|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **文件名** | | | | 舱单规格 | |
| **文档所有者** | | | | AUTOSAR | |
| **文件责任** | | | | AUTOSAR | |
| **文件识别号** | | | | 713 | |
| **文件状态** | | | | 发表 | |
| **AUTOSAR 标准的一部分** | | | | 自适应平台 | |
| **标准版本的一部分** | | | | R21-11 | |
|  | | | | | |
| **文档更改历史** | | | | | |
|  | | | | | |
| **日期** | **发布** | **更改者** | | **描述** | |
| 2021-11-25 | R21-11 | AUTOSAR  发布  管理 | | * 移除 REST 支持 * 信号到服务转换的返工 * 原始数据流配置的返工 * 诊断映射的返工 * 较小的更正/澄清/编辑更改 | |
| 2020-11-30 | R20-11 | AUTOSAR  发布  管理 | | * Phm贡献的重塑 * 安全事件报告 * 支持加密操作 * 诊断映射的重塑 * 较小的更正/澄清/编辑更改 | |
| 2019-11-28 | R19-11 | AUTOSAR  发布  管理 | | * 信号到服务转换的大修 * 支持原始数据流 * 支持车辆包 * 支持服务版本控制 * 将文档状态从最终更改为已发布 | |

|  |  |  |  |
| --- | --- | --- | --- |
| 2019-03-29 | 19-03 | AUTOSAR  发布  管理 | * 诊断端口接口介绍 * 软件集群的大修和软件包的引入 * 支持身份和访问管理 * 网络管理配置 |
| 2018-10-31 | 18-10 | AUTOSAR  发布  管理 | * 完成介绍   CppImplementationDataType   * 支持结构中的可选元素 * 自适应平台模块的返工配置 |
| 2018-03-29 | 18-03 | AUTOSAR  发布  管理 | * 时间同步 * DDS 部署 |
| 2017-10-27 | 17-10 | AUTOSAR  发布  管理 | * 服务接口中的可选元素 * 与 Web 服务的交互 * 安全通信 * 支持与加密和持久性的交互 * 信号到服务的翻译 * 支持端到端通信 * 平台健康管理 * 可上传的软件包 |
| 2017-03-31 | 17-03 | AUTOSAR  发布  管理 | •初始版本 |

**免责声明**

AUTOSAR 发布的本作品（规范和/或软件实施）及其中包含的材料仅供参考。 AUTOSAR 和为其做出贡献的公司对作品的任何使用概不负责。

本作品中包含的材料受版权和其他类型的知识产权保护。对本作品中包含的材料进行商业利用需要获得此类知识产权的许可。

可以以任何形式或通过任何方式未经任何修改地使用或复制本作品，仅供参考。未经出版商书面许可，不得以任何形式或任何方式出于任何其他目的使用或复制作品的任何部分。

这项工作仅针对汽车应用而开发。它既没有针对非汽车应用进行开发，也没有经过测试。

AUTOSAR 一词和 AUTOSAR 标志是注册商标。

# 目录

1. 简介 18
   1. 建模方法。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
   2. 定期服务。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
   3. 术语和缩写。 . . . . . . . . . . . . . . . . . . . . . . . . . 20
   4. 文档约定。 . . . . . . . . . . . . . . . . . . . . . . . . . . 22
   5. 需求跟踪 。 . . . . . . . . . . . . . . . . . . . . . . . . . . 24
2. 清单定义大图 29
   1. 设计与部署。 . . . . . . . . . . . . . . . . . . . . . . . . . . 29
      1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
      2. 设计和部署模型之间的关系。 . . . . . 29
      3. 文件的结构。 . . . . . . . . . . . . . . . . . . . 31
   2. 关于清单 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
   3. 序列化格式 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
   4. 范围 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
   5. 本文档中描述的清单 。 . . . . . . . . . . . . . . . . . 33
3. 应用程序设计 35
   1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
   2. 软件组件。 . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
   3. 数据类型 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
      1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
      2. 应用程序数据类型。 . . . . . . . . . . . . . . . . . . . . . . 37
         1. 字符串数据类型 。 . . . . . . . . . . . . . . . . . . . 38
         2. 关联地图数据类型 。 . . . . . . . . . . . . . 40
         3. SwDataDefProps 的属性。 . . . . . . . . . . . . 46
      3. CppImplementationDataType 。 . . . . . . . . . . . . . . . . . 47
         1. 概述 。 . . . . . . . . . . . . . . . . . . . . . . . 47
         2. SwDataDefProps 的属性。 . . . . . . . . . . . . 61
         3. 原始数据类型。 . . . . . . . . . . . . . . . . . 63
         4. 字符串数据类型 。 . . . . . . . . . . . . . . . . . . . 63
         5. 数组数据类型。 . . . . . . . . . . . . . . . . . . . . 64
         6. 矢量数据类型。 . . . . . . . . . . . . . . . . . . . 66
         7. 结构数据类型 。 . . . . . . . . . . . . . . . . . . . 69
         8. 枚举数据类型 。 . . . . . . . . . . . . . . . 70
         9. 地图数据类型 。 . . . . . . . . . . . . . . . . . . . . 72
         10. 变体数据类型。 . . . . . . . . . . . . . . . . . . . 73
         11. 位域数据类型。 . . . . . . . . . . . . . . . . . . . 74
      4. ApplicationDataType 和 CppImplementa 的兼容性

化数据类型。 . . . . . . . . . . . . . . . . . . . . . . . . . . 75

* 1. 服务接口 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
     1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
     2. 事件 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
     3. 触发 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 81
     4. 场。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 81
     5. 方法。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 83
        1. 火灾和忘记方法 。 . . . . . . . . . . . . . . . 84
     6. ServiceInterfaces 的版本控制。 . . . . . . . . . . . . . 85
        1. 由传输层驱动的版本控制。 . . . . . . . . . 86
     7. 命名空间 。 . . . . . . . . . . . . . . . . . . . . . . . . . . 87
     8. 错误处理。 . . . . . . . . . . . . . . . . . . . . . . . . . 90
     9. 服务接口数据类型映射 。 . . . . . . . . . . . . 93
     10. 通信组模式 。 . . . . . . . . . . . . . . . . 95
  2. 服务接口映射。 . . . . . . . . . . . . . . . . . . . . . . . . 96
  3. 服务接口元素映射。 . . . . . . . . . . . . . . . . . . . 102
     1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 102
     2. 服务接口事件映射。 . . . . . . . . . . . . . . . 106
     3. 服务接口触发器映射 。 . . . . . . . . . . . . . . 107 3.6.4 服务接口字段映射 。 . . . . . . . . . . . . . . . 108 3.6.5 服务接口方法映射。 . . . . . . . . . . . . . . 109
  4. 持久性接口。 . . . . . . . . . . . . . . . . . . . . . . . . . . . 111
     1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 111
        1. 大图 。 . . . . . . . . . . . . . . . . . . . . 111
        2. 持久性接口的建模。 . . . . . . . . . . 112
        3. 冗余处理 。 . . . . . . . . . . . . . . . . 113
        4. 更新处理 。 . . . . . . . . . . . . . . . . . . . 118
     2. 持久性键值存储接口 。 . . . . . . . . . . 119
     3. 持久性文件存储接口 。 . . . . . . . . . . . . . . 121
  5. 时间同步接口 。 . . . . . . . . . . . . . . . . . . . . . 123
  6. 平台健康管理界面。 . . . . . . . . . . . . . . . . . 126
     1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 126 3.9.2 受 监管实体和检查站 。 . . . . . . . . . . . . 127
     2. 健康频道。 . . . . . . . . . . . . . . . . . . . . . . . . 129
     3. 状态管理的恢复通知。 . . . . . . . . . 131
  7. 诊断接口 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . 134
     1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 134 3.10.2 诊断程序接口 。 . . . . . . . . . . . . . . . . . 136

3.10.3 数据标识符接口和数据标识符元素。 . 139 3.10.4 诊断事件接口。 . . . . . . . . . . . . . . . . 142 3.10.5 诊断条件接口。 . . . . . . . . . . . . . . . 143 3.10.6 指示灯界面。 . . . . . . . . . . . . . . . . . . . . . . . 144

* + 1. 安全级别接口。 . . . . . . . . . . . . . . . . . . . . 145
    2. 服务验证接口。 . . . . . . . . . . . . . . . . . . 145 3.10.9 操作循环界面。 . . . . . . . . . . . . . . . . . . . 146
    3. 通用 UDS 接口 。 . . . . . . . . . . . . . . . . . . . . 147
    4. DoIP 接口 。 . . . . . . . . . . . . . . . . . . . . . . . . 148 3.10.12 上传和下载诊断接口 。 . . . . . . 150

3.10.13 支持管理 EcuReset 的接口 。 . . . . . . . . 151 3.10.14 诊断认证接口 。 . . . . . . . . . . . . . 152

* + 1. 诊断外部认证接口 。 . . . . . . . . 152
    2. 诊断通信控制接口 。 . . . . . . . . 153
  1. 加密接口。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 154
     1. 与加密软件的交互 。 . . . . . . . . . . . . . . . 154
     2. 加密密钥槽接口 。 . . . . . . . . . . . . . . . . . . . 155
     3. 加密证书接口。 . . . . . . . . . . . . . . . . . . 160
     4. 加密提供者接口。 . . . . . . . . . . . . . . . . . . . 160
     5. 加密 TrustMaster 接口 。 . . . . . . . . . . . . . . . . . 161
     6. 将加密证书链接到加密密钥槽。 . . . . . . 161
  2. 原始数据流接口 。 . . . . . . . . . . . . . . . . . . . . . . . 164
  3. 安全事件报告界面。 . . . . . . . . . . . . . . . . . . . . . 165
  4. 日志和跟踪接口 。 . . . . . . . . . . . . . . . . . . . . . . . . . 167
  5. 应用程序的交互端点。 . . . . . . . . . . . . . . . . . . . 168
     1. 面向服务的通信。 . . . . . . . . . . . . . . . 168
     2. 与持久键值存储交互。 . . . . . . . . 169 3.15.3 与持久文件存储的交互 。 . . . . . . . . . . . 170
     3. 端口原型道具 。 . . . . . . . . . . . . . . . . . . . . . 170
     4. 端口原型 ComSpec 。 . . . . . . . . . . . . . . . . . . . 172
        1. 服务接口类型的端口原型。 . . . . 174
        2. 由持久性数据接口键入的端口原型 183
  6. 可执行文件。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 184
  7. 复杂数据结构中的可选成员。 . . . . . . . . . . . . . 190
     1. 背景 。 . . . . . . . . . . . . . . . . . . . . . . . . . . 190
     2. 可选性的定义 。 . . . . . . . . . . . . . . . . . . . . 191
  8. 序列化属性。 . . . . . . . . . . . . . . . . . . . . . . . . . . 193
     1. 序列化属性的默认值。 . . . . . . . . . . 194
     2. 序列化属性的单独定义 。 . . . . . . . 199
     3. TLV 属性的分配 。 . . . . . . . . . . . . . . . . 205
        1. TLV 数据 ID 的分配。 . . . . . . . . . . . . . 205
        2. 线型选择的分配。 . . . . . . . . . 211
  9. 工艺设计。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 211
     1. 确定性客户端资源 。 . . . . . . . . . . . . . . . . 213
  10. 格兰特设计 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 216
      1. Com 格兰特设计。 . . . . . . . . . . . . . . . . . . . . . . . 217
      2. 原始流数据的授权设计。 . . . . . . . . . . . . 223
      3. 远程访问控制 。 . . . . . . . . . . . . . . . . . . . . 224
         1. TLS 情况下的远程主题 。 . . . . . . . . . . . 226 3.20.3.2 IPsec 情况下的远程主题 。 . . . . . . . . . . 228
         2. IP 通信时的远程主体 。 . . . 230
         3. SecOC 通信时的远程主体 。 231

