum	0..1	attr	This attribute provides options for the configuration of the reaction to a violated safety condition.

Table 16.22: VehicleRolloutStep

Enumeration	**ViolatedSafetyConditionBehaviorEnum**
Note	This enumeration provides formal options for the configuration of the reaction to a violated safety condition. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	VehicleRolloutStep.violatedSafetyConditionBehavior
Literal	***Description***

▽

△

Enumeration	ViolatedSafetyConditionBehaviorEnum
cancelCampaign	This enumerator supports the ability to cancel the update campaign in response to a violated safety condition. **Tags:** atp.EnumerationLiteralIndex=1
waitForVehicleSafe State	This enumerator supports the ability to wait for the vehicle to acquire a safe state in response to a violated safety condition. **Tags:** atp.EnumerationLiteralIndex=0

Table 16.23: ViolatedSafetyConditionBehaviorEnum

[constr_10429] Existence of attribute `VehicleRolloutStep.violatedSafetyConditionBehavior`

Imposition time: `IT_Mani`

⌈For each `VehicleRolloutStep`, the attribute `violatedSafetyConditionBehavior` shall exist.⌋

[constr_10547] Reference in the role `UcmStep.ucm` in the context of the enclosing `VehicleRolloutStep`

Imposition time: `IT_Mani`

⌈Each `VehiclePackage.ucm` shall be referenced in the role `UcmStep.ucm` **at most once** in the context of an enclosing `VehicleRolloutStep`.⌋

16.5.3 UcmStep

[TPS_MANI_01297] Semantics of meta-class `UcmStep` ⌈Each `VehicleRolloutStep` consists of a number of `UcmStep`s (aggregated by `VehicleRolloutStep` in the role `ucmProcessing`). Each `UcmStep` refers to a specific UCM (represented by `UcmDescription`) in the role `ucm`.⌋

[TPS_MANI_01298] No ordering of `VehicleRolloutStep.ucmProcessing` ⌈Each `UcmStep` defined in the context of an enclosing `VehicleRolloutStep` can be handled without the consideration of a dedicated order. Therefore, the aggregation `VehicleRolloutStep.ucmProcessing` is not labeled as `ordered`.⌋

[constr_10530] Each UCM shall process at least one software package

Imposition time: `IT_Mani`

⌈Within the context of all `UcmStep`s referencing a specific `UcmDescription` in the role `ucm`, **at least one** `SoftwarePackageStep` shall exist that defines a reference to a `SoftwarePackage` in the role `process`.⌋

Class	UcmStep			
Note	This meta-class represents one rollout step in which software packages are processed on a specific Ucm. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	VehicleRolloutStep.ucmProcessing			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
software PackageStep (ordered)	SoftwarePackageStep	*	aggr	This aggregation represents the sequence of activities to be carried out in the context of the respective UCM.
ucm	UcmDescription	0..1	ref	This reference identifies the UCM for which the rollout step applies. **Tags:** atp.Splitkey=ucm

Table 16.24: UcmStep

16.5.4 SoftwarePackageStep

[TPS_MANI_01299] Aggregation of `SoftwarePackageStep`s at `UcmStep` ⌈Each `UcmStep` consists of an `ordered` collection of `SoftwarePackageStep`s. This means that the order in which `SoftwarePackage`s are handled in the scope of one `UcmStep` is significant.⌋

Class	SoftwarePackageStep			
Note	This meta-class represents the configuration of an activation step in the context of software package activation. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	UcmStep.softwarePackageStep			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
process	SoftwarePackage	0..1	ref	This reference identifies the SoftwarePackage to be processed in the enclosing SoftwarePackageStep. **Tags:** atp.Splitkey=process
transfer	SoftwarePackage Storing	*	aggr	This aggregation clarifies the storing of the Software Package. **Tags:** atp.Splitkey=transfer

Table 16.25: SoftwarePackageStep

[TPS_MANI_01300] Semantics of reference `SoftwarePackageStep.transfer.transfer` ⌈The reference `SoftwarePackageStep.transfer.transfer` identifies the `SoftwarePackage`s that are supposed to be transferred in the context of the enclosing `SoftwarePackageStep`.

It is explicitly supported that `SoftwarePackage`s are transferred in parallel and therefore the multiplicity of the reference in the role `transfer` has been set to 0..*.⌋

[TPS_MANI_01301] Semantics of aggregation `SoftwarePackageStep.transfer` ⌈By means of the aggregation of `SoftwarePackageStoring` it is possible to

specify for each individual SoftwarePackage whether and where the SoftwarePackage is stored in the vehicle.

This information is specifically provided by attribute SoftwarePackageStoring.storing of type SoftwarePackageStoringEnum.⌋

Class	SoftwarePackageStoring			
Note	This meta-class provides the ability to specify whether and where the referenced SoftwarePackage is stored. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARObject			
Aggregated by	SoftwarePackageStep.transfer			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
storing	SoftwarePackageStoringEnum	0..1	attr	This attribute clarifies whether and where the referenced SoftwarePackage is stored.
transfer	SoftwarePackage	*	ref	This reference identifies the SoftwarePackage(s) to be transferred in the enclosing SoftwarePackageStep. **Tags:** atp.Splitkey=transfer

Table 16.26: SoftwarePackageStoring

[constr_10256] Multiplicity of reference in the role SoftwarePackageStoring.storing

Imposition time: IT_Mani

⌈For each SoftwarePackageStoring, the reference in the role storing shall exist.⌋

Enumeration	SoftwarePackageStoringEnum
Note	This enumeration controls where software packages are stored. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	SoftwarePackageStoring.storing
Literal	**Description**
none	No storing in vehicle. **Tags:** atp.EnumerationLiteralIndex=0
ucm	Storing in UCM (subordinate). **Tags:** atp.EnumerationLiteralIndex=2
ucmMaster	Storing in Ucm Master. **Tags:** atp.EnumerationLiteralIndex=1

Table 16.27: SoftwarePackageStoringEnum

[TPS_MANI_01302] Semantics of reference SoftwarePackageStep.process ⌈The reference SoftwarePackageStep.process identifies the SoftwarePackage that is supposed to be processed in the context of the enclosing SoftwarePackageStep.

The processing of SoftwarePackages happens strictly one after the other and therefore the reference process can only have the multiplicity 0..1. The strict order of

processing is guaranteed by the aggregation of the SoftwarePackageStep at UcmStep.⌋

[TPS_MANI_01306] Simultaneous existence of references in the role SoftwarePackageStoring.transfer and SoftwarePackageStep.process ⌈It is possible that the references SoftwarePackageStoring.transfer and SoftwarePackageStep.process simultaneously exist to the identical SoftwarePackage in the context of the same SoftwarePackageStep.

The semantics of such a configuration is that the SoftwarePackage that is referenced by the two roles owned by the same SoftwarePackageStep is “streamed”, i.e. transferred and processed in one step represented by the SoftwarePackageStep.⌋

Examples of the usage of SoftwarePackageStep are given in Chapter B.11.2.

17 Interoperability between Classic Platform and Adaptive Platform

This chapter collects restrictions for interoperability between Classic Platform and Adaptive Platform.

17.1 Usage of major Version in the SOME/IP network binding

In the case of a SOME/IP communication between Classic Platform and Adaptive Platform, the usage of SomeipServiceInterfaceDeployment.serviceInterfaceVersion.majorVersion is restricted.

If several ProvidedSomeipServiceInstances are defined with the same SomeipServiceInterfaceDeployment.serviceInterfaceId and different SomeipServiceInterfaceDeployment.serviceInterfaceVersion.majorVersions and these ProvidedSomeipServiceInstances are mapped to the same Socket Address (UDP/TCP Port and IP Address) by SomeipServiceInstanceToMachineMapping then particular restrictions apply if the ProvidedSomeipServiceInstances **are consumed on a Classic Platform ECU**.

In such a scenario the same MessageId may be used for ServiceInterface elements like an event in the different ProvidedSomeipServiceInstances that have different majorVersions.

On the Classic Platform in the SoAd module, this will result in different Pdus that have the same headerId in the same SocketConnection. In the AUTOSAR Architecture of the Classic Platform one part of the SOME/IP Header is evaluated in the SOME/IP Transformer (RequestId, Protocol Version, Interface Version, Message Type, Return Code) and the other part in the Socket Adaptor (MessageId, Length).

This means that the Socket Adaptor is not able to evaluate the MajorVersion in the Pdu and can not determine the source of the Pdu.

The following restrictions apply if the ServiceInterface contains methods and/or fields with hasGetter or hasSetter set to true: If two or more ProvidedSomeipServiceInstances are defined using the same serviceInterfaceId and different majorVersions and these ProvidedSomeipServiceInstances are mapped to the same Socket Address (same UDP/TCP Port and IP Address) then the destination IP address, the destination port number, and the Level 4 protocol (Udp/Tcp) fields of header of IP packets containing calls that are sent to the ProvidedSomeipServiceInstances are identical.

In such a scenario, the ProvidedSomeipServiceInstances may still use identical methodIds if the following condition applies:

(a) At any point in time only one of the ProvidedSomeipServiceInstances is active, and only clients of that ProvidedSomeipServiceInstance send requests to the ProvidedSomeipServiceInstance.

In all other cases, the methodIds of the two ProvidedSomeipServiceInstances should not overlap.

The following restrictions apply for ServiceInterface events and fields with hasNotifier set to true: If two or more ProvidedSomeipServiceInstances are defined using the same serviceInterfaceId and different majorVersions and these ProvidedSomeipServiceInstances are mapped to the same Socket Address (same UDP/TCP Port and IP Address), then the source IP address, the source port number, and the Level 4 protocol (Udp) fields of header of IP packets containing event PDUs that are sent to the clients of the ProvidedSomeipServiceInstances are identical.

In such a scenario, the ProvidedSomeipServiceInstances may use identical eventIds if at least one of the following conditions holds for any pair of the ProvidedSomeipServiceInstances:

(a) At any point in time only one of the ProvidedSomeipServiceInstances is active

(b) If two or more ProvidedSomeipServiceInstances can send events or Field notifiers at the same time, the ProvidedSomeipServiceInstances may still use identical eventIds if at least one of the following IP header fields of the IP packet containing the event is different for any pair of SomeipEventDeployments identified by the same eventId:

(b1) Destination IP address (== IP address of client)

(b2) Destination port number (== client port number)

A Reference Material

A.1 Terms and Abbreviations

The content of this appendix chapter is *informative* in nature and shall **not** be considered as *normative* content.

The main list of terms and abbreviations are defined in [47, TR Glossary]. The following table contains the list of terms and abbreviations used in the scope of this document which are not already defined in [47, TR Glossary] 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
DDS	Data Distribution Service
DES	Data Encryption Standard
DHCP	Dynamic Host Configuration Protocol
DM	Diagnostic Manager
DNS	Domain Name System
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
GCM-SIV	Galios/Counter Mode with Synthetic Initialization Vector
HMAC	Hash-based Message Authentication Code
HTTP	Hypertext Transport Protocol
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
ICV	MACsec Integrity Check Value
ID	Identifier
IP	Internet Protocol
ISO	International Standardization Organization
JSON	JavaScript Object Notation
KaY	MACsec Key Agreement Entity
LAN	Local Area Network

▽

△

Abbreviation	*Meaning*
MAC	Media Access Control
MAC	Message Authentication Code
MACsec	Media Access Control security
MD	Message Digest
MTU	Maximum Transmission Unit
mDNS	Multicast DNS
NDP	Neighbor Discovery Protocol
NM	Network Management
NV	Non-Volatile
PAE	MACsec Port Access Entity
PHM	Platform Health Management
PKCS	Public Key Cryptography Standards
POSIX	Portable Operating System Interface
PQC	Post-Quantum Cryptography
PSK	Pre-Shared Key
ROM	Read-Only Memory
RSA	Cryptographic approach according to Rivest, Shamir, and Adleman
SAK	MACsec Secure Association key
SD	Service Discovery
SDG	Special Data Group
SecY	MACsec Security Entity
SHA	Secure Hash Algorithm
SOME/IP	Scalable service-Oriented MiddlewarE over IP
SOVD	Service-Oriented Vehicle Diagnostics
SWC	Software Component
TLS	Transport Layer Security
TLV	Tag Length Value
TLS	Transport Layer Security
TTL	Time to Live
UDP	User Datagram Protocol
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 A.1: Terms and Abbreviations used in the scope of this Document

A.2 Imposition Times of Constraints

The constraints formulated in this document have different *actual* imposition times which denote the steps in the workflow when the respective constraint has to be imposed.

The imposition times that are considered applicable in the scope of this document[1] are listed in Table A.2.

Please note that the imposition times are intentionally rendered as technical terms such that it is possible to link back from each constraint to the definition of the affected imposition time in Table A.2.

This document has been created to apply primarily for the *AUTOSAR adaptive platform* and therefore the discussed imposition times also apply exclusively to the *AUTOSAR adaptive platform*.

Imposition Time	Description	Motivation
IT_BefAraApiGen	Before the generation of the ara API starts	This imposition time is aimed at the time when a software-component is ready for generating the header files such that the implementation of the software-component can be started.
IT_DiagDes	Diagnostic design is completed	This imposition time denotes the step in the workflow, where the diagnostic design is about to be finished independent of the standard where it is applied. This includes the finalization, e.g., of the DEXT.
IT_GraDes	GrantDesign is completed	This imposition time is aimed at the time when a `GrantDesign` is finished, i.e. it shall be used in constraints that target the consistency of the modeling of `GrantDesign`.
IT_MachDes	Machine design is completed	This imposition time denotes the step in the workflow, where the machine design is about to be finished.
IT_Mani	Creation of the manifest is finished	This imposition time denotes the step in the workflow, where the manifest is considered complete such that the installation on a target platform can be started.
IT_ProDes	ProcessDesign is completed	This imposition time is aimed at the time when a `ProcessDesign` is finished, i.e. it shall be used in constraints that target the consistency of the modeling of `ProcessDesign`.
IT_SovdSysDes	SOVD System Design is complete	This imposition time denotes the step in the workflow, where the SOVD system design is about to be finished.
IT_SubSysDes	Sub-system design is completed	This imposition time denotes the step in the workflow, where the sub-system design is about to be finished.
IT_SysDes	System design is completed	This imposition time denotes the step in the workflow, where the system design is about to be finished.

Table A.2: Imposition Times of constraints in this document

[1] Different imposition times may be defined in the context of other AUTOSAR standard documents

B Examples

The content of this appendix chapter is *informative* in nature and shall **not** be considered as *normative* content.

This chapter contains a collection of examples that reflect concepts described in different chapters of this document. The content of the chapter provides mere explanation and does not add anything to the model semantics.

B.1 Data Type

B.1.1 Definition of Map Data Type

Figure B.1 contains a graphical representation of an example model for `ApplicationAssocMapDataType`.

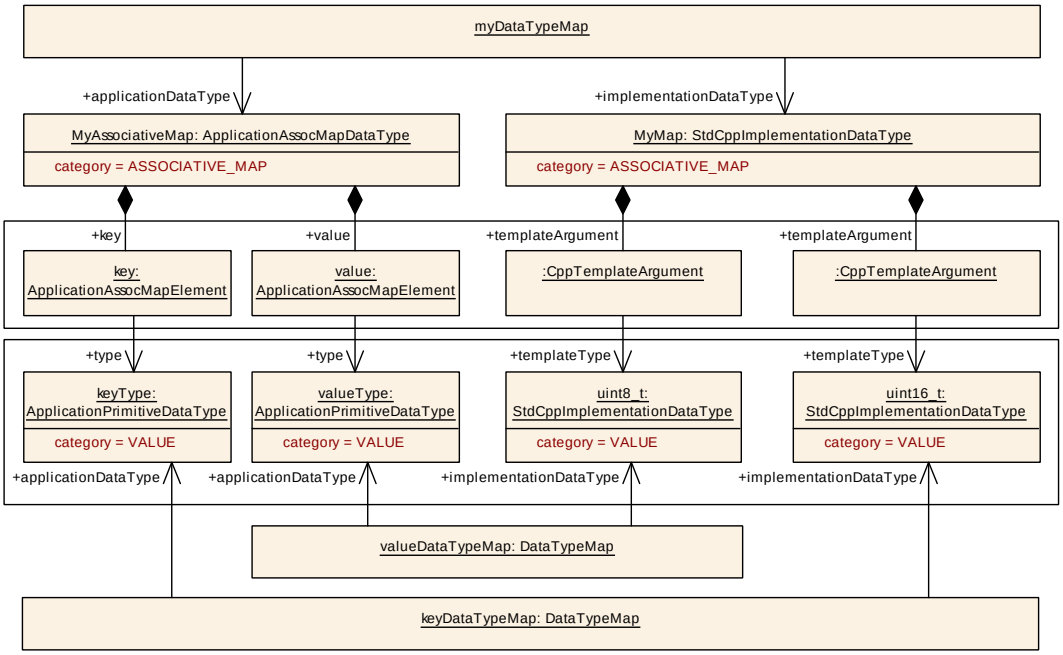


Figure B.1: Example of the model of an associative map

Listing B.1 provides the corresponding ARXML serialization of the example model of an `ApplicationAssocMapDataType` depicted in Figure B.1.

```
<APPLICATION-ASSOC-MAP-DATA-TYPE>
  <SHORT-NAME>MyAssociativeMap</SHORT-NAME>
  <KEY>
    <SHORT-NAME>MyKey</SHORT-NAME>
    <TYPE-TREF BASE="default" DEST="APPLICATION-PRIMITIVE-DATA-TYPE">keyType</TYPE-TREF>
  </KEY>
  <VALUE>
    <SHORT-NAME>MyValue</SHORT-NAME>
    <TYPE-TREF BASE="default" 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>
```

Listing B.1: Example for the definition of an `ApplicationAssocMapDataType`

B.1.2 Definition of String Data Type

The example depicted in Figure B.2 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 `category STRING` shall only be implemented on the basis of an `ara::core::String`.

Another aspect of the example in Figure B.2 is that it defines the intended encoding of the modeled data type in the scope of the `ApplicationPrimitiveDataType`.

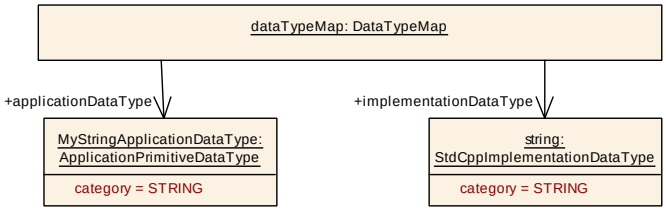


Figure B.2: Example of the model of a string with `UTF-8` encoding

B.1.3 Definition of String Data Type

Figure B.3 shows an example of an one-dimensional array of uint16 elements with `arraySize` = 5.

Figure B.4 shows an example of a multidimensional array where a `CppImplementationDataType` of `category ARRAY` with `arraySize` = 5 has a `templateArgument` that points to the inner `CppImplementationDataType` of `category ARRAY` in the role `templateType`.

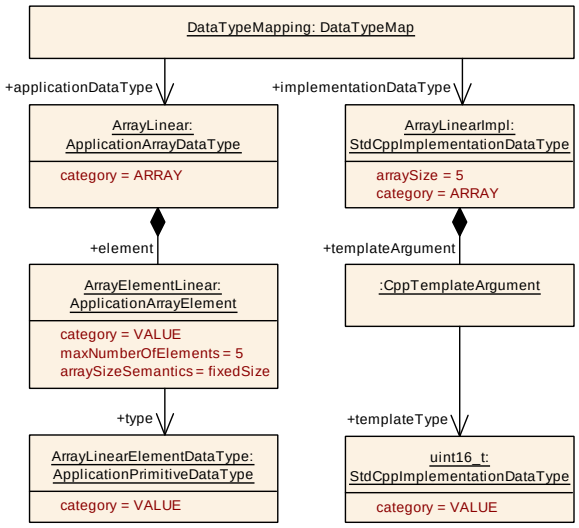


Figure B.3: Example of the model of a one-dimensional array

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

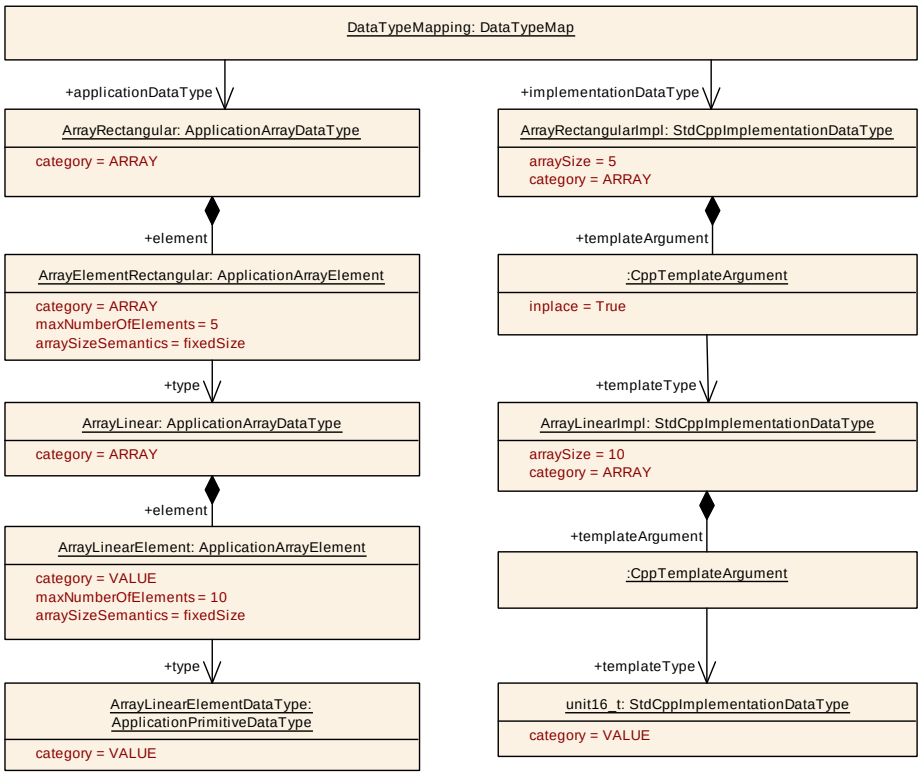


Figure B.4: Example of the model of a multidimensional array

B.1.4 Definition of Vector Data Type

Figure B.5 shows an example of an one-dimensional vector of uint16 elements.

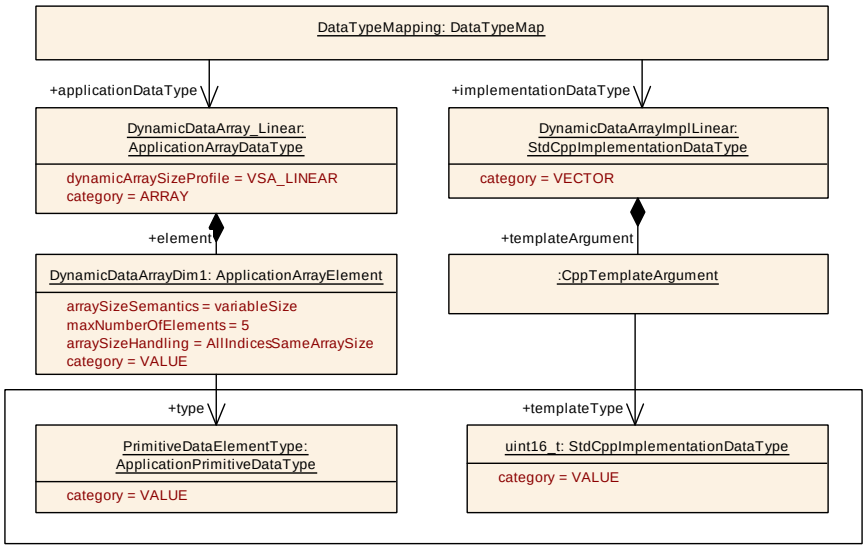


Figure B.5: Example of the model of a one-dimensional vector

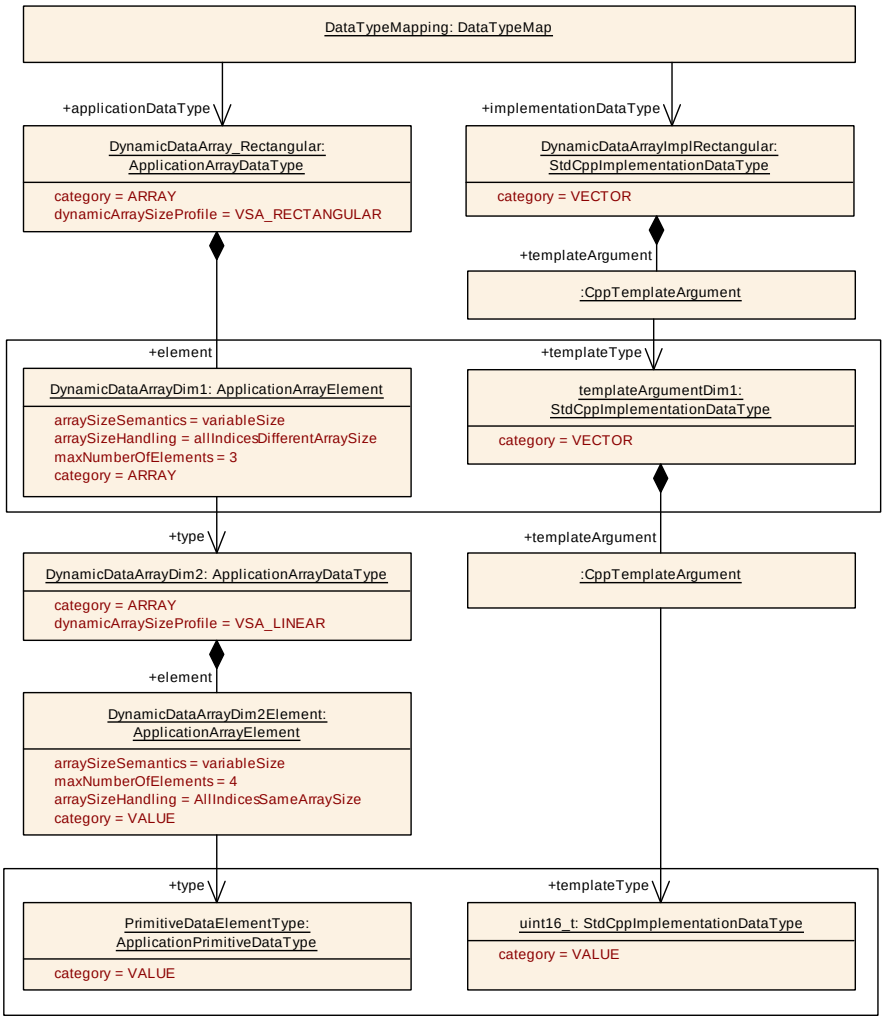


Figure B.6: Example of the model of a multidimensional vector

Figure B.6 shows an example of a multidimensional vector where a CppImplementationDataType of category VECTOR 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.

B.1.5 Definition of Struct Data Type

The example depicted in Figure B.7 shows the definition of a Structure, called MyStruct, that has two members. The typeReference of the subElements with the shortName ArrayElement is classified with inplace = **true**.

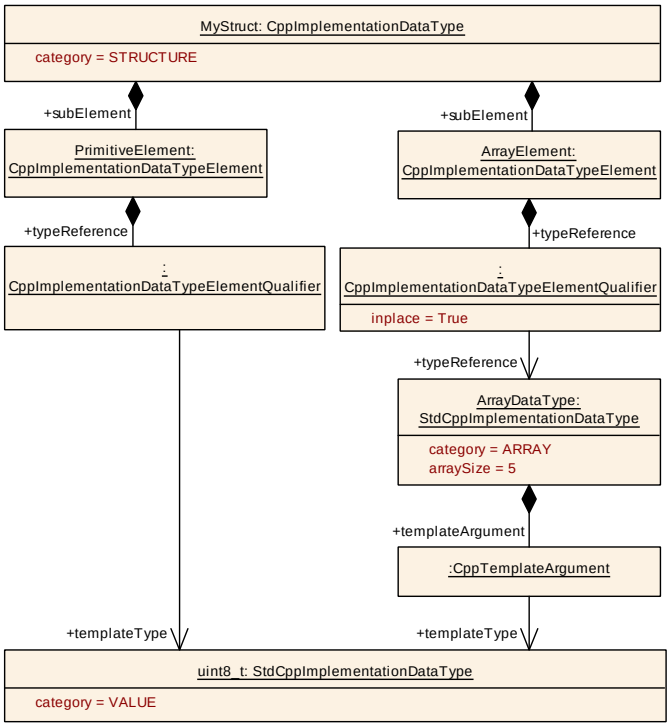


Figure B.7: Example of the model of a Struct

If 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.

B.1.6 Definition of Enumeration Data Type

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

```
<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>
```

Listing B.2: example for enumeration

B.1.7 Definition of Map Data Type

The example depicted in Figure B.8 shows the definition of a `ASSOCIATIVE_MAP` that has two `CppTemplateArgument`s, one for the key and one for the value.

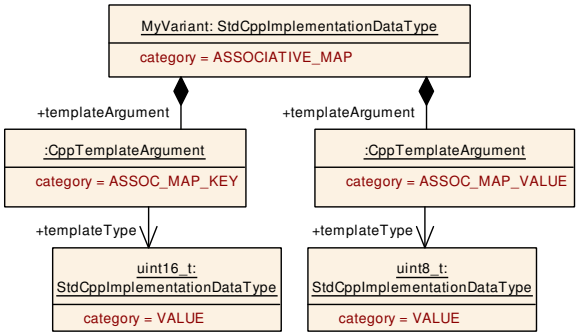


Figure B.8: Example of the model of an `ASSOCIATIVE_MAP`

B.1.8 Definition of Variant Data Type

The example depicted in Figure B.9 shows the definition of a `VARIANT` that has two `CppTemplateArgument`s. Each one represents one alternative type. Please note that the `CppTemplateArgument`s of a `CppImplementationDataType` are ordered in ARXML and this order is not visible in the object diagram.

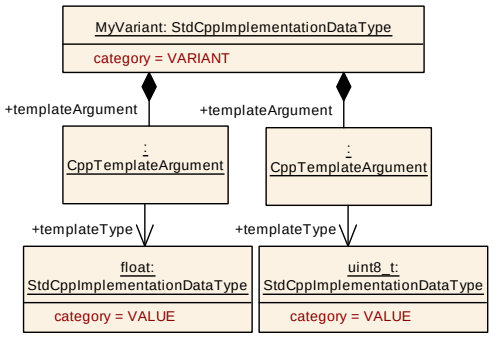


Figure B.9: Example of the model of an VARIANT

B.1.9 Scenarios to define a Vector

This section contains a non-comprehensive list of possible scenarios for the definition of a CppImplementationDataType of category VECTOR.

Please note that the general information contained in this chapter does not exclusively apply on to the vector data type. The latter has been picked as an arbitrary example for the visualization of the effect of configuration settings on the language binding.

Consequently, there is no further discussion of this topic with respect to a different kind of container data type.

Please note that for these example scenarios the namespace of a CustomCpp-ImplementationDataType is assumed to be set to x::y and the shortName is assumed to be set to CustVec.

The shortName of a StdCppImplementationDataType is assumed to be set to MyVec.

If a custom Allocator is used in a scenario the value of Allocator.shortName shall be assumed to have the value CustAlloc.

Scenario	array size	custom allocator	custom type	Resulting C++ Code
I	No	No	No	`using MyVec = ara::core::Vector<std::uint8_t>`
II	Yes	No	No	`using MyVec = ara::core::Vector<std::uint8_t> //generator warning`
III	Yes	Yes	No	`using MyVec = ara::core::Vector<std::uint8_t, CustAlloc<std::uint8_t, MaxSize>>`
IV	No	Yes	No	`using MyVec = ara::core::Vector<std::uint8_t,CustAlloc<std::uint8_t>>`
V	Yes	Yes	Yes	`x::y::CustVec<ara::core::uint8_t, CustAlloc<std::uint8_t, MaxSize>>`
VI	Yes	No	Yes	`x::y::CustVec<std::uint8_t>`

▽

△

Scenario	array size	custom allocator	custom type	Resulting C++ Code
VII	No	Yes	Yes	`x::y::CustVec<std::uint8_t, CustAlloc<std::uint8_t>>`
VIII	No	No	Yes	`x::y::CustVec<std::uint8_t>`

Table B.1: Example definitions of a `CppImplementationDataType` of `category VECTOR`

B.2 Service-oriented Communication

B.2.1 Radar and Camera ServiceInterface example

The example in Figure B.10 shows a *Radar* `ServiceInterface` with a *BrakeEvent* and two `method`s: *Calibrate* and *Adjust*. The *Camera* `ServiceInterface` shown in Figure B.11 has two events: *LaneEvent* and *SpeedLimitEvent* and one *Calibrate* `method`.

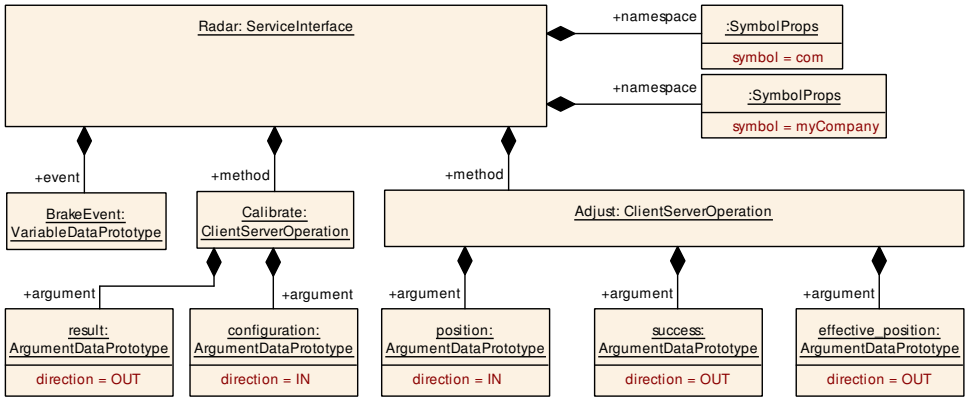


Figure B.10: Radar Service Interface

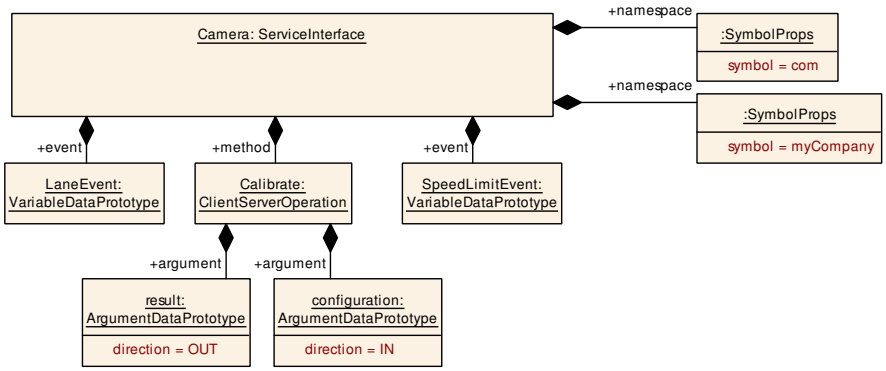


Figure B.11: Camera Service Interface

Both ServiceInterfaces *Radar* and *Camera* are mapped to a combined *RadarAndCamera* ServiceInterface with an Service Interface Element Mapping since both ServiceInterfaces have a method with the same name: *Calibrate*.

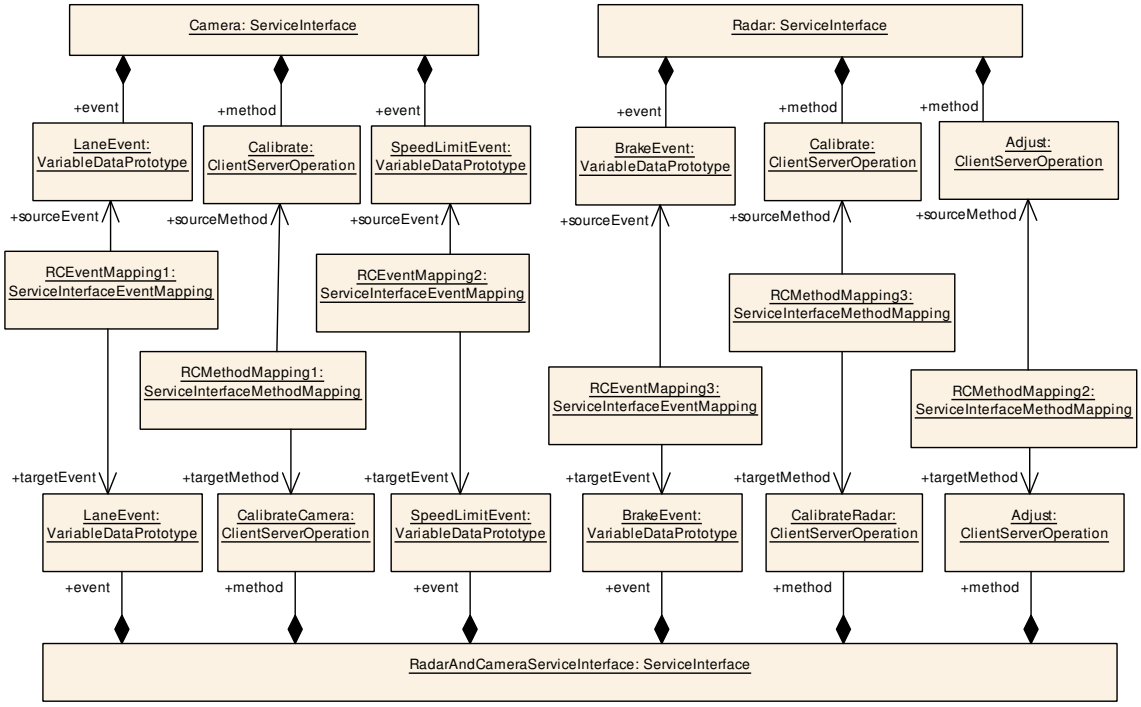


Figure B.12: Service Interface Element Mapping example

The combined ServiceInterface is offered over the network as a SOME/IP Service. Figure B.13 shows the assignment of the SOME/IP serviceInterfaceId to 31.

In addition SOME/IP eventIds are assigned to the events and methodIds are assigned to the methods. Furthermore a single SomeipEventGroup is defined to which all SomeipEventDeployments of the *RadarAndCamera* ServiceInterface are assigned.

Figure B.14 shows a modeled ProvidedSomeipServiceInstance that is mapped to a Machine.

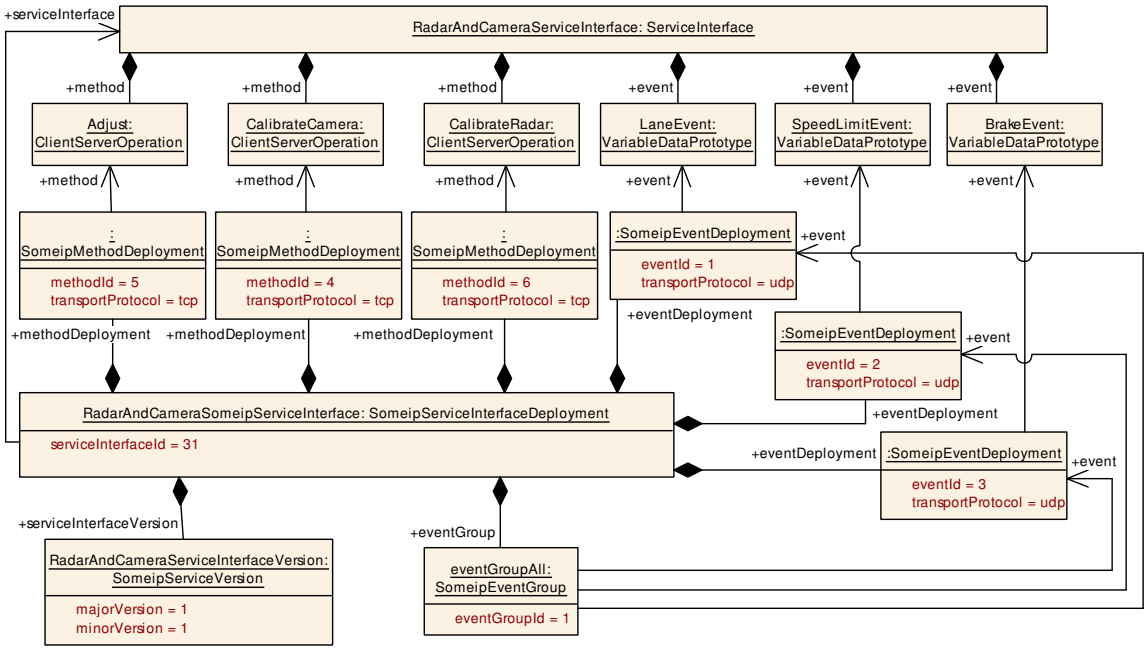


Figure B.13: SOME/IP Deployment

The displayed configuration in Figure B.14 leads to a SOME/IP OfferService Message with the following content:

- ServiceId => `serviceInterfaceId` = 31
- InstanceId => `serviceInstanceId` = 1
- MajorVersion => 1
- MinorVersion => 1
- TTL => 3
- IPv4 Endpoint Option with IPv4 Address (170.88.199.94), Protocol (TCP), Port-Number (30567)
- IPv4 Endpoint Option with IPv4 Address (170.88.199.94), Protocol (UDP), Port-Number (30501)
- IP Multicast Endpoint Option with IPv4 Address (239.255.0.1), Protocol (UDP), PortNumber (30502)

An example of a `RequiredSomeipServiceInstance` is shown in Figure B.15.

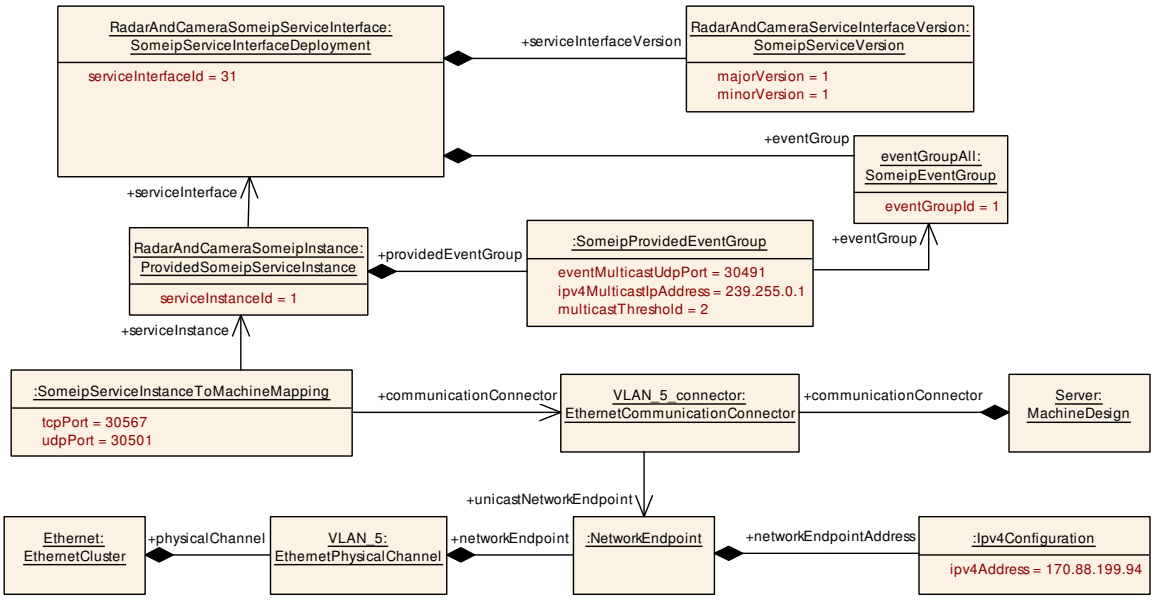


Figure B.14: SOME/IP Provided Service Instance

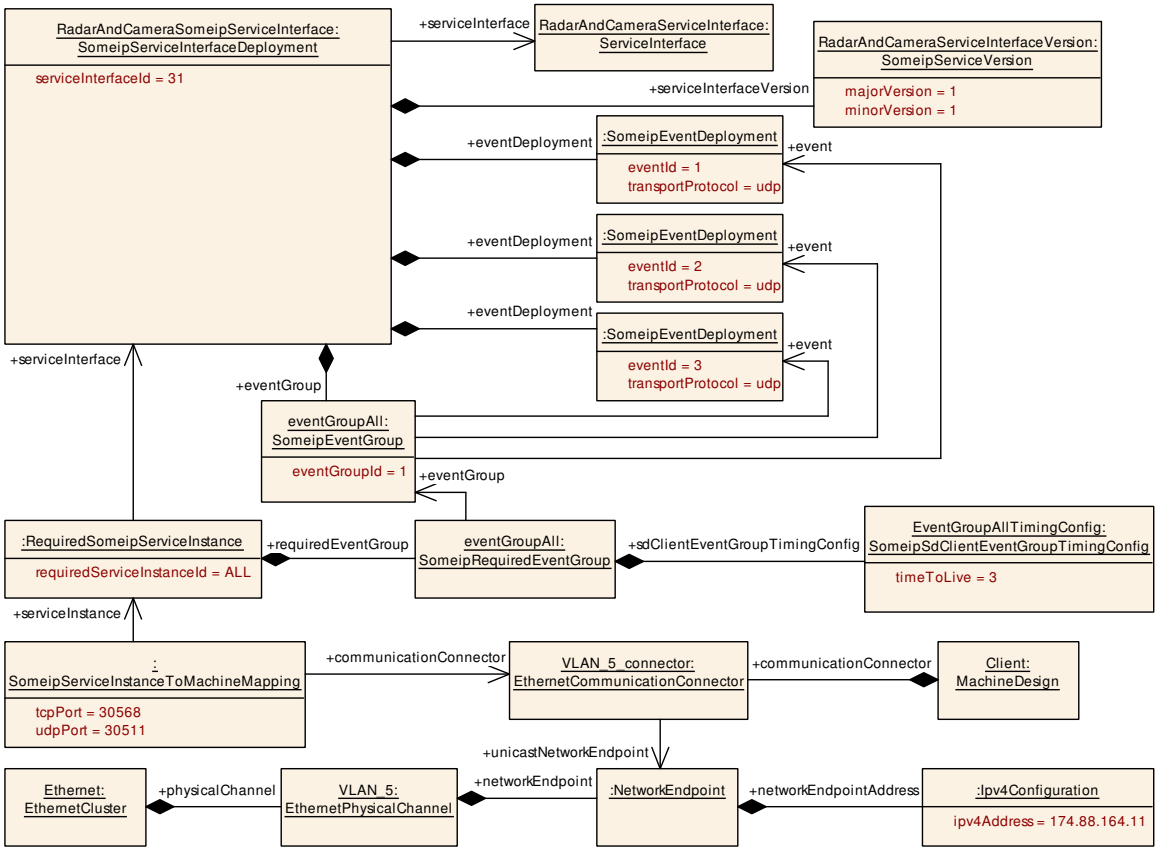


Figure B.15: SOME/IP Required Service Instance

The displayed configuration in Figure B.15 leads to a SOME/IP Find Service Message with the following content:

- ServiceId => `serviceInterfaceId` = 31

- InstanceId => RequiredSomeipServiceInstance.requiredServiceInstanceId = ALL
- MajorVersion => majorVersion = 1
- MinorVersion => minorVersion = 1

The displayed configuration in Figure B.14 also leads to a SOME/IP SubscribeEventGroup Message content that is sent from the Service Requester to the Service Provider:

- ServiceId => taken from the OfferMessage
- InstanceId => taken from the OfferMessage
- MajorVersion => taken from the OfferMessage
- MinorVersion => taken from the OfferMessage
- Eventgroup ID => RequiredSomeipServiceInstance.requiredEventGroup.eventGroup.eventGroupId = 1
- TTL => RequiredSomeipServiceInstance.requiredEventGroup.sdClientEventGroupTimingConfig.timeToLive = 3
- IPv4 Endpoint Option with IPv4 Address (170.88.164.11), Protocol (UDP), PortNumber (30511)

B.2.2 Service Interface Mapping

B.2.2.1 Service Interface Element Mapping

Figure B.16 provides an example of how a mapping on element basis looks like. Note that, in this example, both ServiceInterface1 and ServiceInterface2 aggregate a field with the shortName field1.

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 shortName field1 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 Chapter 4.4.

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

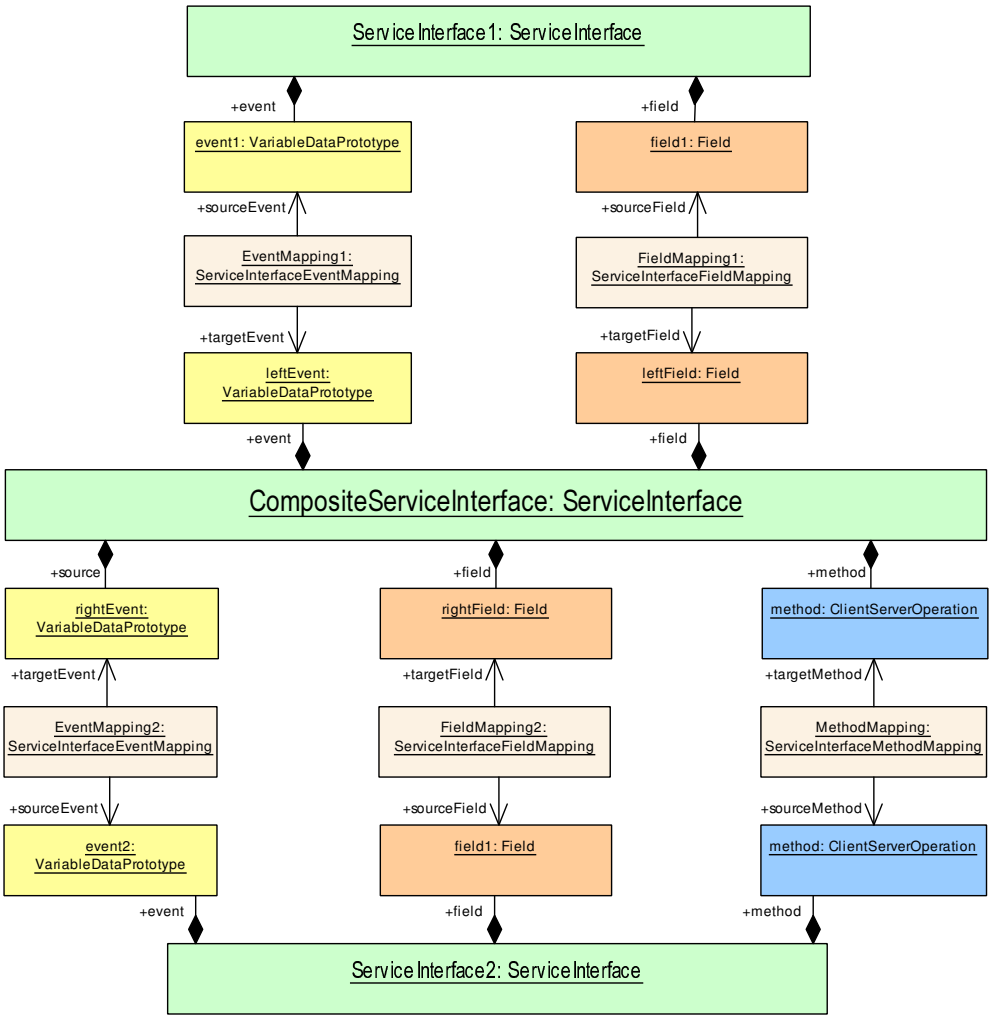


Figure B.16: Example for a mapping of elements of `ServiceInterface`

In other words, as soon as a mapping between two `ServiceInterface`s exists, it is not supported that a mapping between elements of the same pair of `ServiceInterface`s exists. This important restriction is formalized by [constr_1482].

B.2.2.2 Service Interface Event Mapping

It is 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 B.17.

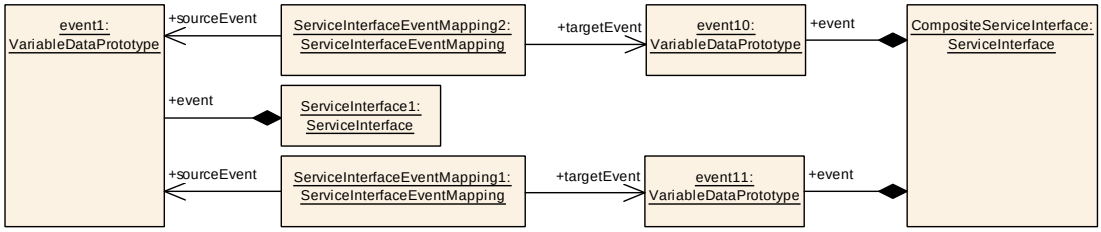


Figure B.17: Example for the application of a `ServiceInterfaceEventMapping`

B.2.2.3 Service Interface Field Mapping

It is 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 B.18.

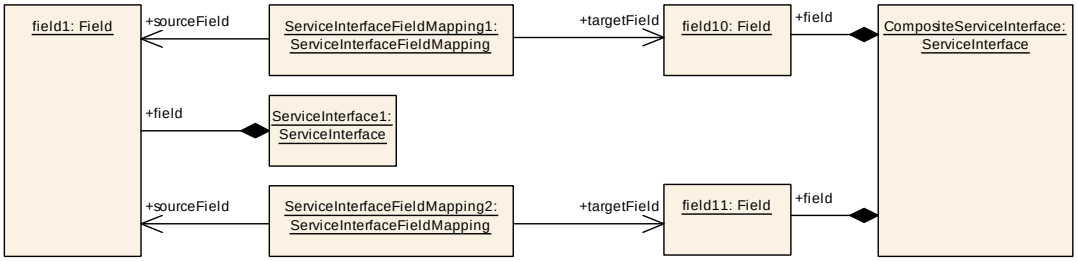


Figure B.18: Example for the application of a `ServiceInterfaceFieldMapping`

B.2.2.4 Service Interface Method Mapping

It is 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 B.19.

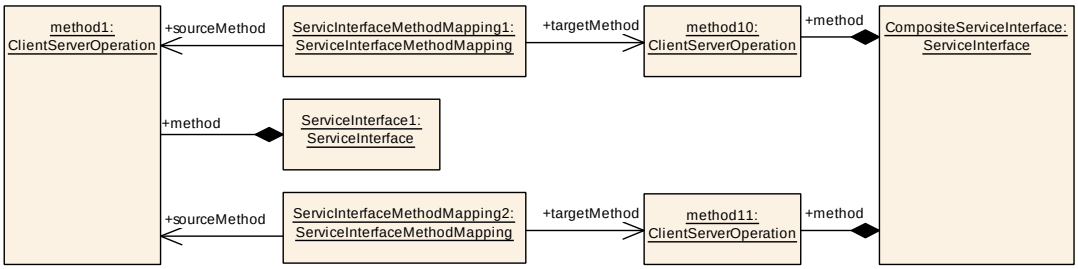


Figure B.19: Example for the application of a `ServiceInterfaceMethodMapping`

B.2.3 Service Instance Deployment

B.2.3.1 Service Instance Deployment by Service Interface Mapping

The example in Figure B.21 sketches the modeling of a `ProvidedSomeipService-Instance` in the presence of a `ServiceInterfaceMapping`, that references two `ServiceInterface`s in the role `sourceServiceInterface`.

For support, Figure B.20 contains an excerpt from the meta-model that contains the relevant meta-classes that have been instantiated to create the example sketched in Figure B.21.

Note further that the example depicted in Figure B.21 is not limited to the explanation of the actual `ServiceInterfaceMapping`.

As the main use case for this is the usage of ServiceInterfaces for the definition of an “outside” communication binding the example also contains the modeling of such a binding, in this case to SOME/IP.

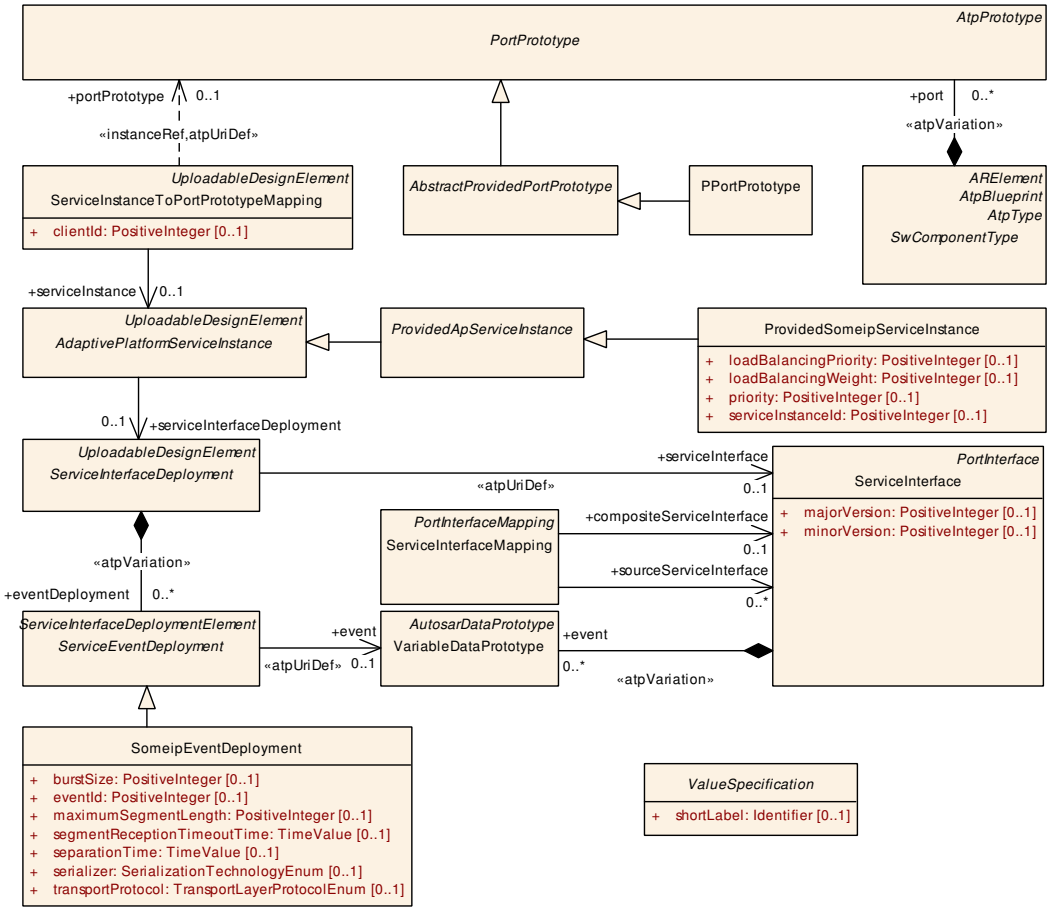


Figure B.20: Meta-model excerpt relevant for the example

Please note that the modeling of the binding requires the existence of a PortPrototype, which in turn is aggregated by an SwComponentType (not depicted).

This approach still contains some degrees of freedom with respect to the role of the SwComponentType that aggregates the mentioned PortPrototype. This document does not go further in discussing the nature of such a configuration.

For reasons of keeping the example as simple as possible, each of the ServiceInterfaces in the role sourceServiceInterface aggregate a single event.

The ServiceInterface referenced in the role compositeServiceInterface aggregates two event with shortNames that match the mentioned event of the source ServiceInterfaces (see [TPS_MANI_01022]).

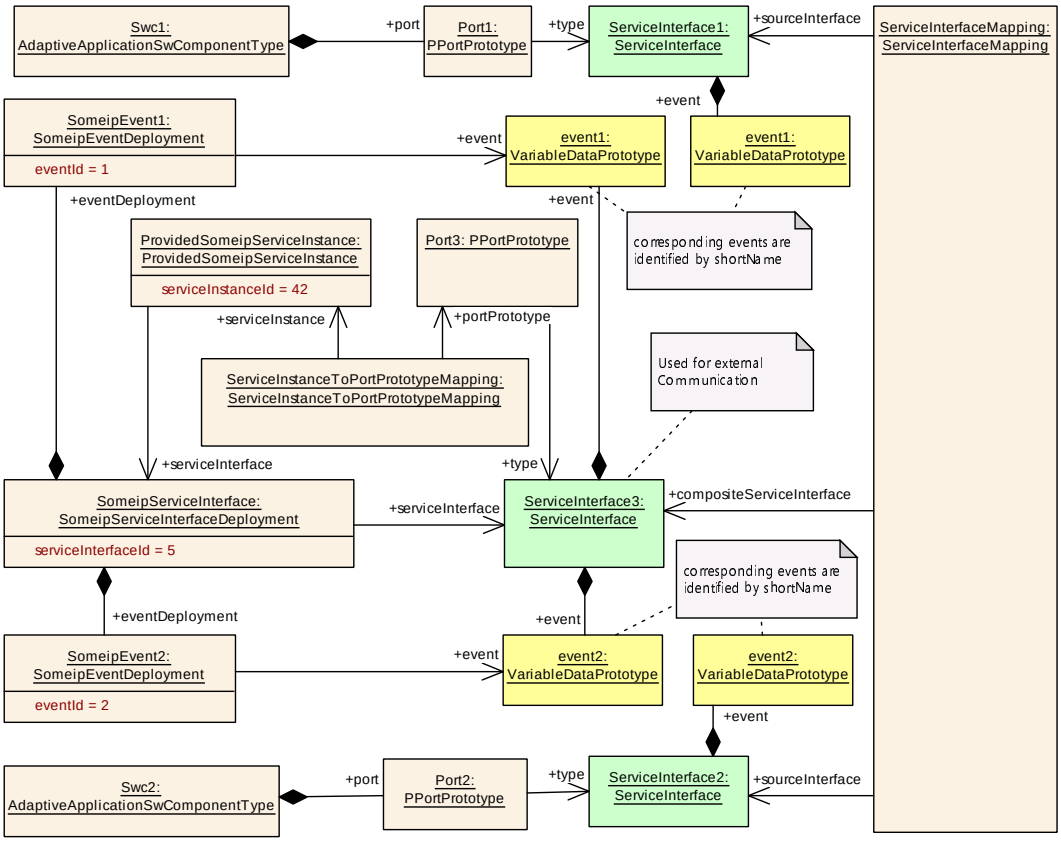


Figure B.21: Example for the deployments of a service in the presence of a `ServiceInterfaceMapping`

B.2.3.2 Service Instance Deployment by Service Interface Element Mapping

The example in Figure B.23 sketches the modeling of a `ProvidedSomeipServiceInstance` in the presence of a `ServiceInterfaceEventMapping`s. In principle, this example is very close to the example described in Figure B.21.

In contrast to the example sketched in Figure B.21, the example depicted in Figure B.23 uses a mapping to individual elements of a `ServiceInterface` instead of the entire `ServiceInterface`.

Please find the corresponding excerpt of relevant meta-classes for the utilization of `ServiceInterfaceEventMapping` sketched in Figure B.22.

Note further that the example depicted in Figure B.22 is not limited to the explanation of the actual `ServiceInterfaceElementMapping`.

As the main use case for this is the usage of `ServiceInterface`s for the definition of an “outside” communication binding the example also contains the modeling of such a binding, in this case to SOME/IP.

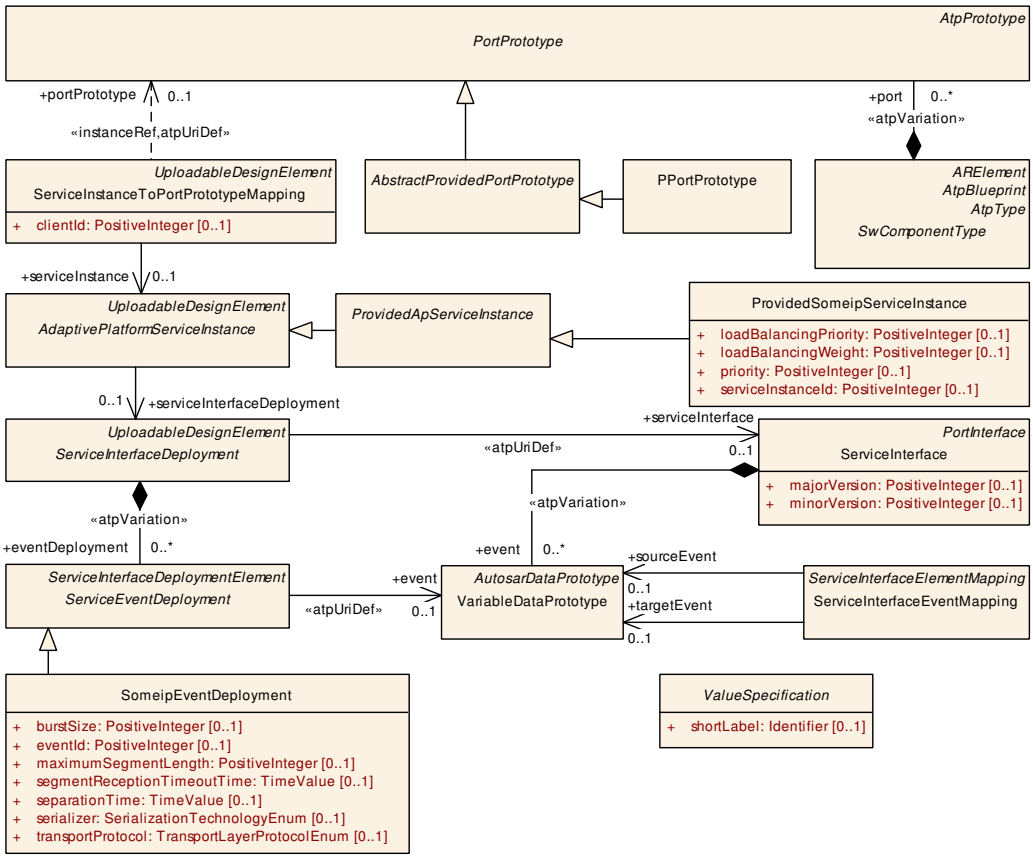


Figure B.22: Excerpt of the relevant meta-classes for the `ServiceInterfaceEventMapping` example

Please note that the modeling of the binding requires the existence of a `PortPrototype`, which in turn is aggregated by an `SwComponentType` (not depicted).

This approach still contains some degrees of freedom with respect to the role of the `SwComponentType` that aggregates the mentioned `PortPrototype`. This document does not go further in discussing the nature of such a configuration.

By mapping individual elements of `ServiceInterface`s, it is possible to map element with different `shortName`s to each other. In this example, the `event` with the `shortName` `event1` is mapped to another `event` with the `shortName` `eventLeft`.

In Figure B.23, two different `ServiceInterface`s exist that each aggregate an `event` with the identical `shortName`. This scenario **requires** the existence of `ServiceInterfaceElementMapping`s.

As an extension to the scenario depicted in Figure B.23, Figure B.24 describes a model where the **same** `event` of a `ServiceInterface` is used in two different event deployments by means of two `ServiceInterfaceEventMapping`s that each refer to said `event` in the role `ServiceInterfaceEventMapping.sourceEvent`.

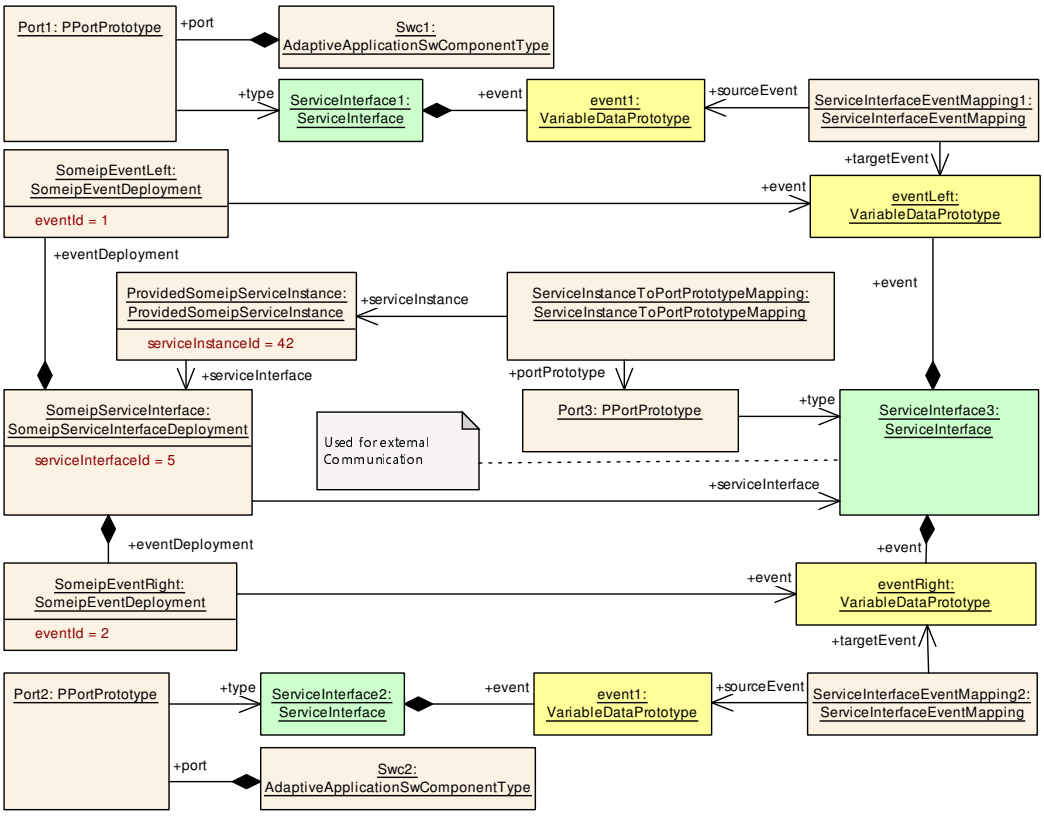


Figure B.23: Example for the deployment of a service in the presence of a `ServiceInterfaceEventMapping`

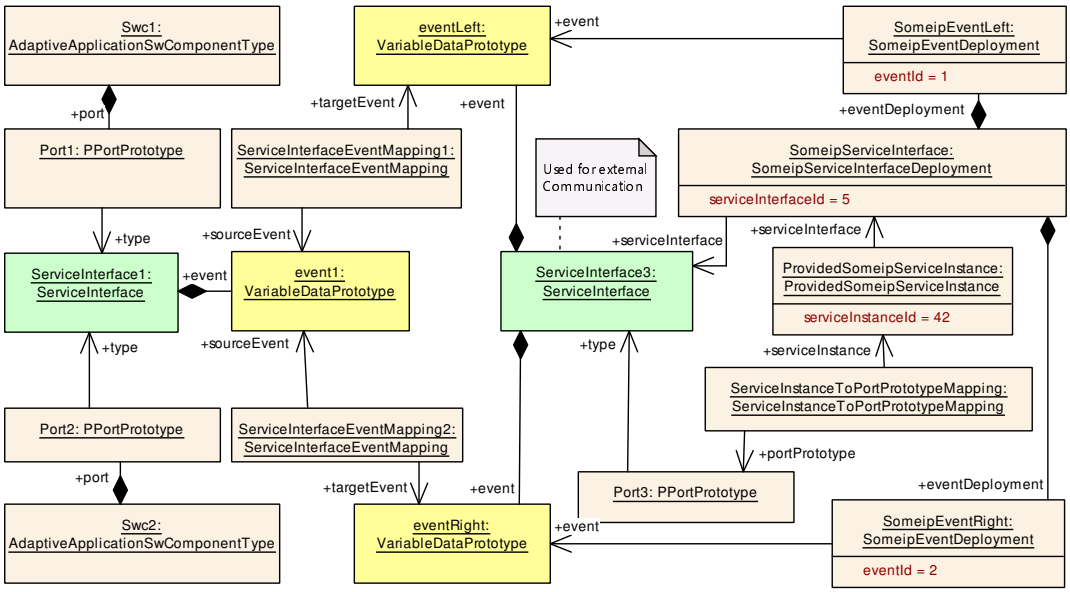


Figure B.24: Example for the deployment of a service in the presence of a `ServiceInterfaceEventMapping` to the same source `ServiceInterface`

Again, this scenario **requires** the existence of appropriately configured `ServiceInterfaceElementMapping`s.

B.2.4 Service Instance Mapping

B.2.4.1 SOME/IP Service Instance Mapping

This section contains some examples that explain the modeling of a mapping between a service instance and the application. The examples have been created to show both the “find” and the “offer” side of the service binding.

In the first example, depicted in Figure B.25 shows the binding of `PortPrototype`s to a SOME/IP-based transport layer. The left part of the diagram contains the modeling of the “find” aspect and the right part contains the modeling of the “offer” aspect.

Please note that the `shortName`s of the two affected `PortPrototype`s are different. In other words, the `shortName`s of the `PortPrototype`s are not used as a way to identify the opposite end of the service binding.

Instead, the existence of a `ServiceInstanceToPortPrototypeMapping` that maps a `PortPrototype` to a `ProvidedSomeipServiceInstance` or `RequiredSomeipServiceInstance` with the **identical value** of attribute `serviceInstanceId` creates the actual binding between the “find” and the “offer” end.

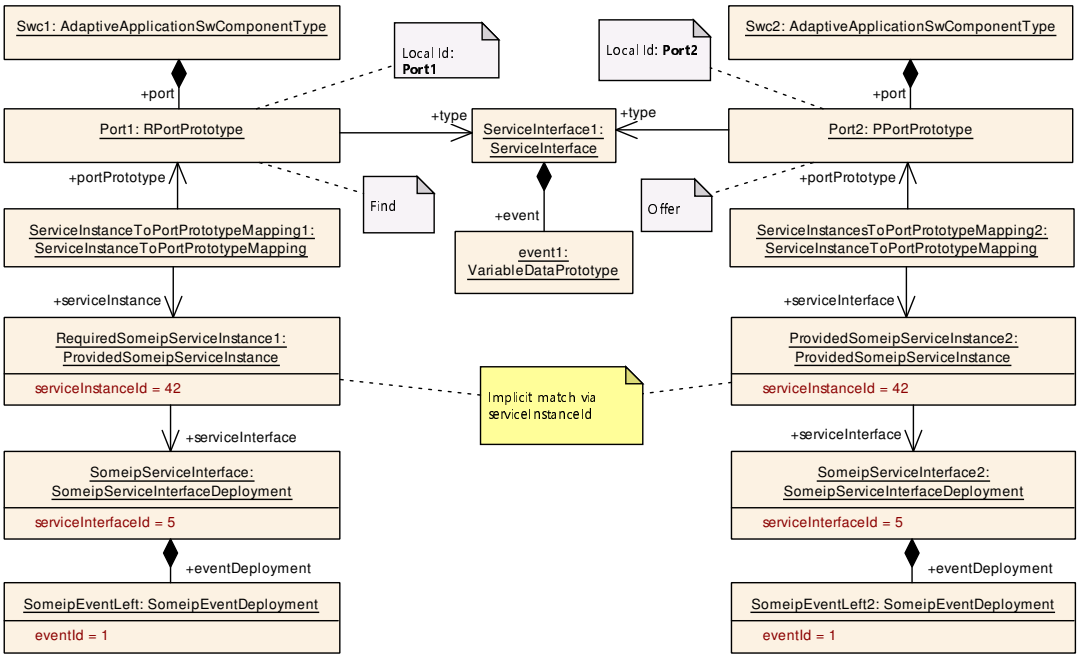


Figure B.25: Port-based binding of a service instance to the application using SOME/IP

B.2.4.2 User-Defined Service Instance Mapping

The next example (depicted in Figure B.26) shows a binding of `PortPrototype`s to a user-defined transport layer. The left part of the diagram contains the modeling of the “find” aspect and the right part contains the modeling of the “offer” aspect.

Because the binding is user-defined, there are no attributes modeled on the level of the meta-model available to identify an instance according to the user-defined service implementation. There is just no way to define attributes that are “needed anyway” for a user-defined binding.

Therefore, the only option in this case it the usage of `AdminData`, `Sdg`, and `Sd` to define an identification of the user-defined transport layer.

In order to support the comparison to the example depicted in Figure B.25, the example described in Figure B.26 uses a simple identification based on a numerical value. Again, this is an arbitrary scenario created just for the sake of explanation.

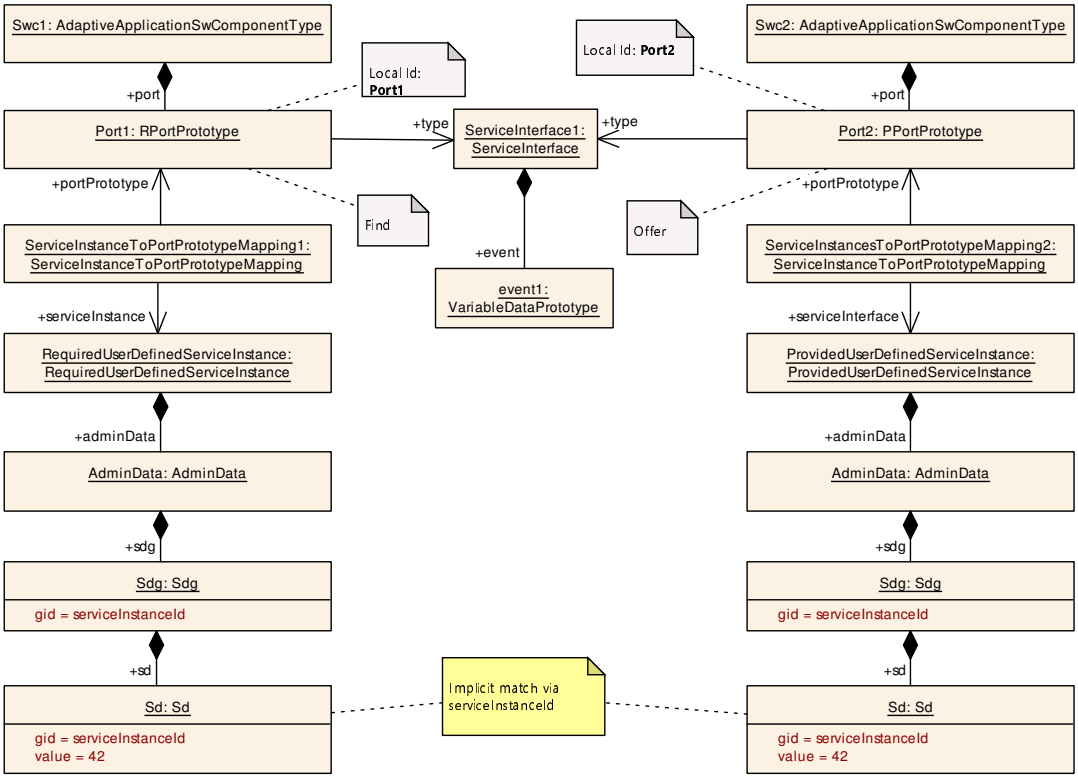


Figure B.26: Port-based binding of a service instance to the application using a user-defined binding

B.2.5 Definition of VLAN Priority

The following table shows an example for the usage of the VLAN priority (please find the background information in Chapter 13.1.1). A `CouplingPort` defines a `defaultPriority` equal to 0. In the first row, no other priorities are defined, so the effective priority is 0.

In the second row, a `priority` of 4 is defined on the `NetworkEndpoint`, so the effective priority is 4. In the third row, a `priority` of 2 is defined on the `ApApplicationEndpoint`, so the effective priority is 2. In the last row, a `priority`

of 1 is defined on the ProvidedSomeipServiceInstance, so the effective priority is 1.

Physical Port (defaultPriority of CouplingPort)	NetworkEndpoint.priority (e.g. IP Address)	ApApplicationEndpoint.priority (e.g. Tcp Port)	ProvidedSomeipServiceInstance.priority	Effective priority
0	-	-	-	**0**
0	**4**	-	-	**4**
0	4	**2**	-	**2**
0	4	2	**1**	**1**

Table B.2: VLAN Priority Example

B.2.6 Static Configuration of remote Peers

B.2.6.1 Provided Service Instance with remote Peers

The following example in Figure B.27 shows the setup of a Service Provider with two statically defined remote peers. The ServiceInstanceToMachineMapping maps two ProvidedSomeipServiceInstances to ApApplicationEndpoints: Camera and Radar_1.

Two remote peers are referenced by the same ServiceInstanceToMachineMapping in the role remoteUnicastConfig: RemotePeer1 and RemotePeer2.

RemotePeer1 is subscribed to EG2 and EG3 of ServiceInstance Radar_1. RemotePeer2 is subscribed to EG1 and EG2 of ServiceInstance Radar_1.

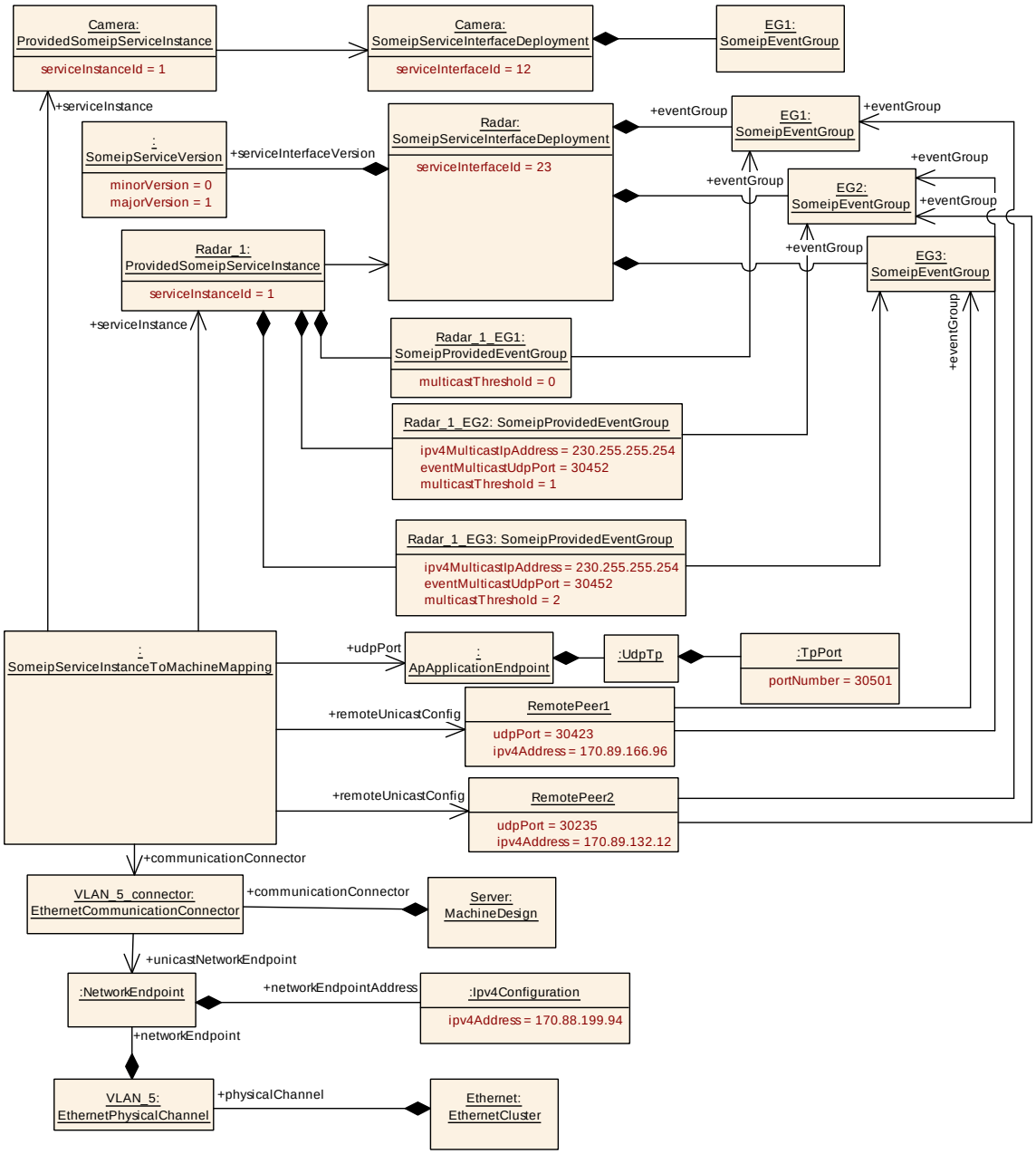


Figure B.27: Example of a Service Provider with static configured remote peers

B.2.6.2 Required Service Instance with remote Peers

The following example in Figure B.28 shows the setup of a Service Consumer with a statically defined remote peer. The `ServiceInstanceToMachineMapping` maps the Radar_CSI1 `RequiredSomeipServiceInstance` to an `ApApplicationEndpoint`.

The remote peer is referenced by the same `ServiceInstanceToMachineMapping` in the role `remoteUnicastConfig`.

The service consumer may receive event notifications of EG2 and EG3 over the multicast address and therefore the `ServiceInstanceToMachineMapping` references the `SomeipRemoteMulticastConfig`.

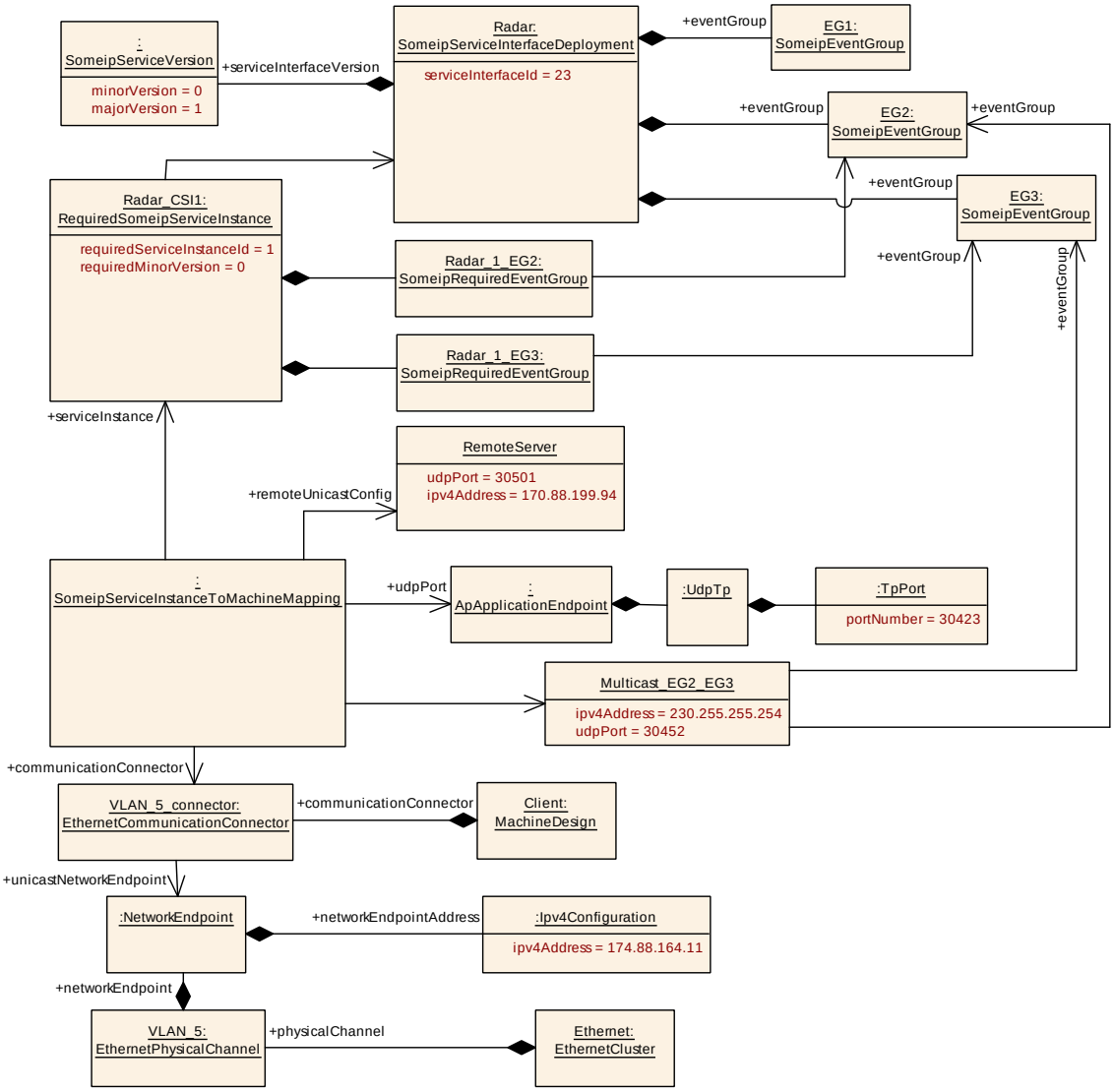


Figure B.28: Example of a Service Consumer with static configured remote peer

B.3 Execution Management

B.3.1 Definition of Startup Configuration

As already mentioned, the mode-dependent startup configuration is directly aggregated by the definition of a `Process`:

```
<PROCESS>
  <SHORT-NAME>AA1</SHORT-NAME>
  <STATE-DEPENDENT-STARTUP-CONFIGS>
    <STATE-DEPENDENT-STARTUP-CONFIG>
      <EXECUTION-DEPENDENCYS>
```

```
        <EXECUTION-DEPENDENCY>
          <PROCESS-STATE-IREF>
            <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-DECLARATION-GROUP-
                PROTOTYPE">/Processes/MWC/ProcessStateMachine</CONTEXT-MODE-DECLARATION-GROUP-
                PROTOTYPE-REF>
            <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/ModeDeclarationGroups/
                ProcessStateMachine/Running</TARGET-MODE-DECLARATION-REF>
          </PROCESS-STATE-IREF>
        </EXECUTION-DEPENDENCY>
        <EXECUTION-DEPENDENCY>
          <PROCESS-STATE-IREF>
            <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-DECLARATION-GROUP-
                PROTOTYPE">/Processes/MSM/ProcessStateMachine</CONTEXT-MODE-DECLARATION-GROUP-
                PROTOTYPE-REF>
            <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/ModeDeclarationGroups/
                ProcessStateMachine/Running</TARGET-MODE-DECLARATION-REF>
          </PROCESS-STATE-IREF>
        </EXECUTION-DEPENDENCY>
      </EXECUTION-DEPENDENCYS>
      <FUNCTION-GROUP-STATE-IREFS>
        <FUNCTION-GROUP-STATE-IREF>
          <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-DECLARATION-GROUP-PROTOTYPE
              ">/FunctionGroupSets/ExampleFGS/ExampleFG</CONTEXT-MODE-DECLARATION-GROUP-
              PROTOTYPE-REF>
          <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/ModeDeclarationGroups/
              ExampleFG/Running</TARGET-MODE-DECLARATION-REF>
        </FUNCTION-GROUP-STATE-IREF>
      </FUNCTION-GROUP-STATE-IREFS>
      <PROCESS-ARGUMENTS>
        <PROCESS-ARGUMENT>
          <ARGUMENT>-a</ARGUMENT>
        </PROCESS-ARGUMENT>
        <PROCESS-ARGUMENT>
          <ARGUMENT>-b</ARGUMENT>
        </PROCESS-ARGUMENT>
        <PROCESS-ARGUMENT>
          <ARGUMENT>-d</ARGUMENT>
        </PROCESS-ARGUMENT>
        <PROCESS-ARGUMENT>
          <ARGUMENT>XYZ</ARGUMENT>
        </PROCESS-ARGUMENT>
      </PROCESS-ARGUMENTS>
      <RESOURCE-GROUP-REF DEST="RESOURCE-GROUP">/Machines/ExampleMachine/Linux/resourceGroup2<
          /RESOURCE-GROUP-REF>
      <SCHEDULING-POLICY>SCHEDULING-POLICY-FIFO</SCHEDULING-POLICY>
      <SCHEDULING-PRIORITY>20</SCHEDULING-PRIORITY>
    </STATE-DEPENDENT-STARTUP-CONFIG>
  </STATE-DEPENDENT-STARTUP-CONFIGS>
</PROCESS>
```

Listing B.3: Example for the definition of the `StateDependentStartupConfig` owned by a `Process`

In this example, launch dependencies exist on two other `Process`es. Both `Process`es MWC and MSM need to be in the ProcessState “Running” before AA1 is started.

The reference `StateDependentStartupConfig.functionGroupState` refers to a `ModeDeclaration` with the `shortName` “Running” within the `Function Group` “ExampleFG”.

The `StateDependentStartupConfig` of the `Process` is assigned to the `ResourceGroup` named “ResourceGroup2” that is defined in the Machine Manifest.

The corresponding definition of a Machine contains a OsModuleInstantiation that in turn owns the two ResourceGroups named “ResourceGroup1” and “ResourceGroup2”. This aspect can be found in Listing B.4.

```
<MACHINE>
  <SHORT-NAME>ExampleMachine</SHORT-NAME>
  <MODULE-INSTANTIATIONS>
    <OS-MODULE-INSTANTIATION>
      <SHORT-NAME>Linux</SHORT-NAME>
      <RESOURCE-GROUPS>
        <RESOURCE-GROUP>
          <SHORT-NAME>resourceGroup1</SHORT-NAME>
          <CPU-USAGE>60</CPU-USAGE>
          <MEM-USAGE>1000000</MEM-USAGE>
        </RESOURCE-GROUP>
        <RESOURCE-GROUP>
          <SHORT-NAME>resourceGroup2</SHORT-NAME>
          <CPU-USAGE>70</CPU-USAGE>
          <MEM-USAGE>2000000</MEM-USAGE>
        </RESOURCE-GROUP>
      </RESOURCE-GROUPS>
    </OS-MODULE-INSTANTIATION>
  </MODULE-INSTANTIATIONS>
</MACHINE>
```

Listing B.4: Example for the definition of a Machine

The example definition of a FunctionGroupSet is sketched in Listing B.5.

```
<FUNCTION-GROUP-SET>
  <SHORT-NAME>ExampleFGS</SHORT-NAME>
  <FUNCTION-GROUPS>
    <MODE-DECLARATION-GROUP-PROTOTYPE>
      <SHORT-NAME>ExampleFG</SHORT-NAME>
      <TYPE-TREF DEST="MODE-DECLARATION-GROUP">/ModeDeclarationGroups/ExampleFG</TYPE-TREF>
    </MODE-DECLARATION-GROUP-PROTOTYPE>
  </FUNCTION-GROUPS>
</FUNCTION-GROUP-SET>
```

Listing B.5: Example for the definition of a FunctionGroupSet

The definition of the ModeDeclarationGroup that represents the Function Group is contained in Listing B.6.

```
<MODE-DECLARATION-GROUP>
  <SHORT-NAME>ExampleFG</SHORT-NAME>
  <INITIAL-MODE-REF DEST="MODE-DECLARATION">/ModeDeclarationGroups/ExampleFG/Off</INITIAL-MODE
      -REF>
  <MODE-DECLARATIONS>
    <MODE-DECLARATION>
      <SHORT-NAME>Off</SHORT-NAME>
    </MODE-DECLARATION>
    <MODE-DECLARATION>
      <SHORT-NAME>Running</SHORT-NAME>
    </MODE-DECLARATION>
    <MODE-DECLARATION>
      <SHORT-NAME>Fallback</SHORT-NAME>
    </MODE-DECLARATION>
    <MODE-DECLARATION>
      <SHORT-NAME>Diag</SHORT-NAME>
    </MODE-DECLARATION>
  </MODE-DECLARATIONS>
</MODE-DECLARATION-GROUP>
```

Listing B.6: Example for the definition of ModeDeclarationGroups

The definition of the ModeDeclarationGroup that represents the process state machine is contained in Listing B.7.

```
<MODE-DECLARATION-GROUP>
  <SHORT-NAME>ProcessStateMachine</SHORT-NAME>
  <INITIAL-MODE-REF DEST="MODE-DECLARATION">/ModeDeclarationGroups/ProcessStateMachine/
    Terminated</INITIAL-MODE-REF>
  <MODE-DECLARATIONS>
    <MODE-DECLARATION>
      <SHORT-NAME>Running</SHORT-NAME>
    </MODE-DECLARATION>
    <MODE-DECLARATION>
      <SHORT-NAME>Terminated</SHORT-NAME>
    </MODE-DECLARATION>
  </MODE-DECLARATIONS>
</MODE-DECLARATION-GROUP>
```

Listing B.7: Example for the definition of `ModeDeclarationGroup`s

B.4 Persistency

B.4.1 Definition of Persistent Data

This chapter contains examples for the modeling of persistent data and file storage starting form the design aspect down to the definition of the persistent storage and the mapping between design and target configuration.

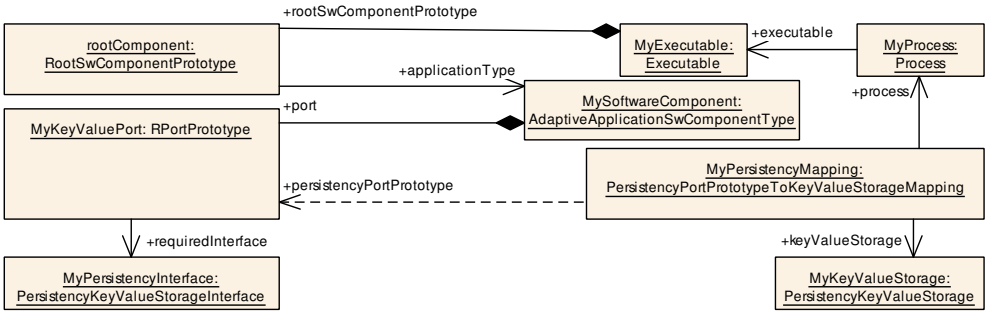


Figure B.29: Simple example modeling of persistent data (design + target configuration)

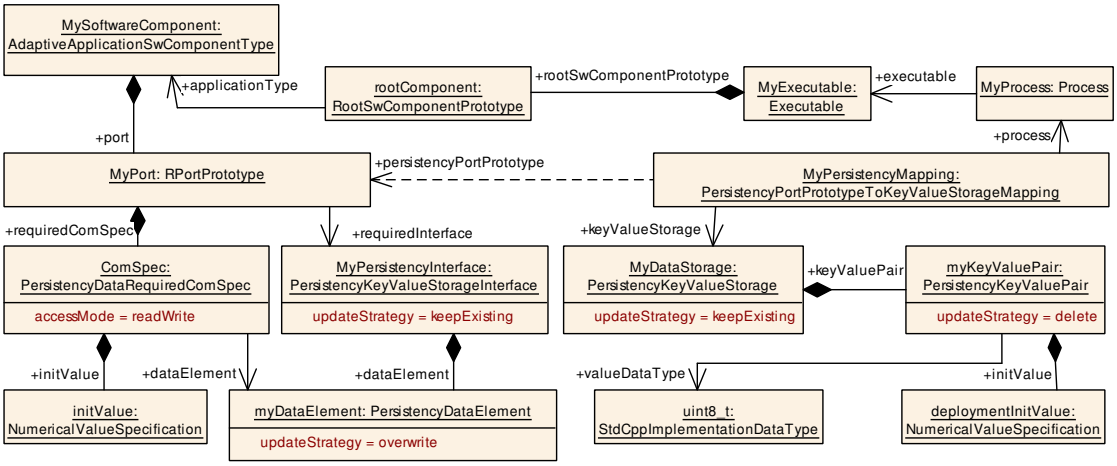


Figure B.30: Advanced example modeling of persistent data (design + target configuration)

The setup presented in Figure B.29 represents a case with reduced modeling of persistent data.

It is possible to extend the modeling to a deeper level of detail and also formally describe the individual data that is subject to persistency on both design and target configuration level, see Figure B.30.

B.4.2 Definition of Persistent File

Figure B.31 represents a case with reduced modeling of persistent files.

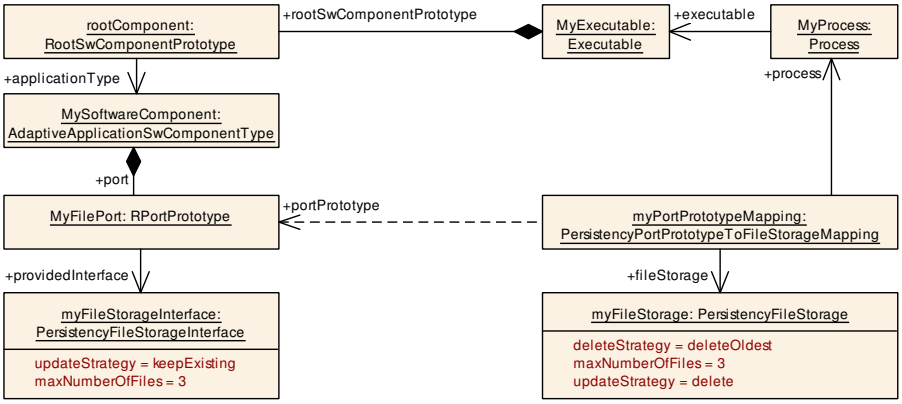


Figure B.31: Simple example modeling of persistent file (design + target configuration)

It is possible to extend the modeling to a deeper level of detail and also formally describe the individual file that is subject to persistency on both design and target configuration level, see Figure B.32.