1. 诊断设计 232
   1. 诊断映射。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 232
      1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 232
      2. 诊断监视器到端口映射 。 . . . . . . . . . . . . . 235
      3. 端口映射的诊断事件 。 . . . . . . . . . . . . . . 238
      4. 诊断操作周期到端口映射 。 . . . . . . . . 240
      5. 端口映射的诊断启用条件。 . . . . . . . . 242 4.1.6 诊断清除条件到端口映射。 . . . . . . . . . 244
      6. 端口映射的诊断指示器。 . . . . . . . . . . . . . 246
      7. 端口映射的诊断内存目的地 。 . . . . . . 248
      8. 端口映射的诊断安全性。 . . . . . . . . . . . . . 250
      9. 诊断数据标识符到端口映射。 . . . . . . . . . . 252 4.1.11 诊断通用 UDS 服务处理程序到端口映射。 256
      10. 诊断通用映射 。 . . . . . . . . . . . . . . . . . 256
      11. 诊断服务验证映射 。 . . . . . . . . . . . 260 4.1.14 诊断认证映射。 . . . . . . . . . . . . . . 261

4.1.15 诊断外部认证映射 。 . . . . . . . . 264

* 1. 诊断清除条件。 . . . . . . . . . . . . . . . . . . . . . . . . 266
  2. 安全访问。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 267
  3. 诊断提供的数据映射 。 . . . . . . . . . . . . . . . . . . . . 268
  4. 诊断杂项。 . . . . . . . . . . . . . . . . . . . . . . . . . . 269

1. 系统设计 270
   1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 270
   2. 通信系统结构规范。 . . . . . . . . . . . 272
      1. 网络连接。 . . . . . . . . . . . . . . . . . . . . . . 275
         1. 支持 10BASE-T1S 网络拓扑 。 . . . 280
      2. TcpIp 堆栈配置属性。 . . . . . . . . . . . . . 280
         1. IP 配置属性。 . . . . . . . . . . . . . . 281 5.2.2.2 TCP 和 UDP 配置属性。 . . . . . . . 287

5.2.2.3 ICMP 配置属性。 . . . . . . . . . . . . 289

* + 1. 保护与 IPsec 的通信。 . . . . . . . . . . . . . 290
    2. 服务发现配置。 . . . . . . . . . . . . . . . 298
       1. SOME/IP 服务发现配置。 . . . . . 298
    3. 部分网络。 . . . . . . . . . . . . . . . . . . . . . . . . . 300
  1. 应用软件系统结构规范。 . . . . . . . . 302
  2. Classic 和 Classic 之间面向服务的通信建模

自适应平台。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 304 5.4.1 方法映射 。 . . . . . . . . . . . . . . . . . . . . . . . . 306 5.4.2 事件映射 。 . . . . . . . . . . . . . . . . . . . . . . . . . 306

* + 1. 场映射。 . . . . . . . . . . . . . . . . . . . . . . . . . . 307
    2. FireAndForgetMapping 。 . . . . . . . . . . . . . . . . . . . . 309

1. 子系统设计 311
   1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 311 6.2 软件集群设计。 . . . . . . . . . . . . . . . . . . . . . . . . . 311 6.3 提供和要求的软件集群设计服务 。 . . . . . 317

6.4 服务到可执行文件的映射 。 . . . . . . . . . . . . . . . . . . 320

1. 机器清单 324
   1. 过程到机器映射。 . . . . . . . . . . . . . . . . . . . . . . 328
      1. 一般建模方法 。 . . . . . . . . . . . . . . . . . 328
      2. 核心亲和力 。 . . . . . . . . . . . . . . . . . . . . . . . . . . 330
      3. 默认启动和终止超时。 . . . . . . . . . . 331
2. 执行清单 333
   1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 333
   2. 启动配置。 . . . . . . . . . . . . . . . . . . . . . . . . . . . 336
      1. 状态相关的启动配置 。 . . . . . . . . . . . 337
      2. 调度。 . . . . . . . . . . . . . . . . . . . . . . . . . . . 340
      3. 过程参数 。 . . . . . . . . . . . . . . . . . . . . . . 341
      4. 与资源组的关联 。 . . . . . . . . . . . . . . 341
      5. 执行依赖 。 . . . . . . . . . . . . . . . . . . . . 342
      6. 将进程分配给功能组状态 。 . . . . 344
      7. 资源消耗边界。 . . . . . . . . . . . . . 345
      8. 错误和终止行为。 . . . . . . . . . . . . . . . . 347
   3. 确定性客户端。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 348
   4. 功能组 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 349
      1. 功能组的语义。 . . . . . . . . . . . . . . . . . 349
      2. 机器功能组 。 . . . . . . . . . . . . . . . . . . . 352
   5. 安全事件的报告。 . . . . . . . . . . . . . . . . . . . . . . . 353
3. 平台模块开发 354
   1. 操作系统模块配置。 . . . . . . . . . . . . . . . . . . . . . . . . . 356
   2. 持久性部署 。 . . . . . . . . . . . . . . . . . . . . . . . . . 358
      1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 358
         1. 冗余处理 。 . . . . . . . . . . . . . . . . 361
         2. 更新处理 。 . . . . . . . . . . . . . . . . . . . 362
         3. 尺寸处理。 . . . . . . . . . . . . . . . . . . . . . 363
         4. 安全处理。 . . . . . . . . . . . . . . . . . . . 364
      2. 持久键值存储的部署 。 . . . . . . . . 364
      3. 文件存储部署。 . . . . . . . . . . . . . . . . . . 368
   3. 平台健康管理部署。 . . . . . . . . . . . . . . . 371
      1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 371
      2. 设计和部署之间的关系。 . . . . . . . . . . 374
      3. 监督部署。 . . . . . . . . . . . . . . . . . . . . 375
         1. 全球监管 。 . . . . . . . . . . . . . . . . . . 378
         2. 监督模式 。 . . . . . . . . . . . . . . . . . . 381
         3. 监督模式条件。 . . . . . . . . . . . . . 383
         4. 无监督定义 。 . . . . . . . . . . . . . 386
         5. AliveSupervision 定义。 . . . . . . . . . . . 387
         6. CheckpointTransition 定义。 . . . . . . . . 389
         7. 逻辑监督定义 。 . . . . . . . . . 390
         8. DeadlineSupervision 定义。 . . . . . . . . . 391
      4. 健康通道部署。 . . . . . . . . . . . . . . . . . . 392
         1. 监督健康通道部署 。 . . . . . . 393
         2. 外部健康通道部署 。 . . . . . . . . 393
      5. 恢复通知 。 . . . . . . . . . . . . . . . . . . . . . 396
   4. 时间同步部署 。 . . . . . . . . . . . . . . . . . . . 397
      1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 397
      2. 时间同步功能集群配置 。 . . . 398
      3. 时基 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . 399
         1. 同步时基。 . . . . . . . . . . . . . . . 400
         2. 持久时基值存储。 . . . . . . . . . 403
         3. 以太网同步时间 。 . . . . . . . . . . . . . 404
      4. 时基到端口原型映射。 . . . . . . . . . . . . 406
   5. DoIP 配置 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 408
   6. 日志和跟踪模块配置 。 . . . . . . . . . . . . . . . . . . 414
      1. 记录和跟踪部署。 . . . . . . . . . . . . . . . . . . . 416
      2. 设计和部署之间的关系。 . . . . . . . . . . 419
   7. 网络管理配置。 . . . . . . . . . . . . . . . . . . . 422
      1. UdpNm 配置约束。 . . . . . . . . . . . . . . . 429
   8. 更新和配置管理 。 . . . . . . . . . . . . . . . . 430
   9. IAM 配置。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 432
      1. Com Grant 部署。 . . . . . . . . . . . . . . . . . . . . 433
      2. 为原始流数据授予部署。 . . . . . . . . . 438
      3. 远程访问控制 。 . . . . . . . . . . . . . . . . . . . . 439
   10. 加密部署。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 441
       1. 加密提供者 。 . . . . . . . . . . . . . . . . . . . . . . . . 442
       2. 加密密钥槽 。 . . . . . . . . . . . . . . . . . . . . . . . . 444
       3. 加密证书 。 . . . . . . . . . . . . . . . . . . . . . . . 446
   11. IdsM 部署。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 448
       1. IdsM 实例化 。 . . . . . . . . . . . . . . . . . . . . . . . 448
       2. 获取安全事件的自定义时间戳 。 . . . . 449
       3. 安全事件部署 。 . . . . . . . . . . . . . . . 450
4. 服务实例清单 452
   1. 应用程序端点 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . 452
      1. VLAN 优先级。 . . . . . . . . . . . . . . . . . . . . . . . . . . 454
   2. 服务接口部署。 . . . . . . . . . . . . . . . . . . . . . . 455
      1. SOME/IP 服务接口部署。 . . . . . . . . . . . 460
      2. DDS 服务接口部署 。 . . . . . . . . . . . . . 469
      3. 用户定义的服务接口。 . . . . . . . . . . . . . . . . 474
   3. 服务实例部署 。 . . . . . . . . . . . . . . . . . . . . . . 478
      1. SOME/IP 服务实例部署。 . . . . . . . . . . . 484
         1. 提供的服务实例。 . . . . . . . . . . . . . . 485
         2. 必需的服务实例。 . . . . . . . . . . . . . . 506
      2. DDS 服务实例部署。 . . . . . . . . . . . . . . 517
         1. 提供 DDS 服务实例。 . . . . . . . . . . . 518
         2. 必需的 DDS 服务实例。 . . . . . . . . . . . 522
         3. DDS 服务实例到机器映射 。 . . . . 524
      3. 用户定义的服务实例部署。 . . . . . . . . . 525
      4. 服务部署关系 。 . . . . . . . . . . . . . . . . 527
   4. 端到端保护 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . 528
   5. 安全通信 。 . . . . . . . . . . . . . . . . . . . . . . . . . 536
      1. 通过 TLS 进行安全通信 。 . . . . . . . . . . . . . . 540
      2. 通过 SecOC 进行安全通信。 . . . . . . . . . . . . . 548
      3. 通过 DDS 进行安全通信 。 . . . . . . . . . . . . . . 551
5. 原始数据流清单 555
   1. 原始数据流部署。 . . . . . . . . . . . . . . . . . . . . . . 555
   2. 原始数据流映射 。 . . . . . . . . . . . . . . . . . . . . . . . 555
   3. 以太网上的原始数据流。 . . . . . . . . . . . . . . . . . . . . 556
      1. 以太网原始数据流客户端映射。 . . . . . . . . . 561 11.3.2 以太网原始数据流服务器映射。 . . . . . . . . . 562

11.3.3 以太网原始数据流配置示例。 . . . . 564

* + - 1. 序言 。 . . . . . . . . . . . . . . . . . . . . . . . 564
      2. 完全形式化的模型示例 。 . . . . . . . . . . 566 11.3.3.3 只有客户端是正式的。 . . . . . . . . . . . . . . . 568

11.3.3.4 只有 Server 正式化 。 . . . . . . . . . . . . . . 568

1. 基于信号的通信 570
   1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 570
   2. 基于信号的先决条件。 . . . . . . . . . . . . . . . . . . . . . . . . 574
      1. 消息头。 . . . . . . . . . . . . . . . . . . . . . . . . 575
   3. 基于信号的部署 。 . . . . . . . . . . . . . . . . . . . . . . . . 577
   4. 信号/服务转换映射。 . . . . . . . . . . . . . . . . . . . 578
      1. 信号/服务转换映射的方向 。 . . . . . . 578
      2. SignalBasedEvent 映射。 . . . . . . . . . . . . . . . . . . 580
      3. SignalBasedTrigger 映射。 . . . . . . . . . . . . . . . . . 585
      4. SignalBasedField 映射。 . . . . . . . . . . . . . . . . . . 586
      5. SignalBasedMethod 映射。 . . . . . . . . . . . . . . . . . 590
   5. 服务发现和静态通信配置 。 . . . . . 592
      1. 服务发现控制。 . . . . . . . . . . . . . . . . . . . 593 12.5.2 翻译开始后的服务控制。 . . . . . . . . . . 595

12.5.3 由于相关服务实例的可用性导致的服务控制 595

* + - 1. 信号服务翻译 。 . . . . . . . . . . . . . . 596 12.5.3.2 服务信号转换 。 . . . . . . . . . . . . . . 596

12.5.4 静态通讯配置。 . . . . . . . . . . . . . 596

* + - 1. 服务器的静态通信配置 。 596
      2. 客户端的静态通信配置。 . 597
  1. 翻译行为 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . 597
     1. 翻译数据的 ServiceInterface 表示。 . . 598
        1. 可选 元素在 signal-servicetranslation中。 . . . . . . . . . . . . . . . . . . . . 598
        2. 可选 元素以防 service-signaltranslation 。 . . . . . . . . . . . . . . . . . . . . 599
     2. 事件和更新位语义 。 . . . . . . . . . . . . . . . 600
     3. 数据过滤 。 . . . . . . . . . . . . . . . . . . . . . . . . . . 603 12.6.4 从一个来源翻译 。 . . . . . . . . . . . . . . . . . 605