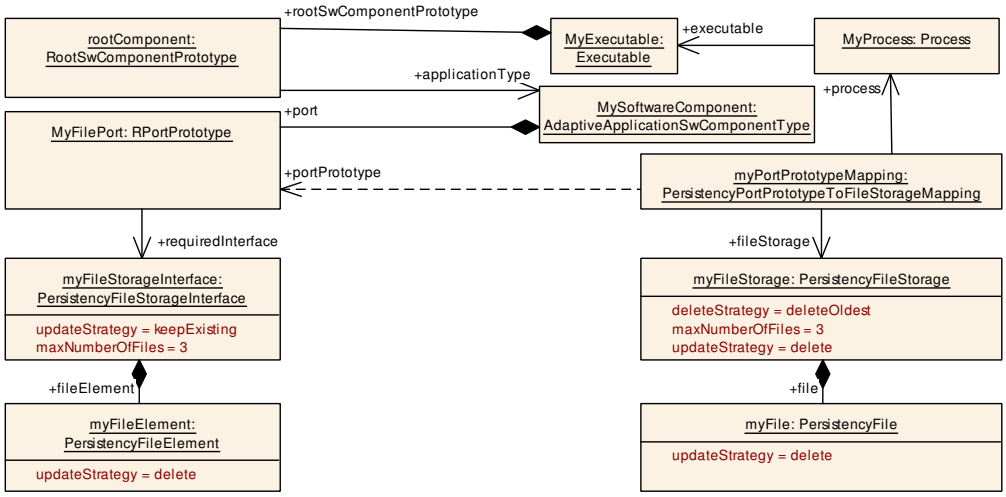


Figure B.32: Advanced example modeling of persistent file (design + target configuration)

B.5 Interaction with the PHM

This chapter contains examples for the modeling of platform health management. The example is structured into Application design and platform health management configuration.

B.5.1 Relation between Instance Specifier and Instance Ref

Figure B.33 exemplifies the relation between an `InstanceSpecifier` and an instanceRef. If the `InstanceSpecifier` as well as the model structure is known then the instanceRef can be constructed on the base of the information contained in the `InstanceSpecifier` and vice versa.

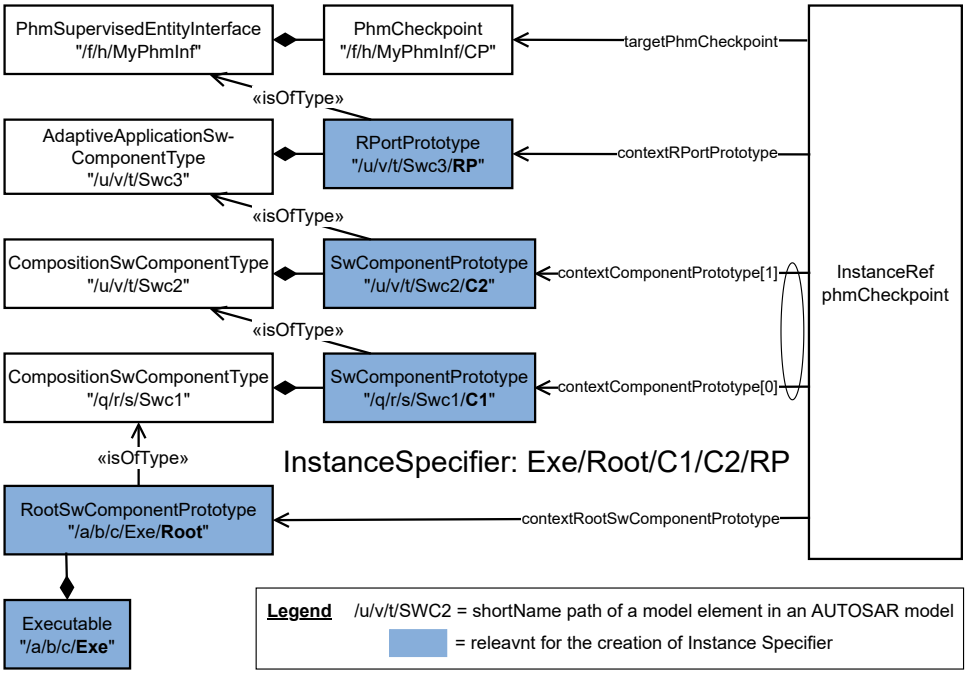


Figure B.33: Example for the creation of an `InstanceSpecifier` of a *SupervisedEntity*

B.5.2 Phm Application Design example

The simple example provided in Figure B.34 shows the definition of a `PhmSupervisedEntityInterface`. This example will also be used in the subsequent section to define the platform health management configuration.

The `PhmSupervisedEntityInterface` *SupervisedEntity_B* defines two checkpoints:

- *CP1*
- *CP2*

The AdaptiveApplicationSwComponentType *AdaptiveApplication* defines an RPortPrototype *Se_B* typed by *SupervisedEntity_B*.

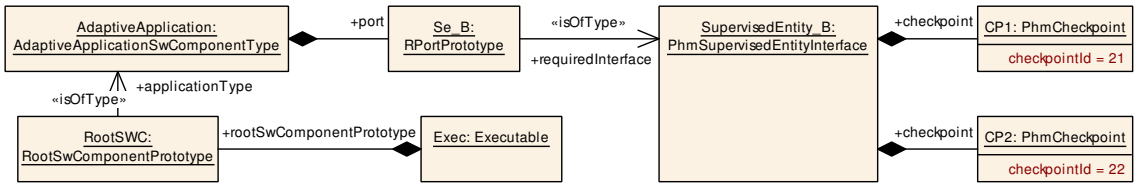


Figure B.34: Example modeling of Supervised Entity

B.5.3 Phm configuration example

When defining the configuration contribution for Phm it is required to first create representatives of the application design model artifacts (supervised entity checkpoints) in the Phm configuration context. This is shown in Figure B.35.

In this example the *PHM* PlatformHealthManagementContribution defines placeholder elements which refer to the respective application design model artifacts:

Example supervision checkpoint:

- *Se_B_Cp1* refers to the *CP1* checkpoint of *SupervisedEntity_B*
- *Se_B_Cp2* refers to the *CP2* checkpoint of *SupervisedEntity_B*

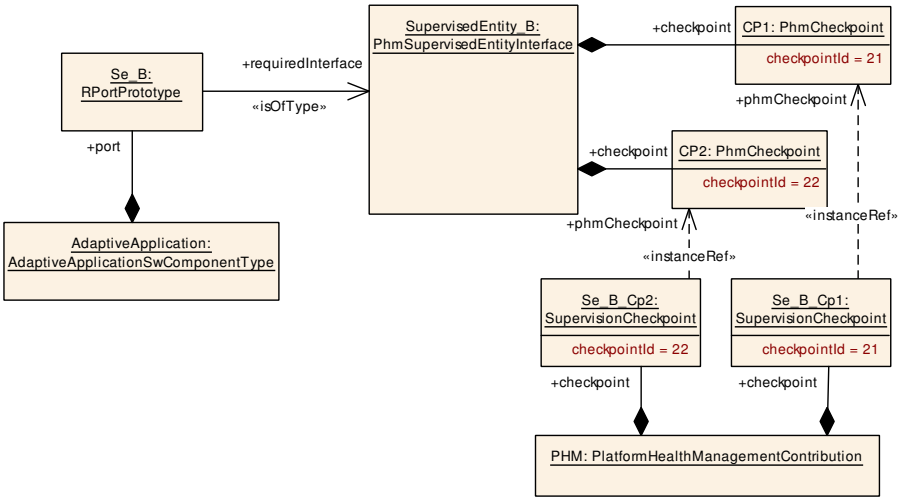


Figure B.35: Example modeling of Phm placeholder definition

Note how the numbers from the PhmCheckpoint.checkpointId have been replicated in SupervisionCheckpoint.checkpointId.

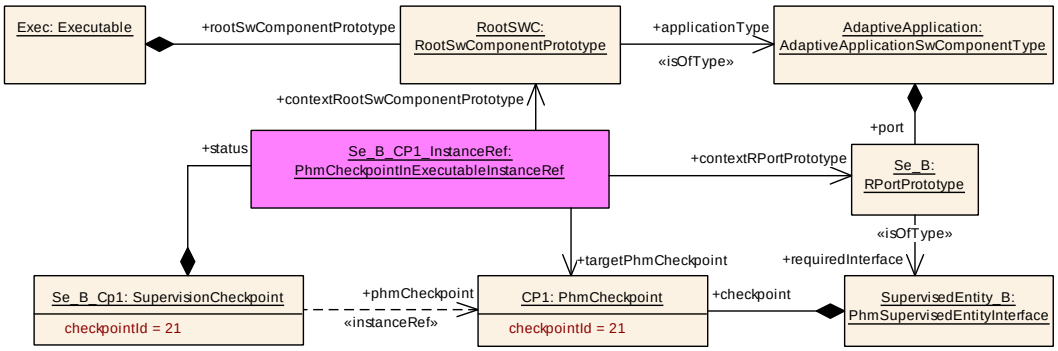


Figure B.36: Example modeling of Phm instance reference

Note that these instance references have a composite nature, which is shown in example Figure B.36. Here it is shown that in order to instance reference from the `SupervisionCheckpoint` *Se_B_Cp1* to the `PhmCheckpoint` *CP1* there is the structured reference required consisting of

- `contextRootSwComponentPrototype`
- `contextRPortPrototype`
- `targetPhmCheckpoint`

B.6 State Management

B.6.1 Consideration of User Perspective

The modeling of state management is described in Chapter 4.5 and Chapter 12.14.

Obviously, the definition of the meta-classes to serialize the state management configuration does not have to directly reflect the mental model which some creators of a specific state management instance have in mind.

This means that the design approach in the user domain might be entirely different from the definition of the respective meta-classes. Such a situation is manageable as long as the mental model applied on the side of the users can be mapped to the meta-classes and their relationships.

To be more concrete, some of the users who define state management configurations, **prefer to think in terms of tables that combine an input trigger or error value with a target state**, see Table B.3 as an example.

This chapter exemplifies such an approach for how a table-based configuration of state changes could look like and how the table columns map to the meta-model.

B.6.2 Scenario for Trigger

For the definition of the response to triggers, some people define a table, which maps the *trigger value* to the assumed *current state* and the *next state*.

In other words, if the *trigger value* is found in one row and the value of the current state in the state machine is equal to the assumed *current state* in the applicable table row, then the state machine shall be switched to the *next state* listed in the table row.

Please note that the concrete table entries for the states might consist of numerals or literals, depending on the personal preference of the user.

Of course, the meta-model allows to create a connection between the table values either by means of the numerical `ModeDeclaration.value` or the textual `ModeDeclaration.shortName`.

An example for such a table is listed in Table B.3.

Trigger Value	Current State	Next State
1001	0	1
1000	1	0
1000	2	0
1000	3	0
1000	4	0

Table B.3: Example table for associating trigger with state changes using the `value`s

It is important that it is possible to map this mental model of state management configuration to the existing meta-model.

Alternatively, it would be possible to create the table using the `shortName` of the respective `ModeDeclaration`s and thereby create the exact same semantics, see Table B.4.

Trigger Value	Current State	Next State
1001	StateOff	StateFull_primary
1000	StateFull_primary	StateOff
1000	State_Fallback	StateOff
1000	State3	StateOff
1000	State4	StateOff

Table B.4: Example table for associating trigger with state changes using the `shortName`

Thus, the correspondence of table columns to meta-model elements is documented in Table B.5.

Table Column of Table B.3	Meta-Model Element, see Figure 12.75
Trigger Value	StateManagementCompare-Condition.compareValue
Current State	StateManagementTriggerCompare-Rule.assumedCurrentState
Next State	StateManagementRequestRule.next-State

Table B.5: Correspondence of table columns and meta-model elements for triggers

In particular, the Figure B.37 and, with more detail, Listing B.8 depict the modeling of the relevant part for the first row of Table B.3.

The relevant parts of the table row are located inside the red box in Figure B.37.

The concrete usage of StateManagementCompareCondition.compareValue can be found in the listing as part of the STATE-MANAGEMENT-TRIGGER-COMPARE-RULE, specifically by COMPARE-TYPE and COMPARE-VALUE.

The StateManagementTriggerCompareRule.assumedCurrentState is represented by the ASSUMED-CURRENT-STATE-IREF.

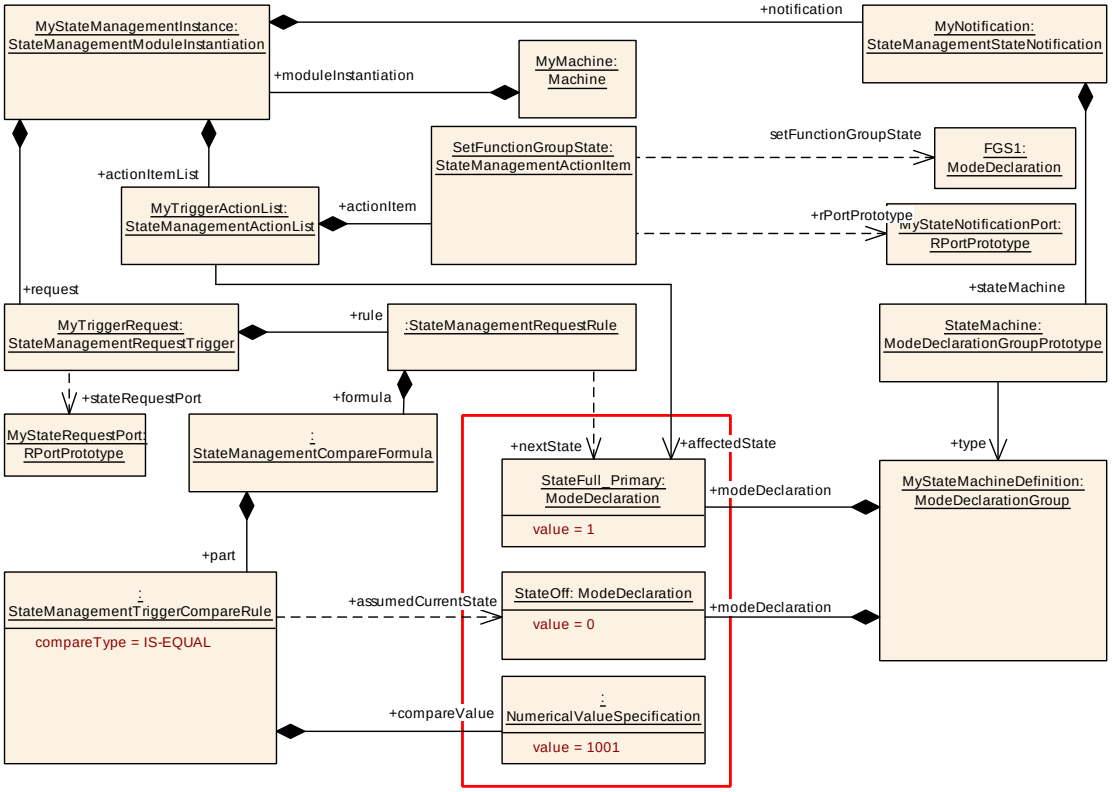


Figure B.37: Example state management modeling for a trigger request

And finally, the StateManagementRequestRule.nextState can be found at the NEXT-STATE-IREF inside the STATE-MANAGEMENT-RQUEST-RULE.

<STATE-MANAGEMENT-REQUEST-TRIGGER>

```
  <SHORT-NAME>MyTriggerRequest</SHORT-NAME>
  <STATE-REQUEST-PORT-IREF>
    <CONTEXT-ROOT-SW-COMPONENT-PROTOTYPE-REF DEST="ROOT-SW-COMPONENT-PROTOTYPE">/Executables/
        MyStateManager/MyRootComponent</CONTEXT-ROOT-SW-COMPONENT-PROTOTYPE-REF>
    <TARGET-R-PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE">/SwComponentTypes/StateManagerComp/
        MyStateRequestPort</TARGET-R-PORT-PROTOTYPE-REF>
  </STATE-REQUEST-PORT-IREF>
  <RULES>
    <STATE-MANAGEMENT-REQUEST-RULE>
      <FORMULA>
        <PARTS>
          <STATE-MANAGEMENT-TRIGGER-COMPARE-RULE>
            <COMPARE-TYPE>IS-EQUAL</COMPARE-TYPE>
            <COMPARE-VALUE>
              <NUMERICAL-VALUE-SPECIFICATION>
                <VALUE>1001</VALUE>
              </NUMERICAL-VALUE-SPECIFICATION>
            </COMPARE-VALUE>
            <ASSUMED-CURRENT-STATE-IREF>
              <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-DECLARATION-GROUP-
                  PROTOTYPE">/Machines/MyMachine/MyStateManagementInstance/MyNotification/
                  StateMachine</CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF>
              <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/ModeDeclarationGroups/
                  MyStateMachineDefinition/StateOff</TARGET-MODE-DECLARATION-REF>
            </ASSUMED-CURRENT-STATE-IREF>
          </STATE-MANAGEMENT-TRIGGER-COMPARE-RULE>
        </PARTS>
      </FORMULA>
      <NEXT-STATE-IREF>
        <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-DECLARATION-GROUP-PROTOTYPE">
            /Machines/MyMachine/MyStateManagementInstance/MyNotification/StateMachine</CONTEXT-
            MODE-DECLARATION-GROUP-PROTOTYPE-REF>
        <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/ModeDeclarationGroups/
            MyStateMachineDefinition/StateFull_primary</TARGET-MODE-DECLARATION-REF>
      </NEXT-STATE-IREF>
    </STATE-MANAGEMENT-REQUEST-RULE>
  </RULES>
</STATE-MANAGEMENT-REQUEST-TRIGGER>
```

Listing B.8: Example for the definition of a `StateManagementRequestTrigger`

The `ModeDeclarationGroup` that types the internal state machine is depicted in Listing B.9. The definition of values that represent the states in Table B.3 are visible in the listing, see `VALUE`.

```
<MODE-DECLARATION-GROUP>
  <SHORT-NAME>MyStateMachineDefinition</SHORT-NAME>
  <MODE-DECLARATIONS>
    <MODE-DECLARATION>
      <SHORT-NAME>StateOff</SHORT-NAME>
      <VALUE>0</VALUE>
    </MODE-DECLARATION>
    <MODE-DECLARATION>
      <SHORT-NAME>StateFull_primary</SHORT-NAME>
      <VALUE>1</VALUE>
    </MODE-DECLARATION>
    <MODE-DECLARATION>
      <SHORT-NAME>State_Fallback</SHORT-NAME>
      <VALUE>2</VALUE>
    </MODE-DECLARATION>
  </MODE-DECLARATIONS>
</MODE-DECLARATION-GROUP>
```

Listing B.9: Example for the definition of a `ModeDeclarationGroup` to define the states of an internal state machine

The modeling of the applicable StateManagementActionList is sketched in Listing B.10.

Please note that the modeling of the PortPrototypes and the internal structure of the Executable is not explicitly documented in this example because there are other example demonstrating this aspect already.

```
<STATE-MANAGEMENT-ACTION-LIST>
  <SHORT-NAME>MyTriggerActionList</SHORT-NAME>
  <ACTION-ITEMS>
    <STATE-MANAGEMENT-SET-FUNCTION-GROUP-STATE-ACTION-ITEM>
      <SHORT-NAME>SetFunctionGroupState</SHORT-NAME>
      <R-PORT-PROTOTYPE-IREF>
        <CONTEXT-ROOT-SW-COMPONENT-PROTOTYPE-REF DEST="ROOT-SW-COMPONENT-PROTOTYPE">/
            Executables/MyStateManager/MyRootComponent</CONTEXT-ROOT-SW-COMPONENT-PROTOTYPE-REF
            >
        <TARGET-R-PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE">/SwComponentTypes/
            StateManagerComp/MyStateNotificationPort</TARGET-R-PORT-PROTOTYPE-REF>
      </R-PORT-PROTOTYPE-IREF>
      <SET-FUNCTION-GROUP-STATE-IREF>
        <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-DECLARATION-GROUP-PROTOTYPE">
            /FunctionGroups/MyFumctionGroups/FG1</CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF>
        <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/ModeDeclarationGroups/
            MyFunctionGroupStates/FGS1</TARGET-MODE-DECLARATION-REF>
      </SET-FUNCTION-GROUP-STATE-IREF>
    </STATE-MANAGEMENT-SET-FUNCTION-GROUP-STATE-ACTION-ITEM>
  </ACTION-ITEMS>
</STATE-MANAGEMENT-ACTION-LIST>
```

Listing B.10: Example for the definition of a StateManagementActionList for the trigger request

The modeling of the FunctionGroupSet mentioned in the example is sketched in Listing B.11.

```
<FUNCTION-GROUP-SET>
  <SHORT-NAME>MyFumctionGroups</SHORT-NAME>
  <FUNCTION-GROUPS>
    <MODE-DECLARATION-GROUP-PROTOTYPE>
      <SHORT-NAME>FG1</SHORT-NAME>
      <TYPE-TREF DEST="MODE-DECLARATION-GROUP">/ModeDeclarationGroups/MyFunctionGroupStates</
          TYPE-TREF>
    </MODE-DECLARATION-GROUP-PROTOTYPE>
  </FUNCTION-GROUPS>
</FUNCTION-GROUP-SET>
```

Listing B.11: Example for the definition of a FunctionGroupSet for the discussed example

The modeling of the ModeDeclarationGroup that defines the Function Group States named “FGS1” and “FGS2” that are used in the example is sketched in Listing B.12.

```
<MODE-DECLARATION-GROUP>
  <SHORT-NAME>MyFunctionGroupStates</SHORT-NAME>
  <MODE-DECLARATIONS>
    <MODE-DECLARATION>
      <SHORT-NAME>FGS1</SHORT-NAME>
    </MODE-DECLARATION>
    <MODE-DECLARATION>
      <SHORT-NAME>FGS2</SHORT-NAME>
    </MODE-DECLARATION>
  </MODE-DECLARATIONS>
```

```
</MODE-DECLARATION-GROUP>
```

Listing B.12: Example for the definition of a `ModeDeclarationGroup` for defining the `Function Group State`s used in the discussed example

Finally, the modeling of the `StateManagementStateNotification` is sketched in Listing B.13.

Please note that the definition of the `STATE-MACHINE` in Listing B.13 relates to the definition of internal states (modeled as a `MODE-DECLARATION-GROUP`), as sketched in Listing B.9.

```
<STATE-MANAGEMENT-STATE-NOTIFICATION>
  <SHORT-NAME>MyNotification</SHORT-NAME>
  <NOTIFICATION-PORT-IREF>
    <CONTEXT-ROOT-SW-COMPONENT-PROTOTYPE-REF DEST="ROOT-SW-COMPONENT-PROTOTYPE">/Executables/
       MyStateManager/MyRootComponent</CONTEXT-ROOT-SW-COMPONENT-PROTOTYPE-REF>
    <TARGET-P-PORT-PROTOTYPE-REF DEST="P-PORT-PROTOTYPE">/SwComponentTypes/StateManagerComp/
       MyStateChangeNotificationPort</TARGET-P-PORT-PROTOTYPE-REF>
  </NOTIFICATION-PORT-IREF>
  <STATE-MACHINE>
    <SHORT-NAME>StateMachine</SHORT-NAME>
    <TYPE-TREF DEST="MODE-DECLARATION-GROUP">/ModeDeclarationGroups/MyStateMachineDefinition</
       TYPE-TREF>
  </STATE-MACHINE>
</STATE-MANAGEMENT-STATE-NOTIFICATION>
```

Listing B.13: Example for the definition of a `StateManagementStateNotification` for communicating state changes within the discussed example

B.6.3 Scenario for Error

For the definition of the response to errors, some people define a table, which maps the *trigger value* and the *next state* that is to be switched to if the particular error value is received.

In other words, if the value of *error code* is found in one row, then the state machine shall be switched to the *next state* listed in the table row, **independently** of the values of the current state of the state machine.

An example for such a table is listed in Table B.6.

Error Code	Next State
11	2
12	3
111	2
23	4
24	0
244	5

Table B.6: Example table for associating error with state changes

The correspondence of table columns to meta-model elements is documented in Table B.7.

Table Column of Table B.6	Meta-Model Element, see Figure 12.76
ErrorCode	`StateManagementCompareCondition.compareValue`
Next State	`StateManagementRequestRule.nextState`

Table B.7: Correspondence of table columns and meta-model elements for errors

In particular, the Figure B.38 and, with more detail, Listing B.14 depict the modeling of the relevant part for the first row of Table B.6.

The relevant parts of the table row are located inside the red box in Figure B.38.

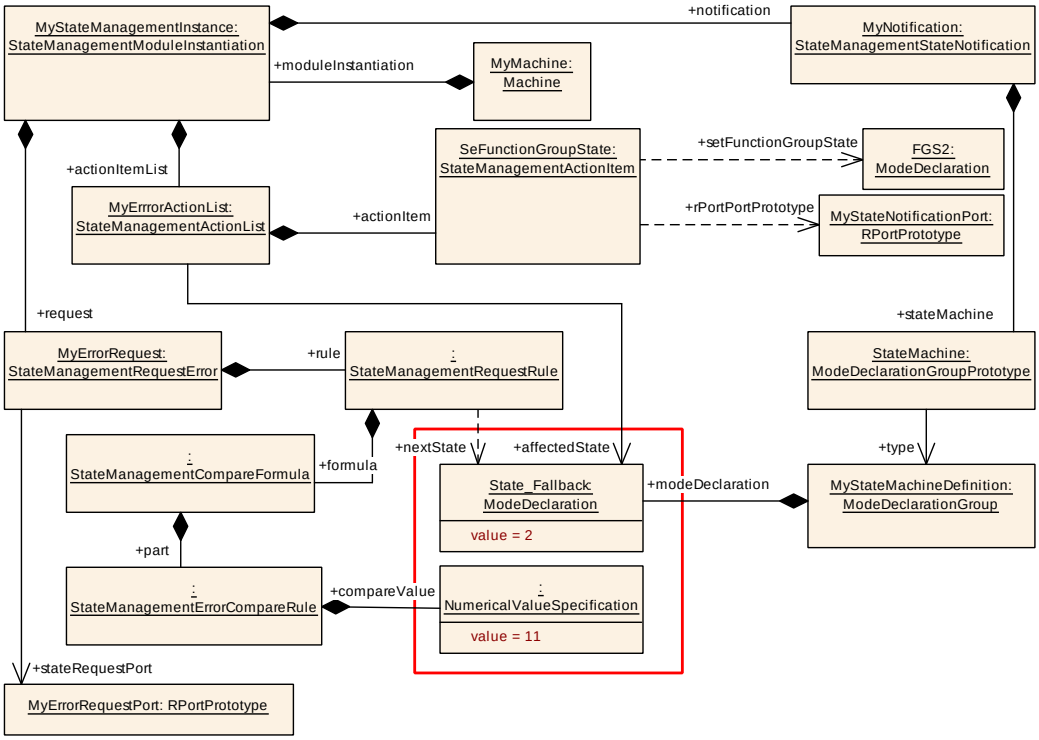


Figure B.38: Example state management modeling for an error request

This example features partly the same model elements as the previous trigger example.

But in this specific case, there is a different `StateManagementActionList` named “MyErrorActionList” that refers to different `modeDeclaration` named “State_Fallback” (that is represented by the numerical value 2) in the role `affectedState`.

Obviously, a `StateManagementRequestError` named “MyErrorRequest” exists that aggregates its own `rule`. The `StateManagementErrorCompareRule.compareValue` has the `value` 11.

This fits to the first row of Table B.6.

```
<STATE-MANAGEMENT-REQUEST-ERROR>
  <SHORT-NAME>MyErrorRequest</SHORT-NAME>
  <STATE-REQUEST-PORT-IREF>
    <CONTEXT-ROOT-SW-COMPONENT-PROTOTYPE-REF DEST="ROOT-SW-COMPONENT-PROTOTYPE">/Executables/
        MyStateManager/MyRootComponent</CONTEXT-ROOT-SW-COMPONENT-PROTOTYPE-REF>
    <TARGET-R-PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE">/SwComponentTypes/StateManagerComp/
        MyErrorRequestPort</TARGET-R-PORT-PROTOTYPE-REF>
  </STATE-REQUEST-PORT-IREF>
  <RULES>
    <STATE-MANAGEMENT-REQUEST-RULE>
      <FORMULA>
        <PARTS>
          <STATE-MANAGEMENT-ERROR-COMPARE-RULE>
            <COMPARE-TYPE>IS-EQUAL</COMPARE-TYPE>
            <COMPARE-VALUE>
              <NUMERICAL-VALUE-SPECIFICATION>
                <VALUE>11</VALUE>
              </NUMERICAL-VALUE-SPECIFICATION>
            </COMPARE-VALUE>
          </STATE-MANAGEMENT-ERROR-COMPARE-RULE>
        </PARTS>
      </FORMULA>
      <NEXT-STATE-IREF>
        <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-DECLARATION-GROUP-PROTOTYPE">
            /Machines/MyMachine/MyStateManagementInstance/MyNotification/StateMachine</CONTEXT-
            MODE-DECLARATION-GROUP-PROTOTYPE-REF>
        <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/ModeDeclarationGroups/
            MyStateMachineDefinition/State_Fallback</TARGET-MODE-DECLARATION-REF>
      </NEXT-STATE-IREF>
    </STATE-MANAGEMENT-REQUEST-RULE>
  </RULES>
</STATE-MANAGEMENT-REQUEST-ERROR>
```

Listing B.14: Example for the definition of a `StateManagementRequestError`

The modeling of the applicable `StateManagementActionList` is sketched in Listing B.15.

```
<STATE-MANAGEMENT-ACTION-LIST>
  <SHORT-NAME>MyErrorActionList</SHORT-NAME>
  <ACTION-ITEMS>
    <STATE-MANAGEMENT-SET-FUNCTION-GROUP-STATE-ACTION-ITEM>
      <SHORT-NAME>SetFunctionGroupState</SHORT-NAME>
      <R-PORT-PROTOTYPE-IREF>
        <CONTEXT-ROOT-SW-COMPONENT-PROTOTYPE-REF DEST="ROOT-SW-COMPONENT-PROTOTYPE">/
            Executables/MyStateManager/MyRootComponent</CONTEXT-ROOT-SW-COMPONENT-PROTOTYPE-REF
            >
        <TARGET-R-PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE">/SwComponentTypes/
            StateManagerComp/MyStateNotificationPort</TARGET-R-PORT-PROTOTYPE-REF>
      </R-PORT-PROTOTYPE-IREF>
      <SET-FUNCTION-GROUP-STATE-IREF>
        <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-DECLARATION-GROUP-PROTOTYPE">
            /FunctionGroups/MyFumctionGroups/FG1</CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF>
        <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/ModeDeclarationGroups/
            MyFunctionGroupStates/FGS2</TARGET-MODE-DECLARATION-REF>
      </SET-FUNCTION-GROUP-STATE-IREF>
    </STATE-MANAGEMENT-SET-FUNCTION-GROUP-STATE-ACTION-ITEM>
  </ACTION-ITEMS>
</STATE-MANAGEMENT-ACTION-LIST>
```

Listing B.15: Example for the definition of a `StateManagementActionList` for the error request

B.7 Crypto

B.7.1 Relation between a Crypto Certificate and a Crypto Key Slot

The following Figure B.39 shows an example how the `SwcServiceDependency` is used to create a relation between a Crypto Certificate and a Crypto KeySlot.

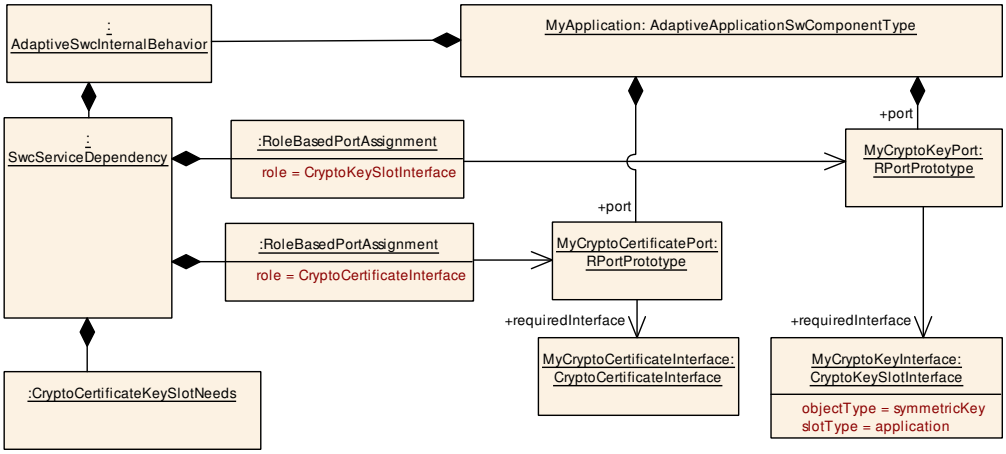


Figure B.39: Example that shows a link between a Port typed by `CryptoKeySlotInterface` and a Port typed by `CryptoCertificateInterface`

B.8 Machine Manifest

B.8.1 Distributed Definition of Environment Variables

The aggregation `Machine.environmentVariable` may comprise of contributions to the definition of environment variables from **different sources**.

As an example, assume two partial models (sketched in Listing B.16 and Listing B.17) 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 B.16: 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 B.17: 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 B.18 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 B.18: 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;
```

B.9 Ethernet Raw Data Stream Configuration

B.9.1 Preamble

The examples discussed in this section of the document are based on the modeling of two `EthernetCommunicationConnector`s, one for the client (see Listing B.19) and one for the server (see Listing B.20)

```
<ETHERNET-COMMUNICATION-CONNECTOR>
  <SHORT-NAME>ClientConn</SHORT-NAME>
  <AP-APPLICATION-ENDPOINTS>
    <AP-APPLICATION-ENDPOINT>
      <SHORT-NAME>ClientUnicastAEP</SHORT-NAME>
      <TP-CONFIGURATION>
        <UDP-TP>
          <UDP-TP-PORT>
            <PORT-NUMBER>3333</PORT-NUMBER>
          </UDP-TP-PORT>
        </UDP-TP>
      </TP-CONFIGURATION>
    </AP-APPLICATION-ENDPOINT>
    <AP-APPLICATION-ENDPOINT>
      <SHORT-NAME>ClientMulticastAEP</SHORT-NAME>
      <TP-CONFIGURATION>
        <UDP-TP>
          <UDP-TP-PORT>
            <PORT-NUMBER>7654</PORT-NUMBER>
          </UDP-TP-PORT>
```

```
        </UDP-TP>
      </TP-CONFIGURATION>
    </AP-APPLICATION-ENDPOINT>
  </AP-APPLICATION-ENDPOINTS>
  <UNICAST-NETWORK-ENDPOINT-REFS>
    <UNICAST-NETWORK-ENDPOINT-REF DEST="NETWORK-ENDPOINT">/CommunicationClusters/Ether/VLAN1/
      ClientEP</UNICAST-NETWORK-ENDPOINT-REF>
  </UNICAST-NETWORK-ENDPOINT-REFS>
</ETHERNET-COMMUNICATION-CONNECTOR>
```

Listing B.19: Definition of `EthernetCommunicationConnector` for the client

```
<ETHERNET-COMMUNICATION-CONNECTOR>
  <SHORT-NAME>ServerConn</SHORT-NAME>
  <AP-APPLICATION-ENDPOINTS>
    <AP-APPLICATION-ENDPOINT>
      <SHORT-NAME>ServerUnicastAEP</SHORT-NAME>
      <TP-CONFIGURATION>
        <UDP-TP>
          <UDP-TP-PORT>
            <PORT-NUMBER>7777</PORT-NUMBER>
          </UDP-TP-PORT>
        </UDP-TP>
      </TP-CONFIGURATION>
    </AP-APPLICATION-ENDPOINT>
    <AP-APPLICATION-ENDPOINT>
      <SHORT-NAME>ServerMulticastAEP</SHORT-NAME>
      <TP-CONFIGURATION>
        <UDP-TP>
          <UDP-TP-PORT>
            <PORT-NUMBER>7654</PORT-NUMBER>
          </UDP-TP-PORT>
        </UDP-TP>
      </TP-CONFIGURATION>
    </AP-APPLICATION-ENDPOINT>
  </AP-APPLICATION-ENDPOINTS>
  <UNICAST-NETWORK-ENDPOINT-REFS>
    <UNICAST-NETWORK-ENDPOINT-REF DEST="NETWORK-ENDPOINT">/CommunicationClusters/Ether/VLAN1/
      ServerEP</UNICAST-NETWORK-ENDPOINT-REF>
  </UNICAST-NETWORK-ENDPOINT-REFS>
</ETHERNET-COMMUNICATION-CONNECTOR>
```

Listing B.20: Definition of `EthernetCommunicationConnector` for the server

On top of that, the definition of a `EthernetPhysicalChannel` is also considered, see Listing B.21.

```
<ETHERNET-PHYSICAL-CHANNEL>
  <SHORT-NAME>VLAN1</SHORT-NAME>
  <NETWORK-ENDPOINTS>
    <NETWORK-ENDPOINT>
      <SHORT-NAME>ClientEP</SHORT-NAME>
      <NETWORK-ENDPOINT-ADDRESSES>
        <IPV-4-CONFIGURATION>
          <IPV-4-ADDRESS>192.168.2.240</IPV-4-ADDRESS>
        </IPV-4-CONFIGURATION>
      </NETWORK-ENDPOINT-ADDRESSES>
    </NETWORK-ENDPOINT>
    <NETWORK-ENDPOINT>
      <SHORT-NAME>ServerEP</SHORT-NAME>
      <NETWORK-ENDPOINT-ADDRESSES>
        <IPV-4-CONFIGURATION>
          <IPV-4-ADDRESS>192.168.2.223</IPV-4-ADDRESS>
        </IPV-4-CONFIGURATION>
      </NETWORK-ENDPOINT-ADDRESSES>
    </NETWORK-ENDPOINT>
  </NETWORK-ENDPOINTS>
```

```
</ETHERNET-PHYSICAL-CHANNEL>
```

Listing B.21: Definition of `EthernetPhysicalChannel` for connection between client and server

B.9.2 Fully formalized Model Example

A scenario where both the raw data stream client and raw data stream server are fully formalized inside an AUTOSAR model is depicted in Figure B.40.

In the case of a fully formalized model it may be possible to derive the client configuration (that also consists of the credentials of the remote server) from the server's model if the server model is accessible.

The AUTOSAR meta-model foresees the specification of the server credentials in the client's configuration by means of the aggregation `EthernetRawDataStreamClientMapping.remoteServerConfig`.

If multicast communication from server to client is foreseen, then the aggregation in the role `EthernetRawDataStreamClientMapping.remoteServerConfig.multicastCredentials` also needs to exist.

For the sake of simplicity, this example has been created to communicate the general idea of the configuration approach rather than to demonstrate a true-to-the-meta-model representation.

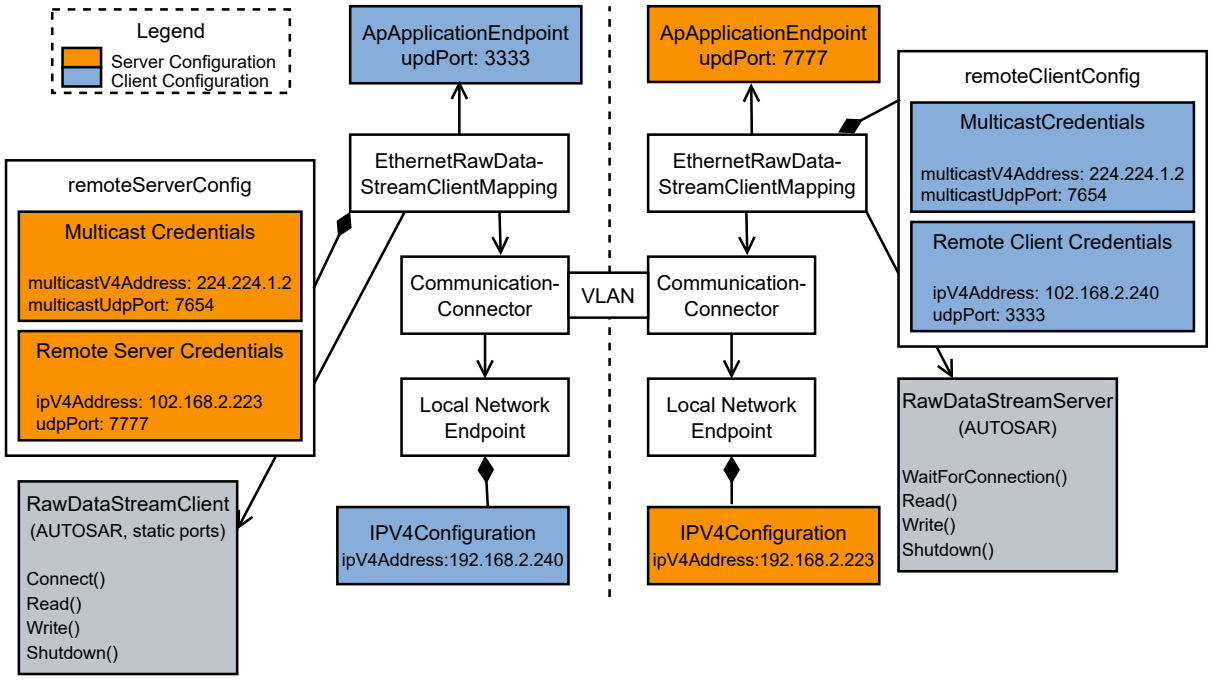


Figure B.40: Scenario where both the raw data stream client and raw data stream server are fully formalized inside an AUTOSAR model

Please note that this example utilizes UDP communication only. If the unicast communication were based on TCP, the server's configuration would (as mentioned in Chapter 14.3.2) not need the client's TCP credentials, i.e. `EthernetRawDataStreamServerMapping.remoteClientConfig` (the "Remote Client Credentials" in Figure B.40) would not exist.

In a fully formalized model, the configuration of the multicast credentials shall be identical for all affected communication ends. The simplified (the focus is on the Ethernet configuration, based on the modeling described in Chapter B.9.1) modeling of the client mapping is sketched in Listing B.22.

```
<ETHERNET-RAW-DATA-STREAM-CLIENT-MAPPING>
  <SHORT-NAME>ClientMap</SHORT-NAME>
  <LOCAL-ENDPOINT-CONFIG>
    <LOCAL-COMM-CONNECTOR-REF DEST="ETHERNET-COMMUNICATION-CONNECTOR">/MachineDesigns/Mach/
        ClientConn</LOCAL-COMM-CONNECTOR-REF>
    <LOCAL-UDP-PORT-REF DEST="AP-APPLICATION-ENDPOINT">/MachineDesigns/Mach/ClientConn/
        ClientUnicastAEP</LOCAL-UDP-PORT-REF>
  </LOCAL-ENDPOINT-CONFIG>
  <REMOTE-SERVER-CONFIG>
    <MULTICAST-CREDENTIALS>
      <IP-V-4-ADDRESS>224.224.1.2</IP-V-4-ADDRESS>
      <UDP-PORT>7654</UDP-PORT>
    </MULTICAST-CREDENTIALS>
    <UNICAST-CREDENTIALS>
      <IP-V-4-ADDRESS>192.168.2.223</IP-V-4-ADDRESS>
      <UDP-PORT>7777</UDP-PORT>
    </UNICAST-CREDENTIALS>
  </REMOTE-SERVER-CONFIG>
</ETHERNET-RAW-DATA-STREAM-CLIENT-MAPPING>
```

Listing B.22: Modeling of the client mapping

The simplified (the focus is on the Ethernet configuration, based on the modeling described in Chapter B.9.1) modeling of the server mapping is sketched in Listing B.23.

```
<ETHERNET-RAW-DATA-STREAM-SERVER-MAPPING>
  <SHORT-NAME>ServerMap</SHORT-NAME>
  <LOCAL-ENDPOINT-CONFIG>
    <LOCAL-COMM-CONNECTOR-REF DEST="ETHERNET-COMMUNICATION-CONNECTOR">/MachineDesigns/Mach/
        ServerConn</LOCAL-COMM-CONNECTOR-REF>
    <LOCAL-UDP-PORT-REF DEST="AP-APPLICATION-ENDPOINT">/MachineDesigns/Mach/ServerConn/
        ServerUnicastAEP</LOCAL-UDP-PORT-REF>
  </LOCAL-ENDPOINT-CONFIG>
  <REMOTE-CLIENT-CONFIG>
    <UNICAST-UDP-CREDENTIALS>
      <IP-V-4-ADDRESS>192.168.2.240</IP-V-4-ADDRESS>
      <UDP-PORT>3333</UDP-PORT>
    </UNICAST-UDP-CREDENTIALS>
  </REMOTE-CLIENT-CONFIG>
</ETHERNET-RAW-DATA-STREAM-SERVER-MAPPING>
```

Listing B.23: Modeling of the server mapping

B.9.3 Only Client is formalized

The case that **only** the client is formalized (and the server does not appear in the form of an `EthernetRawDataStreamServerMapping`) can be represented by a subset of the modeling described in Chapter B.9.2 as well as the model sketched in Listing B.22.

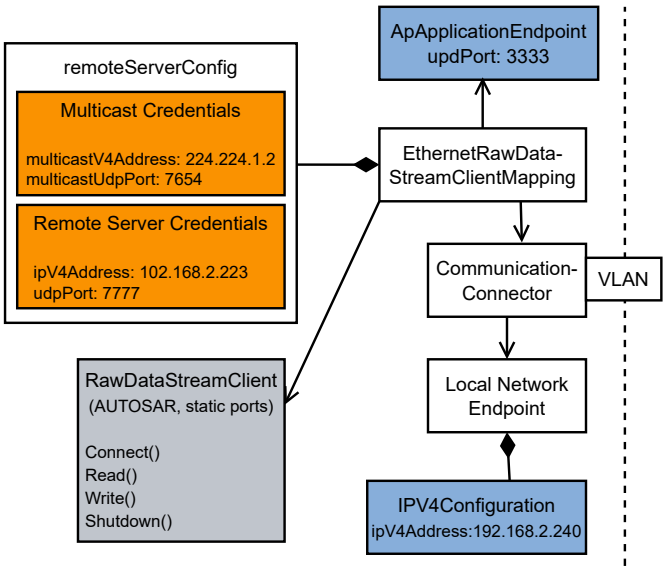


Figure B.41: Scenario where only the raw data stream client is fully formalized inside an AUTOSAR model

B.9.4 Only Server is formalized

The case that **only** the server is formalized (and the client does not appear in the form of an `EthernetRawDataStreamClientMapping`) can be represented by a subset of the modeling described in Chapter B.9.2 as well as the model sketched in Listing B.23.

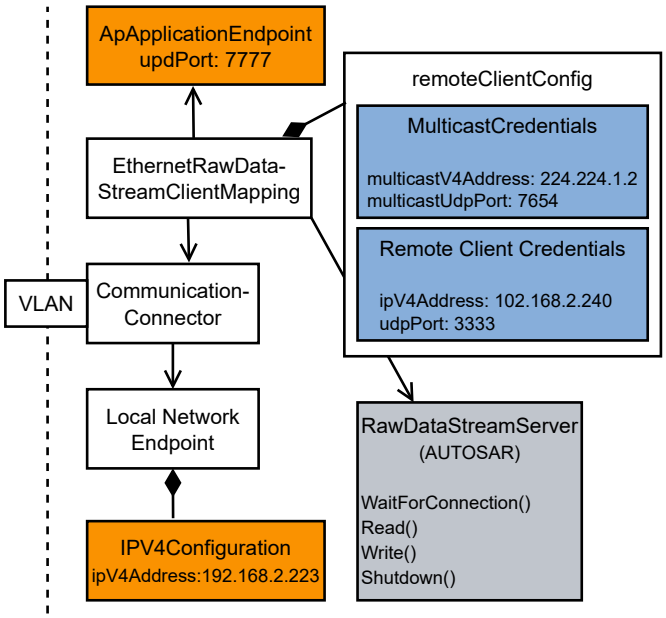


Figure B.42: Scenario where only the raw data stream server is fully formalized inside an AUTOSAR model

B.10 Signal/Service Translation

B.10.1 Signal/Service Mapping

B.10.1.1 Map Event to Signal

In the example sketched in Figure B.43 the *TestEvent* in the *TestServiceInterface* is of type *struct1* that consists of a primitive element *a* and struct *b*. The struct *b* consists of the primitive elements *x*, *y* and *z*.

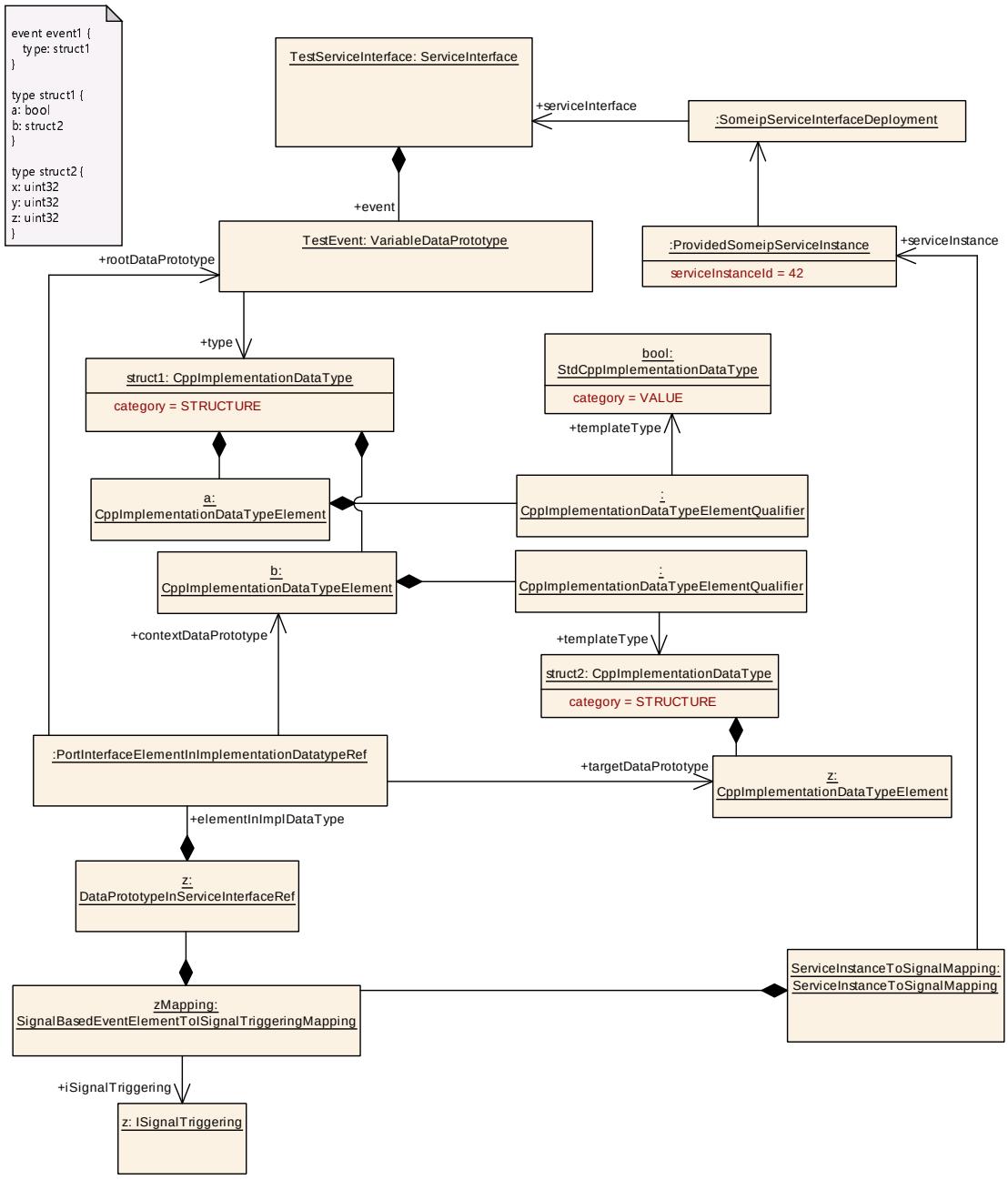


Figure B.43: Example for a mapping of event content to a Signal

Several SignalBasedEventElementToISignalTriggeringMappings are used to map the primitive DataPrototypes *a*, *x*, *y* and *z* to ISignalTriggerings of ISignals located in the ISignalGroup representing the *TestEvent*.

The example shows the mapping of *z* to the ISignalTriggering. Here, the elementInImplDatatype role of DataPrototypeInServiceInterfaceRef is used to refer to the targetDataPrototype since it refers to the internal structure of an AutosarDataPrototype which is typed by a CppImplementationDataType. The context of this reference is defined by the *TestEvent* and struct *b*.

B.10.1.2 Map Field to Signal

In the example sketched in Figure B.44 the *testField* in the *testServiceInterface* is of type *struct1* that consists of the primitive elements *a* and *b*. The *testField* defines a notifier and a setter method.

One SignalBasedFieldToISignalTriggeringMapping maps the *TestField* to ISignalTriggerings for the Setter-Call and Setter-Return.

Here the dataPrototype role of DataPrototypeInServiceInterfaceRef is used to refer to the targetDataPrototype (field) according to the rules defined in [TPS_MANI_01136] and [TPS_MANI_01137].

Additional SignalBasedFieldToISignalTriggeringMappings are necessary to map the field notifier to the corresponding ISignalTriggerings.

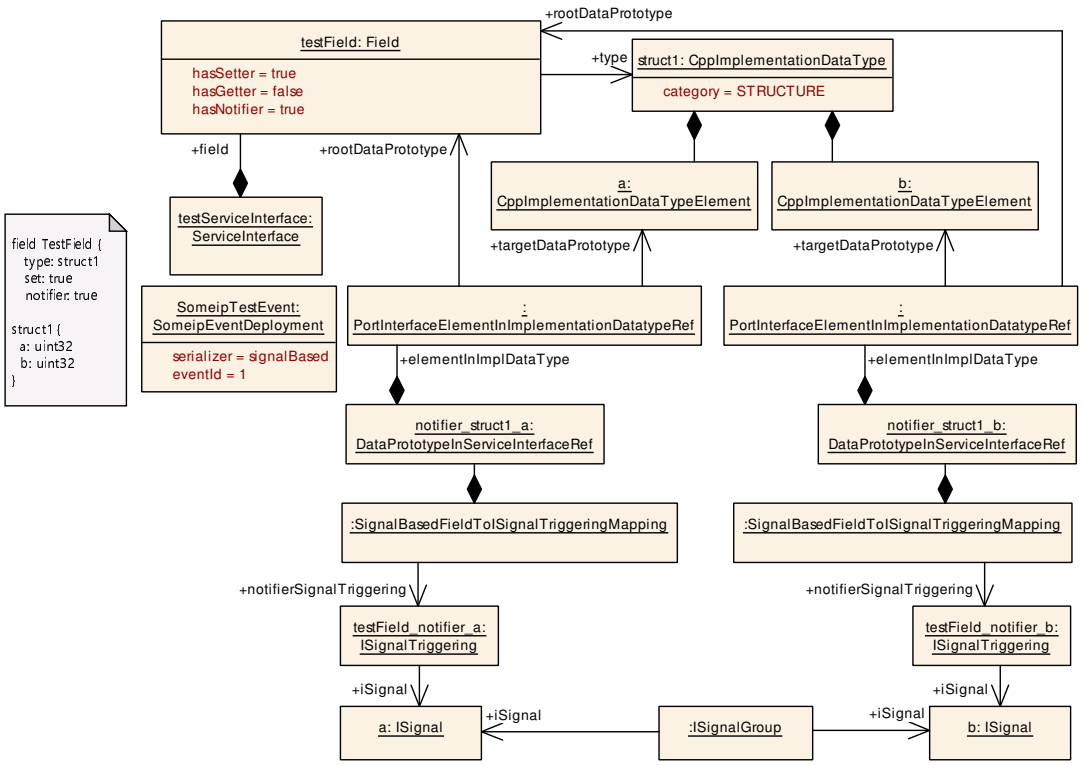


Figure B.44: Example for a mapping of a field to Signals

The primitive DataPrototypes *a* and *b* are mapped by additional SignalBasedFieldToISignalTriggeringMappings to ISignalTriggerings of ISignals located in the ISignalGroup.

Here, the elementInImplDatatype role of DataPrototypeInServiceInterfaceRef is used to refer to the targetDataPrototype since it refers to the internal structure of an AutosarDataPrototype which is typed by a CppImplementationDataType. The context of this reference is defined by the *testField*.

B.10.2 Signal/Service Translation Behavior

B.10.2.1 Optional Structure Element in the context of Signal/Service Translation

Figure B.45 illustrates the usage of update-bits in the source ISignalIPdu as the cause for the definition of isOptional members of the event.

The left part of Figure B.45 shows that the *signal group x1*, *signal y*, and *signal z* have update-bits defined and thus the resulting event definition for the ServiceInterface needs to respect this. In this example the event *xyz* is defined as an ApplicationRecordDataType containing 3 ApplicationRecordElements where each element has the attribute isOptional set to true.

The right part of Figure B.45 takes one specific ISignalIPdu instance with only the *updateBit_y* set to 1, thus only the *signal y* shall be considered as updated. Therefore the resulting event notification contains an event *xyz* where only the member *y* is defined. The other two members of *xyz* are not set in this example.

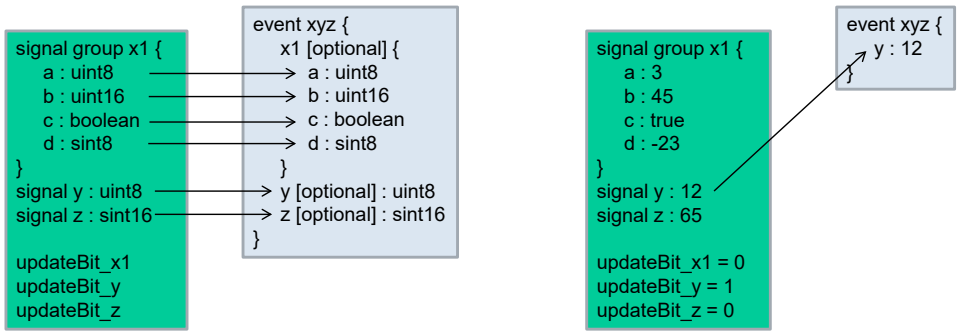


Figure B.45: Example of update-bits for signal-service-translation

B.10.2.2 Optional Structure Element in the context of Service/Signal Translation

Figure B.46 illustrates the usage of isOptional elements in the event / field definition as the cause for the definition of update-bits in the target ISignalIPdu.

The left part of Figure B.46 shows that the *event klm* has two optional members (*k* and *l*) and one mandatory member (*m*). These are mapped into an `ISignalIPdu` with two update-bits (*updateBit_k* and *updateBit_l*).

The right part of Figure B.46 takes one specific *event klm* instance with values defined for *l* (which is optional but present) and *m* (which is not optional and therefore has to be present). There is no value defined for the optional member *k*. The resulting `ISignal-IPdu` takes the values for *l* and *m* and sets the update-bits accordingly (*updateBit_k=0* and *updateBit_l=1*). The values for the not-set optional members stay unchanged.

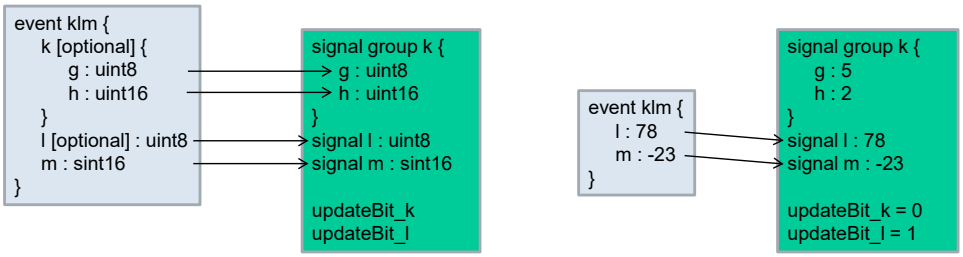


Figure B.46: Example of update-bits for `service-signal-translation`

B.10.2.3 Translation from different Sources

B.10.2.3.1 Signal/Service Translation from different Sources

The example in Figure B.47 shows a setup where several sources of information are coming from `signal-service-translation`s.

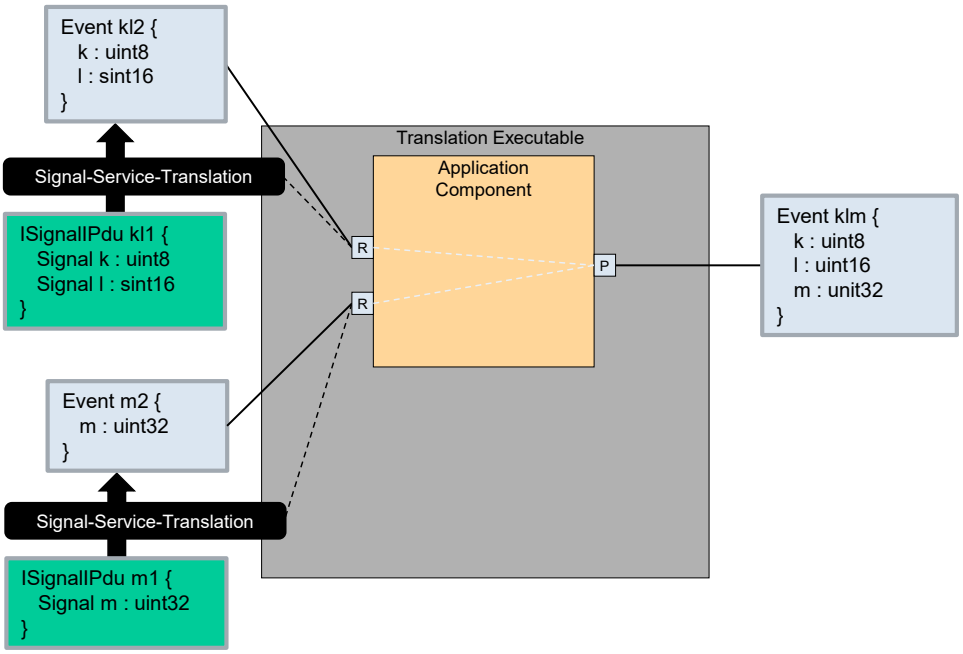


Figure B.47: Example `signal-service-translation` and forwarding from several sources

As the source information is accessed using ara::com APIs and the reception semantics is implementing a queue for each input source, the application component is responsible to fetch the data from the various sources and to decide which input data shall be put to the output event *klm*. As the output event *klm* does not have any optional members defined, it is required to provide all three members *k*, *l*, and *m* with valid values for each instance of *klm*.

B.10.2.3.2 ServiceSignalTranslation from different Sources

Figure B.48 shows an example for several sources coming from service-oriented communication. The task of the application component is to fetch the input information for the source events *x1*, *y*, and *z*. The output event *xyz1* does not have any optional members, also the output `ISignalIPdu` *xyz2* does not use any update bits, thus it is required for the application component to provide the full set of information for *a*, *b*, *y*, and *z* when the output event *xyz1* is produced.

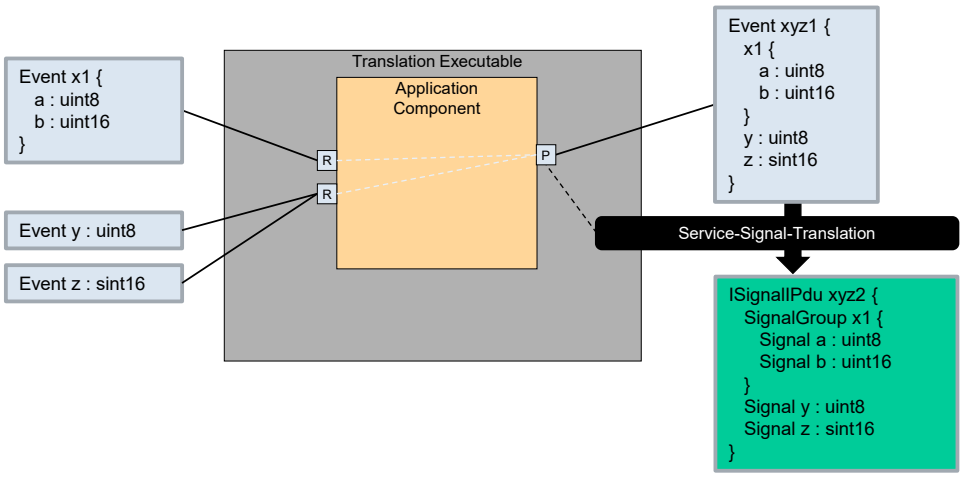


Figure B.48: Example `service-signal-translation` and forwarding from several sources

B.10.2.4 Translation to different Targets

An example of such a setup is illustrated in Figure B.49: The `ISignalIPdu` *xyz* consists of an end-to-end protected `ISignalGroup` *x1* and two not protected `ISignal`s *y* and *z*.

It is essential for the application component to be able to check the end-to-end status for the `event` *x1* which holds the end-to-end protected `ISignalGroup` payload. The `event` *x1* covers just the end-to-end protected part of the `ISignalIPdu` and is handled according to the definitions in Chapter 9.9.1.

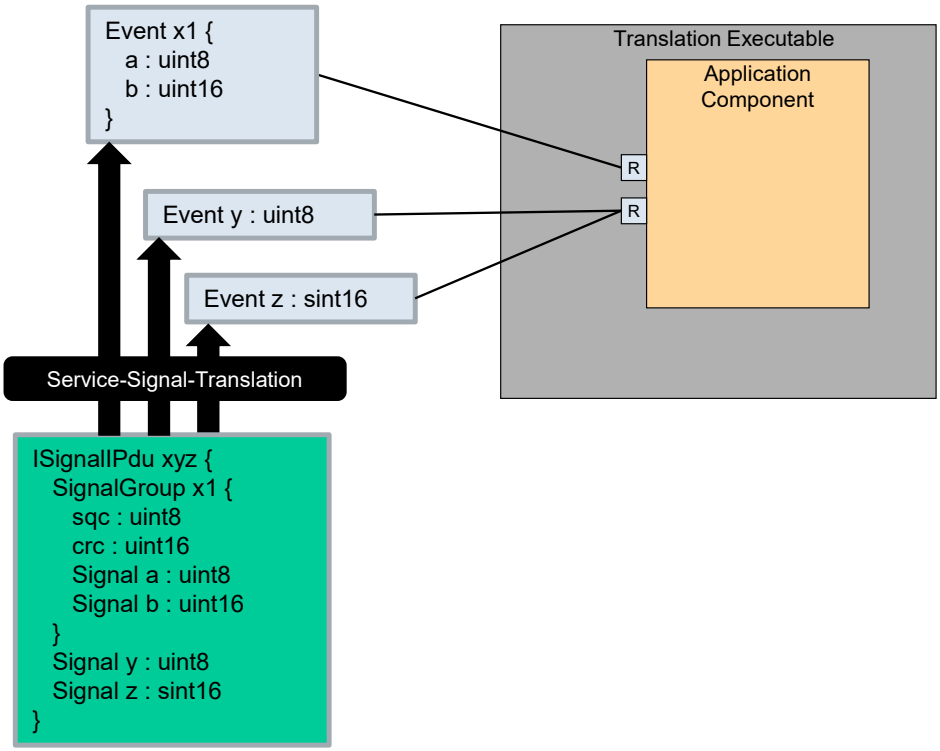


Figure B.49: Example `signal-service-translation` to several targets

The single `event`s *y* and *z* are accessible individually and there is not end-to-end protection involved in their translation.

B.11 Software Cluster Design

B.11.1 Dependencies among Software Cluster Designs

The definition of dependencies among `SoftwareClusterDesign`s is exemplified by the sketch in Figure B.50 (where the orange ellipsis represent `SoftwareClusterDependencyFormula` and the blue ellipsis represent `SoftwareClusterDependencyCompareCondition`) and a corresponding ARXML formalization, see Listing B.24:

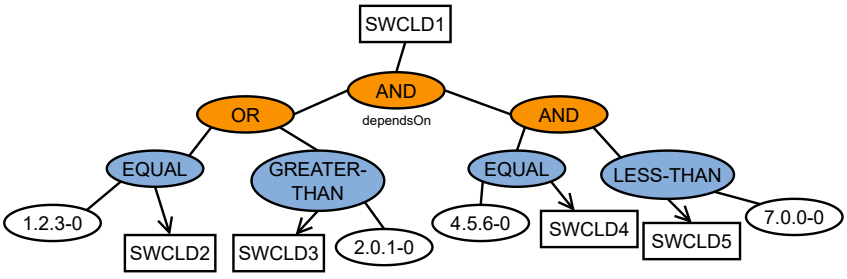


Figure B.50: Example how dependencies among `SoftwareCluster`s can be defined

```
<SOFTWARE-CLUSTER-DESIGN>
```

```
<SHORT-NAME>SWCL1</SHORT-NAME>
<DEPENDS-ON>
  <OPERATOR>LOGICAL-AND</OPERATOR>
  <PARTS>
    <SOFTWARE-CLUSTER-DEPENDENCY-FORMULA>
      <PARTS>
        <SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
          <COMPARE-TYPE>IS-EQUAL</COMPARE-TYPE>
          <SOFTWARE-CLUSTER-DESIGN-REF DEST="SOFTWARE-CLUSTER-DESIGN">/SoftwareClusters/
              SWCL2</SOFTWARE-CLUSTER-DESIGN-REF>
          <VERSION>1.2.3-0</VERSION>
        </SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
        <SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
          <COMPARE-TYPE>IS-GREATER-THAN</COMPARE-TYPE>
          <SOFTWARE-CLUSTER-DESIGN-REF DEST="SOFTWARE-CLUSTER-DESIGN">/SoftwareClusters/
              SWCL3</SOFTWARE-CLUSTER-DESIGN-REF>
          <VERSION>2.0.1-0</VERSION>
        </SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
      </PARTS>
    </SOFTWARE-CLUSTER-DEPENDENCY-FORMULA>
    <SOFTWARE-CLUSTER-DEPENDENCY-FORMULA>
      <OPERATOR>LOGICAL-AND</OPERATOR>
      <PARTS>
        <SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
          <COMPARE-TYPE>IS-EQUAL</COMPARE-TYPE>
          <SOFTWARE-CLUSTER-DESIGN-REF DEST="SOFTWARE-CLUSTER-DESIGN">/SoftwareClusters/
              SWCL4</SOFTWARE-CLUSTER-DESIGN-REF>
          <VERSION>4.5.6-0</VERSION>
        </SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
        <SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
          <COMPARE-TYPE>IS-LESS-THAN</COMPARE-TYPE>
          <SOFTWARE-CLUSTER-DESIGN-REF DEST="SOFTWARE-CLUSTER-DESIGN">/SoftwareClusters/
              SWCL5</SOFTWARE-CLUSTER-DESIGN-REF>
          <VERSION>7.0.0-0</VERSION>
        </SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
      </PARTS>
    </SOFTWARE-CLUSTER-DEPENDENCY-FORMULA>
  </PARTS>
</DEPENDS-ON>
</SOFTWARE-CLUSTER-DESIGN>
```

Listing B.24: Example for the definition of a dependency between individual SoftwareClusterDesigns

B.11.2 Usage of SoftwarePackageStep

The semantics of the references `transfer` and `process` shall be explained along a set of examples.

B.11.2.1 Usage of transfer and process

The first example (as depicted in Figure B.51) assumes a scenario where three SoftwarePackages shall be activated and the three SoftwarePackages are transferred in parallel in the context of one SoftwarePackageStep.

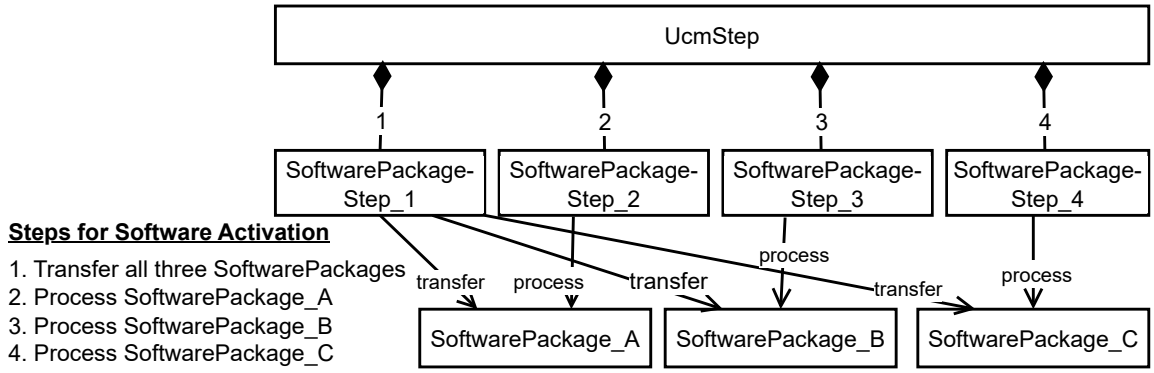


Figure B.51: Example 1 of the configuration of `SoftwarePackageStep`

After that, the three `SoftwarePackage`s are processed one by one in the context of three further `SoftwarePackageStep`s.

The ordering of the `SoftwarePackageStep`s in the context of the enclosing `UcmStep` is depicted explicitly by assigning numerical values to the aggregation as well as naming the respective `SoftwarePackageStep` using the same numerical values.

The second example introduces a scenario where the three `SoftwarePackage`s are transferred in the first `SoftwarePackageStep`. One of the `SoftwarePackage`s is additionally referenced in the role `process` and this indicates that the respective `SoftwarePackage` is streamed.

Because the first `SoftwarePackage` is streamed, only three[1] `SoftwarePackageStep`s are needed to model the example.

The rest of the `SoftwarePackage`s are processed in a dedicated `SoftwarePackageStep`. The example is depicted in Figure B.52.

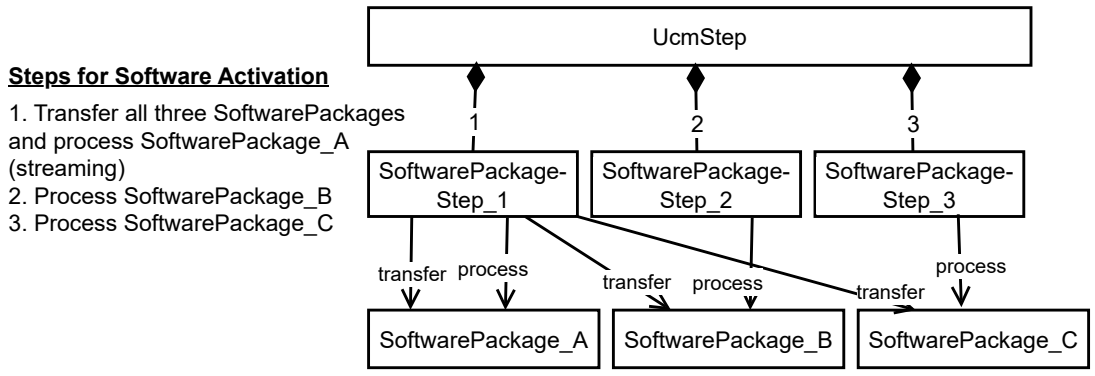


Figure B.52: Example 2 of the configuration of `SoftwarePackageStep`

The third example, depicted in Figure B.53, sketches a scenario where three `SoftwarePackage`s are streamed one after the other.

[1] As opposed to four `SoftwarePackageStep`s required to model the first example.

Consequently, this scenario requires the existence of three SoftwarePackageSteps that each reference the same SoftwarePackage in the roles transfer and process.

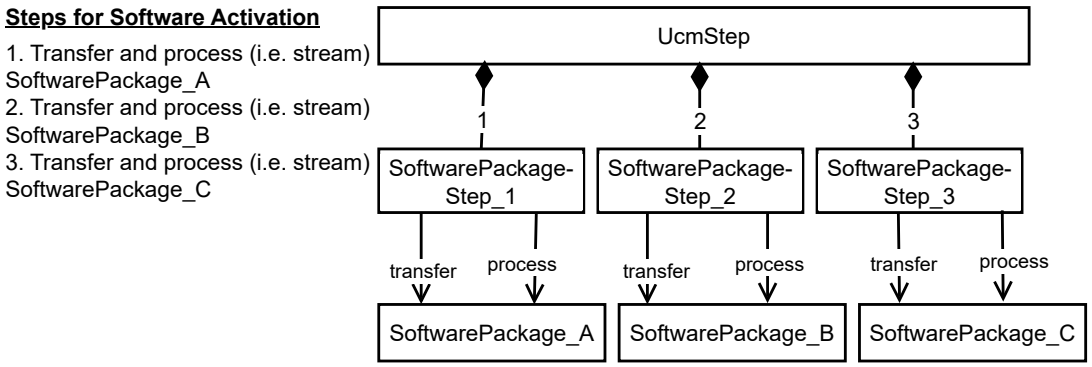


Figure B.53: Example 3 of the configuration of SoftwarePackageStep

B.12 Miscellaneous Examples

B.12.1 Relation of ApApplicationError and ApApplicationErrorDomain

This chapter contains examples for the clarification of the relation between ApApplicationError and ApApplicationErrorDomain.

The following example shows a configuration that fulfills [constr_10574]:

method.shortName	ApApplicationError.shortName	ApApplicationErrorDomain.shortName	ApApplicationErrorDomain.value
Method1	AppError1	ErrorDomain1	42
Method1	AppError2	ErrorDomain2	50
Method1	AppError3	ErrorDomain3	26

Table B.8: Valid example for the definition of ApApplicationErrorDomain

The following example does not fulfill the intended constraint (two different ApApplicationErrorDomains carry the same value in attribute value):

method.shortName	ApApplicationError.shortName	ApApplicationErrorDomain.shortName	ApApplicationErrorDomain.value
Method1	AppError1	ErrorDomain1	42
Method1	AppError2	ErrorDomain2	50
Method1	AppError3	ErrorDomain3	50

Table B.9: Invalid example for the definition of ApApplicationErrorDomain

C Custom Model Extension

The content of this appendix chapter is *informative* in nature and shall **not** be considered as *normative* content.

C.1 Overview

The AUTOSAR meta-model is obviously limited to cover the standardized part of AUTOSAR modeling. This means that the modeling stops in some cases although it is clear that the modeling capability provided by the meta-model is not sufficient to create a viable configuration.

One example for this aspect is the meta-class `ProvidedUserDefinedService-Instance`. AUTOSAR intentionally does nothing but provide this meta-class as a basis for custom additions that configure custom transport for services.

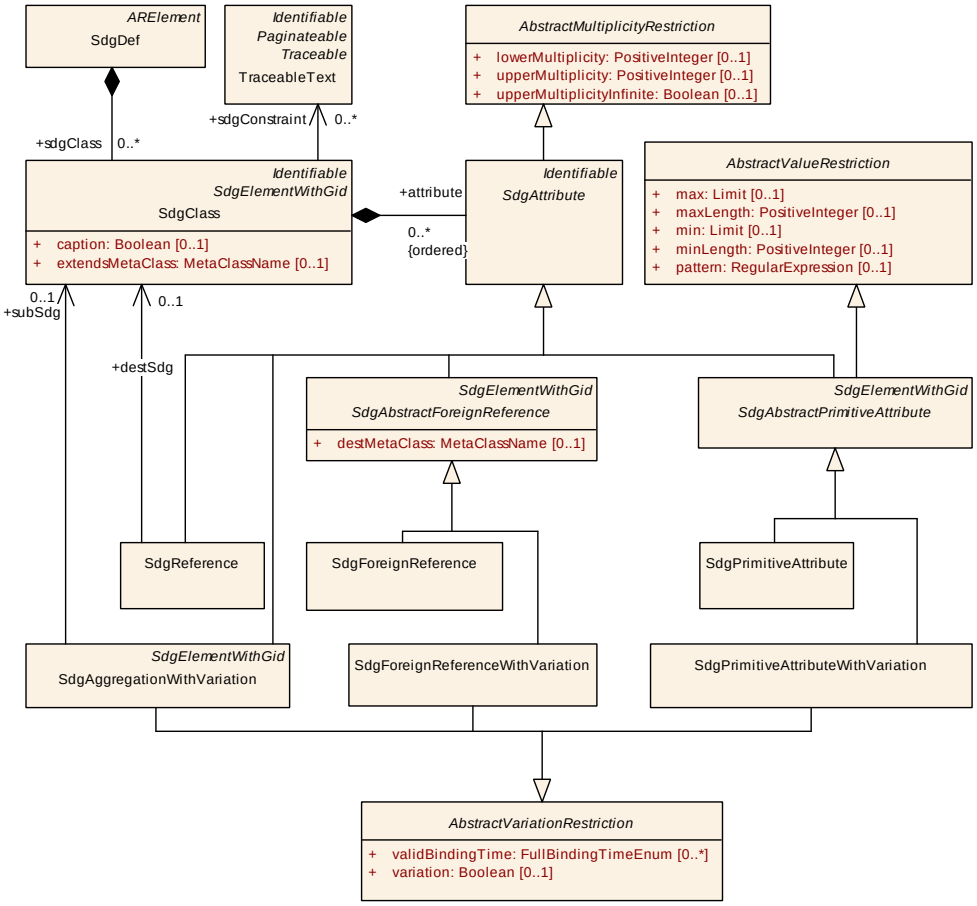


Figure C.1: Modeling of the custom model extensions

The necessary details that are required in order to actually work with a user-defined service instance are not mentioned because there are too many possibilities of how such a `ProvidedUserDefinedServiceInstance` could be modeled that AUTOSAR is simply unable to cover them all.

The typical solution for such a problem is to rely on the definition of *special data groups*, formalized as `Sdg`. However, the direct usage of an `Sdg` with in a project comes with the risk that the `Sdg` is used slightly different in certain parts of the project.

Class	**Sdg**			
Note	Sdg (SpecialDataGroup) is a generic model which can be used to keep arbitrary information which is not explicitly modeled in the meta-model. Sdg can have various contents as defined by sdgContentsType. Special Data should only be used moderately since all elements should be defined in the meta-model. Thereby SDG should be considered as a temporary solution when no explicit model is available. If an sdg Caption is available, it is possible to establish a reference to the sdg structure.			
Base	*ARObject*			
Aggregated by	AdminData.sdg, BuildActionEnvironment.sdg, BuildActionInvocator.sdg, BuildActionIoElement.sdg, File InfoComment.sdg, RptHook.sdg, SdgContents.sdg, VariationPoint.sdg			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
gid	NameToken	0..1	attr	This attributes specifies an identifier. Gid comes from the SGML/XML-Term "Generic Identifier" which is the element name in XML. The role of this attribute is the same as the name of an XML - element. **Tags:** xml.attribute=true
sdgCaption	SdgCaption	0..1	aggr	This aggregation allows to assign the properties of Identifiable to the sdg. By this, a shortName etc. can be assigned to the Sdg. **Stereotypes:** atpIdentityContributor **Tags:** xml.sequenceOffset=20
sdgContents Type	SdgContents	0..1	aggr	This is the content of the Sdg. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

Table C.1: Sdg

It would therefore be good if there were a way to describe in terms of the AUTOSAR meta-model how a `Sdg` is supposed to be used.

Fortunately, this is possible by means of a corner of the meta-model that had been created for exactly this purpose: `SdgClass` aggregates abstract class `SdgAttribute` that in turn inherits to a bunch of different sub-classes.

Class	**SdgClass**			
Note	An SdgClass specifies the name and structure of the SDG that may be used to store proprietary data in an AUTOSAR model. The SdgClass is similar to an UML stereotype.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgElementWithGid*			
Aggregated by	SdgDef.sdgClass			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
attribute (ordered)	SdgAttribute	*	aggr	Defintion of the structure of the Sdg **Tags:** xml.sequenceOffset=30

▽

△

Class	**SdgClass**			
caption	Boolean	0..1	attr	Specifies if a caption is required. Note: only Sdgs that have a caption can be referenced **Tags:** xml.sequenceOffset=20
extendsMeta Class	MetaClassName	0..1	attr	The AUTOSAR Meta-Class that may be extended by this SdgClass. **Tags:** xml.sequenceOffset=10
sdgConstraint	TraceableText	*	ref	Semantic constraints that restrict the structure of the special data group. **Tags:** atp.Splitkey=sdgConstraint xml.sequenceOffset=40

Table C.2: SdgClass

Class	***SdgAttribute*** (abstract)			
Note	Describes the attributes of an Sdg.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*SdgAbstractForeignReference*, *SdgAbstractPrimitiveAttribute*, SdgAggregationWithVariation, Sdg Reference			
Aggregated by	SdgClass.attribute			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table C.3: SdgAttribute

C.2 Custom Attribute Definition

C.2.1 Custom Primitive Attribute Definition

In other words, `SdgClass` and `SdgAttribute` mimic the pattern found in the meta-model itself: meta-classes have attributes of different kinds.

With this mechanism it is possible to extend meta-class `ProvidedUserDefined-ServiceInstance` in order to e.g. add the ability to describe an instance Id. This chapter contains a comprehensive description of how the extension mechanism can be used to implement the instance Id.

The definition starts with an `SdgDef` that aggregates an `SdgClass` with the shortName `ProvidedUserDefinedServiceInstance`. Attribute `extendsMetaClass` names meta-class `ProvidedUserDefinedServiceInstance` as the subject to extension.

The extension itself is modeled by an `SdgPrimitiveAttribute` named instanceId that is mandatory for the implementation of the user-defined service and thus has lower and upper multiplicity set to 1. The supported value interval ranges from 0..4294967295.

Class	**SdgDef**			
Note	A SdgDef groups several SdgClasses which belong to the same extension. The concept of an SdgDef is similiar to an UML Profile. **Tags:** atp.recommendedPackage=SdgDefs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sdgClass	SdgClass	*	aggr	The owned sdgClasses which define the structure of the Sdgs **Tags:** atp.Splitkey=sdgClass, sdgClass.shortName xml.namePlural=SDG-CLASSES

Table C.4: SdgDef

Class	**SdgPrimitiveAttribute**			
Note	Describes primitive special data attributes without variation. This class accepts a special data "sd" attribute.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *AbstractValueRestriction*, *Identifiable*, *Multilanguage Referrable*, *Referrable*, *SdgAbstractPrimitiveAttribute*, *SdgAttribute*, *SdgElementWithGid*			
Aggregated by	SdgClass.attribute			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table C.5: SdgPrimitiveAttribute

```
<SDG-DEF>
  <SHORT-NAME>InstanceExtensions</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>ProvidedUserDefinedServiceInstance</SHORT-NAME>
      <GID>acme:instanceExtensions</GID>
      <EXTENDS-META-CLASS>ProvidedUserDefinedServiceInstance</EXTENDS-META-CLASS>
      <ATTRIBUTES>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>instanceId</SHORT-NAME>
          <CATEGORY>INTEGER</CATEGORY>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:instanceId</GID>
          <MAX>4294967295</MAX>
          <MIN>0</MIN>
        </SDG-PRIMITIVE-ATTRIBUTE>
        </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
```

Listing C.1: Example for the definition of a custom service instance id via `SdgClass`

Please note the definition of `gid` with value “acme:instanceExtensions” on the level of the `SdgPrimitiveAttribute` and the `gid` with value “acme:instanceId” on the level of the `SdgAttribute`.

The usage of the extension is summarized below. Note the usage of the `gid` that reflects the definition in the `SdgClass` and `SdgPrimitiveAttribute`.

Please note further that the definition of the SdgPrimitiveAttribute defines the "data type" of the

```
<PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
  <SHORT-NAME>UDSI</SHORT-NAME>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="acme:instanceExtensions">
        <SD GID="acme:instanceId">42</SD>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
</PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
```

Listing C.2: Example for the specification of the value of a custom service instance id

C.2.2 Custom Complex Attribute Definition

Other extensions, e.g. using SdgAggregationWithVariation can be used to implement the aggregation of a complex attribute (that in turn may own primitive attributes or references).

In this case it makes sense to put the role of the aggregation into the value of SdgAggregationWithVariation.gid.

Note that the SdgAggregationWithVariation doesn't aggregate further elements but refers to an SdgClass that in turn contains the attributes and references.

Class	**SdgAggregationWithVariation**			
Note	Describes that the Sdg may contain another Sdg. The gid of the nested Sdg is defined by subSdg. Represents 'sdg'.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *AbstractVariationRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgAttribute*, *SdgElementWithGid*			
Aggregated by	SdgClass.attribute			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
subSdg	SdgClass	0..1	ref	Supported sub Sdg Class **Tags:** atp.Splitkey=subSdg

Table C.6: SdgAggregationWithVariation

```
<SDG-DEF>
  <SHORT-NAME>DepExt</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>UserDefinedServiceInterfaceDeployment</SHORT-NAME>
      <GID>acme:deploymentExtensions</GID>
      <EXTENDS-META-CLASS>UserDefinedServiceInterfaceDeployment</EXTENDS-META-CLASS>
      <ATTRIBUTES>
        <SDG-AGGREGATION-WITH-VARIATION>
          <SHORT-NAME>Version</SHORT-NAME>
          <GID>acme:version</GID>
          <VARIATION>false</VARIATION>
          <SUB-SDG-REF DEST="SDG-CLASS">/CustomME/DepExt/VersionInformation</SUB-SDG-REF>
        </SDG-AGGREGATION-WITH-VARIATION>
      </ATTRIBUTES>
    </SDG-CLASS>
    <SDG-CLASS>
      <SHORT-NAME>VersionInformation</SHORT-NAME>
      <GID>acme:deploymentExtensions</GID>
```

```
      <ATTRIBUTES>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>MajorVersion</SHORT-NAME>
          <CATEGORY>INTEGER</CATEGORY>
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:majorVersion</GID>
          <MAX>4294967295</MAX>
          <MIN>0</MIN>
        </SDG-PRIMITIVE-ATTRIBUTE>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>MinorVersion</SHORT-NAME>
          <CATEGORY>INTEGER</CATEGORY>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:minorVersion</GID>
          <MAX>4294967295</MAX>
          <MIN>0</MIN>
        </SDG-PRIMITIVE-ATTRIBUTE>
      </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
```

Listing C.3: Example for the definition of a complex version information `SdgClass`

The modeling of the complex aggregation is sketched in Listing C.3. An `SdgClass` with the `gid` set to “acme:deploymentExtensions” defines an attribute that is an `SdgAggregationWithVariation` with the `shortName` “Version” that has attribute `gid` set to “acme:version”.

This means that the `SdgAggregationWithVariation` with the `shortName` “Version” is aggregated in the role “version” (derived from the value of the `gid`) at the `SdgClass`.

The `SdgAggregationWithVariation` in turn references another `SdgClass` with the `shortName` set to “VersionInformation”. This `SdgClass` contains two `SdgPrimitiveAttribute`s for carrying the optional major version and the minor version. The value range of both version information is from 0 to 4294967295.

The definition side of this example is sketched in Listing C.3, the respective value side is provided in Listing C.4.

```
<USER-DEFINED-SERVICE-INTERFACE-DEPLOYMENT>
  <SHORT-NAME>UDSID</SHORT-NAME>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="acme:deploymentExtensions">
        <SDG GID="acme:version">
          <SD GID="acme:majorVersion">1</SD>
          <SD GID="acme:minorVersion">0</SD>
        </SDG>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
</USER-DEFINED-SERVICE-INTERFACE-DEPLOYMENT>
```

Listing C.4: Example for the specification of the value of a custom version information

C.3 Custom Foreign Reference Definition

Another aspect of custom modeling is the creation of references to meta-classes derived from `Referrable`. For this purpose, the meta-class `SdgForeignReference` resp. `SdgForeignReferenceWithVariation`.

Class	**SdgForeignReference**			
Note	A reference without variation support that can point to any referrable object in an AUTOSAR Model. This class accepts the special data "Sdx" reference.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgAbstractForeignReference*, *SdgAttribute*, *SdgElementWithGid*			
Aggregated by	SdgClass.attribute			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table C.7: SdgForeignReference

Class	**SdgForeignReferenceWithVariation**			
Note	A reference with variation support that can point to any referrable object in an AUTOSAR Model. This class accepts the special data "Sdxf" reference.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *AbstractVariationRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgAbstractForeignReference*, *SdgAttribute*, *SdgElementWithGid*			
Aggregated by	SdgClass.attribute			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table C.8: SdgForeignReferenceWithVariation

The particle “foreign” in the name of these classes represents a hint that the reference’s target is situated outside the custom modeling “bubble” depicted in Figure C.1[1].

The creation of a reference to another meta-class obviously implies the specification of the role in which the reference shall be used.

In the case of the `SdgForeignReference` the role of the reference may be defined in the attribute `gid`, contributed by the inheritance from meta-class `SdgElementWithGid`.

Class	***SdgElementWithGid*** (abstract)			
Note	A special data group element with gid is an abstract element that shall have a name (gid, "Generic Identifier").			
Base	*ARObject*			
Subclasses	*SdgAbstractForeignReference*, *SdgAbstractPrimitiveAttribute*, SdgAggregationWithVariation, SdgClass			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
gid	NameToken	0..1	attr	Specifies the name that identifies the element.

Table C.9: SdgElementWithGid

In true AUTOSAR fashion, a reference should always announce the intended meta-class to which it refers to. In the case of the `SdgForeignReference`, this information

[1] Of course, this naming is also a hat tip to the meta-class `EcucForeignReferenceDef`

can be explicitly provided by means of the attribute destMetaClass, inherited from SdgAbstractForeignReference.

Class	***SdgAbstractForeignReference*** (abstract)			
Note	An abstract reference that can point to any referrable object in an AUTOSAR Model.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgAttribute*, *SdgElementWithGid*			
Subclasses	SdgForeignReference, SdgForeignReferenceWithVariation			
Aggregated by	SdgClass.attribute			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destMetaClass	MetaClassName	0..1	attr	specifies the destination meta-class of the reference.

Table C.10: SdgAbstractForeignReference

The example created for the explanation of SdgForeignReference assumes that a ProvidedUserDefinedServiceInstance wants to re-use an existing configuration for SOME/IP SD.

For this purpose, a custom extension of the meta-class ProvidedUserDefinedServiceInstance with the ability to refer to at most one SomeipSdServerServiceInstanceConfig is created.

In particular, the extension consists of the definition of aSdgForeignReference with the gid set to the value “acme:sdServerTimeConfig” as the representation of the role of the reference is created.

The nature of the SdgForeignReference defined in this example is determined by means of the value of attribute destMetaClass, in this case “SomeipSdServerServiceInstanceConfig”.

The custom definition of the SdgForeignReference is sketched in Listing C.5.

```
<SDG-DEF>
  <SHORT-NAME>InstanceExtensions</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>ProvidedUserDefinedServiceInstance</SHORT-NAME>
      <GID>acme:instanceExtensions</GID>
      <EXTENDS-META-CLASS>ProvidedUserDefinedServiceInstance</EXTENDS-META-CLASS>
      <ATTRIBUTES>
        <SDG-FOREIGN-REFERENCE>
          <SHORT-NAME>SdServerTimeConfig</SHORT-NAME>
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:sdServerTimeConfig</GID>
          <DEST-META-CLASS>SomeipSdServerServiceInstanceConfig</DEST-META-CLASS>
        </SDG-FOREIGN-REFERENCE>
      </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
```

Listing C.5: Example for the specification of a custom foreign reference

The value side of the example in Listing C.5 can be found in Listing C.6. Note that the formalization of the reference to the respective SomeipSdServerServiceInstanceConfig is implemented by means of the reference SdgContents.sdx,

wrapped into an Sdg where the attribute gid is set to the role of the reference, in this case “acme:sdServerTimeConfig”.

Class	«atpMixed» **SdgContents**			
Note	This meta-class represents the possible contents of a special data group. It can be an arbitrary mix of references, of primitive special data and nested special data groups.			
Base	*ARObject*			
Aggregated by	Sdg.sdgContentsType			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
sd	Sd	0..1	aggr	This is one particular special data element. **Tags:** xml.sequenceOffset=40
sdf	Sdf	0..1	aggr	This is one particular special data element. **Tags:** xml.sequenceOffset=60
sdg	Sdg	0..1	aggr	This aggregation allows to express nested special data groups. By this, any structure can be represented in SpeicalData. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=50
sdx	Referrable	0..1	ref	Reference to any identifiable element. This allows to use Sdg even to establish arbitrary relationships.
sdxf	Referrable	0..1	ref	Additional reference with variant support. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild

Table C.11: SdgContents

```
<PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
  <SHORT-NAME>UDSI</SHORT-NAME>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="acme:instanceExtensions">
        <SDG GID="acme:sdServerTimeConfig">
          <SDX-REF DEST="SOMEIP-SD-SERVER-SERVICE-INSTANCE-CONFIG">/SD/MyConf</SDX-REF>
        </SDG>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
</PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
```

Listing C.6: Example for the specification of the value of a custom foreign reference

C.4 Custom Subclass Configuration

Using the mechanism of custom model extensions it is also possible to mimic the creation of custom “subclasses”.

A possible candidate for the creation of a “subclass” outside the explicitly formalized meta-model could be the ProvidedUserDefinedServiceInstance. The intention could, for example, be to provide a configuration for an IPC-specific “subclass”.

A simple example of how an extension that defines a “subclass” might look like is sketched in Listing C.7.

The specific approach in this case is to define an SdgClass to extend ProvidedUserDefinedServiceInstance and this extension refers - via the aggregation of an SdgAggregationWithVariation - to another SdgClass (in this case with the shortName “IpcProvidedServiceInstanceAttributes”) where the attributes of the custom “subclass” are defined.

On the value side, the definition of the attribute values of the “subclass” are defined in a quite straight-forward manner (see Listing C.8).

Please note that the value side in this example does not reflect the definition side in terms of the usage of values of attribute gid 100%. The existence of the SdgClass with shortName “IpcProvidedServiceInstanceAttributes” is not represented on the value side.

But, on the other hand, the intended semantics of defining an instance identifier in the context of the custom model of a “ProvidedIpcServiceInstance” can be conveyed perfectly without it.

```
<SDG-DEF>
  <SHORT-NAME>SubclassConfiguration</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>ProvidedUserDefinedServiceInstance</SHORT-NAME>
      <GID>acme:providedIpcServiceInstance</GID>
      <EXTENDS-META-CLASS>ProvidedUserDefinedServiceInstance</EXTENDS-META-CLASS>
      <ATTRIBUTES>
        <SDG-AGGREGATION-WITH-VARIATION>
          <SHORT-NAME>IpcSubclass</SHORT-NAME>
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:attributes</GID>
          <VARIATION>false</VARIATION>
          <SUB-SDG-REF DEST="SDG-CLASS">/CustomME/SubclassConfiguration/
              IpcProvidedServiceInstanceAttributes</SUB-SDG-REF>
        </SDG-AGGREGATION-WITH-VARIATION>
      </ATTRIBUTES>
    </SDG-CLASS>
    <SDG-CLASS>
      <SHORT-NAME>IpcProvidedServiceInstanceAttributes</SHORT-NAME>
      <GID>acme:ipcProvidedServiceInstanceAttributes</GID>
      <ATTRIBUTES>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>instanceId</SHORT-NAME>
          <CATEGORY>INTEGER</CATEGORY>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:instanceId</GID>
          <MAX>65535</MAX>
          <MIN>0</MIN>
        </SDG-PRIMITIVE-ATTRIBUTE>
      </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
```

Listing C.7: Example for the specification of a custom “subclass”

And since the point of the whole approach is the creation of a custom modeling anyway, the only relevant condition for the validity of such modeling is that the affected AUTOSAR tools know how to properly parse and interpret the resulting model.

```
<PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
  <SHORT-NAME>ipcProvidedSI</SHORT-NAME>
  <ADMIN-DATA>
```

```
    <SDGS>
      <SDG GID="acme:providedIpcServiceInstance">
        <SDG GID="acme:attributes">
          <SD GID="acme:instanceId">3485</SD>
        </SDG>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
</PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
```

Listing C.8: Example for the specification of a service instance in an IPC "subclass"

C.5 Custom Constraints

Another aspect that can be solved by means of model customizations is the definition of model constraints that go beyond the potentially existing constraints formulated in AUTOSAR.

In the example sketched in Listing C.9 a constraint is formulated for attribute `Executable.minimumTimerGranularity`.