12.6.5 多个来源的翻译。 . . . . . . . . . . . . . . . 606

* 1. 翻译直通作文。 . . . . . . . . . . . . . . . . . . 607
  2. Classic 平台的预期功能。 . . . . . . . . . . . . . . . . . . 608
     1. 处理订单。 . . . . . . . . . . . . . . . . . . . . . . . . 608
     2. 接收数据过滤器。 . . . . . . . . . . . . . . . . . . . . . . 608
     3. 接收到无效信号 。 . . . . . . . . . . . . . . . . . . 608
     4. 更新位处理。 . . . . . . . . . . . . . . . . . . . . . . 609 12.8.5 Service-的传输属性和传输模式

信号翻译 。 . . . . . . . . . . . . . . . . . . . . . . . 609

* + 1. 期限监控。 . . . . . . . . . . . . . . . . . . . . . . 611
    2. 信号和IPdu传输 。 . . . . . . . . . . . . . . . . 611
    3. IPdu 多路复用。 . . . . . . . . . . . . . . . . . . . . . . . . 611
  1. 端到端的考虑。 . . . . . . . . . . . . . . . . . . . . . . . . 613
     1. 安全。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 613
        1. 信号服务翻译 。 . . . . . . . . . . . . . . 616
        2. 服务信号翻译 。 . . . . . . . . . . . . . . 616
     2. 安全。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 617
        1. 经典平台 SecOC 的限制。 . . . . . . 619
        2. 经典平台 SecOC 和 Cryp- 之间的链接

toKeySlot 。 . . . . . . . . . . . . . . . . . . . . . . 619

1. 跨功能集群交互 621
   1. ComCertificateToCryptoCertificateMapping 。 . . . . . . . . . . . . . . 623
   2. ComKeyToCryptoKeySlotMapping 。 . . . . . . . . . . . . . . . . . . . . 624 13.3 ComSecOcToCryptoKeySlotMapping 。 . . . . . . . . . . . . . . . . . . 626
   3. PersistencyDeploymentToCryptoKeySlotMapping 。 . . . . . . . . . . . 628
   4. PersistencyDeploymentElementToCryptoKeySlotMapping 。 . . . . . . 629
   5. PersistencyDeploymentToDltLogSinkMapping 。 . . . . . . . . . . . . . 631
2. 软件分发 632
   1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 632
   2. 软件集群 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 633
      1. 软件集群通用建模 。 . . . . . . . . . . . . . 633
      2. 诊断软件集群的相关性 。 . . . . . . . 640
         1. 诊断道具。 . . . . . . . . . . . . . . . . . . . 640
         2. 诊断地址。 . . . . . . . . . . . . . . . . . . 642 14.2.2.3 与诊断贡献集的关系 。 . . . . 644

14.2.2.4 诊断服务验证 。 . . . . . . . . . . . . 645

* + 1. 子软件集群 。 . . . . . . . . . . . . . . . . . . . . . 646
    2. 软件集群依赖 。 . . . . . . . . . . . . . . . . 647 14.2.5 软件集群之间的引用 。 . . . . . . . . . . . 653 14.2.6 软件集群工件校验和。 . . . . . . . . . . . . . 654
  1. 软件包。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 656
  2. 车辆包 。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 661
     1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 661 14.4.2 车辆推出步骤。 . . . . . . . . . . . . . . . . . . . . . . . 664
     2. Ucm 步长。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 666
     3. 软件包步骤。 . . . . . . . . . . . . . . . . . . . . . 667
     4. SoftwarePackageStep 的使用示例 。 . . . . . 669
        1. 转移和过程的用法示例 。 . 669

1. 经典平台和自适应平台之间的互操作性 673
   1. 在 SOME/IP 网络绑定中使用 majorVersion 。 . . . . . . . 673
2. 示例 675
   1. 通过服务接口映射部署服务实例 。 . . . . . 675
   2. 通过服务接口元素映射部署服务实例 。 677
   3. 启动配置的定义。 . . . . . . . . . . . . . . . . . . . . 679
   4. 服务实例映射。 . . . . . . . . . . . . . . . . . . . . . . . . 682
   5. 雷达和摄像头服务接口示例 。 . . . . . . . . . . . . . 685
   6. 持久数据的定义。 . . . . . . . . . . . . . . . . . . . . . . . 690
   7. 持久文件的定义。 . . . . . . . . . . . . . . . . . . . . . . . . 691
   8. Phm 交互的定义。 . . . . . . . . . . . . . . . . . . . . . . . 692
      1. Phm 应用程序设计示例 。 . . . . . . . . . . . . . . 692
      2. Phm 配置示例。 . . . . . . . . . . . . . . . . . . 693
   9. 定义 Vector 的场景。 . . . . . . . . . . . . . . . . . . . . . . . 695
3. 自定义模型扩展 696
   1. 概述。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 696
   2. 自定义属性定义。 . . . . . . . . . . . . . . . . . . . . . . . . 698
      1. 自定义原始属性定义。 . . . . . . . . . . . . . 698
      2. 自定义复杂属性定义。 . . . . . . . . . . . . . 700
   3. 自定义外部参考定义。 . . . . . . . . . . . . . . . . . . 702
   4. 自定义子类配置。 . . . . . . . . . . . . . . . . . . . . . 705
   5. 自定义约束。 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 707
   6. 从 SdgClass 到 SdgClass 的引用定义。 . . . . . . . . . . 708
4. 通用建模 711
   1. 对 PortInterface 中的 DataPrototype 的引用。 . . . . . . . . . . . . 711
      1. 对 ApplicationDataType 内部的引用。 . . 711
      2. 对 CppImplementationDataType 714内部的引用
   2. 对 Executable 中的 AutosarDataPrototype 的引用。 . . . . . . . . 716
   3. 对 Executable 中的 PortPrototype 的引用 。 . . . . . . . . . . . . 719
   4. 可执行文件中的方法建模。 . . . . . . . . . . . . . . . . . 725
   5. 模式相关 InstanceRefs 的建模 。 . . . . . . . . . . . . . . . . 728
   6. 诊断相关 InstanceRefs 的建模。 . . . . . . . . . . . . . . 730
   7. PHM 相关 InstanceRefs 的建模。 . . . . . . . . . . . . . . . . . 731
   8. 与时间相关的 InstanceRefs 的建模。 . . . . . . . . . . . . . . . . . 735
   9. 持久性相关 InstanceRefs 的建模 。 . . . . . . . . . . . . . 736
   10. SoftwareClusterDesign 相关的 InstanceRefs 的建模。 . . . . . . . 736
5. 提到的类表 740
6. 约束和规范项目的历史 806
   1. 本文档的约束和规范项目历史根据

至 AUTOSAR Release R17-03（原始版本） 。 . . . . . . . . . . . . 806

* + 1. 在 R17-03 中创建了约束。 . . . . . . . . . . . . . . . . 806
    2. 在 R17-03 中创建了规范项目。 . . . . . . . . . . . . 808
  1. 本文档的约束和规范项目历史根据

到 AUTOSAR 版本 R17-10 。 . . . . . . . . . . . . . . . . . . . . . . 813

* + 1. 在 R17-10 中添加了 Traceables 。 . . . . . . . . . . . . . . . . . 813
    2. 在 R17-10 中更改了 Traceables 。 . . . . . . . . . . . . . . . . 817
    3. 删除了 R17-10 中的 Traceables 。 . . . . . . . . . . . . . . . . 818
    4. 在 R17-10 中添加了约束。 . . . . . . . . . . . . . . . . . 818
    5. 更改了 R17-10 中的约束 。 . . . . . . . . . . . . . . . 820
    6. 删除了 R17-10 中的约束。 . . . . . . . . . . . . . . . . 820
  1. 本文档的约束和规范项目历史根据

到 AUTOSAR 版本 R18-03 。 . . . . . . . . . . . . . . . . . . . . . . 820

* + 1. 在 R18-03 中添加了 Traceables 。 . . . . . . . . . . . . . . . . . 820
    2. 在 R18-03 中更改了 Traceables 。 . . . . . . . . . . . . . . . . 824
    3. 删除了 R18-03 中的 Traceables 。 . . . . . . . . . . . . . . . . 826
    4. 在 R18-03 中添加了约束。 . . . . . . . . . . . . . . . . . 826
    5. 更改了 R18-03 中的约束 。 . . . . . . . . . . . . . . . 829
    6. 删除了 R18-03 中的约束。 . . . . . . . . . . . . . . . . 829
  1. 本文档的约束和规范项目历史根据

到 AUTOSAR 版本 R18-10 。 . . . . . . . . . . . . . . . . . . . . . . 830

* + 1. 在 R18-10 中添加了 Traceables 。 . . . . . . . . . . . . . . . . . 830
    2. 在 R18-10 中更改了 Traceables 。 . . . . . . . . . . . . . . . . 832
    3. 删除了 R18-10 中的 Traceables 。 . . . . . . . . . . . . . . . . 834
    4. 在 R18-10 中添加了约束。 . . . . . . . . . . . . . . . . . 836
    5. 更改了 R18-10 中的约束 。 . . . . . . . . . . . . . . . 838
    6. 删除了 R18-10 中的约束。 . . . . . . . . . . . . . . . . 839
  1. 本文档的约束和规范项目历史根据

到 AUTOSAR 版本 R19-03 。 . . . . . . . . . . . . . . . . . . . . . . 840

* + 1. 在 R19-03 中添加了 Traceables 。 . . . . . . . . . . . . . . . . . 840
    2. 在 R19-03 中更改了 Traceables 。 . . . . . . . . . . . . . . . . 842
    3. 删除了 R19-03 中的 Traceables 。 . . . . . . . . . . . . . . . . 843
    4. 在 R19-03 中添加了约束。 . . . . . . . . . . . . . . . . . 843
    5. 更改了 R19-03 中的约束 。 . . . . . . . . . . . . . . . 845
    6. 删除了 R19-03 中的约束。 . . . . . . . . . . . . . . . . 845
  1. 本文档的约束和规范项目历史根据

到 AUTOSAR 版本 R19-11 。 . . . . . . . . . . . . . . . . . . . . . . 846

* + 1. 在 R19-11 中添加了 Traceables 。 . . . . . . . . . . . . . . . . . 846
    2. 在 R19-11 中更改了 Traceables 。 . . . . . . . . . . . . . . . . 849
    3. 删除了 R19-11 中的 Traceables 。 . . . . . . . . . . . . . . . . 850
    4. 在 R19-11 中添加了约束。 . . . . . . . . . . . . . . . . . 850
    5. 更改了 R19-11 中的约束 。 . . . . . . . . . . . . . . . 852
    6. 删除了 R19-11 中的约束。 . . . . . . . . . . . . . . . . 853
  1. 本文档的约束和规范项目历史根据

到 AUTOSAR 版本 R20-11 。 . . . . . . . . . . . . . . . . . . . . . . 853

* + 1. 在 R20-11 中添加了 Traceables 。 . . . . . . . . . . . . . . . . . 853
    2. 在 R20-11 中更改了 Traceables 。 . . . . . . . . . . . . . . . . 856
    3. 在 R20-11 中删除了 Traceables 。 . . . . . . . . . . . . . . . . 859
    4. 在 R20-11 中添加了约束。 . . . . . . . . . . . . . . . . . 860
    5. 更改了 R20-11 中的约束 。 . . . . . . . . . . . . . . . 862
    6. 删除了 R20-11 中的约束。 . . . . . . . . . . . . . . . . 863
  1. 本文档的约束和规范项目历史根据

到 AUTOSAR 版本 R21-11 。 . . . . . . . . . . . . . . . . . . . . . . 864

* + 1. 在 R21-11 中添加了 Traceables 。 . . . . . . . . . . . . . . . . . 864
    2. 在 R21-11 中更改了 Traceables 。 . . . . . . . . . . . . . . . . 867
    3. 删除了 R21-11 中的 Traceables 。 . . . . . . . . . . . . . . . . 869
    4. 在 R21-11 中添加了约束。 . . . . . . . . . . . . . . . . . 870
    5. 更改了 R21-11 中的约束 。 . . . . . . . . . . . . . . . 874
    6. 删除了 R21-11 中的约束 。 . . . . . . . . . . . . . . . . 875