```
<SDG-DEF>
  <SHORT-NAME>ModelConstraint</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>Executable</SHORT-NAME>
      <GID>acme:executableExtensions</GID>
      <EXTENDS-META-CLASS>Executable</EXTENDS-META-CLASS>
      <ATTRIBUTES>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>minimumTimerGranularity</SHORT-NAME>
          <CATEGORY>FLOAT</CATEGORY>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <MAX>0.5</MAX>
          <MIN>0.001</MIN>
        </SDG-PRIMITIVE-ATTRIBUTE>
      </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
```

Listing C.9: Example for the specification of a custom model constraint

On the one hand, the example expresses the expectation that the attribute `Executable.minimumTimerGranularity` shall exist and the second aspect of the constraint is that it shall have a value that's between 0.001 and 0.5.

C.6 Definition of Reference from SdgClass to SdgClass

Another case that could be relevant for a custom model extension is the creation of a reference between two `Sdg` elements on the value side.

Semi-formal support for this scenario can be defined by means of the definition of two `SdgClass` elements where one defines a reference to the other.

A caveat applies. Of course, two SdgClass elements could reference each other by means of their shortName-paths (because they are derived from Identifiable).

But that's not the point, the intended reference on the value side shall exist from one Sdg (that corresponds to one of the SdgClass elements on the definition side) to another Sdg (which corresponds to the other SdgClass element on the definition side).

Sdg itself is not derived from Referrable and therefore does not have a shortName that could be used for reference building purposes.

In order to support the creation of a reference from one Sdg to another a mechanism was created using a reference from an Sdg to an SdgCaption in the role sdx.

In other words, an Sdg may aggregate an SdgCaption in the role sdgCaption. And if it does, it becomes (by extension) a valid target of a reference to this SdgCaption.

Class	**SdgCaption**			
Note	This meta-class represents the caption of a special data group. This allows to have some parts of special data as identifiable.			
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	Sdg.sdgCaption			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
desc	MultiLanguageOverview Paragraph	0..1	aggr	This represents a general but brief (one paragraph) description what the special data in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the special data in question. **Tags:** atp.Splitkey=desc

Table C.12: SdgCaption

Therefore, the SdgClass that represents the Sdg on the target side of the reference shall define attribute sdgCaption and set it to true.

Class	**SdgReference**			
Note	Describes an attribute of a SdgClass which is used on the definition side to model a reference from one Sdg to another Sdg on the value side.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgAttribute*			
Aggregated by	SdgClass.attribute			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destSdg	SdgClass	0..1	ref	Refers to a SdgClass which is used on the definition side to model the destination type of the referenced Sdg. On the value side the reference is realized by means of the originating Sdg defining an sdgx attribute which refers to the sdgCaption of the referenced Sdg. **Tags:** atp.Splitkey=destSdg

Table C.13: SdgReference

The SdgClass on the source side of the reference shall define an attribute that is actually an SdgReference to the SdgClass that represents the target Sdg on the definition side by means of the reference in the role destSdg.

As a first step, the following model fragment defines a custom model extension of an IPC channel[2] (note that attribute caption is set to true):

```
<SDG-DEF>
  <SHORT-NAME>IpcChannels</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>IpcChannel</SHORT-NAME>
      <GID>acme:ipcChannels</GID>
      <CAPTION>true</CAPTION>
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
```

Listing C.10: Example for the specification of a custom model extension that represents an IPC channel

This IPC channel (named “MyIpcChannel”) shall be referenced from the definition of an SdgClass that is supposed to extend the ProvidedUserDefinedService-Instance:

```
<SDG-DEF>
  <SHORT-NAME>InstanceExtensions</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>ProvidedUserDefinedServiceInstance</SHORT-NAME>
      <GID>acme:instanceExtensions</GID>
      <EXTENDS-META-CLASS>ProvidedUserDefinedServiceInstance</EXTENDS-META-CLASS>
      <CAPTION>false</CAPTION>
      <ATTRIBUTES>
        <SDG-REFERENCE>
          <SHORT-NAME>IpcChannelRef</SHORT-NAME>
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <DEST-SDG-REF DEST="SDG-CLASS">/SdgClasses/IpcChannels/IpcChannel</DEST-SDG-REF>
        </SDG-REFERENCE>
      </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
```

Listing C.11: Example for the specification of a custom model extension for a ProvidedUserDefinedServiceInstance with a reference to an IPC channel

The model on the value side where the ProvidedUserDefinedServiceInstance references the IPC channel is sketched by the following model fragment:

```
<AR-PACKAGE>
  <SHORT-NAME>ServiceInstances</SHORT-NAME>
  <ELEMENTS>
    <PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
      <SHORT-NAME>MyService</SHORT-NAME>
      <ADMIN-DATA>
        <SDGS>
          <SDG GID="acme:ipcChannelRef">
            <SDX-REF DEST="SDG-CAPTION">/IpcChannels/IpcChannel1</SDX-REF>
          </SDG>
        </SDGS>
      </ADMIN-DATA>
    </PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
  </ELEMENTS>
</AR-PACKAGE>
<AR-PACKAGE>
```

[2]Please note that this example represents an incomplete model that does not care about details of the actual configuration of the hypothetical IPC Channel and is entirely focused on the referencing topic.

```
  <SHORT-NAME>IpcChannels</SHORT-NAME>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="acme:ipcChannel1">
        <SDG-CAPTION>
          <SHORT-NAME>IpcChannel1</SHORT-NAME>
        </SDG-CAPTION>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
</AR-PACKAGE>
```

Listing C.12: Example for the specification of reference between `Sdg`

D General Modeling

This chapter has been created to explain model elements that are not directly related to specific design or target configuration usage but have a more general scope. In other words, this chapter describes the structure and usage of some widely reusable modeling content.

D.1 Reference to a DataPrototype in a PortInterface

D.1.1 Reference to the inside of an `ApplicationDataType`

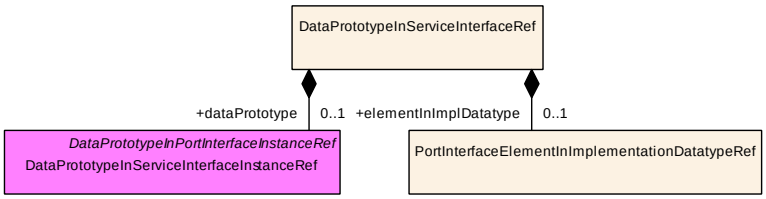


Figure D.1: Modeling of `DataPrototypeInServiceInterfaceRef`

Class	**DataPrototypeInServiceInterfaceRef**			
Note	This meta-class represents the ability to refer to an AUTOSAR DataPrototype in the context of a Service Interface. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	SignalBasedEventElementToISignalTriggeringMapping.dataPrototypeInServiceInterfaceRef, SignalBasedFieldToISignalTriggeringMapping.dataPrototypeInServiceInterfaceRef, SomeipDataPrototypeTransformationProps.dataPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataPrototype	DataPrototype	0..1	iref	This element represents the ability to: • refer to a DataPrototype in the context of a Service Interface. • refer to the internal structure of a DataPrototype in which is typed by an ApplicationDatatype the context of a ServiceInterface. **Tags:** atp.Splitkey=dataPrototype, dataPrototype.root DataPrototype, dataPrototype.contextDataPrototype, data Prototype.targetDataPrototype **InstanceRef implemented by:** DataPrototypeInService InterfaceInstanceRef
elementInImpl Datatype	PortInterfaceElementIn Implementation DatatypeRef	0..1	aggr	This element represents the ability to refer to the internal structure of an AutosarDataPrototype which is typed by an ImplementationDatatype in the context of a Service Interface. **Tags:** atp.Splitkey=elementInImplDatatype

Table D.1: DataPrototypeInServiceInterfaceRef

Please note that the modeling of the reference to a DataPrototype in the context of a PortInterface can only be executed as the abstract template for concrete specializations because the abstract meta-class PortInterface does not aggregate a DataPrototype directly.

The abstract modeling of meta-class DataPrototypeInPortInterface-InstanceRef is depicted in Figure D.2.

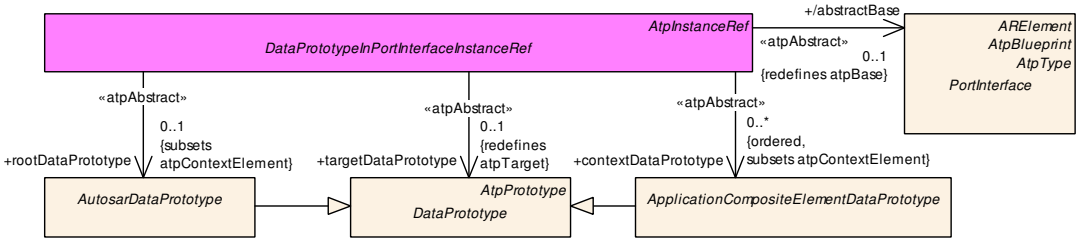


Figure D.2: Modeling of DataPrototypeInPortInterfaceInstanceRef

Class	***DataPrototypeInPortInterfaceInstanceRef*** (abstract)			
Note	This meta-class represents the ability to: • refer to a DataPrototype in the context of a PortInterface. • refer to the internal structure of a DataPrototype which is typed by an ApplicationDatatype in the context of a PortInterface.			
Base	*ARObject*, *AtpInstanceRef*			
Subclasses	DataPrototypeInServiceInterfaceInstanceRef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
abstractBase	PortInterface	0..1	ref	**Stereotypes:** atpAbstract
contextData Prototype (ordered)	ApplicationComposite ElementDataPrototype	*	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=20
rootData Prototype	AutosarDataPrototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=10
targetData Prototype	DataPrototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=30

Table D.2: DataPrototypeInPortInterfaceInstanceRef

The concrete specialization for the aggregation of a DataPrototype in the concrete ServiceInterface is depicted in Figure D.3.

The meta-class DataPrototypeInServiceInterfaceInstanceRef inherits from DataPrototypeInPortInterfaceInstanceRef.

The individual references modeled in the context of DataPrototypeInService-InterfaceInstanceRef specialize the abstract structure defined in the context of DataPrototypeInPortInterfaceInstanceRef

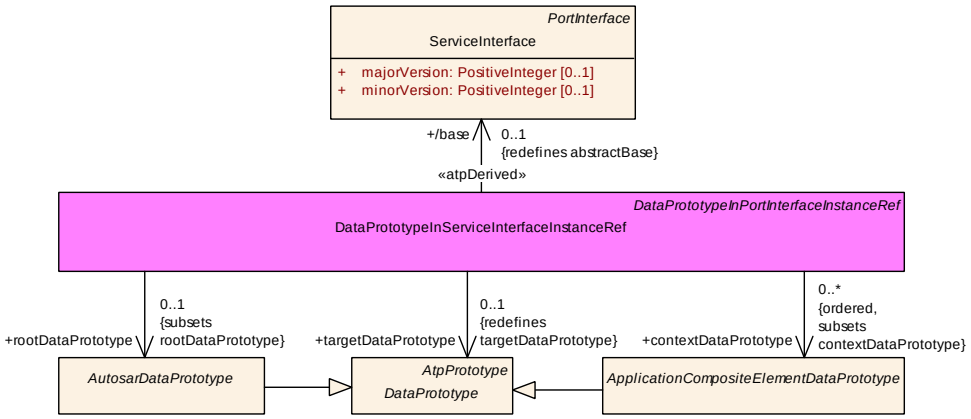


Figure D.3: Modeling of `DataPrototypeInServiceInterfaceInstanceRef`

Class	**DataPrototypeInServiceInterfaceInstanceRef**			
Note	This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpInstanceRef*, *DataPrototypeInPortInterfaceInstanceRef*			
Aggregated by	DataPrototypeInPortInterfaceRef.dataPrototypeInServiceInterface, DataPrototypeInServiceInterface Ref.dataPrototype, SignalBasedFireAndForgetMethodToISignalTriggeringMapping.dataPrototypeIn MethodArgumentInstanceRef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	ServiceInterface	0..1	ref	**Stereotypes:** atpDerived
contextData Prototype (ordered)	ApplicationComposite ElementDataPrototype	*	ref	**Tags:** xml.sequenceOffset=20
rootData Prototype	AutosarDataPrototype	0..1	ref	**Tags:** xml.sequenceOffset=10
targetData Prototype	DataPrototype	0..1	ref	**Tags:** xml.sequenceOffset=30

Table D.3: DataPrototypeInServiceInterfaceInstanceRef

D.1.2 Reference to the inside of a `CppImplementationDataType`

Please note that the modeling of instanceRef-like references into the internals of a `CppImplementationDataType` differs from the way how internals of an `ImplementationDataType` could be referenced.

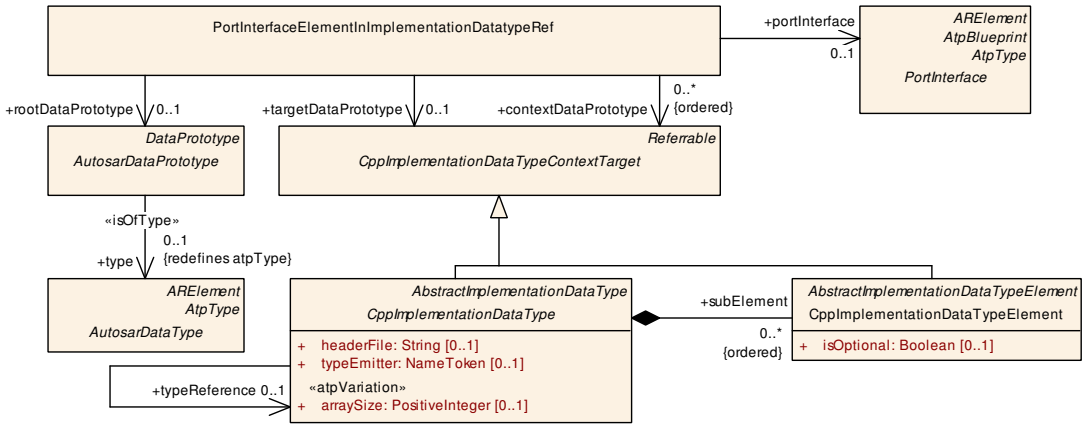


Figure D.4: Modeling of `PortInterfaceElementInImplementationDatatypeRef`

In particular, references to context elements can be directed to `ImplementationDataTypeElement` because both arrays and structures are modeled by means of `ImplementationDataTypeElement`.

This approach has changed with the advent of `CppImplementationDataType` and therefore the same approach is not possible for `CppImplementationDataTypeElement`.

In the case of `CppImplementationDataType`, both `CppImplementationDataTypeElement` and `CppImplementationDataType` can become the target of a context reference. And since the context reference is supposed to be ordered it is simply not possible to straight up model two context references, one for `CppImplementationDataType` and one for `CppImplementationDataTypeElement`.

Instead, it is necessary to introduce an abstract base class named `CppImplementationDataTypeContextTarget` for both `CppImplementationDataType` and `CppImplementationDataTypeElement` and then **direct context references at the abstract base class**.

Class	**PortInterfaceElementInImplementationDatatypeRef**			
Note	This meta-class represents the ability to refer to the internal structure of an AutosarDataPrototype which is typed by an implementationDatatype in the context of a PortInterface. In other words, this meta-class shall not be used to model a reference to the **AutosarDataPrototype as a target itself**, **even** if the AutosarDataPrototype is typed by an ImplementationDataType **and even** if that ImplementationDataType represents a composite data type. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	DataPrototypeInServiceInterfaceRef.elementInImplDatatype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contextData Prototype (ordered)	CppImplementation DataTypeContextTarget	*	ref	This is a context in case there are subelements with explicit types. The reference has to be ordered to properly reflect the nested structure.

△

Class	**PortInterfaceElementInImplementationDatatypeRef**			
portInterface	PortInterface	0..1	ref	This is the PortInterface that contains the rootData Prototype. **Tags:** atp.Splitkey=portInterface
rootData Prototype	AutosarDataPrototype	0..1	ref	This rootDataPrototype defines the AutosarDataPrototype in which the target can be found. **Tags:** atp.Splitkey=rootDataPrototype
targetData Prototype	CppImplementation DataTypeContextTarget	0..1	ref	This is the target reference to a subElement that is defined inside of the rootDataPrototype. **Tags:** atp.Splitkey=targetDataPrototype

Table D.4: PortInterfaceElementInImplementationDatatypeRef

Class	***CppImplementationDataTypeContextTarget*** (abstract)			
Note	This meta-class has the ability to serve as the context in instanceRef-like modeling for Cpp ImplementationDataType and CppImplementationDataTypeElement This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Referrable*			
Subclasses	*CppImplementationDataType*, CppImplementationDataTypeElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table D.5: CppImplementationDataTypeContextTarget

D.2 Reference to a AutosarDataPrototype in an Executable

The creation of the meta-model for creating a reference to an `AutosarDataPrototype` in the context of an `Executable` is executed in a two-step approach where first an abstract structure of the reference is created.

The abstract structure is the basis for the refinement with respect to specific roles of `AutosarDataPrototype`s.

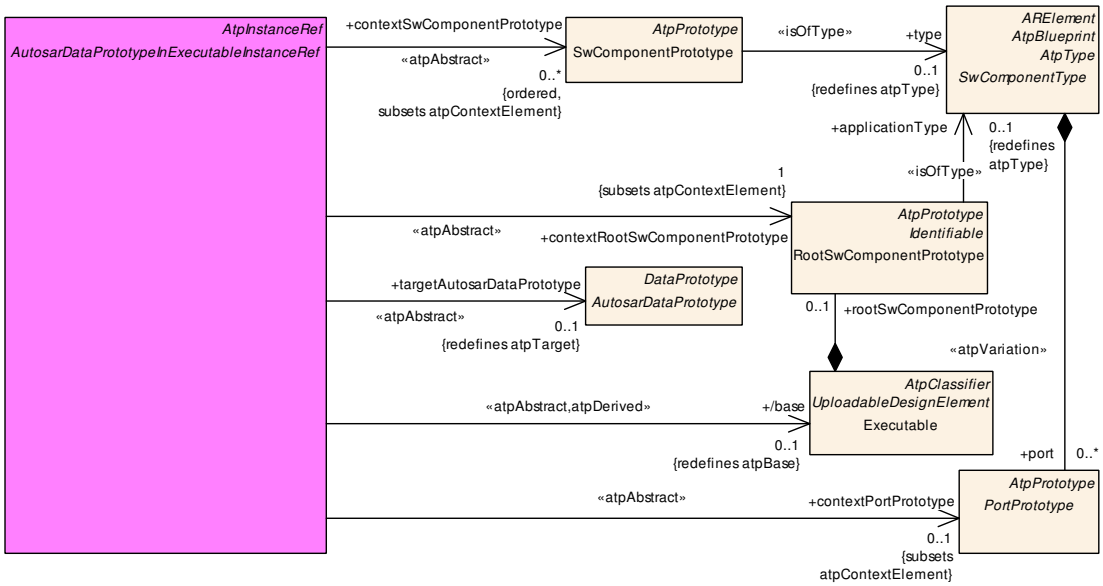


Figure D.5: Modeling of abstract `AutosarDataPrototypeInExecutableInstanceRef`

Class	***AutosarDataPrototypeInExecutableInstanceRef*** (abstract)			
Note	This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpInstanceRef*			
Subclasses	EventInExecutableInstanceRef, FieldInExecutableInstanceRef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpAbstract; atpDerived
contextPort Prototype	PortPrototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=30
contextRootSw Component Prototype	RootSwComponent Prototype	1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=10
contextSw Component Prototype (ordered)	SwComponent Prototype	*	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=20
targetAutosar DataPrototype	AutosarDataPrototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=40

Table D.6: AutosarDataPrototypeInExecutableInstanceRef

Two specializations of `AutosarDataPrototypeInExecutableInstanceRef` exist:

- `EventInExecutableInstanceRef`
- `FieldInExecutableInstanceRef`

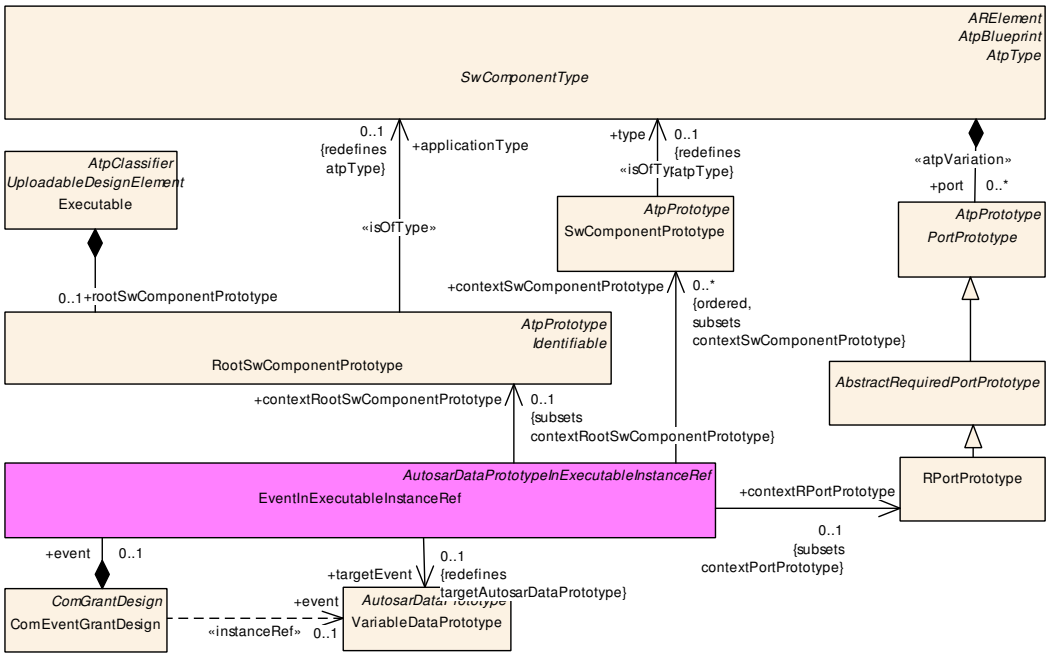


Figure D.6: Modeling of concrete `EventInExecutableInstanceRef` derived from `AutosarDataPrototypeInExecutableInstanceRef`

Class	**EventInExecutableInstanceRef**			
Note	This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpInstanceRef*, *AutosarDataPrototypeInExecutableInstanceRef*			
Aggregated by	ComEventGrantDesign.event			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Tags:** xml.sequenceOffset=10
contextRPort Prototype	RPortPrototype	0..1	ref	**Tags:** xml.sequenceOffset=30
contextSw Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** xml.sequenceOffset=20
targetEvent	VariableDataPrototype	0..1	ref	**Tags:** xml.sequenceOffset=40

Table D.7: EventInExecutableInstanceRef

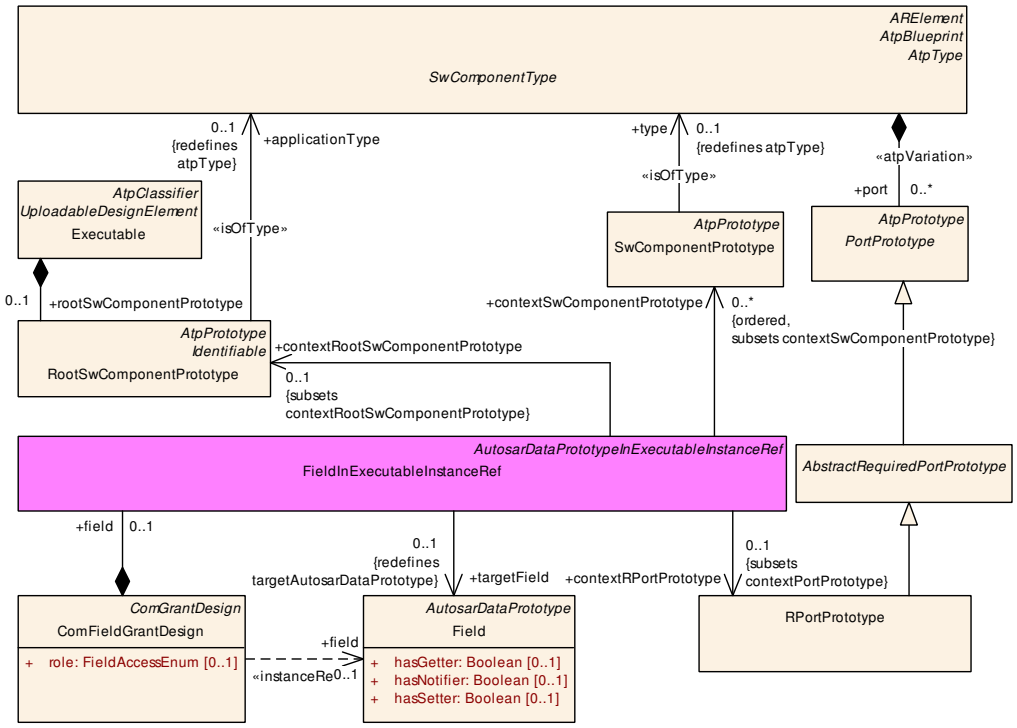


Figure D.7: Modeling of concrete `FieldInExecutableInstanceRef` derived from `AutosarDataPrototypeInExecutableInstanceRef`

Class	**FieldInExecutableInstanceRef**			
Note	This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpInstanceRef*, *AutosarDataPrototypeInExecutableInstanceRef*			
Aggregated by	ComFieldGrantDesign.field			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Tags:** xml.sequenceOffset=10
contextRPort Prototype	RPortPrototype	0..1	ref	**Tags:** xml.sequenceOffset=30
contextSw Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** xml.sequenceOffset=20
targetField	Field	0..1	ref	**Tags:** xml.sequenceOffset=40

Table D.8: FieldInExecutableInstanceRef

D.3 Reference to a Trigger in an Executable

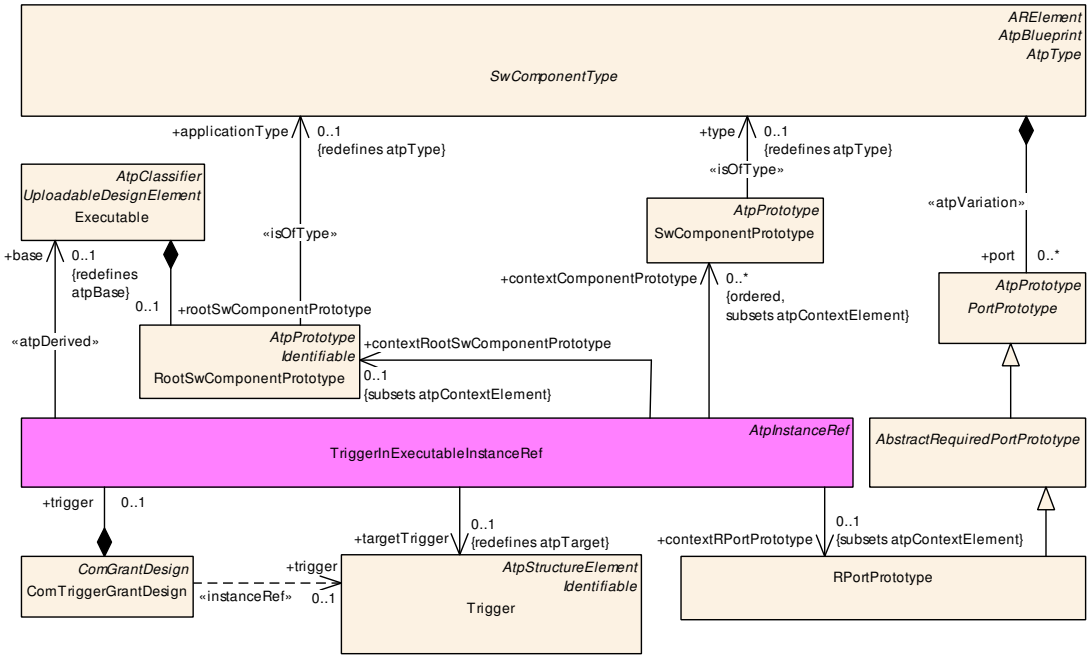


Figure D.8: Modeling of `TriggerInExecutableInstanceRef`

Class	**TriggerInExecutableInstanceRef**			
Note	This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpInstanceRef*			
Aggregated by	ComTriggerGrantDesign.trigger			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpDerived
contextComponentPrototype (ordered)	SwComponentPrototype	*	ref	**Tags:** xml.sequenceOffset=20
contextRootSwComponentPrototype	RootSwComponentPrototype	0..1	ref	**Tags:** xml.sequenceOffset=10
contextRPortPrototype	RPortPrototype	0..1	ref	**Tags:** xml.sequenceOffset=30
targetTrigger	Trigger	0..1	ref	**Tags:** xml.sequenceOffset=40

Table D.9: TriggerInExecutableInstanceRef

D.4 Reference to a PortPrototype in an Executable

The creation of the meta-model for creating a reference to a `PortPrototype` in the context of an `Executable` is executed in a two-step approach where first an abstract structure of the reference is created.

The abstract structure is the basis for the refinement with respect to specific roles of `PortPrototype`s.

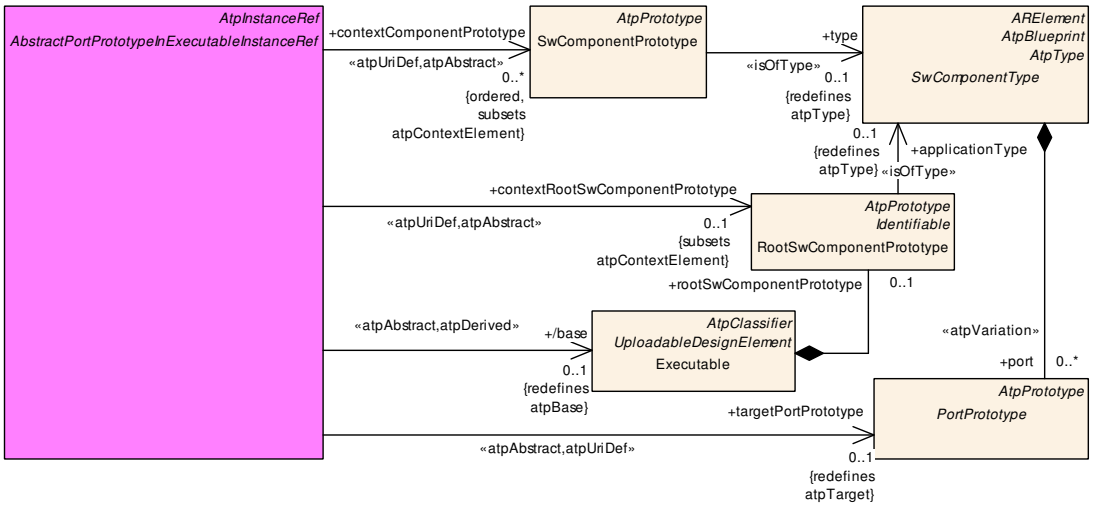


Figure D.9: Modeling of abstract `AbstractPortPrototypeInExecutableInstanceRef`

Class	***AbstractPortPrototypeInExecutableInstanceRef*** (abstract)			
Note	This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpInstanceRef*			
Subclasses	PPortPrototypeInExecutableInstanceRef, PortPrototypeInExecutableInstanceRef, RPortPrototypeIn ExecutableInstanceRef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpAbstract; atpDerived
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Stereotypes:** atpAbstract; atpUriDef **Tags:** xml.sequenceOffset=20
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Stereotypes:** atpAbstract; atpUriDef **Tags:** xml.sequenceOffset=10
targetPort Prototype	PortPrototype	0..1	ref	**Stereotypes:** atpAbstract; atpUriDef **Tags:** xml.sequenceOffset=30

Table D.10: AbstractPortPrototypeInExecutableInstanceRef

Three specializations of `AbstractPortPrototypeInExecutableInstanceRef` exist:

- `PPortPrototypeInExecutableInstanceRef`
- `RPortPrototypeInExecutableInstanceRef`
- `PortPrototypeInExecutableInstanceRef`

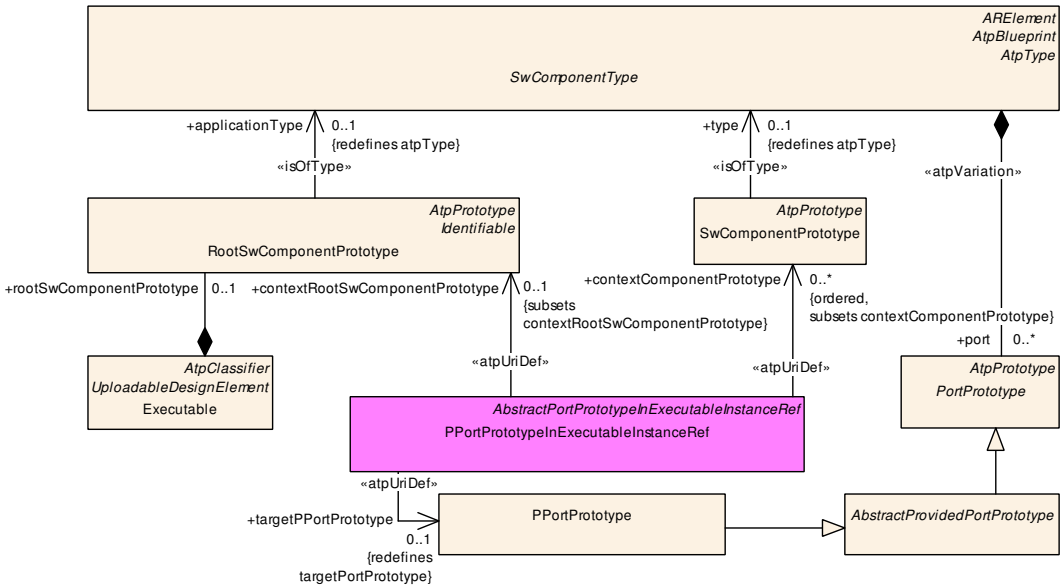


Figure D.10: Modeling of concrete `PPortPrototypeInExecutableInstanceRef` derived from `AbstractPortPrototypeInExecutableInstanceRef`

Class	**PPortPrototypeInExecutableInstanceRef**			
Note	This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractPortPrototypeInExecutableInstanceRef*, *AtpInstanceRef*			
Aggregated by	*AbstractDoIpPortMapping*.pPortPrototypeInExecutable, *AbstractSuspendToRamMapping*.pPortPrototypeInExecutable, ComOfferServiceGrantDesign.providedServicePort, CompositionPPortToExecutablePPortMapping.executableProvidedPort, DiagnosticAuthenticationPortMapping.pPortPrototypeInExecutable, DiagnosticDataPortMapping.pPortPrototypeInExecutable, DiagnosticDoIPCustomPayloadRxPortMapping.pPortPrototypeInExecutable, DiagnosticEnvDataElementCondition.pPortPrototype, DiagnosticInputOutputControlPortMapping.pPortPrototypeInExecutable, DiagnosticSecurityLevelPortMapping.pPortPrototypeInExecutable, DiagnosticServiceGenericMapping.pPortPrototypeInExecutable, DiagnosticServiceValidationMapping.pPortPrototypeInExecutable, DiagnosticSovdAuthorizationPortMapping.pPortPrototypeInExecutable, DiagnosticSovdBulkDataPortMapping.pPortPrototypeInExecutable, DiagnosticSovdConfigurationPortMapping.pPortPrototypeInExecutable, DiagnosticSovdContentPortMapping.pPortPrototypeInExecutable, DiagnosticSovdOperationPortMapping.pPortPrototypeInExecutable, DiagnosticSovdProximityChallengePortMapping.pPortPrototypeInExecutable, DiagnosticSovdServiceValidationPortMapping.pPortPrototypeInExecutable, DiagnosticSovdUpdatePortMapping.pPortPrototypeInExecutable, DltLogSinkToPortPrototypeMapping.pPortPrototype, IdsmContextProviderMapping.pPortPrototypeInExecutable, IdsmTimestampProviderMapping.pPortPrototypeInExecutable, NetworkHandlePortMapping.pPortPrototypeInExecutable, RecoveryNotificationToPPortPrototypeMapping.recoveryAction, StateManagementStateNotification.notificationPort, TimeSyncPortPrototypeToTimeBaseMapping.timeSyncPPortPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=20
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=10
targetPPort Prototype	PPortPrototype	0..1	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=30

Table D.11: PPortPrototypeInExecutableInstanceRef

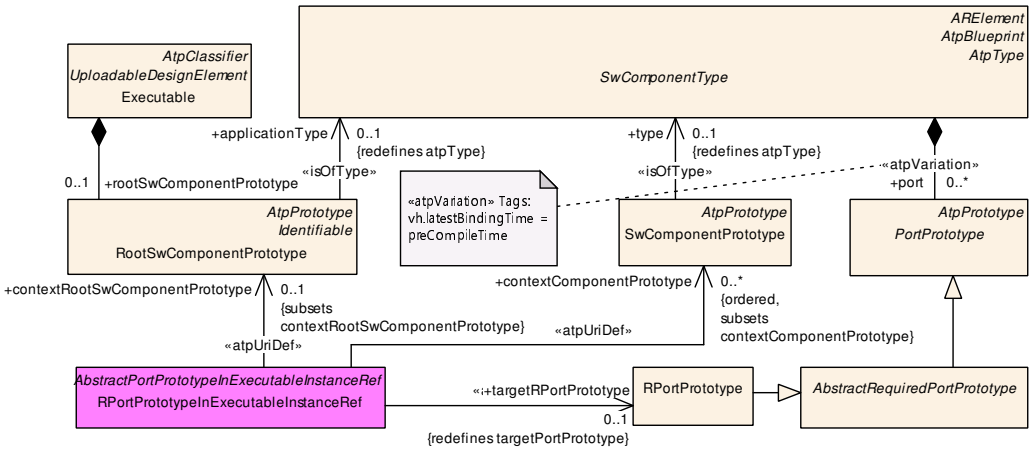


Figure D.11: Modeling of concrete `RPortPrototypeInExecutableInstanceRef` derived from `AbstractPortPrototypeInExecutableInstanceRef`

Class	**RPortPrototypeInExecutableInstanceRef**			
Note	This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractPortPrototypeInExecutableInstanceRef*, *AtpInstanceRef*			
Aggregated by	*AbstractCryptoKeySlotToPortPrototypeMapping*.portPrototype, *AbstractDoIpPortMapping*.rPortPrototypeInExecutable, AdaptiveFirewallToPortPrototypeMapping.rPort, CompositionRPortToExecutableRPortMapping.executableRequiredPort, CryptoCertificateGroupToPortPrototypeMapping.portPrototype, CryptoCertificateToPortPrototypeMapping.portPrototype, CryptoProviderToPortPrototypeMapping.portPrototype, DiagnosticClearConditionPortMapping.rPortPrototypeInExecutable, DiagnosticDoIPCustomPayloadTxPortMapping.rPortPrototypeInExecutable, DiagnosticEnableConditionPortMapping.rPortPrototypeInExecutable, DiagnosticEventPortMapping.rPortPrototypeInExecutable, DiagnosticExtendedDataRecordClientPortMapping.rPortPrototypeInExecutable, DiagnosticExternalAuthenticationPortMapping.rPortPrototypeInExecutable, DiagnosticIndicatorPortMapping.rPortPrototypeInExecutable, DiagnosticMemoryDestinationPortMapping.rPortPrototypeInExecutable, DiagnosticMonitorPortMapping.rPortPrototypeInExecutable, DiagnosticMultipleConditionPortMapping.rPortPrototypeInExecutable, DiagnosticMultipleEventPortMapping.rPortPrototypeInExecutable, DiagnosticMultipleMonitorPortMapping.rPortPrototypeInExecutable, DiagnosticOperationCyclePortMapping.rPortPrototypeInExecutable, DltLogSinkToPortPrototypeMapping.rPortPrototype, FunctionGroupPortMapping.rPortPrototypeInExecutable, IdsmQualifiedEventReceiverMapping.rPortPrototypeInExecutable, IdsmReportingModeProviderMapping.rPortPrototypeInExecutable, LTMessageCollectionToPortPrototypeMapping.rPort, NoSupervision.targetPhmSupervisedEntity, *RawDataStreamMapping*.portPrototype, SecurityEventMapping.reportingPortPrototype, StateManagementSetFunctionGroupStateActionItem.rPortPrototype, *StateManagementStateRequest*.stateRequestPort, TimeSyncPortPrototypeToTimeBaseMapping.timeSyncRPortPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=20
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=10
targetRPort Prototype	RPortPrototype	0..1	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=30

Table D.12: RPortPrototypeInExecutableInstanceRef

Class	**PortPrototypeInExecutableInstanceRef**			
Note	This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractPortPrototypeInExecutableInstanceRef*, *AtpInstanceRef*			
Aggregated by	*PersistencyPortPrototypeToDeploymentMapping*.portPrototype, ServiceInstanceToPortPrototype Mapping.portPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpDerived **Tags:** xml.sequenceOffset=10
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** xml.sequenceOffset=30
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Tags:** xml.sequenceOffset=20
targetPort Prototype	PortPrototype	0..1	ref	**Tags:** xml.sequenceOffset=40

Table D.13: PortPrototypeInExecutableInstanceRef

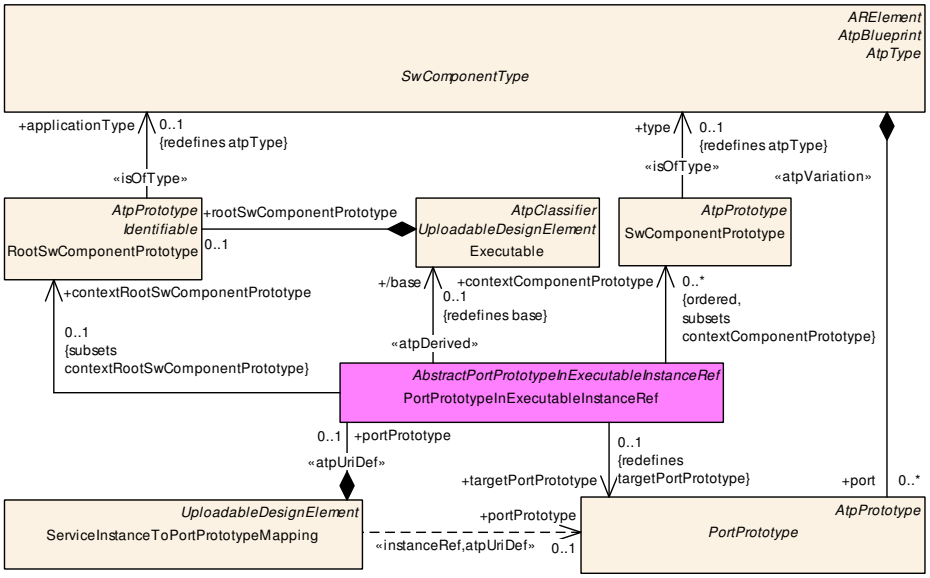


Figure D.12: Modeling of `PortPrototypeInExecutableInstanceRef`

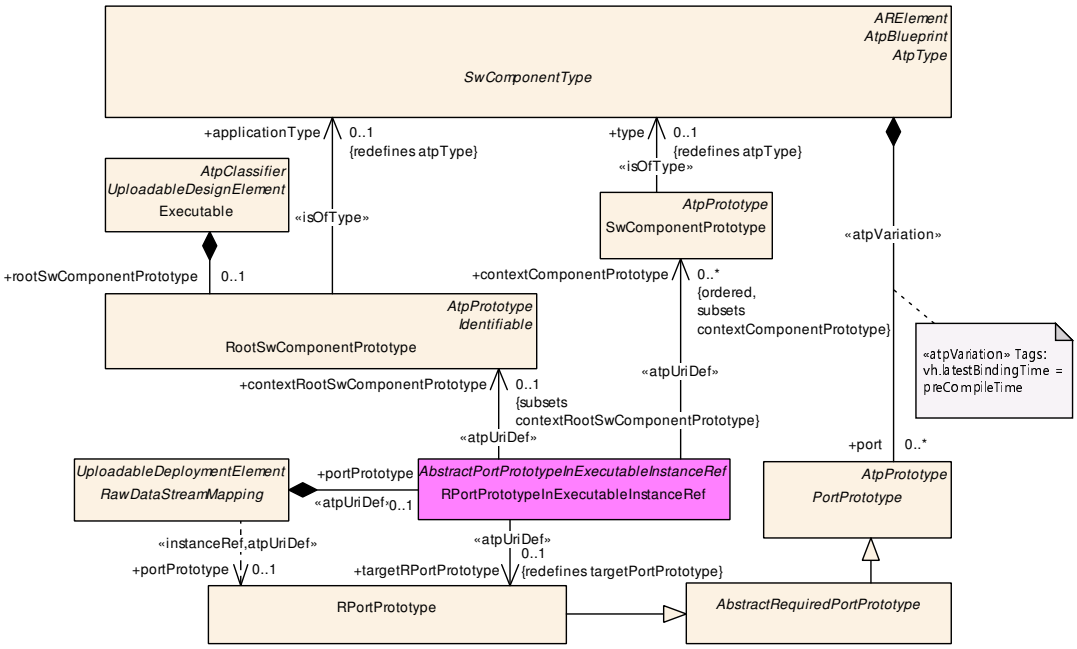


Figure D.13: Modeling of raw data streaming by means of an `RPortPrototype`

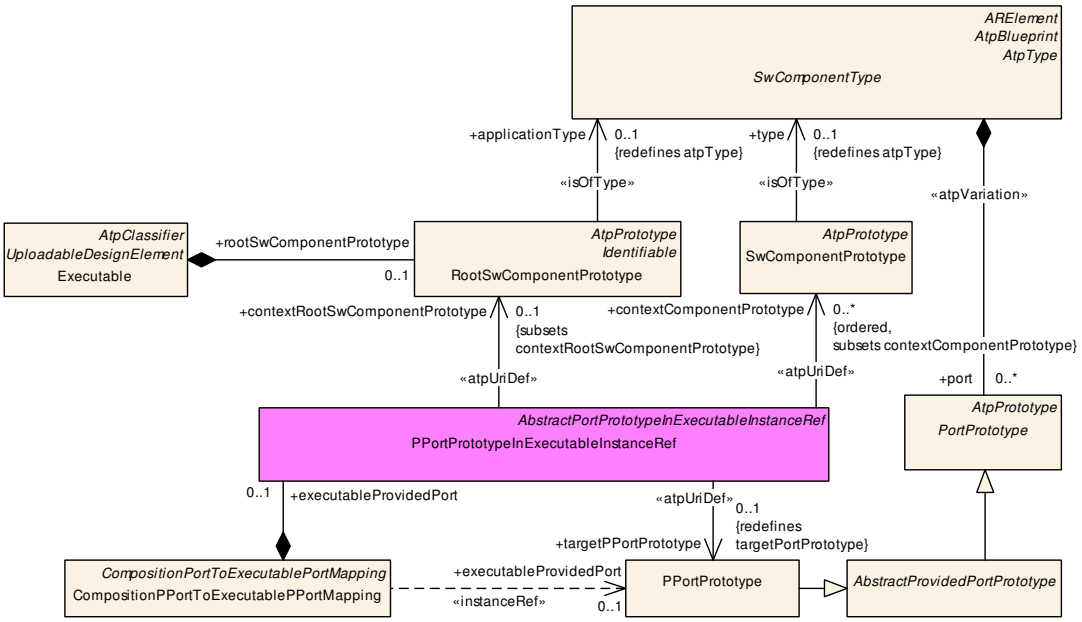


Figure D.14: Modeling of `CompositionPPortToExecutablePPortMapping`

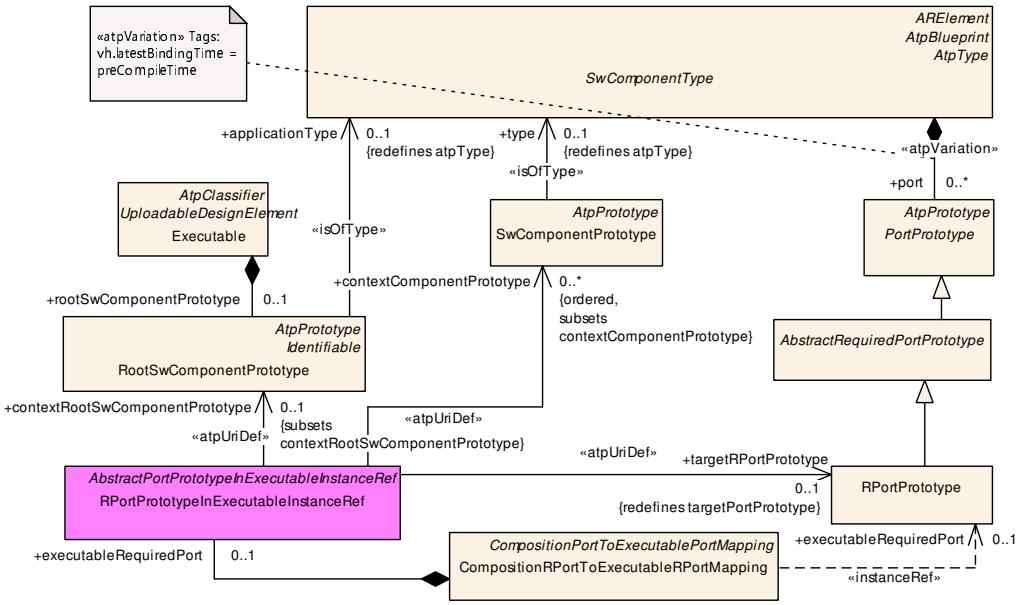


Figure D.15: Modeling of `CompositionRPortToExecutableRPortMapping`

D.5 Modeling of a Method in an Executable

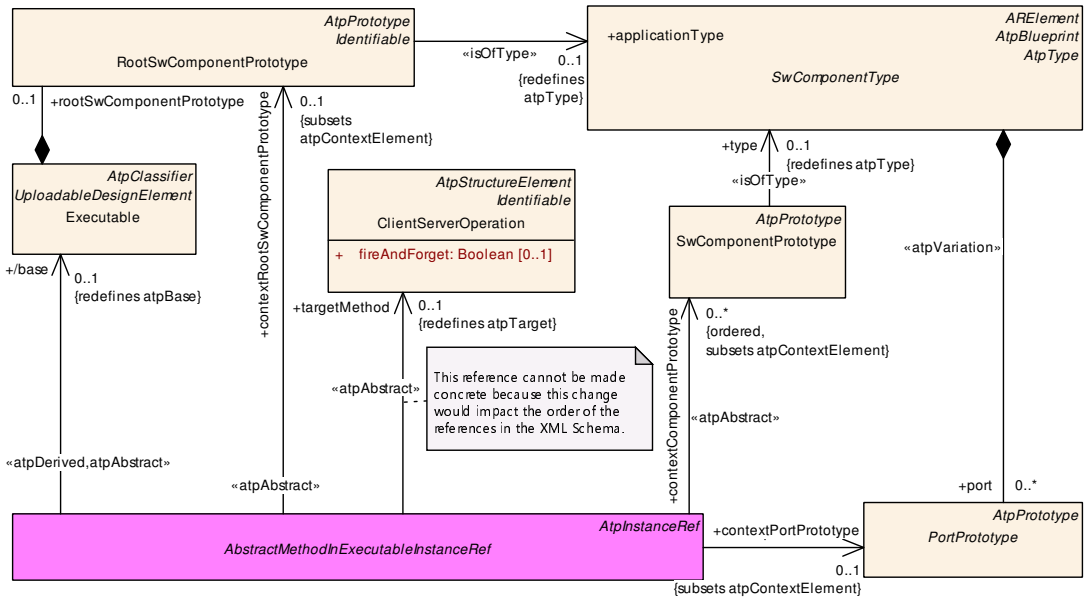


Figure D.16: Modeling of `AbstractMethodInExecutableInstanceRef`

Class	***AbstractMethodInExecutableInstanceRef*** (abstract)
Note	This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARObject*, *AtpInstanceRef*
Subclasses	RequiredMethodInExecutableInstanceRef

△

Class	***AbstractMethodInExecutableInstanceRef*** (abstract)			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpAbstract; atpDerived
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=20
contextPort Prototype	PortPrototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=30
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=10
targetMethod	ClientServerOperation	0..1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=40

Table D.14: AbstractMethodInExecutableInstanceRef

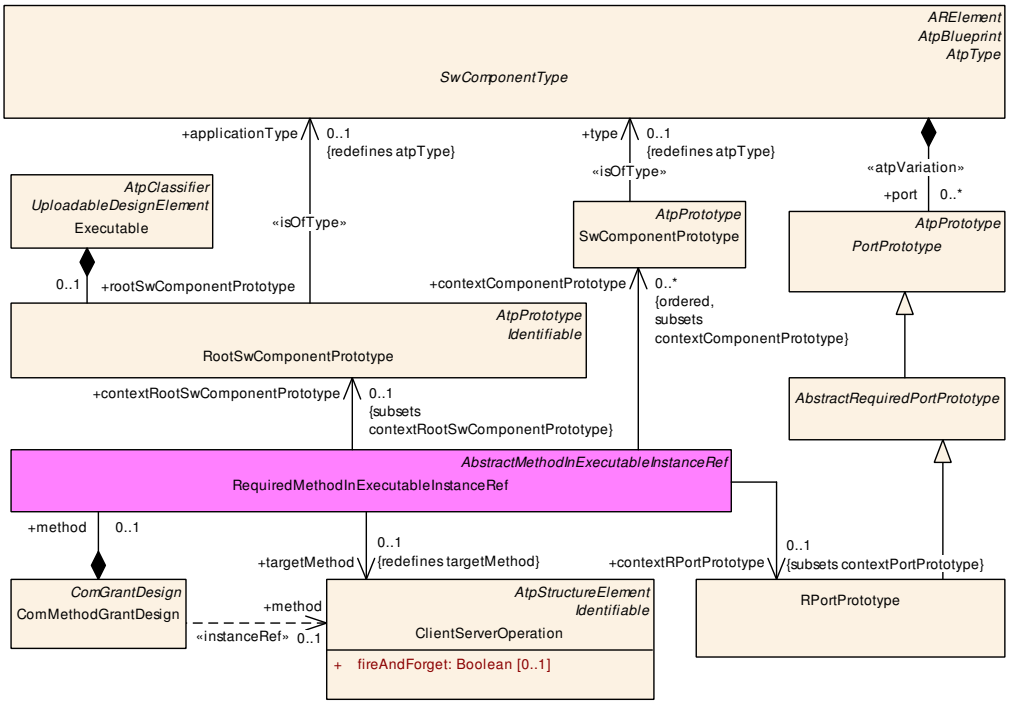


Figure D.17: Modeling of `RequiredMethodInExecutableInstanceRef`

Class	**RequiredMethodInExecutableInstanceRef**			
Note	This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractMethodInExecutableInstanceRef*, *AtpInstanceRef*			
Aggregated by	ComMethodGrantDesign.method			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** xml.sequenceOffset=20

▽

△

Class	**RequiredMethodInExecutableInstanceRef**			
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Tags:** xml.sequenceOffset=10
contextRPort Prototype	RPortPrototype	0..1	ref	**Tags:** xml.sequenceOffset=30
targetMethod	ClientServerOperation	0..1	ref	**Tags:** xml.sequenceOffset=40

Table D.15: RequiredMethodInExecutableInstanceRef

D.6 Modeling of Diagnostic-related InstanceRefs

The wider context of the usage of the aggregations documented in Figure D.18 can be viewed in Figure 5.24, Figure 5.18, Figure 5.17, Figure 5.16, Figure 5.20, and Figure 5.15.

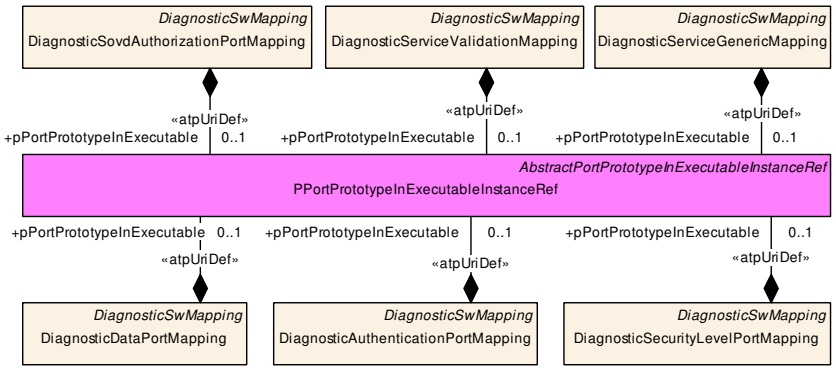


Figure D.18: Modeling of various `DiagnosticSwMapping`s based on `PPortPrototypeInExecutableInstanceRef` (part 1)

The wider context of the usage of the aggregations documented in Figure D.19 can be viewed in Figure 5.26, Figure 5.27, Figure 5.25, Figure 5.43, Figure 5.23, and Figure 5.28.

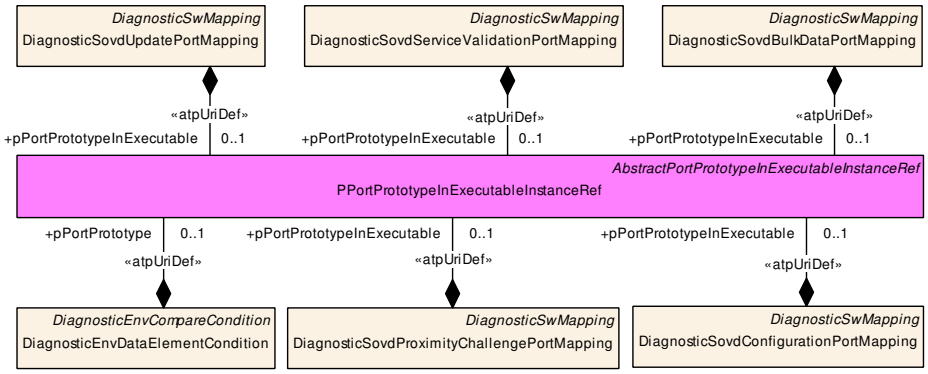


Figure D.19: Modeling of various `DiagnosticSwMapping`s based on `PPortPrototypeInExecutableInstanceRef` (part 2)

The wider context of the usage of the aggregations documented in Figure D.18 can be viewed in Figure 5.4, Figure 5.6, Figure 5.10, Figure 5.12, Figure 5.8, Figure 5.13, and Figure 5.8.

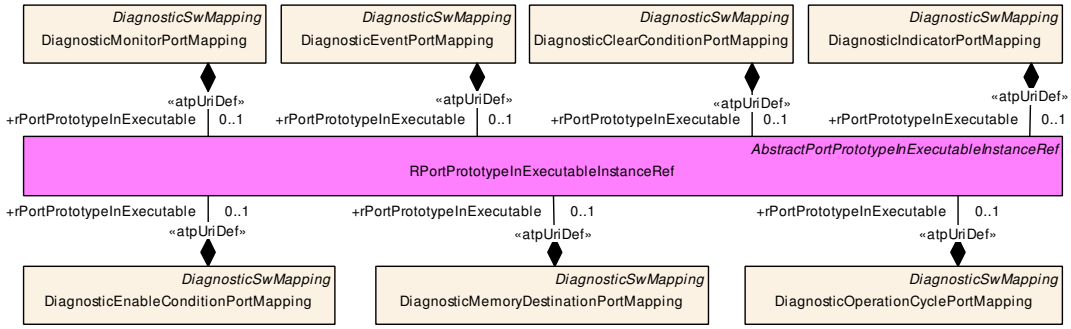


Figure D.20: Modeling of various `DiagnosticSwMapping`s based on `RPortPrototype-InExecutableInstanceRef` (part 1)

The wider context of the usage of the aggregations documented in Figure D.18 can be viewed in Figure 5.21, Figure 5.5, Figure 5.7, and Figure 5.11.

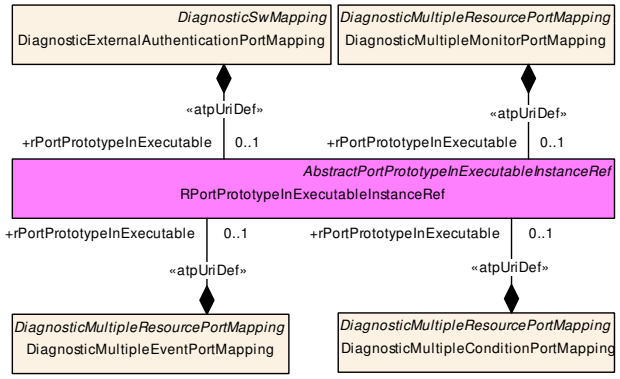


Figure D.21: Modeling of various `DiagnosticSwMapping`s based on `RPortPrototype-InExecutableInstanceRef`(part 2)

D.7 Modeling of PHM-related InstanceRefs

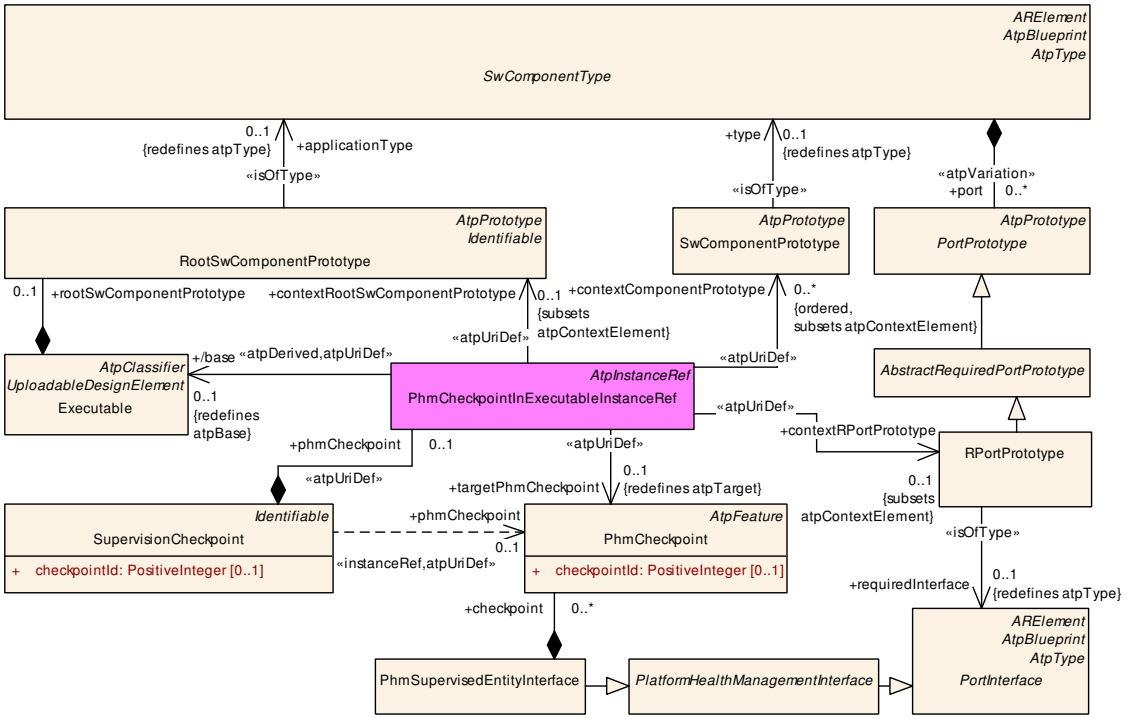


Figure D.22: Modeling of `PhmCheckpointInExecutableInstanceRef`

Class	**PhmCheckpointInExecutableInstanceRef**			
Note	This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpInstanceRef*			
Aggregated by	SupervisionCheckpoint.phmCheckpoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpDerived; atpUriDef **Tags:** xml.sequenceOffset=10
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=30
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=20
contextRPort Prototype	RPortPrototype	0..1	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=40
targetPhm Checkpoint	PhmCheckpoint	0..1	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=50

Table D.16: PhmCheckpointInExecutableInstanceRef

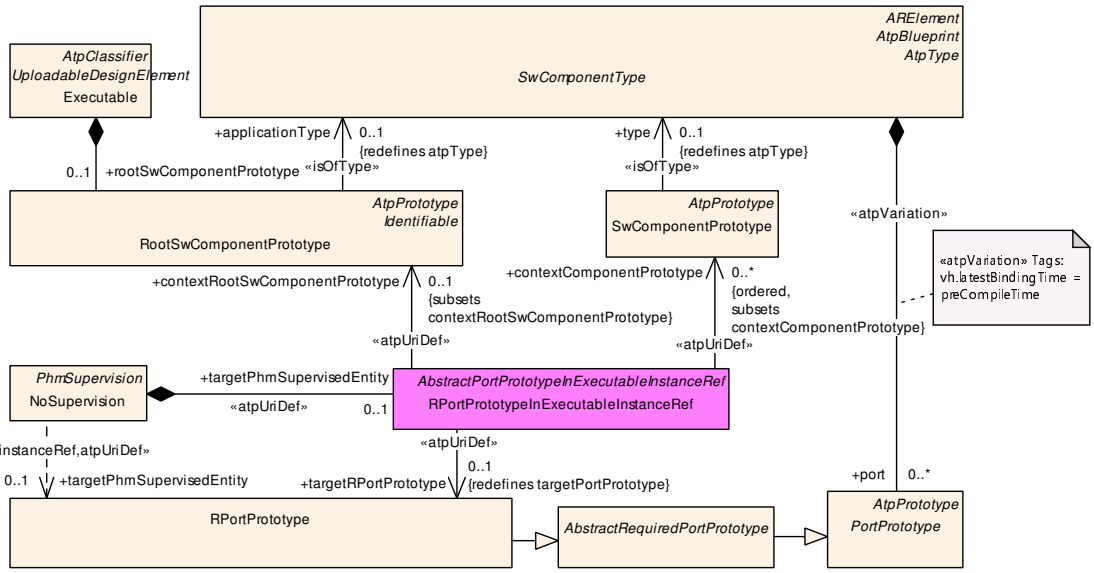


Figure D.23: Modeling of `NoSupervision RPortPrototypeInExecutableInstance-Ref`

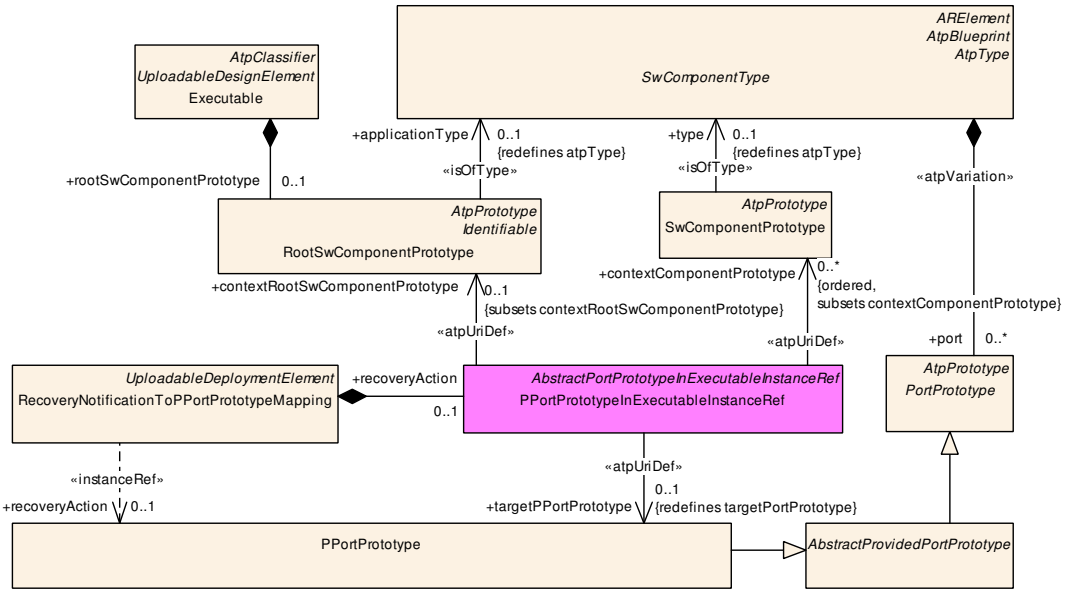


Figure D.24: Modeling of `RecoveryNotificationToPPortPrototypeMapping`

D.8 Modeling of Time-related InstanceRefs

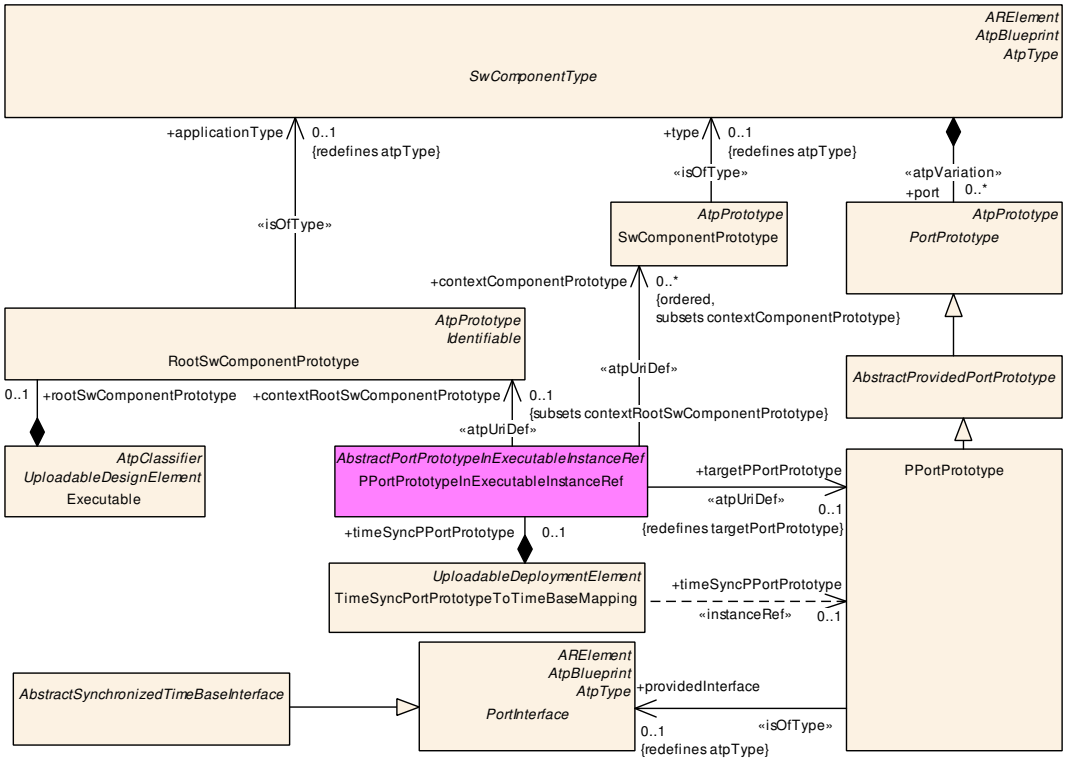


Figure D.25: Modeling of `TimeSyncPortPrototypeToTimeBaseMapping.timeSyncPPortPrototype`

D.9 Modeling of Persistency-related InstanceRefs

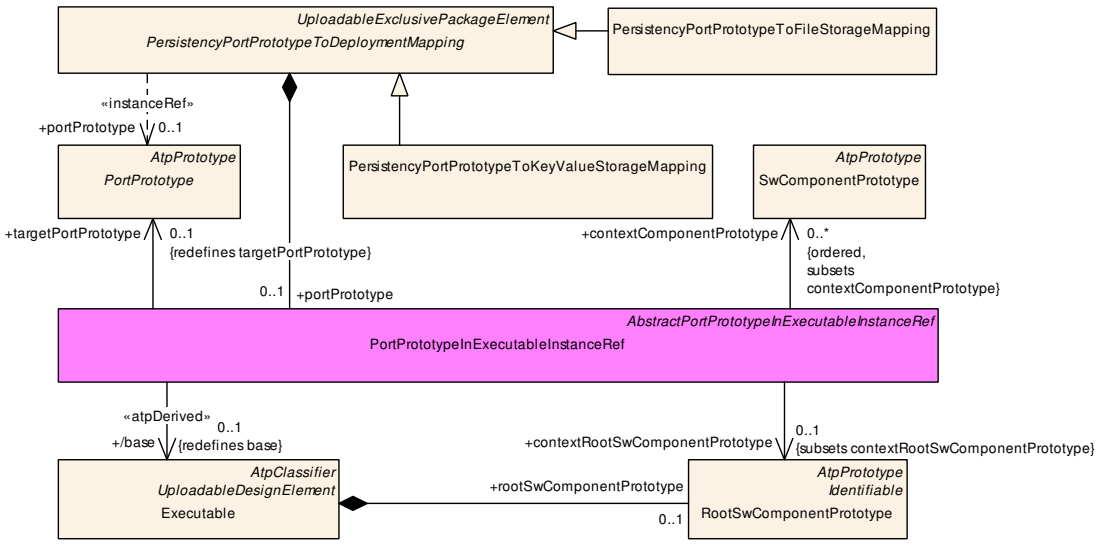


Figure D.26: Modeling of `PersistencyPortPrototypeToDeploymentMapping`

D.10 Modeling of SoftwareClusterDesign-related InstanceRefs

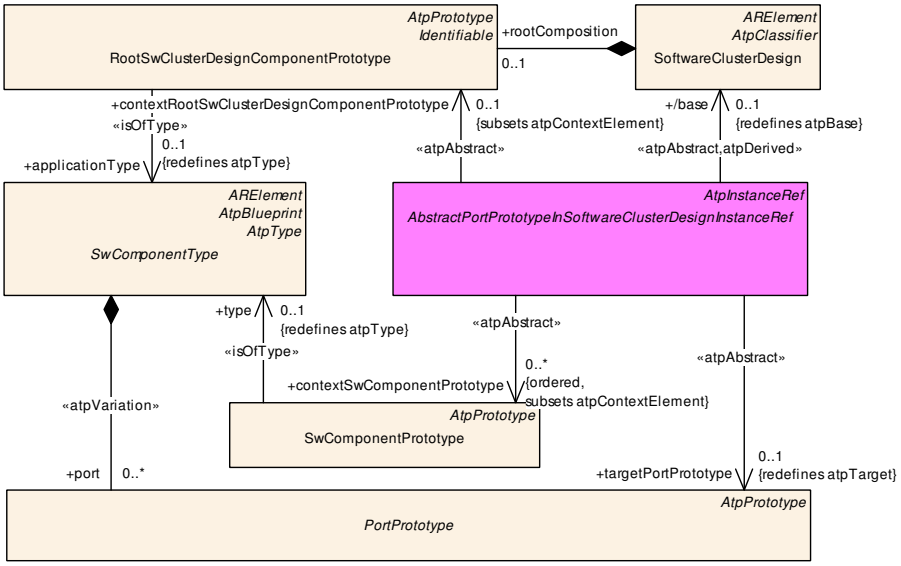


Figure D.27: Modeling of `AbstractPortPrototypeInSoftwareClusterDesign-InstanceRef`

Class	***AbstractPortPrototypeInSoftwareClusterDesignInstanceRef*** (abstract)			
Note	This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpInstanceRef*			
Subclasses	PPortPrototypeInSoftwareClusterDesignInstanceRef, RPortPrototypeInSoftwareClusterDesignInstanceRef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	SoftwareClusterDesign	0..1	ref	**Stereotypes:** atpAbstract; atpDerived
contextRootSwClusterDesignComponentPrototype	RootSwClusterDesignComponentPrototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=10
contextSwComponentPrototype (ordered)	SwComponentPrototype	*	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=20
targetPortPrototype	PortPrototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=30

Table D.17: AbstractPortPrototypeInSoftwareClusterDesignInstanceRef

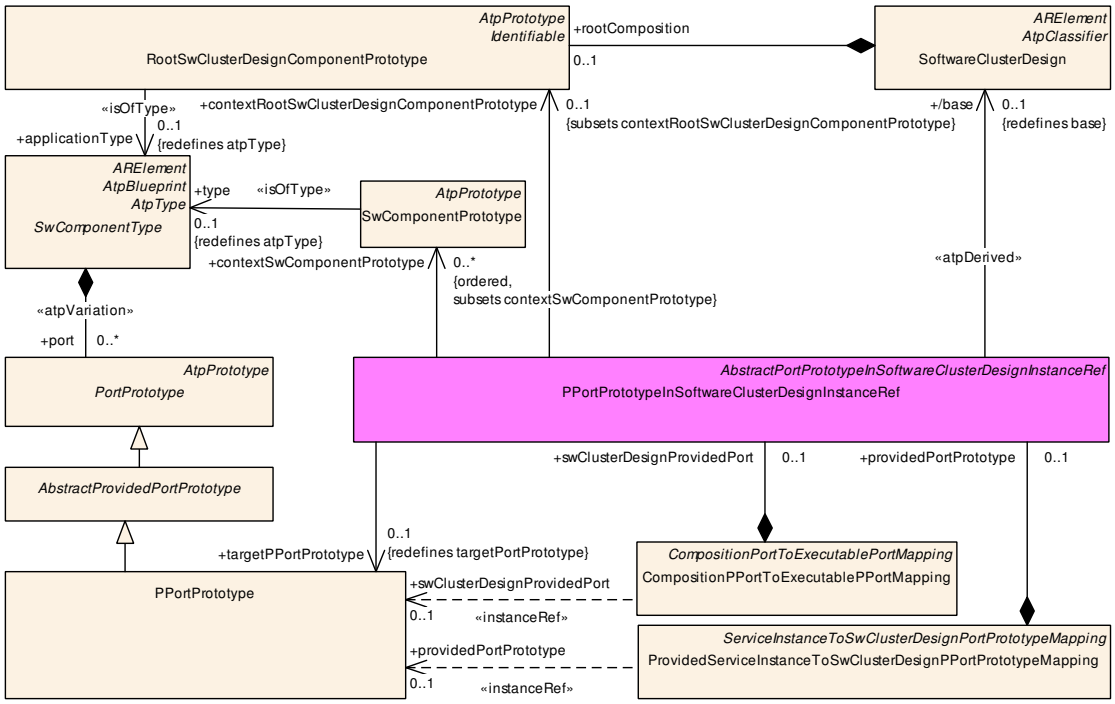


Figure D.28: Modeling of `PPortPrototypeInSoftwareClusterDesignInstanceRef`

Class	**PPortPrototypeInSoftwareClusterDesignInstanceRef**			
Note	This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractPortPrototypeInSoftwareClusterDesignInstanceRef*, *AtpInstanceRef*			
Aggregated by	CompositionPPortToExecutablePPortMapping.swClusterDesignProvidedPort, ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping.providedPortPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	SoftwareClusterDesign	0..1	ref	**Stereotypes:** atpDerived
contextRootSwClusterDesignComponentPrototype	RootSwClusterDesignComponentPrototype	0..1	ref	**Tags:** xml.sequenceOffset=10
contextSwComponentPrototype (ordered)	SwComponentPrototype	*	ref	**Tags:** xml.sequenceOffset=20
targetPPortPrototype	PPortPrototype	0..1	ref	**Tags:** xml.sequenceOffset=30

Table D.18: PPortPrototypeInSoftwareClusterDesignInstanceRef

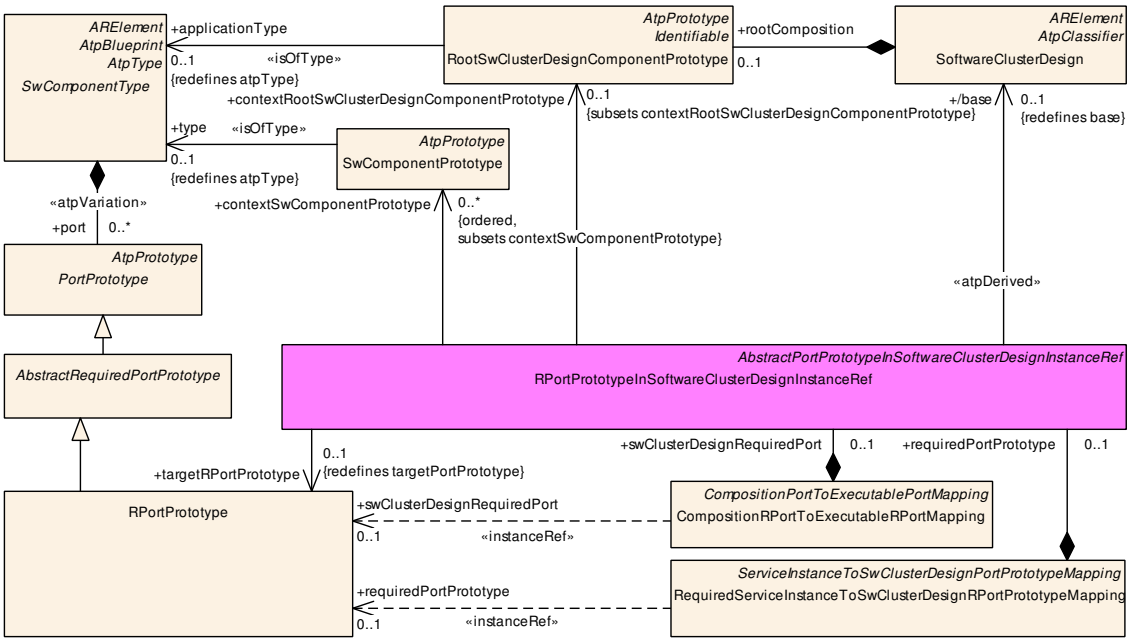


Figure D.29: Modeling of `RPortPrototypeInSoftwareClusterDesignInstanceRef`

Class	**RPortPrototypeInSoftwareClusterDesignInstanceRef**			
Note	This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractPortPrototypeInSoftwareClusterDesignInstanceRef*, *AtpInstanceRef*			
Aggregated by	CompositionRPortToExecutableRPortMapping.swClusterDesignRequiredPort, RequiredServiceInstanceToSwClusterDesignRPortPrototypeMapping.requiredPortPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	SoftwareClusterDesign	0..1	ref	**Stereotypes:** atpDerived
contextRootSwClusterDesignComponentPrototype	RootSwClusterDesignComponentPrototype	0..1	ref	**Tags:** xml.sequenceOffset=10
contextSwComponentPrototype (ordered)	SwComponentPrototype	*	ref	**Tags:** xml.sequenceOffset=20
targetRPortPrototype	RPortPrototype	0..1	ref	**Tags:** xml.sequenceOffset=30

Table D.19: RPortPrototypeInSoftwareClusterDesignInstanceRef

D.11 Modeling of State-Management-related InstanceRefs

This section illustrates the concrete modeling of the instance references used in the modeling of the state management deployment in this document.

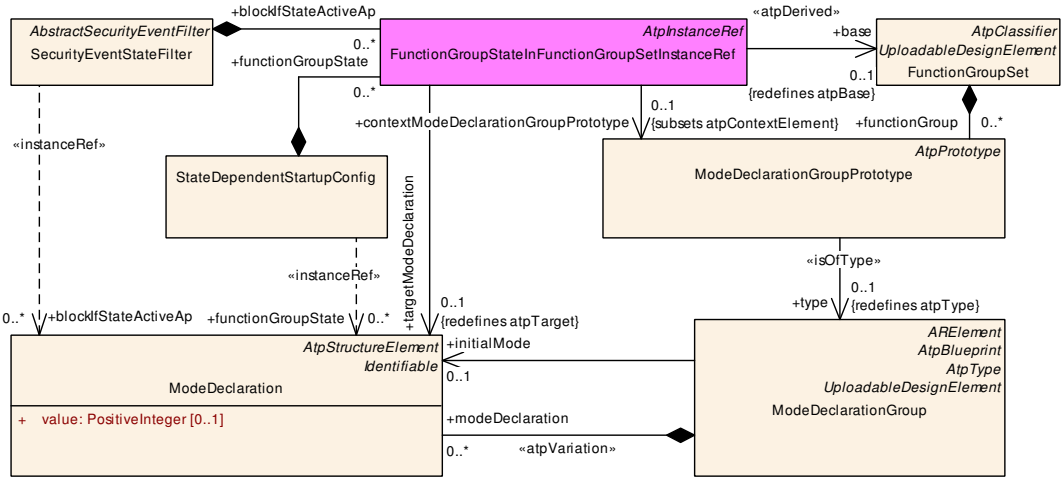


Figure D.30: Modeling of `FunctionGroupStateInFunctionGroupSetInstanceRef`

Class	**FunctionGroupStateInFunctionGroupSetInstanceRef**			
Note	This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpInstanceRef*			
Aggregated by	FunctionGroupPhmStateReference.functionGroupState, NmHandleToFunctionGroupState Mapping.functionGroupState, SecurityEventStateFilter.blockIfStateActiveAp, StateDependentStartup Config.functionGroupState, StateManagementSetFunctionGroupStateActionItem.setFunctionGroupState			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	FunctionGroupSet	0..1	ref	**Stereotypes:** atpDerived
contextMode Declaration GroupPrototype	ModeDeclarationGroup Prototype	0..1	ref	**Tags:** xml.sequenceOffset=10
targetMode Declaration	ModeDeclaration	0..1	ref	**Tags:** xml.sequenceOffset=20

Table D.20: FunctionGroupStateInFunctionGroupSetInstanceRef

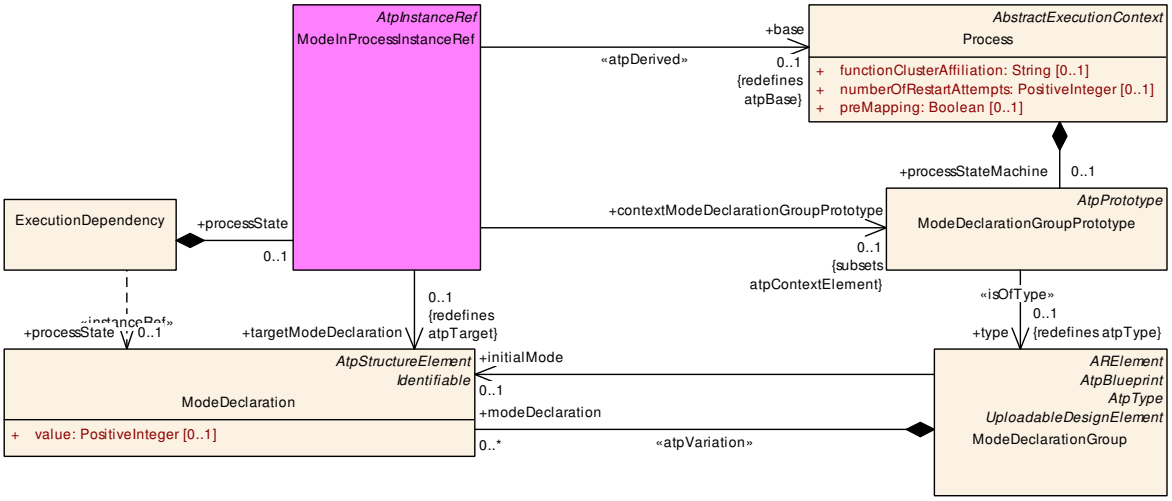


Figure D.31: Modeling of `ModeInProcessInstanceRef`

Class	**ModeInProcessInstanceRef**			
Note	This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpInstanceRef*			
Aggregated by	ExecutionDependency.processState			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Process	0..1	ref	**Stereotypes:** atpDerived **Tags:** xml.sequenceOffset=10
contextMode Declaration GroupPrototype	ModeDeclarationGroup Prototype	0..1	ref	**Tags:** xml.sequenceOffset=20
targetMode Declaration	ModeDeclaration	0..1	ref	**Tags:** xml.sequenceOffset=30

Table D.21: ModeInProcessInstanceRef

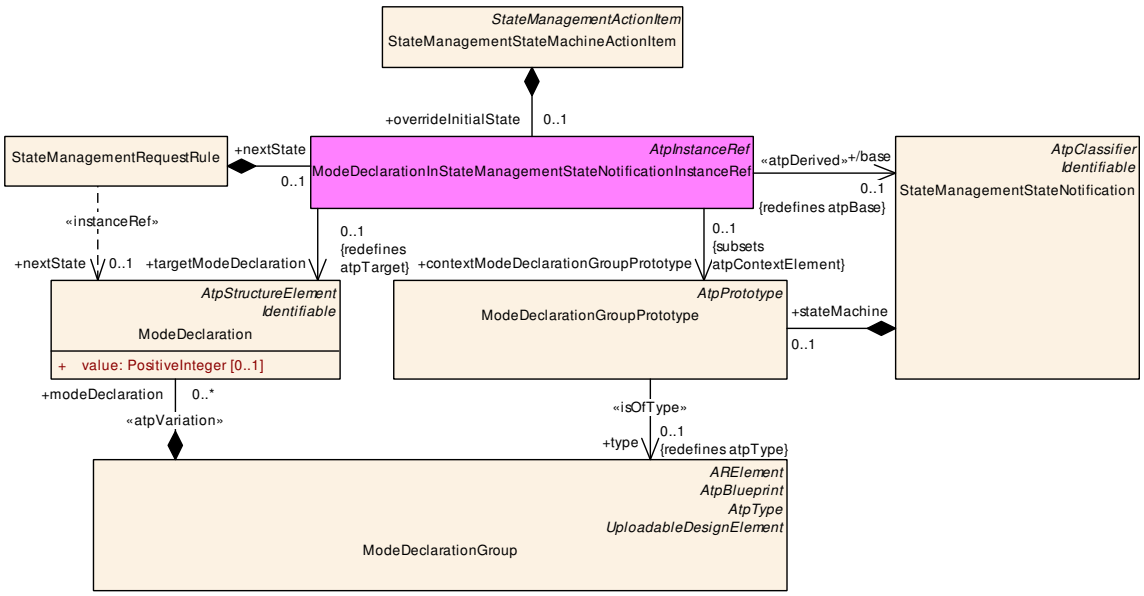


Figure D.32: Modeling of `ModeDeclarationInStateManagementState-NotificationInstanceRef`

Class	**ModeDeclarationInStateManagementStateNotificationInstanceRef**			
Note	**Tags:** atp.Status=draft This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpInstanceRef*			
Aggregated by	StateManagementActionList.affectedState, StateManagementRequestRule.nextState, StateManagement StateMachineActionItem.overrideInitialState, StateManagementTriggerCompareRule.assumedCurrent State			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	StateManagementState Notification	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft
contextMode Declaration GroupPrototype	ModeDeclarationGroup Prototype	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=10

△

Class	**ModeDeclarationInStateManagementStateNotificationInstanceRef**			
targetMode Declaration	ModeDeclaration	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20

Table D.22: ModeDeclarationInStateManagementStateNotificationInstanceRef

D.12 Modeling of Intrusion-Detection-related InstanceRefs

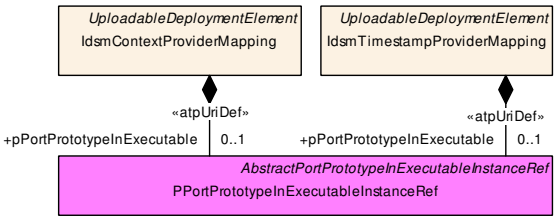


Figure D.33: Modeling of `PPortPrototypeInExecutableInstanceRef` in the context of intrusion detection management

The wider context of the usage of the aggregations documented in Figure D.33 can be viewed in Figure 12.55 and Figure 12.54.

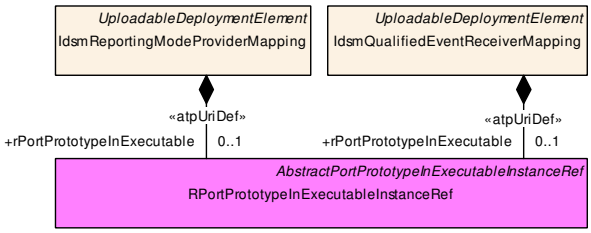


Figure D.34: Modeling of `RPortPrototypeInExecutableInstanceRef` in the context of intrusion detection management

The wider context of the usage of the aggregations documented in Figure D.34 can be viewed in Figure 12.56 and Figure 12.57.

D.13 Modeling of Crypto-related InstanceRefs

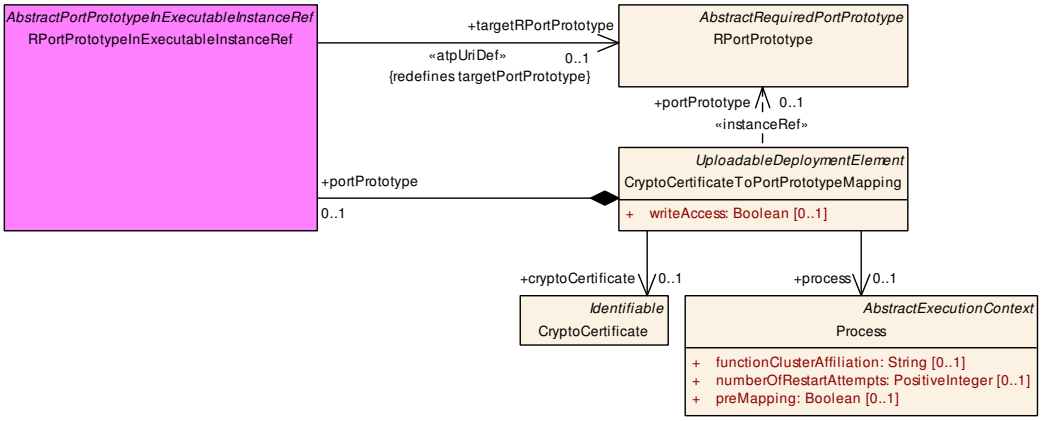


Figure D.35: Modeling of `RPortPrototypeInExecutableInstanceRef` in the context of crypto certificate management

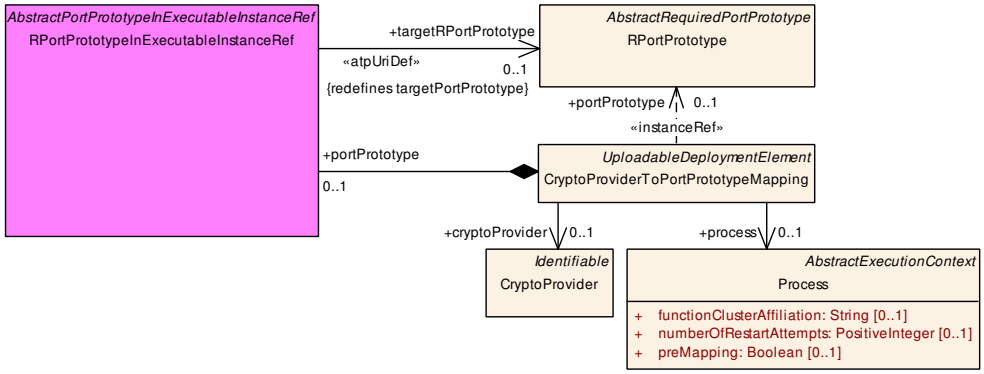


Figure D.36: Modeling of `RPortPrototypeInExecutableInstanceRef` in the context of crypto provider management

D.14 Firewall InstanceRef

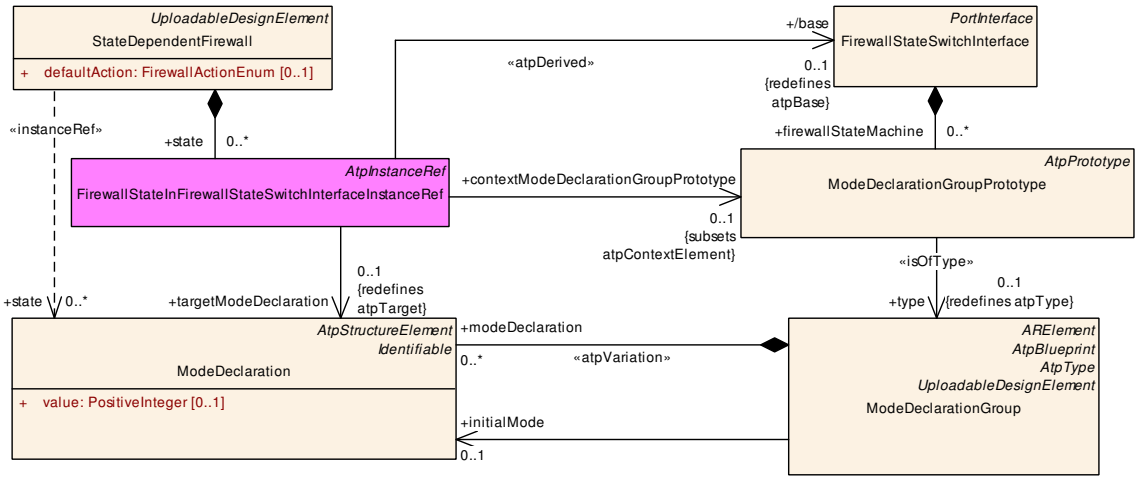


Figure D.37: FirewallStateInFirewallStateSwitchInterface InstanceRef

Class	**FirewallStateInFirewallStateSwitchInterfaceInstanceRef**			
Note	This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AtpInstanceRef*			
Aggregated by	StateDependentFirewall.state			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	FirewallStateSwitch Interface	0..1	ref	**Stereotypes:** atpDerived
contextMode Declaration GroupPrototype	ModeDeclarationGroup Prototype	0..1	ref	**Tags:** xml.sequenceOffset=10
targetMode Declaration	ModeDeclaration	0..1	ref	**Tags:** xml.sequenceOffset=20

Table D.23: FirewallStateInFirewallStateSwitchInterfaceInstanceRef

E Mentioned Class Tables

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.

Class	***ARElement*** (abstract)			
Note	An element that can be defined stand-alone, i.e. without being part of another element (except for packages of course).			
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractMessageToSecureChannelMapping*, AclObjectSet, AclOperation, AclPermission, AclRole, AliasNameSet, Allocator, ApApplicationError, ApApplicationErrorDomain, ApApplicationErrorSet, ApplicabilityInfoSet, *AutosarDataType*, *BaseType*, BlueprintMappingSet, BuildActionManifest, CalibrationParameterValueSet, CanXlProps, ClientIdDefinitionSet, Collection, *CompositionPortToExecutablePortMapping*, CompuMethod, ConsistencyNeedsBlueprintSet, ConstantSpecification, ConstantSpecificationMappingSet, CryptoEllipticCurveProps, CryptoServiceCertificate, CryptoServiceKey, CryptoServicePrimitive, CryptoServiceQueue, CryptoSignatureScheme, DataConstr, DataTransformationSet, DataTypeMappingSet, DdsCpConfig, *DiagnosticCommonElement*, DiagnosticConnection, DiagnosticContributionSet, DltArgumentPropsSet, DltContext, DltEcu, Documentation, E2EProfileCompatibilityProps, E2EProfileConfigurationSet, EthIpProps, EthTcpIpIcmpProps, EthTcpIpProps, EvaluatedVariantSet, FMFeature, FMFeatureMap, FMFeatureModel, FMFeatureSelectionSet, FirewallRule, GeneralPurposeConnection, *GrantDesign*, HwCategory, HwElement, HwType, *IEEE1722TpConnection*, IPSecConfigProps, *IdsCommonElement*, IdsDesign, ImpositionTimeDefinitionGroup, InterfaceMapping, InterpolationRoutineMappingSet, KeywordSet, LTMessageCollectionToPortPrototypeMapping, LifeCycleInfoSet, LifeCycleStateDefinitionGroup, LogAndTraceMessageCollectionSet, MacSecGlobalKayProps, MacSecParticipantSet, McFunction, McGroup, ModeDeclarationGroup, ModeDeclarationMappingSet, PhysicalDimension, PhysicalDimensionMappingSet, *PlatformModuleEndpointConfiguration*, *PortInterface*, PortInterfaceMappingSet, PortInterfaceToDataTypeMapping, PortPrototypeBlueprint, PostBuildVariantCriterion, PostBuildVariantCriterionValueSet, PredefinedVariant, ProcessDesign, ProcessDesignToMachineDesignMapping, RapidPrototypingScenario, SdgDef, *SecureComProps*, *SecurityConnectivityAssociation*, *ServiceInstanceToSwClusterDesignPortPrototypeMapping*, *ServiceInterfaceElementMapping*, ServiceInterfacePedigree, SignalServiceTranslationPropsSet, SoftwareClusterDesign, SomeipDataPrototypeTransformationProps, SomeipRemoteUnicastConfig, SomeipSdClientEventGroupTimingConfig, SomeipSdClientServiceInstanceConfig, SomeipSdServerEventGroupTimingConfig, SomeipSdServerServiceInstanceConfig, SwAddrMethod, SwAxisType, *SwComponentType*, SwRecordLayout, SwSystemconst, SwSystemconstantValueSet, System, SystemComSpecDefinitionSet, SystemSignal, SystemSignalGroup, *TimingExtension*, TlsConnectionGroup, TlvDataIdDefinitionSet, TraceSwitchConfig, TransformationPropsSet, TransformationPropsToServiceInterfaceElementMapping, Unit, UnitGroup, *UploadablePackageElement*, ViewMapSet			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table E.1: ARElement

Class	**ARPackage**			
Note	AUTOSAR package, allowing to create top level packages to structure the contained ARElements. ARPackages are open sets. This means that in a file based description system multiple files can be used to partially describe the contents of a package. This is an extended version of MSR's SW-SYSTEM.			
Base	*ARObject*, *AtpBlueprint*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ARPackage.arPackage, AUTOSAR.arPackage			
Attribute	**Type**	**Mult.**	**Kind**	**Note**



△

Class	**ARPackage**			
arPackage	ARPackage	*	aggr	This represents a sub package within an ARPackage, thus allowing for an unlimited package hierarchy. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=arPackage, arPackage.shortName, ar Package.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
element	PackageableElement	*	aggr	Elements that are part of this package **Stereotypes:** atpVariation **Tags:** atp.Splitkey=element, element.shortName, element.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=20
referenceBase	ReferenceBase	*	aggr	This denotes the reference bases for the package. This is the basis for all relative references within the package. The base needs to be selected according to the base attribute within the references. **Tags:** atp.Splitkey=referenceBase, referenceBase.shortLabel xml.sequenceOffset=10

Table E.2: ARPackage

Class	**AUTOSAR**			
Note	Root element of an AUTOSAR description, also the root element in corresponding XML documents. **Tags:** xml.globalElement=true			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data of an Autosar file. **Tags:** atp.Splitkey=adminData xml.sequenceOffset=10
arPackage	ARPackage	*	aggr	This is the top level package in an AUTOSAR model. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=arPackage, arPackage.shortName, ar Package.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
fileInfo Comment	FileInfoComment	0..1	aggr	This represents a possibility to provide a structured comment in an AUTOSAR file. **Stereotypes:** atpStructuredComment **Tags:** atp.Splitkey=fileInfoComment xml.roleElement=true xml.sequenceOffset=-10 xml.typeElement=false

▽

△

Class	AUTOSAR			
introduction	DocumentationBlock	0..1	aggr	This represents an introduction on the Autosar file. It is intended for example to represent disclaimers and legal notes. **Tags:** atp.Splitkey=introduction xml.sequenceOffset=20

Table E.3: AUTOSAR

Class	***AbstractServiceInstance*** (abstract)			
Note	Provided and Consumed Ethernet Service Instances that are available at the ApplicationEndpoint.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ConsumedServiceInstance, *DdsCpServiceInstance*, ProvidedServiceInstance			
Aggregated by	ServiceInstanceCollectionSet.serviceInstance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
capability Record	TagWithOptionalValue	*	aggr	A sequence of records to store arbitrary name/value pairs conveying additional information about the named service. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=capabilityRecord, capabilityRecord.variation Point.shortLabel vh.latestBindingTime=postBuild
majorVersion	PositiveInteger	0..1	attr	Major Version of the ServiceInterface. Value can be set to a number that represents the Major Version of the service.
method Activation RoutingGroup	PduActivationRouting Group	0..1	aggr	The ServiceDiscovery module is able to activate and deactivate the PDU routing for ClientServerOperations (SOME/IP methods). **Stereotypes:** atpVariation **Tags:** atp.Splitkey=methodActivationRoutingGroup, method ActivationRoutingGroup.shortName, methodActivation RoutingGroup.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table E.4: AbstractServiceInstance

Class	**AdminData**			
Note	AdminData represents the ability to express administrative information and custom extensions for an element. This administration information is to be treated as meta-data such as revision id or state of the file. There are basically the following kinds of meta-data • The language and/or used languages. • Revision information covering e.g. revision number, state, release date, changes. Note that this information can be given in general as well as related to a particular company. • Document meta-data specific for a company Beside that a custom extension of model-data is possible by • Special data			
Base	*ARObject*			
Aggregated by	AUTOSAR.adminData, *Describable*.adminData, *Identifiable*.adminData			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**AdminData**			
docRevision (ordered)	DocRevision	*	aggr	This allows to denote information about the current revision of the object. Note that information about previous revisions can also be logged here. The entries shall be sorted descendant by date in order to reflect the history. Therefore the most recent entry representing the current version is denoted first. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=50 xml.typeElement=false xml.typeWrapperElement=false
language	LEnum	0..1	attr	This attribute specifies the master language of the document or the document fragment. The master language is the one in which the document is maintained and from which the other languages are derived from. In particular in case of inconsistencies, the information in the master language is priority. **Tags:** xml.sequenceOffset=20
sdg	Sdg	*	aggr	This property allows to keep special data which is not represented by the standard model. It can be utilized to keep e.g. tool specific data. **Tags:** atp.Splitkey=sdg, sdg.sdgCaption.shortName xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=60 xml.typeElement=false xml.typeWrapperElement=false
usedLanguages	MultiLanguagePlainText	0..1	aggr	This property specifies the languages which are provided in the document. Therefore it should only be specified in the top level admin data. For each language provided in the document there is one entry in MultilanguagePlain Text. The content of each entry can be used for illustration of the language. The used language itself depends on the language attribute in the entry. **Tags:** atp.Splitkey=usedLanguages xml.sequenceOffset=30

Table E.5: AdminData

Class	**ApplicationArrayDataType**			
Note	An application data type which is an array, each element is of the same application data type. **Tags:** atp.recommendedPackage=ApplicationDataTypes			
Base	*ARElement*, *ARObject*, *ApplicationCompositeDataType*, *ApplicationDataType*, *AtpBlueprint*, *Atp Classifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicArray SizeProfile	String	0..1	attr	Specifies the profile which the array will follow if it is a variable size array.

▽

△

Class	ApplicationArrayDataType			
element	ApplicationArray Element	0..1	aggr	This association implements the concept of an array element. That is, in some cases it is necessary to be able to identify single array elements, e.g. as input values for an interpolation routine. **Tags:** atp.Splitkey=element, element.shortName

Table E.6: ApplicationArrayDataType

Class	ApplicationArrayElement			
Note	Describes the properties of the elements of an application array data type.			
Base	*ARObject*, *ApplicationCompositeElementDataPrototype*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ApplicationArrayDataType.element, *AtpClassifier*.atpFeature			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
arraySize Handling	ArraySizeHandling Enum	0..1	attr	The way how the size of the array is handled.
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls how the information about the array size shall be interpreted.
indexDataType	ApplicationPrimitive DataType	0..1	ref	This reference can be taken to assign a `CompuMethod` of category `TEXTTABLE` to the array. The texttable entries associate a textual value to an index number such that the element with that index number is represented by a symbolic name. **Tags:** atp.Splitkey=indexDataType
maxNumberOf Elements	PositiveInteger	0..1	attr	The maximum number of elements that the array can contain. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table E.7: ApplicationArrayElement

Class	***ApplicationCompositeDataType*** (abstract)			
Note	Abstract base class for all application data types composed of other data types.			
Base	*ARElement*, *ARObject*, *ApplicationDataType*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	ApplicationArrayDataType, ApplicationAssocMapDataType, ApplicationRecordDataType			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table E.8: ApplicationCompositeDataType

Class	***ApplicationCompositeElementDataPrototype*** (abstract)
Note	This class represents a data prototype which is aggregated within a composite application data type (record or array). It is introduced to provide a better distinction between target and context in instance Refs.
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*
Subclasses	ApplicationArrayElement, ApplicationAssocMapElement, ApplicationRecordElement
Aggregated by	*AtpClassifier*.atpFeature

▽

△

Class	***ApplicationCompositeElementDataPrototype*** (abstract)			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
type	ApplicationDataType	0..1	tref	This represents the corresponding data type. **Stereotypes:** isOfType **Tags:** atp.Splitkey=type

Table E.9: ApplicationCompositeElementDataPrototype

Class	**ApplicationError**			
Note	This is a user-defined error that is associated with an element of an AUTOSAR interface. It is specific for the particular functionality or service provided by the AUTOSAR software component.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ClientServerInterface.possibleError			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
errorCode	Integer	0..1	attr	The RTE generator is forced to assign this value to the corresponding error symbol. Note that for error codes certain ranges are predefined (see RTE specification).

Table E.10: ApplicationError

Class	**ApplicationSwComponentType**			
Note	The `ApplicationSwComponentType` is used to represent the application software. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.11: ApplicationSwComponentType

Class	**ApplicationValueSpecification**			
Note	This meta-class represents values for DataPrototypes typed by ApplicationDataTypes (this includes in particular compound primitives). For further details refer to ASAM CDF 2.0. This meta-class corresponds to some extent with SW-INSTANCE in ASAM CDF 2.0.			
Base	*ARObject*, *CompositeRuleBasedValueArgument*, *ValueSpecification*			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValueSpecification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, CompositeRuleBasedValueSpecification.compoundPrimitiveArgument, ConstantSpecification.valueSpec, CryptoServiceKey.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElementCondition.compareValue, DiagnosticEnvSovdDataCondition.compareValue, FieldSenderComSpec.initValue, ISignal.initValue, ISignal.timeoutSubstitutionValue, NonqueuedReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, NvProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.initValue, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireComSpec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, PortDefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalidValue, VariableDataPrototype.initValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ApplicationValueSpecification**			
category	Identifier	0..1	attr	Specifies to which category of ApplicationDataType this ApplicationValueSpecification can be applied (e.g. as an initial value), thus imposing constraints on the structure and semantics of the contained values, see [constr_1006] and [constr_1519].
swAxisCont (ordered)	SwAxisCont	*	aggr	This represents the axis values of a Compound Primitive Data Type (curve or map). The first swAxisCont describes the x-axis, the second sw AxisCont describes the y-axis, the third swAxisCont describes the z-axis. In addition to this, the axis can be denoted in swAxisIndex.
swValueCont	SwValueCont	0..1	aggr	This represents the values of a Compound Primitive Data Type. **Tags:** atp.Splitkey=swValueCont

Table E.12: ApplicationValueSpecification

Class	**ArgumentDataPrototype**			
Note	An argument of an operation, carries direction and implementation information.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, ClientServerOperation.argument			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
direction	ArgumentDirection Enum	0..1	attr	This attribute specifies the direction of the argument.
serverArgument ImplPolicy	ServerArgumentImpl PolicyEnum	0..1	attr	This defines how the argument type of the servers `RunnableEntity` is implemented. If the attribute is not defined this has the same semantics as if the attribute is set to the value `useArgumentType` for primitive arguments and structures.

Table E.13: ArgumentDataPrototype

Enumeration	**ArgumentDirectionEnum**
Note	Use cases: • Arguments in ClientServerOperation can have different directions that need to be formally indicated because they have an impact on how the function signature looks like eventually. • Arguments in BswModuleEntry already determine a function signature, but the direction is used to specify the semantics, especially of pointer arguments.
Aggregated by	ArgumentDataPrototype.direction, DiagnosticSovdAccessArgument.direction, RunnableEntity Argument.direction, SwServiceArg.direction
Literal	***Description***
in	The argument value is passed to the callee. **Tags:** atp.EnumerationLiteralIndex=0
inout	The argument value is passed to the callee but also passed back from the callee to the caller. **Tags:** atp.EnumerationLiteralIndex=1
out	The argument value is passed from the callee to the caller. **Tags:** atp.EnumerationLiteralIndex=2

Table E.14: ArgumentDirectionEnum

Class	ArrayValueSpecification			
Note	Specifies the values for an array.			
Base	*ARObject*, *CompositeValueSpecification*, *ValueSpecification*			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValue Specification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, CompositeRuleBasedValueSpecification.argument, Constant Specification.valueSpec, CryptoServiceKey.developmentValue, DiagnosticEnvDataCondition.compare Value, DiagnosticEnvDataElementCondition.compareValue, DiagnosticEnvSovdDataCondition.compare Value, FieldSenderComSpec.initValue, ISignal.initValue, ISignal.timeoutSubstitutionValue, Nonqueued ReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, Nonqueued SenderComSpec.initValue, NvProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInit Value, NvRequireComSpec.initValue, ParameterDataPrototype.initValue, ParameterProvideComSpec.init Value, ParameterRequireComSpec.initValue, PersistencyDataRequiredComSpec.initValue, Persistency KeyValuePair.initValue, PortDefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, Record ValueSpecification.field, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalid Value, VariableDataPrototype.initValue			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
element (ordered)	ValueSpecification	*	aggr	The value for a single array element. All Value Specifications aggregated by ArrayValueSpecification shall have the same structure. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
intendedPartial Initialization Count	PositiveInteger	0..1	attr	This attribute shall only have a meaning for dynamic arrays and shall be taken as a sanity check: the number filled in the attribute shall be identical to the number of ArrayValueSpecification.element. If the attribute does not exist it means that no partial initialization is intended.

Table E.15: ArrayValueSpecification

Class	AssemblySwConnector			
Note	`AssemblySwConnector`s are exclusively used to connect `SwComponentPrototype`s in the context of a `CompositionSwComponentType`.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SwConnector*			
Aggregated by	*AtpClassifier*.atpFeature, CompositionSwComponentType.connector			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
provider	AbstractProvidedPort Prototype	0..1	iref	Instance of providing port. **Tags:** atp.Splitkey=provider, provider.contextComponent, provider.targetPPort **InstanceRef implemented by:** PPortInComposition InstanceRef
requester	AbstractRequiredPort Prototype	0..1	iref	Instance of requiring port. **Tags:** atp.Splitkey=requester, requester.context Component, requester.targetRPort **InstanceRef implemented by:** RPortInComposition InstanceRef

Table E.16: AssemblySwConnector

Class	***AutosarDataPrototype*** (abstract)			
Note	Base class for prototypical roles of an `AutosarDataType`.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ArgumentDataPrototype, Field, ParameterDataPrototype, PersistencyDataElement, VariableData Prototype			
Aggregated by	*AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
type	AutosarDataType	0..1	tref	This represents the corresponding data type. **Stereotypes:** isOfType **Tags:** atp.Splitkey=type

Table E.17: AutosarDataPrototype

Class	***AutosarDataType*** (abstract)			
Note	Abstract base class for user defined AUTOSAR data types for software.			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractImplementationDataType*, *ApplicationDataType*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
swDataDef Props	SwDataDefProps	0..1	aggr	The properties of this AutosarDataType. **Tags:** atp.Splitkey=swDataDefProps

Table E.18: AutosarDataType

Class	***BaseType*** (abstract)			
Note	This abstract meta-class represents the ability to specify a platform dependent base type.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	SwBaseType			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
baseType Definition	BaseTypeDefinition	1	aggr	This is the actual definition of the base type. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

Table E.19: BaseType

Class	**BaseTypeDirectDefinition**			
Note	This BaseType is defined directly (as opposite to a derived BaseType)			
Base	*ARObject*, *BaseTypeDefinition*			
Aggregated by	*BaseType*.baseTypeDefinition			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	**BaseTypeDirectDefinition**			
baseType Encoding	BaseTypeEncoding String	0..1	attr	This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence. **Tags:** xml.sequenceOffset=90
baseTypeSize	PositiveInteger	0..1	attr	Describes the length of the data type specified in the container in bits. **Tags:** xml.sequenceOffset=70
byteOrder	ByteOrderEnum	0..1	attr	This attribute specifies the byte order of the base type. **Tags:** xml.sequenceOffset=110
memAlignment	PositiveInteger	0..1	attr	This attribute describes the alignment of the memory object in bits. E.g. "8" specifies, that the object in question is aligned to a byte while "32" specifies that it is aligned four byte. If the value is set to "0" the meaning shall be interpreted as "unspecified". **Tags:** xml.sequenceOffset=100
native Declaration	NativeDeclarationString	0..1	attr	This attribute describes the declaration of such a base type in the native programming language, primarily in the Programming language C. This can then be used by a code generator to include the necessary declarations into a header file. For example BaseType with shortName: "MyUnsignedInt" native Declaration: "unsigned short" Results in typedef unsigned short MyUnsignedInt; If the attribute is not defined the referring Implementation DataTypes will not be generated as a typedef by RTE. If a nativeDeclaration type is given it shall fulfill the characteristic given by basetypeEncoding and baseType Size. This is required to ensure the consistent handling and interpretation by software components, RTE, COM and MCM systems. **Tags:** xml.sequenceOffset=120

Table E.20: BaseTypeDirectDefinition

Class	**CanControllerXlConfiguration**			
Note	This meta-class represents the CAN XL-specific controller attributes.			
Base	*ARObject*			
Aggregated by	*AbstractCanCommunicationControllerAttributes*.canControllerXlAttributes, CanXlProps.canXlConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
errorSignaling Enabled	Boolean	0..1	attr	Specifies if error signaling shall be enabled. This is not possible when the transceiver is switched to PWM mode (trcvPwmModeEnabled set to TRUE). TRUE: Error signaling shall be enabled. FALSE: Error signaling shall be disabled.
propSeg	PositiveInteger	0..1	attr	Specifies propagation delay in time quantas.
pwmL	PositiveInteger	0..1	attr	Specifies the PWM long phase length.
pwmO	PositiveInteger	0..1	attr	Specifies the PWM time offset.
pwmS	PositiveInteger	0..1	attr	Specifies the PWM short phase length.

▽

△

Class	**CanControllerXlConfiguration**			
sspOffset	PositiveInteger	0..1	attr	Specifies the Transmitter Delay Compensation Offset in minimum time quanta. Transmitter Delay Compensation Offset is used to adjust the position of the Secondary Sample Point (SSP), relative to the beginning of the received bit. If this parameter is configured, the Transmitter Delay Compensation is done by measurement of the CAN controller. If not specified Transmitter Delay Compensation is disabled.
syncJumpWidth	PositiveInteger	0..1	attr	Specifies the synchronization jump width for the controller in time quantas.
timeSeg1	PositiveInteger	0..1	attr	Specifies phase segment 1 in time quantas.
timeSeg2	PositiveInteger	0..1	attr	Specifies phase segment 2 in time quantas.
trcvPwmMode Enabled	Boolean	0..1	attr	Specifies if the transceiver shall be set to the PWM mode. TRUE: The transceiver shall be switched to PWM mode. FALSE: The transceiver shall work in classic CAN mode.

Table E.21: CanControllerXlConfiguration

Class	**CanControllerXlConfigurationRequirements**			
Note	This element allows the specification of ranges for the CAN XL configuration parameters. These ranges are taken as requirements and have to be respected by the ECU developer.			
Base	*ARObject*			
Aggregated by	*AbstractCanCommunicationControllerAttributes*.canControllerXlRequirements, CanXlProps.canXlConfig Reqs			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
errorSignaling Enabled	Boolean	0..1	attr	Specifies if error signaling shall be enabled. This is not possible when the transceiver is switched to PWM mode (trcvPwmModeEnabled set to TRUE). TRUE: Error signaling shall be enabled. FALSE: Error signaling shall be disabled.
maxNumberOf TimeQuantaPer Bit	Integer	0..1	attr	Maximum number of time quanta in the bit time.
maxPwmL	PositiveInteger	0..1	attr	Specifies the maximum PWM long phase length.
maxPwmO	PositiveInteger	0..1	attr	Specifies the maximum PWM time offset.
maxPwmS	PositiveInteger	0..1	attr	Specifies the maximum PWM short phase length.
maxSample Point	Float	0..1	attr	The max. value of the sample point as a percentage of the total bit time.
maxSyncJump Width	Float	0..1	attr	The max. Synchronization Jump Width value as a percentage of the total bit time. The (Re-)Synchronization Jump Width (SJW) defines how far a resynchronization may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.
maxTrcvDelay Compensation Offset	TimeValue	0..1	attr	Specifies the maximum Transceiver Delay Compensation Offset in seconds. If not specified Transceiver Delay Compensation is disabled.
minNumberOf TimeQuantaPer Bit	Integer	0..1	attr	Minimum number of time quanta in the bit time.
minPwmL	PositiveInteger	0..1	attr	Specifies the minimum PWM long phase length.
minPwmO	PositiveInteger	0..1	attr	Specifies the minimum PWM time offset.

▽

△

Class	**CanControllerXlConfigurationRequirements**			
minPwmS	PositiveInteger	0..1	attr	Specifies the minimum PWM short phase length.
minSamplePoint	Float	0..1	attr	The min. value of the sample point as a percentage of the total bit time.
minSyncJump Width	Float	0..1	attr	The min. Synchronization Jump Width value as a percentage of the total bit time. The (Re-)Synchronization Jump Width (SJW) defines how far a resynchronization may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.
minTrcvDelay Compensation Offset	TimeValue	0..1	attr	Specifies the minimum Transceiver Delay Compensation Offset in seconds. If not specified Transceiver Delay Compensation is disabled.
trcvPwmMode Enabled	Boolean	0..1	attr	Specifies if the transceiver shall be set to the PWM mode. TRUE: The transceiver shall be switched to PWM mode. FALSE: The transceiver shall work in classic CAN mode.

Table E.22: CanControllerXlConfigurationRequirements

Class	**ClientIdRange**			
Note	This element restricts the Client Identifiers to an allowed range of numerical values on the Machine. The ClientId allows to differentiate calls from multiple clients located on the same Machine to the same method.			
Base	*ARObject*			
Aggregated by	EcuInstance.clientIdRange, MachineDesign.clientIdRange			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the ClientIdRange. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
upperLimit	Limit	0..1	attr	This specifies the upper limit of the ClientIdRange. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table E.23: ClientIdRange

Class	**ClientServerInterface**			
Note	A client/server interface declares a number of operations that can be invoked on a server by a client. **Tags:** atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
possibleError	ApplicationError	*	aggr	Application errors that are defined as part of this interface. **Tags:** atp.Splitkey=possibleError, possibleError.shortName

Table E.24: ClientServerInterface

Class	***CommConnectorPort*** (abstract)			
Note	The Ecu communication relationship defines which signals, Pdus and frames are actually received and transmitted by this ECU. For each signal, Pdu or Frame that is transmitted or received and used by the Ecu an association between an ISignalPort, IPduPort or FramePort with the corresponding Triggering shall be created. An ISignalPort shall be created only if the corresponding signal is handled by COM (RTE or Signal Gateway). If a Pdu Gateway ECU only routes the Pdu without being interested in the content only a FramePort and an IPduPort needs to be created.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	FramePort, IPduPort, ISignalPort			
Aggregated by	*CommunicationConnector*.ecuCommPortInstance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Direction	Communication DirectionType	0..1	attr	Communication Direction of the Connector Port (input or output Port).

Table E.25: CommConnectorPort

Enumeration	**CommunicationDirectionType**
Note	Describes the communication direction.
Aggregated by	*CommConnectorPort*.communicationDirection, *IEEE1722TpConnection*.communicationDirection, IPSecRule.direction, ISignalIPduGroup.communicationDirection
Literal	***Description***
in	Reception (Input) **Tags:** atp.EnumerationLiteralIndex=0
out	Transmission (Output) **Tags:** atp.EnumerationLiteralIndex=1

Table E.26: CommunicationDirectionType

Class	**CompuConst**			
Note	This meta-class represents the fact that the value of a computation method scale is constant.			
Base	*ARObject*			
Aggregated by	Compu.compuDefaultValue, CompuScale.compuInverseValue, CompuScaleConstantContents.compuConst			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compuConst ContentType	CompuConstContent	0..1	aggr	This is the actual content of the constant compu method scale. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=10 xml.typeElement=false xml.typeWrapperElement=false

Table E.27: CompuConst

Class	**CompuConstTextContent**			
Note	This meta-class represents the textual content of a scale.			
Base	*ARObject*, *CompuConstContent*			
Aggregated by	CompuConst.compuConstContentType			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
vt	VerbatimString	0..1	attr	This represents a textual constant in the computation method.

Table E.28: CompuConstTextContent

Class	CompuMethod			
Note	This meta-class represents the ability to express the relationship between a physical value and the mathematical representation. Note that this is still independent of the technical implementation in data types. It only specifies the formula how the internal value corresponds to its physical pendant. **Tags:** atp.recommendedPackage=CompuMethods			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
compuInternalToPhys	Compu	0..1	aggr	This specifies the computation from internal values to physical values. **Tags:** atp.Splitkey=compuInternalToPhys xml.sequenceOffset=80
compuPhysToInternal	Compu	0..1	aggr	This represents the computation from physical values to the internal values. **Tags:** atp.Splitkey=compuPhysToInternal xml.sequenceOffset=90
displayFormat	DisplayFormatString	0..1	attr	This property specifies, how the physical value shall be displayed e.g. in documents or measurement and calibration tools. **Tags:** xml.sequenceOffset=20
unit	Unit	0..1	ref	This is the physical unit of the Physical values for which the CompuMethod applies. **Tags:** atp.Splitkey=unit xml.sequenceOffset=30

Table E.29: CompuMethod

Class	CompuScale			
Note	This meta-class represents the ability to specify one segment of a segmented computation method.			
Base	*ARObject*			
Aggregated by	CompuScales.compuScale			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
a2lDisplayText	String	0..1	attr	The value of this attribute shall be taken for generating one display text (specifically the OutVal) within the equivalent of the enclosing `CompuMethod` in A2L.
compuInverseValue	CompuConst	0..1	aggr	This is the inverse value of the constraint. This supports the case that the scale is not reversible per se. **Tags:** atp.Splitkey=compuInverseValue xml.sequenceOffset=60
compuScaleContents	CompuScaleContents	0..1	aggr	This represents the computation details of the scale. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=70 xml.typeElement=false xml.typeWrapperElement=false

△

Class	**CompuScale**			
desc	MultiLanguageOverview Paragraph	0..1	aggr	<desc> represents a general but brief description of the object in question. **Tags:** atp.Splitkey=desc xml.sequenceOffset=30
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=40
mask	PositiveUnlimitedInteger	0..1	attr	In difference to all the other computational methods every COMPU-SCALE will be applied including the bit MASK. Therefore it is allowed for this type of COMPU-METHOD, that COMPU-SCALES overlap. To calculate the string reverse to a value, the string has to be split and the according value for each substring has to be summed up. The sum is finally transmitted. The processing has to be done in order of the COMPU-SCALE elements. **Tags:** xml.sequenceOffset=35
shortLabel	Identifier	0..1	attr	This element specifies a short name for the particular scale. The name can for example be used to derive a programming language identifier. **Tags:** xml.sequenceOffset=20
symbol	CIdentifier	0..1	attr	The symbol, if provided, is used by code generators to get a C identifier for the CompuScale. The name will be used as is for the code generation, therefore it needs to be unique within the generation context. **Tags:** xml.sequenceOffset=25
upperLimit	Limit	0..1	attr	This specifies the upper limit of a of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=50

Table E.30: CompuScale

Class	ConstantReference			
Note	Instead of defining this value inline, a constant is referenced.			
Base	*ARObject*, *ValueSpecification*			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValue Specification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, ConstantSpecification.valueSpec, CryptoService Key.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElement Condition.compareValue, DiagnosticEnvSovdDataCondition.compareValue, FieldSenderComSpec.init Value, ISignal.initValue, ISignal.timeoutSubstitutionValue, NonqueuedReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, Nv ProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.init Value, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireCom Spec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, Port DefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalidValue, VariableData Prototype.initValue			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
constant	ConstantSpecification	0..1	ref	The referenced constant. **Tags:** atp.Splitkey=constant

Table E.31: ConstantReference

Class	ConsumedServiceInstance			
Note	Service instances that are consumed by the ECU that is connected via the ApplicationEndpoint to a CommunicationConnector.			
Base	*ARObject*, *AbstractServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ServiceInstanceCollectionSet.serviceInstance			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
allowedService Provider	NetworkEndpoint	*	ref	NetworkEndpoint on which the ProvidedServiceInstance that is communicating with this ConsumedService Instance is allowed to be located so that the ACL check in the ServiceDiscovery is successful and the connection is allowed to be established. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=allowedServiceProvider, allowedService Provider.networkEndpoint, allowedService Provider.variationPoint.shortLabel atp.Status=draft vh.latestBindingTime=postBuild
autoRequire	Boolean	0..1	attr	Defines that this ConsumedServiceInstance shall be required (searched for) by the service discovery at ECU start.
blocklisted Version	SomeipServiceVersion	*	aggr	Collection of blocklisted versions **Tags:** atp.Splitkey=blocklistedVersion atp.Status=draft
consumedEvent Group	ConsumedEventGroup	*	aggr	Selection of event-groups the consumer wants to subscribe for. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=consumedEventGroup, consumedEvent Group.shortName, consumedEventGroup.variation Point.shortLabel vh.latestBindingTime=postBuild

△

Class	**ConsumedServiceInstance**			
eventMulticast Subscription Address	ApplicationEndpoint	0..1	ref	Multicast Address that is used by the client to subscribe to the server: This enables the multicast subscription feature. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=eventMulticastSubscriptionAddress, event MulticastSubscriptionAddress.applicationEndpoint, event MulticastSubscriptionAddress.variationPoint.shortLabel vh.latestBindingTime=postBuild
instance Identifier	AnyServiceInstanceId	0..1	attr	This attribute represents the ability to describe the required service instance ID.
localUnicast Address	ApplicationEndpoint	0..2	ref	The local address over which the CSI is consumed (udp, tcp or both). **Stereotypes:** atpVariation **Tags:** atp.Splitkey=localUnicastAddress, localUnicast Address.applicationEndpoint, localUnicast Address.variationPoint.shortLabel vh.latestBindingTime=postBuild
minorVersion	AnyVersionString	0..1	attr	Minor Version of the ServiceInterface. Value can be set to a number that represents the Minor Version of the searched service or to ANY.
remoteUnicast Address	ApplicationEndpoint	0..2	ref	This reference defines the remote address where the service provider is located. This reference shall ONLY be used if the remote address is determined from the configuration and not at runtime from the Service Discovery. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=remoteUnicastAddress, remoteUnicast Address.applicationEndpoint, remoteUnicast Address.variationPoint.shortLabel vh.latestBindingTime=postBuild
sdClientTimer Config	SomeipSdClientService InstanceConfig	0..1	ref	Client specific configuration settings relevant for the SOME/IP service discovery. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=sdClientTimerConfig, sdClientTimer Config.someipSdClientServiceInstanceConfig, sdClient TimerConfig.variationPoint.shortLabel vh.latestBindingTime=postBuild
serviceIdentifier	PositiveInteger	0..1	attr	This attribute represents the ability to describe the SOME/IP service ID that is searched.
versionDriven FindBehavior	ServiceVersion AcceptanceKindEnum	0..1	attr	Defines the service discovery find behavior. **Tags:** atp.Status=draft

Table E.32: ConsumedServiceInstance

Class	**CouplingElement**
Note	A CouplingElement is used to connect EcuInstances to the VLAN of an EthernetCluster. Coupling Elements can reach from a simple hub to a complex managed switch or even devices with functionalities in higher layers. A CouplingElement that is not related to an EcuInstance occurs as a dedicated single device. **Tags:** atp.recommendedPackage=CouplingElements

▽

△

Class	CouplingElement			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
communication Cluster	EthernetCluster	0..1	ref	This relationship defines to which cluster the Coupling Element belongs. **Tags:** atp.Splitkey=communicationCluster
couplingPort	CouplingPort	*	aggr	Hardware Port of the CouplingElement that is used to connect this CouplingPort to EcuInstances or other CouplingElements. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=couplingPort, couplingPort.shortName, couplingPort.variationPoint.shortLabel vh.latestBindingTime=postBuild
couplingType	CouplingElementEnum	0..1	attr	Describes the coupling type of this CouplingElement.
ecuInstance	EcuInstance	0..1	ref	Optional reference to the ECU where the Coupling Element is located. **Tags:** atp.Splitkey=ecuInstance
firewallRule	StateDependentFirewall	*	ref	Firewall rules defined in the context of a Coupling Element. **Tags:** atp.Splitkey=firewallRule
switchMac Address LearningMode	SwitchMacAddress LearningEnum	0..1	attr	Defines the MAC address learning mode of the Ethernet switch.

Table E.33: CouplingElement

Class	CouplingPort			
Note	A CouplingPort is used to connect a CouplingElement with an EcuInstance or two CouplingElements with each other via a CouplingPortConnection. Optionally, the CouplingPort may also have a reference to a macMulticastGroup and a defaultVLAN.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	CouplingElement.couplingPort, EthernetCommunicationController.couplingPort			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
connection Negotiation Behavior	EthernetConnection NegotiationEnum	0..1	attr	Specifies the connection negotiation of the CouplingPort.
couplingPort Details	CouplingPortDetails	0..1	aggr	Defines more details of a CouplingPort in case a more specific configuration is required. **Tags:** atp.Splitkey=couplingPortDetails
couplingPort Role	CouplingPortRoleEnum	0..1	attr	Defines the role this CouplingPort takes in the context of the CouplingElement.

▽

△

Class	CouplingPort			
defaultVlan	EthernetPhysical Channel	0..1	ref	The vLanIdentifier of the referenced VLAN is the Default-PVID (port VLAN ID). A Port VLAN ID is a default VLAN ID that is assigned to an access CouplingPort to designate the VLAN segment to which this port is connected. Also, if a CouplingPort has not been configured with any VLAN memberships, the virtual switch's Port VLAN ID (pvid) becomes the default VLAN ID for the ports connection. This identifier/tag is added for incoming untagged messages at the port (ingress tagging). For outgoing messages with this identifier, the tag is removed at the port (egress untagging, depending on the Vlan Membership.sendActivity). **Tags:** atp.Splitkey=defaultVlan
macAddress VlanAssignment	MacAddressVlan Membership	*	aggr	Statically defines the assignment of MAC-Multicast-Addresses, optionally together with VLANs, to this CouplingPort. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=macAddressVlanAssignment, macAddress VlanAssignment.shortName, macAddressVlan Assignment.variationPoint.shortLabel vh.latestBindingTime=postBuild
macLayerType	EthernetMacLayerType Enum	0..1	attr	Specifies the mac layer type of the CouplingPort.
macSecProps	MacSecProps	*	aggr	Properties to configure MACsec (Media access control security) and the MKA (MACsec Key Agreement) for the CouplingPort (PHY). **Tags:** atp.Splitkey=macSecProps atp.Status=candidate
physicalLayer Type	EthernetPhysicalLayer TypeEnum	0..1	attr	Specifies the physical layer type of the CouplingPort.
plcaProps	PlcaProps	0..1	aggr	Optional properties for configuration of PLCA (Physical Layer Collision Avoidance) in case 10-BASE-T1S Ethernet is used and PLCA is enabled on the Coupling Port (PHY). **Tags:** atp.Splitkey=plcaProps
pncMapping	PncMappingIdent	*	ref	Reference to the partial networks this CouplingPort participates in. **Tags:** atp.Splitkey=pncMapping
receiveActivity	EthernetSwitchVlan IngressTagEnum	0..1	attr	Defines the handling of frames at the ingress port.
vlan Membership	VlanMembership	*	aggr	Messages of VLANs that are defined here can be communicated via the CouplingPort. **Tags:** atp.Splitkey=vlanMembership
wakeupSleep OnDataline Config	EthernetWakeupSleep OnDatalineConfig	0..1	ref	Optional reference to EthernetWakeupSleepOnDataline Config. **Tags:** atp.Splitkey=wakeupSleepOnDatalineConfig

Table E.34: CouplingPort

Class	CouplingPortConnection			
Note	Connection between two CouplingPorts (firstPort and secondPort) or between a collection of Ports that are all referenced by the portCollection reference.			
Base	*ARObject*			
Aggregated by	EthernetCluster.couplingPortConnection			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
firstPort	CouplingPort	0..1	ref	Reference to the first CouplingPort that is connected via the CouplingPortConnection. **Tags:** atp.Splitkey=firstPort
nodePort	CouplingPort	*	ref	Reference to a number of CouplingPorts that are connected via the CouplingPortConnection. This reference shall be used to describe a 10BASE-T1S topology architecture where several CouplingPorts of EthernetCommunicationControllers are connected via one CouplingPortConnection. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=nodePort, nodePort.couplingPort, node Port.variationPoint.shortLabel vh.latestBindingTime=postBuild
plcaLocalNode Count	PositiveInteger	0..1	attr	Defines the number of communication participants in case 10BASE-T1S and the nodePort reference is used.
plcaTransmit Opportunity Timer	PositiveInteger	0..1	attr	Timer for the transmission in bit time to evaluate if a Transmission Opportunity is yield or not.
secondPort	CouplingPort	0..1	ref	Reference to the second CouplingPort that is connected via the CouplingPortConnection. **Tags:** atp.Splitkey=secondPort

Table E.35: CouplingPortConnection

Class	DataConstr			
Note	This meta-class represents the ability to specify constraints on data. **Tags:** atp.recommendedPackage=DataConstrs			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dataConstrRule	DataConstrRule	*	aggr	This is one particular rule within the data constraints. **Tags:** atp.Splitkey=dataConstrRule xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

Table E.36: DataConstr

Class	DataConstrRule			
Note	This meta-class represents the ability to express one specific data constraint rule.			
Base	*ARObject*			
Aggregated by	DataConstr.dataConstrRule			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**DataConstrRule**			
constrLevel	Integer	0..1	attr	This attribute describes the category of a constraint. One of its functions is in the area of constraint violation, where it can be used from a certain level, to produce error messages. The lower the level, the more stringent the check. Used to distinguish hard or soft limits. **Tags:** xml.sequenceOffset=20
internalConstrs	InternalConstrs	0..1	aggr	Describes the limitations applicable on the internal domain (as opposed to the physical domain). **Tags:** atp.Splitkey=internalConstrs xml.sequenceOffset=40
physConstrs	PhysConstrs	0..1	aggr	Describes the limitations applicable on the physical domain (as opposed to the internal domain). **Tags:** atp.Splitkey=physConstrs xml.sequenceOffset=30

Table E.37: DataConstrRule

Class	**DataFilter**			
Note	Base class for data filters. The type of the filter is specified in attribute dataFilterType. Some of the filter types require additional arguments which are specified as attributes of this class.			
Base	*ARObject*			
Aggregated by	ISignalPort.dataFilter, NonqueuedReceiverComSpec.filter, NonqueuedSenderComSpec.dataFilter, SignalBasedEventElementToISignalTriggeringMapping.filter, SignalBasedFieldToISignalTriggering Mapping.filter, SignalServiceTranslationElementProps.filter, TransmissionModeCondition.dataFilter			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dataFilterType	DataFilterTypeEnum	0..1	attr	This attribute specifies the type of the filter.
mask	UnlimitedInteger	0..1	attr	Mask for old and new value.
max	UnlimitedInteger	0..1	attr	Value to specify the upper boundary
min	UnlimitedInteger	0..1	attr	Value to specify the lower boundary
offset	PositiveInteger	0..1	attr	Specifies the initial number of messages to occur before the first message is passed
period	PositiveInteger	0..1	attr	Specifies number of messages to occur before the message is passed again
x	UnlimitedInteger	0..1	attr	Value to compare with

Table E.38: DataFilter

Enumeration	**DataFilterTypeEnum**
Note	This enum specifies the supported DataFilterTypes.
Aggregated by	DataFilter.dataFilterType
Literal	*Description*
always	No filtering is performed so that the message always passes. **Tags:** atp.EnumerationLiteralIndex=0

▽

△

Enumeration	**DataFilterTypeEnum**
maskedNewDiffersMaskedOld	Pass messages where the masked value has changed. (new_value&mask) !=(old_value&mask) new_value: current value of the message old_value: last value of the message (initialized with the initial value of the message, updated with new_value if the new message value is not filtered out) **Tags:** atp.EnumerationLiteralIndex=1
maskedNewDiffersX	Pass messages whose masked value is not equal to a specific value x (new_value&mask) != x new_value: current value of the message **Tags:** atp.EnumerationLiteralIndex=2
maskedNewEqualsX	Pass messages whose masked value is equal to a specific value x (new_value&mask) == x new_value: current value of the message **Tags:** atp.EnumerationLiteralIndex=3
never	The filter removes all messages. **Tags:** atp.EnumerationLiteralIndex=4
newIsOutside	Pass a message if its value is outside a predefined boundary. (min > new_value) OR (new_value > max) **Tags:** atp.EnumerationLiteralIndex=5
newIsWithin	Pass a message if its value is within a predefined boundary. min <= new_value <= max **Tags:** atp.EnumerationLiteralIndex=6
oneEveryN	Pass a message once every N message occurrences. Algorithm: occurrence % period == offset Start: occurrence = 0. Each time the message is received or transmitted, occurrence is incremented by 1 after filtering. Length of occurrence is 8 bit (minimum). **Tags:** atp.EnumerationLiteralIndex=7

Table E.39: DataFilterTypeEnum

Class	***DataPrototype*** (abstract)			
Note	Base class for prototypical roles of any data type.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*ApplicationCompositeElementDataPrototype*, *AutosarDataPrototype*			
Aggregated by	*AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
swDataDefProps	SwDataDefProps	0..1	aggr	This property allows to specify data definition properties which apply on data prototype level. **Tags:** atp.Splitkey=swDataDefProps

Table E.40: DataPrototype

Class	***DataPrototypeReference*** (abstract)			
Note	This meta-class provides the ability to reference a DataPrototype.			
Base	*ARObject*			
Subclasses	DataPrototypeInPortInterfaceRef, ImplementationDataTypeElementInPortInterfaceRef			
Aggregated by	DataPrototypeTransformationProps.dataPrototypeInPortInterfaceRef, SignalServiceTranslationElement Props.element, TransmissionComSpecProps.onChangeDataPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.41: DataPrototypeReference

Class	**DataTransformation**			
Note	A DataTransformation represents a transformer chain. It is an ordered list of transformers.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DataTransformationSet.dataTransformation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
data Transformation Kind	DataTransformationKind Enum	0..1	attr	This attribute controls the kind of DataTransformation to be applied.
executeDespite Data Unavailability	Boolean	0..1	attr	Specifies whether the transformer chain is executed even if no input data are available.
transformer Chain (ordered)	Transformation Technology	*	ref	This attribute represents the definition of a chain of transformers that are supposed to be executed according to the order of being referenced from DataTransformation.

Table E.42: DataTransformation

Class	***DiagnosticAbstractParameter*** (abstract)			
Note	This meta-class represents an abstract base class for modeling a diagnostic parameter.			
Base	*ARObject*			
Subclasses	DiagnosticParameter, DiagnosticParameterElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bitOffset	PositiveInteger	0..1	attr	This represents the bitOffset of the DiagnosticParameter. The value of the bitOffset shall always be interpreted as relative to the start of the enclosing DiagnosticData Identifier, DiagnosticParameterIdentifier, or Diagnostic RoutineSubfunction. **Stereotypes:** atpIdentityContributor **Tags:** atp.Status=candidate
dataElement	DiagnosticDataElement	0..1	aggr	This represents the related dataElement of the Diagnostic Parameter **Stereotypes:** atpVariation **Tags:** atp.Splitkey=dataElement, dataElement.shortName, data Element.variationPoint.shortLabel vh.latestBindingTime=postBuild
parameterSize	PositiveInteger	0..1	attr	This attribute allows for the specification of the parameter size. This information is relevant if there is a gap between one diagnostic parameter and the following diagnostic parameter (or the tail of the telegram). The unit is bit and the values shall be multiples of 8. **Tags:** atp.Status=candidate

Table E.43: DiagnosticAbstractParameter

Class	DiagnosticAccessPermission			
Note	This represents the specification of whether a given service can be accessed according to the existence of meta-classes referenced by a particular DiagnosticAccessPermission. In other words, this meta-class acts as a mapping element between several (otherwise unrelated) pieces of information that are put into context for the purpose of checking for access rights. **Tags:** atp.recommendedPackage=DiagnosticAccessPermissions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
authenticationEnabled Proxy	DiagnosticAuthRole	0..1	aggr	The existence of this aggregation indicates that an authentication is foreseen. The details are clarified by the aggregated class. **Tags:** atp.Splitkey=authenticationEnabled
diagnosticSession	DiagnosticSession	*	ref	This represents the associated DiagnosticSessions **Tags:** atp.Splitkey=diagnosticSession
environmentalCondition	DiagnosticEnvironmentalCondition	0..1	ref	This represents the environmental conditions associated with the access permission. **Tags:** atp.Splitkey=environmentalCondition
securityLevel	DiagnosticSecurityLevel	*	ref	This represents the associated DiagnosticSecurityLevels **Tags:** atp.Splitkey=securityLevel
sovdLock	DiagnosticSovdLock	0..1	ref	This represents the associated SOVD lock. **Tags:** atp.Splitkey=sovdLock This Attribute is only used by the AUTOSAR Adaptive Platform.

Table E.44: DiagnosticAccessPermission

Class	DiagnosticAuthTransmitCertificate			
Note	This meta-class represents the sub-function to transmit a certificate **Tags:** atp.recommendedPackage=DiagnosticAuthTransmitCertificates			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticAuthentication*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
certificateEvaluation	DiagnosticAuthTransmitCertificateEvaluation	*	aggr	This aggregation represents a collection of certificate evaluation configurations. **Tags:** atp.Splitkey=certificateEvaluation, certificateEvaluation.shortName

Table E.45: DiagnosticAuthTransmitCertificate

Class	DiagnosticAuthTransmitCertificateEvaluation			
Note	This meta-class represents the ability to configure a certificate evaluation in the context of a diagnostic authentication.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DiagnosticAuthTransmitCertificate.certificateEvaluation			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
evaluationId	PositiveInteger	0..1	attr	This attributes represents the ID of the certificate evaluation.

Table E.46: DiagnosticAuthTransmitCertificateEvaluation

Class	**DiagnosticCondition** (abstract)			
Note	Abstract element for StorageConditions and EnableConditions.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticClearCondition, DiagnosticEnableCondition, DiagnosticStorageCondition			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
initValue	Boolean	0..1	attr	Defines the initial status for enable or disable of acceptance/storage of event reports of a diagnostic event. The value is the initialization after power up (before this condition is reported the first time). true: acceptance/storage of a diagnostic event enabled false: acceptance/storage of a diagnostic event disabled

Table E.47: DiagnosticCondition

Class	**DiagnosticContributionSet**			
Note	This meta-class represents a root node of a diagnostic extract. It bundles a given set of diagnostic model elements. The granularity of the DiagonsticContributionSet is arbitrary in order to support the aspect of decentralized configuration, i.e. different contributors can come up with an own DiagnosticContribution Set. **Tags:** atp.recommendedPackage=DiagnosticContributionSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
common Properties	DiagnosticCommon Props	0..1	aggr	This attribute represents a collection of diagnostic properties that are shared among the entire Diagnostic ContributionSet. **Tags:** atp.Splitkey=commonProperties
element	DiagnosticCommon Element	*	ref	This represents a DiagnosticCommonElement considered in the context of the DiagnosticContributionSet **Stereotypes:** atpVariation **Tags:** atp.Splitkey=element, element.diagnosticCommon Element, element.variationPoint.shortLabel vh.latestBindingTime=postBuild
serviceTable	DiagnosticServiceTable	*	ref	This represents the collection of DiagnosticServiceTables to be considered in the scope of this Diagnostic ContributionSet. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=serviceTable, serviceTable.diagnosticService Table, serviceTable.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table E.48: DiagnosticContributionSet

Class	**DiagnosticDataTransfer**
Note	This represents an instance of the "Data Transfer" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticMemoryByAdresss
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMemoryByAddress*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

△

Class	DiagnosticDataTransfer			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
dataTransfer Class	DiagnosticDataTransfer 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 DiagnosticDataTransfer in the given context. **Tags:** atp.Splitkey=dataTransferClass

Table E.49: DiagnosticDataTransfer

Class	DiagnosticEdrDataProviderMapping			
Note	This meta-class represents the definition of a data provider for the content of an extended data record. **Tags:** atp.recommendedPackage=DiagnosticDataProviders			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
dataProvider	NameToken	0..1	attr	This attribute defines the kind of data that is associated with the referenced DiagnosticExtendedDataRecord Element.
extendedData RecordElement	DiagnosticExtended DataRecordElement	0..1	ref	This reference identifies the DiagnosticExtendedData RecordElement to which the data provider is associated with. **Tags:** atp.Splitkey=extendedDataRecordElement

Table E.50: DiagnosticEdrDataProviderMapping

Class	DiagnosticExtendedDataRecord			
Note	Description of an extended data record. **Tags:** atp.recommendedPackage=DiagnosticExtendedDataRecords			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
customTrigger	String	0..1	attr	This attribute shall be taken to verbally describe the nature of the custom trigger.
element (ordered)	DiagnosticExtended DataRecordElement	*	aggr	This aggregation represents the collection of elements of the enclosing Extentded Data Record.
recordNumber	PositiveInteger	0..1	attr	This attribute specifies an unique identifier for an extended data record.
trigger	DiagnosticRecord TriggerEnum	0..1	attr	This attribute specifies the primary trigger to allocate an event memory entry.
update	Boolean	0..1	attr	This attribute defines when an extended data record is captured. true: This extended data record is captured every time. false: This extended data record is only captured for new event memory entries.

Table E.51: DiagnosticExtendedDataRecord

Class	DiagnosticExtendedDataRecordElement			
Note	This meta-class represents an element of an extended data record. Such an element represents a primitive data object or a fixed-size array of primitive data objects without further internal structure (no structure allowed).			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DiagnosticExtendedDataRecord.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
numberOf Elements	PositiveInteger	0..1	attr	The existence of this attribute turns the data instance into a fixed-size array of data. The attribute determines the size of the array in terms of how many elements the array shall take. The aggregation of SwDataDefProps (especially with respect to SwBaseType) applies specifically to an element of the array (as a representation of all array elements) instead of the entire array itself!
swDataDef Props	SwDataDefProps	0..1	aggr	This aggregation allows for the specification of various properties for the enclosing element. **Tags:** atp.Splitkey=swDataDefProps

Table E.52: DiagnosticExtendedDataRecordElement

Class	DiagnosticIndicator			
Note	Definition of an indicator. **Tags:** atp.recommendedPackage=DiagnosticIndicators			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
type	DiagnosticIndicatorTypeEnum	0..1	attr	Defines the type of the indicator. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table E.53: DiagnosticIndicator

Class	***DiagnosticMapping*** (abstract)			
Note	Abstract element for different kinds of diagnostic mappings.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticEdrDataProviderMapping, DiagnosticEventToDebounceAlgorithmMapping, DiagnosticEventToEnableConditionGroupMapping, DiagnosticEventToOperationCycleMapping, DiagnosticEventToSecurityEventMapping, DiagnosticEventToTroubleCodeUdsMapping, DiagnosticFimAliasEventGroupMapping, DiagnosticFimAliasEventMapping, DiagnosticInhibitSourceEventMapping, DiagnosticMasterToSlaveEventMapping, DiagnosticProvidedDataMapping, DiagnosticSecureCodingMapping, DiagnosticSovdConfigContentMapping, *DiagnosticSwMapping*, DiagnosticTroubleCodeUdsToClearConditionGroupMapping, DiagnosticTroubleCodeUdsToTroubleCodeObdMapping			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table E.54: DiagnosticMapping

Class	**DiagnosticParameter**			
Note	This meta-class represents the ability to describe information relevant for the execution of a specific diagnostic service, i.e. it can be taken to parameterize the service.			
Base	*ARObject*, *DiagnosticAbstractParameter*			
Aggregated by	DiagnosticDataIdentifier.dataElement, DiagnosticInfoType.dataElement, DiagnosticParameterIdentifier.dataElement, DiagnosticRequestRoutineResults.request, DiagnosticRequestRoutineResults.response, DiagnosticStartRoutine.request, DiagnosticStartRoutine.response, DiagnosticStopRoutine.request, DiagnosticStopRoutine.response			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ident	DiagnosticParameterIdent	0..1	aggr	The aggregation in the role ident provides the ability to make the DiagnosticAbstractParameter identifiable. From the semantical point of view, the AbstractDiagnosticParameter is considered a first-class Identifiable and therefore the aggregation in the role ident shall always exist (until it may be possible to let AbstractDiagnosticParameter directly inherit from Identifiable). **Stereotypes:** atpIdentityContributor
supportInfo	DiagnosticParameterSupportInfo	0..1	aggr	This attribute represents the ability to define which bit of the support info byte is representing this part of the PID. **Tags:** atp.Splitkey=supportInfo

Table E.55: DiagnosticParameter

Class	***DiagnosticServiceInstance*** (abstract)			
Note	This represents a concrete instance of a diagnostic service.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*DiagnosticAuthentication*, DiagnosticClearDiagnosticInformation, DiagnosticClearResetEmissionRelatedInfo, DiagnosticComControl, DiagnosticControlDTCSetting, DiagnosticCustomServiceInstance, *DiagnosticDataByIdentifier*, DiagnosticDynamicallyDefineDataIdentifier, DiagnosticEcuReset, DiagnosticIOControl, *DiagnosticMemoryByAddress*, DiagnosticReadDTCInformation, DiagnosticReadDataByPeriodicID, DiagnosticRequestControlOfOnBoardDevice, DiagnosticRequestCurrentPowertrainData, DiagnosticRequestEmissionRelatedDTC, DiagnosticRequestEmissionRelatedDTCPermanentStatus, DiagnosticRequestFileTransfer, DiagnosticRequestOnBoardMonitoringTestResults, DiagnosticRequestPowertrainFreezeFrameData, DiagnosticRequestVehicleInfo, DiagnosticResponseOnEvent, DiagnosticRoutineControl, DiagnosticSecurityAccess, DiagnosticSessionControl			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
accessPermission	DiagnosticAccessPermission	0..1	ref	This represents the collection of DiagnosticAccess Permissions that allow for the execution of the referencing DiagnosticServiceInstance.. **Tags:** atp.Splitkey=accessPermission
serviceClass	DiagnosticServiceClass	0..1	ref	This represents the corresponding "class", i.e. this meta-class provides properties that are shared among all instances of applicable sub-classes of DiagnosticServiceInstance. The subclasses that affected by this pattern implement references to the applicable "class"-role that substantiate this abstract reference. **Stereotypes:** atpAbstract

Table E.56: DiagnosticServiceInstance

Class	DiagnosticTransferExit			
Note	This represents an instance of the "Transfer Exit" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticMemoryByAdresss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMemoryByAddress*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
transferExit Class	DiagnosticTransferExit 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 DiagnosticTransferExit in the given context. **Tags:** atp.Splitkey=transferExitClass

Table E.57: DiagnosticTransferExit

Class	DiagnosticTroubleCodeProps			
Note	This element defines common Dtc properties that can be reused by different DTCs. **Tags:** atp.recommendedPackage=DiagnosticTroubleCodePropss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
aging	DiagnosticAging	0..1	ref	Reference to an aging algorithm in case that an aging/ unlearning of the event is allowed. **Tags:** atp.Splitkey=aging
diagnostic Memory	DiagnosticMemory Destination	0..1	ref	Reference to the applicable DiagnosticMemory Destination. **Tags:** atp.Splitkey=diagnosticMemory
extendedData Record	DiagnosticExtended DataRecord	*	ref	Defines the links to an extended data class sampler. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=extendedDataRecord, extendedData Record.diagnosticExtendedDataRecord, extendedData Record.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
freezeFrame	DiagnosticFreezeFrame	*	ref	Define the links to a freeze frame class sampler. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=freezeFrame, freezeFrame.diagnosticFreeze Frame, freezeFrame.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
immediateNv DataStorage	Boolean	0..1	attr	Change description for Class immediateNvDataStorage in table "Table A.111: DiagnosticTroubleCodeProps": Switch to enable immediate storage triggering of an according event memory entry persistently to NVRAM. true: immediate non-volatile storage triggering on first occurrence and shutdown. false: immediate non-volatile storage triggering on shutdown.

▽

△

Class	**DiagnosticTroubleCodeProps**			
legislated FreezeFrame ContentUdsObd	DiagnosticDataIdentifier Set	0..1	ref	This reference identifies the layout of legislated freeze frames used for emission related diagnostics over the UDS protocol such as OBDonUDS or WWH-OBD. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=legislatedFreezeFrameContentUdsObd, legislatedFreezeFrameContentUdsObd.diagnosticData IdentifierSet, legislatedFreezeFrameContentUds Obd.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
legislatedSovd FreezeFrame Content	DiagnosticSovdContent Group	0..1	ref	This reference identifies the layout of legislated freeze frames used for emission related diagnostics over the SOVD protocol. **Tags:** atp.Splitkey=legislatedSovdFreezeFrameContent This Attribute is only used by the AUTOSAR Adaptive Platform.
maxNumber FreezeFrame Records	PositiveInteger	0..1	attr	This attribute defines the number of according freeze frame records, which can maximal be stored for this event. Therefore all these freeze frame records have the same freeze frame class.
obdRelevance	Boolean	0..1	attr	The attribute controls whether the enclosing DTC is relevant for OBD. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
priority	PositiveInteger	0..1	attr	Priority of the event, in view of full event buffer. A lower value means higher priority. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
significance	DiagnosticSignificance Enum	0..1	attr	Significance of the event, which indicates additional information concerning fault classification and resolution.
snapshot RecordContent	DiagnosticDataIdentifier Set	0..1	ref	This represents the freeze frame layout as a set of DIDs. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=snapshotRecordContent, snapshotRecord Content.diagnosticDataIdentifierSet, snapshotRecord Content.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
sovdSnapshot Content	DiagnosticSovdContent Group	0..1	ref	This represents the freeze frame layout as a set of SOVD Content. **Tags:** atp.Splitkey=sovdSnapshotContent This Attribute is only used by the AUTOSAR Adaptive Platform.

Table E.58: DiagnosticTroubleCodeProps

Class	**DiagnosticTroubleCodeUds**			
Note	This element is used to describe diagnostic trouble codes (DTCs). **Tags:** atp.recommendedPackage=DiagnosticTroubleCodes			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticTroubleCode*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**DiagnosticTroubleCodeUds**			
considerPto Status	Boolean	0..1	attr	This attribute describes the affection of the event by the Dem PTO handling. true: the event is affected by the Dem PTO handling. false: the event is not affected by the Dem PTO handling. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
eventReadiness Group	EventObdReadiness Group	0..1	aggr	This attribute specifies the Event OBD Readiness group for PID $01 and PID $41 computation. This attribute is only applicable for emission-related ECUs. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=eventReadinessGroup, eventReadiness Group.eventObdReadinessGroup, eventReadiness Group.variationPoint.shortLabel vh.latestBindingTime=postBuild
functionalUnit	PositiveInteger	0..1	attr	This attribute specifies a 1-byte value which identifies the corresponding basic vehicle / system function which reports the DTC. This parameter is necessary for the report of severity information. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
obdDtcValue3 Byte	PositiveInteger	0..1	attr	3 Byte OBD DTC value based on the definition from SAE J2012. The existence of this attribute is only required if separated UDS and OBD DTC values are used for SAE J1979-2. If this attribute does not exist, then UDS DTC values are used with J1979-2. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
severity	DiagnosticUdsSeverity Enum	0..1	attr	DTC severity according to ISO 14229-1. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
troubleCode Props	DiagnosticTroubleCode Props	0..1	ref	Defined properties associated with the DemDTC. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=troubleCodeProps, troubleCode Props.diagnosticTroubleCodeProps, troubleCode Props.variationPoint.shortLabel vh.latestBindingTime=postBuild
udsDtcValue	PositiveInteger	0..1	attr	Unique Diagnostic Trouble Code value for UDS. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
wwhObdDtc Class	DiagnosticWwhObdDtc ClassEnum	0..1	attr	This attribute is used to identify (if applicable) the corresponding severity class of an WWH-OBD DTC. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild

Table E.59: DiagnosticTroubleCodeUds

Class	**DltApplication**
Note	This meta-class represents the application from which the log and trace message originates.
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*
Aggregated by	DltEcu.application

▽

△

Class	DltApplication			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
applicationDescription	String	0..1	attr	This attribute can be used to describe the applicationId that is used in the log and trace message in more detail.
applicationId	String	0..1	attr	This attribute identifies the SW-C/BSW module in the log and trace message.
context	DltContext	*	ref	Definition of ContextIds for the Application. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=context, context.dltContext, context.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime

Table E.60: DltApplication

Class	DltContext			
Note	This meta-class represents the Context that groups Log and Trace messages that are generated by an application. **Tags:** atp.recommendedPackage=DltContexts			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
contextDescription	String	0..1	attr	This attribute can be used to describe the contextId that is used in the log and trace message in more detail.
contextId	String	0..1	attr	This attribute is used to group log and trace messages produced by an application to distinguish functionality.
dltMessage	DltMessage	*	ref	Group of Log and Trace messages assigned to the Dlt Context **Stereotypes:** atpVariation **Tags:** atp.Splitkey=dltMessage, dltMessage.dltMessage, dltMessage.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime

Table E.61: DltContext

Class	DltEcu			
Note	This element represents an Ecu or Machine that produces logging and tracing information. **Tags:** atp.recommendedPackage=DltEcus			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
application	DltApplication	*	aggr	Application on DltEcu that provides log or trace data. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=application, application.shortName, application.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime
ecuId	String	0..1	attr	This attribute defines the name of the ECU for use within the Dlt protocol.

Table E.62: DltEcu

Class	**DltMessage**			
Note	This element defines a DltMessage.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	LogAndTraceMessageCollectionSet.dltMessage			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dltArgument (ordered)	DltArgument	*	aggr	Ordered collection of DltArguments in the DltMessage.
messageId	PositiveInteger	0..1	attr	This attribute defines the unique Id for the DltMessage.
messageLine Number	PositiveInteger	0..1	attr	This attribute describes the position in the source file in which this log message was called.
messageSource File	String	0..1	attr	This attribute describes the source file in which this log message was called.
messageType Info	String	0..1	attr	This attribute describes the message Type
privacyLevel	PrivacyLevel	0..1	aggr	The Privacy Level helps to identify the Log and Trace content towards the degree of privacy to it. **Tags:** atp.Splitkey=privacyLevel

Table E.63: DltMessage

Enumeration	**DoIpEidRetrievalEnum**
Note	Enumeration with options to retrieve EID. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	DoIpNetworkConfiguration.eidRetrieval, DoIpNetworkConfigurationDesign.eidRetrieval
Literal	*Description*
eidUseApi	API DiagnosticDoIPEntityIdentification is used to retrieve eid **Tags:** atp.EnumerationLiteralIndex=1
eidUseConfigValue	eid is configured manually by `DoIpFunctionalClusterDesign.eid`. **Tags:** atp.EnumerationLiteralIndex=2
eidUseMac	MAC of the network interface is used as eid **Tags:** atp.EnumerationLiteralIndex=0

Table E.64: DoIpEidRetrievalEnum

Class	**DoIpLogicalAddress**			
Note	This meta-class describes the DoIP logical address. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DoIpFunctionalClusterDesign.doIpLogicalAddress			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
logicalAddress	PositiveInteger	0..1	attr	This attribute holds the DoIP logical address.

Table E.65: DoIpLogicalAddress

Class	**DoIpNetworkConfiguration**
Note	This element collects DoIP properties that are network interface specific. This Class is only used by the AUTOSAR Adaptive Platform.
Base	*ARObject*

▽

△

Class	**DoIpNetworkConfiguration**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eidRetrieval	DoIpEidRetrievalEnum	0..1	attr	This attribute defines how DoIp Entitiy Identification is retrieved.
isActivationLine Dependent	Boolean	0..1	attr	This attribute defines whether the network interface • is started "on-demand" when an activation line is sensed or • is always available.
maxInitial Vehicle Announcement Time	TimeValue	0..1	attr	Upper bound for the time to wait in [s] for sending first vehicle announcement message after IP address assignment. Represents parameter A_DoIP_Announce_ Wait of ISO 13400-2:2019. The value of this timing shall be determined randomly in the closed interval [0..max InitialVehicleAnnouncementTime].
maxTester Connections	PositiveInteger	0..1	attr	Maximum amount of tester connections that shall be maintained at one time before alive check is performed.
network Configuration	PlatformModule EthernetEndpoint Configuration	*	ref	Network configuration (Protocol, Port, IP Address) for transmission of DoIP messages on a specific VLAN. **Tags:** atp.Splitkey=networkConfiguration
network Configuration Design	DoIpNetwork ConfigurationDesign	0..1	ref	Reference to the DoIP network configuration design that this DoIpNetworkConfiguration implements. **Tags:** atp.Splitkey=networkConfigurationDesign
network InterfaceId	PositiveInteger	0..1	attr	This attribute defines the identifier for the DoIPInterface.
tcpAliveCheck Response Timeout	TimeValue	0..1	attr	Timeout in [s] for waiting for a response to an Alive Check request before the connection is considered to be disconnected. Represents parameter T_TCP_AliveCheck of ISO 13400-2:2019.
tcpGeneral InactivityTime	TimeValue	0..1	attr	Timeout in [s] for maximum inactivity of a TCP socket connection before the DoIP module will close the according socket connection. Represents parameter T_ TCP_General_Inactivity of ISO 13400-2:2019.
tcpInitial InactivityTime	TimeValue	0..1	attr	Timeout in [s] used for initial inactivity of a connected TCP socket connection directly after socket connection. Represents parameter T_TCP_Initial_Inactivity of ISO 13400-2:2019.
vehicle Announcement Count	PositiveInteger	0..1	attr	Number of vehicle announcement messages on IP address assignment. Represents parameter A_DoIP_ Announce_Num of ISO 13400-2:2019.
vehicle Announcement Interval	TimeValue	0..1	attr	Time to wait in [s] for sending subsequent vehicle announcement messages. Represents parameter A_Do IP_Announce_Interval of ISO 13400-2:2019.
vehicle Identification SyncStatus	Boolean	0..1	attr	Defines if the optional VIN/GID synchronization status is used additionally in the vehicle identification/ announcement.