1. 本文档范围内的可拆分元素 876
2. 本文件范围内的变化点 878

# 参考

1. 软件组件模板

AUTOSAR\_TPS\_SoftwareComponentTemplate

1. 分层软件架构

AUTOSAR\_EXP\_LayeredSoftwareArchitecture

1. 面向服务架构的参考模型 1.0 <https://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf>
2. 词汇表

AUTOSAR\_TR\_词汇表

1. 标准化模板

AUTOSAR\_TPS\_标准化模板

1. 通用结构模板

AUTOSAR\_TPS\_GenericStructureTemplate

1. SOME/IP 协议规范

AUTOSAR\_PRS\_SOMEIP协议

1. 通讯管理规范

AUTOSAR\_SWS\_CommunicationManagement

1. 持久性规范

AUTOSAR\_SWS\_Persistency

1. IEEE 信息技术标准-标准化应用环境配置文件 (AEP)-POSIX 实时和嵌入式应用支持<https://standards.ieee.org/findstds/standard/1003.13-2003.html>
2. 时间同步规范

AUTOSAR\_SWS\_TimeSynchronization

1. 平台健康管理规范

AUTOSAR\_SWS\_PlatformHealthManagement

1. 密码学规范

AUTOSAR\_SWS\_密码学

1. 身份和访问管理规范

AUTOSAR\_SWS\_IdentityAndAccessManagement

1. ara::com API 说明

AUTOSAR\_EXP\_ARAComAPI

1. 信息技术 – 通用编码字符集 (UCS) [http://www.iso.org](http://www.iso.org/)
2. 系统模板

AUTOSAR\_TPS\_SystemTemplate

1. 执行管理规范

AUTOSAR\_SWS\_ExecutionManagement

1. 诊断提取模板

AUTOSAR\_TPS\_DiagnosticExtractTemplate

1. 诊断规范

AUTOSAR\_SWS\_Diagnostics

1. 安全提取模板

AUTOSAR\_TPS\_SecurityExtractTemplate

1. 道路车辆 – 基于 Internet 协议 (DoIP) 的诊断通信[http://www.iso.org](http://www.iso.org/)
2. 日志和跟踪协议规范

AUTOSAR\_PRS\_LogAndTraceProtocol

1. SOME/IP 服务发现协议规范

AUTOSAR\_PRS\_SOMEIPServiceDiscoveryProtocol

1. 数据分发服务 (DDS)，版本 1.4 <http://www.omg.org/spec/DDS/1.4>
2. RPC over DDS，版本 1.0 <https://www.omg.org/spec/DDS-RPC/1.0>
3. DDS 合并 XML 语法，版本 1.0 <https://www.omg.org/spec/DDS-XML>
4. SW-C端到端通信保护库规范

AUTOSAR\_SWS\_E2EL库

1. E2E 协议规范

AUTOSAR\_PRS\_E2E协议

1. DDS 安全性，版本 1.1 <https://www.omg.org/spec/DDS-SECURITY/1.1>
2. 安全板载通信规范

AUTOSAR\_SWS\_SecureOnboardCommunication

# 1 简介

*AUTOSAR 自适应平台*上所谓的*Manifest的规范*。 1.1节对整体建模方法进行了描述。 1.2节给出了对术语*服务*定义的参考。

术语*Manifest*在本规范中用于配置内容的正式规范的含义。请在第2节中找到对该术语的更详细描述以及对*AUTOSAR 自适应平台的含义*。

请注意，文档的内容（不管名称）延伸到对为*AUTOSAR 自适应平台开发软件所必需的设计元素的描述*。

设计相关的建模主要集中在*AUTOSAR自适应平台上的应用软件开发*以及应用与诊断之间的联系，并进行了详细的描述[[1]](#footnote-1) 在第3节和第4节中。

第5节特别描述了*AUTOSAR 经典平台*和*AUTOSAR 自适应平台*通过面向服务通信进行通信的总体情况。

第7节描述了通过*清单配置机器的选项*。

第8节代表与第3节有关部署级别的对应部分，它描述了所谓的*执行清单的内容*。

第9节包含解释平台模块功能的清单内容的一系列小节。

第10节详细描述了如何在*清单*级别配置面向服务的通信。

第11节描述了原始数据流通信的部署建模。

第12节解释了如何将基于信号的通信转换为面向服务的通信，反之亦然，以参与*AUTOSAR 经典平台上 ECU 之间的通信*。

第13节解释了 AUTOSAR 堆栈中功能集群之间交互的建模。

第14节描述了可上传软件包的概念和配置。

最后，第15节对*AUTOSAR 自适应平台与 AUTOSAR 自适应平台上*的软件的交互进行了一些评论。

## 1.1 建模方法

*AUTOSAR 自适应平台*是在AUTOSAR*经典平台*已经是汽车领域稳定且完善的标准时推出的。

然而， *AUTOSAR 自适应平台并不是AUTOSAR 经典平台*的继承者。对于可以由一个或另一个平台更好地实现的特定用例，两个平台相互补充。

*AUTOSAR 自适应平台*上进行建模：

* *AUTOSAR自适应平台*与*AUTOSAR经典平台*基于不同的原理，因此建模方法也可以**尽可能地与AUTOSAR经典平台的规范脱钩，**以宣传两个平台具有不同目的的事实。

因此，即使特定模型元素在各自的其他平台中有明确的对应物，也要使用不同的术语以免混淆两个平台的用户。

* 尽管两个平台之间存在不可否认的差异，但仍有大量惊人的相似之处强烈鼓励使用来自*AUTOSAR 经典平台***的现有建模概念**，尤其是来自 AUTOSAR 软件组件模板 [ 1 ] 的规范，同样如此尽可能。

因此，结论是在两个平台上将相同的元类用于类似目的。然后有必要在适用的情况下扩展一些受影响的特定于平台的元类，并添加约束以阐明所提到的扩展的特定于平台的用法。

事不宜迟， *AUTOSAR 自适应平台的建模方法*遵循第二种选择。

这意味着，例如， *AUTOSAR 自适应平台上的一个应用软件*应由SwComponentType表示。这包括对CompositionSwComponentType的定义，该定义反过来聚合由例如（在*AUTOSAR 自适应平台的情况下*） AdaptiveApplicationSwComponentType键入的SwComponentPrototype 。

重用现有模型元素来定义*AUTOSAR 自适应平台的元模型*具有副作用，即现有模型元素的描述可能包含对仅在*AUTOSAR 经典平台上有意义的技术细节的引用*。

毕竟，模型元素是在只有*AUTOSAR 经典平台*存在时创建的。

这些参考资料应持保留态度。预计读者可以从这些细节中抽象出来，并提取这些模型元素的各个方面，这些元素为*AUTOSAR 自适应平台的描述创造了相关性*。

## 1.2 定期服务

重要的是要记住，*服务*一词在本文档中经常使用，特别是在*AUTOSAR 自适应平台*中。

尽管*AUTOSAR 自适应平台*上的术语*服务的含义与 AUTOSAR*中使用的其他含义相冲突，但这种用法有其原因。

综上所述，在AUTOSAR 的范围内，*服务一词的含义如下：*

* 术语*服务*用于分层软件架构 [ 2 ]，表示与应用程序交互的 AUTOSAR 软件架构的最高层。在此上下文中，已在*AUTOSAR 经典平台上创建了*ServiceSwComponentType 、 SwcServiceDependency 、 ServiceNeeds或PortInterface .isService 等模型元素。
* *服务*一词用于表示信息在**维修汽车的车间中是相关的或需要的**。在这种情况下，定义*了仅服务诊断故障代码*(DTC)。
* *服务*一词用于描述**诊断服务的处理**，例如 UDS 服务*ReadDataByIdentifier* ，用于诊断测试仪和 (AUTOSAR) ECU 上的诊断堆栈之间的通信。
* *服务*一词的含义由**面向服务的体系结构**(SOA) [ 3 ] 定义。这个含义与*AUTOSAR 自适应平台*上术语*服务*的使用有最密切的关系。

## 1.3 术语和缩写

术语和缩写的主要列表在 [ 4 ] 中定义。下表包含在本文档范围内使用的术语和缩写列表，这些术语和缩写尚未在 [ 4 ] 中定义，以及每个缩写的拼写含义。

|  |  |
| --- | --- |
| *缩写* | *意义* |
| AES | 高级加密标准 |
| 三磷酸腺苷 | AUTOSAR 模板配置文件 |
| ARXML | AUTOSAR XML |
|  | 计数器模式 |

5

4

|  |  |
| --- | --- |
| *缩写* | *意义* |
| DDS | 数据分发服务 |
| DES | 数据加密标准 |
| DM | 诊断管理器 |
| 故障诊断码 | 诊断故障代码 |
| 欧洲央行 | 电子密码本 |
| ECC | 椭圆曲线密码学 |
| ECDSA | 椭圆曲线数字签名算法 |
| ECIES | 椭圆曲线综合加密方案 |
| EDDSA | 爱德华兹曲线数字签名算法 |
| 全域名 | 完全限定域名 |
| GCM | 加里奥斯/计数器模式 |
| HMAC | 基于散列的消息认证码 |
| HTTP | 超文本传输协议 |
| ID | 标识符 |
| 知识产权 | 互联网协议 |
| 国际标准化组织 | 国际标准化组织 |
| JSON | JavaScript 对象表示法 |
| 局域网 | 局域网 |
| 苹果电脑 | 媒体访问控制 |
| 苹果电脑 | 消息验证码 |
| 医学博士 | 信息摘要 |
| MTU | 最大传输单位 |
| 纳米 | 网络管理 |
| 内华达州 | 非挥发性 |
| PHM | 平台健康管理 |
| PKCS | 公钥加密标准 |
| POSIX | 便携式操作系统接口 |
| PSK | 预共享密钥 |
| 只读存储器 | 只读存储器 |
| RSA | 根据 Rivest、Shamir 和  阿德尔曼 |
| 标清 | 服务发现 |
| 可持续发展目标 | 特殊数据组 |
| 沙 | 安全哈希算法 |
| 一些/IP | IP上可扩展的面向服务的中间件 |
| SWC | 软件组件 |

5 4

|  |  |
| --- | --- |
| *缩写* | *意义* |
| TLS | 传输层安全 |
| TLV | 标签长度值 |
| TTL | 生存时间 |
| UDS | 统一诊断服务 |
| UML | 统一建模语言 |
| URI | 统一资源标识符 |
| 网址 | 统一资源定位器 |
| UUID | 通用唯一标识符 |
| VLAN | 虚拟局域网 |
| VSA | 可变大小数组 |

**表 1.1：本文档范围内使用的术语和缩略语**

## 1.4 文档约定

技术术语以等距字体排版，例如PortPrototype 。作为一般规则，技术术语的复数形式是通过在单数形式中添加“s”来创建的，例如PortPrototype s。通过这种方式，该文档类似于 AUTOSAR XML Schema 中使用的术语。

d字符开始并以c字符结束的实际约束文本与文本的其余部分区分开来。

这些约束的目的是从字面上约束 AUTOSAR 元模型的解释，以便可以检测在元模型实例（即 M1 级别）中实施的标准化行为的违反。

鼓励 AUTOSAR 工具的制造商添加与 M1 建模问题相对应的约束的数字 ID，作为工具发出的诊断消息的一部分。

本文档介绍的类的属性以类表的形式列出。它们具有顶级元素 AUTOSAR 示例中所示的形式：

请注意，在 AUTOSAR 工作流程中，约束不应在任何给定时间强制执行。在模型的开发过程中，可能会合理地违反约束，因为不完整的模型会明显显示出不一致。

但是，在工作流的特定点，应强制执行约束以防止配置错误。

工作流中应强制执行约束的点，有时也称为约束的“绑定时间”，对于每个模型类别都是不同的，例如在经典平台上，为软件组件定义的约束通常在RTE 的生成，而对 Ecu 提取定义的约束应在创建 Com 堆栈的 Ecu 配置时应用。

对于每个文档，定义了可能的约束绑定时间，并且约束本身通常会提到绑定时间，以便为 AUTOSAR 创作工具的实施者提供正确的方向。

让AUTOSAR是一个典型的类表的例子。表中的第一行具有以下含义：

**Class** ：在 UML 模型中定义的类的名称。

**包**：定义类的 UML 包。列出它只是为了帮助在整个元模型中定位类。

**注意**：建模者对课程的评论（课程笔记）。类的刻板印象和 UML 标签也在这里表示。

**基类**：如果适用，直接基类列表。

表中的标题具有以下含义：

**属性**：类的属性名称。请注意，AUTOSAR 不区分类属性和拥有的关联端。

**Type** ：类的属性的类型。

**穆尔。** ：属性的分配多重性，即给定数据类型的多少实例与属性相关联。

**Kind** ：指定属性是在类中聚合（aggr 聚合）、类中的 UML 属性（attr 原始属性），还是仅被它引用（ref 引用）。在该字段中还指示了实例引用（iref 实例引用）。

**注意**：建模者对类属性的注释（角色注释）。类的刻板印象和 UML 标签也在这里表示。

请注意，以字母而不是数值开头的章节代表文档的附录。附录的目的是支持对文档某些方面的解释，并不代表标准的约束性约定。

[TPS\_STDT\_00053] 中规定的义务表达的口头形式应用于指示要求，见标准化模板，支持一章

可追溯性（[ 5 ]）。

AUTOSAR 文档中的需求表示遵循 [TPS\_STDT\_00078] 中指定的表格，请参阅标准化模板，支持可追溯性一章

（[ 5 ]）。

## 1.5 需求追踪

针对本文件的要求仅在相应的要求文件中说明。

下表1.2引用了相应要求文档中指定的要求，并提供了有关满足给定要求的各个规范项的信息。

|  |  |  |
| --- | --- | --- |
| **要求** | **描述** | **满意** |
| **[RS\_MANI\_00001]** | 自适应 AUTOSAR 应用 | [ TPS\_MANI\_01001 ] [ TPS\_MANI\_01010 ] |
| **[RS\_MANI\_00002]** | 应用程序中提供和所需服务的声明 | [ TPS\_MANI\_01039 ] [ TPS\_MANI\_01040 ] [ TPS\_MANI\_01053 ]  [ TPS\_MANI\_01057 ][ TPS\_MANI\_01190 ][ TPS\_MANI\_03210 ]  [ TPS\_MANI\_03211 ] [ TPS\_MANI\_03212 ] |
| **[RS\_MANI\_00003]** | 服务接口规范 | [ TPS\_MANI\_01001 ] [ TPS\_MANI\_01004 ] [ TPS\_MANI\_01005 ]  [ TPS\_MANI\_01006 ] [ TPS\_MANI\_01007 ] [ TPS\_MANI\_01033 ]  [ TPS\_MANI\_01034 ] [ TPS\_MANI\_01035 ] [ TPS\_MANI\_01064 ]  [ TPS\_MANI\_03118 ][ TPS\_MANI\_03119 ][ TPS\_MANI\_03223 ][ TPS\_MANI\_03291 ] |
| **[RS\_MANI\_00004]** | 应用程序设计支持 | [ TPS\_MANI\_01010 ] [ TPS\_MANI\_01228 ] [ TPS\_MANI\_01229 ] |
| **[RS\_MANI\_00005]** | 配置应用程序的诊断功能 | [ TPS\_MANI\_01048 ] [ TPS\_MANI\_01049 ] [ TPS\_MANI\_01050 ]  [ TPS\_MANI\_01230 ] [ TPS\_MANI\_01259 ] [ TPS\_MANI\_01260 ]  [ TPS\_MANI\_01261 ] [ TPS\_MANI\_01262 ] [ TPS\_MANI\_01263 ]  [ TPS\_MANI\_01326 ] [ TPS\_MANI\_01350 ] [ TPS\_MANI\_01351 ]  [ TPS\_MANI\_01352 ] [ TPS\_MANI\_01358 ] [ TPS\_MANI\_01360 ]  [ TPS\_MANI\_01361 ] [ TPS\_MANI\_01362 ] |
| **[RS\_MANI\_00006]** | 支持应用部署 | [ TPS\_MANI\_01011 ] [ TPS\_MANI\_01308 ] [ TPS\_MANI\_01337 ]  [ TPS\_MANI\_03147 ] |
| **[RS\_MANI\_00007]** | 应用程序启动行为的配置 | [ TPS\_MANI\_01012 ] [ TPS\_MANI\_01013 ] [ TPS\_MANI\_01017 ]  [ TPS\_MANI\_01041 ] [ TPS\_MANI\_01046 ] [ TPS\_MANI\_01061 ]  [ TPS\_MANI\_01188 ] [ TPS\_MANI\_01209 ] [ TPS\_MANI\_01277 ]  [ TPS\_MANI\_01278 ][ TPS\_MANI\_01328 ][ TPS\_MANI\_01334 ][ TPS\_MANI\_03151 ] |
| **[RS\_MANI\_00008]** | 服务接口部署到传输层机制 | [ TPS\_MANI\_01136 ] [ TPS\_MANI\_01137 ] [ TPS\_MANI\_01210 ]  [ TPS\_MANI\_03036 ] [ TPS\_MANI\_03037 ] [ TPS\_MANI\_03038 ]  [ TPS\_MANI\_03039 ] [ TPS\_MANI\_03070 ] [ TPS\_MANI\_03071 ]  [ TPS\_MANI\_03072 ] [ TPS\_MANI\_03073 ] [ TPS\_MANI\_03074 ]  [ TPS\_MANI\_03101 ] [ TPS\_MANI\_03103 ] [ TPS\_MANI\_03104 ]  [ TPS\_MANI\_03105 ] [ TPS\_MANI\_03106 ] [ TPS\_MANI\_03107 ]  [ TPS\_MANI\_03108 ] [ TPS\_MANI\_03116 ] [ TPS\_MANI\_03117 ]  [ TPS\_MANI\_03217 ][ TPS\_MANI\_03235 ][ TPS\_MANI\_03278 ][ TPS\_MANI\_03288 ] |
| **[RS\_MANI\_00009]** | 网络级别的服务实例配置 | [ TPS\_MANI\_01316 ] [ TPS\_MANI\_01317 ] [ TPS\_MANI\_03001 ]  [ TPS\_MANI\_03002 ] [ TPS\_MANI\_03003 ] [ TPS\_MANI\_03004 ]  [ TPS\_MANI\_03005 ] [ TPS\_MANI\_03006 ] [ TPS\_MANI\_03007 ]  [ TPS\_MANI\_03008 ] [ TPS\_MANI\_03009 ] [ TPS\_MANI\_03010 ]  [ TPS\_MANI\_03022 ] [ TPS\_MANI\_03023 ] [ TPS\_MANI\_03024 ]  [ TPS\_MANI\_03049 ][ TPS\_MANI\_03061 ][ TPS\_MANI\_03236 ][ TPS\_MANI\_03237 ][ TPS\_MANI\_03554 ][ TPS\_MANI\_03555 ][ TPS\_MANI\_03619 ] |
| **[RS\_MANI\_00011]** | 应用程序中提供和所需服务的实例化 | [ TPS\_MANI\_01275 ] [ TPS\_MANI\_01276 ] [ TPS\_MANI\_01282 ]  [ TPS\_MANI\_03000 ] |
| **[RS\_MANI\_00014]** | 用户定义的传输层机制 | [ TPS\_MANI\_01165 ] [ TPS\_MANI\_03032 ] [ TPS\_MANI\_03045 ]  [ TPS\_MANI\_03046 ] [ TPS\_MANI\_03047 ] [ TPS\_MANI\_03048 ]  [ TPS\_MANI\_03102 ] [ TPS\_MANI\_03280 ] [ TPS\_MANI\_03281 ] |
| **[RS\_MANI\_00015]** | 清单性质的定义 | [ TPS\_MANI\_01000 ] [ TPS\_MANI\_01019 ] [ TPS\_MANI\_01020 ]  [ TPS\_MANI\_01021 ] |

5

|  |  |  |
| --- | --- | --- |
| **要求** | **描述** | **满意** |
| **[RS\_MANI\_00016]** | 在 AUTOSAR 自适应平台上专门使用数据类型 | [ TPS\_MANI\_01016 ] [ TPS\_MANI\_01027 ] [ TPS\_MANI\_01047 ]  [ TPS\_MANI\_01100 ] [ TPS\_MANI\_01309 ] |
| **[RS\_MANI\_00017]** | 映射规范  服务接口 | [ TPS\_MANI\_01002 ] [ TPS\_MANI\_01003 ] [ TPS\_MANI\_01022 ]  [ TPS\_MANI\_01024 ] [ TPS\_MANI\_01025 ] [ TPS\_MANI\_01026 ]  [ TPS\_MANI\_01032 ] [ TPS\_MANI\_03277 ] [ TPS\_MANI\_03289 ] |
| **[RS\_MANI\_00018]** | 机器的网络连接 | [ TPS\_MANI\_03052 ] [ TPS\_MANI\_03053 ] |
| **[RS\_MANI\_00019]** | 服务发现消息交换配置 | [ TPS\_MANI\_03064 ] |
| **[RS\_MANI\_00020]** | 机器硬件资源 | [ TPS\_MANI\_01269 ] [ TPS\_MANI\_03035 ] [ TPS\_MANI\_03148 ] |
| **[RS\_MANI\_00021]** | 机器状态描述 | [ TPS\_MANI\_03035 ] |
| **[RS\_MANI\_00022]** | 自适应平台配置 | [ TPS\_MANI\_01208 ] [ TPS\_MANI\_01273 ] [ TPS\_MANI\_03035 ] |
| **[RS\_MANI\_00023]** | 自适应模块配置 | [ TPS\_MANI\_01208 ] [ TPS\_MANI\_01226 ] [ TPS\_MANI\_01227 ]  [ TPS\_MANI\_01271 ] [ TPS\_MANI\_01279 ] [ TPS\_MANI\_01343 ]  [ TPS\_MANI\_02384 ] [ TPS\_MANI\_02385 ] [ TPS\_MANI\_02386 ]  [ TPS\_MANI\_02387 ] [ TPS\_MANI\_02388 ] [ TPS\_MANI\_03035 ]  [ TPS\_MANI\_03056 ] [ TPS\_MANI\_03096 ] [ TPS\_MANI\_03098 ]  [ TPS\_MANI\_03162 ] [ TPS\_MANI\_03163 ] [ TPS\_MANI\_03164 ]  [ TPS\_MANI\_03165 ] [ TPS\_MANI\_03166 ] [ TPS\_MANI\_03167 ]  [ TPS\_MANI\_03218 ] [ TPS\_MANI\_03219 ] [ TPS\_MANI\_03220 ]  [ TPS\_MANI\_03221 ] [ TPS\_MANI\_03222 ] [ TPS\_MANI\_03226 ]  [ TPS\_MANI\_03260 ] [ TPS\_MANI\_03261 ] [ TPS\_MANI\_03262 ]  [ TPS\_MANI\_03263 ] [ TPS\_MANI\_03264 ] [ TPS\_MANI\_03265 ]  [ TPS\_MANI\_03266 ] [ TPS\_MANI\_03267 ] [ TPS\_MANI\_03268 ]  [ TPS\_MANI\_03269 ] [ TPS\_MANI\_03270 ] [ TPS\_MANI\_03271 ]  [ TPS\_MANI\_03272 ] [ TPS\_MANI\_03273 ] [ TPS\_MANI\_03274 ]  [ TPS\_MANI\_03276 ] [ TPS\_MANI\_03279 ] [ TPS\_MANI\_03282 ]  [ TPS\_MANI\_03283 ] [ TPS\_MANI\_03285 ] [ TPS\_MANI\_03286 ]  [ TPS\_MANI\_03502 ] [ TPS\_MANI\_03503 ] [ TPS\_MANI\_03505 ]  [ TPS\_MANI\_03506 ] [ TPS\_MANI\_03508 ] [ TPS\_MANI\_03509 ]  [ TPS\_MANI\_03510 ] [ TPS\_MANI\_03511 ] [ TPS\_MANI\_03512 ]  [ TPS\_MANI\_03513 ] [ TPS\_MANI\_03514 ] [ TPS\_MANI\_03515 ]  [ TPS\_MANI\_03516 ] [ TPS\_MANI\_03517 ] [ TPS\_MANI\_03544 ]  [ TPS\_MANI\_03545 ] [ TPS\_MANI\_03546 ] [ TPS\_MANI\_03553 ]  [ TPS\_MANI\_03573 ] [ TPS\_MANI\_03574 ] [ TPS\_MANI\_03575 ]  [ TPS\_MANI\_03576 ] [ TPS\_MANI\_03625 ] [ TPS\_MANI\_03626 ]  [ TPS\_MANI\_03633 ] [ TPS\_MANI\_03651 ] |
| **[RS\_MANI\_00024]** | SOME/IP 传输层机制 | [ TPS\_MANI\_01136 ] [ TPS\_MANI\_01137 ] [ TPS\_MANI\_03002 ]  [ TPS\_MANI\_03003 ] [ TPS\_MANI\_03004 ] [ TPS\_MANI\_03005 ]  [ TPS\_MANI\_03006 ] [ TPS\_MANI\_03007 ] [ TPS\_MANI\_03008 ]  [ TPS\_MANI\_03009 ] [ TPS\_MANI\_03010 ] [ TPS\_MANI\_03011 ]  [ TPS\_MANI\_03012 ] [ TPS\_MANI\_03013 ] [ TPS\_MANI\_03014 ]  [ TPS\_MANI\_03015 ] [ TPS\_MANI\_03016 ] [ TPS\_MANI\_03017 ]  [ TPS\_MANI\_03018 ] [ TPS\_MANI\_03020 ] [ TPS\_MANI\_03021 ]  [ TPS\_MANI\_03022 ] [ TPS\_MANI\_03023 ] [ TPS\_MANI\_03024 ]  [ TPS\_MANI\_03025 ] [ TPS\_MANI\_03026 ] [ TPS\_MANI\_03027 ]  [ TPS\_MANI\_03028 ] [ TPS\_MANI\_03029 ] [ TPS\_MANI\_03030 ]  [ TPS\_MANI\_03031 ] [ TPS\_MANI\_03040 ] [ TPS\_MANI\_03041 ]  [ TPS\_MANI\_03042 ] [ TPS\_MANI\_03043 ] [ TPS\_MANI\_03044 ]  [ TPS\_MANI\_03049 ] [ TPS\_MANI\_03050 ] [ TPS\_MANI\_03051 ]  [ TPS\_MANI\_03057 ] [ TPS\_MANI\_03059 ] [ TPS\_MANI\_03061 ]  [ TPS\_MANI\_03067 ] [ TPS\_MANI\_03068 ] [ TPS\_MANI\_03069 ]  [ TPS\_MANI\_03070 ] [ TPS\_MANI\_03071 ] [ TPS\_MANI\_03072 ]  [ TPS\_MANI\_03073 ] [ TPS\_MANI\_03074 ] [ TPS\_MANI\_03116 ]  [ TPS\_MANI\_03154 ] [ TPS\_MANI\_03155 ] [ TPS\_MANI\_03156 ]  [ TPS\_MANI\_03157 ] [ TPS\_MANI\_03158 ] [ TPS\_MANI\_03159 ]  [ TPS\_MANI\_03168 ] [ TPS\_MANI\_03217 ] [ TPS\_MANI\_03227 ]  [ TPS\_MANI\_03230 ][ TPS\_MANI\_03231 ][ TPS\_MANI\_03235 ][ TPS\_MANI\_03237 ][ TPS\_MANI\_03278 ][ TPS\_MANI\_03554 ][ TPS\_MANI\_03555 ] |