Table E.66: DoIpNetworkConfiguration

Class	**DoIpRequestConfiguration**			
Note	This meta-class specifies a range of target addresses and its interpretation as either physical or functional request. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	DoIpRequestConfiguration			
endAddress	PositiveInteger	0..1	attr	End address for range of target-addresses (including this address).
requestType	RequestTypeEnum	0..1	attr	Determines the type of request.
startAddress	PositiveInteger	0..1	attr	Start address for range of target-addresses (including this address).

Table E.67: DoIpRequestConfiguration

Class	EcuInstance			
Note	ECUInstances are used to define the ECUs used in the topology. The type of the ECU is defined by a reference to an ECU specified with the ECU resource description. **Tags:** atp.recommendedPackage=EcuInstances			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
associated ConsumedProvided ServiceInstance Group	ConsumedProvided ServiceInstanceGroup	*	ref	With this reference it is possible to identify which ConsumedProvidedServiceInstanceGroups are applicable for which ECUInstance. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=associatedConsumedProvidedService InstanceGroup, associatedConsumedProvidedService InstanceGroup.consumedProvidedServiceInstanceGroup, associatedConsumedProvidedServiceInstance Group.variationPoint.shortLabel vh.latestBindingTime=postBuild
associatedPdur IPduGroup	PdurIPduGroup	*	ref	With this reference it is possible to identify which PduR IPdu Groups are applicable for which Communication Connector/ ECU. **Tags:** atp.Splitkey=associatedPdurIPduGroup
channel Synchronous Wakeup	Boolean	0..1	attr	If this parameter is available and set to true, then all available channels will be woken up as soon as at least one channel wakeup occurs. If PNCs are configured, then all PNCs will be requested upon a channel wakeup.
clientIdRange	ClientIdRange	0..1	aggr	Restriction of the Client Identifier for this Ecu to an allowed range of numerical values. The Client Identifier of the transaction handle is generated by the client RTE for inter-Ecu Client/Server communication. **Tags:** atp.Splitkey=clientIdRange
commController	Communication Controller	*	aggr	CommunicationControllers of the ECU. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=commController, commController.short Name, commController.variationPoint.shortLabel vh.latestBindingTime=postBuild
connector	Communication Connector	*	aggr	All channels controlled by a single controller. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=connector, connector.shortName, connector.variationPoint.shortLabel vh.latestBindingTime=postBuild

▽

△

Class	**EcuInstance**			
firewallRule	StateDependentFirewall	*	ref	Firewall rules defined in the context of an EcuInstance. **Tags:** atp.Splitkey=firewallRule
pncNmRequest	Boolean	0..1	attr	Defines if this EcuInstance shall request Nm on all its PhysicalChannels which have Nm variant set to FULL each time a PNC is requested.
pncPrepare SleepTimer	TimeValue	0..1	attr	Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP.
pnc Synchronous Wakeup	Boolean	0..1	attr	If this parameter is available and set to true then all available PNCs will be woken up as soon as a channel wakeup occurs. This is ensured by adding all PNCs to all channel wakeup sources during upstream mapping.
pnResetTime	TimeValue	0..1	attr	Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests in the EIRA and in the ERA.

Table E.68: EcuInstance

Class	**EndToEndTransformationDescription**			
Note	EndToEndTransformationDescription holds these attributes which are profile specific and have the same value for all E2E transformers.			
Base	*ARObject*, *Describable*, *TransformationDescription*			
Aggregated by	TransformationTechnology.transformationDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clearFromValid ToInvalid	Boolean	0..1	attr	Clear monitoring window on transition from state Valid to state Invalid.
counterOffset	PositiveInteger	0..1	attr	Offset of the counter in the Data[] array in bits.
crcOffset	PositiveInteger	0..1	attr	Offset of the CRC in the Data[] array in bits.
dataIdMode	DataIdModeEnum	0..1	attr	This attribute describes the inclusion mode that is used to include the implicit two-byte Data ID in the one-byte CRC.
dataIdNibble Offset	PositiveInteger	0..1	attr	Offset of the Data ID nibble in the Data[] array in bits.
e2eProfile Compatibility Props	E2EProfileCompatibility Props	0..1	ref	Reference to additional settings for the E2E state machine. **Tags:** atp.Splitkey=e2eProfileCompatibilityProps
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.
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.
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.
maxNoNewOr RepeatedData	PositiveInteger	0..1	attr	The maximum allowed amount of consecutive failed counter checks.
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.

▽

△

Class	**EndToEndTransformationDescription**			
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.
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.
offset	PositiveInteger	0..1	attr	Offset of the E2E header in the Data[] array in bits.
profileBehavior	EndToEndProfile BehaviorEnum	0..1	attr	Behavior of the check functionality
profileName	NameToken	0..1	attr	Definition of the E2E profile.
syncCounterInit	PositiveInteger	0..1	attr	Number of checks required for validating the consistency of the counter that shall be received with a valid counter (i.e. counter within the allowed lock-in range) after the detection of an unexpected behavior of a received counter.
upperHeader BitsToShift	PositiveInteger	0..1	attr	This attribute describes the number of upper-header bits to be shifted. value = 0 or not present: shift of upper header is NOT performed. value > 0: the E2E Transformer on the protect-side, takes the first upperHeaderBitsToShift bits from the upper buffer (e.g. SOME/IP header part generated by SOME/IP transformer) and shifts them towards the lower bytes and bits within the Data[] for the length of the E2E header (e.g. 12 bytes in case of E2E Profile 4). This means the shift distance is fixed - it depends on the E2E header size - what is configured here is the number of bits that are to be shifted. This option is defined because the Some/IP header generated by SOME/IP transformer shall be, due to compatibility between non-protected and E2 E-protected communication, at the same position, which is before E2E header.
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 E.69: EndToEndTransformationDescription

Class	«atpVariation» **EthernetCommunicationController**			
Note	Ethernet specific communication port attributes.			
Base	*ARObject*, *CommunicationController*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcuInstance.commController, MachineDesign.communicationController			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
canXlConfig	AbstractCan Communication Controller	0..1	ref	If the Ethernet frames handled by this Ethernet CommunicationController are to be tunneled through CAN XL, then this reference shall refer to the Abstract CanCommunicationController that aggregates the Can ControllerXlConfiguration of the physical CAN XL channel to be used for tunneling. **Tags:** atp.Splitkey=canXlConfig

▽

△

Class	«atpVariation» **EthernetCommunicationController**			
couplingPort	CouplingPort	*	aggr	Optional CouplingPort that can be used to connect the ECU to a CouplingElement (e.g. a switch). **Tags:** atp.Splitkey=couplingPort, couplingPort.shortName
macLayerType	EthernetMacLayerType Enum	0..1	attr	Specifies the mac layer type of the ethernet controller.
macUnicast Address	MacAddressString	0..1	attr	Media Access Control address (MAC address) that uniquely identifies each EthernetCommunication Controller in the network.
maximum ReceiveBuffer Length	Integer	0..1	attr	Determines the maximum receive buffer length (frame length) in bytes.
maximum TransmitBuffer Length	Integer	0..1	attr	Determines the maximum transmit buffer length (frame length) in bytes.
slaveActAs Passive Communication Slave	Boolean	0..1	attr	This attribute specifies if the EcuInstance is acting as a passive communication slave on the connected Physical Channel. This is used for EthernetCommunication Controllers that use Ethernet hardware which supports wake-up and sleep on the network (e.g. Open Alliance TC10 compliant Ethernet hardware).
slaveQualified UnexpectedLink DownTime	TimeValue	0..1	attr	This attribute specifies time when an unexpected link down is evaluated as link down and indicated to the AUTOSAR communication stack.

Table E.70: EthernetCommunicationController

Class	**EthernetPhysicalChannel**			
Note	The EthernetPhysicalChannel represents a VLAN or an untagged channel. An untagged channel is modeled as an EthernetPhysicalChannel without an aggregated VLAN.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhysicalChannel*, *Referrable*			
Aggregated by	*CommunicationCluster*.physicalChannel			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
comMLight Timeout	TimeValue	0..1	attr	Defines the timeout (in seconds) after COMM_FULL_ COMMUNICATION sub-state COMM_FULL_COM_ READY_SLEEP is left in the ComM channel state machine.
network Endpoint	NetworkEndpoint	*	aggr	Collection of NetworkEndpoints that are used in the VLan. **Tags:** atp.Splitkey=networkEndpoint, network Endpoint.shortName
vlan	VlanConfig	0..1	aggr	VLAN Configuration. **Tags:** atp.Splitkey=vlan, vlan.shortName

Table E.71: EthernetPhysicalChannel

Class	**ExecutionDependency**			
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. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	StateDependentStartupConfig.executionDependency			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	ExecutionDependency			
processState	ModeDeclaration	0..1	iref	This represent the applicable modeDeclaration that represents an ProcessState. **Tags:** atp.Splitkey=processState, processState.context ModeDeclarationGroupPrototype, processState.target ModeDeclaration **InstanceRef implemented by:** ModeInProcessInstance Ref

Table E.72: ExecutionDependency

Class	GlobalTimeCorrectionProps			
Note	This meta-class defines the attributes for rate and offset correction.			
Base	*ARObject*			
Aggregated by	GlobalTimeDomain.globalTimeCorrectionProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
allowProvider RateCorrection	Boolean	0..1	attr	Defines whether the rate correction value of a Time Base can be set by means of the method setRateCorrection(). false: rate correction cannot be set by method setRate Correction(). true: rate correction can be set by method setRateCorrection().
offsetCorrection AdaptionInterval	TimeValue	0..1	attr	Defines the interval during which the adaptive rate correction cancels out the rate- and time deviation.
offsetCorrection JumpThreshold	TimeValue	0..1	attr	Threshold for the correction method. Deviations below this value will be corrected by a linear reduction over a defined timespan. Values equal- and greater than this value will be corrected by immediately setting the correct time- and rate in form of a jump.
providerRate DeviationMax	PositiveInteger	0..1	attr	This attribute describes the maximum allowed absolute value of the rate deviation value [unit: ppm].
rateCorrection Measurement Duration	TimeValue	0..1	attr	Definition of the time span which is used to calculate the rate deviation.
rateCorrections Per Measurement Duration	PositiveInteger	0..1	attr	Defines the number of simultaneous rate measurements to determine the current rate deviation.
rateCorrection Threshold	PositiveInteger	0..1	attr	This attribute defines the threshold for rate correction calculation to determine whether the absolute value of the calculated rate deviation exceeds the accepted range. The value is given in ppm. A value of 0 deactivates the check.
rateDeviation Measurement Duration	TimeValue	0..1	attr	Time span used to calculate the rate deviation. Unit: seconds.

Table E.73: GlobalTimeCorrectionProps

Class	GlobalTimeDomain
Note	This represents the ability to define a global time domain. **Tags:** atp.recommendedPackage=GlobalTimeDomains
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*
Aggregated by	ARPackage.element

▽

△

Class	**GlobalTimeDomain**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clonedFrom TimeDomain	GlobalTimeDomain	*	ref	Reference to a set of GlobalTimeDomains from which the time information of the reference source GlobalTime Domain is cloned. This allows to model the cloning source of global time information. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=clonedFromTimeDomain, clonedFromTime Domain.globalTimeDomain, clonedFromTime Domain.variationPoint.shortLabel vh.latestBindingTime=postBuild
debounceTime	TimeValue	0..1	attr	Defines the minimum amount of time between two time sync messages are transmitted.
domainId	PositiveInteger	0..1	attr	This represents the ID of the GlobalTimeDomain used in the network messages sent on behalf of global time management. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
gateway	GlobalTimeGateway	*	aggr	A GlobalTimeGateway may exist in the context of a GlobalTimeDomain to actively update the global time information as it is routed from one GlobalTimeDomain to another. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=gateway, gateway.shortName, gateway.variationPoint.shortLabel vh.latestBindingTime=postBuild
globalTime CorrectionProps	GlobalTimeCorrection Props	0..1	aggr	Defintion of attributes for rate and offset correction. **Tags:** atp.Splitkey=globalTimeCorrectionProps
globalTime Domain Property	AbstractGlobalTime DomainProps	0..1	aggr	Additional properties of the GlobalTimeDomain. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=globalTimeDomainProperty, globalTime DomainProperty.variationPoint.shortLabel vh.latestBindingTime=postBuild
globalTime Master	GlobalTimeMaster	0..1	aggr	This represents the single master of a GlobalTime Domain. A GlobalTimeDomain may have no GlobalTime Domain.master, e.g. when it gets its time from a GPS receiver. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=globalTimeMaster, globalTimeMaster.short Name, globalTimeMaster.variationPoint.shortLabel vh.latestBindingTime=postBuild
globalTimeSub Domain	GlobalTimeDomain	*	ref	By this means it is possible to create a hierarchy of sub Domains where one global time domain can declare one or more other global time domains as its subDomains. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=globalTimeSubDomain, globalTimeSub Domain.globalTimeDomain, globalTimeSub Domain.variationPoint.shortLabel vh.latestBindingTime=postBuild
icvFreshness ValueId	PositiveInteger	0..1	attr	This attribute defines the Id of the Freshness Value for the Integrity Check Value (ICV) calculation and verification.

▽

△

Class	GlobalTimeDomain			
icvSecureCom Props	SecOcSecureCom Props	0..1	ref	Reference to a SecureComProps definition to be used for the Integrity Check Value (ICV) calculation and verification. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=icvSecureComProps, icvSecureCom Props.secOcSecureComProps, icvSecureCom Props.variationPoint.shortLabel vh.latestBindingTime=postBuild
maxProgression Mismatch Threshold	TimeValue	0..1	attr	This attribute defines the maximum allowed difference between local time and fallback time of the time base in seconds.
network SegmentId	NetworkSegment Identification	0..1	aggr	Defines the numerical identification of a GlobalTime sub domain. **Tags:** atp.Splitkey=networkSegmentId
pduTriggering	PduTriggering	0..1	ref	This PduTriggering will be taken to transmit the global time information from a GlobalTimeMaster to a the associated GlobalTimeSlaves. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=pduTriggering, pduTriggering.pduTriggering, pduTriggering.variationPoint.shortLabel vh.latestBindingTime=postBuild
slave	GlobalTimeSlave	*	aggr	This represents the collections of slaves of the Global TimeDomain. A GlobalTimeDomain may have no Global TimeDomain.slaves, e.g. when it propagates its time directly to sub domains. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=slave, slave.shortName, slave.variation Point.shortLabel vh.latestBindingTime=postBuild
syncLoss Timeout	TimeValue	0..1	attr	This attribute describes the timeout for the situation that the time synchronization gets lost in the scope of the time domain.

Table E.74: GlobalTimeDomain

Class	*GlobalTimeMaster* (abstract)			
Note	This represents the generic concept of a global time master.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	GlobalTimeCanMaster, GlobalTimeEthMaster, GlobalTimeFrMaster, UserDefinedGlobalTimeMaster			
Aggregated by	GlobalTimeDomain.globalTimeMaster			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
communication Connector	Communication Connector	0..1	ref	The GlobalTimeMaster is bound to the Communication Connector. **Tags:** atp.Splitkey=communicationConnector
immediate ResumeTime	TimeValue	0..1	attr	Defines the minimum time between an "immediate" message and the next periodic message.
isSystemWide GlobalTime Master	Boolean	0..1	attr	If set to TRUE, the GlobalTimeMaster is supposed to act as the root of global time information.
syncPeriod	TimeValue	0..1	attr	This represents the period. Unit: seconds

Table E.75: GlobalTimeMaster

Class	***GlobalTimeSlave*** (abstract)			
Note	This represents the generic concept of a global time slave.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	GlobalTimeCanSlave, GlobalTimeEthSlave, GlobalTimeFrSlave, UserDefinedGlobalTimeSlave			
Aggregated by	GlobalTimeDomain.slave			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Connector	Communication Connector	0..1	ref	The GlobalTimeSlave is bound to the Communication Connector. **Tags:** atp.Splitkey=communicationConnector
followUp TimeoutValue	TimeValue	0..1	attr	Rx timeout for the follow-up message.
timeLeapFuture Threshold	TimeValue	0..1	attr	Defines the maximum allowed positive difference between the current Local Time Base value and a newly received Global Time Base value.
timeLeap HealingCounter	PositiveInteger	0..1	attr	Defines the required number of updates to the Time Base where the time difference to the previous received value has to remain within the bounds of timeLeapFuture Threshold and timeLeapPastThreshold until that Time Base is considered healed.
timeLeapPast Threshold	TimeValue	0..1	attr	Defines the maximum allowed negative difference between the current Local Time Base value and a newly received Global Time Base value.

Table E.76: GlobalTimeSlave

Enumeration	**HandleInvalidEnum**
Note	Strategies of handling the reception of invalidValue.
Aggregated by	InvalidationPolicy.handleInvalid, ISignalPort.handleInvalid
Literal	***Description***
dontInvalidate	Invalidation is switched off. **Tags:** atp.EnumerationLiteralIndex=0
external Replacement	Replace a received invalidValue. The replacement value is sourced from the aggregation in the role replaceWith. **Tags:** atp.EnumerationLiteralIndex=1
keep	The application software is supposed to handle signal invalidation on RTE API level either by Data ReceiveErrorEvent or check of error code on read access. **Tags:** atp.EnumerationLiteralIndex=2
replace	Replace a received invalidValue. The replacement value is specified by the initValue. **Tags:** atp.EnumerationLiteralIndex=3

Table E.77: HandleInvalidEnum

Class	***IEEE1722TpAcfBus*** (abstract)			
Note	Abstract class to define various busses to be transported over a IEEE1722TP ACF connection. **Tags:** atp.Status=candidate			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	IEEE1722TpAcfCan, IEEE1722TpAcfLin			
Aggregated by	IEEE1722TpAcfConnection.acfTransportedBus			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	***IEEE1722TpAcfBus*** (abstract)			
acfPart	IEEE1722TpAcfBusPart	*	aggr	One part transported over IEEE1722Tp channel. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=acfPart, acfPart.shortName, acfPart.variationPoint.shortLabel atp.Status=candidate vh.latestBindingTime=postBuild
busId	PositiveInteger	0..1	attr	Id of the transported bus over the ACF connection.

Table E.78: IEEE1722TpAcfBus

Class	***IEEE1722TpAcfBusPart*** (abstract)			
Note	Definition of one IEEE1722Tp ACF part transported over the IEEE1722Tp channel. **Tags:** atp.Status=candidate			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	IEEE1722TpAcfCanPart, IEEE1722TpAcfLinPart			
Aggregated by	*IEEE1722TpAcfBus*.acfPart			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
collectionTrigger	PduCollectionTrigger Enum	0..1	attr	Defines whether putting this AcfPart to the IEEE1722Tp ACF message triggers immediate sending of the IEEE1722Tp ACF message.

Table E.79: IEEE1722TpAcfBusPart

Class	**IEEE1722TpAcfCan**			
Note	ACF IEEE1722Tp bus used for CAN transport. **Tags:** atp.Status=candidate atp.recommendedPackage=IEEE1722TpConnections			
Base	*ARObject*, *IEEE1722TpAcfBus*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	IEEE1722TpAcfConnection.acfTransportedBus			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
messageType	IEEE1722TpAcfCan MessageTypeEnum	0..1	attr	Definition of the ACF CAN stream message type.

Table E.80: IEEE1722TpAcfCan

Class	**IEEE1722TpAcfCanPart**			
Note	Definition of one CAN part (frame or frame range) transported over the IEEE1722Tp channel. **Tags:** atp.Status=candidate			
Base	*ARObject*, *IEEE1722TpAcfBusPart*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*IEEE1722TpAcfBus*.acfPart			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
canAddressing Mode	CanAddressingMode Type	0..1	attr	Defines whether standard or extended address format shall be used.
canBitRate Switch	Boolean	0..1	attr	Defines whether the bit rate switch bit shall be set.
canFrameTx Behavior	CanFrameTxBehavior Enum	0..1	attr	Defines which CAN protocol shall be used for frame transmission.

▽

△

Class	**IEEE1722TpAcfCanPart**			
canIdentifier	PositiveInteger	0..1	attr	Optional Can Id defined in case the Can Id can not be determined during runtime.
canIdentifier Mask	PositiveInteger	0..1	attr	CAN identifier mask which denotes relevant bits in the CAN Identifier. This attribute defines a CAN Identifier range in an alternative way to canIdentifierRange. It identifies the bits of the configured CAN Identifier that must match the received CAN Identifier.
canIdentifier Range	RxIdentifierRange	0..1	aggr	Definition of the identifier range for IEEE1722Tp ACF Can messages. **Tags:** atp.Splitkey=canIdentifierRange atp.Status=candidate
sdu	PduTriggering	0..1	ref	Reference to the Pdu transported in the IEEE1722Tp channel. **Tags:** atp.Splitkey=sdu atp.Status=candidate

Table E.81: IEEE1722TpAcfCanPart

Class	**IEEE1722TpAcfConnection**			
Note	ACF IEEE1722Tp connection. **Tags:** atp.Status=candidate atp.recommendedPackage=IEEE1722TpConnections			
Base	*ARElement*, *ARObject*, *CollectableElement*, *IEEE1722TpConnection*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
acfMaxTransitTime	TimeValue	0..1	attr	Defines the time offset that is added to the current time at the producer in order to get the "presentation time" (in seconds) when content shall be presented at the consumers.
acfTransportedBus	IEEE1722TpAcfBus	*	aggr	Definition of the transported busses over this ACF connection. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=acfTransportedBus, acfTransportedBus.shortName, acfTransportedBus.variationPoint.shortLabel atp.Status=candidate vh.latestBindingTime=postBuild
collectionThreshold	PositiveInteger	0..1	attr	Defines the size threshold in bytes which, when exceeded, triggers the sending of the IEEE1722Tp ACF message, even when the maxium IEEE1722Tp ACF message size has not been reached yet.
collectionTimeout	TimeValue	0..1	attr	When this timeout expires the IEEE1722Tp ACF message is triggered for sending. The respective timer is started when the first Pdu is put into the IEEE1722Tp ACF message. Defined in seconds.
mixedBusTypeCollection	Boolean	0..1	attr	Defines if this ACF-stream is allowed to collect ACF-messages of different bus kinds (i.e. whether it is allowed to collect CAN and LIN ACF-messages in one ACF-stream message).

Table E.82: IEEE1722TpAcfConnection

Class	IEEE1722TpAcfLin			
Note	ACF IEEE1722Tp bus used for LIN transport. **Tags:** atp.Status=candidate atp.recommendedPackage=IEEE1722TpConnections			
Base	*ARObject*, *IEEE1722TpAcfBus*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	IEEE1722TpAcfConnection.acfTransportedBus			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
baseFrequency	PositiveInteger	0..1	attr	CRF base frequency in Hz.
frameSync Enabled	Boolean	0..1	attr	Defines whether the "fs" (frame sync) shall be enabled.
timestamp Interval	PositiveInteger	0..1	attr	CRF timestamp interval as multiple of the baseFrequency.

Table E.83: IEEE1722TpAcfLin

Class	IEEE1722TpAcfLinPart			
Note	Definition of one LIN part transported over the IEEE1722Tp channel. **Tags:** atp.Status=candidate			
Base	*ARObject*, *IEEE1722TpAcfBusPart*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*IEEE1722TpAcfBus*.acfPart			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
linIdentifier	PositiveInteger	0..1	attr	Optional Lin Id defined in case the Lin Id can not be determined during runtime. **Tags:** atp.Status=candidate

Table E.84: IEEE1722TpAcfLinPart

Class	***IEEE1722TpConnection*** (abstract)			
Note	Definition of the IEEE1722Tp protocol. **Tags:** atp.Status=candidate			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	IEEE1722TpAcfConnection, *IEEE1722TpAvConnection*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
communication Direction	Communication DirectionType	0..1	attr	Communication Direction of the IEEE1722TpConnection. **Tags:** atp.Status=candidate This Attribute is only used by the AUTOSAR Adaptive Platform.
destinationMac Address	MacAddressString	0..1	attr	Optional definition of the destination MAC address for this stream. If no given then macAddressStreamId is used as destination MAC address. **Tags:** atp.Status=candidate

▽

△

Class	***IEEE1722TpConnection*** (abstract)			
globalTime Domain	GlobalTimeDomain	0..1	ref	Reference to the GlobalTimeDomain this IEEE1722Tp Connection shall be synchronized with. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=globalTimeDomain, globalTime Domain.globalTimeDomain, globalTimeDomain.variation Point.shortLabel vh.latestBindingTime=systemDesignTime
macAddress StreamId	MacAddressString	0..1	attr	MAC Address part of the Stream Id. **Tags:** atp.Status=candidate
uniqueStreamId	PositiveInteger	0..1	attr	Unique Id part of the Stream Id. **Tags:** atp.Status=candidate
version	PositiveInteger	0..1	attr	Version of the IEEE1722TP stream. **Tags:** atp.Status=candidate
vlanPriority	PositiveInteger	0..1	attr	Optional definition of the VLAN priority for this stream.

Table E.85: IEEE1722TpConnection

Class	**IPduPort**			
Note	Connectors reception or send port on the referenced channel referenced by a PduTriggering.			
Base	*ARObject*, *CommConnectorPort*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*CommunicationConnector*.ecuCommPortInstance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iPduSignal Processing	IPduSignalProcessing Enum	0..1	attr	Definition of the two signal processing modes Immediate and Deferred for both Tx and Rx IPdus.
rxSecurity Verification	Boolean	0..1	attr	This attribute defines the bypassing of signature authentication or MAC verification in the receiving ECU. If not defined or set to true the signature authentication or MAC verification shall be performed for the SecuredIPdu. If set to false the signature authentication or MAC verification shall not be performed for the SecuredIPdu.
timestampRx Acceptance Window	TimeValue	0..1	attr	This attribute is used to define the maximum allowed deviation in seconds from the expected timestamp for which a SecuredIPdu is still deemed authentic. Please note that this attribute is for documentation only to allow the configuration of required freshness value manager and no upstream mapping is defined for it.
useAuthData Freshness	Boolean	0..1	attr	This attribute describes whether a part of AuthenticPdu contained in a SecuredIPdu shall be passed on to the SWC that verifies and generates the Freshness. The part of the Authentic-PDU is defined by the authData FreshnessStartPosition and authDataFreshnessLength.

Table E.86: IPduPort

Class	IPduTiming			
Note	AUTOSAR COM provides the possibility to define two different TRANSMISSION MODES for each IPdu. The Transmission Mode of an IPdu that is valid at a specific point in time is selected using the values of the signals that are mapped to this IPdu. For each IPdu a Transmission Mode Selector is defined. The Transmission Mode Selector is calculated by evaluating the conditions for a subset of signals (class TransmissionModeCondition in the System Template). The Transmission Mode Selector is defined to be true, if at least one Condition evaluates to true and is defined to be false, if all Conditions evaluate to false.			
Base	*ARObject*, *Describable*			
Aggregated by	ISignalIPdu.iPduTimingSpecification			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
minimumDelay	TimeValue	0..1	attr	Minimum Delay in seconds between successive transmissions of this I-PDU, independent of the Transmission Mode.
transmission Mode Declaration	TransmissionMode Declaration	0..1	aggr	AUTOSAR COM allows configuring statically two different transmission modes for each I-PDU (True and False). The Transmission Mode Selector evaluates the conditions for a subset of signals and decides the transmission mode. It is possible to switch between the transmission modes during runtime. **Tags:** atp.Splitkey=transmissionModeDeclaration

Table E.87: IPduTiming

Class	ISignal			
Note	Signal of the Interaction Layer. The RTE supports a "signal fan-out" where the same System Signal is sent in different SignalIPdus to multiple receivers. To support the RTE "signal fan-out" each SignalIPdu contains ISignals. If the same System Signal is to be mapped into several SignalIPdus there is one ISignal needed for each ISignalToIPduMapping. ISignals describe the Interface between the Precompile configured RTE and the potentially Postbuild configured Com Stack (see ECUC Parameter Mapping). In case of the SystemSignalGroup an ISignal shall be created for each SystemSignal contained in the SystemSignalGroup. **Tags:** atp.recommendedPackage=ISignals			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
data Transformation	DataTransformation	0..1	ref	Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignal. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=dataTransformation, data Transformation.dataTransformation, data Transformation.variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime

▽

△

Class	ISignal			
dataTypePolicy	DataTypePolicyEnum	0..1	attr	With the aggregation of SwDataDefProps an ISignal specifies how it is represented on the network. This representation follows a particular policy. Note that this causes some redundancy which is intended and can be used to support flexible development methodology as well as subsequent integrity checks. If the policy "networkRepresentationFromComSpec" is chosen the network representation from the ComSpec that is aggregated by the PortPrototype shall be used. If the "override" policy is chosen the requirements specified in the PortInterface and in the ComSpec are not fulfilled by the networkRepresentationProps. In case the System Description doesn't use a complete Software Component Description (VFB View) the "legacy" policy can be chosen.
initValue	ValueSpecification	0..1	aggr	Optional definition of a ISignal's initValue in case the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy system signals. This value can be used to configure the Signal's "Init Value". If a full DataMapping exist for the SystemSignal this information may be available from a configured Sender ComSpec and ReceiverComSpec. In this case the initvalues in SenderComSpec and/or ReceiverComSpec override this optional value specification. Further restrictions apply from the RTE specification. **Tags:** atp.Splitkey=initValue
iSignalProps	ISignalProps	0..1	aggr	Additional optional ISignal properties that may be stored in different files. **Tags:** atp.Splitkey=iSignalProps
iSignalType	ISignalTypeEnum	0..1	attr	This attribute defines whether this iSignal is an array that results in a UINT8_N / UINT8_DYN ComSignalType in the COM configuration or a primitive type.
length	UnlimitedInteger	0..1	attr	Size of the signal in bits. The size needs to be derived from the mapped VariableDataPrototype according to the mapping of primitive DataTypes to BaseTypes as used in the RTE. Indicates maximum size for dynamic length signals. The ISignal length of zero bits is allowed.

▽

△

Class	**ISignal**			
network Representation Props	SwDataDefProps	0..1	aggr	Specification of the actual network representation. The usage of SwDataDefProps for this purpose is restricted to the attributes compuMethod and baseType. The optional baseType attributes "memAllignment" and "byteOrder" shall not be used. The attribute "dataTypePolicy" in the SystemTemplate element defines whether this network representation shall be ignored and the information shall be taken over from the network representation of the ComSpec. If "override" is chosen by the system integrator the network representation can violate against the requirements defined in the PortInterface and in the network representation of the ComSpec. In case that the System Description doesn't use a complete Software Component Description (VFB View) this element is used to configure "ComSignalDataInvalid Value" and the Data Semantics. **Tags:** atp.Splitkey=networkRepresentationProps
systemSignal	SystemSignal	0..1	ref	Reference to the System Signal that is supposed to be transmitted in the ISignal. **Tags:** atp.Splitkey=systemSignal
timeout Substitution Value	ValueSpecification	0..1	aggr	Defines and enables the ComTimeoutSubstituition for this ISignal. **Tags:** atp.Splitkey=timeoutSubstitutionValue
transformation ISignalProps	TransformationISignal Props	*	aggr	A transformer chain consists of an ordered list of transformers. The ISignal specific configuration properties for each transformer are defined in the TransformationISignalProps class. The transformer configuration properties that are common for all ISignals are described in the TransformationTechnology class. **Tags:** atp.Splitkey=transformationISignalProps

Table E.88: ISignal

Class	**ISignalGroup**			
Note	SignalGroup of the Interaction Layer. The RTE supports a "signal fan-out" where the same System Signal Group is sent in different SignalIPdus to multiple receivers. An ISignalGroup refers to a set of ISignals that shall always be kept together. A ISignalGroup represents a COM Signal Group. Therefore it is recommended to put the ISignalGroup in the same Package as ISignals (see atp.recommendedPackage) **Tags:** atp.recommendedPackage=ISignalGroups			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ISignalGroup**			
comBased SignalGroup Transformation	DataTransformation	0..1	ref	Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignalGroup based on the COMBasedTransformer approach. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=comBasedSignalGroupTransformation, comBasedSignalGroupTransformation.dataTransformation, comBasedSignalGroupTransformation.variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime
iSignal	ISignal	*	ref	Reference to a set of ISignals that shall always be kept together. **Tags:** atp.Splitkey=iSignal
systemSignal Group	SystemSignalGroup	0..1	ref	Reference to the SystemSignalGroup that is defined on VFB level and that is supposed to be transmitted in the ISignalGroup. **Tags:** atp.Splitkey=systemSignalGroup
transformation ISignalProps	TransformationISignal Props	*	aggr	A transformer chain consists of an ordered list of transformers. The ISignalGroup specific configuration properties for each transformer are defined in the TransformationISignalProps class. The transformer configuration properties that are common for all ISignal Groups are described in the TransformationTechnology class. **Tags:** atp.Splitkey=transformationISignalProps

Table E.89: ISignalGroup

Class	**ISignalIPdu**			
Note	Represents the IPdus handled by Com. The ISignalIPdu assembled and disassembled in AUTOSAR COM consists of one or more signals. In case no multiplexing is performed this IPdu is routed to/from the Interface Layer. A maximum of one dynamic length signal per IPdu is allowed. **Tags:** atp.recommendedPackage=Pdus			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *IPdu*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Pdu*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iPduTiming Specification	IPduTiming	0..1	aggr	Timing specification for Com IPdus (Transmission Modes). This information is mandatory for the sender in a System Extract. This information may be omitted on receivers in a System Extract. atpVariation: The timing of a Pdu can vary. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=iPduTimingSpecification, iPduTimingSpecification.variationPoint.shortLabel vh.latestBindingTime=postBuild

▽

△

Class	**ISignalIPdu**			
iSignalToPdu Mapping	ISignalToIPduMapping	*	aggr	Definition of SignalToIPduMappings included in the Signal IPdu. atpVariation: The content of a PDU can be variable. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=iSignalToPduMapping, iSignalToPdu Mapping.shortName, iSignalToPduMapping.variation Point.shortLabel vh.latestBindingTime=postBuild
unusedBit Pattern	Integer	0..1	attr	AUTOSAR COM and AUTOSAR IPDUM are filling not used areas of an IPDU with this bit-pattern. This attribute is mandatory to avoid undefined behavior. This byte-pattern will be repeated throughout the IPdu.

Table E.90: ISignalIPdu

Class	**ISignalPort**			
Note	Connectors reception or send port on the referenced channel referenced by an ISignalTriggering. If different timeouts or DataFilters for ISignals need to be specified several ISignalPorts may be created.			
Base	*ARObject*, *CommConnectorPort*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*CommunicationConnector*.ecuCommPortInstance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataFilter	DataFilter	0..1	aggr	Optional specification of a signal COM filter at the receiver side in case that the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy system signals. If a full DataMapping exist for the SystemSignal this information may be available from a configured ReceiverComSpec. In this case the ReceiverComSpec overrides this optional specification. **Tags:** atp.Splitkey=dataFilter
ddsQosProfile	DdsCpQosProfile	0..1	ref	Reference to the DDS Qos profile used for this ISignal. **Tags:** atp.Splitkey=ddsQosProfile atp.Status=candidate
firstTimeout	TimeValue	0..1	attr	• ISignalPort with communicationDirection = in: Optional first timeout value in seconds for the reception of the ISignal. • ISignalPort with communicationDirection = out: Optional first timeout value in seconds for transmission deadline monitoring.
handleInvalid	HandleInvalidEnum	0..1	attr	This attribute defines how invalidation is applied to the ISignals received in the context of this ISignalPort.

▽

△

Class	**ISignalPort**			
timeout	TimeValue	0..1	attr	• ISignalPort with communicationDirection = in: Optional timeout value in seconds for the reception of the ISignal. The attribute value is used to configure the Com Timeout in the COM module. The RTE ignores this attribute. The timeout can also be specified with the NonqueuedReceiverComSpec.aliveTimeout attribute. If a full DataMapping exists for the SystemSignal and the value is available in the configured ReceiverComSpec, then the timeout value in the ReceiverComSpec overrides this optional timeout specification during the creation of the Base Ecu Configuration of the COM module. • ISignalPort with communicationDirection = out: Optional timeout value in seconds for the transmission of the ISignal. The attribute value is used to configure the ComTimeout in the COM module. The RTE ignores this attribute. The timeout can also be specified with the SenderComSpec.transmissionAcknowledge.timeout attribute. If a full DataMapping exists for the System Signal and the value is available in the configured Sender ComSpec, then the timeout value in the SenderComSpec overrides this optional timeout specification during the creation of the Base Ecu Configuration of the COM module. This attribute can be used in the following cases: • legacy signal where the System Description doesn't use a complete Software Component Description (VFB View) and where the DataMapping is missing. • bus monitoring use cases in which the DataMapping is ignored.

Table E.91: ISignalPort

Class	**ISignalToIPduMapping**			
Note	An ISignalToIPduMapping describes the mapping of ISignals to ISignalIPdus and defines the position of the ISignal within an ISignalIPdu.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ISignalIPdu.iSignalToPduMapping, NmPdu.iSignalToIPduMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iSignal	ISignal	0..1	ref	Reference to a ISignal that is mapped into the ISignal IPdu. Each ISignal contained in the ISignalGroup shall be mapped into an IPdu by an own ISignalToIPduMapping. The references to the ISignal and to the ISignalGroup in an ISignalToIPduMapping are mutually exclusive. **Tags:** atp.Splitkey=iSignal

▽

△

Class	**ISignalToIPduMapping**			
iSignalGroup	ISignalGroup	0..1	ref	Reference to an ISignalGroup that is mapped into the SignalIPdu. If an ISignalToIPduMapping for an ISignal Group is defined, only the UpdateIndicationBitPosition and the transferProperty is relevant. The startPosition and the packingByteOrder shall be ignored. Each ISignal contained in the ISignalGroup shall be mapped into an IPdu by an own ISignalToIPduMapping. The references to the ISignal and to the ISignalGroup in an ISignalToIPduMapping are mutually exclusive. **Tags:** atp.Splitkey=iSignalGroup
packingByte Order	ByteOrderEnum	0..1	attr	This parameter defines the order of the bytes of the signal and the packing into the SignalIPdu. The byte ordering "Little Endian" (MostSignificantByteLast), "Big Endian" (MostSignificantByteFirst) and "Opaque" can be selected. For opaque data endianness conversion shall be configured to Opaque. The value of this attribute impacts the absolute position of the signal into the SignalIPdu (see the startPosition attribute description). For an ISignalGroup the packingByteOrder is irrelevant and shall be ignored.
startPosition	UnlimitedInteger	0..1	attr	This parameter is necessary to describe the bitposition of a signal within an SignalIPdu. It denotes the least significant bit for "Little Endian" and the most significant bit for "Big Endian" packed signals within the IPdu (see the description of the packingByteOrder attribute). In AUTOSAR the bit counting is always set to "sawtooth" and the bit order is set to "Decreasing". The bit counting in byte 0 starts with bit 0 (least significant bit). The most significant bit in byte 0 is bit 7. Please note that the way the bytes will be actually sent on the bus does not impact this representation: they will always be seen by the software as a byte array. If a mapping for the ISignalGroup is defined, this attribute is irrelevant and shall be ignored.
transferProperty	TransferPropertyEnum	0..1	attr	Defines how the referenced ISignal contributes to the send triggering of the ISignalIPdu.
update IndicationBit Position	UnlimitedInteger	0..1	attr	The UpdateIndicationBit indicates to the receivers that the signal (or the signal group) was updated by the sender. Length is always one bit. The UpdateIndicationBitPosition attribute describes the position of the update bit within the SignalIPdu. For Signals of a ISignalGroup this attribute is irrelevant and shall be ignored. Note that the exact bit position of the updateIndicationBit Position is linked to the value of the attribute packingByte Order because the method of finding the bit position is different for the values mostSignificantByteFirst and most SignificantByteLast. This means that if the value of packingByteOrder is changed while the value of update IndicationBitPosition remains unchanged the exact bit position of updateIndicationBitPosition within the enclosing ISignalIPdu still undergoes a change. This attribute denotes the least significant bit for "Little Endian" and the most significant bit for "Big Endian" packed signals within the IPdu (see the description of the packingByteOrder attribute). In AUTOSAR the bit counting is always set to "sawtooth" and the bit order is set to "Decreasing". The bit counting in byte 0 starts with bit 0 (least significant bit). The most significant bit in byte 0 is bit 7.

Table E.92: ISignalToIPduMapping

Class	**ISignalTriggering**			
Note	A ISignalTriggering allows an assignment of ISignals to physical channels.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*PhysicalChannel*.iSignalTriggering			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
iSignal	ISignal	0..1	ref	This reference shall be used if an ISignal is transported on the PhysicalChannel. This reference forms an XOR relationship with the ISignalTriggering-ISignalGroup reference. **Tags:** atp.Splitkey=iSignal
iSignalGroup	ISignalGroup	0..1	ref	This reference shall be used if an ISignalGroup is transported on the PhysicalChannel. This reference forms an XOR relationship with the ISignal Triggering-ISignal reference. **Tags:** atp.Splitkey=iSignalGroup
iSignalPort	ISignalPort	*	ref	References to the ISignalPort on every ECU of the system which sends and/or receives the ISignal. References for both the sender and the receiver side shall be included when the system is completely defined. **Tags:** atp.Splitkey=iSignalPort

Table E.93: ISignalTriggering

Class	***Identifiable*** (abstract)
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*
Subclasses	ARPackage, *AbstractDoIpLogicAddressProps*, *AbstractEvent*, *AbstractFunctionalClusterDesign*, *AbstractImplementationDataTypeElement*, *AbstractSecurityEventFilter*, *AbstractSecurityIdsmInstance Filter*, *AbstractServiceInstance*, *AbstractSignalBasedToISignalTriggeringMapping*, AdaptiveSwcInternal Behavior, ApApplicationEndpoint, *ApmcAbstractDefinition*, *ApmcConfigurationElementDef*, *Apmc ContainerElementValue*, ApmcContainerValue, ApmcEnumerationLiteralDef, ApplicationEndpoint, ApplicationError, AppliedStandard, ArtifactChecksum, ArtifactLocator, *AtpBlueprint*, *AtpClassifier*, *Atp Feature*, AutosarOperationArgumentInstance, AutosarVariableInstance, *BuildActionEntity*, BuildAction Environment, CanSecSecureChannel, Chapter, CheckpointTransition, ClientIdDefinition, ClientServer Operation, Code, *CollectableElement*, ComManagementMapping, *CommConnectorPort*, *CommunicationConnector*, *CommunicationController*, Compiler, ConsistencyNeeds, ConsumedEvent Group, CouplingPort, *CouplingPortAbstractShaper*, *CouplingPortStructuralElement*, CryptoCertificate, CryptoCertificateGroup, CryptoKeySlot, CryptoKeySlotDesign, CryptoKeySlotUsageDesign, Crypto Provider, *CryptoServiceMapping*, DataPrototypeGroup, DataPrototypeTransformationPropsIdent, Data Transformation, *DdsAbstractServiceInstanceElementCp*, DdsCpDomain, DdsCpPartition, DdsCpQos Profile, DdsCpTopic, DdsDomainRange, DependencyOnArtifact, *DiagEventDebounceAlgorithm*, *DiagnosticAbstractSovdContent*, DiagnosticAuthTransmitCertificateEvaluation, DiagnosticConnected Indicator, DiagnosticDataElement, DiagnosticDebounceAlgorithmProps, DiagnosticExtendedDataRecord Element, DiagnosticFunctionInhibitSource, DiagnosticParameterElement, *DiagnosticRoutineSubfunction*, DiagnosticSovdMethodPrimitive, DltApplication, DltArgument, DltArgumentProps, DltMessage, DoIp Interface, DoIpLogicAddress, DoIpLogicalAddress, DoIpNetworkConfigurationDesign, DoIpRouting Activation, E2EProfileConfiguration, End2EndEventProtectionProps, End2EndMethodProtectionProps, EthernetWakeupSleepOnDatalineConfig, EventHandler, EventMapping, ExclusiveArea, *Executable Entity*, *ExecutionTime*, FMAttributeDef, FMFeatureMapAssertion, FMFeatureMapCondition, FMFeature MapElement, FMFeatureRelation, FMFeatureRestriction, FMFeatureSelection, FieldMapping, FireAnd ForgetMethodMapping, FlexrayTpPduPool, *FrameTriggering*, GlobalSupervision, GlobalTimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, *HealthChannel*, *HeapUsage*, HwAttributeDef, HwAttributeLiteral Def, HwPin, HwPinGroup, *IEEE1722TpAcfBus*, *IEEE1722TpAcfBusPart*, IPSecRule, IPv6ExtHeader FilterList, ISignalToIPduMapping, ISignalTriggering, *IdentCaption*, ImpositionTime, InternalTriggering Point, Keyword, LifeCycleState, Linker, MacAddressVlanMembership, MacMulticastGroup, MacSecKay Participant, McDataInstance, MemorySection, MemoryUsage, MethodMapping, ModeDeclaration, Mode ▽ ▽

△

Class	***Identifiable*** (abstract)
	△ DeclarationMapping, ModeSwitchPoint, ModeSwitchSenderComSpecProps, NetworkEndpoint, *Nm Cluster*, *NmNode*, *PackageableElement*, ParameterAccess, PduActivationRoutingGroup, PduToFrame Mapping, PduTriggering, PerInstanceMemory, *PersistencyDeploymentElement*, *PersistencyInterface Element*, *PhmSupervision*, *PhysicalChannel*, PortGroup, *PortInterfaceMapping*, ProcessToMachine Mapping, Processor, ProcessorCore, PskIdentityToKeySlotMapping, QueuedReceiverComSpecProps, ResourceConsumption, ResourceGroup, RootSwClusterDesignComponentPrototype, RootSw ComponentPrototype, RootSwCompositionPrototype, RptComponent, RptContainer, RptExecutable Entity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptServicePoint, RunnableEntity Group, *SdgAttribute*, SdgClass, SecOcJobMapping, SecOcJobRequirement, SecureCommunication AuthenticationProps, *SecureCommunicationDeployment*, SecureCommunicationFreshnessProps, SecurityEventContextDataElement, SecurityEventContextProps, ServerComSpecProps, *Service InterfaceDeploymentElement*, ServiceInterfaceElementSecureComConfig, *ServiceNeeds*, SignalService TranslationEventProps, SignalServiceTranslationProps, SocketAddress, SoftwarePackageStep, Someip EventGroup, SomeipProvidedEventGroup, SomeipTpChannel, *StackUsage*, *StateManagementAction Item*, StateManagementActionList, StateManagementStateNotification, *StateManagementStateRequest*, StaticSocketConnection, StructuredReq, SupervisionCheckpoint, SupervisionMode, SupervisionMode Condition, SwGenericAxisParamType, SwServiceArg, SwcServiceDependency, SystemMapping, *Time BaseResource*, *TimingClock*, TimingClockSyncAccuracy, TimingCondition, *TimingConstraint*, *Timing Description*, TimingExtensionResource, TimingModeInstance, TlsCryptoCipherSuite, TlsCryptoCipher SuiteProps, TlsJobMapping, Topic1, TpAddress, TraceableTable, TraceableText, *TracedFailure*, TransformationISignalPropsIdent, *TransformationProps*, TransformationTechnology, Trigger, Ucm Description, UcmRetryStrategy, UcmStep, UriDescription, VariableAccess, VariationPointProxy, Vehicle RolloutStep, ViewMap, VlanConfig, WaitPoint

Attribute	***Type***	***Mult.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Tags:** atp.Splitkey=adminData xml.sequenceOffset=-40
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. **Tags:** atp.Splitkey=annotation xml.sequenceOffset=-25
category	CategoryString	0..1	attr	The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints. **Tags:** xml.sequenceOffset=-50
desc	MultiLanguageOverview Paragraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question. More elaborate documentation, (in particular how the object is built or used) should go to "introduction". **Tags:** atp.Splitkey=desc xml.sequenceOffset=-60
introduction	DocumentationBlock	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock. **Tags:** atp.Splitkey=introduction xml.sequenceOffset=-30

▽

△

Class	***Identifiable*** (abstract)			
uuid	String	0..1	attr	The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". The uuid attribute has no semantic meaning for an AUTOSAR model and there is no requirement for AUTOSAR tools to manage the timestamp. **Tags:** xml.attribute=true

Table E.94: Identifiable

Class	**ImplementationDataType**			
Note	Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code. **Tags:** atp.recommendedPackage=ImplementationDataTypes			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *Autosar DataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicArray SizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.
isStructWith Optional Element	Boolean	0..1	attr	This attribute is only valid if the attribute category is set to STRUCTURE. If set to true, this attribute indicates that the ImplementationDataType has been created with the intention to define at least one element of the structure as optional.
subElement (ordered)	ImplementationData TypeElement	*	aggr	Specifies an element of an array, struct, or union data type. The aggregation of `ImplementationDataTypeElement` is subject to variability with the purpose to support the conditional existence of elements inside a `ImplementationDataType` representing a structure. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the Implementation DataType. **Tags:** atp.Splitkey=symbolProps, symbolProps.short Name

▽

△

Class	ImplementationDataType			
typeEmitter	NameToken	0..1	attr	This attribute is used to control which part of the AUTOSAR toolchain is supposed to trigger data type definitions.

Table E.95: ImplementationDataType

Class	ImplementationDataTypeElement			
Note	Declares a data object which is locally aggregated. Such an element can only be used within the scope where it is aggregated. This element either consists of further `subElement`s or it is further defined via its `swDataDefProps`. There are several use cases within the system of `ImplementationDataType`s for such a local declaration: • It can represent the elements of an array, defining the element type and array size • It can represent an element of a struct, defining its type • It can be the local declaration of a debug element.			
Base	*ARObject*, *AbstractImplementationDataTypeElement*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, ImplementationDataType.subElement, ImplementationDataTypeElement.subElement			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
arrayImplPolicy	ArrayImplPolicyEnum	0..1	attr	This attribute controls the implementation of the payload of an array. It shall only be used if the enclosing `ImplementationDataType` constitutes an array.
arraySize	PositiveInteger	0..1	attr	The existence of this attributes (if bigger than 0) defines the size of an array and declares that this `ImplementationDataTypeElement` represents the type of each single array element. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
arraySize Handling	ArraySizeHandling Enum	0..1	attr	The way how the size of the array is handled in case of a variable size array.
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls the meaning of the value of the array size.
isOptional	Boolean	0..1	attr	This attribute represents the ability to declare the enclosing `ImplementationDataTypeElement` as optional. This means that, at runtime, the `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.
subElement (ordered)	ImplementationData TypeElement	*	aggr	Element of an array, struct, or union in case of a nested declaration (i.e. without using "typedefs"). The aggregation of `ImplementationDataTypeElement` is subject to variability with the purpose to support the conditional existence of elements inside a `ImplementationDataType` representing a structure. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
swDataDef Props	SwDataDefProps	0..1	aggr	The properties of this ImplementationDataTypeElement. **Tags:** atp.Splitkey=swDataDefProps

Table E.96: ImplementationDataTypeElement

Class	InternalConstrs			
Note	This meta-class represents the ability to express internal constraints.			
Base	*ARObject*			
Aggregated by	DataConstrRule.internalConstrs			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the constraint. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
maxDiff	Numerical	0..1	attr	Maximum difference that is permitted between two consecutive values if the constraint is applied to an axis. **Tags:** xml.sequenceOffset=60
maxGradient	Numerical	0..1	attr	This element specifies the maximum slope that may be used in maps and curves. **Tags:** xml.sequenceOffset=50
monotony	MonotonyEnum	0..1	attr	This element specifies the monotony characteristics of the current internal or physical limits. The following table shows the monotony characteristics which are to be filled through the corresponding values. If the element has no contents or if it is omitted, "no Monotony" is the default content. **Tags:** xml.sequenceOffset=70
upperLimit	Limit	0..1	attr	This specifies the upper limit defined by the constraint. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=30

Table E.97: InternalConstrs

Class	MacMulticastConfiguration			
Note	References a per cluster globally defined MAC-Multicast-Group.			
Base	*ARObject*, *NetworkEndpointAddress*			
Aggregated by	NetworkEndpoint.networkEndpointAddress			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
macMulticast Group	MacMulticastGroup	0..1	ref	Reference to a macMulticastGroup. **Stereotypes:** atpIdentityContributor

Table E.98: MacMulticastConfiguration

Class	ModeSwitchInterface			
Note	A mode switch interface declares a `ModeDeclarationGroupPrototype` to be sent and received. **Tags:** atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**ModeSwitchInterface**			
modeGroup	ModeDeclarationGroup Prototype	0..1	aggr	The `ModeDeclarationGroupPrototype` of this mode interface. **Tags:** atp.Splitkey=modeGroup, modeGroup.shortName

Table E.99: ModeSwitchInterface

Class	**MultiplexedIPdu**			
Note	A MultiplexedPdu (i.e. NOT a COM I-PDU) contains a DynamicPart, an optional StaticPart and a selector Field. In case of multiplexing this IPdu is routed between the Pdu Multiplexer and the Interface Layer. A multiplexer is used to define variable parts within an IPdu that may carry different signals. The receivers of such a IPdu can determine which signalPdus are transmitted by evaluating the selector field, which carries a unique selector code for each sub-part. **Tags:** atp.recommendedPackage=Pdus			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *IPdu*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Pdu*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicPart	DynamicPart	0..1	aggr	According to the value of the selector field some parts of the IPdu have a different layout. In a complete System Description a MultiplexedIPdu shall contain a Dynamic Part. The following use cases support the multiplicity to be 0..1: • If a MultiplexedIPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedIPdu doesn't need to be described in the System Extract/ Ecu Extract. • If a MultiplexedIPdu is received by an ECU which is only interested in the static part of the MultiplexedIPdu then the dynamicPart does not need to be described in the System Extract/Ecu Extract. atpVariation: Content of a multiplexed PDU can vary. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=dynamicPart, dynamicPart.variation Point.shortLabel vh.latestBindingTime=postBuild
selectorField ByteOrder	ByteOrderEnum	0..1	attr	This attribute defines the order of the bytes of the selector Field and the packing into the MultiplexedIPdu. Please consider that [constr_3247] and [constr_3223] are restricting the usage of this attribute. In a complete System Description this attribute is mandatory. If a MultiplexedPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedPdu doesn't need to be described in the System Extract/Ecu Extract. To support this use case the multiplicity is set to 0..1.

▽

△

Class	**MultiplexedIPdu**			
selectorField Length	Integer	0..1	attr	The size in bits of the selector field shall be configurable in a range of 1-16 bits. In a complete System Description this attribute is mandatory. If a MultiplexedPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedPdu doesn't need to be described in the System Extract/Ecu Extract. To support this use case the multiplicity is set to 0..1.
selectorField StartPosition	Integer	0..1	attr	This parameter is necessary to describe the position of the selector field within the IPdu. Note that the absolute position of the selectorField in the MultiplexedIPdu is determined by the definition of the selectorFieldByteOrder attribute of the Multiplexed Pdu. If Big Endian is specified, the start position indicates the bit position of the most significant bit in the IPdu. If Little Endian is specified, the start position indicates the bit position of the least significant bit in the IPdu. In AUTOSAR the bit counting is always set to "sawtooth" and the bit order is set to "Decreasing". The bit counting in byte 0 starts with bit 0 (least significant bit). The most significant bit in byte 0 is bit 7. In a complete System Description this attribute is mandatory. If a MultiplexedPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedPdu doesn't need to be described in the System Extract/Ecu Extract. To support this use case the multiplicity is set to 0..1.
staticPart	StaticPart	0..1	aggr	The static part of the multiplexed IPdu is the same regardless of the selector field. The static part is optional. atpVariation: Content of a multiplexed PDU can vary. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=staticPart, staticPart.variationPoint.shortLabel vh.latestBindingTime=postBuild
triggerMode	TriggerMode	0..1	attr	IPduM can be configured to send a transmission request for the new multiplexed IPdu to the PDU-Router because of the trigger conditions/ modes that are described in the TriggerMode enumeration. In a complete System Description this attribute is mandatory. If a MultiplexedPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedPdu doesn't need to be described in the System Extract/Ecu Extract. To support this use case the multiplicity is set to 0..1.
unusedBit Pattern	Integer	0..1	attr	AUTOSAR COM and AUTOSAR IPDUM are filling not used areas of an IPdu with this bit-pattern. This attribute is mandatory to avoid undefined behavior. This byte-pattern will be repeated throughout the IPdu. In a complete System Description this attribute is mandatory. If a MultiplexedPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedPdu doesn't need to be described in the System Extract/Ecu Extract. To support this use case the multiplicity is set to 0..1.

Table E.100: MultiplexedIPdu

Primitive	**NameToken**
Note	This is an identifier as used in xml, e.g. xml-names. Typical usages are, for example, the names of type emitters, protocols, or profiles. For details see NMTOKEN definition on the W3C website (https://www.w3.org/TR/xml/#NT-Nmtoken). Note: Although NameToken supports a wide range of characters, the actually allowed patterns for a certain attribute typed by NameToken may be further restricted by the specification of that attribute. **Tags:** xml.xsd.customType=NMTOKEN-STRING xml.xsd.type=NMTOKEN

Table E.101: NameToken

Class	**NotAvailableValueSpecification**			
Note	This meta-class provides the ability to specify a ValueSpecification to state that the respective element is not available. This ability is needed to support the existence of ApplicationRecordElements where attribute isOptional ist set to the value true.			
Base	*ARObject*, *ValueSpecification*			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValue Specification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, ConstantSpecification.valueSpec, CryptoService Key.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElement Condition.compareValue, DiagnosticEnvSovdDataCondition.compareValue, FieldSenderComSpec.init Value, ISignal.initValue, ISignal.timeoutSubstitutionValue, NonqueuedReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, Nv ProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.init Value, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireCom Spec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, Port DefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalidValue, VariableData Prototype.initValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defaultPattern	PositiveInteger	0..1	attr	The content of this attribute shall be used to initialize gaps in the memory occupied by a structured data type in the case that an NotAvailableValueSpecification is used. Note that this pattern is only applied during initialization!

Table E.102: NotAvailableValueSpecification

Class	**NumericalValueSpecification**			
Note	A numerical ValueSpecification which is intended to be assigned to a Primitive data element. Note that the numerical value is a variant, it can be computed by a formula.			
Base	*ARObject*, *ValueSpecification*			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValue Specification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, ConstantSpecification.valueSpec, CryptoService Key.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElement Condition.compareValue, DiagnosticEnvSovdDataCondition.compareValue, FieldSenderComSpec.init Value, ISignal.initValue, ISignal.timeoutSubstitutionValue, NonqueuedReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, Nv ProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.init Value, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireCom Spec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, Port DefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalidValue, VariableData Prototype.initValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	NumericalValueSpecification			
value	Numerical	0..1	attr	This is the value itself. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table E.103: NumericalValueSpecification

Class	*PPortComSpec* (abstract)			
Note	Communication attributes of a provided `PortPrototype`. This class will contain attributes that are valid for all kinds of provide ports, independent of client-server or sender-receiver communication patterns.			
Base	*ARObject*			
Subclasses	ModeSwitchSenderComSpec, NvProvideComSpec, ParameterProvideComSpec, *SenderComSpec*, ServerComSpec			
Aggregated by	*AbstractProvidedPortPrototype*.providedComSpec, PortPrototypeBlueprint.providedComSpec			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table E.104: PPortComSpec

Class	PPortPrototype			
Note	Component port providing a certain port interface.			
Base	*ARObject*, *AbstractProvidedPortPrototype*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
provided Interface	PortInterface	0..1	tref	The interface that this port provides. **Stereotypes:** isOfType **Tags:** atp.Splitkey=providedInterface

Table E.105: PPortPrototype

Class	PRPortPrototype			
Note	This kind of PortPrototype can take the role of both a required and a provided PortPrototype.			
Base	*ARObject*, *AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
provided Required Interface	PortInterface	0..1	tref	This represents the `PortInterface` used to type the `PRPortPrototype`. **Stereotypes:** isOfType **Tags:** atp.Splitkey=providedRequiredInterface

Table E.106: PRPortPrototype

Class	*PackageableElement* (abstract)
Note	This meta-class specifies the ability to be a member of an AUTOSAR package.
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*
Subclasses	*ARElement*, EnumerationMappingTable, *FibexElement*

▽

△

Class	**PackageableElement** (abstract)			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table E.107: PackageableElement

Class	**Pdu** (abstract)			
Note	Collection of all Pdus that can be routed through a bus interface.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Subclasses	GeneralPurposePdu, *IPdu*, NmPdu, UserDefinedPdu			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
hasDynamic Length	Boolean	0..1	attr	This attribute defines whether the Pdu has dynamic length (true) or not (false). Please note that the usage of this attribute is restricted by [constr_3448].
length	UnlimitedInteger	0..1	attr	Pdu length in bytes. In case of dynamic length IPdus (containing a dynamical length signal), this value indicates the maximum data length. It should be noted that in former AUTOSAR releases (Rel 2.1, Rel 3.0, Rel 3.1, Rel 4.0 Rev. 1) this parameter was defined in bits. The Pdu length of zero bytes is allowed.

Table E.108: Pdu

Class	**PduTriggering**			
Note	The PduTriggering describes on which channel the IPdu is transmitted. The Pdu routing by the PduR is only allowed for subclasses of IPdu. Depending on its relation to entities such channels and clusters it can be unambiguously deduced whether a fan-out is handled by the Pdu router or the Bus Interface. If the fan-out is specified between different clusters it shall be handled by the Pdu Router. If the fan-out is specified between different channels of the same cluster it shall be handled by the Bus Interface.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*PhysicalChannel*.pduTriggering			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
iPdu	Pdu	0..1	ref	Reference to the Pdu for which the PduTriggering is defined. One I-Pdu can be triggered on different channels (PduR fan-out). The Pdu routing by the PduR is only allowed for subclasses of IPdu. Nevertheless is the reference to the Pdu element necessary since the PduTriggering element is also used to specify the sending and receiving connections to Ecu Ports. **Tags:** atp.Splitkey=iPdu
iPduPort	IPduPort	*	ref	References to the IPduPort on every ECU of the system which sends and/or receives the I-PDU. References for both the sender and the receiver side shall be included when the system is completely defined. **Tags:** atp.Splitkey=iPduPort

▽

△

Class	**PduTriggering**			
iSignal Triggering	ISignalTriggering	*	ref	This reference provides the relationship to the ISignal Triggerings that are implemented by the PduTriggering. The reference is optional since no ISignalTriggering can be defined for DCM and Multiplexed Pdus. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=iSignalTriggering, iSignalTriggering.iSignal Triggering, iSignalTriggering.variationPoint.shortLabel vh.latestBindingTime=postBuild
secOcCrypto Mapping	SecOcCryptoService Mapping	0..1	ref	This reference identifies the crypto profile applicable to the usage (send, receive) of the also referenced Secured IPdu. Obviously, this reference is only applicable if the Pdutriggering also references a SecuredIPdu in the role i Pdu. **Tags:** atp.Splitkey=secOcCryptoMapping atp.Status=obsolete
secOcCrypto ServiceMapping	SecOcCryptoService Mapping	0..1	ref	This reference identifies the crypto profile applicable to the usage (send, receive) of the also referenced Secured IPdu. Obviously, this reference is only applicable if the Pdutriggering also references a SecuredIPdu in the role i Pdu. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=secOcCryptoServiceMapping, secOcCrypto ServiceMapping.secOcCryptoServiceMapping, secOc CryptoServiceMapping.variationPoint.shortLabel vh.latestBindingTime=postBuild
triggerIPduSend Condition	TriggerIPduSend Condition	*	aggr	Defines the trigger for the Com_TriggerIPDUSend API call. Only if all defined TriggerIPduSendConditions evaluate to true (AND associated) the Com_Trigger IPDUSend API shall be called. **Tags:** atp.Splitkey=triggerIPduSendCondition

Table E.109: PduTriggering

Class	***PhmStateReference*** (abstract)			
Note	Definition of state dependency. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Subclasses	FunctionGroupPhmStateReference			
Aggregated by	SupervisionModeCondition.stateReference			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.110: PhmStateReference

Class	***PhmSupervision*** (abstract)			
Note	Defines explicitly that NO supervision shall be applied for a set of SupervisionCheckpoints. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	AliveSupervision, DeadlineSupervision, LogicalSupervision, NoCheckpointSupervision, NoSupervision			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.111: PhmSupervision

Class	**PortGroup**			
Note	Group of ports which share a common functionality. Example: need specific network resources. This information shall be available on the VFB level in order to delegate it properly via compositions. When propagated into the ECU extract, this information is used as input for the configuration of Services like the Communication Manager. A PortGroup is defined locally in a component (which can be a composition) and refers to the "outer" ports belonging to the group as well as to the "inner" groups which propagate this group into the components which are part of a composition. A PortGroup within an atomic SWC cannot be linked to inner groups.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.portGroup			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
innerGroup	PortGroup	*	iref	Links a PortGroup in a composition to another PortGroup, that is defined in a component which is part of this CompositionSwComponentType. **Tags:** atp.Splitkey=innerGroup, innerGroup.context, innerGroup.target **InstanceRef implemented by:** InnerPortGroupInCompositionInstanceRef
outerPort	PortPrototype	*	ref	Outer PortPrototype of this AtomicSwComponentType which belongs to the group. A port can belong to several groups or to no group at all. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=outerPort, outerPort.portPrototype, outerPort.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table E.112: PortGroup

Class	***PortPrototype*** (abstract)			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientServerAnnotation	ClientServerAnnotation	*	aggr	Annotation of this `PortPrototype` with respect to client/server communication. **Tags:** atp.Splitkey=clientServerAnnotation

▽

△

Class	***PortPrototype*** (abstract)			
delegatedPort Annotation	DelegatedPort Annotation	0..1	aggr	Annotations on this delegated port. **Tags:** atp.Splitkey=delegatedPortAnnotation
ioHwAbstraction Server Annotation	IoHwAbstractionServer Annotation	*	aggr	Annotations on this IO Hardware Abstraction port. **Tags:** atp.Splitkey=ioHwAbstractionServerAnnotation
modePort Annotation	ModePortAnnotation	*	aggr	Annotations on this mode port. **Tags:** atp.Splitkey=modePortAnnotation
nvDataPort Annotation	NvDataPortAnnotation	*	aggr	Annotations on this non voilatile data port. **Tags:** atp.Splitkey=nvDataPortAnnotation
parameterPort Annotation	ParameterPort Annotation	*	aggr	Annotations on this parameter port. **Tags:** atp.Splitkey=parameterPortAnnotation
portPrototype Props	PortPrototypeProps	0..1	aggr	This attribute allows for the definition of further qualification of the semantics of a PortPrototype. **Tags:** atp.Splitkey=portPrototypeProps This Attribute is only used by the AUTOSAR Adaptive Platform.
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication. **Tags:** atp.Splitkey=senderReceiverAnnotation
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port. **Tags:** atp.Splitkey=triggerPortAnnotation

Table E.113: PortPrototype

Class	**ProvidedServiceInstance**			
Note	Service instances that are provided by the ECU that is connected via the ApplicationEndpoint to a CommunicationConnector.			
Base	*ARObject*, *AbstractServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ServiceInstanceCollectionSet.serviceInstance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allowedService Consumer	NetworkEndpoint	*	ref	NetworkEndpoints on which the ConsumedService Instances that are communicating with this Provided ServiceInstance are allowed to be located so that the ACL check in the ServiceDiscovery is successful and the connection is allowed to be established. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=allowedServiceConsumer, allowedService Consumer.networkEndpoint, allowedService Consumer.variationPoint.shortLabel atp.Status=draft vh.latestBindingTime=postBuild
autoAvailable	Boolean	0..1	attr	Defines that this ProvidedServiceInstance shall be offered by the service discovery at ECU start.
eventHandler	EventHandler	*	aggr	Collection of event groups provided by the Provided ServiceInstance **Stereotypes:** atpVariation **Tags:** atp.Splitkey=eventHandler, eventHandler.shortName, eventHandler.variationPoint.shortLabel vh.latestBindingTime=postBuild

▽

△

Class	**ProvidedServiceInstance**			
instance Identifier	PositiveInteger	0..1	attr	Instance identifier. Can be used for e.g. service discovery to identify the instance of the service.
loadBalancing Priority	PositiveInteger	0..1	attr	Defines the value to be used for load balancing priority in the service offer. Lower value means higher priority.
loadBalancing Weight	PositiveInteger	0..1	attr	Defines the value to be used for load balancing weight in the service offer. Higher value means higher probability to be chosen.
localUnicast Address	ApplicationEndpoint	0..2	ref	The local address over which the PSI is provided (udp, tcp or both). **Stereotypes:** atpVariation **Tags:** atp.Splitkey=localUnicastAddress, localUnicast Address.applicationEndpoint, localUnicast Address.variationPoint.shortLabel vh.latestBindingTime=postBuild
minorVersion	PositiveInteger	0..1	attr	Minor Version of the Service that is provided by this ProvidedServiceInstance.
priority	PositiveInteger	0..1	attr	Defines the frame priority where values from 0 (best effort) to 7 (highest) are allowed.
remoteMulticast Subscription Address	ApplicationEndpoint	*	ref	This reference defines the remote multicast subscribed addresses of service consumers. This reference shall ONLY be used if the remote address of the clients is determined from the configuration and not at runtime. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=remoteMulticastSubscriptionAddress, remoteMulticastSubscriptionAddress.applicationEndpoint, remoteMulticastSubscriptionAddress.variationPoint.short Label vh.latestBindingTime=postBuild
remoteUnicast Address	ApplicationEndpoint	*	ref	This reference defines the remote addresses of service consumers. This reference shall ONLY be used if the remote address of the clients is determined from the configuration and not at runtime. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=remoteUnicastAddress, remoteUnicast Address.applicationEndpoint, remoteUnicast Address.variationPoint.shortLabel vh.latestBindingTime=postBuild
sdServerTimer Config	SomeipSdServer ServiceInstanceConfig	0..1	ref	Server specific configuration settings relevant for the SOME/IP service discovery. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=sdServerTimerConfig, sdServerTimer Config.someipSdServerServiceInstanceConfig, sdServer TimerConfig.variationPoint.shortLabel vh.latestBindingTime=postBuild
serviceIdentifier	PositiveInteger	0..1	attr	This attribute represents the ability to describe the SOME/IP service ID that is offered.

Table E.114: ProvidedServiceInstance

Class	**RPortComSpec** (abstract)			
Note	Communication attributes of a required PortPrototype. This class will contain attributes that are valid for all kinds of require-ports, independent of client-server or sender-receiver communication patterns.			
Base	*ARObject*			
Subclasses	*AbstractPersistencyRequireComSpec*, ClientComSpec, ModeSwitchReceiverComSpec, NvRequireComSpec, ParameterRequireComSpec, *ReceiverComSpec*			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table E.115: RPortComSpec

Class	**RPortPrototype**			
Note	Component port requiring a certain port interface.			
Base	*ARObject*, *AbstractRequiredPortPrototype*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
requiredInterface	PortInterface	0..1	tref	The interface that this port requires. **Stereotypes:** isOfType **Tags:** atp.Splitkey=requiredInterface

Table E.116: RPortPrototype

Class	**RecordValueSpecification**			
Note	Specifies the values for a record.			
Base	*ARObject*, *CompositeValueSpecification*, *ValueSpecification*			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValueSpecification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, CompositeRuleBasedValueSpecification.argument, ConstantSpecification.valueSpec, CryptoServiceKey.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElementCondition.compareValue, DiagnosticEnvSovdDataCondition.compareValue, FieldSenderComSpec.initValue, ISignal.initValue, ISignal.timeoutSubstitutionValue, NonqueuedReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, NvProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.initValue, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireComSpec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, PortDefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalidValue, VariableDataPrototype.initValue			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
field (ordered)	ValueSpecification	*	aggr	The value for a single record field. This could also be mapped explicitly to a record element of the data type using the shortName of the ValueSpecification. But this would introduce a relationship to the data type that is too strong. As of now, it is only important that the structure of the data type matches the structure of the Value Specification independently of the shortNames. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table E.117: RecordValueSpecification

Primitive	**Ref**			
Note	This primitive denotes a name based reference. For detailed syntax see the xsd.pattern. • first slash (relative or absolute reference) [optional] • Identifier [required] • a sequence of slashes and Identifiers [optional] This primitive is used by the meta-model tools to create the references. **Tags:** xml.xsd.customType=REF xml.xsd.pattern=/?[a-zA-Z][a-zA-Z0-9_]{0,127}(/[a-zA-Z][a-zA-Z0-9_]{0,127})* xml.xsd.type=string			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Identifier	0..1	attr	This attribute reflects the base to be used for this reference. **Tags:** xml.attribute=true
blueprintValue	String	0..1	attr	This represents a description that documents how the value shall be defined when deriving objects from the blueprint. **Tags:** atp.Status=valid xml.attribute=true
index	PositiveInteger	0..1	attr	This attribute supports the use case to point on specific elements in an array. This is in particular required if arrays are used to implement particular data objects. The counting of array indices starts with the value 0, i.e. the index of the first array element is 0. **Tags:** xml.attribute=true

Table E.118: Ref

Class	**ReferenceValueSpecification**			
Note	Specifies a reference to a data prototype to be used as an initial value for a pointer in the software.			
Base	*ARObject*, *ValueSpecification*			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValue Specification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, ConstantSpecification.valueSpec, CryptoService Key.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElement Condition.compareValue, DiagnosticEnvSovdDataCondition.compareValue, FieldSenderComSpec.init Value, ISignal.initValue, ISignal.timeoutSubstitutionValue, NonqueuedReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, Nv ProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.init Value, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireCom Spec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, Port DefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalidValue, VariableData Prototype.initValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
referenceValue	DataPrototype	0..1	ref	The referenced data prototype. **Tags:** atp.Splitkey=referenceValue

Table E.119: ReferenceValueSpecification

Class	***Referrable*** (abstract)			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
Base	*ARObject*			
Subclasses	*AtpDefinition*, BswDistinguishedPartition, *BswModuleCallPoint*, BswModuleClientServerEntry, BswVariableAccess, CouplingPortTrafficClassAssignment, *CppImplementationDataTypeContextTarget*, *DiagnosticEnvModeElement*, EthernetPriorityRegeneration, ExclusiveAreaNestingOrder, *HwDescriptionEntity*, *ImplementationProps*, ModeTransition, *MultilanguageReferrable*, NmNetworkHandle, PncMappingIdent, *SingleLanguageReferrable*, SoConIPduIdentifier, SomeipRequiredEventGroup, TpConnectionIdent			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
shortName	Identifier	1	attr	This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference. **Stereotypes:** atpIdentityContributor **Tags:** xml.enforceMinMultiplicity=true xml.sequenceOffset=-100
shortName Fragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments. **Tags:** atp.Splitkey=shortNameFragment xml.sequenceOffset=-90

Table E.120: Referrable

Class	**Sd**			
Note	This class represents a primitive element in a special data group.			
Base	*ARObject*			
Aggregated by	SdgContents.sd			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
gid	NameToken	0..1	attr	This attributes specifies an identifier. Gid comes from the SGML/XML-Term "Generic Identifier" which is the element name in XML. The role of this attribute is the same as the name of an XML - element. **Tags:** xml.attribute=true
value	VerbatimStringPlain	0..1	attr	This is the value of the special data. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.typeElement=false xml.typeWrapperElement=false
xmlSpace	XmlSpaceEnum	0..1	attr	This attribute is used to signal an intention that in that element, white space should be preserved by applications. It is defined according to xml:space as declared by W3C. **Tags:** xml.attribute=true xml.attributeRef=true xml.enforceMinMultiplicity=true xml.name=space xml.nsPrefix=xml

Table E.121: Sd

Class	SecOcCryptoServiceMapping			
Note	This meta-class has the ability to represent a crypto service mapping for the Pdu-based communication via SecOC.			
Base	*ARObject*, *CryptoServiceMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SystemMapping.cryptoServiceMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
authentication	CryptoServicePrimitive	0..1	ref	This reference identifies the applicable crypto primitive for the authentication. **Tags:** atp.Splitkey=authentication
cryptoService Key	CryptoServiceKey	0..1	ref	This reference identifies the applicable crypto key. **Tags:** atp.Splitkey=cryptoServiceKey
cryptoService Queue	CryptoServiceQueue	0..1	ref	This reference identifies the CryptoServiceQueue the processing of this SecOcCryptoServiceMapping shall be performed in. **Tags:** atp.Splitkey=cryptoServiceQueue

Table E.122: SecOcCryptoServiceMapping

Class	SecureCommunicationAuthenticationProps			
Note	Authentication properties used to configure SecuredIPdus.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SecureCommunicationPropsSet.authenticationProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
authInfoTx Length	PositiveInteger	0..1	attr	This attribute defines the length in bits of the authentication code to be included in the payload of the authenticated Pdu.

Table E.123: SecureCommunicationAuthenticationProps

Class	SecureCommunicationFreshnessProps			
Note	Freshness properties used to configure SecuredIPdus.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SecureCommunicationPropsSet.freshnessProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
freshness CounterSync Attempts	PositiveInteger	0..1	attr	This attribute defines the number of Freshness Counter re-synchronization attempts when a verification failed for a Secured I-PDU. If the value is zero, there will be no additional verification attempt to synchronize with a potentially better fitting Freshness Counter value. This attribute is only applicable if useFreshnessTimestamp is FALSE.
freshness TimestampTime PeriodFactor	PositiveInteger	0..1	attr	This attribute defines a factor that specifies the time period for the Freshness Timestamp. It holds a multiplication factor that specifies the concrete meaning of a Freshness Timestamp increment by one on basis of microseconds.
freshnessValue Length	PositiveInteger	0..1	attr	This attribute defines the complete length in bits of the Freshness Value. As long as the key doesn't change the counter shall not overflow. The length of the counter shall be determined based on the expected life time of the corresponding key and frequency of usage of the counter.



△

Class	SecureCommunicationFreshnessProps			
freshnessValue TxLength	PositiveInteger	0..1	attr	This attribute defines the length in bits of the Freshness Value to be included in the payload of the Secured I-PDU. This length is specific to the least significant bits of the complete Freshness Counter. If the attribute is 0 no Freshness Value is included in the Secured I-PDU.
useFreshness Timestamp	Boolean	0..1	attr	This attribute specifies whether the Freshness Value is generated through individual Freshness Counters or by a Timestamps. The value is set to TRUE when Timestamps are used.

Table E.124: SecureCommunicationFreshnessProps

Class	SecuredIPdu			
Note	If useAsCryptographicPdu is not set or set to false this IPdu contains the payload of an Authentic IPdu supplemented by additional Authentication Information (Freshness Counter and an Authenticator). If useAsCryptographicPdu is set to true this IPdu contains the Authenticator for a payload that is transported in a separate message. The separate Authentic IPdu is described by the Pdu that is referenced with the payload reference from this SecuredIPdu. **Tags:** atp.recommendedPackage=Pdus			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *IPdu*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Pdu*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
authentication Props	SecureCommunication AuthenticationProps	0..1	ref	Reference to authentication properties that are valid for this SecuredIPdu. **Tags:** atp.Splitkey=authenticationProps
dynamic RuntimeLength Handling	Boolean	0..1	attr	Defines whether the length information for handling this SecuredIPdu with SecuredIPdu.useSecuredPdu Header=noHeader is taken from the configuration or from the actually provided length information during runtime. true: SecuredIPdu length information is taken from the actually provided length information during runtime. false: SecuredIPdu length information is taken from the configuration.
freshnessProps	SecureCommunication FreshnessProps	0..1	ref	Reference to freshness properties that are valid for this SecuredIPdu. **Tags:** atp.Splitkey=freshnessProps
payload	PduTriggering	0..1	ref	Reference to a Pdu that will be protected against unauthorized manipulation and replay attacks. **Tags:** atp.Splitkey=payload
secure Communication Props	SecureCommunication Props	0..1	aggr	Specific configuration properties for this SecuredIPdu. **Tags:** atp.Splitkey=secureCommunicationProps
useAs Cryptographic IPdu	Boolean	0..1	attr	If this attribute is set to true the SecuredIPdu contains the Authentication Information for an AuthenticIPdu that is transmitted in a separate message. The AuthenticIPdu contains the original payload, i.e. the secured data. If this attribute is set to false this SecuredIPdu contains the payload of an Authentic IPdu supplemented by additional Authentication Information.

▽

△

Class	SecuredIPdu			
useSecuredPdu Header	SecuredPduHeader Enum	0..1	attr	This attribute defines the size of the header which is inserted into the SecuredIPdu. If this attribute is set to anything but noHeader, the SecuredIPdu contains the Secured I-PDU Header to indicate the length of the AuthenticIPdu. The AuthenticIPdu contains the original payload, i.e. the secured data.

Table E.125: SecuredIPdu

Class	SenderReceiverInterface			
Note	A sender/receiver interface declares a number of data elements to be sent and received. **Tags:** atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DataInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dataElement	VariableDataPrototype	*	aggr	The data elements of this `SenderReceiverInterface`. **Tags:** atp.Splitkey=dataElement, dataElement.shortName
invalidationPolicy	InvalidationPolicy	*	aggr	InvalidationPolicy for a particular dataElement **Tags:** atp.Splitkey=invalidationPolicy
metaDataItemSet	MetaDataItemSet	*	aggr	This aggregation defines fixed sets of meta-data items associated with `dataElement`s of the enclosing `SenderReceiverInterface` **Tags:** atp.Splitkey=metaDataItemSet

Table E.126: SenderReceiverInterface

Class	*ServiceNeeds* (abstract)			
Note	This expresses the abstract needs that a Software Component or Basic Software Module has on the configuration of an AUTOSAR Service to which it will be connected. "Abstract needs" means that the model abstracts from the Configuration Parameters of the underlying Basic Software.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	BswMgrNeeds, ChargeManagerNeeds, ComMgrUserNeeds, CryptoKeyManagementNeeds, *CryptoNeeds*, CryptoServiceJobNeeds, CryptoServiceNeeds, *DiagnosticCapabilityElement*, DltUserNeeds, *DoIpServiceNeeds*, EcuStateMgrUserNeeds, ErrorTracerNeeds, FunctionInhibitionAvailabilityNeeds, FunctionInhibitionNeeds, GeneralPurposeTimerServiceNeeds, GlobalSupervisionNeeds, IdsMgrCustomTimestampNeeds, IdsMgrNeeds, IndicatorStatusNeeds, J1939DcmDm19Support, J1939RmIncomingRequestServiceNeeds, J1939RmOutgoingRequestServiceNeeds, NvBlockNeeds, SecureOnBoardCommunicationNeeds, SupervisedEntityCheckpointNeeds, SupervisedEntityNeeds, SyncTimeBaseMgrUserNeeds, VendorSpecificServiceNeeds			
Aggregated by	BswServiceDependency.serviceNeeds, SwcServiceDependency.serviceNeeds			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table E.127: ServiceNeeds

Class	ServiceSwComponentType
Note	ServiceSwComponentType is used for configuring services for a given ECU. Instances of this class are only to be created in ECU Configuration phase for the specific purpose of the service configuration. **Tags:** atp.recommendedPackage=SwComponentTypes

▽

△

Class	ServiceSwComponentType			
Base	ARElement, ARObject, AtomicSwComponentType, AtpBlueprint, AtpClassifier, AtpType, Collectable Element, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table E.128: ServiceSwComponentType

Class	SoConIPduIdentifier			
Note	Identification of Pdu content on a socket connection. This Identifier is required in case that multiple Pdus are transmitted over the same socket connection.			
Base	ARObject, Referrable			
Aggregated by	SocketConnectionIpduIdentifierSet.iPduIdentifier			
Attribute	Type	Mult.	Kind	Note
headerId	PositiveInteger	0..1	attr	If multiple Pdus are transmitted over the same connection this headerId can be used to distinguish between the different Pdus. For the constraints on constructing the headerId for SOME/IP also see PRS_SOMEIP_00245.
pduCollection PduTimeout	TimeValue	0..1	attr	Defines the timeout in seconds the PDU collection shall be transmitted at the latest after this PDU has been put into the buffer.
pduCollection Semantics	PduCollection SemanticsEnum	0..1	attr	Specifies if the referenced PduTriggering shall be collected using a queued (i.e. all PDU instances) or last-is-best (i.e. only the last PDU instance) semantics. If this attribute is not present the behavior of "queued" is assumed.
pduCollection Trigger	PduCollectionTrigger Enum	0..1	attr	Defines whether the referenced Pdu contributes to the triggering of the socket transmission if Pdu collection is enabled for this socket.
pduTriggering	PduTriggering	0..1	ref	Reference to a Pdu that is transmitted over a socket connection. **Tags:** atp.Splitkey=pduTriggering

Table E.129: SoConIPduIdentifier

Class	SwComponentPrototype			
Note	Role of a software component within a composition.			
Base	ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable			
Aggregated by	AtpClassifier.atpFeature, CompositionSwComponentType.component			
Attribute	Type	Mult.	Kind	Note
type	SwComponentType	0..1	tref	Type of the instance. **Stereotypes:** isOfType **Tags:** atp.Splitkey=type

Table E.130: SwComponentPrototype

Class	***SwConnector*** (abstract)			
Note	The base class for connectors between ports. Connectors have to be identifiable to allow references from the system constraint template.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	AssemblySwConnector, DelegationSwConnector, PassThroughSwConnector			
Aggregated by	*AtpClassifier*.atpFeature, CompositionSwComponentType.connector			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
mapping	PortInterfaceMapping	0..1	ref	Reference to a `PortInterfaceMapping` specifying the mapping of unequal named `PortInterface` elements of the two different `PortInterface`s typing the two `PortPrototype`s which are referenced by the `SwConnector`. **Tags:** atp.Splitkey=mapping

Table E.131: SwConnector

Class	«atpVariation» **SwDataDefProps**
Note	This class is a collection of properties relevant for data objects under various aspects. One could consider this class as a "pattern of inheritance by aggregation". The properties can be applied to all objects of all classes in which SwDataDefProps is aggregated. Note that not all of the attributes or associated elements are useful all of the time. Hence, the process definition (e.g. expressed with an OCL or a Document Control Instance MSR-DCI) has the task of implementing limitations. SwDataDefProps covers various aspects: • Structure of the data element for calibration use cases: is it a single value, a curve, or a map, but also the recordLayouts which specify how such elements are mapped/converted to the DataTypes in the programming language (or in AUTOSAR). This is mainly expressed by properties like swRecordLayout and swCalprmAxisSet • Implementation aspects, mainly expressed by swImplPolicy, swAddrMethod, baseType, implementationDataType and additionalNativeTypeQualifier • Access policy for the MCD system, mainly expressed by swCalibrationAccess • Semantics of the data element, mainly expressed by compuMethod and/or unit, dataConstr, invalid Value • Code generation policy provided by swRecordLayout **Tags:** vh.latestBindingTime=codeGenerationTime
Base	*ARObject*
Aggregated by	*AutosarDataType*.swDataDefProps, CompositeNetworkRepresentation.networkRepresentation, CppImplementationDataTypeElement.swDataDefProps, *DataPrototype*.swDataDefProps, DataPrototypeTransformationProps.networkRepresentationProps, DiagnosticDataElement.swDataDefProps, DiagnosticEnvDataElementCondition.swDataDefProps, DiagnosticExtendedDataRecordElement.swDataDefProps, DiagnosticSovdPrimitiveContentElement.swDataDefProps, DltArgumentProps.networkRep, FlatInstanceDescriptor.swDataDefProps, ImplementationDataTypeElement.swDataDefProps, InstantiationDataDefProps.swDataDefProps, ISignal.networkRepresentationProps, McDataInstance.resultingProperties, ParameterAccess.swDataDefProps, PerInstanceMemory.swDataDefProps, *ReceiverComSpec*.networkRepresentation, SecurityEventContextDataElement.networkRepresentation, *SenderComSpec*.networkRepresentation, SomeipDataPrototypeTransformationProps.networkRepresentation, SwPointerTargetProps.swDataDefProps, SwServiceArg.swDataDefProps, SwSystemconst.swDataDefProps, SystemSignal.physicalProps
Attribute	***Type*** / ***Mult.*** / ***Kind*** / ***Note***

▽

△

Class	«atpVariation» **SwDataDefProps**			
additionalNative TypeQualifier	NativeDeclarationString	0..1	attr	This attribute is used to declare native qualifiers of the programming language which can neither be deduced from the baseType (e.g. because the data object describes a pointer) nor from other more abstract attributes. Examples are qualifiers like "volatile", "strict" or "enum" of the C-language. All such declarations have to be put into one string. **Tags:** xml.sequenceOffset=235
annotation	Annotation	*	aggr	This aggregation allows to add annotations (yellow pads ...) related to the current data object. **Tags:** atp.Splitkey=annotation xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false
baseType	SwBaseType	0..1	ref	Base type associated with the containing data object. **Tags:** atp.Splitkey=baseType xml.sequenceOffset=50
compuMethod	CompuMethod	0..1	ref	Computation method associated with the semantics of this data object. **Tags:** atp.Splitkey=compuMethod xml.sequenceOffset=180
dataConstr	DataConstr	0..1	ref	Data constraint for this data object. **Tags:** atp.Splitkey=dataConstr xml.sequenceOffset=190
displayFormat	DisplayFormatString	0..1	attr	This property describes how a number is to be rendered e.g. in documents or in a measurement and calibration system. **Tags:** xml.sequenceOffset=210
display Presentation	DisplayPresentation Enum	0..1	attr	This attribute controls the presentation of the related data for measurement and calibration tools.
implementation DataType	AbstractImplementation DataType	0..1	ref	This association denotes the ImplementationDataType of a data declaration via its aggregated SwDataDefProps. It is used whenever a data declaration is not directly referring to a base type. Especially • redefinition of an ImplementationDataType via a "typedef" to another ImplementationDatatype • the target type of a pointer (see SwPointerTarget Props), if it does not refer to a base type directly • the data type of an array or record element within an ImplementationDataType, if it does not refer to a base type directly • the data type of an SwServiceArg, if it does not refer to a base type directly **Tags:** atp.Splitkey=implementationDataType xml.sequenceOffset=215

▽

△

Class	«atpVariation» **SwDataDefProps**			
invalidValue	ValueSpecification	0..1	aggr	Optional value to express invalidity of the actual data element. **Tags:** atp.Splitkey=invalidValue xml.sequenceOffset=255
stepSize	Float	0..1	attr	This attribute can be used to define a value which is added to or subtracted from the value of a DataPrototype when using up/down keys while calibrating.
swAddrMethod	SwAddrMethod	0..1	ref	Addressing method related to this data object. Via an association to the same SwAddrMethod it can be specified that several DataPrototypes shall be located in the same memory without already specifying the memory section itself. **Tags:** atp.Splitkey=swAddrMethod xml.sequenceOffset=30
swAlignment	AlignmentType	0..1	attr	The attribute describes the intended typical alignment of the DataPrototype. If the attribute is not defined the alignment is determined by the swBaseType size and the memoryAllocationKeywordPolicy of the referenced Sw AddrMethod. **Tags:** xml.sequenceOffset=33
swBit Representation	SwBitRepresentation	0..1	aggr	Description of the binary representation in case of a bit variable. **Tags:** atp.Splitkey=swBitRepresentation xml.sequenceOffset=60
swCalibration Access	SwCalibrationAccess Enum	0..1	attr	Specifies the read or write access by MCD tools for this data object. **Tags:** xml.sequenceOffset=70
swCalprmAxis Set	SwCalprmAxisSet	0..1	aggr	This specifies the properties of the axes in case of a curve or map etc. This is mainly applicable to calibration parameters. **Tags:** atp.Splitkey=swCalprmAxisSet xml.sequenceOffset=90
swComparison Variable	SwVariableRefProxy	*	aggr	Variables used for comparison in an MCD process. **Tags:** atp.Splitkey=swComparisonVariable xml.sequenceOffset=170 xml.typeElement=false
swData Dependency	SwDataDependency	0..1	aggr	Describes how the value of the data object has to be calculated from the value of another data object (by the MCD system). **Tags:** atp.Splitkey=swDataDependency xml.sequenceOffset=200
swHostVariable	SwVariableRefProxy	0..1	aggr	Contains a reference to a variable which serves as a host-variable for a bit variable. Only applicable to bit objects. **Tags:** atp.Splitkey=swHostVariable xml.sequenceOffset=220 xml.typeElement=false

▽

△

Class	«atpVariation» **SwDataDefProps**			
swImplPolicy	SwImplPolicyEnum	0..1	attr	Implementation policy for this data object. **Tags:** xml.sequenceOffset=230
swIntended Resolution	Numerical	0..1	attr	The purpose of this element is to describe the requested quantization of data objects early on in the design process. The resolution ultimately occurs via the conversion formula present (compuMethod), which specifies the transition from the physical world to the standardized world (and vice-versa) (here, "the slope per bit" is present implicitly in the conversion formula). In the case of a development phase without a fixed conversion formula, a pre-specification can occur through swIntendedResolution. The resolution is specified in the physical domain according to the property "unit". **Tags:** xml.sequenceOffset=240
swInterpolation Method	Identifier	0..1	attr	This is a keyword identifying the mathematical method to be applied for interpolation. The keyword needs to be related to the interpolation routine which needs to be invoked. **Tags:** xml.sequenceOffset=250
swIsVirtual	Boolean	0..1	attr	This element distinguishes virtual objects. Virtual objects do not appear in the memory, their derivation is much more dependent on other objects and hence they shall have a swDataDependency . **Tags:** xml.sequenceOffset=260
swPointerTarget Props	SwPointerTargetProps	0..1	aggr	Specifies that the containing data object is a pointer to another data object. **Tags:** atp.Splitkey=swPointerTargetProps xml.sequenceOffset=280
swRecord Layout	SwRecordLayout	0..1	ref	Record layout for this data object. **Tags:** atp.Splitkey=swRecordLayout xml.sequenceOffset=290
swRefresh Timing	MultidimensionalTime	0..1	aggr	This element specifies the frequency in which the object involved shall be or is called or calculated. This timing can be collected from the task in which write access processes to the variable run. But this cannot be done by the MCD system. So this attribute can be used in an early phase to express the desired refresh timing and later on to specify the real refresh timing. **Tags:** atp.Splitkey=swRefreshTiming xml.sequenceOffset=300
swTextProps	SwTextProps	0..1	aggr	the specific properties if the data object is a text object. **Tags:** atp.Splitkey=swTextProps xml.sequenceOffset=120

▽

△

Class	«atpVariation» **SwDataDefProps**			
swValueBlock Size	Numerical	0..1	attr	This represents the size of a Value Block **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=80
swValueBlock SizeMult (ordered)	Numerical	*	attr	This attribute is used to specify the dimensions of a value block (VAL_BLK) for the case that that value block has more than one dimension. The dimensions given in this attribute are ordered such that the first entry represents the first dimension, the second entry represents the second dimension, and so on. For one-dimensional value blocks the attribute swValue BlockSize shall be used and this attribute shall not exist. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
unit	Unit	0..1	ref	Physical unit associated with the semantics of this data object. This attribute applies if no compuMethod is specified. If both units (this as well as via compuMethod) are specified the units shall be compatible. **Tags:** atp.Splitkey=unit xml.sequenceOffset=350
valueAxisData Type	ApplicationPrimitive DataType	0..1	ref	The referenced ApplicationPrimitiveDataType represents the primitive data type of the value axis within a compound primitive (e.g. curve, map). It supersedes CompuMethod, Unit, and BaseType. **Tags:** atp.Splitkey=valueAxisDataType xml.sequenceOffset=355

Table E.132: SwDataDefProps

Class	**SwPointerTargetProps**			
Note	This element defines, that the data object (which is specified by the aggregating element) contains a reference to another data object or to a function in the CPU code. This corresponds to a pointer in the C-language. The attributes of this element describe the category and the detailed properties of the target which is either a data description or a function signature.			
Base	*ARObject*			
Aggregated by	SwDataDefProps.swPointerTargetProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
swDataDef Props	SwDataDefProps	0..1	aggr	The properties of the target data type. **Tags:** atp.Splitkey=swDataDefProps xml.sequenceOffset=30
targetCategory	Identifier	0..1	attr	This specifies the category of the target: • In case of a data pointer, it shall specify the category of the referenced data. • In case of a function pointer, it could be used to denote the category of the referenced BswModuleEntry. **Tags:** xml.sequenceOffset=5

Table E.133: SwPointerTargetProps

Class	SwRecordLayout			
Note	Defines how the data objects (variables, calibration parameters etc.) are to be stored in the ECU memory. As an example, this definition specifies the sequence of axis points in the ECU memory. Iterations through axis values are stored within the sub-elements swRecordLayoutGroup. **Tags:** atp.recommendedPackage=SwRecordLayouts			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
swRecord LayoutGroup	SwRecordLayoutGroup	0..1	aggr	This is the top level record layout group. **Tags:** atp.Splitkey=swRecordLayoutGroup xml.roleElement=true xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

Table E.134: SwRecordLayout

Class	SwcServiceDependency			