5

|  |  |  |
| --- | --- | --- |
| **要求** | **描述** | **满意** |
| **[RS\_MANI\_00025]** | 序列化的定义和配置 | [ TPS\_MANI\_01210 ] [ TPS\_MANI\_03101 ] [ TPS\_MANI\_03102 ]  [ TPS\_MANI\_03103 ] [ TPS\_MANI\_03104 ] [ TPS\_MANI\_03105 ]  [ TPS\_MANI\_03106 ] [ TPS\_MANI\_03107 ] [ TPS\_MANI\_03108 ]  [ TPS\_MANI\_03109 ] [ TPS\_MANI\_03117 ] [ TPS\_MANI\_03288 ] |
| **[RS\_MANI\_00026]** | 软件组件系统设计 | [ TPS\_MANI\_01191 ] [ TPS\_MANI\_01192 ] [ TPS\_MANI\_01198 ]  [ TPS\_MANI\_03110 ] [ TPS\_MANI\_03111 ] [ TPS\_MANI\_03112 ]  [ TPS\_MANI\_03113 ] [ TPS\_MANI\_03114 ] [ TPS\_MANI\_03115 ] |
| **[RS\_MANI\_00027]** | 支持访问持久数据 | [ TPS\_MANI\_01065 ] [ TPS\_MANI\_01067 ] [ TPS\_MANI\_01068 ]  [ TPS\_MANI\_01073 ] [ TPS\_MANI\_01078 ] [ TPS\_MANI\_01079 ]  [ TPS\_MANI\_01080 ] [ TPS\_MANI\_01081 ] [ TPS\_MANI\_01135 ]  [ TPS\_MANI\_01138 ] [ TPS\_MANI\_01139 ] [ TPS\_MANI\_01140 ]  [ TPS\_MANI\_01142 ] [ TPS\_MANI\_01144 ] [ TPS\_MANI\_01146 ]  [ TPS\_MANI\_01147 ] [ TPS\_MANI\_01148 ] [ TPS\_MANI\_01149 ]  [ TPS\_MANI\_01150 ] [ TPS\_MANI\_01155 ] [ TPS\_MANI\_01156 ]  [ TPS\_MANI\_01157 ] [ TPS\_MANI\_01159 ] [ TPS\_MANI\_01160 ]  [ TPS\_MANI\_01179 ] [ TPS\_MANI\_01180 ] [ TPS\_MANI\_01182 ]  [ TPS\_MANI\_01187 ] [ TPS\_MANI\_01194 ] [ TPS\_MANI\_01196 ]  [ TPS\_MANI\_01197 ] [ TPS\_MANI\_01204 ] [ TPS\_MANI\_01205 ]  [ TPS\_MANI\_01206 ] [ TPS\_MANI\_01207 ] [ TPS\_MANI\_01313 ]  [ TPS\_MANI\_01314 ] [ TPS\_MANI\_01315 ] [ TPS\_MANI\_01319 ]  [ TPS\_MANI\_01320 ][ TPS\_MANI\_01321 ][ TPS\_MANI\_01322 ][ TPS\_MANI\_01323 ] |
| **[RS\_MANI\_00028]** | 安全保护配置 | [ TPS\_MANI\_01324 ] [ TPS\_MANI\_01325 ] [ TPS\_MANI\_01327 ]  [ TPS\_MANI\_03127 ] [ TPS\_MANI\_03128 ] [ TPS\_MANI\_03129 ]  [ TPS\_MANI\_03130 ] [ TPS\_MANI\_03131 ] [ TPS\_MANI\_03132 ]  [ TPS\_MANI\_03228 ] [ TPS\_MANI\_03229 ] [ TPS\_MANI\_03252 ] |
| **[RS\_MANI\_00029]** | 之间的映射描述  基于信号的通信和  面向服务的通信 | [ TPS\_MANI\_03124 ] [ TPS\_MANI\_03125 ] [ TPS\_MANI\_03126 ]  [ TPS\_MANI\_03627 ] [ TPS\_MANI\_03629 ] [ TPS\_MANI\_03635 ] |
| **[RS\_MANI\_00030]** | 复合数据结构中可选元素的定义 | [ TPS\_MANI\_01097 ] [ TPS\_MANI\_01184 ] [ TPS\_MANI\_01185 ]  [ TPS\_MANI\_01186 ] [ TPS\_MANI\_01270 ] [ TPS\_MANI\_01333 ] |
| **[RS\_MANI\_00031]** | 与加密软件的交互 | [ TPS\_MANI\_03253 ] [ TPS\_MANI\_03254 ] [ TPS\_MANI\_03255 ]  [ TPS\_MANI\_03256 ][ TPS\_MANI\_03257 ][ TPS\_MANI\_03258 ][ TPS\_MANI\_03259 ] |
| **[RS\_MANI\_00032]** | 支持平台健康管理 | [ TPS\_MANI\_01280 ] [ TPS\_MANI\_03500 ] [ TPS\_MANI\_03502 ]  [ TPS\_MANI\_03503 ] [ TPS\_MANI\_03505 ] [ TPS\_MANI\_03506 ]  [ TPS\_MANI\_03508 ] [ TPS\_MANI\_03509 ] [ TPS\_MANI\_03510 ]  [ TPS\_MANI\_03511 ] [ TPS\_MANI\_03512 ] [ TPS\_MANI\_03513 ]  [ TPS\_MANI\_03514 ][ TPS\_MANI\_03515 ][ TPS\_MANI\_03516 ]  [ TPS\_MANI\_03517 ][ TPS\_MANI\_03534 ][ TPS\_MANI\_03544 ]  [ TPS\_MANI\_03545 ] [ TPS\_MANI\_03546 ] [ TPS\_MANI\_03553 ]  [ TPS\_MANI\_03573 ] [ TPS\_MANI\_03574 ] [ TPS\_MANI\_03575 ]  [ TPS\_MANI\_03576 ] [ TPS\_MANI\_03623 ] [ TPS\_MANI\_03624 ]  [ TPS\_MANI\_03625 ] [ TPS\_MANI\_03626 ] [ TPS\_MANI\_03630 ]  [ TPS\_MANI\_03631 ] [ TPS\_MANI\_03633 ] [ TPS\_MANI\_03651 ] |
| **[RS\_MANI\_00034]** | 意向规范 | [ TPS\_MANI\_01106 ] [ TPS\_MANI\_01107 ] [ TPS\_MANI\_01108 ]  [ TPS\_MANI\_03209 ] |
| **[RS\_MANI\_00035]** | 可上传软件包的定义 | [ TPS\_MANI\_01109 ] [ TPS\_MANI\_01110 ] [ TPS\_MANI\_01111 ]  [ TPS\_MANI\_01112 ] [ TPS\_MANI\_01113 ] [ TPS\_MANI\_01114 ]  [ TPS\_MANI\_01115 ] [ TPS\_MANI\_01116 ] [ TPS\_MANI\_01117 ]  [ TPS\_MANI\_01118 ] [ TPS\_MANI\_01119 ] [ TPS\_MANI\_01161 ]  [ TPS\_MANI\_01164 ] [ TPS\_MANI\_01189 ] [ TPS\_MANI\_01202 ]  [ TPS\_MANI\_01211 ] [ TPS\_MANI\_01213 ] [ TPS\_MANI\_01214 ]  [ TPS\_MANI\_01215 ] [ TPS\_MANI\_01216 ] [ TPS\_MANI\_01217 ]  [ TPS\_MANI\_01218 ] [ TPS\_MANI\_01219 ] [ TPS\_MANI\_01220 ]  [ TPS\_MANI\_01221 ] [ TPS\_MANI\_01222 ] [ TPS\_MANI\_01223 ]  [ TPS\_MANI\_01225 ] [ TPS\_MANI\_01310 ] [ TPS\_MANI\_01329 ]  [ TPS\_MANI\_01331 ] [ TPS\_MANI\_01335 ] [ TPS\_MANI\_01344 ]  [ TPS\_MANI\_01345 ] [ TPS\_MANI\_01346 ] [ TPS\_MANI\_01349 ] |