Note	Specialization of ServiceDependency in the context of an SwcInternalBehavior. It allows to associate ports, port groups and (in special cases) data defined for an atomic software component to a given ServiceNeeds element.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceDependency*			
Aggregated by	AdaptiveSwcInternalBehavior.serviceDependency, *AtpClassifier*.atpFeature, SwcInternal Behavior.serviceDependency			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
assignedData	RoleBasedData Assignment	*	aggr	Defines the role of an associated data object of the same component. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=assignedData, assignedData.variation Point.shortLabel vh.latestBindingTime=preCompileTime
assignedPort	RoleBasedPort Assignment	*	aggr	Defines the role of an associated port of the same component. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=assignedPort, assignedPort.variation Point.shortLabel vh.latestBindingTime=preCompileTime
representedPort Group	PortGroup	0..1	ref	This reference specifies an association between the ServiceNeeeds and a PortGroup, for example to request a communication mode which applies for communication via these ports. The referred PortGroup shall be local to this atomic SWC, but via the links between the Port Groups, a tool can evaluate this information such that all the ports linked via this port group on the same ECU can be found. **Tags:** atp.Splitkey=representedPortGroup
serviceNeeds	ServiceNeeds	0..1	aggr	The associated ServiceNeeds. **Tags:** atp.Splitkey=serviceNeeds, serviceNeeds.short Name

Table E.135: SwcServiceDependency

Class	SystemSignal			
Note	The system signal represents the communication system's view of data exchanged between SW components which reside on different ECUs. The system signals allow to represent this communication in a flattened structure, with exactly one system signal defined for each data element prototype sent and received by connected SW component instances. **Tags:** atp.recommendedPackage=SystemSignals			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dynamicLength	Boolean	0..1	attr	The length of dynamic length signals is variable in run-time. Only a maximum length of such a signal is specified in the configuration (attribute length in ISignal element).
physicalProps	SwDataDefProps	0..1	aggr	Specification of the physical representation. **Tags:** atp.Splitkey=physicalProps

Table E.136: SystemSignal

Class	TextValueSpecification			
Note	The purpose of TextValueSpecification is to define the labels that correspond to enumeration values.			
Base	*ARObject*, *ValueSpecification*			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValue Specification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, ConstantSpecification.valueSpec, CryptoService Key.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElement Condition.compareValue, DiagnosticEnvSovdDataCondition.compareValue, FieldSenderComSpec.init Value, ISignal.initValue, ISignal.timeoutSubstitutionValue, MetaDataItem.metaDataItemType, Nonqueued ReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, Nonqueued SenderComSpec.initValue, NvProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInit Value, NvRequireComSpec.initValue, ParameterDataPrototype.initValue, ParameterProvideComSpec.init Value, ParameterRequireComSpec.initValue, PersistencyDataRequiredComSpec.initValue, Persistency KeyValuePair.initValue, PortDefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, Record ValueSpecification.field, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalid Value, VariableDataPrototype.initValue			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
value	VerbatimString	0..1	attr	This is the value itself. Note that vt uses the | operator to separate the values for the different bitfield masks in case that the semantics of the related DataPrototype is described by means of a BITFIELD_TEXTTABLE in the associated CompuMethod.

Table E.137: TextValueSpecification

Class	TlsCryptoServiceMapping			
Note	This meta-class has the ability to represent a crypto service mapping for the socket-based configuration of Transport Layer Security (TLS).			
Base	*ARObject*, *CryptoServiceMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SystemMapping.cryptoServiceMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**TlsCryptoServiceMapping**			
keyExchange	CryptoServicePrimitive	*	ref	This reference identifies the shared(i.e. applicable for each of the aggregated cipher suites) crypto service primitive for the execution of key exchange during the handshake phase. **Tags:** atp.Splitkey=keyExchange
tlsCipherSuite	TlsCryptoCipherSuite	*	aggr	This aggregation represents the collection of supported cipher suites. **Tags:** atp.Splitkey=tlsCipherSuite, tlsCipherSuite.shortName
useClientAuthenticationRequest	Boolean	0..1	attr	Defines if client authentication shall be applied for this TLS connection.
useSecurityExtensionRecordSizeLimit	Boolean	0..1	attr	Defines if the security extension for max_fragment_length shall be supported as defined in IETF RFC 8449, chapter 4.1.

Table E.138: TlsCryptoServiceMapping

Class	**TlsPskIdentity**			
Note	This element is used to describe the pre-shared key shared during the handshake among the communication parties, to establish a TLS connection if the handshake is based on the existence of a pre-shared key.			
Base	*ARObject*			
Aggregated by	TlsCryptoCipherSuite.pskIdentity			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
pskIdentity	String	0..1	attr	This attribute provides the key identification.
pskIdentityHint	String	0..1	attr	This attribute provides the identity hint for a pre-shared key.

Table E.139: TlsPskIdentity

Class	***TransformationProps*** (abstract)			
Note	This meta-class represents a abstract base class for transformation settings.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ApSomeipTransformationProps, SOMEIPTransformationProps, UserDefinedTransformationProps			
Aggregated by	TransformationPropsSet.transformationProps			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table E.140: TransformationProps

Enumeration	**TransportLayerProtocolEnum**
Note	This enumeration allows to choose a TCP/IP transport layer protocol. This Enumeration is only used by the AUTOSAR Adaptive Platform.
Aggregated by	SomeipEventDeployment.transportProtocol, SomeipMethodDeployment.transportProtocol
Literal	*Description*

▽

△

Enumeration	**TransportLayerProtocolEnum**
tcp	Transmission control protocol **Tags:** atp.EnumerationLiteralIndex=1
udp	User datagram protocol **Tags:** atp.EnumerationLiteralIndex=0

Table E.141: TransportLayerProtocolEnum

Class	**TriggerInterface**			
Note	A trigger interface declares a number of triggers that can be sent by an trigger source. **Tags:** atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
trigger	Trigger	*	aggr	The Trigger of this trigger interface. **Tags:** atp.Splitkey=trigger, trigger.shortName

Table E.142: TriggerInterface

Class	**Unit**			
Note	This is a physical measurement unit. All units that might be defined should stem from SI units. In order to convert one unit into another factor and offset are defined. For the calculation from SI-unit to the defined unit the factor (factorSiToUnit) and the offset (offsetSiTo Unit) are applied as follows: x [{unit}] := y * {siUnit} * factorSiToUnit [{unit}/{siUnit}] + offsetSiToUnit [{unit}] For the calculation from a unit to SI-unit the reciprocal of the factor (factorSiToUnit) and the negation of the offset (offsetSiToUnit) are applied. y [{siUnit}] := (x*{unit} - offsetSiToUnit [{unit}]) / factorSiToUnit [{unit}/{siUnit}] **Tags:** atp.recommendedPackage=Units			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
displayName	SingleLanguageUnit Names	0..1	aggr	This specifies how the unit shall be displayed in documents or in user interfaces of tools.The displayName corresponds to the Unit.Display in an ASAM MCD-2MC file. **Tags:** atp.Splitkey=displayName xml.sequenceOffset=20
factorSiToUnit	Float	0..1	attr	This is the factor for the conversion from SI Units to units. The inverse is used for conversion from units to SI Units. **Tags:** xml.sequenceOffset=30
offsetSiToUnit	Float	0..1	attr	This is the offset for the conversion from and to siUnits. **Tags:** xml.sequenceOffset=40

▽

△

Class	Unit			
physical Dimension	PhysicalDimension	0..1	ref	This association represents the physical dimension to which the unit belongs to. Note that only values with units of the same physical dimensions might be converted. **Tags:** atp.Splitkey=physicalDimension xml.sequenceOffset=50

Table E.143: Unit

Class	*ValueSpecification* (abstract)			
Note	Base class for expressions leading to a value which can be used to initialize a data object.			
Base	*ARObject*			
Subclasses	*AbstractRuleBasedValueSpecification*, ApplicationValueSpecification, *CompositeValueSpecification*, ConstantReference, NotAvailableValueSpecification, NumericalValueSpecification, ReferenceValueSpecification, TextValueSpecification			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValueSpecification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, ConstantSpecification.valueSpec, CryptoServiceKey.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElementCondition.compareValue, DiagnosticEnvSovdDataCondition.compareValue, FieldSenderComSpec.initValue, ISignal.initValue, ISignal.timeoutSubstitutionValue, NonqueuedReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, NvProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.initValue, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireComSpec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, PortDefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalidValue, VariableDataPrototype.initValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
shortLabel	Identifier	0..1	attr	This can be used to identify particular value specifications for human readers, for example elements of a record type.

Table E.144: ValueSpecification

Class	VlanConfig			
Note	VLAN Configuration attributes			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EthernetPhysicalChannel.vlan			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
vlanIdentifier	PositiveInteger	0..1	attr	A VLAN is identified by this attribute according to IEEE 802.1Q. The allowed values range is from 0..4095.

Table E.145: VlanConfig

Class	VlanMembership			
Note	Static logical channel or VLAN binding to a switch-port. The reference to an EthernetPhysicalChannel without a VLAN defined represents the handling of untagged frames.			
Base	*ARObject*			
Aggregated by	CouplingPort.vlanMembership			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**VlanMembership**			
defaultPriority	PositiveInteger	0..1	attr	Standard output-priority outgoing Frames will be tagged with. Defines the priority that received frames are assigned together with the VLAN Id (defaultVlan). The values from 0 (best effort) to 7 (highest) are allowed. In case modifyVlan and an already tagged received frame, the actual priority of the received frame is not modified.
dhcpAddress Assignment	DhcpServer Configuration	0..1	aggr	Specifies the IP Address which will be assigned to a DHCP Client at this SwitchPort. If no dhcpAddress Assignment is provided all DHCP-Discover messages received at this Port will be discarded by the DHCP Server. **Tags:** atp.Splitkey=dhcpAddressAssignment
sendActivity	EthernetSwitchVlan EgressTaggingEnum	0..1	attr	Attribute denotes whether a VLAN tagged ethernet frame will be 1. sent with its VLAN tag (sentTagged) 2. sent without a VLAN tag (sentUntagged) 3. will be dropped at this port (notSent or VLAN not member of this list)
vlan	EthernetPhysical Channel	0..1	ref	References a channel that represents a VLAN or an untagged channel. **Tags:** atp.Splitkey=vlan

Table E.146: VlanMembership

F Change History of AUTOSAR Traceable Items

The content of this appendix chapter is *informative* in nature and shall **not** be considered as *normative* content.

Please note that the lists in this chapter also include constraints and specification items that have been removed from the specification in a later version. These constraints and specification items do not appear as hyperlinks in the document.

F.1 Traceable item history of this document according to AUTOSAR Release R17-03 (original version)

F.1.1 Created Constraints in R17-03

Number	Heading
[constr_1473]	No support for `PRPortPrototype`
[constr_1474]	`SwDataDefProps` applicable to `ImplementationDataType`s exclusive to the *AUTOSAR adaptive platform*
[constr_1475]	`ImplementationDataType` of `category` `STRING` is limited
[constr_1476]	`ImplementationDataType` of `category` `VECTOR` is limited
[constr_1477]	`ImplementationDataType` of `category` `ASSOCIATIVE_MAP` is limited
[constr_1478]	`SwDataDefProps` applicable to `ApplicationDataType`s exclusive to the *AUTOSAR adaptive platform*
[constr_1479]	No support for certain values of `ImplementationDataType.category`
[constr_1480]	Mutual existence of `CompositionDataPrototypeRef.elementInImplDatatype` vs. attributes of `CompositionDataPrototypeRef.dataPrototype`
[constr_1481]	Usage of `CompositionDataPrototypeRef` in the *AUTOSAR adaptive platform*
[constr_1482]	Mapping of service interfaces vs. mapping of service interface elements
[constr_1483]	Applicability of a `ServiceInterface`
[constr_1484]	Applicability of `ModeDependentStartupConfig.executionDependency`
[constr_1485]	No `subElement` for `ImplementationDataType` of `category` `STRING`
[constr_1486]	`ImplementationDataType` of `category` `STRING` and `SwBaseType`
[constr_1487]	Number of `subElement`s of an `ImplementationDataType` of `category` `ASSOCIATIVE_MAP`
[constr_1488]	Initialization of a `DataPrototype` typed by an `ApplicationAssocMapDataType`
[constr_1489]	Uniqueness of `ApplicationAssocMapValueSpecification.mapElementTuple.key`
[constr_1490]	Allowed value of `category` for reference `AdaptiveModuleInstantiation.process.executable`
[constr_1491]	Reference to `ApplicationError`
[constr_1492]	`SwComponentType` referenced as `Executable.rootSwComponentPrototype.applicationType`

▽

△

Number	Heading
[constr_1493]	ArgumentDataPrototype referenced in the role ApplicationError.errorContext
[constr_1494]	Initial value for event
[constr_1495]	Initial value for field
[constr_1496]	DiagnosticServiceDataMapping.mappedApDataElement shall only refer to specific sub-classes of DataPrototype
[constr_1497]	Attribute optionKind set to commandLineSimpleForm
[constr_1498]	Attribute optionKind set to commandLineShortForm or commandLineLongForm
[constr_1499]	Target SwcServiceDependency of DiagnosticServiceSwMapping.mappedSwcServiceDependencyInExecutable
[constr_1500]	Target SwcServiceDependency of DiagnosticEventPortMapping.swcServiceDependencyInExecutable
[constr_1501]	Target SwcServiceDependency of DiagnosticOperationCyclePortMapping.swcServiceDependencyInExecutable
[constr_1502]	Target SwcServiceDependency of DiagnosticEnableConditionPortMapping.swcServiceDependencyInExecutable
[constr_1503]	Target SwcServiceDependency of DiagnosticStorageConditionPortMapping.swcServiceDependencyInExecutable
[constr_1504]	Number of Process.modeDependentStartupConfig that refer to the same ModeDeclaration
[constr_1505]	Number of Process.modeDependentStartupConfig that do not refer to a ModeDeclaration
[constr_1507]	PortInterfaceToDataTypeMapping is only applicable to ServiceInterface
[constr_1508]	BaseTypeDirectDefinition.nativeDeclaration shall not be set to the value enum
[constr_3320]	Aggregation of CommunicationConnector by Machine
[constr_3287]	Mandatory information of a ProvidedSomeipServiceInstance
[constr_3288]	IP configuration restriction for unicastNetworkEndpoints
[constr_3290]	Usage of ServiceInstancePortConfig defined for a ProvidedSomeipServiceInstance
[constr_3291]	SomeipServiceInstanceToMachineMapping.portConfig aggregation restriction
[constr_3293]	Mandatory information of a RequiredSomeipServiceInstance
[constr_3296]	Usage of ServiceInstancePortConfig defined for a RequiredSomeipServiceInstance
[constr_5155]	SomeipServiceInstanceToMachineMapping only supports a single Address Family
[constr_3300]	Allowed ServiceMethodDeployment.method references
[constr_3301]	Allowed ServiceEventDeployment.event references
[constr_3302]	Allowed ServiceFieldDeployment.field references
[constr_3303]	ANY not allowed for SomeipServiceInterface.serviceInterfaceVersion
[constr_3304]	Value of attribute SomeipEventGroup.eventGroupId shall be unique
[constr_3305]	Value of attribute SomeipEvent.eventId shall be unique
[constr_3306]	Value of attribute SomeipMethod.methodId shall be unique

▽

△

Number	Heading
[constr_5156]	SomeipEvent.transportProtocol setting to udp and the impact on ProvidedSomeip-ServiceInstances
[constr_3308]	SomeipEvent.transportProtocol setting to tcp and the impact on ProvidedSomeip-ServiceInstances
[constr_3309]	SomeipMethod.transportProtocol setting to udp and the impact on Provided-SomeipServiceInstances
[constr_3310]	SomeipMethod.transportProtocol setting to tcp and the impact on Provided-SomeipServiceInstances
[constr_3320]	Aggregation of CommunicationConnector by Machine
[constr_3349]	Usage of ApplicationAssocMapDataType is limited
[constr_3350]	Consistent value of category for AdaptiveAutosarApplications referencing an Executable
[constr_3351]	SOME/IP segmentation allowed for udp SomeipEvents
[constr_3352]	SOME/IP segmentation allowed for udp SomeipMethods
[constr_3353]	Restriction in usage of ApSomeipTransformationProps.sizeOfArrayLength-Field
[constr_3354]	Restriction in usage of ApSomeipTransformationProps.sizeOfStruct-LengthField
[constr_3355]	Restriction in usage of ApSomeipTransformationProps.sizeOfUnionLength-Field
[constr_3356]	Restriction in usage of ApSomeipTransformationProps.alignment
[constr_3357]	Restriction in usage of ApSomeipTransformationProps.sizeOfUnionType-SelectorField
[constr_3358]	Usage of PortPrototype and TransportLayerIndependentInstanceId to define the same Service Instance is not allowed.
[constr_3359]	RPortPrototypeProps are related only to RPortPrototypes.
[constr_3360]	RPortPrototypeProps are related only to TransportLayerIndependentInstanceIds representing a consumer Service Instance.
[constr_3361]	Selective definition of serialization settings.
[constr_3362]	SomeipEvents aggregated by a SomeipField
[constr_3363]	SomeipMethods aggregated by a SomeipField

Table F.1: Added Constraints in original version

F.1.2 Created Specification Items in R17-03

Number	Heading
[TPS_MANI_01000]	Definition of the term Manifest
[TPS_MANI_01001]	Meaning of ServiceInterface
[TPS_MANI_01002]	Semantics of a ServiceInterfaceMapping
[TPS_MANI_01003]	Limitations of the applicability of ServiceInterfaceMapping

▽

△

Number	Heading
[TPS_MANI_01004]	Semantics of `ServiceInterface.namespace`
[TPS_MANI_01005]	The definition of the namespace of a `ServiceInterface` may follow a hierarchical pattern
[TPS_MANI_01006]	Ordered definition of `ServiceInterface.namespace`
[TPS_MANI_01007]	Service-oriented **communication** and service **discovery**
[TPS_MANI_01008]	Semantics of `AdaptiveAutosarApplication`
[TPS_MANI_01009]	Standardized values of `AdaptiveAutosarApplication.category`
[TPS_MANI_01010]	Root element for a hierarchical software-component
[TPS_MANI_01011]	Connection between application design and application deployment
[TPS_MANI_01012]	Formal modeling of application startup behavior
[TPS_MANI_01013]	Semantics of meta-class `ModeDependentStartupConfig`
[TPS_MANI_01014]	Semantics of meta-class `StartupConfigSet`
[TPS_MANI_01015]	Semantics of meta-class `StartupOption`
[TPS_MANI_01016]	Category of `ApplicationAssocMapDataType`
[TPS_MANI_01017]	Relation of startup configuration to resource groups
[TPS_MANI_01018]	`ImplementationDataType` of `category` `VECTOR`
[TPS_MANI_01019]	`Manifest` content may apply to different aspects of the *AUTOSAR adaptive platform*
[TPS_MANI_01020]	Serialization format of the *Manifest* in AUTOSAR
[TPS_MANI_01021]	Serialization format of *Manifest* content on a machine
[TPS_MANI_01022]	Concept behind `ServiceInterfaceMapping`
[TPS_MANI_01024]	Semantics of `ServiceInterfaceEventMapping`
[TPS_MANI_01025]	Semantics of `ServiceInterfaceFieldMapping`
[TPS_MANI_01026]	Semantics of `ServiceInterfaceMethodMapping`
[TPS_MANI_01027]	Semantics of `ApplicationAssocMapDataType`
[TPS_MANI_01028]	`ImplementationDataType` of `category` `ASSOCIATIVE_MAP`
[TPS_MANI_01029]	Usage of `ImplementationDataType`
[TPS_MANI_01030]	`ImplementationDataType` of `category` `STRING`
[TPS_MANI_01031]	Semantics of CompositionDataPrototypeRef
[TPS_MANI_01032]	Usage of `ServiceInterfaceMapping`
[TPS_MANI_01033]	Semantics of `ServiceInterface.event`
[TPS_MANI_01034]	Semantics of `ServiceInterface.field`
[TPS_MANI_01035]	Semantics of `ServiceInterface.method`
[TPS_MANI_01037]	Diagnostic data mapping on the *AUTOSAR adaptive platform*
[TPS_MANI_01038]	Diagnostic software mapping on the *AUTOSAR adaptive platform*
[TPS_MANI_01039]	Representation of provided service
[TPS_MANI_01040]	Representation of required service
[TPS_MANI_01041]	Startup configuration supports the definition of a launch dependency
[TPS_MANI_01042]	Definition of a linear `ImplementationDataType` of `category` `VECTOR`

▽

△

Number	Heading
[TPS_MANI_01043]	Definition of a rectangular ImplementationDataType of category VECTOR
[TPS_MANI_01044]	Structure of an ImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_01045]	Process.modeDependentStartupConfig that does not refer to a ModeDeclaration
[TPS_MANI_01046]	Semantics of ModeDependentStartupConfig.machineMode
[TPS_MANI_01047]	Existence of SwRecordLayout for an ApplicationPrimitiveDataType of category STRING
[TPS_MANI_01048]	Mapping of DiagnosticEvent to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01049]	Mapping of DiagnosticOperationCycle to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01050]	Mapping of DiagnosticEnableCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01051]	Mapping of DiagnosticStorageCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01052]	Semantics of RPortPrototypeProps.portInstantiationBehavior
[TPS_MANI_01053]	Usage of ComSpecs on the *AUTOSAR adaptive platform*
[TPS_MANI_01054]	Definition of the queue length of an event
[TPS_MANI_01055]	Semantics of ServiceInterface.possibleError
[TPS_MANI_01056]	Semantics of ApplicationError.errorContext
[TPS_MANI_01057]	Semantics of RPortPrototypeProps.searchIntention
[TPS_MANI_01058]	Ability to create a mapping of ApplicationErrors aggregated in the role possibleError
[TPS_MANI_01059]	Different values of optionKind within a StartupConfig.startupOption
[TPS_MANI_01060]	Use cases for the application of DiagnosticServiceDataMapping
[TPS_MANI_01061]	Requirements on scheduling
[TPS_MANI_01062]	ImplementationDataType to generate a C++ enum
[TPS_MANI_01063]	Sharing of ImplementationDataType with enumeration semantics
[TPS_MANI_03000]	Mapping of AdaptivePlatformServiceInstance to PortPrototypes
[TPS_MANI_03001]	Mapping of AdaptivePlatformServiceInstance to a Machine
[TPS_MANI_03002]	IP configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03003]	ProvidedSomeipServiceInstance Fanout
[TPS_MANI_03004]	IPv4 Multicast event destination address
[TPS_MANI_03005]	IPv4 Multicast address range
[TPS_MANI_03006]	IPv6 Multicast address range
[TPS_MANI_03007]	Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03008]	Tcp Transport Protocol Configuration for ProvidedSomeipServiceInstance

▽

△

Number	Heading
[TPS_MANI_03009]	Tcp and Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03010]	Udp Transport Protocol Configuration in case of IP-Multicast
[TPS_MANI_03011]	Server Timing configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03012]	Initial Wait Phase configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03013]	Repetition Wait Phase configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03014]	Main Phase configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03015]	TTL for Offer Service Entries
[TPS_MANI_03016]	Servers RequestResponseDelay for received FindService entries
[TPS_MANI_03017]	Server Capability Records
[TPS_MANI_03018]	Usage of SomeipProvidedEventGroup.multicastThreshold
[TPS_MANI_03019]	TTL for SubscribeEventGroupAck Entries
[TPS_MANI_03020]	Servers RequestResponseDelay for received SubscribeEventGroup entries
[TPS_MANI_03021]	Requirements on the service version from the client's point of view
[TPS_MANI_03022]	Context of RequiredSomeipServiceInstance
[TPS_MANI_03023]	Udp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03024]	Tcp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03025]	Client Timing configuration for a RequiredSomeipServiceInstance
[TPS_MANI_03026]	Initial Wait Phase configuration for a RequiredSomeipServiceInstance
[TPS_MANI_03027]	Repetition Wait Phase configuration for a RequiredSomeipServiceInstance
[TPS_MANI_03028]	TTL for Find Service Entries
[TPS_MANI_03029]	Client Capability Records
[TPS_MANI_03030]	SomeipSdClientEventGroupTimingConfig.timeToLive for SubscribeEventGroup Entries
[TPS_MANI_03031]	Clients RequestResponseDelay for received ServiceOffer entries
[TPS_MANI_03032]	Description of middleware technologies not standardized by AUTOSAR
[TPS_MANI_03035]	Content of the Machine configuration
[TPS_MANI_03036]	ServiceInterface deployment to a middleware transport layer
[TPS_MANI_03037]	Purpose of ServiceMethodDeployment
[TPS_MANI_03038]	Purpose of ServiceEventDeployment
[TPS_MANI_03039]	Purpose of ServiceFieldDeployment
[TPS_MANI_03040]	SOME/IP ServiceInterface binding
[TPS_MANI_03041]	Definition of SOME/IP EventGroups
[TPS_MANI_03042]	Definition of SOME/IP Service Version
[TPS_MANI_03043]	SOME/IP VariableDataPrototype binding
[TPS_MANI_03044]	SOME/IP ClientServerOperation binding

▽

△

Number	Heading
[TPS_MANI_03045]	UserDefined `ServiceInterface` binding
[TPS_MANI_03046]	User defined `VariableDataPrototype` binding
[TPS_MANI_03047]	User defined `ClientServerOperation` binding
[TPS_MANI_03048]	User defined `Field` binding
[TPS_MANI_03049]	Tcp and Udp Transport Protocol Configuration for `RequiredSomeip-ServiceInstance`
[TPS_MANI_03050]	Tcp and Udp Transport Protocol Configuration for `RequiredSomeip-ServiceInstance`
[TPS_MANI_03051]	Usage of SomeipMethod.transportProtocol
[TPS_MANI_03052]	Static IPv4 configuration
[TPS_MANI_03053]	Static IPv6 configuration
[TPS_MANI_03056]	Usage of SomeipEvent.transportProtocol
[TPS_MANI_03057]	SOME/IP `Field` binding
[TPS_MANI_03059]	`RequiredSomeipServiceInstance.requiredServiceInstanceId`
[TPS_MANI_03061]	IPv6 Multicast event destination address
[TPS_MANI_03064]	SOME/IP Service Discovery message exchange configuration
[TPS_MANI_03065]	Hardware resources of the machine
[TPS_MANI_03066]	Description of machine states
[TPS_MANI_03067]	SOME/IP segmentation of udp SomeipEvents
[TPS_MANI_03068]	SOME/IP segmentation of SomeipMethod Calls
[TPS_MANI_03069]	SOME/IP segmentation of SomeipMethod Responses
[TPS_MANI_03070]	Size of a length field for a chosen array
[TPS_MANI_03071]	Size of a length field for a chosen structure
[TPS_MANI_03072]	Size of a length field for a chosen union
[TPS_MANI_03073]	Alignment of a dynamic DataPrototype
[TPS_MANI_03074]	Size of a type selector field for a chosen union
[TPS_MANI_03075]	Byte Order of chosen DataPrototype in the serialized data stream
[TPS_MANI_03094]	`Machine`-specific platform configuration settings
[TPS_MANI_03095]	Implementation-specific platform configuration settings
[TPS_MANI_03096]	`Machine`-specific configuration settings for a generic module
[TPS_MANI_03097]	Implementation-specific configuration settings for a generic module
[TPS_MANI_03098]	`Machine`-specific configuration settings for the OS module
[TPS_MANI_03099]	Implementation-specific configuration settings for the OS module
[TPS_MANI_03100]	Transport layer independent TransportLayerIndependentInstanceIds
[TPS_MANI_03101]	SOME/IP serialization
[TPS_MANI_03102]	UserDefined serialization
[TPS_MANI_03103]	Default size for all array length fields
[TPS_MANI_03104]	Default size for all structure length fields
[TPS_MANI_03105]	Default size for all union length fields

▽

△

Number	Heading
[TPS_MANI_03106]	Default size for all union type selector fields
[TPS_MANI_03107]	Default alignment for all dynamic DataPrototypes
[TPS_MANI_03108]	Default Byte Order for all DataPrototypes
[TPS_MANI_03109]	TransformationProps on the level of DataPrototypes overwrites TransformationProps settings on the level of a ServiceInterface

Table F.2: Added Specification Items in original Version

F.2 Traceable item history of this document according to AUTOSAR Release R17-10

F.2.1 Added Specification Items in R17-10

Number	Heading
[TPS_MANI_01064]	Semantics of attribute method.fireAndForget
[TPS_MANI_01065]	Purpose of PersistencyKeyValueDatabaseInterface
[TPS_MANI_01067]	Purpose of PersistencyFileProxyInterface
[TPS_MANI_01068]	Semantics of PersistencyFileProxyInterface.maxNumberOfFiles
[TPS_MANI_01069]	Further qualification of properties of PortPrototypes typed by PersistencyKeyValueDatabaseInterfaces
[TPS_MANI_01073]	Semantics of PortPrototype typed by PersistencyKeyValueDatabaseInterface
[TPS_MANI_01074]	Specification of encryption of persistent data
[TPS_MANI_01075]	Specification of redundancy of persistent data
[TPS_MANI_01077]	Specification of file encryption
[TPS_MANI_01078]	Semantics of PersistencyPortPrototypeToKeyValueDatabaseMapping
[TPS_MANI_01079]	Semantics of PersistencyKeyValueDatabase
[TPS_MANI_01080]	Semantics of PersistencyFileProxyToFileMapping
[TPS_MANI_01081]	Semantics of PortPrototype typed by PersistencyFileProxyInterface
[TPS_MANI_01082]	Eligibility of DataPrototypes for the definition of optionality
[TPS_MANI_01083]	Optionality is supported for ApplicationDataType as well as ImplementationDataType
[TPS_MANI_01084]	Optionality for a DataPrototype typed by an ApplicationDataType
[TPS_MANI_01085]	Definition of optionality for a DataPrototype typed by an ImplementationDataType
[TPS_MANI_01087]	Interaction with crypto software
[TPS_MANI_01088]	Semantics of CryptoNeed

▽

△

Number	Heading
[TPS_MANI_01089]	Relation between CryptoNeed and PortPrototype
[TPS_MANI_01090]	Modeling of crypto software as a platform module
[TPS_MANI_01091]	Semantics of CryptoJob
[TPS_MANI_01092]	Mapping between CryptoNeed and CryptoJob
[TPS_MANI_01093]	Semantics of CryptoDriver
[TPS_MANI_01094]	Scope of CryptoDriver
[TPS_MANI_01095]	Semantics of CryptoKeySlot
[TPS_MANI_01096]	Semantics of the CryptoPrimitive
[TPS_MANI_01097]	Assignment of TLV data ids for data structures with optional members
[TPS_MANI_01098]	Constraints on the definition of an ImplementationDataType of category VECTOR
[TPS_MANI_01099]	Semantics of ImplementationDataTypeElementExtension
[TPS_MANI_01100]	Semantics of Allocator
[TPS_MANI_01101]	Size-constrained allocation of memory
[TPS_MANI_01102]	Specification of a namespace for an ImplementationDataType of category VECTOR
[TPS_MANI_01103]	Three-level approach to REST modeling
[TPS_MANI_01105]	Semantics of RestServiceInterface
[TPS_MANI_01106]	Specification of capabilities for the receiver of events or field notifiers
[TPS_MANI_01107]	Specification of capabilities for the sender of events or field notifiers
[TPS_MANI_01108]	Specification of capabilities for the caller of a methods or field setter/getter
[TPS_MANI_01109]	Semantics of UploadablePackageElement
[TPS_MANI_01110]	Semantics of SoftwareCluster
[TPS_MANI_01111]	Diagnostic Address of a SoftwareCluster
[TPS_MANI_01112]	Semantics of SoftwareClusterDesign
[TPS_MANI_01113]	Semantics of SoftwareClusterDesign.diagnosticAddress
[TPS_MANI_01114]	Relation of DiagnosticContributionSet to SoftwareCluster
[TPS_MANI_01115]	Specification of executable software within SoftwareCluster
[TPS_MANI_01116]	Reference to model elements included in an uploadable software package
[TPS_MANI_01117]	Semantics of SoftwareClusterDesign.intendedTargetMachine
[TPS_MANI_01118]	Relation between SoftwareClusterDesign and DiagnosticContributionSet
[TPS_MANI_01119]	Reference to model elements from SoftwareClusterDesign
[TPS_MANI_01120]	Recursive definition of RestResourceDef
[TPS_MANI_01121]	Semantics of RestResourceDef.endpoint
[TPS_MANI_01122]	Arguments to endpoints
[TPS_MANI_01123]	System Triggered Event
[TPS_MANI_01124]	Semantics of RestElementDef
[TPS_MANI_01125]	Properties of REST elements can either be primitive or have array semantics

▽

△

Number	Heading
[TPS_MANI_01126]	Definition of string properties
[TPS_MANI_01127]	Limited support for data semantics in RestAbstractNumericalPropertyDef
[TPS_MANI_01128]	Difference between RestIntegerPropertyDef and RestNumberPropertyDef
[TPS_MANI_01129]	RestObjectRef is only needed for specific implementations of REST-based communication
[TPS_MANI_01130]	Structure of a typical URI for a REST service
[TPS_MANI_01131]	Impact of nested REST resources on the structure of REST URI
[TPS_MANI_01132]	Semantics of CompositionDataPrototypeRef
[TPS_MANI_01133]	Optional element of an event
[TPS_MANI_01134]	Optional element in the context of a method
[TPS_MANI_03110]	Allowed components in system description with category category SOFTWARE_COMPONENT_SYSTEM_DESIGN_DESCRIPTION.
[TPS_MANI_03111]	Mapping between method and operation
[TPS_MANI_03112]	Mapping between an event and a dataElement
[TPS_MANI_03113]	Mapping between a field and elements of Classic Platform PortInterfaces
[TPS_MANI_03114]	Usage of AssemblySwConnectors in the System Design model
[TPS_MANI_03115]	Mapping between a fire and forget method and elements of Classic Platform PortInterfaces
[TPS_MANI_03116]	Size of a length field for a chosen string
[TPS_MANI_03117]	Default size for all string length fields
[TPS_MANI_03118]	Semantics of ServiceInterface.method with fireAndForget set to true
[TPS_MANI_03119]	Default value for the attribute fireAndForget of meta-class ClientServerOperation
[TPS_MANI_03120]	Signal-based ServiceInterface binding
[TPS_MANI_03121]	Signal-based VariableDataPrototype binding
[TPS_MANI_03122]	Signal-based Field binding
[TPS_MANI_03123]	Signal-based ClientServerOperation binding
[TPS_MANI_03124]	SignalBasedEventDeployment to ISignalTriggering mapping
[TPS_MANI_03125]	SignalBasedMethodDeployment to ISignalTriggerings mapping
[TPS_MANI_03126]	SignalBasedFieldDeployment to ISignalTriggerings mapping
[TPS_MANI_03127]	Usage of End2EndEventProtectionProps
[TPS_MANI_03128]	Usage of same dataId in case of Multi-Binding
[TPS_MANI_03129]	E2E profile
[TPS_MANI_03130]	Standardized E2EProfileConfiguration.profileName values
[TPS_MANI_03131]	Non-Standardized E2EProfileConfiguration.profileName values
[TPS_MANI_03132]	Semantics of E2E attributes in ReceiverComSpec
[TPS_MANI_03133]	Usage of ServiceInterfaceElementSecureComConfig

▽

△

Number	Heading
[TPS_MANI_03134]	Configuration of supported TLS ciphersuites
[TPS_MANI_03135]	Configuration of TLS PSK Identity
[TPS_MANI_03136]	Configuration of requirements for the TLS cryptographic job
[TPS_MANI_03137]	ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are not relevant in case of TLS communication
[TPS_MANI_03138]	SecOC Security Profile
[TPS_MANI_03139]	Standardized SecOC Security Profiles
[TPS_MANI_03140]	Non-Standardized SecOC Security Profiles
[TPS_MANI_03141]	Mapping between SecOcJobRequirement and CryptoJob
[TPS_MANI_03142]	Mapping between TlsJobRequirement and CryptoJob
[TPS_MANI_03143]	Mapping between PresharedKeyIdentity and CryptoKeySlot
[TPS_MANI_03144]	C++ language binding of ImplementationDataTypes of category STRING
[TPS_MANI_03145]	Description of a function group
[TPS_MANI_03146]	Configuration of timeouts for a selected machine state or function group state
[TPS_MANI_03147]	Mapping of a Process to a Machine
[TPS_MANI_03148]	Description of Core affinity
[TPS_MANI_03149]	Definition of a start-up timeout for a Process
[TPS_MANI_03150]	Definition of a termination timeout for a Process
[TPS_MANI_03151]	Default value for termination timeout
[TPS_MANI_03152]	Assignment of a StateDependentStartupConfig to a function group state
[TPS_MANI_03153]	Semantics of ModeDependentStartupConfig.functionGroupMode
[TPS_MANI_03500]	Definition of platform health management checkpoints
[TPS_MANI_03501]	Definition of platform health management supervised entities
[TPS_MANI_03502]	Enabling of PlatformHealthManagementContribution on a Machine
[TPS_MANI_03503]	Applicability of supervision to a specific Process
[TPS_MANI_03504]	Existence of SupervisionEntity
[TPS_MANI_03505]	Existence of PhmCheckpoint
[TPS_MANI_03506]	Optionality of SupervisionEntity and PhmCheckpoint
[TPS_MANI_03508]	Definition of an AliveSupervision for a PhmCheckpoint
[TPS_MANI_03509]	Definition of a CheckpointTransition
[TPS_MANI_03510]	Definition of LogicalSupervision
[TPS_MANI_03511]	Definition of DeadlineSupervision
[TPS_MANI_03512]	Applicability of global supervision to a specific Process
[TPS_MANI_03513]	Collection of SupervisionEntitys into a global supervision
[TPS_MANI_03514]	Expiration tolerance for GlobalSupervisionEntity
[TPS_MANI_03515]	Expiration tolerance for SupervisionEntity

▽

△

Number	Heading
[TPS_MANI_03516]	Condition evaluation for `HealthChannelSupervision`
[TPS_MANI_03517]	Condition evaluation for `HealthChannelExternalMode`
[TPS_MANI_03518]	`LogicalExpression` definition
[TPS_MANI_03519]	`Rule` definition
[TPS_MANI_03520]	Execution of `PhmActionList` with `actionListExecution=triggeredOnEvaluation`
[TPS_MANI_03521]	Execution of `PhmActionList` with `actionListExecution=triggeredOnChange`
[TPS_MANI_03522]	Definition of actions for application software
[TPS_MANI_03523]	Definition of actions for Platform Instance
[TPS_MANI_03524]	Definition of actions for Watchdog

Table F.3: Added Specification Items in R17-10

F.2.2 Changed Specification Items in R17-10

Number	Heading
[TPS_MANI_01004]	Semantics of `ServiceInterface.namespace`
[TPS_MANI_01006]	Ordered definition of `ServiceInterface.namespace`
[TPS_MANI_01017]	Relation of startup configuration to resource group
[TPS_MANI_01018]	`ImplementationDataType` of `category` `VECTOR`
[TPS_MANI_01030]	`ImplementationDataType` of `category` `STRING`
[TPS_MANI_03000]	Mapping of `AdaptivePlatformServiceInstance` to `PortPrototype`s
[TPS_MANI_03007]	Udp Transport Protocol Configuration for `ProvidedSomeipServiceInstance`
[TPS_MANI_03008]	Tcp Transport Protocol Configuration for `ProvidedSomeipServiceInstance`
[TPS_MANI_03009]	Tcp and Udp Transport Protocol Configuration for `ProvidedSomeipServiceInstance`
[TPS_MANI_03010]	Udp Transport Protocol Configuration in case of IP-Multicast
[TPS_MANI_03018]	Usage of `SomeipProvidedEventGroup.multicastThreshold`
[TPS_MANI_03023]	Udp Transport Protocol Configuration for `RequiredSomeipServiceInstance`
[TPS_MANI_03024]	Tcp Transport Protocol Configuration for `RequiredSomeipServiceInstance`
[TPS_MANI_03049]	Tcp and Udp Transport Protocol Configuration for `RequiredSomeipServiceInstance`
[TPS_MANI_03101]	SOME/IP serialization
[TPS_MANI_03102]	UserDefined serialization
[TPS_MANI_03103]	Default size for all array length fields

▽

△

Number	Heading
[TPS_MANI_03104]	Default size for all structure length fields
[TPS_MANI_03105]	Default size for all union length fields
[TPS_MANI_03106]	Default size for all union type selector fields
[TPS_MANI_03107]	Default alignment for all dynamic DataPrototypes
[TPS_MANI_03108]	Default Byte Order for all DataPrototypes
[TPS_MANI_03109]	TransformationProps on the level of DataPrototypes overwrites TransformationProps settings on the level of a ServiceInterface

Table F.4: Changed Specification Items in R17-10

F.2.3 Deleted Specification Items in R17-10

Number	Heading
[TPS_MANI_03100]	Transport layer independent TransportLayerIndependentInstanceIds

Table F.5: Deleted Specification Items in R17-10

F.2.4 Added Constraints in R17-10

Number	Heading
[constr_1522]	Semantics of ClientServerOperation.possibleError
[constr_1524]	Standardized values of PersistencyFileProxyInterface.category
[constr_1525]	Standardized values of PersistencyFile.category
[constr_1526]	Values of PersistencyFileArray.file.category
[constr_1527]	ImplementationDataTypeElement finally referenced as the target element in the context of an ImplementationDataTypeElementInAutosarDataPrototypeRef
[constr_1528]	Definition of optionality for multiple DataPrototypes typed by the same AutosarDataType
[constr_1529]	Standardized values of CryptoNeed.category
[constr_1530]	Standardized values of CryptoPrimitive.algorithmFamily and CryptoKeySlot.algorithmFamily
[constr_1531]	Standardized values of CryptoPrimitive.algorithmMode
[constr_1532]	Consistent assignment of TLV data ids to data structures with optional members
[constr_1533]	Applicability of ImplementationDataTypeElementExtension
[constr_1534]	Existence of DiagnosticSoftwareClusterProps
[constr_1535]	Existence of DiagnosticSoftwareClusterProps in the context of a DiagnosticContributionSet

▽

△

Number	Heading
[constr_1536]	Definition of SoftwareCluster applies for a single Machine
[constr_1537]	Consistent assignment of TLV data ids to arguments of a given ClientServer-Operation
[constr_1542]	No nested definition of SoftwareCluster
[constr_1543]	Only one physical address per SoftwareCluster
[constr_3366]	System category for a system description with Adaptive Platform components
[constr_3367]	FieldMapping.notifierDataElement reference
[constr_3368]	FieldMapping.getterOperation reference
[constr_3369]	FieldMapping.setterOperation reference
[constr_3370]	InterfaceMapping shall map all elements of a single ServiceInterface
[constr_3371]	Mutually exclusive existence of FireAndForgetMapping.dataElement reference and FireAndForgetMapping.trigger reference
[constr_3372]	Restriction in usage of ApSomeipTransformationProps.sizeOfString-LengthField
[constr_3374]	method with attribute fireAndForget set to true shall not have any inout or out arguments
[constr_3375]	method with attribute fireAndForget set to true shall not reference an ApplicationError
[constr_3376]	FireAndForgetMapping shall reference only fire and forget methods
[constr_3377]	Restriction of ISignalTriggering references in SignalBasedFieldTo-ISignalTriggeringMapping
[constr_3380]	End2EndEventProtectionProps shall not reference an event and a notifier at the same time
[constr_3387]	Compatibility of PortPrototypes of different ServiceInterfaces
[constr_3388]	Compatibility of events
[constr_3389]	Compatibility of methods
[constr_3390]	Compatibility of fields
[constr_3391]	ServiceInterfaceElementSecureComConfig references to Service-InterfaceDeployment elements
[constr_3392]	ServiceInterfaceElementSecureComConfig.dataId and Service-InterfaceElementSecureComConfig.freshnessValueId are mandatory in case of SecOC communication
[constr_3393]	Usage of shallRunOn and shallNotRunOn references
[constr_3394]	Default value for start-up timeout on the Machine is not configurable
[constr_3395]	TransformationPropsToServiceInterfaceElementMapping is restricted to one single ServiceInterface
[constr_3396]	Number of Process.modeDependentStartupConfig that refer to the same functionGroupMode
[constr_3397]	ModeDependentStartupConfig that refers to a functionGroupMode and to a machineMode
[constr_3398]	ModeDependentStartupConfig that refers to function group modes of different function groups

▽

△

Number	Heading
[constr_3527]	LogicalExpression referenced by one PhmRule

Table F.6: Added Constraints in R17-10

F.2.5 Changed Constraints in R17-10

Number	Heading
[constr_1486]	ImplementationDataType of category STRING and SwBaseType
[constr_1490]	Allowed value of category for reference ProcessToMachineMapping.process.executable
[constr_3290]	Transport Protocol attributes defined for a ProvidedSomeipServiceInstance
[constr_3296]	Transport Protocol attributes defined for a RequiredSomeipServiceInstance
[constr_3307]	SomeipEventDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3308]	SomeipEventDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3309]	SomeipMethodDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3310]	SomeipMethodDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3361]	Selective definition of serialization settings

Table F.7: Changed Constraints in R17-10

F.2.6 Deleted Constraints in R17-10

Number	Heading
[constr_3291]	SomeipServiceInstanceToMachineMapping.portConfig aggregation restriction
[constr_3358]	Usage of PortPrototype and TransportLayerIndependentInstanceId to define the same Service Instance is not allowed
[constr_3360]	RPortPrototypeProps are related only to TransportLayerIndependentInstanceIds representing a consumer Service Instance

Table F.8: Deleted Constraints in R17-10

F.3 Traceable item history of this document according to AUTOSAR Release R18-03

F.3.1 Added Specification Items in R18-03

Number	Heading
[TPS_MANI_01135]	Semantics of PersistencyKeyValueDatabaseInterface.dataTypeForSerialization
[TPS_MANI_01136]	AutosarDataPrototype is the target of the CompositionDataPrototypeRef
[TPS_MANI_01137]	Applicable use cases for CompositionDataPrototypeRef
[TPS_MANI_01138]	Semantics of PersistencyKeyValueDatabaseInterface.dataElement
[TPS_MANI_01139]	Semantics of PersistencyKeyValueDatabaseInterface.updateStrategy
[TPS_MANI_01140]	Semantics of PersistencyDataElement.updateStrategy
[TPS_MANI_01141]	Semantics of PersistencyFileProxyInterface.updateStrategy
[TPS_MANI_01142]	Semantics of PersistencyFileProxy
[TPS_MANI_01143]	Semantics of PersistencyFileProxy.updateStrategy
[TPS_MANI_01144]	Semantics of PersistencyKeyValuePair
[TPS_MANI_01146]	Initial value for PersistencyKeyValuePair
[TPS_MANI_01147]	Semantics of PersistencyKeyValueDatabase.updateStrategy
[TPS_MANI_01148]	Semantics of PersistencyKeyValuePair.updateStrategy
[TPS_MANI_01149]	Semantics of PersistencyFileArray.file
[TPS_MANI_01150]	Semantics of PersistencyFileArray
[TPS_MANI_01151]	Semantics of PersistencyFileArray.updateStrategy
[TPS_MANI_01152]	Semantics of PersistencyFile.updateStrategy
[TPS_MANI_01154]	PersistencyFileArray.updateStrategy overrides PersistencyFileProxyInterface.updateStrategy
[TPS_MANI_01155]	PersistencyKeyValueDatabase.updateStrategy overrides PersistencyKeyValueDatabaseInterface.updateStrategy
[TPS_MANI_01156]	PersistencyKeyValuePair.updateStrategy overrides PersistencyKeyValueDatabase.updateStrategy
[TPS_MANI_01157]	Semantics of updateStrategy on collection level
[TPS_MANI_01158]	PersistencyFile.updateStrategy overrides PersistencyFileArray.updateStrategy
[TPS_MANI_01159]	Semantics of updateStrategy on element level
[TPS_MANI_01160]	Definition of initial value for PersistencyDataElement
[TPS_MANI_01161]	Impact of values of category on the semantics of SoftwareClusterDesign
[TPS_MANI_01162]	Semantics of SoftwareClusterDesign.dependsOn
[TPS_MANI_01163]	Impact of values of category on the semantics of SoftwareCluster



△

Number	Heading
[TPS_MANI_01164]	Semantics of SoftwareCluster.dependsOn
[TPS_MANI_01165]	Standardized value of UserDefinedServiceInterfaceDeployment.category
[TPS_MANI_01166]	Semantics of CppImplementationDataType
[TPS_MANI_01167]	AbstractImplementationDataType
[TPS_MANI_01168]	Specification of a namespace for a CppImplementationDataType
[TPS_MANI_01169]	Support for template data types
[TPS_MANI_01170]	Semantics of CppTemplateArgument.isVariadicTemplate
[TPS_MANI_01171]	Modeling of structured data types
[TPS_MANI_01172]	Description of type references in the scope of CppImplementationDataType
[TPS_MANI_01173]	Description of type references in the scope of CppImplementationDataTypeElement
[TPS_MANI_01174]	Semantics of reference in the role CppTemplateArgument.templateType
[TPS_MANI_01175]	Semantics of reference in the role CppTemplateArgument.allocator
[TPS_MANI_01176]	Standardized value for attribute CppImplementationDataType.typeEmitter
[TPS_MANI_01177]	Semantics of CppImplementationDataType.typeEmitter
[TPS_MANI_01178]	Semantics of RestHttpPortPrototypeMapping.acceptsEncoding
[TPS_MANI_01179]	Semantics of PersistencyFileProxy.contentUri/PersistencyFile.contentUri vs. PersistencyFileArray.uri and PersistencyFileProxy.fileName/PersistencyFile.fileName
[TPS_MANI_01180]	Collection of data types that requires serialization support
[TPS_MANI_01181]	Use cases for the application of DiagnosticServiceSwMapping
[TPS_MANI_01182]	PersistencyKeyValuePair.updateStrategy overrides PersistencyDataElement.updateStrategy
[TPS_MANI_01183]	PersistencyFile.updateStrategy overrides PersistencyFileProxy.updateStrategy
[TPS_MANI_03154]	ProvidedSomeipServiceInstance related configuration settings for events
[TPS_MANI_03155]	ProvidedSomeipServiceInstance related configuration settings for methods
[TPS_MANI_03156]	RequiredSomeipServiceInstance related configuration settings for methods
[TPS_MANI_03157]	Enabling of data accumulation for upd data transmission
[TPS_MANI_03158]	Configuration of a data accumulation on a ProvidedServiceInstance for transmission over udp
[TPS_MANI_03159]	Configuration of a data accumulation on a RequiredSomeipServiceInstance for transmission over udp
[TPS_MANI_03160]	Log and Trace configuration options in the Application Manifest
[TPS_MANI_03161]	Log and Trace configuration options in the Service Instance Manifest
[TPS_MANI_03162]	Machine-specific configuration settings for the Log and Trace functional cluster

▽

△

Number	Heading
[TPS_MANI_03163]	Network configuration for Log and Trace messages
[TPS_MANI_03164]	Machine-specific configuration settings for DoIP
[TPS_MANI_03165]	Network configuration for DoIP
[TPS_MANI_03166]	Machine-specific configuration settings for NM module
[TPS_MANI_03167]	Network configuration for Nm
[TPS_MANI_03168]	Configuration of the SOME/IP load balancing option
[TPS_MANI_03169]	CppImplementationDataType with fixed size array semantics
[TPS_MANI_03170]	CppImplementationDataType of category ARRAY
[TPS_MANI_03171]	Value type of a CppImplementationDataType of category ARRAY
[TPS_MANI_03172]	Size of a CppImplementationDataType of category ARRAY
[TPS_MANI_03173]	multidimensional Array
[TPS_MANI_03174]	CppImplementationDataType with variable size array semantics
[TPS_MANI_03175]	CppImplementationDataType of category VECTOR
[TPS_MANI_03176]	Value type of a CppImplementationDataType of category VECTOR
[TPS_MANI_03177]	multidimensional Vector
[TPS_MANI_03178]	CppImplementationDataType of category STRING
[TPS_MANI_03179]	C++ language binding of CppImplementationDataTypes of category STRING
[TPS_MANI_03180]	Definition of Structures
[TPS_MANI_03181]	Definition of members in CppImplementationDataType of category STRUCTURE
[TPS_MANI_03182]	Definition of members in CppImplementationDataTypeElement of category STRUCTURE
[TPS_MANI_03183]	CppImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_03184]	CppImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_03185]	Structure of an CppImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_03186]	Usage of arraySize in case of a Vector
[TPS_MANI_03187]	Definition of enumeration types
[TPS_MANI_03188]	Usage of an Allocator for a CppImplementationDataType of category STRING
[TPS_MANI_03189]	Definition of CppImplementationDataType of category VARIANT
[TPS_MANI_03190]	CppImplementationDataType of category VARIANT
[TPS_MANI_03191]	Definition of type alternatives stored in a VARIANT
[TPS_MANI_03192]	CppImplementationDataType of category VALUE
[TPS_MANI_03193]	CppImplementationDataType or CppImplementationDataTypeElement of category TYPE_REFERENCE
[TPS_MANI_03194]	Function Group State
[TPS_MANI_03195]	Off state in Function Group
[TPS_MANI_03196]	Semantics of CppImplementationDataTypeElementQualifier.anonymous attribute

▽

△

Number	Heading
[TPS_MANI_03525]	DDS ServiceInterface binding
[TPS_MANI_03526]	DDS `VariableDataPrototype` binding
[TPS_MANI_03527]	Definition of `ProvidedDdsServiceInstance`
[TPS_MANI_03528]	Definition of `ProvidedDdsEventQosProps`
[TPS_MANI_03529]	Definition of `RequiredDdsServiceInstance`
[TPS_MANI_03530]	Definition of `RequiredDdsEventQosProps`
[TPS_MANI_03531]	`qosProfile` of `ProvidedDdsEventQosProps` is optional
[TPS_MANI_03532]	`qosProfile` of `RequiredDdsEventQosProps` is optional
[TPS_MANI_03533]	`DdsServiceInstanceToMachineMapping`
[TPS_MANI_03534]	Definition of Platform Health Management Health Channel
[TPS_MANI_03535]	Definition of Time Synchronization interaction
[TPS_MANI_03536]	Time Synchronization interaction in a master role
[TPS_MANI_03537]	Time Synchronization interaction in a slave role
[TPS_MANI_03538]	Time Synchronization interaction with a local Time Base
[TPS_MANI_03539]	Definition of Time Bases
[TPS_MANI_03540]	Definition of `PureLocalTimeBase`
[TPS_MANI_03541]	Definition of `SynchronizedSlaveTimeBase`
[TPS_MANI_03542]	Definition of `SynchronizedMasterTimeBase`
[TPS_MANI_03543]	Definition of time sync correction attributes
[TPS_MANI_03544]	Definition of `PlatformHealthManagementContribution`
[TPS_MANI_03545]	Existence of `HealthChannelExternalStatus`
[TPS_MANI_03546]	Definition of reported health status `RPortPrototype`
[TPS_MANI_03547]	Definition of *offset* time domains
[TPS_MANI_03548]	Definition of `TimeSyncPortPrototypeToTimeBaseMapping`
[TPS_MANI_03549]	Usage of `RPortPrototype` for the interaction with Time Synchronization
[TPS_MANI_03550]	Usage of `RPortPrototype` for the interaction with Platform Health Management
[TPS_MANI_03551]	Definition of Time Base kind
[TPS_MANI_03552]	Supervision cycle for `GlobalSupervision`

Table F.9: Added Specification Items in R18-03

F.3.2 Changed Specification Items in R18-03

Number	Heading
[TPS_MANI_01006]	Ordered definition of `ServiceInterface.namespace`
[TPS_MANI_01008]	Semantics of `ExecutableGroup`
[TPS_MANI_01009]	Standardized values of `ExecutableGroup.category`

▽

△

Number	Heading
[TPS_MANI_01013]	Semantics of meta-class ModeDependentStartupConfig
[TPS_MANI_01017]	Relation of startup configuration to resource group
[TPS_MANI_01041]	Startup configuration supports the definition of a launch sequence dependency
[TPS_MANI_01042]	Definition of a linear ImplementationDataType of category VECTOR
[TPS_MANI_01044]	Structure of an ImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_01060]	Use cases for the application of DiagnosticServiceDataMapping
[TPS_MANI_01068]	Semantics of PersistencyFileProxyInterface.maxNumberOfFiles
[TPS_MANI_01069]	Further qualification of properties of PortPrototypes typed by PersistencyKeyValueDatabaseInterfaces
[TPS_MANI_01075]	Specification of redundancy of persistent data
[TPS_MANI_01078]	Semantics of PersistencyPortPrototypeToKeyValueDatabaseMapping
[TPS_MANI_01080]	Semantics of PersistencyPortPrototypeToFileArrayMapping
[TPS_MANI_01097]	Assignment of TLV data ids for data structures with optional members
[TPS_MANI_01100]	Semantics of Allocator
[TPS_MANI_01109]	Semantics of UploadablePackageElement
[TPS_MANI_01112]	Semantics of SoftwareClusterDesign
[TPS_MANI_01113]	Semantics of SoftwareClusterDesign.diagnosticAddress
[TPS_MANI_01116]	Reference to model elements included in an uploadable software package
[TPS_MANI_01117]	Semantics of SoftwareClusterDesign.intendedTargetMachine
[TPS_MANI_01118]	Relation between SoftwareClusterDesign and DiagnosticContributionSet
[TPS_MANI_01119]	Reference to model elements from SoftwareClusterDesign
[TPS_MANI_01133]	Optional element of an event
[TPS_MANI_01134]	Optional element in the context of a method
[TPS_MANI_03001]	Mapping of AdaptivePlatformServiceInstance to a MachineDesign
[TPS_MANI_03002]	IP configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03003]	ProvidedSomeipServiceInstance Fanout
[TPS_MANI_03022]	Context of RequiredSomeipServiceInstance
[TPS_MANI_03110]	Allowed components in system description with category SYSTEM_DESIGN_DESCRIPTION.
[TPS_MANI_03114]	Usage of AssemblySwConnectors in the System Design model
[TPS_MANI_03145]	Description of a function group
[TPS_MANI_03152]	Assignment of a ModeDependentStartupConfig to a function group state
[TPS_MANI_03153]	Semantics of ModeDependentStartupConfig.functionGroupMode
[TPS_MANI_03500]	Definition of Platform Health Management Supervision and Checkpoints
[TPS_MANI_03503]	Applicability of supervision to a specific Process
[TPS_MANI_03505]	Existence of SupervisionCheckpoint

▽

△

Number	Heading
[TPS_MANI_03506]	Optionality of SupervisionCheckpoint
[TPS_MANI_03508]	Definition of an AliveSupervision for a SupervisionCheckpoint
[TPS_MANI_03509]	Definition of a CheckpointTransition
[TPS_MANI_03510]	Definition of LogicalSupervision
[TPS_MANI_03512]	Applicability of global supervision to a specific Process
[TPS_MANI_03513]	Collection of LocalSupervisions into a global supervision
[TPS_MANI_03514]	Expiration tolerance for GlobalSupervision
[TPS_MANI_03515]	Expiration tolerance for LocalSupervision
[TPS_MANI_03516]	Condition evaluation for HealthChannelSupervision
[TPS_MANI_03517]	Condition evaluation for HealthChannelExternalStatus

Table F.10: Changed Specification Items in R18-03

F.3.3 Deleted Specification Items in R18-03

Number	Heading
[TPS_MANI_01031]	Semantics of CompositionDataPrototypeRef
[TPS_MANI_01045]	Process.modeDependentStartupConfig that does not refer to a ModeDeclaration
[TPS_MANI_01132]	Semantics of CompositionDataPrototypeRef
[TPS_MANI_03019]	TTL for SubscribeEventGroupAck Entries
[TPS_MANI_03501]	Definition of platform health management supervised entities
[TPS_MANI_03504]	Existence of SupervisionEntity

Table F.11: Deleted Specification Items in R18-03

F.3.4 Added Constraints in R18-03

Number	Heading
[constr_1546]	Existence of attributes of ServiceInterfaceSubElement
[constr_1547]	Reference from ImplementationDataTypeExtension to ImplementationDataType
[constr_1548]	Reference from ImplementationDataTypeElementExtension to ImplementationDataTypeElement
[constr_1549]	Value of ProcessorCore.coreId
[constr_1550]	Reference from Process to ProcessDesign
[constr_1551]	Existence of CompositionDataPrototypeRef.dataPrototype vs. CompositionDataPrototypeRef.elementInImplDatatype

▽

△

Number	Heading
[constr_1553]	Restriction for ProcessToMachineMapping
[constr_1554]	Restriction regarding PersistencyKeyValuePair.initValue
[constr_1555]	Restriction applicable for PersistencyPortPrototypeToKeyValue-DatabaseMapping.portPrototype
[constr_1556]	Restriction applicable for PersistencyPortPrototypeToFileArrayMapping.portPrototype
[constr_1557]	Standardized values of SoftwareClusterDesign.category and Software-Cluster.category
[constr_1558]	Existence of SoftwareClusterDesign.diagnosticAddress
[constr_1559]	Existence of SoftwareClusterDesign.subSoftwareCluster
[constr_1560]	Usage of SoftwareClusterDesign.requiredARElement
[constr_1561]	Existence of SoftwareClusterDesign.subSoftwareCluster and Software-ClusterDesign.dependsOn.dependentSoftwareClusterDesign
[constr_1562]	Existence of SoftwareClusterDesign.diagnosticContribution
[constr_1563]	Standardized values of SoftwareClusterDesign.category and Software-Cluster.category
[constr_1564]	Existence of SoftwareCluster.diagnosticAddress
[constr_1565]	Existence of SoftwareCluster.subSoftwareCluster
[constr_1566]	Usage of SoftwareCluster.containedARElement
[constr_1567]	Existence of SoftwareCluster.subSoftwareCluster and Software-Cluster.dependsOn.dependentSoftwareCluster
[constr_1568]	Existence of SoftwareCluster.diagnosticExtract
[constr_1569]	Restriction for the scope of RestHttpPortPrototypeMapping.acceptsEncoding
[constr_1570]	Restriction for UserDefinedServiceInterfaceDeployment of category SERVICE_INTERFACE_DEPLOYMENT_IPC
[constr_1571]	CppImplementationDataType is limited
[constr_1572]	Usage of SwDataDefProps.implementationDataType within a Cpp-ImplementationDataType
[constr_1573]	CppTemplateArgument.isVariadicTemplate is set to True
[constr_1574]	Number of CppTemplateArguments with isVariadicTemplate set to True
[constr_1575]	Position of CppTemplateArgument with isVariadicTemplate set to True
[constr_1576]	Existence of CppTemplateArgument.templateType vs. CppTemplate-Argument.allocator
[constr_1577]	Specification of a nativeDeclaration for a CppImplementationDataType
[constr_1578]	applicable data categories
[constr_1579]	SwDataDefProps applicable to CppImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*
[constr_1580]	Restriction for the usage of RestHttpPortPrototypeMapping.acceptsEncoding
[constr_1581]	Value of fileProxy.fileName

▽

△

Number	Heading
[constr_1582]	PersistencyKeyValuePair.valueDataType shall match to ImplementationDataType for the corresponding PersistencyDataElement
[constr_1585]	Standardized values of attribute DiagnosticServiceSwMapping.category
[constr_1586]	DiagnosticServiceSwMapping.category set to DATA_ELEMENT
[constr_1587]	DiagnosticServiceSwMapping.category set to DATA_IDENTIFIER
[constr_1588]	DiagnosticServiceSwMapping.category set to GENERIC_UDS_SERVICE
[constr_1589]	Value of file.fileName
[constr_3408]	Value range of SomeipEventDeployment.eventId
[constr_3409]	Value range of SomeipMethodDeployment.methodId
[constr_3410]	Value range of SomeipServiceInterfaceDeployment.serviceInterfaceId
[constr_3411]	eventMulticastUdpPort, ipv4MulticastIpAddress and ipv6MulticastIpAddress not relevant for RequiredSomeipServiceInstances
[constr_3412]	OsModuleInstantiation shall have at least one ResourceGroup
[constr_3413]	ModeDependentStartupConfig of a Process is mapped to exactly one ResourceGroup
[constr_3414]	Allowed usage of EthernetNetworkConfiguration attributes
[constr_3415]	Value range of loadBalancingPriority
[constr_3416]	Value range of loadBalancingWeight
[constr_3417]	UserDefinedEventDeployments aggregated by a UserDefinedField-Deployment
[constr_3418]	UserDefinedMethodDeployments aggregated by a UserDefinedField-Deployment
[constr_3419]	Allowed usage of EthernetNetworkConfiguration attributes
[constr_3420]	System category for a design description that has one single Adaptive Machine in scope
[constr_3421]	Fibex elements applicable for a MACHINE_DESIGN_EXTRACT
[constr_3422]	CppImplementationDataType of category STRING and SwBaseType
[constr_3423]	ModeDependentStartupConfig of a Process shall reference a functionGroupMode or machineMode
[constr_3424]	ModeDependentStartupConfig shall never reference the functionGroupMode Off
[constr_3425]	Restriction of DoIpInstantiations on a Machine
[constr_3426]	The logTraceFilePath is mandatory in case that logTraceLogMode is set to file
[constr_3427]	The logTraceFilePath is only relevant if logTraceLogMode is set to file
[constr_3428]	Structure shall own at least one element
[constr_3429]	No allocator usage for CppImplementationDataTypes of category VARIANT
[constr_3432]	Allowed subElements for Structures
[constr_3433]	Aggregation of templateArguments for a ARRAY
[constr_3434]	Aggregation of templateArguments for a VECTOR
[constr_3528]	Value range of domainId

▽

△

Number	Heading
[constr_3529]	Value range of `serviceInstanceId`
[constr_3530]	Mandatory definition of `checkpointId`
[constr_3531]	Mandatory definition of `healthChannelId`
[constr_3532]	Mandatory definition of `statusId`
[constr_3536]	Mandatory definition of `supervisedEntityId`

Table F.12: Added Constraints in R18-03

F.3.5 Changed Constraints in R18-03

Number	Heading
[constr_1484]	Applicability of `ModeDependentStartupConfig.executionDependency`
[constr_1507]	`PortInterfaceToDataTypeMapping` is only applicable to `ServiceInterface`
[constr_1532]	Consistent assignment of TLV data ids to data structures with optional members
[constr_1537]	Consistent assignment of TLV data ids to arguments of a given `ClientServerOperation`
[constr_3307]	`SomeipEventDeployment.transportProtocol` setting to `udp` and the impact on `ProvidedSomeipServiceInstance`s
[constr_3308]	`SomeipEventDeployment.transportProtocol` setting to `tcp` and the impact on `ProvidedSomeipServiceInstance`s
[constr_3309]	`SomeipMethodDeployment.transportProtocol` setting to `udp` and the impact on `ProvidedSomeipServiceInstance`s
[constr_3310]	`SomeipMethodDeployment.transportProtocol` setting to `tcp` and the impact on `ProvidedSomeipServiceInstance`s
[constr_3320]	Aggregation of `CommunicationConnector` by `MachineDesign`
[constr_3350]	Consistent value of `category` for `ExecutableGroup`s referencing an `Executable`
[constr_3366]	System `category` for a system design description with Adaptive Platform and Classic Platform content

Table F.13: Changed Constraints in R18-03

F.3.6 Deleted Constraints in R18-03

Number	Heading
[constr_1480]	Mutual existence of `CompositionDataPrototypeRef.elementInImplDatatype` vs. attributes of `CompositionDataPrototypeRef.dataPrototype`
[constr_1505]	Number of `Process.modeDependentStartupConfig` that do not refer to a `ModeDeclaration`

▽

△

Number	Heading
[constr_1525]	Standardized values of PersistencyFile.category
[constr_1526]	Values of PersistencyFileArray.file.category
[constr_1533]	Applicability of ImplementationDataTypeElementExtension

Table F.14: Deleted Constraints in R18-03

F.4 Traceable item history of this document according to AUTOSAR Release R18-10

F.4.1 Added Specification Items in R18-10

Number	Heading
[TPS_MANI_01184]	Definition of optional elements on the level of ApplicationDataType
[TPS_MANI_01185]	Definition of optional elements on the level of CppImplementationDataType
[TPS_MANI_01186]	Definition of the applicable wire type
[TPS_MANI_01187]	Matching pairs of PersistencyFileProxy and PersistencyFile
[TPS_MANI_01188]	Semantics of attribute schedulingPriority
[TPS_MANI_01189]	Software Cluster and DiagnosticContributionSet.category
[TPS_MANI_01190]	Semantics of ApApplicationError
[TPS_MANI_01191]	Modeling of possible errors
[TPS_MANI_01192]	Semantics of ApApplicationErrorDomain
[TPS_MANI_01193]	Combination of ModeDependentStartupConfig.machineMode and ModeDependentStartupConfig.functionGroupMode
[TPS_MANI_01194]	Semantics of PersistencyKeyValueDatabaseInterface.minimumSustainedSize
[TPS_MANI_01195]	Semantics of PersistencyFileProxyInterface.minimumSustainedSize
[TPS_MANI_01196]	Semantics of PersistencyKeyValueDatabase.minimumSustainedSize
[TPS_MANI_01197]	Semantics of PersistencyKeyValueDatabase.maximumAllowedSize
[TPS_MANI_01198]	Semantics of ApApplicationErrorSet
[TPS_MANI_01199]	Semantics of DeterministicClientResourceNeeds
[TPS_MANI_01200]	Semantics of meta-class DeterministicClientResource
[TPS_MANI_01201]	Standardized values for attribute CppTemplateArgument.category
[TPS_MANI_01202]	Semantics of reference SoftwareCluster.moduleInstantiation
[TPS_MANI_01203]	Semantics of DeterministicClient
[TPS_MANI_01204]	Specification of redundancy of persistent data
[TPS_MANI_01205]	Semantics of meta-class PersistencyDeployment

▽

△

Number	Heading
[TPS_MANI_01206]	Modeling of redundancy in the context of PersistencyDeployment
[TPS_MANI_01207]	Standardized values of attribute PersistencyRedundancyCrc.algorithmFamily
[TPS_MANI_01208]	Definition of envionment variables in the scope of a Machine
[TPS_MANI_01209]	Definition of envionment variables in process scope
[TPS_MANI_01210]	Default encoding for all DataPrototypes typed by CppImplementationDataType of category STRING
[TPS_MANI_03197]	Semantics of StdCppImplementationDataType
[TPS_MANI_03198]	Semantics of CustomCppImplementationDataType
[TPS_MANI_03199]	Endpoint protection by SecureComProps
[TPS_MANI_03200]	SecureComProps for udp, tcp and multicast communication
[TPS_MANI_03201]	Semantics of CppTemplateArgument.inplace attribute
[TPS_MANI_03202]	Definition of bitfield types
[TPS_MANI_03203]	Configuration of IPsec
[TPS_MANI_03204]	Definition of IPSecRules
[TPS_MANI_03205]	IPsec connection type
[TPS_MANI_03206]	IPsec AH and ESP protocol configuration
[TPS_MANI_03207]	IPsec Internet Key Exchange protocol configuration
[TPS_MANI_03208]	Protection of AdaptivePlatformServiceInstance by IPsec
[TPS_MANI_03209]	The meaning of MachineDesign.accessControl
[TPS_MANI_03210]	Specification of event specific communication attributes
[TPS_MANI_03211]	Specification of field specific communication attributes
[TPS_MANI_03212]	Specification of initial value for a field
[TPS_MANI_03213]	Semantics of meta-class TlsSecureComProps
[TPS_MANI_03214]	Existence of TlsCryptoCipherSuite.keyExchange vs. TlsSecureComProps.keyExchange
[TPS_MANI_03215]	Semantics of CryptoServiceCertificate
[TPS_MANI_03216]	Existence of TlsCryptoCipherSuite.certificate in the *client* role
[TPS_MANI_03217]	On-the-wire encoding for a chosen string
[TPS_MANI_03218]	Default value for the attribute tcpInitialInactivityTime of meta-class DoIpNetworkConfiguration
[TPS_MANI_03219]	Default value for the attribute tcpGeneralInactivityTime of meta-class DoIpNetworkConfiguration
[TPS_MANI_03220]	Default value for the attribute vehicleAnnouncementCount of meta-class DoIpNetworkConfiguration
[TPS_MANI_03221]	Default value for the attribute vehicleAnnouncementInterval of meta-class DoIpNetworkConfiguration
[TPS_MANI_03222]	Default value for the attribute tcpAliveCheckResponseTimeout of meta-class DoIpNetworkConfiguration
[TPS_MANI_03553]	Applicability of health channel to a specific Process

▽

△

Number	Heading
[TPS_MANI_03554]	Several SomeipServiceInstanceToMachineMappings with equal settings
[TPS_MANI_03555]	Mix of SomeipServiceInstanceToMachineMapping and signal-based communication
[TPS_MANI_03556]	DDS-RPC Service Binding
[TPS_MANI_03557]	DDS ClientServerOperation Binding
[TPS_MANI_03558]	DDS Field Binding
[TPS_MANI_03559]	Definition of DdsProvidedServiceInstance.methodQosProps
[TPS_MANI_03560]	qosProfile of DdsProvidedServiceInstance.methodQosProps is optional
[TPS_MANI_03561]	Definition of DdsProvidedServiceInstance.fieldNotifierQosProps
[TPS_MANI_03562]	qosProfile of DdsProvidedServiceInstance.fieldNotifierQosProps is optional
[TPS_MANI_03563]	Definition of DdsProvidedServiceInstance.fieldGetSetQosProps
[TPS_MANI_03564]	qosProfile of DdsProvidedServiceInstance.fieldGetSetQosProps is optional
[TPS_MANI_03565]	Definition of DdsRequiredServiceInstance.methodQosProps
[TPS_MANI_03566]	qosProfile of DdsRequiredServiceInstance.methodQosProps is optional
[TPS_MANI_03567]	Definition of DdsRequiredServiceInstance.fieldNotifierQosProps
[TPS_MANI_03568]	qosProfile of DdsRequiredServiceInstance.fieldNotifierQosProps is optional
[TPS_MANI_03569]	Definition of DdsRequiredServiceInstance.fieldGetSetQosProps
[TPS_MANI_03570]	qosProfile of DdsRequiredServiceInstance.fieldGetSetQosProps is optional
[TPS_MANI_03571]	transportPlugin for DdsProvidedServiceInstance
[TPS_MANI_03572]	transportPlugin for DdsRequiredServiceInstance

Table F.15: Added Specification Items in R18-10

F.4.2 Changed Specification Items in R18-10

Number	Heading
[TPS_MANI_01001]	Meaning of ServiceInterface
[TPS_MANI_01041]	Startup configuration supports the definition of a launch sequence dependency
[TPS_MANI_01097]	Assignment of TLV data ids
[TPS_MANI_01100]	Semantics of Allocator

▽

△

Number	Heading
[TPS_MANI_01147]	Semantics of PersistencyKeyValueDatabase.updateStrategy
[TPS_MANI_01151]	Semantics of PersistencyFileArray.updateStrategy
[TPS_MANI_01166]	Semantics of CppImplementationDataType
[TPS_MANI_01176]	Standardized value for attribute CppImplementationDataType.typeEmitter
[TPS_MANI_01177]	Semantics of CppImplementationDataType.typeEmitter
[TPS_MANI_01180]	Collection of data types that requires serialization support
[TPS_MANI_03001]	Mapping of AdaptivePlatformServiceInstance to a MachineDesign
[TPS_MANI_03011]	Server Timing configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03021]	Requirements on the searched minor version from the client's point of view
[TPS_MANI_03025]	Client Timing configuration for a RequiredSomeipServiceInstance
[TPS_MANI_03070]	Size of a length field for a chosen array or map
[TPS_MANI_03103]	Default size for all array and map length fields
[TPS_MANI_03124]	ServiceInterface.event to ISignalTriggering mapping
[TPS_MANI_03125]	ServiceInterface.method to ISignalTriggerings mapping
[TPS_MANI_03126]	ServiceInterface.field mapping to ISignalTriggerings
[TPS_MANI_03134]	Configuration of supported TLS ciphersuites
[TPS_MANI_03137]	ServiceInterfaceElementSecureComConfig is not relevant in case of TLS communication
[TPS_MANI_03157]	Enabling of data collection for upd data transmission
[TPS_MANI_03158]	Configuration of a data collection on a ProvidedServiceInstance for transmission over udp
[TPS_MANI_03165]	Network Interface configuration for DoIP
[TPS_MANI_03170]	CppImplementationDataType of category ARRAY
[TPS_MANI_03173]	Definition of a multidimensional Array
[TPS_MANI_03175]	CppImplementationDataType of category VECTOR
[TPS_MANI_03177]	Definition of a multidimensional Vector
[TPS_MANI_03178]	StdCppImplementationDataType of category STRING
[TPS_MANI_03179]	C++ language binding of StdCppImplementationDataTypes of category STRING
[TPS_MANI_03180]	Definition of Structures
[TPS_MANI_03181]	Definition of members in StdCppImplementationDataType of category STRUCTURE
[TPS_MANI_03184]	CppImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_03185]	Structure of an CppImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_03187]	Definition of enumeration types
[TPS_MANI_03193]	CppImplementationDataType of category TYPE_REFERENCE
[TPS_MANI_03196]	Semantics of CppImplementationDataTypeElementQualifier.inplace attribute

▽

△

Number	Heading
[TPS_MANI_03503]	Applicability of checkpoints to a specific `Process`
[TPS_MANI_03512]	Applicability of global supervision without `Process` context
[TPS_MANI_03516]	Condition evaluation for `HealthChannelSupervision`
[TPS_MANI_03518]	`PhmLogicalExpression` definition
[TPS_MANI_03519]	`PhmRule` definition
[TPS_MANI_03520]	Execution of `PhmActionList` with `actionListExecution=triggeredOnEvaluation`
[TPS_MANI_03521]	Execution of `PhmActionList` with `actionListExecution=triggeredOnChange`
[TPS_MANI_03522]	Definition of actions for application software
[TPS_MANI_03523]	Definition of actions for Platform Instance
[TPS_MANI_03524]	Definition of actions for Watchdog
[TPS_MANI_03526]	DDS `VariableDataPrototype` binding
[TPS_MANI_03527]	Definition of `DdsProvidedServiceInstance`
[TPS_MANI_03528]	Definition of `DdsProvidedServiceInstance.eventQosProps`
[TPS_MANI_03529]	Definition of `DdsRequiredServiceInstance`
[TPS_MANI_03530]	Definition of `DdsRequiredServiceInstance.eventQosProps`
[TPS_MANI_03531]	`qosProfile` of `DdsProvidedServiceInstance.eventQosProps` is optional
[TPS_MANI_03532]	`qosProfile` of `DdsRequiredServiceInstance.eventQosProps` is optional
[TPS_MANI_03533]	`DdsServiceInstanceToMachineMapping`
[TPS_MANI_03552]	Supervision cycle for `GlobalSupervision`

Table F.16: Changed Specification Items in R18-10

F.4.3 Deleted Specification Items in R18-10

Number	Heading
[TPS_MANI_01008]	Semantics of `ExecutableGroup`
[TPS_MANI_01009]	Standardized values of `ExecutableGroup.category`
[TPS_MANI_01018]	`ImplementationDataType` of `category` `VECTOR`
[TPS_MANI_01028]	`ImplementationDataType` of `category` `ASSOCIATIVE_MAP`
[TPS_MANI_01029]	Usage of `ImplementationDataType`
[TPS_MANI_01030]	`ImplementationDataType` of `category` `STRING`
[TPS_MANI_01042]	Definition of a linear `ImplementationDataType` of `category` `VECTOR`
[TPS_MANI_01043]	Definition of a rectangular `ImplementationDataType` of `category` `VECTOR`
[TPS_MANI_01044]	Structure of an `ImplementationDataType` of `category` `ASSOCIATIVE_MAP`

▽

△

Number	Heading
[TPS_MANI_01055]	Definition of application-level errors
[TPS_MANI_01056]	Semantics of ApplicationError.errorContext
[TPS_MANI_01058]	Ability to create a mapping of ApplicationErrors aggregated in the role possibleError
[TPS_MANI_01062]	ImplementationDataType to generate a C++ enum
[TPS_MANI_01063]	Sharing of ImplementationDataType with enumeration semantics
[TPS_MANI_01074]	Specification of encryption of persistent data
[TPS_MANI_01075]	Specification of redundancy of persistent data
[TPS_MANI_01077]	Specification of file encryption
[TPS_MANI_01082]	Eligibility of DataPrototypes for the definition of optionality
[TPS_MANI_01083]	Optionality is supported for ApplicationDataType as well as ImplementationDataType
[TPS_MANI_01084]	Optionality for a DataPrototype typed by an ApplicationDataType
[TPS_MANI_01085]	Definition of optionality for a DataPrototype typed by an ImplementationDataType
[TPS_MANI_01087]	Interaction with crypto software
[TPS_MANI_01088]	Semantics of CryptoNeed
[TPS_MANI_01089]	Relation between CryptoNeed and PortPrototype
[TPS_MANI_01090]	Modeling of crypto software as a platform module
[TPS_MANI_01091]	Semantics of CryptoJob
[TPS_MANI_01092]	Mapping between CryptoNeed and CryptoJob
[TPS_MANI_01093]	Semantics of CryptoDriver
[TPS_MANI_01094]	Scope of CryptoDriver
[TPS_MANI_01095]	Semantics of CryptoKeySlot
[TPS_MANI_01096]	Semantics of the CryptoPrimitive
[TPS_MANI_01098]	Constraints on the definition of an ImplementationDataType of category VECTOR
[TPS_MANI_01099]	Semantics of ImplementationDataTypeElementExtension
[TPS_MANI_01101]	Size-constrained allocation of memory
[TPS_MANI_01102]	Specification of a namespace for an ImplementationDataType of category VECTOR
[TPS_MANI_01133]	Optional element of an event
[TPS_MANI_01134]	Optional element in the context of a method
[TPS_MANI_03121]	Signal-based VariableDataPrototype binding
[TPS_MANI_03122]	Signal-based Field binding
[TPS_MANI_03123]	Signal-based ClientServerOperation binding
[TPS_MANI_03135]	Configuration of TLS PSK Identity
[TPS_MANI_03136]	Configuration of requirements for the TLS cryptographic job
[TPS_MANI_03141]	Mapping between SecOcJobRequirement and CryptoJob

▽

△

Number	Heading
[TPS_MANI_03142]	Mapping between TlsJobRequirement and CryptoJob
[TPS_MANI_03143]	Mapping between PresharedKeyIdentity and CryptoKeySlot
[TPS_MANI_03144]	C++ language binding of ImplementationDataTypes of category STRING
[TPS_MANI_03182]	Definition of members in CppImplementationDataTypeElement of category STRUCTURE

Table F.17: Deleted Specification Items in R18-10

F.4.4 Added Constraints in R18-10

Number	Heading
[constr_1593]	Completeness of the existence of a set of TlvDataIdDefinition.tlvArguments
[constr_1594]	Consistent assignment of TLV data ids to ApplicationRecordDataType
[constr_1595]	Consistent assignment of TLV data ids to CppImplementationDataType or CppImplementationDataTypeElement
[constr_1596]	Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to ArgumentDataPrototype
[constr_1597]	Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to ApplicationRecordElement
[constr_1598]	Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to CppImplementationDataTypeElement
[constr_1599]	TlvDataIdDefinition referencing ArgumentDataPrototype
[constr_1600]	TlvDataIdDefinition referencing ApplicationRecordElement
[constr_1601]	TlvDataIdDefinition referencing CppImplementationDataTypeElement
[constr_1603]	Completeness of the existence of a set of TlvDataIdDefinition.tlvRecordElements
[constr_1604]	Completeness of the existence of a set of TlvDataIdDefinition.tlvSubElements
[constr_1605]	Standardized values of attribute Executable.category
[constr_1606]	Processes with mutual ExecutionDependencys
[constr_1613]	File name of matching pairs of PersistencyFileProxy and PersistencyFile
[constr_1614]	Existence of attribute TransformationPropsToServiceInterfaceElementMapping.transformationProps.sessionHandling
[constr_1615]	Existence of attribute SomeipDataPrototypeTransformationProps.someipTransformationProps.sessionHandling
[constr_1618]	Ability to shut down
[constr_1619]	Ability to restart
[constr_1620]	Value of schedulingPriority if schedulingPolicy is set to schedulingPolicyFifo or schedulingPolicyRoundRobin

▽

△

Number	Heading
[constr_1621]	Value of schedulingPriority if schedulingPolicy is set to schedulingPolicyOther
[constr_1625]	Existence of reference ApApplicationError.errorDomain
[constr_1627]	Supported value range for attribute ApApplicationErrorDomain.value
[constr_1628]	Definition of static length field sizes in case of TLV usage
[constr_1629]	Identical sizes of length fields in case of TLV usage
[constr_1630]	No definition of length field sizes on DataPrototype level in case of TLV usage
[constr_1658]	Number of DiagnosticTroubleCodeUdsToClearConditionGroupMapping elements per DiagnosticTroubleCodeUds
[constr_1659]	Restriction for the usage of CppImplementationDataTypeElementQualifier.inplace
[constr_1660]	Restriction for the usage of CppTemplateArgument.inplace
[constr_1661]	Multiplicity of OsModuleInstantiation.resourceGroup
[constr_1663]	Standardized values of attribute DiagnosticServiceDataIdentifierPortMapping.category
[constr_1664]	Unique ApApplicationError.shortName
[constr_1665]	Unique ApApplicationError.errorCode
[constr_1666]	References from PersistencyPortPrototypeToKeyValueDatabaseMapping to PersistencyKeyValueDatabase
[constr_1667]	References from PersistencyPortPrototypeToFileArrayMapping to PersistencyFileArray
[constr_1668]	Allowed combinations of PersistencyRedundancyCrc.length and algorithmFamily
[constr_1673]	Existence of attributes hasGetter, hasSetter, and hasNotifier
[constr_1674]	Suported encoding of StdCppImplementationDataType of category STRING
[constr_1675]	Existence of attribute ApSomeipTransformationProps.stringEncoding
[constr_1676]	Consistency of references shallRunOn and shallNotRunOn
[constr_1677]	Mutual exclusive existence of references shallRunOn and shallNotRunOn
[constr_1678]	Allowed values for attribute ApSomeipTransformationProps.stringEncoding
[constr_3443]	Specification of a namespace for a StdCppImplementationDataType
[constr_3446]	CppTemplateArgument with allocator reference and the inplace flag
[constr_3447]	ApSomeipTransformationProps.sizeOfArrayLengthField that equals 0
[constr_3462]	CppTemplateArgument.templateType reference to StdCppImplementationDataType of category STRUCTURE and the inplace flag
[constr_3485]	UDP endpoint using DTLS can only serve provided or required service instances exclusively
[constr_3486]	TCP endpoint using TLS can only serve provided or required service instances exclusively.
[constr_3487]	TCP endpoint can only serve provided or required service instances exclusively
[constr_3492]	DoIpInstantiation.logicalAddress shall be defined as member in the DoIpRequestConfiguration

▽

△

Number	Heading
[constr_3493]	Applicable attributes for standardized E2E Profiles
[constr_3494]	Mandatory Machine States
[constr_3495]	Supported value range for attribute DoIpInstantiation.eid
[constr_3496]	Supported value range for attribute DoIpInstantiation.gid
[constr_3497]	Supported value range for attribute DoIpInstantiation.maxRequestBytes
[constr_3498]	Supported value range for attribute DoIpInstantiation.logicalAddress
[constr_3499]	Supported value range for attribute DoIpRequestConfiguration.startAddress
[constr_3537]	LocalSupervision referenced once in the context of a GlobalSupervision
[constr_3538]	Only one ServiceInstanceToMachineMapping per technology and CommunicationConnector
[constr_3539]	Only one AliveSupervision per SupervisionCheckpoint
[constr_3540]	SupervisionCheckpoint in supervision graph
[constr_3541]	qosProfile mandatory for DdsProvidedServiceInstance
[constr_3542]	qosProfile mandatory for DdsRequiredServiceInstance
[constr_3543]	At least one transportPlugin definition required for each DdsProvidedServiceInstance
[constr_3544]	At least one transportPlugin definition required for each DdsRequiredServiceInstance
[constr_5000]	Supported value range for attribute DoIpRequestConfiguration.endAddress
[constr_5001]	Usage of DoIpNetworkConfiguration.eidUseMac
[constr_5002]	Supported values of ServiceInstanceToMachineMapping.category
[constr_5003]	Existence of TlsCryptoCipherSuite.certificate in the *server* role
[constr_5004]	Mapping of a Process to a Machine is mandatory in the Execution Manifest

Table F.18: Added Constraints in R18-10

F.4.5 Changed Constraints in R18-10

Number	Heading
[constr_1490]	Allowed value of category for reference ProcessToMachineMapping.process.executable
[constr_1551]	Existence of DataPrototypeInServiceInterfaceRef.dataPrototype vs. DataPrototypeInServiceInterfaceRef.elementInImplDatatype
[constr_1572]	Usage of SwDataDefProps.implementationDataType within a CppImplementationDataType
[constr_1573]	CppTemplateArgument.isVariadicTemplate is set to True
[constr_1582]	PersistencyKeyValuePair.valueDataType shall match to AbstractImplementationDataType for the corresponding PersistencyDataElement
[constr_1585]	Standardized values of attribute DiagnosticServiceSwMapping.category

▽

△

Number	Heading
[constr_1589]	Value of file.fileName
[constr_3375]	method with attribute fireAndForget set to true shall not reference an ApApplicationError
[constr_3392]	ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are mandatory in case of SecOC communication
[constr_3433]	Aggregation of templateArguments for an ARRAY
[constr_3434]	Aggregation of templateArguments for a VECTOR
[constr_3527]	PhmLogicalExpression referenced by one PhmRule
[constr_3528]	Value range of domainId
[constr_3529]	Value range of serviceInstanceId

Table F.19: Changed Constraints in R18-10

F.4.6 Deleted Constraints in R18-10

Number	Heading
[constr_1474]	SwDataDefProps applicable to ImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*
[constr_1475]	ImplementationDataType of category STRING is limited
[constr_1476]	ImplementationDataType of category VECTOR is limited
[constr_1477]	ImplementationDataType of category ASSOCIATIVE_MAP is limited
[constr_1479]	No support for certain values of ImplementationDataType.category
[constr_1484]	Applicability of ModeDependentStartupConfig.executionDependency
[constr_1485]	No subElement for ImplementationDataType of category STRING
[constr_1486]	ImplementationDataType of category STRING and SwBaseType
[constr_1487]	Number of subElements of an ImplementationDataType of category ASSOCIATIVE_MAP
[constr_1491]	Semantics of ServiceInterface.possibleError
[constr_1493]	ArgumentDataPrototype referenced in the role ApplicationError.errorContext
[constr_1495]	Initial value for field
[constr_1506]	ImplementationDataType of category VECTOR shall not define dynamicArraySizeProfile
[constr_1508]	BaseTypeDirectDefinition.nativeDeclaration shall not be set to the value enum
[constr_1522]	Semantics of ClientServerOperation.possibleError
[constr_1527]	ImplementationDataTypeElement finally referenced as the target element in the context of an ImplementationDataTypeElementInAutosarDataPrototypeRef

▽

△

Number	Heading
[constr_1528]	Definition of optionality for multiple DataPrototypes typed by the same AutosarDataType
[constr_1529]	Standardized values of CryptoNeed.category
[constr_1530]	Standardized values of CryptoPrimitive.algorithmFamily and CryptoKeySlot.algorithmFamily
[constr_1531]	Standardized values of CryptoPrimitive.algorithmMode
[constr_1532]	Consistent assignment of TLV data ids to data structures with optional members
[constr_1537]	Consistent assignment of TLV data ids to arguments of a given ClientServerOperation
[constr_1546]	Existence of attributes of ServiceInterfaceSubElement
[constr_1547]	Reference from ImplementationDataTypeExtension to ImplementationDataType
[constr_1548]	Reference from ImplementationDataTypeElementExtension to ImplementationDataTypeElement
[constr_1577]	Specification of a nativeDeclaration for a CppImplementationDataType
[constr_1587]	DiagnosticServiceSwMapping.category set to DATA_IDENTIFIER
[constr_1588]	DiagnosticServiceSwMapping.category set to GENERIC_UDS_SERVICE
[constr_3293]	Mandatory information of a RequiredSomeipServiceInstance
[constr_3303]	ANY not allowed for SomeipServiceInterfaceDeployment.serviceInterfaceVersion
[constr_3350]	Consistent value of category for ExecutableGroups referencing an Executable
[constr_3377]	Restriction of ISignalTriggering references in SignalBasedFieldToISignalTriggeringMapping
[constr_3422]	CppImplementationDataType of category STRING and SwBaseType
[constr_3428]	Structure shall own at least one element
[constr_3432]	Allowed subElements for Structures

Table F.20: Deleted Constraints in R18-10

F.5 Traceable item history of this document according to AUTOSAR Release R19-03

F.5.1 Added Specification Items in R19-03

Number	Heading
[TPS_MANI_01211]	Specification of executable software within SoftwareClusterDesign
[TPS_MANI_01212]	Usage of attribute typeEmitter in the context of a CustomCppImplementationDataType
[TPS_MANI_01213]	Semantics of meta-class StrongRevisionLabelString

▽

△

Number	Heading
[TPS_MANI_01214]	Semantics of SoftwareCluster.conflictsTo
[TPS_MANI_01215]	Semantics of meta-class SoftwareActivationDependencyFormula
[TPS_MANI_01216]	Semantics of meta-class SoftwareActivationDependencyFormula-Part
[TPS_MANI_01217]	Semantics of metaclass SoftwareActivationDependencyCompare-Condition
[TPS_MANI_01218]	Cryptographic signature of SoftwareCluster
[TPS_MANI_01219]	License of software in included SoftwareCluster
[TPS_MANI_01220]	Release notes of software in included SoftwareCluster
[TPS_MANI_01221]	Semantics of meta-class SoftwarePackage
[TPS_MANI_01222]	Cryptographic signature of SoftwarePackage
[TPS_MANI_01223]	Semantics of attribute SoftwarePackage.packagerId
[TPS_MANI_01224]	Actions taken after installation of a SoftwarePackage
[TPS_MANI_01225]	Actions taken during installation of a SoftwarePackage
[TPS_MANI_01226]	Machine-specific configuration settings for the UCM module
[TPS_MANI_01227]	Semantics of attribute UcmModuleInstantiation.identifier
[TPS_MANI_01228]	Semantics of meta-class ProcessDesign
[TPS_MANI_01229]	Pre-allocation of a given ProcessDesign on a specific MachineDesign
[TPS_MANI_01230]	Semantics of DiagnosticProvidedDataMapping
[TPS_MANI_01231]	GrantDesign references ProcessDesign
[TPS_MANI_01232]	Semantics of meta-class ComOfferServiceGrantDesign
[TPS_MANI_01233]	Semantics of meta-class ComFindServiceGrantDesign
[TPS_MANI_01234]	Semantics of ComFieldGrantDesign
[TPS_MANI_01235]	Semantics of ComEventGrantDesign
[TPS_MANI_01236]	Semantics of ComMethodGrantDesign
[TPS_MANI_01237]	Semantics of meta-class ComFieldGrant
[TPS_MANI_01238]	Semantics of meta-class ComMethodGrant
[TPS_MANI_01239]	Semantics of meta-class ComEventGrant
[TPS_MANI_01240]	Semantics of meta-class ComOfferServiceGrant
[TPS_MANI_01241]	Semantics of meta-class ComFindServiceGrant
[TPS_MANI_01242]	PortInterfaces used for communication with the AUTOSAR Diagnostic Manager
[TPS_MANI_01243]	Semantics of DiagnosticDataIdentifierInterface
[TPS_MANI_01244]	Semantics of DiagnosticDataElementInterface
[TPS_MANI_01245]	Semantics of DiagnosticDataIdentifierGenericInterface
[TPS_MANI_01246]	Semantics of DiagnosticMonitorInterface
[TPS_MANI_01247]	Semantics of DiagnosticDTCInformationInterface
[TPS_MANI_01248]	Semantics of DiagnosticEventInterface
[TPS_MANI_01249]	Semantics of DiagnosticConditionInterface

▽

△

Number	Heading
[TPS_MANI_01250]	Semantics of DiagnosticIndicatorInterface
[TPS_MANI_01251]	Semantics of DiagnosticSecurityLevelInterface
[TPS_MANI_01252]	Semantics of DiagnosticServiceValidationInterface
[TPS_MANI_01253]	Semantics of DiagnosticOperationCycleInterface
[TPS_MANI_01254]	Semantics of DiagnosticGenericUdsInterface
[TPS_MANI_01255]	Semantics of DiagnosticGenericUdsInterface
[TPS_MANI_01256]	AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticIndicatorInterface
[TPS_MANI_01257]	AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticConditionInterface
[TPS_MANI_01258]	AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticGenericUdsInterface
[TPS_MANI_01259]	Mapping of DiagnosticClearCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01260]	Mapping of DiagnosticIndicator to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01261]	Mapping of DiagnosticMemoryDestination to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01262]	Mapping of DiagnosticSecurityLevel to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01263]	Mapping of DiagnosticDataIdentifier to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01264]	Mapping of DiagnosticServiceInstance to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01265]	Semantics of DiagnosticDownloadInterface and DiagnosticDownloadInterface
[TPS_MANI_01266]	Mapping of DiagnosticServiceInstance for upload/download to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_03223]	Existence of CppImplementationDataType
[TPS_MANI_03224]	Modeling of a Partial Network Cluster
[TPS_MANI_03225]	References to vlans in PncMapping
[TPS_MANI_03226]	Collection of partialNetworks and vlans in NmNetworkHandle

Table F.21: Added Specification Items in R19-03

F.5.2 Changed Specification Items in R19-03

Number	Heading
[TPS_MANI_01012]	Formal modeling of application startup behavior
[TPS_MANI_01013]	Semantics of meta-class `StateDependentStartupConfig`
[TPS_MANI_01017]	Relation of startup configuration to resource group
[TPS_MANI_01041]	Startup configuration supports the definition of a launch sequence dependency
[TPS_MANI_01046]	Semantics of `StateDependentStartupConfig.functionGroupState`
[TPS_MANI_01049]	Mapping of `DiagnosticOperationCycle` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01050]	Mapping of `DiagnosticEnableCondition` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01051]	Mapping of `DiagnosticStorageCondition` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01136]	`AutosarDataPrototype` is the target of the `DataPrototypeInServiceInterfaceRef`
[TPS_MANI_01137]	Applicable use cases for `DataPrototypeInServiceInterfaceRef`
[TPS_MANI_01164]	Semantics of `SoftwareCluster.dependsOn`
[TPS_MANI_01177]	Semantics of `CppImplementationDataType.typeEmitter`
[TPS_MANI_01207]	Standardized values of attribute `PersistencyRedundancyCrc.algorithmFamily`
[TPS_MANI_03070]	Size of a length field for a chosen array or map
[TPS_MANI_03071]	Size of a length field for a chosen structure
[TPS_MANI_03072]	Size of a length field for a chosen union
[TPS_MANI_03073]	Alignment of a dynamic DataPrototype
[TPS_MANI_03074]	Size of a type selector field for a chosen union
[TPS_MANI_03075]	Byte Order of chosen DataPrototype in the serialized data stream
[TPS_MANI_03116]	Size of a length field for a chosen string
[TPS_MANI_03127]	Usage of `End2EndEventProtectionProps`
[TPS_MANI_03128]	Usage of same `dataId` in case of Multi-Binding
[TPS_MANI_03152]	Assignment of a `StateDependentStartupConfig` to a function group state
[TPS_MANI_03187]	Definition of enumeration types
[TPS_MANI_03190]	`CppImplementationDataType` of `category VARIANT`
[TPS_MANI_03202]	Definition of bitfield types
[TPS_MANI_03217]	On-the-wire encoding for a chosen string

Table F.22: Changed Specification Items in R19-03

F.5.3 Deleted Specification Items in R19-03

Number	Heading
[TPS_MANI_01038]	Diagnostic software mapping on the *AUTOSAR adaptive platform*
[TPS_MANI_01170]	Semantics of `CppTemplateArgument.isVariadicTemplate`
[TPS_MANI_01181]	Use cases for the application of `DiagnosticServiceSwMapping`
[TPS_MANI_01193]	Combination of `ModeDependentStartupConfig.machineMode` and `ModeDependentStartupConfig.functionGroupMode`
[TPS_MANI_03066]	Description of machine states
[TPS_MANI_03153]	Semantics of `ModeDependentStartupConfig.functionGroupMode`

Table F.23: Deleted Specification Items in R19-03

F.5.4 Added Constraints in R19-03

Number	Heading
[constr_1687]	Definition of machine state
[constr_1688]	`StateDependentStartupConfig` shall only refer to function group states of the same function group
[constr_1689]	Modeling of a startup dependency between different `Process`es
[constr_1690]	`SoftwareCluster` shall only be referenced by a single `SoftwarePackage`.
[constr_1691]	`UcmModuleInstantiation.identifier` shall be unique
[constr_1692]	Value of `schedulingPriority`
[constr_1693]	Relation of `Executable`, `ProcessDesign`, and `Process`
[constr_1695]	Semantics of a `Grant` depends on the existence of `IamModuleInstantiation`
[constr_1696]	`ClientServerOperation` aggregated by `DiagnosticRoutineInterface`
[constr_1697]	Restriction for `ClientServerOperation` aggregated by a `DiagnosticDataIdentifierInterface` or `DiagnosticDataElementInterface`
[constr_1698]	Target `SwcServiceDependency` of `DiagnosticClearConditionPortMapping.swcServiceDependencyInExecutable`
[constr_1699]	Target `SwcServiceDependency` of `DiagnosticIndicatorPortMapping.swcServiceDependencyInExecutable`
[constr_1700]	Target `SwcServiceDependency` of `DiagnosticMemoryDestinationPortMapping.swcServiceDependencyInExecutable`
[constr_1701]	Target `SwcServiceDependency` of `DiagnosticSecurityLevelPortMapping.swcServiceDependencyInExecutable`
[constr_1702]	Target `SwcServiceDependency` of `DiagnosticServiceDataIdentifierPortMapping.swcServiceDependencyInExecutable`
[constr_1703]	Target `SwcServiceDependency` of `DiagnosticGenericUdsPortMapping.swcServiceDependencyInExecutable`
[constr_1704]	Target `SwcServiceDependency` of `DiagnosticUploadDownloadPortMapping.swcServiceDependencyInExecutable`

▽

△

Number	Heading
[constr_5033]	Compatibility of data types with `category` `VALUE`
[constr_5034]	Compatibility of data types with `category` `BOOLEAN`
[constr_5035]	Compatibility of data types with `category` `STRING`
[constr_5036]	Compatibility of data types with `category` `ARRAY`
[constr_5037]	Compatibility of data types with `category` `ARRAY` with `variableSize`
[constr_5038]	Compatibility of data types with `category` `ARRAY` with `fixedSize`
[constr_5039]	Compatibility of data types with `category` `STRUCTURE`
[constr_5040]	Compatibility of `ApplicationRecordDataType` and `CppImplementationDataType` that both represent an `Optional Element Structure`
[constr_5041]	Compatibility of data types with `category` `ASSOCIATIVE_MAP`
[constr_5042]	No data type mapping for `CppImplementationDataType` of `category` `VARIANT`
[constr_5043]	Forbidden mappings to `CppImplementationDataType`
[constr_5044]	`DataTypeMap` for composite data types
[constr_5045]	Only one `SomeipServiceDiscovery` configuration per VLAN is allowed
[constr_5046]	Usage of `DoIpNetworkConfiguration.eidUseMac`
[constr_5047]	Supported values of `ServiceInstanceToMachineMapping.category`
[constr_5048]	Existence of `TlsCryptoCipherSuite.certificate` in the *server* role

Table F.24: Added Constraints in R19-03

F.5.5 Changed Constraints in R19-03

Number	Heading
[constr_1481]	Usage of `DataPrototypeInServiceInterfaceRef` in the *AUTOSAR adaptive platform*
[constr_1500]	Target `SwcServiceDependency` of `DiagnosticEventPortMapping.swcServiceDependencyInExecutable`
[constr_1501]	Target `SwcServiceDependency` of `DiagnosticOperationCyclePortMapping.swcServiceDependencyInExecutable`
[constr_1502]	Target `SwcServiceDependency` of `DiagnosticEnableConditionPortMapping.swcServiceDependencyInExecutable`
[constr_1503]	Target `SwcServiceDependency` of `DiagnosticStorageConditionPortMapping.swcServiceDependencyInExecutable`
[constr_1551]	Existence of `DataPrototypeInServiceInterfaceRef.dataPrototype` vs. `DataPrototypeInServiceInterfaceRef.elementInImplDatatype`
[constr_1567]	Existence of `SoftwareCluster.subSoftwareCluster` and `SoftwareCluster.dependsOn/conflictsTo`
[constr_1595]	Consistent assignment of TLV data ids to `CppImplementationDataType` or `CppImplementationDataTypeElement`
[constr_1606]	`Process`es with mutual `ExecutionDependency`s

▽

△

Number	Heading
[constr_1615]	Existence of attribute SomeipDataPrototypeTransformationProps.someip-TransformationProps.sessionHandling
[constr_1618]	Ability to shut down
[constr_1619]	Ability to restart
[constr_3396]	Number of Process.stateDependentStartupConfig that refer to the same functionGroupState
[constr_3413]	StateDependentStartupConfig of a Process is mapped to exactly one ResourceGroup
[constr_3421]	Fibex elements applicable for a System of category MACHINE_DESIGN_EXTRACT
[constr_3423]	StateDependentStartupConfig of a Process shall reference a function-GroupState
[constr_3424]	StateDependentStartupConfig shall never reference the functionGroup-State Off
[constr_3447]	ApSomeipTransformationProps.sizeOfArrayLengthField that equals 0

Table F.25: Changed Constraints in R19-03

F.5.6 Deleted Constraints in R19-03

Number	Heading
[constr_1499]	Target SwcServiceDependency of DiagnosticServiceSwMapping. mappedSwcServiceDependencyInExecutable
[constr_1504]	Number of Process.modeDependentStartupConfig that refer to the same ma-chineMode
[constr_1573]	CppTemplateArgument.isVariadicTemplate is set to True
[constr_1574]	Number of CppTemplateArguments with isVariadicTemplate set to True
[constr_1575]	Position of CppTemplateArgument with isVariadicTemplate set to True
[constr_1585]	Standardized values of attribute DiagnosticServiceSwMapping.category
[constr_1586]	DiagnosticServiceSwMapping.category set to DATA_ELEMENT
[constr_1620]	Value of schedulingPriority if schedulingPolicy is set to scheduling-PolicyFifo or schedulingPolicyRoundRobin
[constr_1621]	Value of schedulingPriority if StartupConfig.schedulingPolicyis set to schedulingPolicyOther
[constr_1663]	Standardized values of attribute DiagnosticServiceDataIdentifierMap-ping.category
[constr_3380]	End2EndEventProtectionProps shall not reference an event and a notifier at the same time
[constr_3397]	ModeDependentStartupConfig that refers to a functionGroupMode and to a machineMode
[constr_3398]	ModeDependentStartupConfig that refers to function group modes of different function groups

▽

△

Number	Heading
[constr_3494]	Mandatory Machine States
[constr_3531]	Mandatory definition of `healthChannelId`
[constr_3536]	Mandatory definition of `supervisedEntityId`
[constr_5001]	Usage of `DoIpNetworkConfiguration`.`eidUseMac`
[constr_5002]	Supported values of `ServiceInstanceToMachineMapping`.`category`
[constr_5003]	Existence of `TlsCryptoCipherSuite`.`certificate` in the *server* role

Table F.26: Deleted Constraints in R19-03

F.6 Traceable item history of this document according to AUTOSAR Release R19-11

F.6.1 Added Specification Items in R19-11

Number	Heading
[TPS_MAIN_01281]	Usage of meta-class `RecoveryViaApplicationAction`
[TPS_MANI_01267]	Semantics of attribute `SoftwareClusterDesign`.`dependsOn`
[TPS_MANI_01268]	Semantics of attribute `SoftwareClusterDesign`.`conflictsTo`
[TPS_MANI_01269]	Specification of boundaries for resource consumption
[TPS_MANI_01270]	Reference from `TransformationPropsToServiceInterfaceElementMapping` to `TlvDataIdDefinitionSet`
[TPS_MANI_01271]	Semantics of `Executable`.`loggingBehavior`
[TPS_MANI_01272]	Duplicate entries in `logTraceLogMode`
[TPS_MANI_01273]	Support for trusted Platform
[TPS_MANI_01274]	System `category` for a design description that has one single Adaptive `Machine` in scope
[TPS_MANI_01275]	Semantics of meta-class `ServiceInstanceToSwClusterDesignPortPrototypeMapping`
[TPS_MANI_01276]	Semantics of `CompositionRPortToExecutableRPortMapping` and `CompositionPPortToExecutablePPortMapping`
[TPS_MANI_01277]	Definition of a start-up timeout for a `StartupConfig` of a `Process`
[TPS_MANI_01278]	Definition of a termination timeout for a `StartupConfig` of a `Process`
[TPS_MANI_01279]	Semantics of `Executable`.`reportingBehavior`
[TPS_MANI_01280]	Semantics of meta-class `PhmRecoveryActionInterface`
[TPS_MANI_01282]	Semantics of reference `CompositionPortToExecutablePortMapping`.`processDesign`
[TPS_MANI_01283]	Semantics of meta-class `RawDataStreamInterface`
[TPS_MANI_01284]	Granularity of meta-class `RawDataStreamGrantDesign`
[TPS_MANI_01285]	Purpose of meta-class `RawDataStreamDeployment`

▽

△

Number	Heading
[TPS_MANI_01286]	Semantics of attribute RawDataStreamMethodDeployment.callTimeout
[TPS_MANI_01287]	Semantics of RawDataStreamMapping
[TPS_MANI_01288]	Impact of the SoftwarePackage on the value of function group states on the target platform
[TPS_MANI_01289]	Order of function group states is relevant
[TPS_MANI_01290]	VehiclePackage names affected UCMs
[TPS_MANI_01291]	Identification of an actual UCM in the context of an update campaign
[TPS_MANI_01292]	Definition of fallback-order for UCM master
[TPS_MANI_01294]	Update campaign depends on driver's acceptance
[TPS_MANI_01295]	Semantics of VehicleRolloutStep
[TPS_MANI_01296]	Ordered execution of rollout steps in an update campaign
[TPS_MANI_01297]	Semantics of meta-class UcmStep
[TPS_MANI_01298]	No ordering of VehicleRolloutStep.ucmProcessing
[TPS_MANI_01299]	Aggregation of SoftwarePackageSteps at UcmStep
[TPS_MANI_01300]	Semantics of reference SoftwarePackageStep.transfer.transfer
[TPS_MANI_01301]	Semantics of aggregation SoftwarePackageStep.transfer
[TPS_MANI_01302]	Semantics of reference SoftwarePackageStep.process
[TPS_MANI_01303]	Semantics of reference SoftwarePackageStep.preActivate
[TPS_MANI_01304]	Semantics of reference SoftwarePackageStep.verify
[TPS_MANI_01305]	Semantics of attribute SoftwarePackageStep.activationSwitch
[TPS_MANI_01306]	Simultaneous existence of attributes SoftwarePackageStep.transfer and SoftwarePackageStep.process
[TPS_MANI_01307]	Semantics of meta-class EthernetRawDataStreamGrant
[TPS_MANI_03227]	Usage of ephemeral ports
[TPS_MANI_03228]	Usage of End2EndMethodProtectionProps
[TPS_MANI_03229]	Usage of same End2EndMethodProtectionProps.dataId in case of Multi-Binding
[TPS_MANI_03230]	Sharing timers for ProvidedSomeipServiceInstance
[TPS_MANI_03231]	Sharing timers for RequiredSomeipServiceInstance
[TPS_MANI_03232]	Definition of general IPsec configuration settings
[TPS_MANI_03233]	IPsec mode
[TPS_MANI_03234]	IPsec AH and ESP CipherSuites
[TPS_MANI_03573]	Definition of no minimum deadline supervision
[TPS_MANI_03574]	Definition of no maximum deadline supervision
[TPS_MANI_03575]	Definition of no minimum alive supervision
[TPS_MANI_03576]	Definition of no maximum alive supervision
[TPS_MANI_03577]	headerId required for signal-service-translation
[TPS_MANI_03578]	Signal-based ServiceInterface binding over Ethernet

▽

△

Number	Heading
[TPS_MANI_03579]	Signal-based `ServiceEventDeployment` over `Ethernet`
[TPS_MANI_03580]	Service offer at startup
[TPS_MANI_03581]	Service find at startup
[TPS_MANI_03582]	Service find for required signal
[TPS_MANI_03583]	Service subscribe for required signal
[TPS_MANI_03584]	Definition of transmission triggers for translations with different sources
[TPS_MANI_03585]	Processing order of COM-Stack features
[TPS_MANI_03586]	No transmission trigger for translations with different sources
[TPS_MANI_03587]	Transmission trigger for translations with different sources
[TPS_MANI_03588]	Full translation before transmission triggering
[TPS_MANI_03589]	Reception data filter of COM-Stack
[TPS_MANI_03590]	Transfer properties and transmission modes of COM-Stack
[TPS_MANI_03591]	`SomeipEventDeployment.serializer` equals `signalBased`
[TPS_MANI_03592]	`ISignal` invalidation of COM-Stack
[TPS_MANI_03593]	`handleInvalid` = `dontInvalidate` behavior of COM-Stack
[TPS_MANI_03594]	`handleInvalid` = `replace` behavior of COM-Stack
[TPS_MANI_03595]	Update Bit support for `ISignal`
[TPS_MANI_03596]	Update Bit support for `ISignalIPdu`
[TPS_MANI_03597]	Support for `MultiplexedIPdu`
[TPS_MANI_03598]	Expected check period of E2E-Protected payload
[TPS_MANI_03599]	Expected update period of E2E-Protected payload
[TPS_MANI_03600]	Signal-service-translation of E2E protected payload
[TPS_MANI_03601]	Signal-service-translation of E2E protected payload - timeout handling
[TPS_MANI_03602]	Signal-service-translation of E2E protected payload - error handling
[TPS_MANI_03603]	Service-signal-translation of E2E protected payload
[TPS_MANI_03604]	Service-signal-translation of E2E protected payload - timeout handling
[TPS_MANI_03605]	Service-signal-translation of E2E protected payload - error handling
[TPS_MANI_03606]	Service offer for provided signal
[TPS_MANI_03607]	Handling of safe signal-service-translation in one `Executable`
[TPS_MANI_03608]	Support for safe signal-service-translation and service-signal-translation
[TPS_MANI_03609]	Support for safe signal-service-translation with same or different E2E profiles
[TPS_MANI_03610]	1:n mapping for E2E protected data
[TPS_MANI_03611]	E2E protected target out of E2E protected sources
[TPS_MANI_03612]	Sufficient ASIL level of translation software
[TPS_MANI_03614]	No translation of not OK E2E protected data out of several sources
[TPS_MANI_03615]	`SomeipEventDeployment.serializer` equals `someip`
[TPS_MANI_03616]	Semantic versioning of `ServiceInterface.majorVersion` and `ServiceInterface.minorVersion`

▽

△

Number	Heading
[TPS_MANI_03617]	Version mapping between `ServiceInterface` and `ServiceInterfaceDeployment`
[TPS_MANI_03618]	Usage of `RequiredSomeipServiceInstance.blacklistedVersion`
[TPS_MANI_03619]	SOME/IP Service search for `requiredMinorVersion`
[TPS_MANI_03620]	Service discovery control
[TPS_MANI_03621]	Data filter inside the signal-service-translation

Table F.27: Added Specification Items in R19-11

F.6.2 Changed Specification Items in R19-11

Number	Heading
[TPS_MANI_01032]	Usage of `ServiceInterfaceMapping`
[TPS_MANI_01057]	Semantics of `RPortPrototypeProps.searchIntention`
[TPS_MANI_01164]	Semantics of `SoftwareCluster.dependsOn`
[TPS_MANI_01196]	Semantics of `PersistencyDeployment.minimumSustainedSize`
[TPS_MANI_01197]	Semantics of `PersistencyDeployment.maximumAllowedSize`
[TPS_MANI_01214]	Semantics of `SoftwareCluster.conflictsTo`
[TPS_MANI_01215]	Semantics of meta-class `SoftwareActivationDependencyFormula`
[TPS_MANI_01216]	Semantics of meta-class `SoftwareActivationDependencyFormulaPart`
[TPS_MANI_01217]	Semantics of metaclass `SoftwareActivationDependencyCompareCondition`
[TPS_MANI_01249]	Semantics of `DiagnosticConditionInterface`
[TPS_MANI_01255]	Semantics of DoIP `DiagnosticPortInterface`s