5

|  |  |  |
| --- | --- | --- |
| **要求** | **描述** | **满意** |
| **[RS\_MANI\_00036]** | 安全防护配置 | [ TPS\_MANI\_03133 ] [ TPS\_MANI\_03134 ] [ TPS\_MANI\_03137 ]  [ TPS\_MANI\_03138 ] [ TPS\_MANI\_03139 ] [ TPS\_MANI\_03140 ]  [ TPS\_MANI\_03199 ] [ TPS\_MANI\_03200 ] [ TPS\_MANI\_03203 ]  [ TPS\_MANI\_03204 ] [ TPS\_MANI\_03205 ] [ TPS\_MANI\_03206 ]  [ TPS\_MANI\_03207 ] [ TPS\_MANI\_03208 ] [ TPS\_MANI\_03213 ]  [ TPS\_MANI\_03214 ] [ TPS\_MANI\_03215 ] [ TPS\_MANI\_03216 ]  [ TPS\_MANI\_03232 ] [ TPS\_MANI\_03233 ] [ TPS\_MANI\_03234 ]  [ TPS\_MANI\_03240 ][ TPS\_MANI\_03241 ][ TPS\_MANI\_03242 ][ TPS\_MANI\_03661 ] |
| **[RS\_MANI\_00037]** | 日志和跟踪的配置 | [ TPS\_MANI\_01271 ] [ TPS\_MANI\_03284 ] |
| **[RS\_MANI\_00038]** | DDS 传输层机制 | [ TPS\_MANI\_03525 ] [ TPS\_MANI\_03526 ] [ TPS\_MANI\_03527 ]  [ TPS\_MANI\_03528 ] [ TPS\_MANI\_03529 ] [ TPS\_MANI\_03530 ]  [ TPS\_MANI\_03531 ] [ TPS\_MANI\_03532 ] [ TPS\_MANI\_03533 ]  [ TPS\_MANI\_03556 ] [ TPS\_MANI\_03557 ] [ TPS\_MANI\_03558 ]  [ TPS\_MANI\_03561 ] [ TPS\_MANI\_03562 ] [ TPS\_MANI\_03567 ]  [ TPS\_MANI\_03568 ][ TPS\_MANI\_03622 ][ TPS\_MANI\_03650 ][ TPS\_MANI\_03662 ] |
| **[RS\_MANI\_00039]** | 实现特定数据类型的使用 | [ TPS\_MANI\_01166 ] [ TPS\_MANI\_01167 ] [ TPS\_MANI\_01168 ]  [ TPS\_MANI\_01169 ] [ TPS\_MANI\_01171 ] [ TPS\_MANI\_01172 ]  [ TPS\_MANI\_01173 ] [ TPS\_MANI\_01174 ] [ TPS\_MANI\_01175 ]  [ TPS\_MANI\_01176 ] [ TPS\_MANI\_01177 ] [ TPS\_MANI\_01201 ]  [ TPS\_MANI\_01212 ] [ TPS\_MANI\_03169 ] [ TPS\_MANI\_03170 ]  [ TPS\_MANI\_03171 ] [ TPS\_MANI\_03172 ] [ TPS\_MANI\_03173 ]  [ TPS\_MANI\_03174 ] [ TPS\_MANI\_03175 ] [ TPS\_MANI\_03176 ]  [ TPS\_MANI\_03177 ] [ TPS\_MANI\_03178 ] [ TPS\_MANI\_03179 ]  [ TPS\_MANI\_03180 ] [ TPS\_MANI\_03181 ] [ TPS\_MANI\_03183 ]  [ TPS\_MANI\_03184 ] [ TPS\_MANI\_03185 ] [ TPS\_MANI\_03186 ]  [ TPS\_MANI\_03187 ] [ TPS\_MANI\_03188 ] [ TPS\_MANI\_03189 ]  [ TPS\_MANI\_03190 ] [ TPS\_MANI\_03191 ] [ TPS\_MANI\_03192 ]  [ TPS\_MANI\_03193 ] [ TPS\_MANI\_03196 ] [ TPS\_MANI\_03197 ]  [ TPS\_MANI\_03198 ] [ TPS\_MANI\_03201 ] [ TPS\_MANI\_03202 ] |
| **[RS\_MANI\_00040]** | 支持访问同步时间 | [ TPS\_MANI\_03535 ] [ TPS\_MANI\_03536 ] [ TPS\_MANI\_03537 ]  [ TPS\_MANI\_03539 ] [ TPS\_MANI\_03541 ] [ TPS\_MANI\_03542 ]  [ TPS\_MANI\_03543 ] [ TPS\_MANI\_03547 ] [ TPS\_MANI\_03548 ]  [ TPS\_MANI\_03549 ] [ TPS\_MANI\_03551 ] [ TPS\_MANI\_03632 ] |
| **[RS\_MANI\_00041]** | 功能组配置 | [ TPS\_MANI\_01330 ] [ TPS\_MANI\_03145 ] [ TPS\_MANI\_03152 ]  [ TPS\_MANI\_03194 ] [ TPS\_MANI\_03195 ] |
| **[RS\_MANI\_00050]** | 确定性客户端的支持 | [ TPS\_MANI\_01199 ] [ TPS\_MANI\_01200 ] [ TPS\_MANI\_01203 ] |
| **[RS\_MANI\_00060]** | 身份和访问支持  管理 | [ TPS\_MANI\_01231 ] [ TPS\_MANI\_01232 ] [ TPS\_MANI\_01233 ]  [ TPS\_MANI\_01234 ] [ TPS\_MANI\_01235 ] [ TPS\_MANI\_01236 ]  [ TPS\_MANI\_01237 ] [ TPS\_MANI\_01238 ] [ TPS\_MANI\_01239 ]  [ TPS\_MANI\_01240 ] [ TPS\_MANI\_01241 ] [ TPS\_MANI\_01284 ]  [ TPS\_MANI\_01307 ] [ TPS\_MANI\_03238 ] [ TPS\_MANI\_03239 ]  [ TPS\_MANI\_03240 ] [ TPS\_MANI\_03241 ] [ TPS\_MANI\_03242 ]  [ TPS\_MANI\_03244 ][ TPS\_MANI\_03245 ][ TPS\_MANI\_03246 ]  [ TPS\_MANI\_03247 ] [ TPS\_MANI\_03248 ] [ TPS\_MANI\_03249 ]  [ TPS\_MANI\_03250 ] [ TPS\_MANI\_03251 ] [ TPS\_MANI\_03290 ] |
| **[RS\_MANI\_00061]** | 支持诊断接口 | [ TPS\_MANI\_01048 ] [ TPS\_MANI\_01049 ] [ TPS\_MANI\_01050 ]  [ TPS\_MANI\_01242 ] [ TPS\_MANI\_01243 ] [ TPS\_MANI\_01244 ]  [ TPS\_MANI\_01245 ] [ TPS\_MANI\_01246 ] [ TPS\_MANI\_01247 ]  [ TPS\_MANI\_01248 ] [ TPS\_MANI\_01249 ] [ TPS\_MANI\_01250 ]  [ TPS\_MANI\_01251 ] [ TPS\_MANI\_01252 ] [ TPS\_MANI\_01253 ]  [ TPS\_MANI\_01254 ] [ TPS\_MANI\_01255 ] [ TPS\_MANI\_01259 ]  [ TPS\_MANI\_01260 ] [ TPS\_MANI\_01261 ] [ TPS\_MANI\_01262 ]  [ TPS\_MANI\_01263 ] [ TPS\_MANI\_01265 ] [ TPS\_MANI\_01326 ]  [ TPS\_MANI\_01332 ] [ TPS\_MANI\_01347 ] [ TPS\_MANI\_01348 ]  [ TPS\_MANI\_01351 ][ TPS\_MANI\_01352 ][ TPS\_MANI\_01353 ][ TPS\_MANI\_01359 ][ TPS\_MANI\_01360 ][ TPS\_MANI\_01361 ]  [ TPS\_MANI\_01362 ] [ TPS\_MANI\_01363 ] |
| **[RS\_MANI\_00062]** | 支持部分网络 | [ TPS\_MANI\_03224 ] [ TPS\_MANI\_03225 ] |

5

|  |  |  |
| --- | --- | --- |
| **要求** | **描述** | **满意** |
| **[RS\_MANI\_00063]** | 清单规范应支持基于信号和面向服务的通信之间的转换 | [ TPS\_MANI\_03287 ] [ TPS\_MANI\_03577 ] [ TPS\_MANI\_03578 ]  [ TPS\_MANI\_03579 ] [ TPS\_MANI\_03580 ] [ TPS\_MANI\_03581 ]  [ TPS\_MANI\_03582 ] [ TPS\_MANI\_03583 ] [ TPS\_MANI\_03585 ]  [ TPS\_MANI\_03589 ] [ TPS\_MANI\_03590 ] [ TPS\_MANI\_03591 ]  [ TPS\_MANI\_03592 ] [ TPS\_MANI\_03593 ] [ TPS\_MANI\_03594 ]  [ TPS\_MANI\_03595 ] [ TPS\_MANI\_03597 ] [ TPS\_MANI\_03598 ]  [ TPS\_MANI\_03599 ] [ TPS\_MANI\_03600 ] [ TPS\_MANI\_03601 ]  [ TPS\_MANI\_03602 ] [ TPS\_MANI\_03603 ] [ TPS\_MANI\_03604 ]  [ TPS\_MANI\_03605 ] [ TPS\_MANI\_03606 ] [ TPS\_MANI\_03607 ]  [ TPS\_MANI\_03608 ] [ TPS\_MANI\_03609 ] [ TPS\_MANI\_03610 ]  [ TPS\_MANI\_03611 ] [ TPS\_MANI\_03612 ] [ TPS\_MANI\_03614 ]  [ TPS\_MANI\_03615 ] [ TPS\_MANI\_03620 ] [ TPS\_MANI\_03621 ]  [ TPS\_MANI\_03636 ] [ TPS\_MANI\_03637 ] [ TPS\_MANI\_03638 ]  [ TPS\_MANI\_03639 ] [ TPS\_MANI\_03640 ] [ TPS\_MANI\_03641 ]  [ TPS\_MANI\_03642 ] [ TPS\_MANI\_03643 ] [ TPS\_MANI\_03644 ]  [ TPS\_MANI\_03645 ] [ TPS\_MANI\_03646 ] [ TPS\_MANI\_03647 ]  [ TPS\_MANI\_03648 ] [ TPS\_MANI\_03649 ] [ TPS\_MANI\_03652 ]  [ TPS\_MANI\_03653 ] [ TPS\_MANI\_03654 ] [ TPS\_MANI\_03655 ]  [ TPS\_MANI\_03656 ] [ TPS\_MANI\_03657 ] [ TPS\_MANI\_03658 ]  [ TPS\_MANI\_03659 ] [ TPS\_MANI\_03660 ] |
| **[RS\_MANI\_00064]** | 服务接口的服务合同版本 | [ TPS\_MANI\_03616 ] |
| **[RS\_MANI\_00065]** | 所有人的服务合同版本控制  传输部署协议 | [ TPS\_MANI\_03617 ] |
| **[RS\_MANI\_00066]** | 服务版本控制黑名单 | [ TPS\_MANI\_03618 ] |
| **[RS\_MANI\_00067]** | 原始数据流部署 | [ TPS\_MANI\_01285 ] [ TPS\_MANI\_01287 ] [ TPS\_MANI\_01307 ]  [ TPS\_MANI\_01354 ][ TPS\_MANI\_01355 ][ TPS\_MANI\_01356 ][ TPS\_MANI\_01357 ] |
| **[RS\_MANI\_00068]** | 支持安全事件报告接口定义 | [ TPS\_MANI\_01338 ] [ TPS\_MANI\_01339 ] [ TPS\_MANI\_01340 ] |
| **[RS\_MANI\_00069]** | 支持入侵检测系统管理器的部署定义 | [ TPS\_MANI\_01341 ] [ TPS\_MANI\_01342 ] |

**表 1.2：需求跟踪**

# 2 清单定义的大图

## 2.1 设计与部署

**2.1.1 概述**

尽管名称如此，但本文档包含对明确绑定到*设计*工作流的模型元素**和与***部署*方面有密切关系的模型元素的描述。

本文档中讨论的模型元素要么与*设计相关，*要么与*部署相关*，两组之间没有重叠。

*部署*相关的模型元素将用于上传到目标平台的模型，请参阅 [ TPS\_MANI\_01000 ]。这些模型元素主要在本文档的部分中描述，其中术语“清单”是部分标题的一部分。

**2.1.2 设计与部署模型的关系**

请注意，在许多情况下，与*部署相关的元模型部分*反映了*设计*域中的类似建模，例如 E2E 配置文件参数的定义。

*设计*和*部署*之间的关系如何影响具体的开发项目没有明确定义的偏好。 *E2E 属性*示例可能会出现以下场景：

* OEM 提供AdaptivePlatformServiceInstance的描述，包括*E2E 属性的定义*。

可以安全地假设模型的后续处理应将*E2E 属性*视为理所当然，并针对给定属性开发软件。

* 存在已通过 ComSpecs定义*E2E 属性的软件。*由于各种原因，可能会出现软件无法更新的情况，因此在*端到端属性的定义上处于“领先地位”* 。

AdaptivePlatformServiceInstance的定义可能必须尊重软件方面的现有建模。

* 也有可能发生的是，现有的定义可能被**真正**知道自己在做什么的工程师**部分**覆盖。

此外，需要注意的是，在工作流的设计和部署步骤中都使用了一些模型元素，这再次表明设计和部署之间的边界不像在*AUTOSAR 经典平台上那样容易定义*。

相比之下，本文档中描述的作为上传到目标平台的内容的一部分的其他模型元素在设计级别具有直接对应物。

这种关系的一个例子是ProcessDesign和Process的定义。在这些情况下，更容易在模型的设计和部署方面划清界限。

设计和部署模型元素之间（至少是预期的）分离的结果是，主要与设计对象相关的内容需要在部署级别上复制，作为保持实际清单内容尽可能精简的一种措施。

例如，如果部署元素需要引用PortPrototype ，并且清单中需要附加到PortPrototype的 ComSpec 中的信息来正确定义预期的语义，那么PortPrototype的所有者（即由SwComponentType类型化的SwComponentPrototype ）也必须出现在清单中。