[TPS_MANI_03004]	IPv4 Multicast event destination address
[TPS_MANI_03010]	Udp Transport Protocol Configuration in case of IP-Multicast
[TPS_MANI_03061]	IPv6 Multicast event destination address
[TPS_MANI_03114]	Usage of `AssemblySwConnector`s in the System Design model
[TPS_MANI_03130]	Standardized `E2EProfileConfiguration.profileName` values
[TPS_MANI_03160]	Log and Trace configuration options in the Execution Manifest
[TPS_MANI_03161]	Log and Trace configuration options in the Service Instance Manifest
[TPS_MANI_03167]	Network configuration for Nm
[TPS_MANI_03205]	IPsec policy
[TPS_MANI_03206]	IPsec AH and ESP protocol configuration
[TPS_MANI_03207]	IPsec Internet Key Exchange protocol configuration

▽

△

Number	Heading
[TPS_MANI_03216]	Existence of TlsCryptoCipherSuite.certificate and TlsCryptoCipherSuite.pskIdentity in the *client* role

Table F.28: Changed Specification Items in R19-11

F.6.3 Deleted Specification Items in R19-11

Number	Heading
[TPS_MANI_01051]	Mapping of DiagnosticStorageCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01052]	Semantics of RPortPrototypeProps.portInstantiationBehavior
[TPS_MANI_01162]	Semantics of SoftwareClusterDesign.dependsOn
[TPS_MANI_03120]	Signal-based ServiceInterface binding
[TPS_MANI_03146]	Configuration of timeouts for a selected machine state or function group state
[TPS_MANI_03149]	Definition of a start-up timeout for a Process
[TPS_MANI_03150]	Definition of a termination timeout for a Process
[TPS_MANI_03550]	Usage of RPortPrototype for the interaction with Platform Health Management

Table F.29: Deleted Specification Items in R19-11

F.6.4 Added Constraints in R19-11

Number	Heading
[constr_1705]	Target of reference SoftwareActivationDependencyCompareCondition.softwareActivationDependency
[constr_1707]	Eligible subclasses of HeapUsage in the context of StateDependentStartupConfig.resourceConsumption
[constr_1708]	Combination of CppImplementationDataTypeElement.isOptional and CppImplementationDataTypeElementQualifier.inplace
[constr_1709]	Applicability of attribute PersistencyRedundancyEnum.redundantPerKey
[constr_1710]	Consistency of values of attributes PersistencyInterface.redundancy and PersistencyRedundancyHandling.scope
[constr_1723]	ProvidedSomeipServiceInstance shall be unique in respect of serviceInstanceId, serviceInterfaceId and majorVersion
[constr_1727]	Qualified combinations of PortPrototypes and PhmSupervisedEntityInterface on application software level
[constr_1728]	Qualified combinations of PortPrototypes and PhmHealthChannelInterface on application software level

▽

△

Number	Heading
[constr_1729]	Qualified combinations of PortPrototypes and PhmRecoveryActionInterface on application software level
[constr_1730]	Restriction regarding the modeling of the PhmRecoveryActionInterface.recovery
[constr_1731]	Value of UcmDescription.identifier in the scope of a VehiclePackage
[constr_1732]	Existence of attribute activationSwitch set to True in the context of the enclosing UcmStep
[constr_1733]	Simultaneous existence of SoftwarePackageStep.preActivate and SoftwarePackageStep.verify
[constr_1734]	Restriction for attribute SoftwarePackageStep.activationSwitch
[constr_1736]	Multiplicity of reference LogicalSupervision.initialCheckpoint
[constr_1737]	Multiplicity of reference LogicalSupervision.finalCheckpoint
[constr_1738]	Multiplicity of reference GlobalSupervision.localSupervision
[constr_1739]	Multiplicity of aggregation LocalSupervision.transition
[constr_1740]	Multiplicity of reference LogicalSupervision.transition
[constr_1742]	Multiplicity of reference SupervisionCheckpoint.phmCheckpoint
[constr_3550]	Existence of ServiceInstanceToSignalMapping for an event with signalBased serialization
[constr_3551]	Full mapping of target ISignalGroup
[constr_3552]	Full mapping of target event
[constr_3553]	Existence of ServiceInstanceToSignalMapping for an field with signalBased serialization
[constr_3554]	E2E protection configuration check
[constr_3555]	No support for useAsCryptographicIPdu is true
[constr_3556]	Unique transport layer mapping
[constr_3557]	Mandatory majorVersion at SomeipServiceInterfaceDeployment.serviceInterfaceVersion
[constr_3558]	RequiredSomeipServiceInstance.blacklistedVersion is restricted to the usage of minorVersion
[constr_3561]	minimumMinorVersion and RequiredSomeipServiceInstance.requiredMinorVersion value
[constr_3562]	Existence of NonqueuedReceiverComSpec.filter
[constr_5052]	SOME/IP ServiceInstances of the same serviceInterface on one Machine
[constr_5056]	Restriction of CompositionSwComponentType.connector usage in AP
[constr_5057]	PassThroughSwConnector and ServiceInterfaceMapping
[constr_5102]	Usage of remote port ranges in IPSecRule is not allowed
[constr_5103]	Usage of local port ranges in IPSecRule is not allowed

Table F.30: Added Constraints in R19-11

F.6.5 Changed Constraints in R19-11

Number	Heading
[constr_1561]	Existence of SoftwareClusterDesign.subSoftwareCluster and SoftwareClusterDesign.dependsOn.dependentSoftwareClusterDesign
[constr_1567]	Existence of SoftwareCluster.subSoftwareCluster and SoftwareCluster.dependsOn/conflictsTo
[constr_1570]	Restriction for UserDefinedServiceInterfaceDeployment of category SERVICE_INTERFACE_DEPLOYMENT_IPC
[constr_1579]	SwDataDefProps applicable to CppImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*
[constr_1630]	No definition of length field sizes on DataPrototype level in case of TLV usage
[constr_3375]	method with attribute fireAndForget set to true shall not reference an ApApplicationError
[constr_3419]	Allowed usage of UdpNmNetworkConfiguration attributes
[constr_3426]	The logTraceFilePath is mandatory in case that logTraceLogMode is set to file
[constr_3427]	The logTraceFilePath is only relevant if logTraceLogMode is set to file
[constr_3493]	Applicable attributes for standardized E2E Profiles
[constr_5048]	Existence of TlsCryptoCipherSuite.certificate and TlsCryptoCipherSuite.pskIdentity in the *server* role

Table F.31: Changed Constraints in R19-11

F.6.6 Deleted Constraints in R19-11

Number	Heading
[constr_1503]	Target SwcServiceDependency of DiagnosticStorageConditionPortMapping.swcServiceDependencyInExecutable
[constr_3387]	Compatibility of PortPrototypes of different ServiceInterfaces
[constr_3388]	Compatibility of events
[constr_3389]	Compatibility of methods
[constr_3390]	Compatibility of fields
[constr_3411]	eventMulticastUdpPort, ipv4MulticastIpAddress and ipv6MulticastIpAddress not relevant for RequiredSomeipServiceInstances
[constr_3420]	System category for a design description that has one single Adaptive Machine in scope

Table F.32: Deleted Constraints in R19-11

F.7 Traceable item history of this document according to AUTOSAR Release R20-11

F.7.1 Added Specification Items in R20-11

Number	Heading
[TPS_MANI_01308]	`Process` is not designed for re-usability
[TPS_MANI_01309]	Semantics of attribute `CppImplementationDataType.headerFile`
[TPS_MANI_01310]	Semantics of `SoftwareClusterDesign.dependsOn`
[TPS_MANI_01311]	Handling of manufacturer checks
[TPS_MANI_01312]	Handling of supplier checks
[TPS_MANI_01313]	Definition of `updateStrategy` on element level
[TPS_MANI_01314]	Further qualification of properties of `PortPrototype`s typed by `PersistencyKeyValueStorageInterface`s
[TPS_MANI_01315]	`PersistencyKeyValuePair.initValue` overrides `PersistencyDataRequiredComSpec.initValue`
[TPS_MANI_01316]	Existence of `ServiceInstanceToPortPrototypeMapping.processDesign`
[TPS_MANI_01317]	Existence of `ServiceInstanceToPortPrototypeMapping.process`
[TPS_MANI_01319]	Modeling of redundancy in the context of `PersistencyInterface`
[TPS_MANI_01320]	Definition of redundancy on interface level may be overruled in deployment
[TPS_MANI_01321]	Semantics of meta-class `PersistencyDeploymentElement`
[TPS_MANI_01322]	Semantics of meta-class `PersistencyPortPrototypeToDeploymentMapping`
[TPS_MANI_01323]	Matching pairs of `PersistencyDataElement` and `PersistencyKeyValuePair`
[TPS_MANI_01324]	Semantics of E2E attributes in `ClientComSpec`
[TPS_MANI_01325]	Semantics of E2E attributes in `ServerComSpec`
[TPS_MANI_01326]	Generic Mapping to a `DiagnosticServiceInstance` on the AUTOSAR Adaptive Platform
[TPS_MANI_01327]	Value of `EndToEndTransformationComSpecProps.disableEndToEndCheck` vs. value of `EndToEndTransformationComSpecProps.disableEndToEndStateMachine`
[TPS_MANI_01328]	Standardized values for attribute `StartupConfig.schedulingPolicy`
[TPS_MANI_01329]	Reference to model elements in different `SoftwareCluster`s
[TPS_MANI_01330]	Definition of machine function group
[TPS_MANI_01331]	Standardized values of attribute `SoftwareCluster.category`
[TPS_MANI_01332]	Semantics of `DiagnosticEcuResetInterface`
[TPS_MANI_01333]	Attribute `NotAvailableValueSpecification.defaultPattern` is not applicable
[TPS_MANI_01334]	Semantics of `StartupConfig.terminationBehavior`



△

Number	Heading
[TPS_MANI_01335]	Semantics of SoftwareClusterDependencyFormula.category
[TPS_MANI_01336]	Two use cases for using the DiagnosticEventPortMapping
[TPS_MANI_01337]	Standardized values for attribute Process.functionCluster-Affiliation
[TPS_MANI_01338]	Semantics of SecurityEventReportToSecurityEventDefinition-Mapping
[TPS_MANI_01339]	Existence of the SecurityEventReportToSecurityEvent-DefinitionMapping is motivated by the AUTOSAR methodology
[TPS_MANI_01340]	Semantics of SecurityEventReportInterface
[TPS_MANI_01341]	Security events that are actually reported by a local IdsM
[TPS_MANI_01342]	Semantics of SecurityEventMapping
[TPS_MANI_03235]	Usage of ApSomeipTransformationProps.sessionHandling
[TPS_MANI_03236]	Mapping of ProvidedSomeipServiceInstance to different PPort-Prototypes
[TPS_MANI_03237]	Transport Protocol attributes defined for a RequiredSomeipService-Instance
[TPS_MANI_03238]	Definition of ComMethodGrantDesign.remoteSubject
[TPS_MANI_03239]	Definition of ComEventGrantDesign.remoteSubject
[TPS_MANI_03240]	Modeling of a remote peer in case of TLS-based secure channel
[TPS_MANI_03241]	Modeling of relevant TlsSecureComProps for TlsIamRemoteSubject
[TPS_MANI_03242]	Modeling of a remote peer in case of IPsec-based secure channel
[TPS_MANI_03244]	Modeling of a remote peer in case of a general IP communication
[TPS_MANI_03245]	Definition of ComMethodGrant.remoteSubjects on server side
[TPS_MANI_03246]	Definition of ComMethodGrant.remoteSubjects on client side
[TPS_MANI_03247]	Definition of ComEventGrant.remoteSubjects on provider side
[TPS_MANI_03248]	Definition of ComEventGrant.remoteSubjects on receiver side
[TPS_MANI_03249]	Definition of ComFieldGrant.remoteSubjects on provider side
[TPS_MANI_03250]	Definition of ComFieldGrant.remoteSubjects on client side
[TPS_MANI_03251]	Definition of ComFieldGrantDesign.remoteSubject
[TPS_MANI_03252]	Usage of same End2EndMethodProtectionProps.sourceId in case of Multi-Binding
[TPS_MANI_03253]	Interaction with crypto software
[TPS_MANI_03254]	Modeling of application that uses and modifies a Crypto Key
[TPS_MANI_03255]	Modeling of Key Manager application that manages a Crypto Key that is used by Stack Services
[TPS_MANI_03256]	Modeling of application that accesses a Crypto Certificate
[TPS_MANI_03257]	Modeling of application that accesses a Crypto Provider
[TPS_MANI_03258]	Modeling of application designed as trust-master
[TPS_MANI_03259]	Linking of Crypto Certificate to a Crypto Key Slot
[TPS_MANI_03260]	Semantics of meta-class CryptoModuleInstantiation

▽

△

Number	Heading
[TPS_MANI_03261]	Support of CryptoProviders
[TPS_MANI_03262]	Semantics of CryptoProviderToPortPrototypeMapping
[TPS_MANI_03263]	Assignment of CryptoKeySlots to CryptoProviders
[TPS_MANI_03264]	Semantics of CryptoKeySlotToPortPrototypeMapping
[TPS_MANI_03265]	Support of CryptoCertificates
[TPS_MANI_03266]	Semantics of CryptoCertificateToCryptoKeySlotMapping
[TPS_MANI_03267]	Semantics of CryptoCertificateToPortPrototypeMapping
[TPS_MANI_03268]	Semantics of FunctionalClusterInteractsWithFunctionalClusterMapping
[TPS_MANI_03269]	Semantics of ComCertificateToCryptoCertificateMapping
[TPS_MANI_03270]	Semantics of ComKeyToCryptoKeySlotMapping
[TPS_MANI_03271]	Semantics of ComSecOcToCryptoKeySlotMapping
[TPS_MANI_03272]	Semantics of PersistencyDeploymentToCryptoKeySlotMapping
[TPS_MANI_03273]	Semantics of PersistencyDeploymentElementToCryptoKeySlotMapping
[TPS_MANI_03274]	Configuration of log and trace message source
[TPS_MANI_03275]	Configuration of log and trace message source on design level
[TPS_MANI_03276]	Semantics of PersistencyDeploymentToDltLogChannelMapping
[TPS_MANI_03622]	DDS Transport Protocols are up to the stack implementer
[TPS_MANI_03623]	Usage of checkpointId in application code
[TPS_MANI_03624]	Usage of statusId in application code
[TPS_MANI_03625]	Consistency of HealthChannelExternalReportedStatus.statusId and PhmHealthChannelStatus.statusId
[TPS_MANI_03626]	Consistency of SupervisionCheckpoint.phmCheckpoint and PhmCheckpoint.checkpointId
[TPS_MANI_03627]	No signal-service-translation for methods
[TPS_MANI_03628]	Standardized values of ServiceInterface.category
[TPS_MANI_03629]	Relation of ServiceInstanceToSignalMapping and CommunicationConnector
[TPS_MANI_03630]	Semantics of triggersRecoveryNotification
[TPS_MANI_03631]	Semantics of meta-class PhmHealthChannelRecoveryNotificationInterface
[TPS_MANI_03632]	Semantics of TimeBaseProviderToPersistencyMapping

Table F.33: Added Specification Items in R20-11

F.7.2 Changed Specification Items in R20-11

Number	Heading
[TPS_MANI_01061]	Requirements on scheduling
[TPS_MANI_01065]	Purpose of PersistencyKeyValueStorageInterface
[TPS_MANI_01067]	Purpose of PersistencyFileStorageInterface
[TPS_MANI_01068]	Semantics of PersistencyFileStorageInterface.maxNumberOfFiles
[TPS_MANI_01073]	Semantics of PortPrototype typed by PersistencyKeyValueStorageInterface
[TPS_MANI_01078]	Semantics of PersistencyPortPrototypeToKeyValueStorageMapping
[TPS_MANI_01079]	Semantics of PersistencyKeyValueStorage
[TPS_MANI_01080]	Semantics of meta-class PersistencyPortPrototypeToFileStorageMapping
[TPS_MANI_01081]	Semantics of PortPrototype typed by PersistencyFileStorageInterface
[TPS_MANI_01106]	Specification of intentions for the receiver of events or field notifiers
[TPS_MANI_01107]	Specification of intentions for the sender of events or field notifiers
[TPS_MANI_01108]	Specification of intentions for the caller of a methods or field setter/getter
[TPS_MANI_01135]	Semantics of PersistencyKeyValueStorageInterface.dataTypeForSerialization
[TPS_MANI_01138]	Semantics of PersistencyKeyValueStorageInterface.dataElement
[TPS_MANI_01139]	Semantics of PersistencyInterface.updateStrategy
[TPS_MANI_01140]	Semantics of PersistencyInterfaceElement.updateStrategy
[TPS_MANI_01142]	Semantics of PersistencyFileElement
[TPS_MANI_01144]	Semantics of PersistencyKeyValuePair
[TPS_MANI_01147]	Semantics of PersistencyDeployment.updateStrategy
[TPS_MANI_01148]	Semantics of PersistencyDeploymentElement.updateStrategy
[TPS_MANI_01149]	Semantics of PersistencyFileStorage.file
[TPS_MANI_01150]	Semantics of PersistencyFileStorage
[TPS_MANI_01155]	PersistencyDeployment.updateStrategy overrides PersistencyInterface.updateStrategy
[TPS_MANI_01156]	PersistencyDeploymentElement.updateStrategy overrides PersistencyDeployment.updateStrategy
[TPS_MANI_01157]	Semantics of updateStrategy on collection level
[TPS_MANI_01159]	Semantics of updateStrategy on element level
[TPS_MANI_01160]	Definition of initial value for PersistencyDataElement
[TPS_MANI_01164]	Semantics of SoftwareCluster.dependsOn
[TPS_MANI_01176]	Standardized value for attribute CppImplementationDataType.typeEmitter
[TPS_MANI_01177]	Semantics of attribute CppImplementationDataType.typeEmitter



△

Number	Heading
[TPS_MANI_01179]	Semantics of PersistencyFileElement.contentUri/PersistencyFile.contentUri vs. PersistencyFileStorage.uri and PersistencyFileElement.fileName/PersistencyFile.fileName
[TPS_MANI_01180]	Collection of data types that requires serialization support
[TPS_MANI_01182]	Value of PersistencyDeploymentElement.updateStrategy overrides PersistencyInterfaceElement.updateStrategy
[TPS_MANI_01187]	Matching pairs of PersistencyFileElement and PersistencyFile
[TPS_MANI_01194]	Semantics of PersistencyInterface.minimumSustainedSize
[TPS_MANI_01196]	Semantics of PersistencyDeployment.minimumSustainedSize
[TPS_MANI_01197]	Semantics of PersistencyDeployment.maximumAllowedSize
[TPS_MANI_01204]	Specification of redundancy of persistent data
[TPS_MANI_01206]	Modeling of redundancy in the context of PersistencyDeployment
[TPS_MANI_01207]	Standardized values of attribute PersistencyRedundancyChecksum.algorithmFamily
[TPS_MANI_01213]	Semantics of meta-class StrongRevisionLabelString
[TPS_MANI_01214]	Semantics of SoftwareCluster.conflictsTo
[TPS_MANI_01215]	Semantics of meta-class SoftwareClusterDependencyFormula
[TPS_MANI_01216]	Semantics of meta-class SoftwareClusterDependencyFormulaPart
[TPS_MANI_01217]	Semantics of meta-class SoftwareClusterDependencyCompareCondition
[TPS_MANI_01263]	Mapping of DiagnosticDataIdentifier or DiagnosticDataElement to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01272]	Duplicate entries in logTraceLogMode
[TPS_MANI_01280]	Semantics of meta-class PhmSupervisionRecoveryNotificationInterface
[TPS_MANI_01284]	Granularity of meta-class RawDataStreamGrantDesign
[TPS_MANI_01285]	Purpose of meta-class RawDataStreamDeployment
[TPS_MANI_01300]	Semantics of reference SoftwarePackageStep.transfer.transfer
[TPS_MANI_03059]	RequiredSomeipServiceInstance.requiredServiceInstanceId
[TPS_MANI_03111]	Mapping between method and operationlocated in a ClientServerInterface
[TPS_MANI_03113]	Mapping between a field and elements of Classic Platform PortInterfaces
[TPS_MANI_03130]	Standardized E2EProfileConfiguration.profileName values
[TPS_MANI_03145]	Description of a function group
[TPS_MANI_03160]	Further configuration options in DltLogChannel
[TPS_MANI_03163]	Network configuration for Log and Trace messages
[TPS_MANI_03165]	Network Interface configuration for DoIP
[TPS_MANI_03194]	Function Group State
[TPS_MANI_03195]	Off state in Function Group
[TPS_MANI_03207]	IPsec Internet Key Exchange protocol configuration

▽

△

Number	Heading
[TPS_MANI_03209]	The meaning of `MachineDesign.accessControl`
[TPS_MANI_03516]	Status for `HealthChannelSupervision`
[TPS_MANI_03517]	Evaluation of `HealthChannelExternalStatus`
[TPS_MANI_03535]	Definition of Time Synchronization interaction
[TPS_MANI_03536]	Time Synchronization interaction in a provider role
[TPS_MANI_03537]	Time Synchronization interaction in a consumer role
[TPS_MANI_03539]	Definition of Time-Base Resources
[TPS_MANI_03541]	Definition of `SynchronizedTimeBaseConsumer`
[TPS_MANI_03542]	Definition of `SynchronizedTimeBaseProvider`
[TPS_MANI_03543]	Definition of time sync correction attributes
[TPS_MANI_03546]	Definition of reported health status `RPortPrototype`
[TPS_MANI_03548]	Definition of `TimeSyncPortPrototypeToTimeBaseMapping`
[TPS_MANI_03549]	Usage of `PortPrototype` for the interaction with Time Synchronization
[TPS_MANI_03551]	Definition of Time Base kind
[TPS_MANI_03556]	DDS-RPC Service Binding
[TPS_MANI_03557]	DDS `ClientServerOperation` Binding
[TPS_MANI_03558]	DDS `Field` Binding
[TPS_MANI_03598]	Expected check period of E2E-Protected payload
[TPS_MANI_03599]	Expected update period of E2E-Protected payload
[TPS_MANI_03612]	Sufficient ASIL level of translation software
[TPS_MANI_03617]	Version mapping between `ServiceInterface` and `ServiceInterfaceDeployment`

Table F.34: Changed Specification Items in R20-11

F.7.3 Deleted Specification Items in R20-11

Number	Heading
[TPS_MAIN_01281]	Usage of meta-class `RecoveryViaApplicationAction`
[TPS_MANI_01015]	Semantics of meta-class `StartupOption`
[TPS_MANI_01059]	Different values of `optionKind` within a `StartupConfig.startupOption`
[TPS_MANI_01069]	Further qualification of properties of `PortPrototype`s typed by `PersistencyKeyValueDatabaseInterface`s
[TPS_MANI_01141]	Semantics of `PersistencyFileProxyInterface.updateStrategy`
[TPS_MANI_01143]	Semantics of `PersistencyFileProxy.updateStrategy`
[TPS_MANI_01151]	Semantics of `PersistencyFileArray.updateStrategy`
[TPS_MANI_01152]	Semantics of `PersistencyFile.updateStrategy`

▽

△

Number	Heading
[TPS_MANI_01154]	PersistencyFileArray.updateStrategy overrides PersistencyFileProxyInterface.updateStrategy
[TPS_MANI_01158]	PersistencyFile.updateStrategy overrides PersistencyFileArray.updateStrategy
[TPS_MANI_01163]	Impact of values of category on the semantics of SoftwareCluster
[TPS_MANI_01183]	PersistencyFile.updateStrategy overrides PersistencyFileProxy.updateStrategy
[TPS_MANI_01195]	Semantics of PersistencyFileProxyInterface.minimumSustainedSize
[TPS_MANI_01224]	Actions taken after installation of a SoftwarePackage
[TPS_MANI_01264]	Mapping of DiagnosticServiceInstance to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01266]	Mapping of DiagnosticServiceInstance for upload/download to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01267]	Semantics of attribute SoftwareClusterDesign.dependsOn
[TPS_MANI_01268]	Semantics of attribute SoftwareClusterDesign.conflictsTo
[TPS_MANI_01283]	Semantics of meta-class RawDataStreamInterface
[TPS_MANI_01286]	Semantics of attribute RawDataStreamMethodDeployment.callTimeout
[TPS_MANI_03065]	Hardware resources of the machine
[TPS_MANI_03161]	Log and Trace configuration options in the Service Instance Manifest
[TPS_MANI_03518]	PhmLogicalExpression definition
[TPS_MANI_03519]	PhmRule definition
[TPS_MANI_03520]	Execution of PhmActionList with actionListExecution=triggeredOnEvaluation
[TPS_MANI_03521]	Execution of PhmActionList with actionListExecution=triggeredOnChange
[TPS_MANI_03522]	Definition of actions for application software
[TPS_MANI_03523]	Definition of actions for Platform Instance
[TPS_MANI_03524]	Definition of actions for Watchdog
[TPS_MANI_03538]	Time Synchronization interaction with a local Time Base
[TPS_MANI_03540]	Definition of PureLocalTimeBase
[TPS_MANI_03559]	Definition of DdsProvidedServiceInstance.methodQosProps
[TPS_MANI_03560]	qosProfile of DdsProvidedServiceInstance.methodQosProps is optional
[TPS_MANI_03563]	Definition of DdsProvidedServiceInstance.fieldGetSetQosProps
[TPS_MANI_03564]	qosProfile of DdsProvidedServiceInstance.fieldGetSetQosProps is optional
[TPS_MANI_03565]	Definition of DdsRequiredServiceInstance.methodQosProps
[TPS_MANI_03566]	qosProfile of DdsRequiredServiceInstance.methodQosProps is optional
[TPS_MANI_03569]	Definition of DdsRequiredServiceInstance.fieldGetSetQosProps

▽

△

Number	Heading
[TPS_MANI_03570]	qosProfile of DdsRequiredServiceInstance.fieldGetSetQosProps is optional
[TPS_MANI_03571]	transportPlugin for DdsProvidedServiceInstance
[TPS_MANI_03572]	transportPlugin for DdsRequiredServiceInstance

Table F.35: Deleted Specification Items in R20-11

F.7.4 Added Constraints in R20-11

Number	Heading
[constr_1743]	CppImplementationDataType.headerFile vs. CppImplementationDataType.typeEmitter
[constr_1744]	Definition of process state In the context of the ExecutionDependency
[constr_1746]	Mutual exclusive existence of PersistencyInterface.redundancy and PersistencyInterface.redundancyHandling
[constr_1747]	Completeness of the SoftwareCluster.version
[constr_1748]	Existence of references TlvDataIdDefinition.tlvArgument, TlvDataIdDefinition.tlvRecordElement, and TlvDataIdDefinition.tlvSubElement
[constr_1751]	Value of PersistencyRedundancyMOutOfN.n and PersistencyRedundancyMOutOfN.m
[constr_1764]	Counterpart of PhmCheckpoint
[constr_1765]	Diagnostic Services eligible for DiagnosticServiceGenericMapping
[constr_1769]	Existence of ProcessArgument.argument
[constr_1770]	Value of ProvidedSomeipServiceInstance.serviceInstanceId
[constr_1784]	Restriction for the reference to UploadableExclusivePackageElement
[constr_1785]	Restriction regarding the reference into another SoftwareCluster
[constr_1786]	Restriction to use functionGroup in terms of SoftwareCluster
[constr_1787]	Restricted use of function groups in the context of a SoftwareCluster
[constr_1788]	Restriction to SoftwareCluster of category PLATFORM_CORE
[constr_1789]	Scope of machine function group
[constr_1809]	Global supervision restricted to one function group
[constr_3563]	Mandatory topic name values
[constr_3564]	Consistency between DDS Service Interface Deployment and Provided DDS Service Instance
[constr_3565]	Consistency between DDS Service Interface Deployment and Required DDS Service Instance
[constr_3568]	No support for cross PlatformHealthManagementContribution references
[constr_3569]	Applicability of attribute invalidValue on CppImplementationDataType of category TYPE_REFERENCE

▽

△

Number	Heading
[constr_3612]	Multiplicity of references recoveryNotification, recoveryAction, and process at RecoveryNotificationToPPortPrototypeMapping
[constr_3613]	Reference to a PhmSupervisionRecoveryNotificationInterface in the context of a HealthChannelSupervision
[constr_3614]	Reference to a PhmHealthChannelRecoveryNotificationInterface in the context of a HealthChannelExternalStatus
[constr_3619]	Mandatory references of TimeBaseProviderToPersistencyMapping
[constr_5115]	Search for a specific SOME/IP ServiceInstance and for all SOME/IP ServiceInstances over the same RPortPrototype
[constr_5155]	SomeipServiceInstanceToMachineMapping only supports a single Address Family
[constr_5156]	SomeipEventDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_5161]	RequiredSomeipServiceInstance that is mapped by a SomeipServiceInstanceToMachineMapping without a configured tcpPort and udpPort
[constr_5227]	Mandatory elements of UdpNmCluster
[constr_5228]	Partial Networking timing constraint
[constr_5230]	Existence of attribute E2EProfileCompatibilityProps.transitToInvalidExtended is mandatory for each E2EProfileConfiguration
[constr_5238]	CryptoKeySlotAllowedModification.restrictUpdate and the relationship to maxNumberOfAllowedUpdates
[constr_5239]	Predefined values for CryptoKeySlotContentAllowedUsage.allowedKeyslotUsage
[constr_5240]	Restriction applicable for CryptoProviderToPortPrototypeMapping.portPrototype
[constr_5241]	Restriction applicable for CryptoKeySlotToPortPrototypeMapping.portPrototype
[constr_5242]	Restriction applicable for CryptoCertificateToPortPrototypeMapping.portPrototype
[constr_5243]	Restriction of LogAndTraceInstantiation.dltEcuId attribute value
[constr_10002]	Only one mapping per PortPrototype
[constr_10003]	Restriction for the existence of DiagnosticServiceDataIdentifierPortMapping.diagnosticDataIdentifier vs. DiagnosticServiceDataIdentifierPortMapping.diagnosticDataElement
[constr_10004]	Consistency of DiagnosticServiceGenericMapping for PortPrototype typed by DiagnosticDataIdentifierGenericInterface
[constr_10007]	Existence of ProcessExecutionError.executionError
[constr_10008]	Value of ProcessExecutionError.executionError
[constr_10010]	Usage of attribute category in a SoftwareClusterDependencyFormula
[constr_10011]	Definition of sub-software-cluster
[constr_10021]	Existence of IdsmModuleInstantiation
[constr_10022]	Restriction for SecurityEventMapping.process.securityEvent.id w.r.t SecurityEventMapping.id

▽

△

Number	Heading
[constr_10023]	Mandatory content of any functionGroup

Table F.36: Added Constraints in R20-11

F.7.5 Changed Constraints in R20-11

Number	Heading
[constr_1490]	Allowed value for Executable.category if ProcessToMachineMapping references a NonOsModuleInstantiation
[constr_1500]	Target SwcServiceDependency of DiagnosticEventPortMapping.swcServiceDependencyInExecutable
[constr_1507]	PortInterfaceToDataTypeMapping is only applicable to ServiceInterface or PersistencyKeyValueStorageInterface
[constr_1555]	Restriction applicable for PersistencyPortPrototypeToKeyValueStorageMapping.portPrototype
[constr_1556]	Restriction applicable for PersistencyPortPrototypeToFileStorageMapping.portPrototype
[constr_1564]	Existence of SoftwareCluster.diagnosticAddress
[constr_1566]	Usage of SoftwareCluster.containedARElement
[constr_1568]	Existence of SoftwareCluster.diagnosticExtract
[constr_1581]	Value of fileElement.fileName
[constr_1589]	Value of file.fileName
[constr_1613]	File name of matching pairs of PersistencyFileElement and PersistencyFile
[constr_1659]	Restriction for the usage of CppImplementationDataTypeElementQualifier.inplace
[constr_1666]	References from PersistencyPortPrototypeToKeyValueStorageMapping to PersistencyKeyValueStorage
[constr_1667]	References from PersistencyPortPrototypeToFileStorageMapping to PersistencyFileStorage
[constr_1668]	Allowed combinations of PersistencyRedundancyChecksum.length and algorithmFamily
[constr_1673]	Existence of attributes hasGetter, hasSetter, and hasNotifier
[constr_1710]	Consistency of values of attributes PersistencyInterface.redundancy and PersistencyRedundancyHandling.scope
[constr_1729]	Qualified combinations of PortPrototypes and PhmSupervisionRecoveryNotificationInterface / PhmHealthChannelRecoveryNotificationInterface on State Management software level
[constr_3305]	Value of attribute SomeipEventDeployment.eventId shall be unique
[constr_3306]	Value of attribute methodId shall be unique per SomeipServiceInterfaceDeployment

▽

△

Number	Heading
[constr_3356]	Restriction in usage of ApSomeipTransformationProps.alignment
[constr_3414]	Allowed usage of PlatformModuleEthernetEndpointConfiguration attributes
[constr_3421]	Fibex elements applicable for a System of category MACHINE_DESIGN_EXTRACT
[constr_3426]	The logTraceFilePath is mandatory in case that logTraceLogMode is set to file
[constr_3427]	The logTraceFilePath is only relevant if logTraceLogMode is set to file
[constr_3493]	Applicable attributes for standardized E2E Profiles
[constr_3552]	Full mapping of target event
[constr_3554]	E2E protection configuration check
[constr_5052]	ProvidedSomeipServiceInstances of the same serviceInterface on one Machine

Table F.37: Changed Constraints in R20-11

F.7.6 Deleted Constraints in R20-11

Number	Heading
[constr_1481]	Usage of DataPrototypeInServiceInterfaceRef in the *AUTOSAR adaptive platform*
[constr_1497]	Attribute optionKind set to commandLineSimpleForm
[constr_1498]	Attribute optionKind set to commandLineShortForm or commandLineLongForm
[constr_1524]	Standardized values of PersistencyFileProxyInterface.category
[constr_1534]	Existence of DiagnosticSoftwareClusterProps
[constr_1542]	No nested definition of SoftwareCluster
[constr_1561]	Existence of SoftwareClusterDesign.subSoftwareCluster and SoftwareClusterDesign.dependsOn.dependentSoftwareClusterDesign
[constr_1563]	Standardized values of SoftwareClusterDesign.category and SoftwareCluster.category
[constr_1565]	Existence of SoftwareCluster.subSoftwareCluster
[constr_1567]	Existence of SoftwareCluster.subSoftwareCluster and SoftwareCluster.dependsOn/conflictsTo
[constr_1615]	Existence of attribute SomeipDataPrototypeTransformationProps.someipTransformationProps.sessionHandling
[constr_1687]	Definition of machine state
[constr_1703]	Target SwcServiceDependency of DiagnosticGenericUdsPortMapping.swcServiceDependencyInExecutable
[constr_1704]	Target SwcServiceDependency of DiagnosticUploadDownloadPortMapping.swcServiceDependencyInExecutable

▽

△

Number	Heading
[constr_1705]	Target of reference SoftwareActivationDependencyCompareCondition.softwareActivationDependency
[constr_1709]	Applicability of attribute PersistencyRedundancyEnum.redundantPerKey
[constr_1730]	Restriction regarding the modeling of the PhmRecoveryActionInterface.recovery
[constr_3296]	Transport Protocol attributes defined for a RequiredSomeipServiceInstance
[constr_3297]	SomeipServiceInstanceToMachineMapping only supports a single Address Family
[constr_3307]	SomeipEventDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3412]	OsModuleInstantiation shall have at least one ResourceGroup
[constr_3527]	PhmLogicalExpression referenced by one PhmRule
[constr_3543]	At least one transportPlugin definition required for each DdsProvidedServiceInstance
[constr_3544]	At least one transportPlugin definition required for each DdsRequiredServiceInstance
[constr_3556]	Unique transport layer mapping

Table F.38: Deleted Constraints in R20-11

F.8 Traceable item history of this document according to AUTOSAR Release R21-11

F.8.1 Added Specification Items in R21-11

Number	Heading
[TPS_MANI_01343]	Relation between Function Group states and NmNetworkHandle
[TPS_MANI_01344]	Actions taken after installation of a SoftwarePackage
[TPS_MANI_01345]	Ability to attach checksums to SoftwareCluster
[TPS_MANI_01346]	No formal definition of checksum algorithm
[TPS_MANI_01347]	Definition of a DiagnosticDataElement used in the context of a DID obtained from a PPortPrototype typed by a DiagnosticDataElementInterface
[TPS_MANI_01348]	Definition of a DiagnosticDataElement used in the context of a DID obtained from a PPortPrototype typed by a DiagnosticDataIdentifierInterface
[TPS_MANI_01349]	Configuration of diagnostic-related properties of a SoftwareCluster
[TPS_MANI_01350]	Semantics of DiagnosticServiceValidationConfiguration
[TPS_MANI_01351]	Reporting the status of a DiagnosticEvent on the *AUTOSAR adaptive platform*

▽

△

Number	Heading
[TPS_MANI_01352]	Definition of DiagnosticServiceValidationMapping
[TPS_MANI_01353]	Semantics of DiagnosticExternalAuthenticationInterface
[TPS_MANI_01354]	Rationale for the existence of meta-class AbstractRawDataStreamEthernetCredentials
[TPS_MANI_01355]	Definition of local Ethernet credentials
[TPS_MANI_01356]	Definition of remote server's Ethernet credentials
[TPS_MANI_01357]	Definition of remote client's Ethernet credentials
[TPS_MANI_01358]	Restriction for the configuration of diagnostic debouncing
[TPS_MANI_01359]	Semantics of DiagnosticAuthenticationInterface
[TPS_MANI_01360]	Creation of two diagnostic mappings the fulfill different roles in the context of authenticating a diagnostic client
[TPS_MANI_01361]	Support the authentication request of a diagnostic client
[TPS_MANI_01362]	Support the responding to an authentication request of a diagnostic client
[TPS_MANI_01363]	Semantics of DiagnosticComControlInterface
[TPS_MANI_02384]	DltLogSink with category DLT_LOGSINK_REMOTE
[TPS_MANI_02385]	DltLogSink with category DLT_LOGSINK_DLT
[TPS_MANI_02386]	DltLogSink with category DLT_LOGSINK_FILE
[TPS_MANI_02387]	DltLogSink with category DLT_LOGSINK_CONSOLE
[TPS_MANI_02388]	Semantics of DltLogSinkToPortPrototypeMapping
[TPS_MANI_03277]	ServiceInterfaceElementMappings for a subset of elements of a single ServiceInterface
[TPS_MANI_03278]	Usage of ApSomeipTransformationProps.byteOrder
[TPS_MANI_03279]	Priority of Nm messages
[TPS_MANI_03280]	Semantics of ApApplicationEndpoint
[TPS_MANI_03281]	Port specific TCP configuration settings
[TPS_MANI_03282]	Assignment of a Dlt Ecu Identifier to a LogAndTraceInstantiation
[TPS_MANI_03283]	Standardized values for attribute DltLogSink.category
[TPS_MANI_03284]	Semantics of LogAndTraceInterface
[TPS_MANI_03285]	Semantics of PortPrototype.logAndTraceMessageCollectionSet
[TPS_MANI_03286]	Assignment of DltApplication to a Process
[TPS_MANI_03287]	ServiceInterface.trigger to ISignalTriggering mapping
[TPS_MANI_03288]	ApSomeipTransformationProps for triggers
[TPS_MANI_03289]	Semantics of ServiceInterfaceTriggerMapping
[TPS_MANI_03290]	Semantics of ComTriggerGrantDesign
[TPS_MANI_03291]	Semantics of ServiceInterface.trigger
[TPS_MANI_03633]	Semantics of several supervisionMode references for one supervision
[TPS_MANI_03635]	Determination of translation direction
[TPS_MANI_03636]	Definition of ServiceInterfaceElementSecureComConfig in the context of signal/service translation

▽

△

Number	Heading
[TPS_MANI_03637]	Ignored attributes of `ServiceInterfaceElementSecureComConfig` in the context of `signal/service translation`
[TPS_MANI_03638]	Mapping of `MultiplexedIPdu`
[TPS_MANI_03639]	Mapping of `MultiplexedIPdu` static part
[TPS_MANI_03640]	Mapping of `MultiplexedIPdu` dynamic part
[TPS_MANI_03641]	No mapping of `MultiplexedIPdu` selector field value
[TPS_MANI_03642]	Exactly one alternative defined for mapped event / field during runtime
[TPS_MANI_03643]	Translation-`Executable category`
[TPS_MANI_03644]	Supported reception data filters
[TPS_MANI_03645]	Applicability of a filter to the `scoped filtered element`
[TPS_MANI_03646]	Definition of transmission triggers for `signal-service-translation`
[TPS_MANI_03647]	Full translation before transmission triggering
[TPS_MANI_03648]	Transmission trigger for `signal-service-translation`
[TPS_MANI_03649]	No transmission trigger for `signal-service-translation`
[TPS_MANI_03650]	Definition of `DdsServiceInstanceResourceIdentifierTypeEnum`
[TPS_MANI_03651]	Mandatory `SupervisionCheckpoint.process` reference
[TPS_MANI_03652]	`Signal/service translation` header format
[TPS_MANI_03653]	`Signal/service translation` header endianness
[TPS_MANI_03654]	Definition of `Partial SOME/IP Header` ID
[TPS_MANI_03655]	Definition of `No Message Header` - implicit SOME/IP Message Identification
[TPS_MANI_03656]	Definition of `No Message Header` - implicit Arbitrary Message Identification
[TPS_MANI_03657]	`Signal-service-translation` SOME/IP Service Discovery Find
[TPS_MANI_03658]	`Service-signal-translation` SOME/IP Service Discovery Offer
[TPS_MANI_03659]	No `signal-service-translation` SOME/IP Service Discovery Find
[TPS_MANI_03660]	No `signal-service-translation` SOME/IP Service Discovery Offer
[TPS_MANI_03661]	Configuration of Governance in DDS Security
[TPS_MANI_03662]	Configuration of Topic access rules

Table F.39: Added Specification Items in R21-11

F.8.2 Changed Specification Items in R21-11

Number	Heading
[TPS_MANI_01001]	Meaning of `ServiceInterface`
[TPS_MANI_01048]	Retrieving the status of a `DiagnosticEvent` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*

▽

△

Number	Heading
[TPS_MANI_01049]	Mapping of DiagnosticOperationCycle to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01050]	Mapping of DiagnosticEnableCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01111]	Diagnostic Address of a SoftwareCluster
[TPS_MANI_01176]	Standardized value for attribute CppImplementationDataType.typeEmitter
[TPS_MANI_01239]	Semantics of meta-class ComEventGrant
[TPS_MANI_01244]	Semantics of DiagnosticDataElementInterface
[TPS_MANI_01259]	Mapping of DiagnosticClearCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01260]	Mapping of DiagnosticIndicator to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01261]	Mapping of DiagnosticMemoryDestination to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01262]	Mapping of DiagnosticSecurityLevel to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01263]	Mapping of DiagnosticDataIdentifier or DiagnosticDataElement to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01274]	Standardized System Category Definitions for the Adaptive Platform
[TPS_MANI_01287]	Semantics of RawDataStreamMapping
[TPS_MANI_01326]	Generic Mapping to a DiagnosticServiceInstance on the AUTOSAR Adaptive Platform
[TPS_MANI_01331]	Standardized values of attribute SoftwareCluster.category
[TPS_MANI_03007]	Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03008]	Tcp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03010]	Udp Transport Protocol Configuration in case of IP-Multicast
[TPS_MANI_03038]	Purpose of ServiceEventDeployment
[TPS_MANI_03101]	SOME/IP serialization
[TPS_MANI_03109]	TransformationProps on the level of DataPrototypes overwrites TransformationProps settings on the level of a ServiceInterface
[TPS_MANI_03114]	Usage of AssemblySwConnectors in the System Design model
[TPS_MANI_03130]	Standardized values of the attribute E2EProfileConfiguration.profileName
[TPS_MANI_03131]	Non-Standardized values of attribute E2EProfileConfiguration.profileName
[TPS_MANI_03163]	Network configuration for Log and Trace messages
[TPS_MANI_03165]	Network Interface configuration for DoIP
[TPS_MANI_03185]	Structure of a CppImplementationDataType of category ASSOCIATIVE_MAP

▽

△

Number	Heading
[TPS_MANI_03225]	References to VLANs in `PncMapping`
[TPS_MANI_03227]	Usage of ephemeral ports
[TPS_MANI_03252]	Usage of same `End2EndMethodProtectionProps.sourceId` in case of Multi-Binding
[TPS_MANI_03274]	Configuration of log and trace sinks
[TPS_MANI_03276]	Semantics of `PersistencyDeploymentToDltLogSinkMapping`
[TPS_MANI_03512]	Applicability of global supervision without `Process` context
[TPS_MANI_03513]	Collection of supervisions into a `GlobalSupervision`
[TPS_MANI_03514]	Expiration tolerance for `SupervisionMode`
[TPS_MANI_03515]	Failure tolerance for `AliveSupervision`
[TPS_MANI_03527]	Definition of `DdsProvidedServiceInstance`
[TPS_MANI_03529]	Definition of `DdsRequiredServiceInstance`
[TPS_MANI_03577]	Definition of `Arbitrary Message Header` ID
[TPS_MANI_03585]	Processing order of COM-Stack features
[TPS_MANI_03589]	Reception data filter of COM-Stack
[TPS_MANI_03590]	Transfer properties and transmission modes of COM-Stack
[TPS_MANI_03592]	`ISignal` invalidation of COM-Stack
[TPS_MANI_03593]	`handleInvalid` = `dontInvalidate` behavior of COM-Stack
[TPS_MANI_03594]	`handleInvalid` = `replace` behavior of COM-Stack
[TPS_MANI_03597]	Support for `MultiplexedIPdu`
[TPS_MANI_03600]	`Signal-service-translation` of E2E protected payload
[TPS_MANI_03601]	`Signal-service-translation` of E2E protected payload - timeout handling
[TPS_MANI_03602]	`Signal-service-translation` of E2E protected payload - error handling
[TPS_MANI_03603]	`Service-signal-translation` of E2E protected payload
[TPS_MANI_03604]	`Service-signal-translation` of E2E protected payload - timeout handling
[TPS_MANI_03605]	`Service-signal-translation` of E2E protected payload - error handling
[TPS_MANI_03607]	Handling of safe `signal/service translation` in one `Executable`
[TPS_MANI_03608]	Support for safe `signal/service translation`
[TPS_MANI_03609]	Support for safe `signal/service translation` with same or different E2E profiles
[TPS_MANI_03621]	Data filtering during the `signal-service-translation`
[TPS_MANI_03627]	No `signal/service translation` for methods

Table F.40: Changed Specification Items in R21-11

F.8.3 Deleted Specification Items in R21-11

Number	Heading
[TPS_MANI_01014]	Semantics of meta-class `StartupConfigSet`
[TPS_MANI_01037]	Diagnostic data mapping on the *AUTOSAR adaptive platform*
[TPS_MANI_01054]	Definition of the queue length of an `event` or `field` notifier
[TPS_MANI_01060]	Use cases for the application of `DiagnosticServiceDataMapping`
[TPS_MANI_01103]	Three-level approach to REST modeling
[TPS_MANI_01105]	Semantics of `RestServiceInterface`
[TPS_MANI_01120]	Recursive definition of `RestResourceDef`
[TPS_MANI_01121]	Semantics of `RestResourceDef.endpoint`
[TPS_MANI_01122]	Arguments to endpoints
[TPS_MANI_01123]	System Triggered Event
[TPS_MANI_01124]	Semantics of `RestElementDef`
[TPS_MANI_01125]	Properties of REST elements can either be primitive or have array semantics
[TPS_MANI_01126]	Definition of string properties
[TPS_MANI_01127]	Limited support for data semantics in `RestAbstractNumericalPropertyDef`
[TPS_MANI_01128]	Difference between `RestIntegerPropertyDef` and `RestNumberPropertyDef`
[TPS_MANI_01129]	`RestObjectRef` is only needed for specific implementations of REST-based communication
[TPS_MANI_01130]	Structure of a typical `URI` for a `REST` service
[TPS_MANI_01131]	Impact of nested `REST` resources on the structure of `REST` URI
[TPS_MANI_01178]	Semantics of `RestHttpPortPrototypeMapping.acceptEncoding`
[TPS_MANI_01256]	`AdaptiveApplicationSwComponentType` offers a `PPortPrototype` typed by `DiagnosticIndicatorInterface`
[TPS_MANI_01257]	`AdaptiveApplicationSwComponentType` offers a PPortPrototype typed by DiagnosticConditionInterface
[TPS_MANI_01258]	`AdaptiveApplicationSwComponentType` offers a `PPortPrototype` typed by `DiagnosticGenericUdsInterface`
[TPS_MANI_01272]	Duplicate entries in `logTrageLogMode`
[TPS_MANI_01288]	Impact of the `SoftwarePackage` on the value of function group states on the target platform
[TPS_MANI_01289]	Order of function group states is relevant
[TPS_MANI_01303]	Semantics of reference `SoftwarePackageStep.preActivate`
[TPS_MANI_01304]	Semantics of reference `SoftwarePackageStep.verify`
[TPS_MANI_01311]	Handling of manufacturer checks
[TPS_MANI_01312]	Handling of supplier checks
[TPS_MANI_01336]	Two use cases for using the `DiagnosticEventPortMapping`
[TPS_MANI_03075]	Byte Order of chosen DataPrototype in the serialized data stream



△

Number	Heading
[TPS_MANI_03160]	Further configuration options in `DltLogChannel`
[TPS_MANI_03275]	Configuration of log and trace message source on design level
[TPS_MANI_03552]	Supervision cycle for `GlobalSupervision`
[TPS_MANI_03584]	Definition of transmission triggers for translations with different sources
[TPS_MANI_03586]	No transmission trigger for translations with different sources
[TPS_MANI_03587]	Transmission trigger for translations with different sources
[TPS_MANI_03588]	Full translation before transmission triggering
[TPS_MANI_03596]	Update Bit support for `ISignalIPdu`

Table F.41: Deleted Specification Items in R21-11

F.8.4 Added Constraints in R21-11

Number	Heading
[constr_3623]	`SupervisionCheckpoint`s in the context of a `GlobalSupervision`
[constr_3624]	At least one Supervision defined in the context of a `GlobalSupervision`
[constr_3625]	`DeadlineSupervision` referencing `CheckpointTransition` in the context of a `GlobalSupervision`
[constr_3626]	`LogicalSupervision` referencing `CheckpointTransition` in the context of a `GlobalSupervision`
[constr_3627]	Existence of `SupervisionModeCondition.stateReference`
[constr_3628]	Reference to `Function Group State` from a `SupervisionModeCondition`
[constr_3629]	Identical `Function Group` in the scope of a `GlobalSupervision`
[constr_3630]	`GlobalSupervision` and `Process` relation
[constr_3631]	Global supervision restricted to one `Function Group`
[constr_3632]	Supervision of a Supervised Entity Instance in the scope of a `Function Group State`
[constr_3633]	Mandatory attributes of `AliveSupervision`
[constr_3634]	Multiplicity of `CheckpointTransition.source` and `CheckpointTransition.target`
[constr_3635]	Mandatory attributes of `DeadlineSupervision`
[constr_3636]	Consistent `ISignal` communication direction `in` and `RequiredApServiceInstance`
[constr_3637]	Consistent `ISignal` communication direction `out` and `ProvidedApServiceInstance`
[constr_3639]	Existence of `SupervisionMode.expiredSupervisionTolerance`
[constr_3640]	Existence of `SupervisionMode.modeCondition`

▽

△

Number	Heading
[constr_3641]	Allowed combinations of ServiceInterfaceDeployment, AdaptivePlatformServiceInstance, ServiceInstanceToMachineMapping
[constr_3642]	Restriction of aggregation of PortPrototypeProps to the Adaptive Platform
[constr_3643]	No filter support for service-signal-translation direction
[constr_3644]	No transmissionTrigger support for service-signal-translation direction
[constr_3645]	discoveryType mandatory for DdsProvidedServiceInstance
[constr_3646]	resourceIdentifierType mandatory for DdsProvidedServiceInstance
[constr_3647]	resourceIdentifierType value for USER_DATA QoS-based discovery
[constr_3648]	discoveryType mandatory for DdsRequiredServiceInstance
[constr_3649]	Consistent SupervisionCheckpoint.process reference
[constr_3650]	headerId required in case of Arbitrary Message Header
[constr_3674]	Existence of NoSupervision.targetPhmSupervisedEntity
[constr_3675]	Existence of NoSupervision.process
[constr_3676]	Exclusive usage of NoSupervision
[constr_3677]	ComGrants referencing DDS Service Instances
[constr_3678]	Existence of attributes for DdsSecureComProps
[constr_3679]	Existence of attributes for DdsSecureGovernance
[constr_3680]	Existence of attributes for DdsTopicAccessRule
[constr_3681]	Supported values of DdsTopicAccessRule.dataProtectionKind
[constr_3682]	Values of DdsDomainRange.min and DdsDomainRange.max
[constr_3683]	Attributes referencing DdsTopicAccessRule
[constr_3684]	Mutual exclusivity of Secure Communication Properties
[constr_5250]	Protection of AdaptivePlatformServiceInstances of the same ServiceInterfaceDeployment
[constr_5260]	UDP endpoint using DTLS CLIENT role can only serve required service instances
[constr_5261]	TCP endpoint using TLS CLIENT role can only serve required service instances
[constr_5275]	Existence of LogAndTraceInstantiation.dltEcu
[constr_5276]	Existence of LogAndTraceInstantiation.logSink
[constr_5277]	applicable DltLogSink categories
[constr_5278]	DltLogSink with category DLT_LOGSINK_REMOTE is only allowed to be referenced by DltLogSinkToPortPrototypeMapping
[constr_5279]	DltLogSink with category DLT_LOGSINK_DLT is only allowed to be referenced by LogAndTraceInstantiation
[constr_5280]	Existence of DltLogSink.defaultLogThreshold
[constr_5281]	Existence of DltLogSink.defaultTraceState
[constr_5282]	Existence of DltLogSinkToPortPrototypeMapping.process
[constr_5283]	Existence of DltLogSinkToPortPrototypeMapping.dltLogSink
[constr_5284]	Existence of DltLogSinkToPortPrototypeMapping.dltContext

▽

△

Number	Heading
[constr_5285]	Existence of PortPrototype references in DltLogSinkToPortPrototypeMapping
[constr_5286]	Restriction applicable for DltLogSinkToPortPrototypeMapping.rPortPrototype
[constr_5287]	Restriction applicable for DltLogSinkToPortPrototypeMapping.pPortPrototype
[constr_5288]	Existence of process reference in DltApplicationToProcessMapping
[constr_5289]	Existence of dltApplication reference in DltApplicationToProcessMapping
[constr_5290]	PPortPrototype is not allowed to be typed by LogAndTraceInterface
[constr_5291]	Allowed usage of PortPrototype.logAndTraceMessageCollectionSet
[constr_5292]	Assigned dltSessionId shall be consistent for the same PortPrototype
[constr_5316]	Allowed ServiceEventDeployment.trigger references
[constr_5317]	ServiceEventDeployment not allowed to reference an event and a trigger at the same time
[constr_5318]	Existence of ServiceInstanceToSignalMapping for an trigger with signalBased serialization
[constr_10029]	ServiceInterfaceDeployment shall cover all elements of the corresponding ServiceInterfaceDeployment
[constr_10030]	Existence of DiagnosticDataIdentifierInterface.read
[constr_10031]	Existence of DiagnosticRoutineInterface.start
[constr_10035]	Completeness of the PersistencyDeployment.version
[constr_10037]	Existence of attribute TagWithOptionalValue.sequenceOffset in the context of attribute capabilityRecord owned by ProvidedSomeipServiceInstance, RequiredSomeipServiceInstance, SdServerConfig, SdClientConfig, or AbstractServiceInstance
[constr_10046]	Value of PersistencyRedundancyMOutOfN.n
[constr_10047]	Restriction for the applicability of DiagnosticMonitorPortMapping
[constr_10048]	Existence of reference from DiagnosticMonitorPortMapping to DiagnosticEvent
[constr_10049]	Restriction for the applicability of DiagnosticEventPortMapping
[constr_10050]	Restriction for the applicability of DiagnosticOperationCyclePortMapping
[constr_10051]	Existence of reference from DiagnosticOperationCyclePortMapping to DiagnosticOperationCycle
[constr_10052]	Restriction for the applicability of DiagnosticEnableConditionPortMapping
[constr_10053]	Existence of reference from DiagnosticEnableConditionPortMapping to DiagnosticEnableCondition
[constr_10054]	Restriction for the applicability of DiagnosticClearConditionPortMapping
[constr_10055]	Existence of reference from DiagnosticClearConditionPortMapping to DiagnosticClearCondition
[constr_10056]	Restriction for the applicability of DiagnosticIndicatorPortMapping
[constr_10057]	Restriction for the applicability of DiagnosticMemoryDestinationPortMapping

▽

△

Number	Heading
[constr_10058]	Restriction for the applicability of DiagnosticSecurityLevelPortMapping
[constr_10059]	Existence of reference from DiagnosticSecurityLevelPortMapping to DiagnosticSecurityLevel
[constr_10060]	PortInterface of PPortPrototype referenced by DiagnosticDataPortMapping
[constr_10061]	Mapping to DiagnosticDataIdentifierInterface, DiagnosticDataElementInterface, or DiagnosticDataIdentifierGenericInterface
[constr_10062]	DiagnosticServiceInstances that can be mapped by a DiagnosticServiceGenericMapping
[constr_10063]	Possible values for DiagnosticServiceValidationMapping.category
[constr_10064]	Existence of DiagnosticServiceValidationMapping.pPortPrototypeInExecutable
[constr_10065]	Validity of DiagnosticServiceValidationConfiguration.manufacturer-ValidationOrder
[constr_10066]	Validity of DiagnosticServiceValidationConfiguration.supplier-ValidationOrder
[constr_10069]	Existence of SoftwareClusterDiagnosticProps.powerDownTime
[constr_10070]	Value of RequiredSomeipServiceInstance.requiredServiceInstanceId
[constr_10076]	Existence of RawDataStreamEthernetUdpCredentials.udpPort
[constr_10077]	Existence of ipV4Address and ipV6Address within AbstractRawDataStreamEthernetCredentials
[constr_10078]	Existence of RawDataStreamEthernetTcpUdpCredentials.tcpPort and udpPort
[constr_10079]	Existence of EthernetRawDataStreamMapping.localTcpPort and localUdpPort
[constr_10080]	Existence of initial values for PersistencyFileElement
[constr_10081]	Existence of initial values in the definition of PersistencyDataRequiredComSpec
[constr_10082]	Existence of initial values for PersistencyFile
[constr_10083]	Existence of initial values for PersistencyKeyValuePair
[constr_10086]	Existence of unicastUdpCredentials and multicastCredentials in the context of a EthernetRawDataStreamServerMapping
[constr_10090]	Existence of ProcessToMachineMapping.persistencyCentralStorageURI
[constr_10092]	Restriction for the applicability of DiagnosticAuthenticationPortMapping
[constr_10093]	Existence of reference from DiagnosticAuthenticationPortMapping to DiagnosticAuthentication
[constr_10094]	Restriction for the applicability of DiagnosticExternalAuthenticationPortMapping
[constr_10095]	Existence of reference from DiagnosticExternalAuthenticationPortMapping to DiagnosticAuthentication

Table F.42: Added Constraints in R21-11

F.8.5 Changed Constraints in R21-11

Number	Heading
[constr_1482]	Mapping of service interfaces vs. mapping of service interface elements
[constr_1564]	Existence of SoftwareCluster.diagnosticProps.diagnosticAddress
[constr_1598]	Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to CppImplementationDataTypeElement
[constr_1601]	TlvDataIdDefinition referencing CppImplementationDataTypeElement
[constr_1604]	Completeness of the existence of a set of TlvDataIdDefinition.tlvImplementationDataTypeElements
[constr_1697]	Restriction for ClientServerOperation aggregated by a DiagnosticDataIdentifierInterface or DiagnosticDataElementInterface
[constr_1748]	Existence of references TlvDataIdDefinition.tlvArgument, TlvDataIdDefinition.tlvRecordElement, and TlvDataIdDefinition.tlvImplementationDataTypeElement
[constr_1751]	Value of PersistencyRedundancyMOutOfN.m
[constr_1764]	Counterpart of PhmCheckpoint
[constr_1770]	Value of ProvidedSomeipServiceInstance.serviceInstanceId
[constr_3288]	IP configuration restriction for unicastNetworkEndpoints
[constr_3395]	TransformationPropsToServiceInterfaceElementMapping is restricted to one single ServiceInterface
[constr_3421]	Fibex elements applicable for a System of category MACHINE_DESIGN_EXTRACT
[constr_3485]	UDP endpoint using DTLS SERVER role can only serve provided service instances
[constr_3486]	TCP endpoint using TLS SERVER role can only serve provided service instances
[constr_3493]	Applicable attributes for standardized E2E Profiles
[constr_3539]	Only one AliveSupervision per SupervisionCheckpoint
[constr_3540]	SupervisionCheckpoint in supervision graph
[constr_3541]	qosProfile mandatory for DdsProvidedServiceInstance
[constr_3542]	qosProfile mandatory for DdsRequiredServiceInstance
[constr_3555]	No support for useAsCryptographicIPdu is true
[constr_5047]	Supported values of TlsSecureComProps.category
[constr_5052]	ProvidedSomeipServiceInstances of the same serviceInterface on one Machine
[constr_5056]	Restriction of sub-class of CompositionSwComponentType.connector
[constr_5057]	PassThroughSwConnector and ServiceInterfaceMapping
[constr_5155]	SomeipServiceInstanceToMachineMapping only supports a single Address Family
[constr_10002]	Only one mapping per PortPrototype

▽

△

Number	Heading
[constr_10003]	Restriction for the existence of DiagnosticDataPortMapping.diagnosticDataIdentifier vs. DiagnosticDataPortMapping.diagnosticDataElement

Table F.43: Changed Constraints in R21-11

F.8.6 Deleted Constraints in R21-11

Number	Heading
[constr_1496]	DiagnosticServiceDataMapping.mappedApDataElement shall only refer to specific sub-classes of DataPrototype
[constr_1500]	Target SwcServiceDependency of DiagnosticEventPortMapping. swcServiceDependencyInExecutable
[constr_1501]	Target SwcServiceDependency of DiagnosticOperationCyclePortMapping. swcServiceDependencyInExecutable
[constr_1502]	Target SwcServiceDependency of DiagnosticEnableConditionPortMapping. swcServiceDependencyInExecutable
[constr_1569]	Restriction for the scope of RestHttpPortPrototypeMapping. acceptEncoding
[constr_1580]	Restriction for the usage of RestHttpPortPrototypeMapping. acceptEncoding
[constr_1698]	Target SwcServiceDependency of DiagnosticClearConditionPortMapping. swcServiceDependencyInExecutable
[constr_1699]	Target SwcServiceDependency of DiagnosticIndicatorPortMapping. swcServiceDependencyInExecutable
[constr_1700]	Target SwcServiceDependency of DiagnosticMemoryDestinationPortMapping. swcServiceDependencyInExecutable
[constr_1701]	Target SwcServiceDependency of DiagnosticSecurityLevelPortMapping. swcServiceDependencyInExecutable
[constr_1702]	Target SwcServiceDependency of DiagnosticServiceDataIdentifierPortMapping. swcServiceDependencyInExecutable
[constr_1733]	Simultaneous existence of SoftwarePackageStep.preActivate and SoftwarePackageStep.verify
[constr_1738]	Multiplicity of reference GlobalSupervision.localSupervision
[constr_1739]	Multiplicity of aggregation localSupervision.transition
[constr_1744]	Definition of process state In the context of the ExecutionDependency
[constr_1765]	Diagnostic Services eligible for DiagnosticServiceGenericMapping

▽

△

Number	Heading
[constr_1809]	Global supervision restricted to one function group
[constr_3366]	System `category` for a system design description with Adaptive Platform and Classic Platform content
[constr_3426]	The `logTraceFilePath` is mandatory in case that `logTrageLogMode` is set to `file`
[constr_3427]	The `logTraceFilePath` is only relevant if `logTrageLogMode` is set to `file`
[constr_3537]	`localSupervision` referenced once in the context of a `GlobalSupervision`
[constr_3562]	Existence of `NonqueuedReceiverComSpec.filter`
[constr_5243]	Restriction of `LogAndTraceInstantiation.dltEcuId` attribute value
[constr_10004]	Consistency of `DiagnosticServiceGenericMapping` for `PortPrototype` typed by `DiagnosticDataIdentifierGenericInterface`

Table F.44: Deleted Constraints in R21-11

F.9 Traceable item history of this document according to AUTOSAR Release R22-11

F.9.1 Added Specification Items in R22-11

Number	Heading
[TPS_MANI_01365]	Semantics of attribute `NmHandleToFunctionGroupStateMapping.mappingDirection` set to `functionGroupStateToNmHandle`
[TPS_MANI_01366]	Semantics of attribute `NmHandleToFunctionGroupStateMapping.mappingDirection` set to `nmHandleInactiveToFunctionGroupState`
[TPS_MANI_01367]	Semantics of attribute `NmHandleToFunctionGroupStateMapping.mappingDirection` set to `nmHandleActiveToFunctionGroupState`
[TPS_MANI_01369]	Semantics of reference `SoftwareClusterDesign.requiredARElement`
[TPS_MANI_01370]	Semantics of `ExecutableImplementationProps`
[TPS_MANI_01371]	`PersistencyDeployment.redundancyHandling` does not exist
[TPS_MANI_01372]	Properties of data obtained from the application software
[TPS_MANI_01373]	Semantics of `DiagnosticRequestFileTransferInterface`
[TPS_MANI_01374]	Semantics of `UcmToTimeBaseResourceMapping`
[TPS_MANI_01375]	Semantics of meta-class `UcmRetryStrategy`
[TPS_MANI_01376]	Semantics of meta-class `UcmMasterModuleInstantiation`
[TPS_MANI_01377]	Semantics of meta-class `UcmSubordinateModuleInstantiation`
[TPS_MANI_01378]	Semantics of meta-class `PersistencyKeyValueDataTypeMapping`

▽

△

Number	Heading
[TPS_MANI_01379]	Semantics of FunctionalClusterInteractsWithPersistency-DeploymentMapping
[TPS_MANI_01380]	Usage of PersistencyDeployment
[TPS_MANI_01381]	Semantics of meta-class ArtifactLocator
[TPS_MANI_01382]	Location of artifact that contains executable code represented by Executable
[TPS_MANI_01383]	Semantics of ExecutableLoggingImplementationProps
[TPS_MANI_01384]	Semantics of StateManagementRequestRule
[TPS_MANI_01387]	Semantics of meta-class StateManagementActionList
[TPS_MANI_01388]	Semantics of the meta-class StateManagementStateMachineActionItem
[TPS_MANI_01389]	Semantics of meta-class StateManagementSyncActionItem
[TPS_MANI_01390]	Semantics of meta-class StateManagementSetFunctionGroupStateActionItem
[TPS_MANI_01391]	Semantics of StateManagementStateRequest
[TPS_MANI_01392]	Semantics of StateManagementStateNotification
[TPS_MANI_01393]	Initialization of a data object typed by a CppImplementationDataType of category VARIANT
[TPS_MANI_01394]	Support for the authorization of SOVD clients
[TPS_MANI_01395]	Support for the proximity challenge in the context of SOVD
[TPS_MANI_01396]	Semantics of meta-class DiagnosticSovdLock
[TPS_MANI_01397]	Standardized values of DiagnosticDataIdentifier.category of the usage in SOVD context
[TPS_MANI_01398]	Definition of an *SOVD data group*
[TPS_MANI_01399]	Modeling of the configuration of Service-oriented Vehicle Diagnostics
[TPS_MANI_01400]	Secure communication between SOVD Gateway and SOVD Server
[TPS_MANI_01401]	Support for TLS in vehicle-internal SOVD communication
[TPS_MANI_01402]	Identification of the SOVD Server
[TPS_MANI_01403]	External communication of the SOVD gateway
[TPS_MANI_01404]	Secure communication between SOVD gateway and SOVD Client
[TPS_MANI_01405]	Semantics of sub-classes of SoftwareClusterDiagnosticAddress
[TPS_MANI_01406]	Semantics of DeterministicSyncMaster
[TPS_MANI_01407]	Semantics of abstract base class DeterministicSyncVerificationPolicy
[TPS_MANI_01408]	Semantics of reference DeterministicClient. deterministicSyncMaster
[TPS_MANI_01409]	Semantics of meta-class DeterministicSyncMasterToTimeBaseConsumerMapping
[TPS_MANI_01410]	Semantics of attribute SoftwareCluster.installationBehavior
[TPS_MANI_01835]	Semantics of StateManagementRequestTrigger.rule
[TPS_MANI_01836]	Semantics of StateManagementRequestError.rule

▽

△

Number	Heading
[TPS_MANI_01863]	Supported value range for attribute ApApplicationErrorDomain.value
[TPS_MANI_03292]	Semantic of SomeipServiceInstanceToMachineMapping.tcpPort
[TPS_MANI_03293]	Semantic of SomeipServiceInstanceToMachineMapping.udpPort
[TPS_MANI_03294]	Semantics of DoIpNetworkConfiguration.eidRetrieval
[TPS_MANI_03295]	Semantics of FirewallStateSwitchInterface.firewallStateMachine
[TPS_MANI_03296]	Machine-specific configuration settings for Firewall
[TPS_MANI_03297]	Semantics of AdaptiveFirewallToPortPrototypeMapping
[TPS_MANI_03298]	Semantics of StateDependentFirewall
[TPS_MANI_03299]	Semantics of FirewallRuleProps
[TPS_MANI_03300]	Semantics of FirewallRule
[TPS_MANI_03301]	Semantics of FirewallRule.refillAmount and FirewallRule.bucketSize
[TPS_MANI_03302]	Firewall rules on Data Link Layer
[TPS_MANI_03303]	Filtering of packets with a single MAC address and MAC address range
[TPS_MANI_03304]	Firewall rules on Network Layer
[TPS_MANI_03305]	Filtering of packets with a single IP address and IP address range
[TPS_MANI_03306]	Firewall rules on Transport Layer
[TPS_MANI_03307]	Byte-Pattern Firewall rule
[TPS_MANI_03308]	Deep inspection of SOME/IP SD messages
[TPS_MANI_03309]	Deep inspection of SOME/IP messages
[TPS_MANI_03310]	Deep inspection of DoIP messages
[TPS_MANI_03311]	Deep inspection of DDS messages
[TPS_MANI_03312]	Static configuration of remote peer addresses for a ProvidedSomeipServiceInstance
[TPS_MANI_03313]	Semantics of SomeipRemoteUnicastConfig.eventGroup reference
[TPS_MANI_03314]	Static configuration of a remote peer address for a RequiredSomeipServiceInstance
[TPS_MANI_03315]	Semantics of SomeipRemoteMulticastConfig
[TPS_MANI_03316]	Semantics of SomeipRemoteMulticastConfig.eventGroup
[TPS_MANI_03317]	MACsec configuration
[TPS_MANI_03318]	MAC Security Key Agreement Entity configuration
[TPS_MANI_03319]	Standardized values for the attribute cipherSuite of meta-class MacSecCipherSuiteConfig
[TPS_MANI_03320]	Semantics of MacSecCipherSuiteConfig.cipherSuitePriority
[TPS_MANI_03321]	Semantics of FirewallStateSwitchInterface
[TPS_MANI_03663]	Ignored references of ServiceInstanceToMachineMapping in the context of signal/service translation
[TPS_MANI_03664]	ServiceInterfaceElementSecureComConfig.securedRx-Verification

▽

△

Number	Heading
[TPS_MANI_03665]	ServiceInterface.method with fireAndForget equals true to ISignalTriggerings mapping
[TPS_MANI_03666]	Service Discovery for multi-event ISignalIPdus
[TPS_MANI_03667]	Mapping from one ISignalIPdu to several events in case of signal-service-translation
[TPS_MANI_03668]	Supported type of Ethernet tunneling through CAN XL on Adaptive Platform
[TPS_MANI_03669]	Ethernet tunneling through CAN XL
[TPS_MANI_03670]	Definition of NoCheckpointSupervision
[TPS_MANI_03671]	Periodic storage of time base value
[TPS_MANI_03672]	Definition of RecoveryNotification for an HealthChannel
[TPS_MANI_03673]	RecoveryNotification referenced by several HealthChannel.recoveryNotifications

Table F.45: Added Specification Items in R22-11

F.9.2 Changed Specification Items in R22-11

Number	Heading
[TPS_MANI_01111]	Diagnostic Address of a SoftwareCluster
[TPS_MANI_01114]	Relation of DiagnosticContributionSet to SoftwareCluster
[TPS_MANI_01164]	Semantics of SoftwareCluster.dependsOn
[TPS_MANI_01179]	Semantics of PersistencyFileElement.contentUri/PersistencyFile.contentUri vs. PersistencyDeployment.deploymentUri and PersistencyFileElement.fileName/PersistencyFile.fileName
[TPS_MANI_01203]	Semantics of DeterministicClient
[TPS_MANI_01226]	Machine-specific configuration settings for the UCM module
[TPS_MANI_01255]	Semantics of DoIP DiagnosticPortInterfaces
[TPS_MANI_01269]	Specification of boundaries for resource consumption
[TPS_MANI_01295]	Semantics of VehicleRolloutStep
[TPS_MANI_01297]	Semantics of meta-class UcmStep
[TPS_MANI_01349]	Configuration of diagnostic-related properties of a SoftwareCluster
[TPS_MANI_01360]	Creation of two diagnostic mappings the fulfill different roles in the context of authenticating a diagnostic client
[TPS_MANI_01361]	Support the authentication request of a diagnostic client
[TPS_MANI_01362]	Convey the Authentication state of a diagnostic client to the diagnostic server instance
[TPS_MANI_03013]	Repetition Phase configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03014]	Main Phase configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03015]	TTL for Offer Service Entries

▽

△

Number	Heading
[TPS_MANI_03027]	Repetition Phase configuration for a `RequiredSomeipServiceInstance`
[TPS_MANI_03115]	Mapping between a fire and forget `method` and elements of Classic Platform `PortInterface`s
[TPS_MANI_03178]	`StdCppImplementationDataType` of `category STRING`
[TPS_MANI_03285]	Semantics of `LTMessageCollectionToPortPrototypeMapping`
[TPS_MANI_03505]	Existence of `SupervisionCheckpoint`
[TPS_MANI_03546]	Definition of reported health status `RPortPrototype`
[TPS_MANI_03618]	Usage of `RequiredSomeipServiceInstance.blocklistedVersion`
[TPS_MANI_03654]	Definition of `Partial SOME/IP Header` ID
[TPS_MANI_03659]	No `signal-service-translation` SOME/IP Service Discovery Find
[TPS_MANI_03660]	No `signal-service-translation` SOME/IP Service Discovery Offer

Table F.46: Changed Specification Items in R22-11

F.9.3 Deleted Specification Items in R22-11

Number	Heading
[TPS_MANI_01100]	Semantics of `Allocator`
[TPS_MANI_01113]	Semantics of `SoftwareClusterDesign.diagnosticAddress`
[TPS_MANI_01161]	Impact of values of `category` on the semantics of `SoftwareClusterDesign`
[TPS_MANI_01271]	Semantics of `Executable.loggingBehavior`
[TPS_MANI_01305]	Semantics of attribute `SoftwarePackageStep.activationSwitch`
[TPS_MANI_01310]	Semantics of `SoftwareClusterDesign.dependsOn`
[TPS_MANI_01335]	Semantics of `SoftwareClusterDependencyFormula.category`
[TPS_MANI_03028]	TTL for Find Service Entries
[TPS_MANI_03126]	`ServiceInterface.field` mapping to `ISignalTriggering`s
[TPS_MANI_03506]	Optionality of `SupervisionCheckpoint`
[TPS_MANI_03623]	Usage of `checkpointId` in application code

Table F.47: Deleted Specification Items in R22-11

F.9.4 Added Constraints in R22-11

Number	Heading
[constr_3690]	DdsServiceInterfaceDeployment.serviceInterfaceId value shall not conflict with topic-based service discovery
[constr_3691]	Existence of ServiceInterfaceElementSecureComConfig.securedRxVerification
[constr_3692]	DataPrototypeInServiceInterfaceInstanceRef.targetDataPrototype in the context of a SignalBasedFireAndForgetMethodToISignalTriggeringMapping
[constr_3693]	EthernetCommunicationConnector.category is set to CAN_XL
[constr_3694]	Existence of canXlConfig vs. canXlConfigReqs
[constr_3709]	AliveSupervision.terminatingCheckpoint required for self terminating Processes
[constr_3710]	Process referenced by AliveSupervision.terminatingCheckpoint
[constr_3711]	AliveSupervision.terminatingCheckpointTimeoutUntilTermination
[constr_3712]	Exclusive usage of NoCheckpointSupervision
[constr_3715]	Reference in the role SomeipEventGroup.event
[constr_3719]	RecoveryNotification referenced either by HealthChannelExternalStatus or HealthChannelSupervision
[constr_3720]	Upper multiplicity of reference in the role ComGrantDesign.remoteSubject
[constr_3721]	Upper multiplicity of reference in the role EthernetCommunicationConnector.unicastNetworkEndpoint
[constr_3722]	Upper multiplicity of reference in the role EthernetCommunicationConnector.canXlProps
[constr_3723]	Upper multiplicity of reference in the role MachineDesign.tcpIpProps
[constr_3724]	Upper multiplicity of reference in the role MachineDesign.tcpIpIcmpProps
[constr_3725]	Upper multiplicity of reference in the role MachineDesign.ethIpProps
[constr_3727]	Upper multiplicity of reference in the role SoftwareClusterDesign.intendedTargetMachine
[constr_3728]	Upper multiplicity of reference in the role networkInterface
[constr_3729]	Upper multiplicity of reference in the role LogAndTraceInstantiation.timeBaseResource
[constr_3730]	Upper multiplicity of reference in the role HealthChannel.recoveryNotification
[constr_3731]	Upper multiplicity of reference in the role ProcessDesign.executable
[constr_3732]	Upper multiplicity of reference in the role Process.executable
[constr_3733]	Upper multiplicity of aggregation in the role methodMapping
[constr_3734]	Upper multiplicity of reference in the role DoIpNetworkConfiguration.networkConfiguration
[constr_5324]	MachineDesign.communicationController aggregation restriction
[constr_5332]	Mandatory multicast endpoint in case of multicastThreshold different from 0

▽

△

Number	Heading
[constr_5333]	No multicast in case of TCP
[constr_5338]	ProvidedSomeipServiceInstance shall offer all SomeipEventGroups for subscription
[constr_5339]	SomeipEventGroups of a SomeipServiceInterfaceDeployment shall be referenced at most once from a RequiredSomeipServiceInstance that instantiates the SomeipServiceInterfaceDeployment
[constr_5343]	Usage of DoIpNetworkConfiguration.eidRetrieval
[constr_5347]	Supported value range for attribute SecOcSecureComProps.authenticationVerifyAttempts
[constr_5348]	Mandatory initialMode in ModeDeclarationGroup that is referenced by StateDependentFirewall
[constr_5349]	Mandatory defaultAction in StateDependentFirewall
[constr_5350]	Mandatory action in FirewallRuleProps
[constr_5351]	FirewallRule is allowed to aggregate at most one protocol subelement
[constr_5352]	DdsRule.submessageType value restriction
[constr_5353]	DdsRule.readerEntityId and DdsRule.writerEntityId value restriction
[constr_5355]	SomeipServiceInstanceToMachineMapping with configured remote peer addresses shall not mix ProvidedSomeipServiceInstances and RequiredSomeipServiceInstances
[constr_5356]	RequiredSomeipServiceInstance is allowed to have only a single statically configured remote peer as service provider
[constr_5357]	SomeipRemoteMulticastConfig shall only be used on required side
[constr_5358]	AdaptiveFirewallToPortPrototypeMapping.rPortPrototype restriction
[constr_6815]	Existence of CppTemplateArgument.templateType for CppImplementationDataType of category STRING
[constr_10098]	Relation of MachineDesign.pnResetTimer and UdpNmCluster.nmMsgCycleTime
[constr_10101]	Attribute NmHandleToFunctionGroupStateMapping.mappingDirection is set to nmHandleActiveToFunctionGroupState or nmHandleInactiveToFunctionGroupState
[constr_10102]	Existence of initial values for PersistencyKeyValuePair
[constr_10103]	Existence of initial values for PersistencyFile
[constr_10105]	Existence of UcmRetryStrategy.maximumNumberOfRetries
[constr_10106]	Existence of UcmRetryStrategy.retryIntervalTime
[constr_10107]	Existence of the attribute UcmMasterModuleInstantiation.blockInconsistent
[constr_10108]	Existence of the attribute UcmMasterModuleInstantiation.serviceBusy
[constr_10109]	Existence of the attribute UcmMasterModuleInstantiation.updateSessionRejected
[constr_10110]	Existence of UcmSubordinateModuleInstantiation on a Machine
[constr_10111]	Existence of attribute DiagnosticAuthentication.authenticationTimeout
[constr_10113]	Restriction for the existence of ExecutableLoggingImplementationProps

▽

△

Number	Heading
[constr_10114]	Existence of attributes of DiagnosticEnvDataElementCondition if the reference in the role pPortPrototype exists
[constr_10124]	Multiplicity of attribute ApplicationAssocMapDataType.key
[constr_10125]	Multiplicity of attribute ApplicationAssocMapDataType.value
[constr_10126]	Multiplicity of attribute ApplicationAssocMapElementValueSpecification.key
[constr_10127]	Multiplicity of attribute ApplicationAssocMapElementValueSpecification.value
[constr_10128]	Multiplicity of attribute CppImplementationDataTypeElementQualifier.typeReference
[constr_10129]	Multiplicity of attribute Field.hasGetter
[constr_10130]	Multiplicity of attribute Field.hasSetter
[constr_10131]	Multiplicity of attribute Field.hasNotifier
[constr_10132]	Multiplicity of attribute ApApplicationError.errorCode
[constr_10133]	Multiplicity of attribute ApApplicationErrorDomain.value
[constr_10134]	Multiplicity of reference in the role PortInterfaceToDataTypeMapping.dataTypeMappingSet
[constr_10135]	Multiplicity of reference in the role PortInterfaceToDataTypeMapping.portInterface
[constr_10136]	Multiplicity of reference in the role ServiceInterfaceMapping.compositeServiceInterface
[constr_10137]	Multiplicity of reference in the role ServiceInterfaceMapping.sourceServiceInterface
[constr_10138]	Multiplicity of reference in the role ServiceInterfaceEventMapping.sourceEvent
[constr_10139]	Multiplicity of reference in the role ServiceInterfaceEventMapping.targetEvent
[constr_10140]	Multiplicity of reference in the role ServiceInterfaceFieldMapping.sourceField
[constr_10141]	Multiplicity of reference in the role ServiceInterfaceFieldMapping.targetField
[constr_10142]	Multiplicity of reference in the role ServiceInterfaceMethodMapping.sourceMethod
[constr_10143]	Multiplicity of reference in the role ServiceInterfaceMethodMapping.targetMethod
[constr_10144]	Multiplicity of reference in the role PersistencyRedundancyChecksum.algorithmFamily
[constr_10145]	Multiplicity of reference in the role PersistencyRedundancyChecksum.length
[constr_10146]	Multiplicity of reference in the role PersistencyRedundancyMOutOfN.m
[constr_10147]	Multiplicity of reference in the role PersistencyRedundancyMOutOfN.n
[constr_10148]	Multiplicity of reference in the role PersistencyFileElement.contentUri
[constr_10149]	Multiplicity of reference in the role PersistencyFileElement.fileName

▽

△

Number	Heading
[constr_10150]	Multiplicity of reference in the role SynchronizedTimeBaseProviderInterface.timeBaseKind
[constr_10151]	Multiplicity of reference in the role PhmCheckpoint.checkpointId
[constr_10152]	Multiplicity of reference in the role FieldSenderComSpec.initValue
[constr_10153]	Multiplicity of reference in the role PersistencyDataRequiredComSpec.dataElement
[constr_10154]	Multiplicity of reference in the role ProcessDesignToMachineDesignMapping.processDesign
[constr_10155]	Multiplicity of reference in the role ComOfferServiceGrantDesign.providedServicePort
[constr_10156]	Multiplicity of reference in the role ComFindServiceGrantDesign. requiredServicePort
[constr_10157]	Multiplicity of reference in the role ComFieldGrantDesign.field
[constr_10158]	Multiplicity of reference in the role ComFieldGrantDesign.role
[constr_10159]	Multiplicity of reference in the role ComEventGrantDesign.event
[constr_10160]	Multiplicity of reference in the role ComTriggerGrantDesign.trigger
[constr_10161]	Multiplicity of reference in the role ComMethodGrantDesign.method
[constr_10162]	Multiplicity of reference in the role DiagnosticClearConditionPortMapping.clearCondition
[constr_10163]	Multiplicity of reference in the role DiagnosticIndicatorPortMapping.indicator
[constr_10164]	Multiplicity of reference in the role DiagnosticMemoryDestinationPortMapping.memoryDestination
[constr_10165]	Multiplicity of reference in the role DiagnosticDataPortMapping.process
[constr_10166]	Multiplicity of attribute DiagnosticProvidedDataMapping.dataProvider
[constr_10167]	Multiplicity of attribute SomeipServiceDiscovery.someipServiceDiscoveryPort
[constr_10169]	Multiplicity of reference in the role Machine.machineDesign
[constr_10170]	Multiplicity of attribute Machine.trustedPlatformExecutableLaunchBehavior
[constr_10171]	Multiplicity of attribute Machine.processor
[constr_10172]	Multiplicity of attribute Processor.core
[constr_10173]	Multiplicity of attribute ProcessorCore.coreId
[constr_10174]	Multiplicity of the reference in the role ProcessToMachineMapping.process
[constr_10175]	Multiplicity of attribute StateDependentStartupConfig.resourceGroup
[constr_10176]	Multiplicity of attribute StateDependentStartupConfig.startupConfig
[constr_10177]	Multiplicity of attribute PersistencyDeployment.updateStrategy
[constr_10178]	Multiplicity of the reference in the role PersistencyPortPrototypeToDeploymentMapping.process
[constr_10179]	Multiplicity of attribute PersistencyKeyValuePair.valueDataType
[constr_10180]	Multiplicity of the reference in the role PersistencyPortPrototypeToKey-ValueStorageMapping.keyValueStorage

▽

△

Number	Heading
[constr_10182]	Multiplicity of the reference in the role PersistencyPortPrototypeToFileStorageMapping.fileStorage
[constr_10183]	Multiplicity of attribute PersistencyFile.fileName
[constr_10184]	Multiplicity of the reference in the role SynchronizedTimeBaseConsumer.networkTimeConsumer
[constr_10185]	Multiplicity of the reference in the role SynchronizedTimeBaseProvider.networkTimeProvider
[constr_10186]	Multiplicity of attribute DoIpInstantiation.entityStatusMaxByteFieldUse
[constr_10187]	Multiplicity of attribute DoIpInstantiation.gidInvalidityPattern
[constr_10188]	Multiplicity of attribute DoIpInstantiation.logicalAddress
[constr_10189]	Multiplicity of attribute DoIpInstantiation.maxRequestBytes
[constr_10190]	Multiplicity of attribute DoIpInstantiation.vinInvalidityPattern
[constr_10191]	Multiplicity of attribute DoIpNetworkConfiguration.isActivationLineDependent
[constr_10192]	Multiplicity of attribute DoIpNetworkConfiguration.maxInitialVehicleAnnouncementTime
[constr_10193]	Multiplicity of attribute DoIpNetworkConfiguration.maxTesterConnections
[constr_10194]	Multiplicity of attribute DoIpNetworkConfiguration.networkInterfaceId
[constr_10195]	Multiplicity of attribute DoIpNetworkConfiguration.vehicleIdentificationSyncStatus
[constr_10197]	Multiplicity of attribute DoIpRequestConfiguration.endAddress
[constr_10198]	Multiplicity of attribute DoIpRequestConfiguration.requestType
[constr_10199]	Multiplicity of attribute DoIpRequestConfiguration.startAddress
[constr_10200]	Multiplicity of attribute UcmModuleInstantiation.identifier
[constr_10201]	Multiplicity of of the reference in the role ComGrant.serviceInstance
[constr_10202]	Multiplicity of attribute ComFieldGrant.role
[constr_10203]	Multiplicity of the reference in the role ComFieldGrant.serviceDeployment
[constr_10204]	Multiplicity of the reference in the role ComMethodGrant.serviceDeployment
[constr_10205]	Multiplicity of the reference in the role ComEventGrant.serviceDeployment
[constr_10206]	Multiplicity of the reference in the role ComOfferServiceGrant.serviceInstance
[constr_10207]	Multiplicity of the reference in the role CryptoProviderToPortPrototypeMapping.cryptoProvider
[constr_10208]	Multiplicity of the reference in the role CryptoProviderToPortPrototypeMapping.process
[constr_10209]	Multiplicity of the reference in the role CryptoKeySlotToPortPrototypeMapping.keySlot
[constr_10210]	Multiplicity of the reference in the role CryptoKeySlotToPortPrototypeMapping.process
[constr_10211]	Multiplicity of the reference in the role CryptoCertificateToCryptoKeySlotMapping.cryptoCertificate

▽

△

Number	Heading
[constr_10212]	Multiplicity of attribute SomeipServiceInterfaceDeployment.serviceInterfaceId
[constr_10213]	Multiplicity of attribute SomeipServiceInterfaceDeployment.serviceInterfaceVersion
[constr_10214]	Multiplicity of attribute SomeipEventGroup.eventGroupId
[constr_10215]	Multiplicity of attribute SomeipEventDeployment.eventId
[constr_10216]	Multiplicity of attribute SomeipEventDeployment.transportProtocol
[constr_10217]	Multiplicity of the attribute DdsServiceInterfaceDeployment.serviceInterfaceId
[constr_10218]	Multiplicity of reference in the role ProvidedSomeipServiceInstance.sdServerConfig
[constr_10219]	Multiplicity of attribute ProvidedSomeipServiceInstance.serviceInstanceId
[constr_10220]	Multiplicity of attribute SomeipProvidedEventGroup.multicastThreshold
[constr_10221]	Multiplicity of reference in the role RequiredSomeipServiceInstance.sdClientConfig
[constr_10222]	Multiplicity of the reference in the role SomeipRequiredEventGroup.sdClientEventGroupTimingConfig
[constr_10223]	Multiplicity of attribute DdsServiceInstanceProps.domainId
[constr_10224]	Multiplicity of reference in the role DdsEventQosProps.event
[constr_10225]	Multiplicity of reference in the role DdsFieldQosProps.field
[constr_10226]	Multiplicity of attribute E2EProfileConfiguration.profileName
[constr_10227]	Multiplicity of attribute SecOcJobRequirement.secOcJobSemantic
[constr_10228]	Multiplicity of attribute SignalBasedFieldToISignalTriggering-Mapping.dataPrototypeInServiceInterfaceRef
[constr_10229]	Multiplicity of reference in the role SignalBasedMethodToISignalTriggeringMapping.method
[constr_10230]	Multiplicity of attribute SignalServiceTranslationEventProps.safeTranslation
[constr_10231]	Multiplicity of attribute SignalServiceTranslationEventProps.secureTranslation
[constr_10232]	Multiplicity of reference in the role PersistencyDeploymentToCryptoKey-SlotMapping.persistencyDeployment
[constr_10233]	Multiplicity of the reference in the role SoftwareCluster.vendorSignature
[constr_10234]	Multiplicity of attribute SoftwareCluster.version
[constr_10235]	Multiplicity of attribute SoftwareCluster.vendorId
[constr_10236]	Multiplicity of attribute SoftwareClusterDiagnosticAddress.addressSemantics
[constr_10237]	Multiplicity of attribute SoftwareClusterDependencyCompareCondition.compareType
[constr_10240]	Multiplicity of attribute SoftwarePackage.actionType
[constr_10241]	Multiplicity of attribute SoftwarePackage.compressedSoftwarePackageSize

▽

△

Number	Heading
[constr_10242]	Multiplicity of attribute SoftwarePackage.minimumSupportedUcmVersion
[constr_10243]	Multiplicity of attribute SoftwarePackage.packagerId
[constr_10244]	Multiplicity of reference in the role SoftwarePackage.packagerSignature
[constr_10245]	Multiplicity of reference in the role SoftwarePackage.softwareCluster
[constr_10246]	Multiplicity of attribute SoftwarePackage.uncompressedSoftwareClusterSize
[constr_10247]	Multiplicity of reference in the role VehiclePackage.packagerSignature
[constr_10248]	Multiplicity of reference in the role UcmDescription.identifier
[constr_10249]	Multiplicity of reference in the role VehicleDriverNotification.approvalRequired
[constr_10250]	Multiplicity of reference in the role VehicleDriverNotification.notificationState
[constr_10251]	Multiplicity of the reference in the role ServiceFieldDeployment.field
[constr_10252]	Multiplicity of attribute SignalBasedEventElementToISignalTriggering-Mapping.dataPrototypeInServiceInterfaceRef
[constr_10253]	Multiplicity of attribute SoftwareClusterDependencyCompareCondition.considerBuildNumber
[constr_10254]	Multiplicity of attribute SoftwareClusterDependencyCompareCondition.version
[constr_10255]	Multiplicity of attribute SignalServiceTranslationProps.serviceControl
[constr_10256]	Multiplicity of reference in the role SoftwarePackageStoring.storing
[constr_10365]	Existence of PersistencyDeployment.deploymentUri
[constr_10366]	Possible multiplicities of PersistencyDeployment.deploymentUri
[constr_10367]	Condition for the multiplicity of attribute PersistencyDeployment.deploymentUri
[constr_10374]	Existence of the attribute UcmSubordinateModuleInstantiation.verifyUpdate
[constr_10375]	Existence of the attribute UcmSubordinateModuleInstantiation.prepareUpdate
[constr_10376]	Existence of the attribute UcmSubordinateModuleInstantiation.prepareRollback
[constr_10377]	Completeness of the modeling of PersistencyKeyValueDataTypeMapping
[constr_10378]	PersistencyKeyValueDataTypeMapping references AbstractImplementationDataType in the role currentDataType
[constr_10379]	PersistencyKeyValueDataTypeMapping references ApplicationDataType in the role currentDataType
[constr_10380]	Target of ArtifactLocator.representedModelElement
[constr_10381]	Existence of attribute ArtifactLocator.uri
[constr_10382]	Existence of attribute ArtifactLocator.representedModelElement
[constr_10384]	PortInterface used for trigger state requests
[constr_10385]	PortInterface used for error state requests

▽

△

Number	Heading
[constr_10386]	Existence of references StateManagementStateMachineActionItem.start and stop
[constr_10387]	Consistency of StateManagementSetFunctionGroupStateActionItem.portPrototype and StateManagementSetFunctionGroupStateActionItem.setFunctionGroupState
[constr_10388]	Restriction for a PortInterface used for state switch notifications
[constr_10389]	Existence of attribute StateManagementFunctionGroupSwitchNotificationInterface.modeGroup
[constr_10390]	Existence of attribute StateManagementStateRequest.stateRequestPort
[constr_10391]	Existence of attribute StateManagementStateNotification.notificationPort
[constr_10392]	Existence of attribute StateManagementRequestRule.formula
[constr_10393]	Existence of reference in the role StateManagementRequestRule.nextState
[constr_10394]	Existence of attribute StateManagementCompareCondition.compareType
[constr_10395]	Existence of attribute StateManagementCompareCondition.compareValue
[constr_10396]	Existence of reference in the role StateManagementTriggerCompareRule.assumedCurrentState
[constr_10397]	Existence of reference in the role StateManagementSetFunctionGroupStateActionItem.portPrototype
[constr_10398]	Existence of reference in the role StateManagementSetFunctionGroupStateActionItem.setFunctionGroupState
[constr_10399]	Allowed interval of the “index” field according to the initialization rule for data object typed by a CppImplementationDataType of category VARIANT
[constr_10400]	Existence of SovdServerInstantiation.componentQualifier
[constr_10401]	Existence of SovdGatewayLocalEndpointTcpConfig.tcpPort
[constr_10402]	Existence of SovdGatewayEthernetCredentials.ipv4Address vs. SovdGatewayEthernetCredentials.ipv6Address
[constr_10403]	Existence of SovdGatewayEthernetCredentials.udpPort
[constr_10404]	Existence of SoftwareClusterSovdAddress.componentQualifier
[constr_10405]	Existence of reference in the role StateManagementActionList.affectedState
[constr_10406]	Existence of DeterministicSyncMOutOfN.numberOfConnectedClients
[constr_10407]	Existence of DeterministicSyncMOutOfN.minimumNumberOfRequests
[constr_10408]	Existence of DeterministicSyncMasterToTimeBaseConsumerMapping.deterministicSyncMaster
[constr_10409]	Existence of DeterministicSyncMasterToTimeBaseConsumerMapping.timeBaseConsumer
[constr_10410]	Value of SoftwareCluster.installationBehavior for a SoftwareCluster of category PLATFORM_CORE
[constr_10411]	Existence of ExecutionDependency and references to Function Group States

Table F.48: Added Constraints in R22-11

F.9.5 Changed Constraints in R22-11

Number	Heading
[constr_1543]	Only one physical address per SoftwareCluster.diagnosticDeploymentProps
[constr_1554]	Restriction regarding attribute PersistencyKeyValuePair.initValue
[constr_1785]	Restriction regarding the reference into another SoftwareCluster
[constr_3305]	Value of attribute SomeipEventDeployment.eventId shall be unique
[constr_3371]	Mutually exclusive existence of FireAndForgetMethodMapping.dataElement reference and FireAndForgetMethodMapping.trigger reference
[constr_3376]	FireAndForgetMethodMapping shall reference only fire and forget methods
[constr_3558]	RequiredSomeipServiceInstance.blocklistedVersion is restricted to the usage of minorVersion
[constr_3632]	Supervision of a Supervised Entity Instance in the scope of a Function Group State
[constr_3639]	Existence of SupervisionMode.expiredSupervisionTolerance
[constr_3676]	Exclusive usage of NoSupervision
[constr_5035]	Compatibility of data types with category STRING
[constr_5291]	Allowed usage of LTMessageCollectionToPortPrototypeMapping.rPortPrototype
[constr_10002]	Only one mapping per PortPrototype
[constr_10062]	DiagnosticServiceInstances that can be mapped by a DiagnosticServiceGenericMapping
[constr_10069]	Existence of SoftwareClusterDiagnosticDeploymentProps.powerDownTime
[constr_10090]	Existence of ProcessToMachineMapping.persistencyCentralStorageURI

Table F.49: Changed Constraints in R22-11

F.9.6 Deleted Constraints in R22-11

Number	Heading
[constr_1535]	Existence of DiagnosticSoftwareClusterProps in the context of a DiagnosticContributionSet
[constr_1557]	Standardized values of SoftwareClusterDesign.category and SoftwareCluster.category
[constr_1558]	Existence of SoftwareClusterDesign.diagnosticAddress
[constr_1559]	Existence of SoftwareClusterDesign.subSoftwareCluster
[constr_1562]	Existence of SoftwareClusterDesign.diagnosticContribution
[constr_1564]	Existence of SoftwareCluster.diagnosticProps.diagnosticAddress
[constr_1568]	Existence of SoftwareCluster.diagnosticExtract
[constr_1627]	Supported value range for attribute ApApplicationErrorDomain.value

▽

△

Number	Heading
[constr_1674]	Supported encoding of StdCppImplementationDataType of category STRING
[constr_1707]	Eligible subclasses of HeapUsage in the context of StateDependentStartupConfig.resourceConsumption
[constr_1732]	Existence of attribute activationSwitch set to True in the context of the enclosing UcmStep
[constr_1734]	Restriction for attribute SoftwarePackageStep.activationSwitch
[constr_5046]	Usage of DoIpNetworkConfiguration.eidUseMac
[constr_5280]	Existence of DltLogSink.defaultLogThreshold
[constr_10010]	Usage of attribute category in a SoftwareClusterDependencyFormula
[constr_10011]	Definition of sub-software-cluster

Table F.50: Deleted Constraints in R22-11

F.9.7 Added Advisories in R22-11

Number	Heading
[advisory_01000]	Existence of the reference MethodMapping.clientServerOperation
[advisory_01001]	Existence of reference MethodMapping.method
[advisory_01002]	Existence of reference EventMapping.dataElement
[advisory_01003]	Existence of reference EventMapping.event
[advisory_01004]	Existence of reference FieldMapping.field
[advisory_01005]	Existence of references FieldMapping.getterOperation, setterOperation, and notifierDataElement
[advisory_01006]	Existence of reference FireAndForgetMethodMapping.method
[advisory_01007]	Existence of references FireAndForgetMethodMapping.dataElement and trigger
[advisory_01008]	Multiplicity of reference in the role RootSwClusterDesignComponentPrototype.applicationType
[advisory_01009]	PersistencyKeyValueDataTypeMapping.currentDataType shall refer to a data type used in the context of the PersistencyKeyValueStorageInterface
[advisory_02000]	Define a ServiceInstanceToSignalMapping per member of an AdaptivePlatformServiceInstance
[advisory_02001]	No explicit signal/service translation for ISignalGroups
[advisory_02002]	ServiceInterface.field mapping to ISignalTriggerings for primitive data type fields
[advisory_02003]	ServiceInterface.field mapping to ISignalTriggerings for composite data type fields

Table F.51: Added Advisories in R22-11

F.9.8 Changed Advisories in R22-11

none

F.9.9 Deleted Advisories in R22-11

none

F.10 Traceable item history of this document according to AUTOSAR Release R23-11

F.10.1 Added Specification Items in R23-11

Number	Heading
[TPS_MANI_01411]	Initialization of a DataPrototype typed by a StdCppImplementationDataType of category VALUE that represents a Boolean
[TPS_MANI_01412]	Default value for startup timeout
[TPS_MANI_01413]	Semantics of UploadableDeploymentElement
[TPS_MANI_01414]	Semantics of UploadableDesignElement
[TPS_MANI_01415]	Semantics of attribute DiagnosticMultipleResourcePortMapping.overrideId
[TPS_MANI_01416]	Reporting the status of multiple DiagnosticEvents in the context of one PortInterface on the *AUTOSAR adaptive platform*
[TPS_MANI_01417]	Retrieving the status of multiple DiagnosticEvents to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01418]	Mapping of multiple DiagnosticConditions to a PortPrototype on the *AUTOSAR adaptive platform*
[TPS_MANI_01419]	Semantics of NetworkManagementPortInterface
[TPS_MANI_01420]	Support for the transport of bulk data in the context of SOVD
[TPS_MANI_01421]	Support for service validation in the context of SOVD
[TPS_MANI_01422]	Support for update procedures in the context of SOVD
[TPS_MANI_01423]	Support for configuration in the context of SOVD
[TPS_MANI_01424]	Semantics of DiagnosticSovdProximityChallengePortMapping
[TPS_MANI_01425]	Semantics of DiagnosticSovdAuthorizationPortMapping
[TPS_MANI_01426]	Mapping of DiagnosticSovdBulkData to PortPrototype on the *AUTOSAR adaptive platform*
[TPS_MANI_01427]	Mapping of DiagnosticSovdUpdate to PortPrototype on the *AUTOSAR adaptive platform*
[TPS_MANI_01428]	Mapping of DiagnosticSovdServiceInstance to PortPrototype on the *AUTOSAR adaptive platform*

▽

△

Number	Heading
[TPS_MANI_01429]	Mapping of `DiagnosticSovdConfiguration` to `PortPrototype` on the *AUTOSAR adaptive platform*
[TPS_MANI_01430]	Mapping of `DiagnosticSovdConfigurationParameter` to `DiagnosticDataIdentifier` on the *AUTOSAR adaptive platform*
[TPS_MANI_01431]	Definition of SOVD service instance
[TPS_MANI_01432]	Semantics of `DiagnosticSovdMethodPrimitive`
[TPS_MANI_01433]	Semantics of attribute `OsModuleInstantiation.osArtiAdapterLaunchBehavior`
[TPS_MANI_01434]	Diagnostic Address defined in the context of a `DiagnosticCommonProps`
[TPS_MANI_01435]	Semantics of meta-class `DiagnosticExternalAuthenticationIdentification`
[TPS_MANI_01436]	Relation of `StateManagementStateMachineActionItem` to internal state machine instances
[TPS_MANI_01437]	Standardized values of attribute `StateManagementStateNotification.stateMachine.category`
[TPS_MANI_01438]	Existence of `StateManagementStateMachineActionItem.overrideInitialState`
[TPS_MANI_01439]	Semantics of meta-class `IdsmTimestampProviderInterface`
[TPS_MANI_01440]	Using the `IdsmTimestampProviderMapping` for the provision of timestamps in the context of the *Intrusion Detection System Management*
[TPS_MANI_01441]	Using the `IdsmContextProviderMapping` for the provision of context in the context of the *Intrusion Detection System Management*
[TPS_MANI_01442]	Semantics of meta-class `IdsmContextProviderInterface`
[TPS_MANI_01443]	Semantics of meta-class `StateManagementNmActionItem`
[TPS_MANI_01444]	Semantics of attribute `StateManagementSleepActionItem.sleepTime`
[TPS_MANI_01445]	Semantics of meta-class `StateManagementSleepActionItem`
[TPS_MANI_01446]	Semantics of meta-class `SmInteractsWithNmMapping`
[TPS_MANI_01447]	Semantics of meta-class `NmInteractsWithSmMapping`
[TPS_MANI_03322]	Semantics of meta-class `ComTriggerGrant`
[TPS_MANI_03323]	Support for TLS in DoIP communication
[TPS_MANI_03324]	Semantics of `FunctionalClusterToSecurityEventDefinitionMapping`
[TPS_MANI_03325]	`DltLogSink` with `category` `DLT_LOGSINK_ARTI`

Table F.52: Added Specification Items in R23-11

F.10.2 Changed Specification Items in R23-11

Number	Heading
[TPS_MANI_01109]	Semantics of `UploadablePackageElement`
[TPS_MANI_01119]	Reference to model elements from `SoftwareClusterDesign`
[TPS_MANI_01137]	Possible use cases for the usage of `DataPrototypeInServiceInterfaceRef`
[TPS_MANI_01157]	Semantics of `updateStrategy` on collection level
[TPS_MANI_01159]	Semantics of `updateStrategy` on element level
[TPS_MANI_01191]	Modeling of possible errors
[TPS_MANI_01210]	Default encoding for all `DataPrototype`s typed by `CppImplementationDataType` of `category STRING`
[TPS_MANI_01239]	Semantics of meta-class `ComEventGrant`
[TPS_MANI_01274]	Standardized System Category Definitions for the Adaptive Platform
[TPS_MANI_01347]	Definition of a `DiagnosticDataElement` used in the context of a DID obtained from a `PPortPrototype` typed by a `DiagnosticDataElementInterface`
[TPS_MANI_01348]	Definition of a `DiagnosticDataElement` used in the context of a DID obtained from a `PPortPrototype` typed by a `DiagnosticDataIdentifierInterface`
[TPS_MANI_01362]	Convey the Authentication state of a diagnostic client to the diagnostic server instance
[TPS_MANI_01397]	Standardized values of `DiagnosticDataIdentifier.category` of the usage in SOVD context
[TPS_MANI_01405]	Semantics of sub-classes of `SoftwareClusterDiagnosticAddress`
[TPS_MANI_03101]	SOME/IP serialization
[TPS_MANI_03102]	UserDefined serialization
[TPS_MANI_03103]	Default size for all array and map length fields
[TPS_MANI_03104]	Default size for all structure length fields
[TPS_MANI_03105]	Default size for all union length fields
[TPS_MANI_03106]	Default size for all union type selector fields
[TPS_MANI_03107]	Default alignment for all dynamic `DataPrototype`s
[TPS_MANI_03108]	Default Byte Order for all `DataPrototype`s
[TPS_MANI_03117]	Default size for all string length fields
[TPS_MANI_03283]	Standardized values for attribute `DltLogSink.category`
[TPS_MANI_03299]	Semantics of `FirewallRuleProps`
[TPS_MANI_03546]	Definition of reported health status `RPortPrototype`
[TPS_MANI_03628]	Standardized values of `ServiceInterface.category`

Table F.53: Changed Specification Items in R23-11

F.10.3 Deleted Specification Items in R23-11

Number	Heading
[TPS_MANI_01111]	Diagnostic Address of a SoftwareCluster
[TPS_MANI_01199]	Semantics of DeterministicClientResourceNeeds
[TPS_MANI_01200]	Semantics of meta-class DeterministicClientResource
[TPS_MANI_01203]	Semantics of DeterministicClient
[TPS_MANI_01233]	Semantics of meta-class ComFindServiceGrantDesign
[TPS_MANI_01241]	Semantics of meta-class ComFindServiceGrant
[TPS_MANI_01330]	Definition of machine Function Group
[TPS_MANI_01406]	Semantics of DeterministicSyncMaster
[TPS_MANI_01407]	Semantics of abstract base class DeterministicSyncVerificationPolicy
[TPS_MANI_01408]	Semantics of reference DeterministicClient. deterministicSyncMaster
[TPS_MANI_01409]	Semantics of meta-class DeterministicSyncMasterToTimeBaseConsumerMapping
[TPS_MANI_03125]	ServiceInterface.method to ISignalTriggerings mapping
[TPS_MANI_03235]	Usage of ApSomeipTransformationProps.sessionHandling
[TPS_MANI_03288]	ApSomeipTransformationProps for triggers

Table F.54: Deleted Specification Items in R23-11

F.10.4 Added Constraints in R23-11

Number	Heading
[constr_5366]	Allowed target of ComEventGrant.serviceDeployment reference
[constr_5367]	Allowed target of ComTriggerGrant.serviceDeployment reference
[constr_5368]	Multiplicity of the reference in the role ComTriggerGrant.serviceDeployment
[constr_5372]	SecureComProps for a PlatformModuleEthernetEndpointConfiguration that contains a UDP configuration
[constr_5373]	SecureComProps for a PlatformModuleEthernetEndpointConfiguration that contains a TCP configuration
[constr_5381]	Modeling of Security Event reports by FunctionalCluster shall not be done via ProcessToMachineMapping
[constr_5386]	Executable.traceSwitchConfiguration
[constr_5387]	Existence of Executable.traceSwitchConfiguration.traceMessage
[constr_5388]	Existence of Executable.traceSwitchConfiguration.traceSwitch
[constr_5392]	Assignment of the same event to several SomeipEventGroups is forbidden in case one of the SomeipEventGroups has the multicastThreshold set to to a value greater than 1

▽

△

Number	Heading
[constr_6905]	CppTemplateArgument.allocator
[constr_10416]	Aggregation of optional templateArgument that defines an Allocator for a VECTOR
[constr_10417]	Existence of attributes of CppImplementationDataTypes depending on the category
[constr_10420]	Restriction for the existence of initial values for PersistencyDataElement
[constr_10425]	Existence of initial values for PersistencyDataElement
[constr_10426]	Multiplicity of attribute PersistencyDeploymentElement.updateStrategy
[constr_10427]	Multiplicity of attribute PersistencyInterfaceElement.updateStrategy
[constr_10428]	Existence of attribute UcmModuleInstantiation.identifier in subclasses
[constr_10429]	Existence of attribute VehicleRolloutStep.violatedSafetyConditionBehavior
[constr_10430]	Existence of attribute Machine.defaultApplicationTimeout
[constr_10431]	Existence of attribute Machine.defaultApplicationTimeout.enterTimeoutValue
[constr_10432]	Existence of attribute Machine.defaultApplicationTimeout.exitTimeoutValue
[constr_10436]	Restriction for the applicability of DiagnosticMultipleMonitorPortMapping
[constr_10437]	Restriction for the applicability of DiagnosticMultipleEventPortMapping
[constr_10438]	Restriction for the applicability of DiagnosticMultipleConditionPortMapping
[constr_10441]	Restriction for NetworkHandlePortMapping.pPortPrototypeInExecutable
[constr_10442]	Restriction for the applicability of DiagnosticSovdProximityChallengePortMapping
[constr_10443]	Restriction for the applicability of DiagnosticSovdProximityChallengeInterface
[constr_10444]	Existence of DiagnosticSovdProximityChallengePortMapping
[constr_10445]	Existence of the reference in the role DiagnosticSovdProximityChallenge-PortMapping.pPortPrototypeInExecutable
[constr_10446]	Existence of the reference in the role DiagnosticSovdProximityChallengePortMapping.process
[constr_10447]	Restriction for the applicability of DiagnosticSovdAuthorizationPortMapping
[constr_10448]	Restriction for the applicability of DiagnosticSovdAuthorizationInterface
[constr_10449]	Existence of DiagnosticSovdAuthorizationPortMapping
[constr_10450]	Existence of the reference in the role DiagnosticSovdAuthorizationPort-Mapping.pPortPrototypeInExecutable
[constr_10451]	Existence of the reference in the role DiagnosticSovdAuthorizationPortMapping.process
[constr_10452]	Restriction for the applicability of DiagnosticSovdBulkDataPortMapping
[constr_10453]	Restriction for the applicability of DiagnosticSovdBulkDataInterface

▽

△

Number	Heading
[constr_10454]	Uniqueness of reference from DiagnosticSovdBulkDataPortMapping to DiagnosticSovdBulkData
[constr_10455]	Existence of the reference in the role DiagnosticSovdBulkDataPortMapping.serviceInstance
[constr_10456]	Existence of the reference in the role DiagnosticSovdBulkDataPortMapping.pPortPrototypeInExecutable
[constr_10457]	Existence of the reference in the role DiagnosticSovdBulkDataPortMapping.process
[constr_10458]	Restriction for the applicability of DiagnosticSovdUpdatePortMapping
[constr_10459]	Restriction for the applicability of DiagnosticSovdUpdateInterface
[constr_10460]	Uniqueness of reference from DiagnosticSovdUpdatePortMapping to DiagnosticSovdUpdate
[constr_10461]	Existence of the reference in the role DiagnosticSovdUpdatePortMapping.serviceInstance
[constr_10462]	Existence of the reference in the role DiagnosticSovdUpdatePortMapping.pPortPrototypeInExecutable
[constr_10463]	Existence of the reference in the role DiagnosticSovdUpdatePortMapping.process
[constr_10464]	Restriction for the applicability of DiagnosticSovdServiceValidationPortMapping
[constr_10465]	Restriction for the applicability of DiagnosticSovdServiceValidationInterface
[constr_10466]	Existence of the reference in the role DiagnosticSovdServiceValidation-PortMapping.pPortPrototypeInExecutable
[constr_10467]	Existence of the reference in the role DiagnosticSovdServiceValidationPortMapping.process
[constr_10468]	Restriction for the applicability of DiagnosticSovdConfigurationPortMapping
[constr_10469]	Restriction for the applicability of DiagnosticSovdConfigurationInterface
[constr_10470]	Uniqueness of reference from DiagnosticSovdConfigurationPortMapping to DiagnosticSovdConfiguration
[constr_10471]	Existence of the reference in the role DiagnosticSovdConfigurationPortMapping.serviceInstance
[constr_10472]	Existence of the reference in the role DiagnosticSovdConfigurationPort-Mapping.pPortPrototypeInExecutable
[constr_10473]	Existence of the reference in the role DiagnosticSovdConfigurationPortMapping.process
[constr_10474]	Uniqueness of reference from DiagnosticSovdConfigurationDataIdentifierMapping to DiagnosticSovdConfigurationParameter
[constr_10475]	Existence of the reference in the role DiagnosticSovdConfigurationDataIdentifierMapping. serviceInstance

▽

△

Number	Heading
[constr_10476]	Existence of the reference in the role DiagnosticSovdConfigurationDataIdentifierMapping.dataIdentifier
[constr_10477]	Existence of DiagnosticSovdLog
[constr_10478]	Existence of DiagnosticSovdUpdate
[constr_10479]	Restriction on values of attribute DiagnosticSovdMethodPrimitive.category aggregated in the role put in the context of SOVD Update
[constr_10480]	Restriction on value of attribute DiagnosticSovdMethodPrimitive.category aggregated in the role get in the context of SOVD Update
[constr_10481]	Restriction on values of attribute DiagnosticSovdMethodPrimitive.category aggregated in the role get in the context of SOVD Log
[constr_10482]	Restriction on values of attribute DiagnosticSovdMethodPrimitive.category aggregated in the roles put and delete in the context of SOVD Log
[constr_10483]	Applicable values of baseTypeEncoding in the context of the definition of a DiagnosticDataElement
[constr_10484]	Existence of FunctionalClusterInteractsWithPersistencyDeployment-Mapping.contractVersion
[constr_10485]	Existence of PersistencyInterface.contractVersion
[constr_10486]	Existence of PersistencyKeyValueDataTypeMapping.previousContractVersion
[constr_10487]	Only one physical address per DiagnosticCommonProps
[constr_10488]	Existence of attribute DiagnosticCommonProps.authenticationTimeout
[constr_10489]	Existence of StateManagementModuleInstantiation in the context of a Machine
[constr_10490]	DiagnosticDataElement shall be directly or indirectly the target of a reference owned by a subclass of DiagnosticMapping
[constr_10491]	Type of PPortPrototype referenced in the role IdsmTimestampProviderMapping.pPortPrototypeInExecutable
[constr_10492]	PPortPrototype typed by a IdsmTimestampProviderInterface
[constr_10493]	Existence of IdsmTimestampProviderMapping.idsPlatformInstantiation
[constr_10494]	Existence of IdsmTimestampProviderMapping.pPortPrototypeInExecutable
[constr_10495]	Existence of IdsmTimestampProviderMapping.process
[constr_10496]	Type of PPortPrototype referenced in the role IdsmContextProviderMapping.pPortPrototypeInExecutable
[constr_10497]	PPortPrototype typed by a IdsmContextProviderInterface
[constr_10498]	Existence of IdsmContextProviderMapping.idsPlatformInstantiation
[constr_10499]	Existence of IdsmContextProviderMapping.pPortPrototypeInExecutable
[constr_10500]	Existence of IdsmContextProviderMapping.process
[constr_10501]	Access to time stamps for *Intrusion Detection System Management*

▽

△

Number	Heading
[constr_10508]	Existence of the attribute UcmSubordinateModuleInstantiation.maxAvailablePersistencyStorageSpace
[constr_10509]	Existence of attribute PersistencyDeployment.maximumAllowedSize
[constr_10510]	Existence of attribute PersistencyDeployment.minimumSustainedSize
[constr_10511]	Reference StateManagementStateRequest.stateRequestPort shall not exist if StateManagementStateRequest is referenced from NmInteractsWithSmMapping
[constr_10512]	Restriction for the value of StateManagementSleepActionItem.sleepTime
[constr_10513]	Existence of the reference in the role SmInteractsWithNmMapping.actionItem
[constr_10514]	Existence of the reference in the role SmInteractsWithNmMapping.nmNetworkHandle
[constr_10515]	Existence of the reference in the role NmInteractsWithSmMapping.stateRequest
[constr_10516]	Existence of the reference in the role NmInteractsWithSmMapping.nmNetworkHandle
[constr_11000]	SomeipEventDeployment.eventReceptionDefaultValue shall not be specified in the physical domain

Table F.55: Added Constraints in R23-11

F.10.5 Changed Constraints in R23-11

Number	Heading
[constr_1675]	Existence of attribute ApSomeipTransformationProps.stringEncoding
[constr_3395]	TransformationPropsToServiceInterfaceElementMapping is restricted to one single ServiceInterface
[constr_3434]	Aggregation of templateArguments for a VECTOR
[constr_3550]	Existence of ServiceInstanceToSignalMapping for an event with signalBased serialization
[constr_3551]	Full mapping of target ISignalGroup
[constr_3552]	Full mapping of target event
[constr_3553]	Existence of ServiceInstanceToSignalMapping for an field with signalBased serialization
[constr_3555]	No support for SecuredIPdu.useAsCryptographicIPdu set to true
[constr_3636]	Consistent ISignal communication direction in and RequiredApServiceInstance
[constr_3637]	Consistent ISignal communication direction out and ProvidedApServiceInstance
[constr_5052]	ProvidedSomeipServiceInstances of the same serviceInterface on one Machine

▽

△

Number	Heading
[constr_5318]	Existence of ServiceInstanceToSignalMapping for an trigger with signalBased serialization
[constr_5351]	FirewallRule is allowed to aggregate at most one protocol subelement
[constr_5352]	DdsRule.submessageType value restriction
[constr_5353]	DdsRule.readerEntityId and DdsRule.writerEntityId value restriction
[constr_5358]	AdaptiveFirewallToPortPrototypeMapping.rPortPrototype restriction
[constr_10048]	Existence of reference from DiagnosticMonitorPortMapping to DiagnosticEvent
[constr_10148]	Multiplicity of reference in the role PersistencyFileElement.contentUri
[constr_10386]	Existence of references StateManagementStateMachineActionItem. startStateMachine and stopStateMachine
[constr_10390]	Existence of reference StateManagementStateRequest.stateRequestPort

Table F.56: Changed Constraints in R23-11

F.10.6 Deleted Constraints in R23-11

Number	Heading
[constr_1543]	Only one physical address per SoftwareCluster.diagnosticDeploymentProps
[constr_1572]	Usage of SwDataDefProps.implementationDataType within a CppImplementationDataType
[constr_1614]	Existence of attribute TransformationPropsToServiceInterfaceElement-Mapping.transformationProps.sessionHandling
[constr_1789]	Scope of machine Function Group
[constr_3394]	Default value for start-up timeout on the Machine is not configurable
[constr_3733]	Upper multiplicity of aggregation in the role ServiceInstanceToSignalMapping.methodMapping
[constr_10095]	Existence of reference from DiagnosticExternalAuthenticationPortMapping to DiagnosticAuthentication
[constr_10111]	Existence of attribute DiagnosticAuthentication.authenticationTimeout
[constr_10156]	Multiplicity of reference in the role ComFindServiceGrantDesign. requiredServicePort
[constr_10229]	Multiplicity of reference in the role SignalBasedMethodToISignalTriggeringMapping.method
[constr_10406]	Existence of DeterministicSyncMOutOfN.numberOfConnectedClients
[constr_10407]	Existence of DeterministicSyncMOutOfN.minimumNumberOfRequests
[constr_10408]	Existence of DeterministicSyncMasterToTimeBaseConsumerMapping. deterministicSyncMaster

▽

△

Number	Heading
[constr_10409]	Existence of `DeterministicSyncMasterToTimeBaseConsumerMapping.timeBaseConsumer`

Table F.57: Deleted Constraints in R23-11

F.10.7 Added Advisories in R23-11

none

F.10.8 Changed Advisories in R23-11

Number	Heading
[advisory_02000]	Define a `ServiceInstanceToSignalMapping` per member of an `AdaptivePlatformServiceInstance`
[advisory_02001]	No explicit `signal/service translation` for `ISignalGroup`s
[advisory_02002]	`ServiceInterface.field` mapping to `ISignalTriggering`s for primitive data type `field`s
[advisory_02003]	`ServiceInterface.field` mapping to `ISignalTriggering`s for composite data type `field`s

Table F.58: Changed Advisories in R23-11

F.10.9 Deleted Advisories in R23-11

none

F.11 Traceable item history of this document according to AUTOSAR Release R24-11

F.11.1 Added Specification Items in R24-11

Number	Heading
[TPS_MANI_01448]	Semantics of `FunctionalClusterInteractsWithDiagnosticEventMapping`
[TPS_MANI_01449]	Service validation in `SOVD` context
[TPS_MANI_01450]	Definition of `invalidValue` for `DataPrototype` typed by a `CppImplementationDataType` of `category STRING`

▽

△

Number	Heading
[TPS_MANI_01451]	Definition of `invalidValue` for `DataPrototype` typed by a `CppImplementationDataType` of `category` `TYPE_REFERENCE`
[TPS_MANI_01452]	Semantics of `DiagnosticTransmitCertificateInterface`
[TPS_MANI_01453]	`DiagnosticServiceInstance`s that can be mapped by a `DiagnosticServiceGenericMapping`
[TPS_MANI_01454]	Semantics of meta-class `IdsmReportingModeProviderInterface`
[TPS_MANI_01455]	Using the `IdsmReportingModeProviderMapping` for the setting and getting the reporting mode in the context of the *Intrusion Detection System Management*
[TPS_MANI_01456]	Definition of `PersistencyDataRequiredComSpec.accessMode`
[TPS_MANI_01457]	Further qualification of properties of `PortPrototype`s typed by `PersistencyFileStorageInterface`s
[TPS_MANI_01458]	Usage of attribute `ComCertificateToCryptoCertificateMapping.category`
[TPS_MANI_01459]	Usage of attribute `ComKeyToCryptoKeySlotMapping.category`
[TPS_MANI_01460]	Retrieve the status of the DoIP group identification to the diagnostic server instance
[TPS_MANI_01461]	Retrieve the status of the DoIP activation line to the diagnostic server instance
[TPS_MANI_01462]	Trigger the DoIP vehicle announcement at the diagnostic server instance
[TPS_MANI_01463]	Retrieving the DoIP entity identification (EID) to the diagnostic server instance
[TPS_MANI_01464]	Retrieving the status of the vehicle's power mode to the diagnostic server instance
[TPS_MANI_01465]	Semantics of reference `SoftwareClusterDesign.doIpLogicalAddress`
[TPS_MANI_01466]	Two ways to interact with crypto key slots
[TPS_MANI_01467]	Modeling of application that uses a Crypto Key
[TPS_MANI_01468]	Semantics of meta-class `CryptoKeySlotUserDesignMapping`
[TPS_MANI_01469]	Semantics of meta-class `CryptoKeySlotUsageDesignMapping`
[TPS_MANI_01470]	Standardized values of attribute `CryptoKeySlotDesign.cryptoAlgorithmId`
[TPS_MANI_01471]	Semantics of meta-class `CryptoKeySlotToClientPortPrototypeMapping`
[TPS_MANI_01472]	Semantics of reference `CryptoKeySlot.cryptoKeySlotDesign`
[TPS_MANI_01473]	Semantics of `StateClientInterface`
[TPS_MANI_01474]	Using the `IdsmQualifiedEventReceiverMapping` for the receiving of qualified security events in the context of the *Intrusion Detection System Management*
[TPS_MANI_01475]	Semantics of meta-class `IdsmQualifiedEventReceiverInterface`
[TPS_MANI_03326]	MachineDesign-specific configuration settings for DoIP
[TPS_MANI_03327]	Network Interface configuration for DoIP
[TPS_MANI_03328]	Reception Monitoring of SOME/IP segments

▽

△

Number	Heading
[TPS_MANI_03329]	Reception Monitoring of SOME/IP segments of the Method Call Message
[TPS_MANI_03330]	Reception Monitoring of SOME/IP segments of the Method Response Message
[TPS_MANI_03331]	local ApApplicationEndpoint and NetworkEndpoint for transmission of security events
[TPS_MANI_03332]	Remote credentials for transmission of security events
[TPS_MANI_03333]	Semantics of DoIpNetworkConfigurationDesign.tpConnection
[TPS_MANI_03335]	Semantics of GenericDiagnosticTransportInstantiation
[TPS_MANI_03674]	Adaptive platform SOME/IP Service Instance interacting with a classic platform SOME/IP Service Instance
[TPS_MANI_03675]	Consuming an IEEE1722 stream
[TPS_MANI_03676]	Producing an IEEE1722 stream
[TPS_MANI_03677]	Producing a Non-Time-Synchronous Control Format messages
[TPS_MANI_03678]	Producing a Time-Synchronous Control Format messages
[TPS_MANI_03679]	Consuming all IEEE1722 ACF messages of an IEEE1722 ACF stream
[TPS_MANI_03680]	Consuming a set of buses from an IEEE1722 ACF stream
[TPS_MANI_03681]	Consuming a set of frames from an IEEE1722 ACF stream
[TPS_MANI_03682]	IEEE1722TpAcfConnection contributes to signal/service translation
[TPS_MANI_03683]	Identification of an ISignalIPdu inside an IEEE1722 ACF stream
[TPS_MANI_03684]	Mapping of an IEEE1722 ACF stream to an EthernetCommunicationConnector

Table F.59: Added Specification Items in R24-11

F.11.2 Changed Specification Items in R24-11

Number	Heading
[TPS_MANI_01287]	Semantics of RawDataStreamMapping
[TPS_MANI_01401]	Support for TLS in vehicle-internal SOVD communication
[TPS_MANI_01416]	Reporting the status of multiple DiagnosticEvents in the context of one PortInterface on the *AUTOSAR adaptive platform*
[TPS_MANI_01417]	Retrieving the status of multiple DiagnosticEvents to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01433]	Semantics of attribute OsModuleInstantiation.osArtiAdapterLaunchBehavior
[TPS_MANI_03264]	Semantics of CryptoKeySlotToPortPrototypeMapping
[TPS_MANI_03269]	Semantics of ComCertificateToCryptoCertificateMapping
[TPS_MANI_03270]	Semantics of ComKeyToCryptoKeySlotMapping

▽

△

Number	Heading
[TPS_MANI_03294]	Semantics of `DoIpNetworkConfiguration.eidRetrieval`
[TPS_MANI_03323]	Support for TLS in DoIP communication
[TPS_MANI_03527]	Definition of `DdsProvidedServiceInstance`
[TPS_MANI_03529]	Definition of `DdsRequiredServiceInstance`
[TPS_MANI_03650]	Definition of `DdsServiceInstanceResourceIdentifierTypeEnum`

Table F.60: Changed Specification Items in R24-11

F.11.3 Deleted Specification Items in R24-11

Number	Heading
[TPS_MANI_01019]	`Manifest` content may apply to different aspects of the *AUTOSAR adaptive platform*
[TPS_MANI_01073]	Semantics of `PortPrototype` typed by `PersistencyKeyValueStorageInterface`
[TPS_MANI_01081]	Semantics of `PortPrototype` typed by `PersistencyFileStorageInterface`
[TPS_MANI_03218]	Default value for the attribute `tcpInitialInactivityTime` of meta-class `DoIpNetworkConfiguration`
[TPS_MANI_03219]	Default value for the attribute `tcpGeneralInactivityTime` of meta-class `DoIpNetworkConfiguration`
[TPS_MANI_03220]	Default value for the attribute `vehicleAnnouncementCount` of meta-class `DoIpNetworkConfiguration`
[TPS_MANI_03221]	Default value for the attribute `vehicleAnnouncementInterval` of meta-class `DoIpNetworkConfiguration`
[TPS_MANI_03222]	Default value for the attribute `tcpAliveCheckResponseTimeout` of meta-class `DoIpNetworkConfiguration`
[TPS_MANI_03517]	Evaluation of `HealthChannelExternalStatus`
[TPS_MANI_03534]	Definition of Platform Health Management Health Channel
[TPS_MANI_03545]	Existence of `HealthChannelExternalStatus`
[TPS_MANI_03546]	Definition of reported health status `RPortPrototype`
[TPS_MANI_03547]	Definition of *offset* time domains
[TPS_MANI_03551]	Definition of Time Base kind
[TPS_MANI_03553]	Applicability of health channel to a specific `Process`
[TPS_MANI_03624]	Usage of `statusId` in application code
[TPS_MANI_03625]	Consistency of `HealthChannelExternalReportedStatus.statusId` and `PhmHealthChannelStatus.statusId`
[TPS_MANI_03630]	Semantics of `triggersRecoveryNotification`

▽

△

Number	Heading
[TPS_MANI_03631]	Semantics of meta-class `PhmHealthChannelRecoveryNotificationInterface`

Table F.61: Deleted Specification Items in R24-11

F.11.4 Added Constraints in R24-11

Number	Heading
[constr_10517]	Upper multiplicity of reference in the role `IdsPlatformInstantiation.idsTimeBase`
[constr_10518]	Mutually exclusive existence of `FunctionalClusterInteractsWithDiagnosticEventMapping.persistencyDeployment` vs. `functionalCluster`
[constr_10519]	Existence of reference `FunctionalClusterInteractsWithDiagnosticEventMapping.diagnosticEvent`
[constr_10526]	`DiagnosticAuthenticationPortMapping` refers to a `PPortPrototype` that is typed by a `DiagnosticTransmitCertificateInterface`
[constr_10528]	Existence of `FirewallStateSwitchInterface.firewallStateMachine`
[constr_10530]	Each Ucm shall process at least one software package
[constr_10535]	Restriction for attribute `SenderComSpec.transmissionProps.onChangeDataPrototype`
[constr_10536]	Existence of `TransmissionComSpecProps.onChangeDataPrototype.dataPrototypeInServiceInterface.rootDataPrototype`
[constr_10537]	Existence of `TransmissionComSpecProps.onChangeDataPrototype.rootDataPrototype`
[constr_10540]	Arguments of a `ClientServerOperation` shall be fully covered by `SignalBasedFireAndForgetMethodToISignalTriggeringMapping`s
[constr_10541]	Consistent assignment of `ISignalTriggering`s to an `ISignalGroup` for `SignalBasedFireAndForgetMethodToISignalTriggeringMapping`
[constr_10546]	`PortPrototype` typed by `DiagnosticGenericUdsInterface` shall only be referenced by `DiagnosticServiceGenericMapping`
[constr_10547]	Reference in the role `UcmStep.ucm` in the context of the enclosing `VehicleRolloutStep`
[constr_10566]	Allowed values of attribute `CppImplementationDataType.swDataDefProps.compuMethod.category`
[constr_10567]	Existence of `DiagnosticSovdMethod.get` resp. `delete` in the context of `DiagnosticSovdFaultMemoryAccess`
[constr_10568]	Type of RPortPrototype referenced in the role `IdsmReportingModeProviderMapping.rPortPrototypeInExecutable`
[constr_10569]	`RPortPrototype` typed by an `IdsmReportingModeProviderInterface`

▽

△

Number	Heading
[constr_10570]	Existence of IdsmReportingModeProviderMapping.idsPlatformInstantiation
[constr_10571]	Existence of IdsmReportingModeProviderMapping.rPortPrototypeInExecutable
[constr_10572]	Existence of IdsmReportingModeProviderMapping.process
[constr_10574]	Value of ApApplicationErrorDomain.value shall be unique in the context of a ClientServerOperation
[constr_10576]	Existence of attribute PersistencyDataRequiredComSpec.accessMode
[constr_10578]	Existence of the attribute UcmMasterModuleInstantiation.ucmNotAvailableOnTheNetwork
[constr_10586]	Restriction for the applicability of DiagnosticDoIpGroupIdentificationPortMapping
[constr_10587]	Restriction for the applicability of DiagnosticDoIPGroupIdentificationInterface
[constr_10588]	Existence of DiagnosticDoIpGroupIdentificationPortMapping.pPort-PrototypeInExecutable
[constr_10589]	Existence of DiagnosticDoIpGroupIdentificationPortMapping.process
[constr_10590]	Restriction for the applicability of DiagnosticDoIpActivationLinePortMapping
[constr_10591]	Restriction for the applicability of DiagnosticDoIPActivationLineInterface
[constr_10592]	Existence of DiagnosticDoIpActivationLinePortMapping.pPort-PrototypeInExecutable
[constr_10593]	Existence of DiagnosticDoIpActivationLinePortMapping.process
[constr_10594]	Restriction for the applicability of DiagnosticDoIpTriggerVehicleAnnouncementPortMapping
[constr_10595]	Restriction for the applicability of DiagnosticDoIPTriggerVehicleAnnouncementInterface
[constr_10596]	Existence of DiagnosticDoIpTriggerVehicleAnnouncementPortMapping.rPortPrototypeInExecutable
[constr_10597]	Existence of DiagnosticDoIpTriggerVehicleAnnouncementPortMapping.process
[constr_10598]	Restriction for the applicability of DiagnosticDoIpEntityIdentificationPortMapping
[constr_10599]	Restriction for the applicability of DiagnosticDoIPEntityIdentificationInterface
[constr_10600]	Existence of DiagnosticDoIpEntityIdentificationPortMapping.pPort-PrototypeInExecutable
[constr_10601]	Existence of DiagnosticDoIpEntityIdentificationPortMapping.process
[constr_10602]	Restriction for the applicability of DiagnosticDoIpPowerModePortMapping
[constr_10603]	Restriction for the applicability of DiagnosticDoIPPowerModeInterface
[constr_10604]	Existence of DiagnosticDoIpPowerModePortMapping.pPortPrototypeInExecutable

▽

△

Number	Heading
[constr_10605]	Existence of DiagnosticDoIpPowerModePortMapping.process
[constr_10609]	Existence of references in the role AbstractDoIpPortMapping.rPortPrototypeInExecutable and AbstractDoIpPortMapping.pPortPrototypeInExecutable
[constr_10611]	Existence of attribute DoIpLogicalAddress.logicalAddress
[constr_10612]	Standardized values for attribute CryptoKeySlotDesign.cryptoObjectType
[constr_10613]	CryptoKeySlotDesign where attribute shared is set to false
[constr_10614]	Existence of attribute CryptoKeySlotDesign.shared
[constr_10616]	Existence of the reference in the role StateClientInterface.functionGroupType
[constr_10617]	Existence of the reference in the role FunctionGroupPortMapping.functionGroup
[constr_10618]	Existence of the reference in the role FunctionGroupPortMapping.process
[constr_10619]	Existence of the reference in the role FunctionGroupPortMapping.rPortPrototypeInExecutable
[constr_10620]	Reference to function group from FunctionGroupPortMapping
[constr_10621]	Same ModeDeclarationGroup for StateClientInterface and FunctionGroupPortMapping.functionGroup
[constr_10622]	Existence of FunctionGroupPortMapping depending on the role rPortPrototypeInExecutable and process
[constr_10623]	Type of RPortPrototype referenced in the role IdsmQualifiedEventReceiverMapping.rPortPrototypeInExecutable
[constr_10624]	RPortPrototype typed by an IdsmQualifiedEventReceiverInterface
[constr_10625]	Existence of IdsmQualifiedEventReceiverMapping.idsPlatformInstantiation
[constr_10626]	Existence of IdsmQualifiedEventReceiverMapping.rPortPrototypeInExecutable
[constr_10627]	Existence of IdsmQualifiedEventReceiverMapping.process
[constr_10628]	Allowed range of attribute providerRateDeviationMax
[constr_3772]	Range of ApApplicationError.errorCode for interoperability with the AUTOSAR classic platform
[constr_3773]	Uniqueness of ApApplicationError.errorCode for interoperability with the AUTOSAR classic platform
[constr_3774]	Existence of EthernetMacRawDataStreamMapping.localCommConnector
[constr_3775]	Existence of IEEE1722RawDataStreamConsumerMapping.ieee1722Stream
[constr_3776]	Existence of IEEE1722RawDataStreamProducerMapping.ieee1722Stream
[constr_3777]	Existence of IEEE1722AcfBusRawDataStreamConsumerMapping.acfBus
[constr_3778]	Existence of IEEE1722AcfBusPartRawDataStreamConsumerMapping.acfBusPart
[constr_9323]	Restriction of DoIpFunctionalClusterDesign on a MachineDesign
[constr_9324]	Allowed PlatformModuleEthernetEndpoint-Configuration.communicationConnector references

▽

△

Number	Heading
[constr_9325]	DoIpFunctionalClusterDesign.doIpLogicalAddress.logicalAddress shall be within the physical address boundaries defined by DoIpRequestConfiguration
[constr_9327]	SOME/IP segment reception monitoring allowed for udp SomeipEventDeployments
[constr_9328]	Reception Monitoring of SOME/IP segments is allowed for udp SomeipMethodDeployments
[constr_9329]	Allowed PlatformModuleEthernetEndpointConfiguration.tcpPort and PlatformModuleEthernetEndpointConfiguration.udpPort references
[constr_9334]	Supported value range for attribute GenericTpConnection.sourceAddressRangeStart
[constr_9335]	Supported value range for attribute GenericTpConnection.sourceAddressRangeEnd
[constr_9336]	DoIpNetworkConfigurationDesign shall reference a GenericTpConnection
[constr_9337]	Same priority for overlapping address ranges in the context of the same MachineDesign
[constr_9338]	GenericTpConnection.sourceAddressRangeStart shall be less or equal of GenericTpConnection.sourceAddressRangeEnd
[constr_9344]	Supported value range for attribute DoIpFunctionalClusterDesign.doIpProtocolVersion

Table F.62: Added Constraints in R24-11

F.11.5 Changed Constraints in R24-11

Number	Heading
[constr_10186]	Multiplicity of attribute DoIpFunctionalClusterDesign.entityStatusMaxByteFieldUse
[constr_10189]	Multiplicity of attribute DoIpFunctionalClusterDesign.maxRequestBytes
[constr_10209]	Existence of the reference in the role AbstractCryptoKeySlotToPortPrototypeMapping.keySlot
[constr_10210]	Existence of the reference in the role AbstractCryptoKeySlotToPortPrototypeMapping.process
[constr_10438]	Restriction for the applicability of DiagnosticMultipleConditionPortMapping
[constr_10483]	Applicable values of baseTypeEncoding in the context of the definition of a DiagnosticDataElement
[constr_10487]	Only one physical address per DiagnosticCommonProps
[constr_10488]	Existence of attribute DiagnosticCommonProps.authenticationTimeout
[constr_10511]	Reference StateManagementStateRequest.stateRequestPort shall not exist if StateManagementStateRequest is referenced from NmInteractsWithSmMapping

▽

△

Number	Heading
[constr_1478]	SwDataDefProps applicable to ApplicationDataTypes exclusive to the *AUTOSAR adaptive platform*
[constr_1579]	SwDataDefProps applicable to CppImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*
[constr_1695]	Semantics of a Grant depends on the existence of CmModuleInstantiation
[constr_1729]	Qualified combinations of PortPrototypes and PhmSupervisionRecoveryNotificationInterface on State Management software level
[constr_3414]	Allowed usage of PlatformModuleEthernetEndpointConfiguration attributes that are allowed to be used to configure the network communication in the different platform modules
[constr_3495]	Supported value range for attribute DoIpFunctionalClusterDesign.eid
[constr_3497]	Supported value range for attribute DoIpFunctionalClusterDesign.maxRequestBytes
[constr_3645]	discoveryType mandatory for DdsProvidedServiceInstance
[constr_3646]	resourceIdentifierType mandatory for DdsProvidedServiceInstance
[constr_3647]	resourceIdentifierType value for USER_DATA QoS-based discovery
[constr_3648]	discoveryType mandatory for DdsRequiredServiceInstance
[constr_3677]	ComGrants referencing DDS Service Instances
[constr_5241]	Restriction applicable for AbstractCryptoKeySlotToPortPrototypeMapping.portPrototype
[constr_5343]	Usage of DoIpNetworkConfiguration.eidRetrieval
[constr_5372]	SecureComProps for a PlatformModuleEthernetEndpointConfiguration that contains a UDP configuration
[constr_5373]	SecureComProps for a PlatformModuleEthernetEndpointConfiguration that contains a TCP configuration
[constr_5381]	Modeling of Security Event reports by FunctionalCluster shall not be done via ProcessToMachineMapping
[constr_5386]	Existence of Executable.traceSwitchConfiguration
[constr_5387]	Existence of reference Executable.traceSwitchConfiguration.traceMessage
[constr_5388]	Existence of attribute Executable.traceSwitchConfiguration.traceSwitch
[constr_5392]	Assignment of the same event to several SomeipEventGroups is forbidden in case one of the SomeipEventGroups has the multicastThreshold set to a value greater than 0

Table F.63: Changed Constraints in R24-11

F.11.6 Deleted Constraints in R24-11

Number	Heading
[constr_10022]	Restriction for SecurityEventMapping.process.securityEvent.id w.r.t SecurityEventMapping.id
[constr_10062]	DiagnosticServiceInstances that can be mapped by a DiagnosticServiceGenericMapping
[constr_10150]	Multiplicity of reference in the role SynchronizedTimeBaseProviderInterface.timeBaseKind
[constr_10153]	Multiplicity of reference in the role PersistencyDataRequiredComSpec.dataElement
[constr_10187]	Multiplicity of attribute DoIpInstantiation.gidInvalidityPattern
[constr_10190]	Multiplicity of attribute DoIpInstantiation.vinInvalidityPattern
[constr_10241]	Multiplicity of attribute SoftwarePackage.compressedSoftwarePackageSize
[constr_1690]	SoftwareCluster shall only be referenced by a single SoftwarePackage.
[constr_1728]	Qualified combinations of PortPrototypes and PhmHealthChannelInterface on application software level
[constr_3532]	Mandatory definition of statusId
[constr_3614]	Reference to a PhmHealthChannelRecoveryNotificationInterface in the context of a HealthChannelExternalStatus
[constr_3719]	RecoveryNotification referenced either by HealthChannelExternalStatus or HealthChannelSupervision

Table F.64: Deleted Constraints in R24-11

F.11.7 Added Advisories in R24-11

Number	Heading
[advisory_01011]	Existence of the reference in the role CryptoKeySlotUserDesignMapping.processDesign
[advisory_01012]	Existence of the reference in the role CryptoKeySlotUserDesignMapping.cryptoKeySlotUserDesign
[advisory_01013]	Existence of the reference in the role CryptoKeySlotUserDesignMapping.processDesign
[advisory_01014]	Existence of the reference in the role CryptoKeySlotUsageDesignMapping.portPrototype
[advisory_01015]	Existence of the reference in the role CryptoKeySlotUsageDesignMapping.cryptoKeySlotUsageDesign
[advisory_01016]	Existence of the aggregation in the role CryptoKeySlotUserDesign.keySlotUsage
[advisory_03003]	Secure communication should not be lost by switching to multicast

Table F.65: Added Advisories in R24-11

F.11.8 Changed Advisories in R24-11

Number	Heading
[advisory_02001]	No explicit `signal/service translation` for `ISignalGroup`s

Table F.66: Changed Advisories in R24-11

F.11.9 Deleted Advisories in R24-11

none

F.12 Traceable item history of this document according to AUTOSAR Release R25-11

F.12.1 Added Specification Items in R25-11

Number	Heading
[TPS_MANI_01476]	Semantics of meta-class `FunctionalClusterToDltLogSinkMapping`
[TPS_MANI_01477]	Description of Core affinity
[TPS_MANI_01478]	Supported error handling scenarios in the context of state management
[TPS_MANI_01479]	Semantics of meta-class `SoftwareClusterDependencyFormula`
[TPS_MANI_01480]	Semantics of `SoftwareClusterDesign.dependsOn`
[TPS_MANI_01481]	Semantics of meta-class `SoftwareClusterDependencyCompareCondition`
[TPS_MANI_01482]	Semantics of `SoftwareClusterDesign.conflictsTo`
[TPS_MANI_01484]	Mapping of `DiagnosticExtendedDataRecordElement` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01485]	Support of `CryptoCertificateGroup`s
[TPS_MANI_01486]	Modeling of application that accesses a Crypto Certificate Group
[TPS_MANI_01487]	Support for the execution of operations in the context of `SOVD`
[TPS_MANI_01488]	Modeling of the `DiagnosticSovdOperationInterface`
[TPS_MANI_01489]	Support for data access in the context of `SOVD`
[TPS_MANI_01490]	Mapping of `DiagnosticAbstractSovdContent` to `PortPrototype` on the *AUTOSAR adaptive platform*
[TPS_MANI_01491]	Mapping of `DiagnosticSovdOperation` to a `PPortPrototype` on the *AUTOSAR adaptive platform*
[TPS_MANI_01492]	Standardized values for attribute `DiagnosticSovdDataCategory.category`
[TPS_MANI_01493]	Semantics of meta-class `DiagnosticSovdContent`
[TPS_MANI_01494]	Semantics of meta-class `DiagnosticSovdContentElement`

△

Number	Heading
[TPS_MANI_01495]	Modeling of hierarchical data structures using DiagnosticSovdContentElement
[TPS_MANI_01496]	DiagnosticSovdArrayContentElement can have fixed or variable size
[TPS_MANI_01497]	DiagnosticSovdContentElement can be declared optional
[TPS_MANI_01498]	Semantics of attribute DiagnosticSovdPrimitiveContent-Element.dataConstr.dataConstrRule.physConstrs
[TPS_MANI_01499]	InternalConstrs is not supported in the context of a DiagnosticSovdPrimitiveContentElement
[TPS_MANI_01500]	Semantics of attribute DiagnosticSovdPrimitiveContentElement.swDataDefProps.unit
[TPS_MANI_01501]	Semantics of DiagnosticSovdPrimitiveContentElement.type
[TPS_MANI_01502]	Definition of DiagnosticAccessPermission for DiagnosticSovdOperation
[TPS_MANI_01503]	Semantics of DiagnosticSovdOperationPrimitive
[TPS_MANI_01504]	Semantics of DiagnosticEnvSovdDataCondition
[TPS_MANI_01505]	Semantics of DiagnosticTroubleCodeProps.sovdSnapshotContent
[TPS_MANI_01506]	Semantics of DiagnosticTroubleCodeProps. legislatedSovdFreezeFrameContent
[TPS_MANI_01507]	Semantics of AbstractSuspendToRamInterface
[TPS_MANI_01508]	Semantics of meta-class StateManagementSuspendToRamActionItem
[TPS_MANI_01509]	Semantics of SuspendToRamSatelliteInterface
[TPS_MANI_01510]	Mapping of SuspendToRamModuleInstantiation to a PortPrototype in the suspend-to-RAM satellite role
[TPS_MANI_01511]	Semantics of SuspendToRamHubInterface
[TPS_MANI_01512]	Mapping of SuspendToRamModuleInstantiation to a PortPrototype in the suspend-to-RAM hub role
[TPS_MANI_01513]	Semantics of Executable.suspendToRamAwareness
[TPS_MANI_01514]	Semantics of DiagnosticSovdProps.singleElementSerialization
[TPS_MANI_01515]	Mapping of DiagnosticSovdConfigurationParameter to DiagnosticSovdContent on the *AUTOSAR adaptive platform*
[TPS_MANI_03336]	Semantics of TraceSwitchConfig
[TPS_MANI_03337]	The definition of the namespace of a TraceSwitchConfig may follow a hierarchical pattern
[TPS_MANI_03338]	Ordered definition of TraceSwitchConfig.namespace

Table F.67: Added Specification Items in R25-11

F.12.2 Changed Specification Items in R25-11

Number	Heading
[TPS_MANI_01112]	Semantics of SoftwareClusterDesign
[TPS_MANI_01116]	Reference to model elements included in an uploadable software package
[TPS_MANI_01119]	Reference to model elements from SoftwareClusterDesign
[TPS_MANI_01229]	Pre-allocation of a given ProcessDesign on a specific MachineDesign
[TPS_MANI_01230]	Semantics of DiagnosticProvidedDataMapping
[TPS_MANI_01329]	Reference to model elements in different SoftwareClusters
[TPS_MANI_01331]	Standardized values of attribute SoftwareClusterDesign.category and SoftwareCluster.category
[TPS_MANI_01347]	Definition of a DiagnosticDataElement used in the context of a DID obtained from a PPortPrototype typed by a DiagnosticDataElementInterface
[TPS_MANI_01348]	Definition of a DiagnosticDataElement used in the context of a DID obtained from a PPortPrototype typed by a DiagnosticDataIdentifierInterface
[TPS_MANI_01387]	Semantics of meta-class StateManagementActionList
[TPS_MANI_01396]	Semantics of meta-class DiagnosticSovdLock
[TPS_MANI_01399]	Modeling of the configuration of Service-oriented Vehicle Diagnostics
[TPS_MANI_01400]	Secure communication between SOVD Gateway and SOVD Server
[TPS_MANI_01401]	Support for TLS in vehicle-internal SOVD communication
[TPS_MANI_01402]	Identification of the SOVD Server
[TPS_MANI_01403]	External communication of the SOVD gateway
[TPS_MANI_01404]	Secure communication between SOVD gateway and SOVD Client
[TPS_MANI_01405]	Semantics of sub-classes of SoftwareClusterDiagnosticAddress
[TPS_MANI_01410]	Semantics of attribute SoftwareClusterDesign.installationBehavior
[TPS_MANI_01418]	Mapping of multiple DiagnosticConditions to a PortPrototype on the *AUTOSAR adaptive platform*
[TPS_MANI_01424]	Semantics of DiagnosticSovdProximityChallengePortMapping
[TPS_MANI_01425]	Semantics of DiagnosticSovdAuthorizationPortMapping
[TPS_MANI_01426]	Mapping of DiagnosticSovdBulkData to PortPrototype on the *AUTOSAR adaptive platform*
[TPS_MANI_01427]	Mapping of DiagnosticSovdUpdate to PortPrototype on the *AUTOSAR adaptive platform*
[TPS_MANI_01428]	Mapping of DiagnosticSovdServiceInstance to PortPrototype on the *AUTOSAR adaptive platform*
[TPS_MANI_01429]	Mapping of DiagnosticSovdConfiguration to PortPrototype on the *AUTOSAR adaptive platform*
[TPS_MANI_01431]	Definition of SOVD service instance
[TPS_MANI_01432]	Semantics of DiagnosticSovdMethodPrimitive

▽

△

Number	Heading
[TPS_MANI_01438]	Existence of `StateManagementStateMachineActionItem.override-InitialState`
[TPS_MANI_01449]	Service validation in `SOVD` context
[TPS_MANI_01461]	Retrieve the status of the DoIP activation line to the diagnostic server instance
[TPS_MANI_01462]	Trigger the DoIP vehicle announcement at the diagnostic server instance
[TPS_MANI_03015]	TTL for Offer Service Entries
[TPS_MANI_03207]	IPsec Internet Key Exchange protocol configuration
[TPS_MANI_03514]	Expiration tolerance for `GlobalSupervision`
[TPS_MANI_03543]	Definition of time sync correction attributes
[TPS_MANI_03577]	Definition of `Arbitrary Message Header` ID

Table F.68: Changed Specification Items in R25-11

F.12.3 Deleted Specification Items in R25-11

Number	Heading
[TPS_MANI_01164]	Semantics of `SoftwareCluster.dependsOn`
[TPS_MANI_01214]	Semantics of `SoftwareCluster.conflictsTo`
[TPS_MANI_01215]	Semantics of meta-class `SoftwareClusterDependencyFormula`
[TPS_MANI_01217]	Semantics of meta-class `SoftwareClusterDependencyCompareCondition`
[TPS_MANI_01284]	Granularity of meta-class `RawDataStreamGrantDesign`
[TPS_MANI_01307]	Semantics of meta-class `EthernetRawDataStreamGrant`
[TPS_MANI_01397]	Standardized values of `DiagnosticDataIdentifier.category` of the usage in `SOVD` context
[TPS_MANI_01398]	Definition of an *SOVD data group*
[TPS_MANI_01430]	Mapping of `DiagnosticSovdConfigurationParameter` to `DiagnosticDataIdentifier` on the *AUTOSAR adaptive platform*
[TPS_MANI_03258]	Modeling of application designed as trust-master
[TPS_MANI_03266]	Semantics of `CryptoCertificateToCryptoKeySlotMapping`
[TPS_MANI_03628]	Standardized values of `ServiceInterface.category`

Table F.69: Deleted Specification Items in R25-11

F.12.4 Added Constraints in R25-11

Number	Heading
[constr_10629]	DdsServiceInterfaceDeployment shall cover all applicable elements of the corresponding ServiceInterface
[constr_10630]	PPortPrototype that is typed by a DiagnosticDataIdentifierInterface shall be mapped to a DiagnosticDataIdentifier
[constr_10631]	PPortPrototype that is typed by a DiagnosticDataElementInterface shall be mapped to a DiagnosticDataElement
[constr_10633]	Existence of references StateManagementStateMachineActionItem.overrideInitialState and stopAgent
[constr_10634]	ClientServerOperation aggregated by meta-class DiagnosticDataIdentifierInterface or meta-class DiagnosticDataElementInterface
[constr_10641]	Existence of reference in the role StateManagementModuleInstantiation.maxActionListDuration
[constr_10648]	Target of the reference in the role DiagnosticDoIpActivationLinePortMapping.networkConfigurationDesign
[constr_10649]	Existence of reference in the role DiagnosticDoIpActivationLinePortMapping.networkConfigurationDesign
[constr_10650]	Target of the reference in the role DiagnosticDoIpTriggerVehicleAnnouncementPortMapping.networkConfigurationDesign
[constr_10651]	Existence of reference in the role DiagnosticDoIpTriggerVehicleAnnouncementPortMapping.networkConfigurationDesign
[constr_10652]	SystemMappings applicable for a System of category MACHINE_DESIGN_EXTRACT
[constr_10653]	Existence of attribute UriDescription.uri
[constr_10654]	Existence of PersistencyFileRequiredComSpec
[constr_10655]	Allowed range of values for attribute ServiceInterfaceDeploymentElement.minimumSendInterval
[constr_10657]	Multiplicity of reference in the role ProcessDesignToMachineDesignMapping.machineDesign
[constr_10658]	Multiplicity of the reference in the role ProcessToMachineMapping.machine
[constr_10659]	Consistency of references in the roles ProcessDesignToMachineDesignMapping.machineDesign and shallRunOn resp. shallNotRunOn
[constr_10660]	Usage of shallRunOn and shallNotRunOn references
[constr_10661]	Consistency of references shallRunOn and shallNotRunOn
[constr_10662]	Mutual exclusive existence of references shallRunOn and shallNotRunOn
[constr_10663]	Existence of attribute DiagnosticCommonProps.authenticationTimeout

▽

△

Number	Heading
[constr_10666]	Restriction for StateManagementStateMachineActionItem
[constr_10682]	Restriction of the multiplicity of DiagnosticExtendedDataRecordInterface.provide.argument
[constr_10683]	Restriction for the data type of DiagnosticExtendedDataRecordInterface.provide.argument
[constr_10684]	Restriction for the applicability of DiagnosticExtendedDataRecordClientPortMapping
[constr_10685]	Restriction for the reference to an RPortPrototype typed by a DiagnosticExtendedDataRecordInterface
[constr_10686]	Multiplicity of reference in the role DiagnosticExtendedDataRecordClientPortMapping.recordElement
[constr_10690]	Existence of CryptoAlgorithmDescription.cryptoAlgorithmId
[constr_10691]	Existence of CryptoAlgorithmDescription.cryptoKeyType
[constr_10692]	Existence of aggregation in the role DiagnosticSovdOperationInterface.start
[constr_10693]	Existence of reference in the role DiagnosticSovdAccessArgument.contentObject
[constr_10694]	Restriction for DiagnosticSovdAccessOperation aggregated by a DiagnosticSovdContentInterface
[constr_10695]	Existence of DiagnosticSovdAccessOperation.argument if DiagnosticSovdAccessOperation that is aggregated in the role DiagnosticSovdContentElementInterface.read
[constr_10696]	Target of the reference DiagnosticSovdContentPortMapping.pPortPrototypeInExecutable
[constr_10697]	DiagnosticSovdContentPortMapping maps on the level of SOVD Data
[constr_10698]	DiagnosticSovdContentPortMapping maps on the level of SOVD Data Element
[constr_10699]	Existence of the Reference in the role DiagnosticSovdContentPortMapping.pPortPrototypeInExecutable
[constr_10700]	Existence of the Reference in the role DiagnosticSovdContentPortMapping.content
[constr_10701]	Existence of the Reference in the role DiagnosticSovdContentPortMapping.process
[constr_10702]	Existence of the Reference in the role DiagnosticSovdOperationPortMapping.pPortPrototypeInExecutable
[constr_10703]	Existence of the Reference in the role DiagnosticSovdOperationPortMapping.sovdOperation
[constr_10704]	Existence of the Reference in the role DiagnosticSovdOperationPortMapping.process
[constr_10705]	Restriction regarding references in the roles DiagnosticSovdDataCategory.group and DiagnosticSovdData.sovdGroup
[constr_10706]	Existence of reference in the role DiagnosticSovdData.sovdDataCategory
[constr_10707]	Relation of the values of attributes minNumberOfElements and maxNumberOfElements

▽

△

Number	Heading
[constr_10708]	DiagnosticSovdContentElement aggregated in the role DiagnosticSovdArrayContentElement.element shall not be declared *optional*
[constr_10711]	Existence of attribute DiagnosticSovdArrayContentElement.maxNumberOfElements
[constr_10712]	Allowed roles within SwDataDefProps in the context of DiagnosticSovdPrimitiveContentElement
[constr_10713]	Target of the reference DiagnosticSovdOperationPortMapping.pPortPrototypeInExecutable
[constr_10714]	PPortPrototype that is typed by a DiagnosticSovdOperationInterface as a mapping target
[constr_10715]	DiagnosticSovdOperation shall not refer to DiagnosticSovdMethod
[constr_10716]	Existence of DiagnosticSovdOperation.start
[constr_10717]	Existence of DiagnosticSovdOperation.proximityProofRequired
[constr_10718]	Existence of DiagnosticEnvSovdDataCondition.compareValue
[constr_10719]	Existence of DiagnosticEnvSovdDataCondition.contentElement
[constr_10720]	Restriction for the value of StateManagementSuspendToRamActionItem.maxActionItemDuration
[constr_10721]	Target of AbstractSuspendToRamMapping.pPortPrototypeInExecutable
[constr_10722]	Restriction for PPortPrototype that is typed by a SuspendToRamSatelliteInterface
[constr_10723]	Existence of SuspendToRamSatelliteMapping.moduleInstantiation
[constr_10724]	Existence of SuspendToRamSatelliteMapping.pPortPrototypeInExecutable
[constr_10725]	Existence of SuspendToRamSatelliteMapping.process
[constr_10726]	Target of AbstractSuspendToRamMapping.pPortPrototypeInExecutable
[constr_10727]	Restriction for PPortPrototype that is typed by a SuspendToRamHubInterface
[constr_10728]	Restriction for SuspendToRamHubMapping.moduleInstantiation
[constr_10729]	Existence of SuspendToRamHubMapping.moduleInstantiation
[constr_10730]	Existence of SuspendToRamHubMapping.pPortPrototypeInExecutable
[constr_10731]	Existence of SuspendToRamHubMapping.process
[constr_10732]	Existence of attributes DiagnosticExtendedDataRecord.trigger and update on the *AUTOSAR adaptive platform*
[constr_10733]	Uniqueness of reference from DiagnosticSovdConfigContentMapping to DiagnosticSovdConfigurationParameter
[constr_10734]	Existence of the reference in the role DiagnosticSovdConfigContentMapping.serviceInstance
[constr_10735]	Existence of the reference in the role DiagnosticSovdConfigContentMapping.content
[constr_3795]	No Platform Health Management supervision for non-reporting Executables
[constr_3798]	Applicable attributes for an Ethernet node on the adaptive platform

▽

△

Number	Heading
[constr_3823]	Existence of `GlobalSupervision.expiredSupervisionTolerance`
[constr_9350]	All Provided Service Instances using the same UDP endpoint shall use the same DTLS SERVER configuration settings
[constr_9351]	All Provided Service Instances using the same TCP endpoint shall use the same TLS SERVER configuration settings
[constr_9352]	Mutual exclusive existence of `RemoteEndpointConfiguration.tcpPort` and `RemoteEndpointConfiguration.udpPort`
[constr_9353]	Mutual exclusive existence of `RemoteEndpointConfiguration.ipv4Address` and `RemoteEndpointConfiguration.ipv6Address`
[constr_9356]	Upper multiplicity of reference in the role `DoIpNetworkConfigurationDesign.networkConfiguration`
[constr_9370]	`ApSomeipTransformationProps` shall be defined at most once for a `ServiceInterface` element
[constr_9392]	Existence of reference `TraceSwitchConfig.traceMessage`
[constr_9393]	Existence of attribute `TraceSwitchConfig.traceSwitch`
[constr_9394]	Existence of reference `TraceSwitchConfig.executable`

Table F.70: Added Constraints in R25-11

F.12.5 Changed Constraints in R25-11

Number	Heading
[constr_10029]	`ServiceInterfaceDeployment` (except `DdsServiceInterfaceDeployment`) shall cover all elements of the corresponding `ServiceInterface`
[constr_10037]	Existence of attribute `TagWithOptionalValue.sequenceOffset` in the context of attribute `capabilityRecord` owned by `ProvidedSomeipServiceInstance`, `RequiredSomeipServiceInstance`, or `AbstractServiceInstance`
[constr_10154]	Multiplicity of reference in the role `ProcessDesignToMachineDesignMapping.processDesign`
[constr_10172]	Multiplicity of attribute `Processor.core`
[constr_10173]	Multiplicity of attribute `ProcessorCore.coreId`
[constr_10386]	Existence of references `StateManagementStateMachineActionItem.startAgent` and `stopAgent`
[constr_10400]	Existence of `SovdServerInstantiation.componentQualifier`
[constr_10401]	Existence of `SovdGatewayLocalEndpointTcpConfig.tcpPort`
[constr_10402]	Existence of `SovdGatewayEthernetCredentials.ipv4Address` vs. `SovdGatewayEthernetCredentials.ipv6Address`
[constr_10403]	Existence of `SovdGatewayEthernetCredentials.udpPort`
[constr_10404]	Existence of `SoftwareClusterSovdAddress.componentQualifier`

▽

△

Number	Heading
[constr_10410]	Value of SoftwareClusterDesign.installationBehavior for a SoftwareClusterDesign of category PLATFORM_CORE
[constr_10442]	Restriction for the applicability of DiagnosticSovdProximityChallengePortMapping
[constr_10443]	Restriction for the applicability of DiagnosticSovdProximityChallengeInterface
[constr_10444]	Existence of DiagnosticSovdProximityChallengePortMapping
[constr_10445]	Existence of the reference in the role DiagnosticSovdProximityChallenge-PortMapping.pPortPrototypeInExecutable
[constr_10446]	Existence of the reference in the role DiagnosticSovdProximityChallengePortMapping.process
[constr_10447]	Restriction for the applicability of DiagnosticSovdAuthorizationPortMapping
[constr_10448]	Restriction for the applicability of DiagnosticSovdAuthorizationInterface
[constr_10449]	Existence of DiagnosticSovdAuthorizationPortMapping
[constr_10450]	Existence of the reference in the role DiagnosticSovdAuthorizationPort-Mapping.pPortPrototypeInExecutable
[constr_10451]	Existence of the reference in the role DiagnosticSovdAuthorizationPortMapping.process
[constr_10452]	Restriction for the applicability of DiagnosticSovdBulkDataPortMapping
[constr_10453]	Restriction for the applicability of DiagnosticSovdBulkDataInterface
[constr_10454]	Uniqueness of reference from DiagnosticSovdBulkDataPortMapping to DiagnosticSovdBulkData
[constr_10455]	Existence of the reference in the role DiagnosticSovdBulkDataPortMapping.serviceInstance
[constr_10456]	Existence of the reference in the role DiagnosticSovdBulkDataPortMapping.pPortPrototypeInExecutable
[constr_10457]	Existence of the reference in the role DiagnosticSovdBulkDataPortMapping.process
[constr_10458]	Restriction for the applicability of DiagnosticSovdUpdatePortMapping
[constr_10459]	Restriction for the applicability of DiagnosticSovdUpdateInterface
[constr_10460]	Uniqueness of reference from DiagnosticSovdUpdatePortMapping to DiagnosticSovdUpdate
[constr_10461]	Existence of the reference in the role DiagnosticSovdUpdatePortMapping.serviceInstance
[constr_10462]	Existence of the reference in the role DiagnosticSovdUpdatePortMapping.pPortPrototypeInExecutable
[constr_10463]	Existence of the reference in the role DiagnosticSovdUpdatePortMapping.process
[constr_10464]	Restriction for the applicability of DiagnosticSovdServiceValidationPortMapping
[constr_10465]	Restriction for the applicability of DiagnosticSovdServiceValidationInterface

▽

△

Number	Heading
[constr_10466]	Existence of the reference in the role DiagnosticSovdServiceValidationPortMapping.pPortPrototypeInExecutable
[constr_10467]	Existence of the reference in the role DiagnosticSovdServiceValidationPortMapping.process
[constr_10468]	Restriction for the applicability of DiagnosticSovdConfigurationPortMapping
[constr_10469]	Restriction for the applicability of DiagnosticSovdConfigurationInterface
[constr_10470]	Uniqueness of reference from DiagnosticSovdConfigurationPortMapping to DiagnosticSovdConfiguration
[constr_10471]	Existence of the reference in the role DiagnosticSovdConfigurationPortMapping.serviceInstance
[constr_10472]	Existence of the reference in the role DiagnosticSovdConfigurationPortMapping.pPortPrototypeInExecutable
[constr_10473]	Existence of the reference in the role DiagnosticSovdConfigurationPortMapping.process
[constr_10477]	Existence of DiagnosticSovdLog
[constr_10478]	Existence of DiagnosticSovdUpdate
[constr_10479]	Restriction on values of attribute DiagnosticSovdMethodPrimitive.category aggregated in the role put in the context of SOVD Update
[constr_10480]	Restriction on value of attribute DiagnosticSovdMethodPrimitive.category aggregated in the role get in the context of SOVD Update
[constr_10481]	Restriction on values of attribute DiagnosticSovdMethodPrimitive.category aggregated in the role get in the context of SOVD Log
[constr_10482]	Restriction on values of attribute DiagnosticSovdMethodPrimitive.category aggregated in the roles put and delete in the context of SOVD Log
[constr_10483]	Applicable values of baseTypeEncoding in the context of the definition of a DiagnosticDataElement
[constr_10490]	DiagnosticDataElement shall be directly or indirectly the target of a reference owned by a subclass of DiagnosticMapping
[constr_10567]	Existence of DiagnosticSovdMethod.get resp. delete in the context of DiagnosticSovdFaultMemoryAccess
[constr_10612]	Standardized values for attribute CryptoKeySlotDesignDescription.cryptoObjectType
[constr_1507]	PortInterfaceToDataTypeMapping is only applicable to PortInterfaces that contain modeled data types
[constr_1549]	Value of ProcessorCore.coreId
[constr_1693]	Relation of Executable, ProcessDesign, and Process
[constr_1697]	Restriction for ClientServerOperation aggregated by a DiagnosticDataIdentifierInterface
[constr_1785]	Restriction regarding the reference into another SoftwareCluster
[constr_3421]	Fibex elements applicable for a System of category MACHINE_DESIGN_EXTRACT
[constr_3650]	headerId required in case of Arbitrary Message Header

▽

△

Number	Heading
[constr_5239]	Predefined values for `CryptoKeySlotContentAllowedUsage.allowedKeyslotUsage`
[constr_5291]	Allowed usage of `LTMessageCollectionToPortPrototypeMapping.rPort`

Table F.71: Changed Constraints in R25-11

F.12.6 Deleted Constraints in R25-11

Number	Heading
[constr_10007]	Existence of `ProcessExecutionError.executionError`
[constr_10008]	Value of `ProcessExecutionError.executionError`
[constr_10195]	Multiplicity of attribute `DoIpNetworkConfiguration.vehicleIdentificationSyncStatus`
[constr_10211]	Multiplicity of the reference in the role `CryptoCertificateToCryptoKeySlotMapping.cryptoCertificate`
[constr_10474]	Uniqueness of reference from `DiagnosticSovdConfigurationDataIdentifierMapping` to `DiagnosticSovdConfigurationParameter`
[constr_10475]	Existence of the reference in the role `DiagnosticSovdConfigurationDataIdentifierMapping.serviceInstance`
[constr_10476]	Existence of the reference in the role `DiagnosticSovdConfigurationDataIdentifierMapping.dataIdentifier`
[constr_10488]	Existence of attribute `DiagnosticCommonProps.authenticationTimeout`
[constr_11000]	`eventReceptionDefaultValue` or `eventReceptionDefaultValue` shall not be specified in the physical domain
[constr_3639]	Existence of `SupervisionMode.SupervisionMode`
[constr_5386]	Existence of `Executable.traceSwitchConfiguration`
[constr_5387]	Existence of reference `Executable.traceSwitchConfiguration.traceMessage`
[constr_5388]	Existence of attribute `Executable.traceSwitchConfiguration.traceSwitch`

Table F.72: Deleted Constraints in R25-11

F.12.7 Added Advisories in R25-11

Number	Heading
[advisory_01017]	Existence of networkInterface if IdsmModuleInstantiation.reportableSecurityEvent exists
[advisory_01019]	Existence of attribute DiagnosticSovdArrayContentElement.element
[advisory_01020]	Existence of attribute DiagnosticSovdArrayContentElement.minNumberOfElements
[advisory_01021]	Existence of attribute DiagnosticSovdRecordContentElement.element
[advisory_03004]	Defintion of SOME/IP serialization properties via ApSomeipTransformationProps

Table F.73: Added Advisories in R25-11

F.12.8 Changed Advisories in R25-11

none

F.12.9 Deleted Advisories in R25-11

none

F.13 Traceable item history of this document according to AUTOSAR Release R26-11

F.13.1 Added Specification Items in R26-11

Number	Heading
[TPS_MANI_01516]	Different transformationProps for the methodCall and methodReturn where attribute argument is set to direction inout
[TPS_MANI_01517]	Semantics of meta-class CryptoCertificateGroupToPortPrototypeMapping
[TPS_MANI_01518]	Definition of launch attributes for Process in the context of Process
[TPS_MANI_01519]	Definition of launch attributes for Process in the context of StateDependentStartupConfig
[TPS_MANI_01520]	Receiving custom payloads from a DoIP client
[TPS_MANI_01521]	Sending custom payloads to a DoIP client
[TPS_MANI_01522]	Access scenarios for the fault memory
[TPS_MANI_01523]	Semantics of IdsmModuleInstantiationToCryptoKeySlotMapping

▽

△

Number	Heading
[TPS_MANI_01524]	Semantics of the primitive attribute IdsmModuleInstantiationToCryptoKeySlotMapping.authenticatorFormat
[TPS_MANI_01525]	Semantics of DiagnosticInputOutputControlInterface
[TPS_MANI_01526]	Semantics of DiagnosticInputOutputControlGenericInterface
[TPS_MANI_01527]	Executing an I/O control request of the DM on the application software component
[TPS_MANI_01528]	Root element of the SOVD system design
[TPS_MANI_01529]	Access to diagnostic content from within a SovdAbstractContributingEntity
[TPS_MANI_01530]	Usage of SovdAbstractContributingEntity.dependsOn
[TPS_MANI_01531]	Semantics of SovdServerInstantiation
[TPS_MANI_01532]	Length of a fixed size string
[TPS_MANI_03339]	Semantics of IdsFunctionalClusterDesign
[TPS_MANI_03340]	local ApApplicationEndpoint and NetworkEndpoint for transmission of security events
[TPS_MANI_03341]	Semantics of SovdFunctionalClusterDesign
[TPS_MANI_03342]	Semantics of SovdServerDesign
[TPS_MANI_03343]	Semantics of SovdGatewayDesign
[TPS_MANI_03344]	Possible interval of the attribute DoIpLogicalAddress.logicalAddress
[TPS_MANI_03345]	Semantics of IdsmModuleInstantiation.reportableSecurityEvent
[TPS_MANI_03346]	Number of instances of SovdModuleInstantiation on a Machine
[TPS_MANI_03347]	Semantics of MachineDesign.clientIdRange
[TPS_MANI_03349]	Semantics of the attribute ServiceInstanceToPortPrototypeMapping.clientId

Table F.74: Added Specification Items in R26-11

F.13.2 Changed Specification Items in R26-11

Number	Heading
[TPS_MANI_01013]	Semantics of meta-class StateDependentStartupConfig
[TPS_MANI_01061]	Requirements on scheduling
[TPS_MANI_01188]	Semantics of attribute schedulingPriority
[TPS_MANI_01209]	Definition of environment variables in process scope
[TPS_MANI_01232]	Semantics of meta-class ComOfferServiceGrantDesign
[TPS_MANI_01255]	Semantics of DoIP DiagnosticPortInterfaces
[TPS_MANI_01277]	Definition of a start-up timeout for a StateDependentStartupConfig of a Process

▽

△

Number	Heading
[TPS_MANI_01278]	Definition of a termination timeout for a StateDependentStartupConfig of a Process
[TPS_MANI_01308]	Process is not designed for re-usability
[TPS_MANI_01328]	Standardized values for attribute StateDependentStartupConfig.schedulingPolicy
[TPS_MANI_01334]	Semantics of attribute StateDependentStartupConfig.terminationBehavior
[TPS_MANI_01405]	Semantics of sub-classes of meta-class SoftwareClusterDiagnosticAddress
[TPS_MANI_01478]	Supported error handling scenarios in the context of state management
[TPS_MANI_02388]	Semantics of DltLogSinkToPortPrototypeMapping
[TPS_MANI_03139]	Standardized SecOC Security Profiles
[TPS_MANI_03158]	Configuration of a data collection on a ProvidedSomeipServiceInstance for transmission over udp
[TPS_MANI_03271]	Semantics of ComSecOcToCryptoKeySlotMapping
[TPS_MANI_03282]	Assignment of a Dlt Ecu Identifier to a LogAndTraceInstantiation
[TPS_MANI_03297]	Semantics of AdaptiveFirewallToPortPrototypeMapping
[TPS_MANI_03298]	Semantics of StateDependentFirewall

Table F.75: Changed Specification Items in R26-11

F.13.3 Deleted Specification Items in R26-11

Number	Heading
[TPS_MANI_01399]	Modeling of the configuration of Service-oriented Vehicle Diagnostics
[TPS_MANI_01462]	Trigger the DoIP vehicle announcement at the diagnostic server instance
[TPS_MANI_01467]	Modeling of application that uses a Crypto Key
[TPS_MANI_03165]	Network Interface configuration for DoIP
[TPS_MANI_03254]	Modeling of application that uses and modifies a Crypto Key
[TPS_MANI_03255]	Modeling of Key Manager application that manages a Crypto Key that is used by Stack Services
[TPS_MANI_03259]	Linking of Crypto Certificate to a Crypto Key Slot
[TPS_MANI_03331]	local ApApplicationEndpoint and NetworkEndpoint for transmission of security events

Table F.76: Deleted Specification Items in R26-11

F.13.4 Added Constraints in R26-11

Number	Heading
[constr_01523]	Existence of the role IdsmModuleInstantiationToCryptoKeySlotMapping.idsmModuleInstantiation
[constr_01524]	Existence of the role IdsmModuleInstantiationToCryptoKeySlotMapping.cryptoKeySlot
[constr_10736]	Restriction applicable for CryptoCertificateGroupToPortPrototypeMapping.portPrototype
[constr_10737]	Existence of reference in the role CryptoCertificateGroupToPortPrototypeMapping.process
[constr_10738]	Existence of reference in the role CryptoCertificateGroupToPortPrototypeMapping.certificateGroup
[constr_10739]	Existence of instanceRef in the role CryptoCertificateGroupToPortPrototypeMapping.portPrototype
[constr_10742]	Reference to DiagnosticSovdContent
[constr_10743]	Existence of TransformationPropsToServiceInterfaceElementMapping.methodReturn
[constr_10746]	Existence of primitive attribute CryptoKeySlotToClientPortPrototypeMapping.allowPlainTextImport
[constr_10747]	Simultaneous existence of Process.posixUser and Process.primaryGroup
[constr_10748]	Existence of PosixUser.uid
[constr_10749]	Existence of PosixGroup.gid
[constr_10753]	Multiplicity of attribute SoftwareClusterUdsDiagnosticAddress.udsAddressSemantics
[constr_10754]	Restriction applicable for CryptoKeySlotToPortPrototypeMapping.portPrototype
[constr_10755]	Restriction applicable for CryptoKeySlotToClientPortPrototypeMapping.portPrototype
[constr_10779]	Reference to NM Network Handle from NmNetworkHandleErrorMapping
[constr_10780]	Uniqueness of the value of DiagnosticDoIPCustomPayloadRxPortMapping.payloadTypeId
[constr_10781]	Restriction for the applicability of DiagnosticDoIPCustomPayloadRxPortMapping
[constr_10782]	Restriction for the applicability of DiagnosticDoIPCustomPayloadRxInterface
[constr_10783]	Existence of DiagnosticDoIPCustomPayloadRxPortMapping.pPortPrototypeInExecutable
[constr_10784]	Existence of DiagnosticDoIPCustomPayloadAbstractPortMapping.process
[constr_10785]	Existence of DiagnosticDoIPCustomPayloadAbstractPortMapping.payloadTypeId
[constr_10786]	Restriction for the applicability of DiagnosticDoIPCustomPayloadTxPortMapping

▽

△

Number	Heading
[constr_10787]	Restriction for the applicability of DiagnosticDoIPCustomPayloadTxInterface
[constr_10788]	Existence of reference DiagnosticDoIPCustomPayloadTxPort-Mapping.rPortPrototypeInExecutable
[constr_10789]	Existence of reference DiagnosticDoIPCustomPayloadTxPortMapping.process
[constr_10790]	Existence of reference DiagnosticDoIPCustomPayloadTxPortMapping.payloadTypeId
[constr_10791]	Existence of DiagnosticSovdMethodPrimitive in the context of a DiagnosticSovdFaultMemoryAccess
[constr_10796]	Restriction for the applicability of DiagnosticInputOutputControlPortMapping
[constr_10797]	Reference in the role SovdAbstractContributingEntity.softwareClusterDesign
[constr_3825]	ProvidedSomeipServiceInstances with SomeipEventDeployment.serializer = signalBased shall be unique with respect to Message ID on one VLAN
[constr_9429]	Upper multiplicity of reference in the role IdsFunctionalClusterDesign.networkInterface
[constr_9430]	Existence of DoIpInstantiation.doIpDesign
[constr_9431]	Existence of SovdServerInstantiation.sovdServerDesign
[constr_9432]	Existence of SovdGatewayInstantiation.sovdGatewayDesign
[constr_9433]	Existence of reference IdsPlatformInstantiation.idsDesign
[constr_9434]	Overlap of ClientIdRanges in the context of the enclosing System
[constr_9435]	ServiceInstanceToPortPrototypeMapping.clientId is only applicable to RPortPrototypes and RequiredApServiceInstances
[constr_9436]	Restriction of allowed values for ServiceInstanceToPortPrototypeMapping.clientId in case that a ClientIdRange is defined
[constr_9437]	ServiceInstanceToPortPrototypeMapping.clientId shall be unique in the context of a Machine
[constr_9438]	Partial configuration of clientIds for ServiceInterface users that are located on the same Machine is not allowed
[constr_9460]	Attribute fixedStringSize of ApSomeipTransformationProps not applicable for TransformationPropsToServiceInterfaceElementMapping
[constr_9461]	fixedStringSize is only applicable for DataPrototypes of category STRING

Table F.77: Added Constraints in R26-11

F.13.5 Changed Constraints in R26-11

Number	Heading
[constr_10248]	Multiplicity of attribute named `UcmDescription.identifier`
[constr_10249]	Multiplicity of attribute named `VehicleDriverNotification.approvalRequired`
[constr_10250]	Multiplicity of attribute named `VehicleDriverNotification.notificationState`
[constr_10380]	Target of `ArtifactLocator.representedModelElement`
[constr_10420]	Restriction for the existence of initial values for `PersistencyDataElement`
[constr_10483]	Applicable values of `baseTypeEncoding` in the context of the definition of a `DiagnosticDataElement`
[constr_10487]	Only one physical address per `DiagnosticCommonProps`
[constr_1692]	Value of `schedulingPriority`
[constr_3709]	`AliveSupervision.terminatingCheckpoint` required for self terminating `Process`es
[constr_5161]	`RequiredSomeipServiceInstance` that is mapped by a `SomeipServiceInstanceToMachineMapping` without a configured `tcpPort` and `udpPort`
[constr_5348]	Mandatory `initialMode` in `ModeDeclarationGroup` that is referenced by `StateDependentFirewall`
[constr_5358]	`AdaptiveFirewallToPortPrototypeMapping.rPort` restriction
[constr_5392]	Assignment of the same `event` to several `SomeipEventGroup`s on the same VLAN is forbidden if one of these `SomeipEventGroup`s has the `multicastThreshold` set to a value greater than 0

Table F.78: Changed Constraints in R26-11

F.13.6 Deleted Constraints in R26-11

Number	Heading
[constr_10176]	Multiplicity of attribute `StateDependentStartupConfig.startupConfig`
[constr_10236]	Multiplicity of attribute `SoftwareClusterDiagnosticAddress.SoftwareClusterDiagnosticAddress.addressSemantics`
[constr_10404]	Existence of `SoftwareClusterSovdAddress.componentQualifier`
[constr_10594]	Restriction for the applicability of `DiagnosticDoIpTriggerVehicleAnnouncementPortMapping`
[constr_10595]	Restriction for the applicability of `DiagnosticDoIPTriggerVehicleAnnouncementInterface`
[constr_10596]	Existence of `DiagnosticDoIpTriggerVehicleAnnouncementPortMapping.rPort-PrototypeInExecutable`
[constr_10597]	Existence of `DiagnosticDoIpTriggerVehicleAnnouncementPortMapping.process`

▽

△

Number	Heading
[constr_10628]	Allowed range of attribute `providerRateDeviationMax`
[constr_10650]	Target of the reference in the role `DiagnosticDoIpTriggerVehicleAnnouncementPortMapping.networkConfigurationDesign`
[constr_10651]	Existence of reference in the role `DiagnosticDoIpTriggerVehicleAnnouncementPortMapping.networkConfigurationDesign`
[constr_3492]	`DoIpInstantiation.logicalAddress` shall be defined as member in the `DoIpRequestConfiguration`
[constr_3498]	Supported value range for attribute `DoIpInstantiation.logicalAddress`
[constr_3728]	Upper multiplicity of reference in the role `networkInterface`
[constr_3734]	Upper multiplicity of reference in the role `DoIpNetworkConfiguration.networkConfiguration`
[constr_5238]	`CryptoKeySlotAllowedModification.CryptoKeySlotAllowedModification.restrictUpdate` and the relationship to `CryptoKeySlotAllowedModification.maxNumberOfAllowedUpdates`
[constr_5239]	Predefined values for `CryptoKeySlotContentAllowedUsage.CryptoKeySlotContentAllowedUsage.allowedKeyslotUsage`
[constr_5241]	Restriction applicable for `AbstractCryptoKeySlotToPortPrototypeMapping.portPrototype`

Table F.79: Deleted Constraints in R26-11

F.13.7 Added Advisories in R26-11

none

F.13.8 Changed Advisories in R26-11

Number	Heading
[advisory_01017]	Existence of the `IdsFunctionalClusterDesign.networkInterface` if `IdsmModuleInstantiation.reportableSecurityEvent` exists

Table F.80: Changed Advisories in R26-11

F.13.9 Deleted Advisories in R26-11

Number	Heading
[advisory_01013]	Existence of the reference in the role `CryptoKeySlotUserDesignMapping.processDesign`

Table F.81: Deleted Advisories in R26-11

G Non-Splitable Elements in the Scope of this Document

This chapter contains a table of all non-splitable model elements in the scope of this document.

For more information about the concept of splitable model elements and how these shall be treated please refer to the [8, TPS Generic Structure Template].

Name of non splitable element
AbstractMethodInExecutableInstanceRef.base
AbstractMethodInExecutableInstanceRef.contextComponentPrototype
AbstractMethodInExecutableInstanceRef.contextPortPrototype
AbstractMethodInExecutableInstanceRef.contextRootSwComponentPrototype
AbstractMethodInExecutableInstanceRef.targetMethod
AbstractPortPrototypeInExecutableInstanceRef.base
AbstractPortPrototypeInExecutableInstanceRef.contextComponentPrototype
AbstractPortPrototypeInExecutableInstanceRef.contextRootSwComponentPrototype
AbstractPortPrototypeInExecutableInstanceRef.targetPortPrototype
AbstractPortPrototypeInSoftwareClusterDesignInstanceRef.base
AbstractPortPrototypeInSoftwareClusterDesignInstanceRef.contextRootSwClusterDesignComponentPrototype
AbstractPortPrototypeInSoftwareClusterDesignInstanceRef.contextSwComponentPrototype
AbstractPortPrototypeInSoftwareClusterDesignInstanceRef.targetPortPrototype
Allocator.namespace
ApApplicationErrorDomain.namespace
ApplicationAssocMapElementValueSpecification.key
ApplicationAssocMapElementValueSpecification.value
ApplicationAssocMapValueSpecification.mapElementTuple
AutosarDataPrototypeInExecutableInstanceRef.base
AutosarDataPrototypeInExecutableInstanceRef.contextPortPrototype
AutosarDataPrototypeInExecutableInstanceRef.contextRootSwComponentPrototype
AutosarDataPrototypeInExecutableInstanceRef.contextSwComponentPrototype
AutosarDataPrototypeInExecutableInstanceRef.targetAutosarDataPrototype
CppImplementationDataType.namespace
CppImplementationDataType.subElement
CppImplementationDataType.templateArgument
CppTemplateArgument.allocator
CppTemplateArgument.templateType
DataPrototypeInServiceInterfaceInstanceRef.base
DataPrototypeInServiceInterfaceInstanceRef.contextDataPrototype
DataPrototypeInServiceInterfaceInstanceRef.rootDataPrototype
DataPrototypeInServiceInterfaceInstanceRef.targetDataPrototype
DiagnosticServiceValidationConfiguration.manufacturerValidationOrder
DiagnosticServiceValidationConfiguration.sovdValidationOrder
DiagnosticServiceValidationConfiguration.supplierValidationOrder

▽

△

Name of non splitable element
EventInExecutableInstanceRef.contextRootSwComponentPrototype
EventInExecutableInstanceRef.contextRPortPrototype
EventInExecutableInstanceRef.contextSwComponentPrototype
EventInExecutableInstanceRef.targetEvent
FieldInExecutableInstanceRef.contextRootSwComponentPrototype
FieldInExecutableInstanceRef.contextRPortPrototype
FieldInExecutableInstanceRef.contextSwComponentPrototype
FieldInExecutableInstanceRef.targetField
FirewallRuleProps.matchingEgressRule
FirewallRuleProps.matchingIngressRule
FirewallStateInFirewallStateSwitchInterfaceInstanceRef.base
FirewallStateInFirewallStateSwitchInterfaceInstanceRef.contextModeDeclarationGroupPrototype
FirewallStateInFirewallStateSwitchInterfaceInstanceRef.targetModeDeclaration
FunctionGroupStateInFunctionGroupSetInstanceRef.base
FunctionGroupStateInFunctionGroupSetInstanceRef.contextModeDeclarationGroupPrototype
FunctionGroupStateInFunctionGroupSetInstanceRef.targetModeDeclaration
ModeDeclarationInStateManagementStateNotificationInstanceRef.base
ModeDeclarationInStateManagementStateNotificationInstanceRef.contextModeDeclarationGroupPrototype
ModeDeclarationInStateManagementStateNotificationInstanceRef.targetModeDeclaration
ModeInProcessInstanceRef.base
ModeInProcessInstanceRef.contextModeDeclarationGroupPrototype
ModeInProcessInstanceRef.targetModeDeclaration
PersistencyDataRequiredComSpec.dataElement
PersistencyDeployment.deploymentUri
PhmCheckpointInExecutableInstanceRef.base
PhmCheckpointInExecutableInstanceRef.contextComponentPrototype
PhmCheckpointInExecutableInstanceRef.contextRootSwComponentPrototype
PhmCheckpointInExecutableInstanceRef.contextRPortPrototype
PhmCheckpointInExecutableInstanceRef.targetPhmCheckpoint
PortInterfaceElementInImplementationDatatypeRef.contextDataPrototype
PortPrototypeInExecutableInstanceRef.base
PortPrototypeInExecutableInstanceRef.contextComponentPrototype
PortPrototypeInExecutableInstanceRef.contextRootSwComponentPrototype
PortPrototypeInExecutableInstanceRef.targetPortPrototype
PPortPrototypeInExecutableInstanceRef.contextComponentPrototype
PPortPrototypeInExecutableInstanceRef.contextRootSwComponentPrototype
PPortPrototypeInExecutableInstanceRef.targetPPortPrototype
PPortPrototypeInSoftwareClusterDesignInstanceRef.base
PPortPrototypeInSoftwareClusterDesignInstanceRef.contextRootSwClusterDesignComponentPrototype
PPortPrototypeInSoftwareClusterDesignInstanceRef.contextSwComponentPrototype
PPortPrototypeInSoftwareClusterDesignInstanceRef.targetPPortPrototype
ProvidedSomeipServiceInstance.capabilityRecord
RequiredMethodInExecutableInstanceRef.contextComponentPrototype

▽

△

Name of non splitable element
RequiredMethodInExecutableInstanceRef.contextRootSwComponentPrototype
RequiredMethodInExecutableInstanceRef.contextRPortPrototype
RequiredMethodInExecutableInstanceRef.targetMethod
RequiredSomeipServiceInstance.capabilityRecord
RPortPrototypeInExecutableInstanceRef.contextComponentPrototype
RPortPrototypeInExecutableInstanceRef.contextRootSwComponentPrototype
RPortPrototypeInExecutableInstanceRef.targetRPortPrototype
RPortPrototypeInSoftwareClusterDesignInstanceRef.base
RPortPrototypeInSoftwareClusterDesignInstanceRef.contextRootSwClusterDesignComponentPrototype
RPortPrototypeInSoftwareClusterDesignInstanceRef.contextSwComponentPrototype
RPortPrototypeInSoftwareClusterDesignInstanceRef.targetRPortPrototype
SoftwareClusterDependencyCompareCondition.softwareClusterDesign
SoftwareClusterDependencyFormula.part
StartupConfig.processArgument
StateDependentFirewall.firewallRuleProps
StateDependentStartupConfig.processArgument
StateManagementActionList.actionItem
StateManagementCompareCondition.compareValue
StateManagementCompareFormula.part
StateManagementTriggerCompareRule.assumedCurrentState
TraceSwitchConfig.namespace
TriggerInExecutableInstanceRef.base
TriggerInExecutableInstanceRef.contextComponentPrototype
TriggerInExecutableInstanceRef.contextRootSwComponentPrototype
TriggerInExecutableInstanceRef.contextRPortPrototype
TriggerInExecutableInstanceRef.targetTrigger
UcmStep.softwarePackageStep
VehiclePackage.rolloutQualification
VehiclePackage.ucmMasterFallback

Table G.1: Usage of non splitable elements

H Variation Points in the Scope of this Document

This chapter contains a table of all model elements stereotyped ≪atpVariation≫ in the scope of this document.

Each entry in the table consists of the identification of the model element itself and the applicable value of the tagged value vh.latestBindingTime.

For more information about the concept of variation points and how model elements that contain variation points shall be treated please refer to [8, TPS Generic Structure Template].

Variation Point	*Latest Binding Time*
AdaptiveApplicationSwComponentType.internalBehavior	preCompileTime
CppImplementationDataType.arraySize	preCompileTime
ServiceInterface.event	blueprintDerivationTime
ServiceInterface.field	blueprintDerivationTime
ServiceInterface.method	blueprintDerivationTime
ServiceInterface.trigger	blueprintDerivationTime
ServiceInterfaceDeployment.eventDeployment	blueprintDerivationTime
ServiceInterfaceDeployment.fieldDeployment	blueprintDerivationTime
ServiceInterfaceDeployment.methodDeployment	blueprintDerivationTime

Table H.1: Usage of variation points

I Upstream Mapping

The content of this appendix chapter is *informative* in nature and shall **not** be considered as *normative* content.

I.1 Introduction

This chapter describes the mapping of the APMC Configuration parameters (M1 model) onto the meta-classes and attributes of the AUTOSAR upstream templates (in this case: Manifest).

The relationships between upstream templates and APMC Configuration are described in order to answer typical questions like:

- How shall a supplier use the information in a Design Description in order to fulfill the needs defined by the systems engineer?
- How is a tool vendor supposed to generate an APMC Configuration Description out of a Design Description?

Please note that the upstream mapping tables contain the following columns:

Column Name	Column Meaning
Functional Cluster	Name of Functional Cluster
Functional Cluster Context	Reference to parameter container
APMC Parameter	Name of the APMC parameter
Kind of APMC Parameter	M2 Class used to type the parameter
APMC Description	Description from the configuration document
Template Description	Class note or attribute note of the M2 model element
M2 Parameter	Name of the upstream template model element
Mapping Rule	Textual description on how to transform between Design and Target Configuration domains
Mapping Type	One of: **local** no mapping needed since parameter local to Target Configuration **partial** some data can be automatically mapped but not all **full** all data can be automatically mapped
Mapping Status	Indication of life-cycle status of the mapping
APMC Parameter ID	ID of the parameter in the respective SWS document (may be empty if the mapping is owned by an enumeration literal)

Table I.1: Upstream mapping table content

I.2 CryptoInstantiation

Functional Cluster	ShortName Path of Parameter Definition
CryptoInstantiation	CryptoInstantiation/CryptoCertificateCollection
APMC Parameter	**Kind of APMC Parameter**
CryptoCertificate	APMC-PARAM-CONF-CONTAINER-DEF
APMC Description	
This container represents the ability to model a cryptographic certificate.	
Template Description	
This meta-class provides the ability to define a PortInterface for a CryptoCertificate.	
M2 Parameter	
CryptoCertificateInterface	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_CRYPT_00006]

Functional Cluster	ShortName Path of Parameter Definition
CryptoInstantiation	CryptoInstantiation/CryptoProviderCollection/CryptoProvider/CryptoKeySlotCollection
APMC Parameter	**Kind of APMC Parameter**
CryptoKeySlot	APMC-PARAM-CONF-CONTAINER-DEF
APMC Description	
This meta-class represents the ability to define a concrete key to be used for a crypto operation.	
Template Description	
This abstract meta-class acts as a base class for crypto key slot port interfaces.	
M2 Parameter	
AbstractCryptoKeySlotInterface	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_CRYPT_00010]

Functional Cluster	ShortName Path of Parameter Definition
CryptoInstantiation	CryptoInstantiation/CryptoProviderCollection/CryptoProvider/CryptoKeySlotCollection/CryptoKeySlot/AlgorithmDescriptionCollection/AlgorithmDescription/CryptoKeyType
APMC Parameter	**Kind of APMC Parameter**
PRIVATE_KEY	APMC-ENUMERATION-LITERAL-DEF
APMC Description	
This denotes a private key.	
Template Description	
cryp::PrivateKey object	
M2 Parameter	
CryptoObjectTypeEnum.privateKey	
Mapping Rule	**Mapping Type**

▽

△

1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	

Functional Cluster	**ShortName Path of Parameter Definition**
CryptoInstantiation	CryptoInstantiation/CryptoProviderCollection/CryptoProvider/CryptoKeySlotCollection/CryptoKeySlot/AlgorithmDescriptionCollection/AlgorithmDescription/CryptoKeyType
APMC Parameter	**Kind of APMC Parameter**
PUBLIC_KEY	APMC-ENUMERATION-LITERAL-DEF
APMC Description	
This denotes a public key.	
Template Description	
cryp::PublicKey object	
M2 Parameter	
`CryptoObjectTypeEnum.publicKey`	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	

Functional Cluster	**ShortName Path of Parameter Definition**
CryptoInstantiation	CryptoInstantiation/CryptoProviderCollection/CryptoProvider/CryptoKeySlotCollection/CryptoKeySlot/AlgorithmDescriptionCollection/AlgorithmDescription/CryptoKeyType
APMC Parameter	**Kind of APMC Parameter**
SYMMETRIC_KEY	APMC-ENUMERATION-LITERAL-DEF
APMC Description	
This denotes a symmetric key.	
Template Description	
cryp::SymmetricKey object	
M2 Parameter	
`CryptoObjectTypeEnum.symmetricKey`	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	

Functional Cluster	**ShortName Path of Parameter Definition**
CryptoInstantiation	CryptoInstantiation/CryptoProviderCollection/CryptoProvider/CryptoKeySlotCollection/CryptoKeySlot
APMC Parameter	**Kind of APMC Parameter**
CryptoServiceKeyForeignRef	APMC-FOREIGN-REFERENCE-DEF
APMC Description	
This reference identifies a corresponding `CryptoServiceKey` for the purpose of interaction between functional clusters, e.g. for SecOC Communication.	
Template Description	

▽

△

This meta-class has the ability to represent a crypto key.	
M2 Parameter	
CryptoServiceKey	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_CRYPT_00045]

I.3 Persistency

Functional Cluster	**ShortName Path of Parameter Definition**	
Persistency	Persistency/PersistencyFileStorageCollection	
APMC Parameter	**Kind of APMC Parameter**	
PersistencyFileStorage	APMC-PARAM-CONF-CONTAINER-DEF	
APMC Description		
This container represents the configuration of one file storage in the context of the enclosing file storage collection.		
Template Description		
This meta-class provides the ability to implement a PortInterface for supporting persistency use cases for files.		
M2 Parameter		
PersistencyFileStorageInterface		
Mapping Rule		**Mapping Type**
1:1 mapping		full
Mapping Status		**APMC Parameter ID**
valid		[APMC_PER_00005]

Functional Cluster	**ShortName Path of Parameter Definition**	
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage	
APMC Parameter	**Kind of APMC Parameter**	
MaxNumberOfFiles	APMC-INTEGER-PARAM-DEF	
APMC Description		
Upper bound for the number of files in this PersistencyFileStorage. If not configured, no limit is imposed.		
Template Description		
This attribute represents the definition of an upper bound for the handling of files at run-time in the context of the enclosing PersistencyFileStorageInterface.		
M2 Parameter		
PersistencyFileStorageInterface.maxNumberOfFiles		
Mapping Rule		**Mapping Type**
1:1 mapping		full
Mapping Status		**APMC Parameter ID**
valid		[APMC_PER_00053]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyCommon
APMC Parameter	**Kind of APMC Parameter**
ContractVersion	APMC-STRONG-REVISION-LABEL-PARAM-DEF
APMC Description	
This parameter represents the contract version. This version changes if a semantical change happens. The existence of this parameter is mutually exclusive to the existence of PortPrototypeInstanceRef.	
Template Description	
This attribute represents the contract version that is used to determine whether the Persistency configuration experienced structural changes and is also used for the check for data type compatibility.	
M2 Parameter	
PersistencyInterface.contractVersion	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00046]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyCommon
APMC Parameter	**Kind of APMC Parameter**
MinimumSustainedSize	APMC-INTEGER-PARAM-DEF
APMC Description	
This configuration parameter shall be used to specify the global update strategy of the respective key-value storage or file storage.	
Template Description	
The value of this attribute represents the minimum size (unit: bytes) required at design time for the enclosing Persistency Interface.	
M2 Parameter	
PersistencyInterface.minimumSustainedSize	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00008]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyCommon/ PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/Persistency RedundancyHandlingChoice
APMC Parameter	**Kind of APMC Parameter**
PersistencyRedundancyChecksum	APMC-PARAM-CONF-CONTAINER-DEF
APMC Description	
This container provides the ability to describe redundancy via a checksum approach.	
Template Description	
Abstract class that defines the common attributes for implementations of redundancy.	
M2 Parameter	
PersistencyRedundancyChecksum	
Mapping Rule	**Mapping Type**



△

1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00030]

Functional Cluster	**ShortName Path of Parameter Definition**
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyCommon/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/PersistencyRedundancyHandlingChoice/PersistencyRedundancyChecksum/PersistencyRedundancyChecksumChoice
APMC Parameter	**Kind of APMC Parameter**
PersistencyRedundancyCrc	APMC-PARAM-CONF-CONTAINER-DEF
APMC Description	
This container is used to configure a CRC-based ckecksum.	
Template Description	
This meta-class formally describes the usage of a CRC for the implementation of redundancy.	
M2 Parameter	
PersistencyRedundancyCrc	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00032]

Functional Cluster	**ShortName Path of Parameter Definition**
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyCommon/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/PersistencyRedundancyHandlingChoice/PersistencyRedundancyChecksum/PersistencyRedundancyChecksumChoice
APMC Parameter	**Kind of APMC Parameter**
PersistencyRedundancyHash	APMC-PARAM-CONF-CONTAINER-DEF
APMC Description	
This container is used to configure a hash-based ckecksum.	
Template Description	
This meta-class formally describes the usage of a Hash for the implementation of redundancy.	
M2 Parameter	
PersistencyRedundancyHash	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00033]

Functional Cluster	**ShortName Path of Parameter Definition**
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyCommon/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/PersistencyRedundancyHandlingChoice/PersistencyRedundancyChecksum/PersistencyRedundancyChecksumChoice/PersistencyRedundancyHash
APMC Parameter	**Kind of APMC Parameter**
InitializationVectorLength	APMC-INTEGER-PARAM-DEF

▽

△

APMC Description	
Length of the initialization vector in bits.	
Template Description	
Length of the initialization vector.	
M2 Parameter	
PersistencyRedundancyHash.initializationVectorLength	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00034]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyCommon/ PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/Persistency RedundancyHandlingChoice/PersistencyRedundancyChecksum
APMC Parameter	**Kind of APMC Parameter**
RedundancyChecksumAlgorithmFamily	APMC-STRING-PARAM-DEF
APMC Description	
This attribute identifies the algorithm family that is used to execute the CRC/Hash.	
Template Description	
This attribute identifies the algorithm family that is used to execute the CRC/Hash.	
M2 Parameter	
PersistencyRedundancyChecksum.algorithmFamily	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00038]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyCommon/ PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/Persistency RedundancyHandlingChoice/PersistencyRedundancyChecksum
APMC Parameter	**Kind of APMC Parameter**
RedundancyChecksumLength	APMC-INTEGER-PARAM-DEF
APMC Description	
This parameter describes the length of the CRC/Hash in the unit bits.	
Template Description	
This attribute describes the length of the CRC/Hash in the unit bits.	
M2 Parameter	
PersistencyRedundancyChecksum.length	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00037]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyCommon/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/PersistencyRedundancyHandlingChoice
APMC Parameter	**Kind of APMC Parameter**
PersistencyRedundancyMOutOfN	APMC-PARAM-CONF-CONTAINER-DEF
APMC Description	
This container 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 required for a reliable read access to the data.	
Template Description	
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.	
M2 Parameter	
PersistencyRedundancyMOutOfN	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00027]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyCommon/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/PersistencyRedundancyHandlingChoice/PersistencyRedundancyMOutOfN
APMC Parameter	**Kind of APMC Parameter**
RedundancyM	APMC-INTEGER-PARAM-DEF
APMC Description	
This configuration parameter represents the "M" coordinate in the "M out of N" scheme.	
Template Description	
This attribute represents the "M" coordinate in the "M out of N" scheme.	
M2 Parameter	
PersistencyRedundancyMOutOfN.m	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00028]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyCommon/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/PersistencyRedundancyHandlingChoice/PersistencyRedundancyMOutOfN
APMC Parameter	**Kind of APMC Parameter**
RedundancyN	APMC-INTEGER-PARAM-DEF
APMC Description	
This configuration parameter represents the "N" coordinate in the "M out of N" scheme.	
Template Description	
This attribute represents the "N" coordinate in the "M out of N" scheme.	
M2 Parameter	

▽

△

PersistencyRedundancyMOutOfN.n	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00029]

Functional Cluster	**ShortName Path of Parameter Definition**
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyCommon/ PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling
APMC Parameter	**Kind of APMC Parameter**
PersistencyRedundancyHandlingScope	APMC-ENUMERATION-PARAM-DEF
APMC Description	
This parameter controls the scope on which the redundancy handling is applied.	
Template Description	
This attribute controls the scope in which the redundancy handling is applied.	
M2 Parameter	
PersistencyRedundancyHandling.scope	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00025]

Functional Cluster	**ShortName Path of Parameter Definition**
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyCommon/ PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/Persistency RedundancyHandlingScope
APMC Parameter	**Kind of APMC Parameter**
SCOPE_ELEMENT	APMC-ENUMERATION-LITERAL-DEF
APMC Description	
The redundancy handling shall be applied on element level (for all key-value pairs and files).	
Template Description	
The redundancy handling shall be applied on element level (key-value pair and file).	
M2 Parameter	
PersistencyRedundancyHandlingScopeEnum.persistencyRedundancyHandlingScopeElement	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	

Functional Cluster	**ShortName Path of Parameter Definition**
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyCommon/ PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/Persistency RedundancyHandlingScope
APMC Parameter	**Kind of APMC Parameter**
SCOPE_STORAGE	APMC-ENUMERATION-LITERAL-DEF

▽

△

APMC Description	
The redundancy handling shall be applied on storage level (for the whole key-value storage and file storage).	
Template Description	
The redundancy handling shall be applied on storage (key-value storage and file storage) level.	
M2 Parameter	
PersistencyRedundancyHandlingScopeEnum.persistencyRedundancyHandlingScopeStorage	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyCommon
APMC Parameter	**Kind of APMC Parameter**
UpdateStrategy	APMC-ENUMERATION-PARAM-DEF
APMC Description	
This configuration parameter shall be used to specify the global update strategy of the respective key-value storage or file storage.	
Template Description	
The update strategy is to keep the existing value of the respective data item.	
M2 Parameter	
PersistencyElementLevelUpdateStrategyEnum.keepExisting	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00010]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyCommon/Update Strategy
APMC Parameter	**Kind of APMC Parameter**
DELETE	APMC-ENUMERATION-LITERAL-DEF
APMC Description	
The update strategy is to delete the value of the respective data item.	
Template Description	
PersistencyElementLevelUpdateStrategyEnum.delete: The update strategy is to delete the value of the respective data item. **PersistencyCollectionLevelUpdateStrategyEnum.delete:** The update strategy is to delete all values on the level of the respective collection.	
M2 Parameter	
PersistencyElementLevelUpdateStrategyEnum.delete, PersistencyCollectionLevelUpdateStrategyEnum.delete	
Mapping Rule	**Mapping Type**

▽

△

If DELETE is aggregated on element level, then the upstream mapping to PersistencyElement LevelStrategyEnum applies. If DELETE is aggregated on collection level, then the upstream mapping to PersistencyCollection LevelStrategyEnum applies.	full
Mapping Status	**APMC Parameter ID**
valid	

Functional Cluster	**ShortName Path of Parameter Definition**
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyCommon/UpdateStrategy

APMC Parameter	**Kind of APMC Parameter**
KEEP_EXISTING	APMC-ENUMERATION-LITERAL-DEF

APMC Description
The update strategy is to keep the existing value of the respective data item.
Template Description
PersistencyElementLevelUpdateStrategyEnum.keepExisting: The update strategy is to keep the existing value of the respective data item. **PersistencyCollectionLevelUpdateStrategyEnum.keepExisting:** The update strategy is to keep the existing values on the level of the respective collection.
M2 Parameter
`PersistencyElementLevelUpdateStrategyEnum.keepExisting`, `PersistencyCollectionLevelUpdateStrategyEnum.keepExisting`

Mapping Rule	**Mapping Type**
If KEEP_EXISTING is aggregated on element level, then the upstream mapping to PersistencyElementLevelStrategyEnum applies. If KEEP_EXISTING is aggregated on collection level, then the upstream mapping to PersistencyCollectionLevelStrategyEnum applies.	full
Mapping Status	**APMC Parameter ID**
valid	

Functional Cluster	**ShortName Path of Parameter Definition**
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage

APMC Parameter	**Kind of APMC Parameter**
PersistencyFileCollection	APMC-PARAM-CONF-CONTAINER-DEF

APMC Description
This container represents the collection of files in the context of the enclosing file storage.
Template Description
This aggregation represents the collection of PersistencyFileStorages in the context of the enclosing PersistencyFileStorage Interface.
M2 Parameter
`PersistencyFileStorageInterface.fileElement`

Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00015]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyFileCollection
APMC Parameter	**Kind of APMC Parameter**
PersistencyFile	APMC-PARAM-CONF-CONTAINER-DEF
APMC Description	
This container represents a file in the context of the enclosing file collection.	
Template Description	
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.	
M2 Parameter	
PersistencyFileElement	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00016]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyFileCollection/ PersistencyFile
APMC Parameter	**Kind of APMC Parameter**
ContentUri	APMC-STRING-PARAM-DEF
APMC Description	
This attribute represents the URI that identifies the initial content of the PersistencyFile.	
Template Description	
This attribute represents the URI that identifies the initial content of the PersistencyFile.	
M2 Parameter	
PersistencyFileElement.contentUri	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00017]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyFileCollection/ PersistencyFile
APMC Parameter	**Kind of APMC Parameter**
ElementUpdateStrategy	APMC-ENUMERATION-PARAM-DEF
APMC Description	
This attribute can be used to specify the update strategy on element level, i.e. for a single key-value pair or file.	
Template Description	
This attribute can be used to specify the update strategy of the respective PersistencyInterfaceElement.	
M2 Parameter	
PersistencyInterfaceElement.updateStrategy	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00036]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyFileCollection/PersistencyFile/ElementUpdateStrategy

APMC Parameter	Kind of APMC Parameter
DELETE	APMC-ENUMERATION-LITERAL-DEF

APMC Description

The update strategy is to delete the value of the respective data item.

Template Description

PersistencyElementLevelUpdateStrategyEnum.delete:
The update strategy is to delete the value of the respective data item.

PersistencyCollectionLevelUpdateStrategyEnum.delete:
The update strategy is to delete all values on the level of the respective collection.

M2 Parameter

`PersistencyElementLevelUpdateStrategyEnum.delete`,
`PersistencyCollectionLevelUpdateStrategyEnum.delete`

Mapping Rule	Mapping Type
If DELETE is aggregated on element level, then the upstream mapping to PersistencyElement LevelStrategyEnum applies. If DELETE is aggregated on collection level, then the upstream mapping to PersistencyCollection LevelStrategyEnum applies.	full
Mapping Status	**APMC Parameter ID**
valid	

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyFileCollection/PersistencyFile/ElementUpdateStrategy

APMC Parameter	Kind of APMC Parameter
KEEP_EXISTING	APMC-ENUMERATION-LITERAL-DEF

APMC Description

The update strategy is to keep the existing value of the respective data item.

Template Description

PersistencyElementLevelUpdateStrategyEnum.keepExisting:
The update strategy is to keep the existing value of the respective data item.

PersistencyCollectionLevelUpdateStrategyEnum.keepExisting:
The update strategy is to keep the existing values on the level of the respective collection.

M2 Parameter

`PersistencyElementLevelUpdateStrategyEnum.keepExisting`,
`PersistencyCollectionLevelUpdateStrategyEnum.keepExisting`

Mapping Rule	Mapping Type
If KEEP_EXISTING is aggregated on element level, then the upstream mapping to Persistency ElementLevelStrategyEnum applies. If KEEP_EXISTING is aggregated on collection level, then the upstream mapping to Persistency CollectionLevelStrategyEnum applies.	full
Mapping Status	**APMC Parameter ID**
valid	

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyFileCollection/PersistencyFile/ElementUpdateStrategy



△

APMC Parameter	Kind of APMC Parameter
OVERWRITE	APMC-ENUMERATION-LITERAL-DEF
APMC Description	
The update strategy is to overwrite the value of this element.	
Template Description	
The update strategy is to overwrite the respective data item.	
M2 Parameter	
`PersistencyElementLevelUpdateStrategyEnum.overwrite`	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyFileStorageCollection/PersistencyFileStorage/PersistencyFileCollection/PersistencyFile
APMC Parameter	**Kind of APMC Parameter**
FileName	APMC-STRING-PARAM-DEF
APMC Description	
This attribute holds filename part of the storage location for the PersistencyFile, e.g. file on the file system.	
Template Description	
This attribute holds the filename part of the storage location, e.g. file on the file system.	
M2 Parameter	
`PersistencyFileElement.fileName`	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00018]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection
APMC Parameter	**Kind of APMC Parameter**
PersistencyKeyValueStorage	APMC-PARAM-CONF-CONTAINER-DEF
APMC Description	
This container represents the configuration of one key-value storage in the context of the enclosing key-value storage collection.	
Template Description	
This meta-class provides the ability to implement a `PortInterface` for supporting persistency use cases for data.	
M2 Parameter	
`PersistencyKeyValueStorageInterface`	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00004]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/Persistency Common
APMC Parameter	**Kind of APMC Parameter**
ContractVersion	APMC-STRONG-REVISION-LABEL-PARAM-DEF
APMC Description	
This parameter represents the contract version. This version changes if a semantical change happens. The existence of this parameter is mutually exclusive to the existence of PortPrototypeInstanceRef.	
Template Description	
This attribute represents the contract version that is used to determine whether the Persistency configuration experienced structural changes and is also used for the check for data type compatibility.	
M2 Parameter	
PersistencyInterface.contractVersion	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00046]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/Persistency Common
APMC Parameter	**Kind of APMC Parameter**
MinimumSustainedSize	APMC-INTEGER-PARAM-DEF
APMC Description	
This configuration parameter shall be used to specify the global update strategy of the respective key-value storage or file storage.	
Template Description	
The value of this attribute represents the minimum size (unit: bytes) required at design time for the enclosing Persistency Interface.	
M2 Parameter	
PersistencyInterface.minimumSustainedSize	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00008]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/Persistency Common/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/ PersistencyRedundancyHandlingChoice
APMC Parameter	**Kind of APMC Parameter**
PersistencyRedundancyChecksum	APMC-PARAM-CONF-CONTAINER-DEF
APMC Description	
This container provides the ability to describe redundancy via a checksum approach.	
Template Description	
Abstract class that defines the common attributes for implementations of redundancy.	
M2 Parameter	
PersistencyRedundancyChecksum	



△

Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00030]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyCommon/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/PersistencyRedundancyHandlingChoice/PersistencyRedundancyChecksum/PersistencyRedundancyChecksumChoice

APMC Parameter	Kind of APMC Parameter
PersistencyRedundancyCrc	APMC-PARAM-CONF-CONTAINER-DEF
APMC Description	
This container is used to configure a CRC-based ckecksum.	
Template Description	
This meta-class formally describes the usage of a CRC for the implementation of redundancy.	
M2 Parameter	
PersistencyRedundancyCrc	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00032]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyCommon/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/PersistencyRedundancyHandlingChoice/PersistencyRedundancyChecksum/PersistencyRedundancyChecksumChoice

APMC Parameter	Kind of APMC Parameter
PersistencyRedundancyHash	APMC-PARAM-CONF-CONTAINER-DEF
APMC Description	
This container is used to configure a hash-based ckecksum.	
Template Description	
This meta-class formally describes the usage of a Hash for the implementation of redundancy.	
M2 Parameter	
PersistencyRedundancyHash	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00033]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyCommon/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/PersistencyRedundancyHandlingChoice/PersistencyRedundancyChecksum/PersistencyRedundancyChecksumChoice/PersistencyRedundancyHash

▽

△

APMC Parameter	Kind of APMC Parameter
InitializationVectorLength	APMC-INTEGER-PARAM-DEF
APMC Description	
Length of the initialization vector in bits.	
Template Description	
Length of the initialization vector.	
M2 Parameter	
PersistencyRedundancyHash.initializationVectorLength	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00034]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyCommon/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/PersistencyRedundancyHandlingChoice/PersistencyRedundancyChecksum
APMC Parameter	**Kind of APMC Parameter**
RedundancyChecksumAlgorithmFamily	APMC-STRING-PARAM-DEF
APMC Description	
This attribute identifies the algorithm family that is used to execute the CRC/Hash.	
Template Description	
This attribute identifies the algorithm family that is used to execute the CRC/Hash.	
M2 Parameter	
PersistencyRedundancyChecksum.algorithmFamily	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00038]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyCommon/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/PersistencyRedundancyHandlingChoice/PersistencyRedundancyChecksum
APMC Parameter	**Kind of APMC Parameter**
RedundancyChecksumLength	APMC-INTEGER-PARAM-DEF
APMC Description	
This parameter describes the length of the CRC/Hash in the unit bits.	
Template Description	
This attribute describes the length of the CRC/Hash in the unit bits.	
M2 Parameter	
PersistencyRedundancyChecksum.length	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00037]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyCommon/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/PersistencyRedundancyHandlingChoice

APMC Parameter	Kind of APMC Parameter
PersistencyRedundancyMOutOfN	APMC-PARAM-CONF-CONTAINER-DEF

APMC Description
This container 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 required for a reliable read access to the data.

Template Description
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.

M2 Parameter
PersistencyRedundancyMOutOfN

Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00027]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyCommon/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/PersistencyRedundancyHandlingChoice/PersistencyRedundancyMOutOfN

APMC Parameter	Kind of APMC Parameter
RedundancyM	APMC-INTEGER-PARAM-DEF

APMC Description
This configuration parameter represents the "M" coordinate in the "M out of N" scheme.

Template Description
This attribute represents the "M" coordinate in the "M out of N" scheme.

M2 Parameter
PersistencyRedundancyMOutOfN.m

Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00028]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyCommon/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/PersistencyRedundancyHandlingChoice/PersistencyRedundancyMOutOfN

APMC Parameter	Kind of APMC Parameter
RedundancyN	APMC-INTEGER-PARAM-DEF

APMC Description
This configuration parameter represents the "N" coordinate in the "M out of N" scheme.

Template Description
This attribute represents the "N" coordinate in the "M out of N" scheme.

M2 Parameter

▽

△

PersistencyRedundancyMOutOfN.n	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00029]

Functional Cluster	**ShortName Path of Parameter Definition**
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/Persistency Common/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling
APMC Parameter	**Kind of APMC Parameter**
PersistencyRedundancyHandlingScope	APMC-ENUMERATION-PARAM-DEF
APMC Description	
This parameter controls the scope on which the redundancy handling is applied.	
Template Description	
This attribute controls the scope in which the redundancy handling is applied.	
M2 Parameter	
PersistencyRedundancyHandling.scope	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00025]

Functional Cluster	**ShortName Path of Parameter Definition**
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/Persistency Common/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/ PersistencyRedundancyHandlingScope
APMC Parameter	**Kind of APMC Parameter**
SCOPE_ELEMENT	APMC-ENUMERATION-LITERAL-DEF
APMC Description	
The redundancy handling shall be applied on element level (for all key-value pairs and files).	
Template Description	
The redundancy handling shall be applied on element level (key-value pair and file).	
M2 Parameter	
PersistencyRedundancyHandlingScopeEnum.persistencyRedundancyHandlingScopeElement	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	

Functional Cluster	**ShortName Path of Parameter Definition**
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/Persistency Common/PersistencyRedundancyHandlingCollection/PersistencyRedundancyHandling/ PersistencyRedundancyHandlingScope
APMC Parameter	**Kind of APMC Parameter**
SCOPE_STORAGE	APMC-ENUMERATION-LITERAL-DEF

▽

△

APMC Description	
The redundancy handling shall be applied on storage level (for the whole key-value storage and file storage).	
Template Description	
The redundancy handling shall be applied on storage (key-value storage and file storage) level.	
M2 Parameter	
PersistencyRedundancyHandlingScopeEnum.persistencyRedundancyHandlingScopeStorage	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyCommon
APMC Parameter	**Kind of APMC Parameter**
UpdateStrategy	APMC-ENUMERATION-PARAM-DEF
APMC Description	
This configuration parameter shall be used to specify the global update strategy of the respective key-value storage or file storage.	
Template Description	
The update strategy is to keep the existing value of the respective data item.	
M2 Parameter	
PersistencyElementLevelUpdateStrategyEnum.keepExisting	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00010]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyCommon/UpdateStrategy
APMC Parameter	**Kind of APMC Parameter**
DELETE	APMC-ENUMERATION-LITERAL-DEF
APMC Description	
The update strategy is to delete the value of the respective data item.	
Template Description	
PersistencyElementLevelUpdateStrategyEnum.delete: The update strategy is to delete the value of the respective data item. **PersistencyCollectionLevelUpdateStrategyEnum.delete:** The update strategy is to delete all values on the level of the respective collection.	
M2 Parameter	
PersistencyElementLevelUpdateStrategyEnum.delete, PersistencyCollectionLevelUpdateStrategyEnum.delete	
Mapping Rule	**Mapping Type**

▽

△

If DELETE is aggregated on element level, then the upstream mapping to PersistencyElement LevelStrategyEnum applies. If DELETE is aggregated on collection level, then the upstream mapping to PersistencyCollection LevelStrategyEnum applies.	full
Mapping Status	**APMC Parameter ID**
valid	

Functional Cluster	**ShortName Path of Parameter Definition**
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyCommon/UpdateStrategy

APMC Parameter	**Kind of APMC Parameter**
KEEP_EXISTING	APMC-ENUMERATION-LITERAL-DEF

APMC Description
The update strategy is to keep the existing value of the respective data item.
Template Description
PersistencyElementLevelUpdateStrategyEnum.keepExisting: The update strategy is to keep the existing value of the respective data item. **PersistencyCollectionLevelUpdateStrategyEnum.keepExisting:** The update strategy is to keep the existing values on the level of the respective collection.
M2 Parameter
`PersistencyElementLevelUpdateStrategyEnum.keepExisting`, `PersistencyCollectionLevelUpdateStrategyEnum.keepExisting`

Mapping Rule	**Mapping Type**
If KEEP_EXISTING is aggregated on element level, then the upstream mapping to Persistency ElementLevelStrategyEnum applies. If KEEP_EXISTING is aggregated on collection level, then the upstream mapping to Persistency CollectionLevelStrategyEnum applies.	full
Mapping Status	**APMC Parameter ID**
valid	

Functional Cluster	**ShortName Path of Parameter Definition**
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage

APMC Parameter	**Kind of APMC Parameter**
PersistencyKeyValuePairCollection	APMC-PARAM-CONF-CONTAINER-DEF

APMC Description
This container represents the collection of key-value pairs in the context of the enclosing key-value storage.
Template Description
This aggregation represents the collection of PersistencyDataElements in the context of the enclosing Persistency KeyValue StorageInterface.
M2 Parameter
`PersistencyKeyValueStorageInterface.dataElement`

Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00019]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyKeyValuePairCollection
APMC Parameter	**Kind of APMC Parameter**
PersistencyKeyValuePair	APMC-PARAM-CONF-CONTAINER-DEF
APMC Description	
This container represents a key-value pair in the context of the enclosing key-value collection.	
Template Description	
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.	
M2 Parameter	
PersistencyDataElement	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00020]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyKeyValuePairCollection/PersistencyKeyValuePair
APMC Parameter	**Kind of APMC Parameter**
ElementUpdateStrategy	APMC-ENUMERATION-PARAM-DEF
APMC Description	
This attribute can be used to specify the update strategy on element level, i.e. for a single key-value pair or file.	
Template Description	
This attribute can be used to specify the update strategy of the respective PersistencyInterfaceElement.	
M2 Parameter	
PersistencyInterfaceElement.updateStrategy	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00036]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyKeyValuePairCollection/PersistencyKeyValuePair/ElementUpdateStrategy
APMC Parameter	**Kind of APMC Parameter**
DELETE	APMC-ENUMERATION-LITERAL-DEF
APMC Description	
The update strategy is to delete the value of the respective data item.	
Template Description	
PersistencyElementLevelUpdateStrategyEnum.delete: The update strategy is to delete the value of the respective data item. **PersistencyCollectionLevelUpdateStrategyEnum.delete:** The update strategy is to delete all values on the level of the respective collection.	

▽

△

M2 Parameter	
PersistencyElementLevelUpdateStrategyEnum.delete, PersistencyCollectionLevelUpdateStrategyEnum.delete	
Mapping Rule	**Mapping Type**
If DELETE is aggregated on element level, then the upstream mapping to PersistencyElement LevelStrategyEnum applies. If DELETE is aggregated on collection level, then the upstream mapping to PersistencyCollection LevelStrategyEnum applies.	full
Mapping Status	**APMC Parameter ID**
valid	

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyKeyValuePairCollection/PersistencyKeyValuePair/ElementUpdateStrategy
APMC Parameter	**Kind of APMC Parameter**
KEEP_EXISTING	APMC-ENUMERATION-LITERAL-DEF
APMC Description	
The update strategy is to keep the existing value of the respective data item.	
Template Description	
PersistencyElementLevelUpdateStrategyEnum.keepExisting: The update strategy is to keep the existing value of the respective data item. **PersistencyCollectionLevelUpdateStrategyEnum.keepExisting:** The update strategy is to keep the existing values on the level of the respective collection.	
M2 Parameter	
PersistencyElementLevelUpdateStrategyEnum.keepExisting, PersistencyCollectionLevelUpdateStrategyEnum.keepExisting	
Mapping Rule	**Mapping Type**
If KEEP_EXISTING is aggregated on element level, then the upstream mapping to Persistency ElementLevelStrategyEnum applies. If KEEP_EXISTING is aggregated on collection level, then the upstream mapping to Persistency CollectionLevelStrategyEnum applies.	full
Mapping Status	**APMC Parameter ID**
valid	

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyKeyValuePairCollection/PersistencyKeyValuePair/ElementUpdateStrategy
APMC Parameter	**Kind of APMC Parameter**
OVERWRITE	APMC-ENUMERATION-LITERAL-DEF
APMC Description	
The update strategy is to overwrite the value of this element.	
Template Description	
The update strategy is to overwrite the respective data item.	
M2 Parameter	
PersistencyElementLevelUpdateStrategyEnum.overwrite	
Mapping Rule	**Mapping Type**
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyKeyValuePairCollection/PersistencyKeyValuePair

APMC Parameter	Kind of APMC Parameter
InitValueRef	APMC-FOREIGN-REFERENCE-DEF

APMC Description
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 configuration parameter UpdateStrategy within the container PersistencyCommon is set to the value DELETE.
Template Description
This aggregation represents the definition of an initial value for the PersistencyDataElement referenced by the enclosing PersistencyDataRequiredComSpec
M2 Parameter
PersistencyDataRequiredComSpec.initValue

Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00021]

Functional Cluster	ShortName Path of Parameter Definition
Persistency	Persistency/PersistencyKeyValueStorageCollection/PersistencyKeyValueStorage/PersistencyKeyValuePairCollection/PersistencyKeyValuePair

APMC Parameter	Kind of APMC Parameter
ValueDataTypeRef	APMC-FOREIGN-REFERENCE-DEF

APMC Description
This reference represents the data type applicable for the value of the key-value pair.
Template Description
This represents the corresponding data type.
M2 Parameter
AutosarDataPrototype.type

Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	**APMC Parameter ID**
valid	[APMC_PER_00022]