很明显，这样一来，整个软件模型就会进入清单并不必要地夸大清单内容。

但是，在这种情况下，倾向于在部署中从设计域复制模型内容，以至少使清单内容尽可能紧凑。

然而，在某些情况下，仅仅存在对设计模型的引用就已经代表了有价值的信息。公开此功能的引用使用构造型 atpUriDef 进行修饰。

具体来说，平台上的软件可以从用 atpUriDef 修饰的 instanceRef 的内容中获取 InstanceSpecifier 的值。这是平台软件与应用层交互的一个非常重要的机制。

另一个例子是在 Platform Health Manager 的配置上下文中定义checkpointId 。

应用层 PHM 交互的建模涉及到PhmCheckpoint的值的定义。 checkpointId和特定 PhmCheckpoint 所在的PortPrototype 。使用的checkpointId是通过定义为 atpUriDef 的引用来标识的。

并且由于引用的目标不一定存在于平台上，即SupervisionCheckpoint 。 checkpointId在清单模型中被复制，以便平台软件可以访问这一重要信息（这方面也在图9.9中进行了说明）。

**2.1.3 文件结构**

*设计*和*部署*之间的划分，因此*设计*方面主要在第 3、4、5、6 节和第12节中的大部分中描述。

相反，第7章、第8章、第9章、第10章、第12.3章、第13章和第14章侧重于与*部署*相关的内容。

## 2.2 关于清单

本章将阐明*AUTOSAR 自适应平台*上下文中Manifest一词的定义。

**[TPS\_MANI\_01000]** { DRAFT }**术语Manifest** d的定义 Manifest 表示为支持*AUTOSAR 自适应平台*产品的配置而创建并上传到*AUTOSAR 自适应平台产品*的一段 AUTOSAR 模型描述，可能与其他包含清单适用的可执行代码的工件（如二进制文件）。 c *( RS\_MANI\_00015 )*

重要的是要强调Manifest的使用确实严格限于*AUTOSAR 自适应平台*，并且没有用例将概念移植到*AUTOSAR 经典平台*。

## 2.3 序列化格式

Manifest的定义与其他 AUTOSAR 模型内容的一个共同点是标准化的序列化格式。

**[TPS\_MANI\_01020]** { DRAFT } **AUTOSAR中Manifest的序列化格式**d AUTOSAR 中Manifest内容的标准化序列化格式是 ARXML。

因此，清单模型内容可以根据 AUTOSAR XML 模式进行验证。 c *( RS\_MANI\_00015 )*

TPS\_MANI\_01020 ]的一个重要结果是，没有限制只限于一个“清单文件”，即“清单”。

6 ]规范中给出的规则，内容可以分布在多个物理文件中。

**[TPS\_MANI\_01021]** { DRAFT }机器**上清单内容的序列化格式**d用于在机器上实际上传清单的序列化格式可以由平台供应商自由选择。

需要**完全保留**原始 ARXML Manifest的内容和语义。 c *( RS\_MANI\_00015 )*

可以预期，在许多情况下，上传Manifest的最佳选择仍然是 ARXML，因为自定义格式显然必须支持Manifest元模型的全部复杂性。

Application1



«manifest»

Application1Service.arxml



«manifest»

Application1Startup.arxml



«binary»

Application1Executable

**图 2.1：在一个软件交付中使用多个清单文件的示例**

请注意，元模型预见到存在从清单相关元类到设计相关元类的引用。

这些引用是为了清楚起见而创建的，但并不强制引用的内容实际上需要是可解析的。

就 AUTOSAR 建模方法而言，这转化为使用构造型 atpUriDef 对这些引用进行修饰。更多信息可以在[ 6 ]中找到。

如果引用的元类包含与清单级别相关的信息，则此信息将在清单级别上复制（这样清单级别模型不必依赖设计级别信息的可用性）。

## 2.4 范围

如前所述， Manifest的使用仅限于*AUTOSAR 自适应平台*。然而，这并不意味着在针对*AUTOSAR 自适应平台的开发项目中生成的所有 ARXML 都*被自动视为Manifest 。

事实上， *AUTOSAR 自适应平台*通常并非专门用于车辆项目。

一辆典型的车辆很可能还配备了许多在*AUTOSAR 经典平台上开发的 ECU* ，因此整个车辆的系统设计必须涵盖在 AUTOSAR 经典平台上构建的 ECU 和在*AUTOSAR 经典平台*上创建的ECU。 *AUTOSAR自适应平台*。

**[TPS\_MANI\_01019]** { DRAFT }**清单内容可能适用于*AUTOSAR 自适应平台的不同方面***d清单内容可以适用于模型的不同方面。目前， Manifest内容大致可以分为三个重点领域：

* Application-related Manifest内容描述了应用程序部署的所有方面，包括但不限于启动配置和应用程序级别的面向服务的通信端点的配置。
* Machine-related Manifest内容仅描述了一台机器的部署，即没有任何应用程序（包括平台模块）在该机器上运行。
* 服务实例相关清单描述了传输层级别的面向服务的通信如何绑定到应用程序和（在某些情况下）平台软件中的端点。

c *( RS\_MANI\_00015 )*

## 2.5 本文档中描述的清单

原则上，可以定义术语Manifest ，使得概念上只有一个“manifest”，并且每个部署方面都将在此上下文中处理。

这似乎不合适，因为很明显存在与典型开发项目的完全不同阶段相关的清单相关模型元素。

这方面被视为将Manifest一词的定义细分为五个不同分区的主要动机：

**Execution Manifest**这种Manifest用于指定在*AUTOSAR自适应平台上运行的应用程序的部署相关信息*。

执行清单与实际的可执行代码捆绑在一起，以支持将可执行代码集成到机器上。

8节中找到有关此主题的更多信息。

**服务实例清单**这种清单用于指定如何根据底层传输协议的要求配置面向服务的通信。

服务实例清单与实现面向服务通信的相应使用的实际可执行代码捆绑在一起。

10节中找到有关此主题的更多信息。

**机器清单**这种清单应该描述部署相关的内容，这些内容适用于运行*AUTOSAR 自适应平台*的底层机器（即机器上没有运行任何应用程序）的配置。

机器清单与用于建立*AUTOSAR 自适应平台*实例的软件捆绑在一起。

7节和第 9节中找到有关此主题的更多信息。

**原始数据流清单**这种清单描述了客户端和服务器的配置，目的是通过原始数据流进行通信。

**软件分发**这种清单描述了*AUTOSAR 自适应平台*上软件的打包和物流方面。

14节中找到有关此主题的更多信息。

Manifest的定义（和使用）之间的时间划分导致在大多数情况下将使用不同的物理文件来存储不同种类Manifest的内容。

但是，与所有种类的 ARXML 内容一样，这不是一个绑定规则。

# 3 应用设计

## 3.1 概述

本章描述了适用于在*AUTOSAR 自适应平台上创建应用软件的所有与设计相关的建模*。

这也扩展到*AUTOSAR 经典平台上使用的现有建模的扩展*，例如引入属性类别的新值。

特别是，本文档的这一部分重点关注以下方面：

* 为*AUTOSAR 自适应平台定义*SwComponentType的专用子类（第3.2节）
* 专门为*AUTOSAR 自适应平台定义的数据类型*（部分

3.3 )

* 服务接口作为面向服务通信的关键元素（第3.4节）
* 域特定PortInterface的定义，例如 用于诊断（第

3.10 ），PHM（第3.9节），持久性（第3.7节），加密（第3.11节）。

* 服务接口映射作为内部和外部通信之间的中介（第3.5节）
* 服务接口**元素**映射作为内部和外部通信之间的中介（第3.6节）
* 从软件组件内部的角度来看与“外部世界”交互的细粒度配置方面（第3.15节）
* 可执行为最小的可执行单元（第3.16节）
* 处理数据结构中的可选元素，见3.17节
* 转换属性的配置（第3.18节）
* 关于如何通过ProcessDesign 预先加载Process的配置的描述，请参见第3.19节。
* GrantDesign描述设计级 IAM 配置，请参阅第3.20节

## 3.2 软件组件

原则上，可以直接接管例如ApplicationSwComponentType的定义，以便在*AUTOSAR 自适应平台上使用*。

*AUTOSAR 自适应平台*独有的模型元素（例如：数据类型，如第3.3节所述）的存在的约束的制定复杂化。

因此， AdaptiveApplicationSwComponentType被定义为*AUTOSAR 自适应平台上的软件组件的表示*。

AdaptiveApplicationSwComponentType的存在提供了一种方便的方式（参见 [ constr\_1492 ]），以锁定为*AUTOSAR 经典平台定义的大多数软件组件，使其*无法在*AUTOSAR 自适应平台上使用*。

相反方向的澄清（即错误使用AdaptiveApplicationSwComponentType ）不太明显。

换句话说，可以在系统中使用AdaptiveApplicationSwComponentType作为*AUTOSAR 经典平台***和***AUTOSAR 自适应平台*上软件的某种整体设计模型。

然而，到目前为止，这方面还没有得到澄清，也没有限制AdaptiveApplicationSwComponentType出现在System的上下文中。

本规范的后续版本可能会修复缺失的规定。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **AdaptiveApplicationSwComponentType** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure | | | |
| ***笔记*** | 这个元类代表了在 AUTOSAR 自适应平台上支持应用软件形式化建模的能力。因此，它只能在 AUTOSAR 自适应平台上使用。  **标签：**  atp.Status=草稿  atp.recommendedPackage=AdaptiveApplicationSwComponentTypes | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *识别*、*多语言可引用*、可*打包元素*、*可引用*、 *SwComponentType* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 内部行为 | AdaptiveSwc内部  行为 | 0..1 | 聚合 | 此聚合表示 AUTOSAR 自适应平台的 AdaptiveApplicationSwComponentType 的内部行为。  **刻板印象：** atpSplitable； atpVariation**标签：**  atp.Splitkey=internalBehavior.shortName，内部 Behavior.variationPoint.shortLabel atp.Status=draft  vh.latestBindingTime=preCompileTime |

**表 3.1：AdaptiveApplicationSwComponentType**

|  |  |
| --- | --- |
| ***班级*** | **AdaptiveSwc内部行为** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::AdaptiveInternalBehavior |
| ***笔记*** | 此元类表示定义用于 AUTOSAR 自适应平台的 AtomicSwComponentType 的内部行为的能力。  请注意，与 AUTOSAR 经典平台的情况形成鲜明对比的是，这种情况下的内部行为模型非常小。  **标签：** atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **AdaptiveSwc内部行为** | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 服务  依赖 | swc服务  依赖 | \* | 聚合 | 这代表了 SwcService 的集合  AdaptiveInternalBehavior 拥有的依赖项。  **标签：** atp.Status=draft |

**表 3.2: AdaptiveSwcInternalBehavior**

## 3.3 数据类型

**3.3.1 概述**

*AUTOSAR 自适应平台*上的数据类型规范遵循与*AUTOSAR 经典平台*上的对应模式相同的模式：数据类型在相互补充的不同抽象级别上定义。

在本文档的上下文中，重点是对ApplicationDataType和CppImplementationDataType的讨论。

一般来说，关于数据类型定义的大部分概念都可以从*AUTOSAR 经典平台上的现有规范中继承*。

但是，某些方面特定于*AUTOSAR 自适应平台*，因此在本文档的范围内进行讨论，而不是 AUTOSAR 软件组件模板 [ 1 ] 的规范。

*AUTOSAR 经典平台*接管的方面之一是初始值的定义。

虽然初始值的效用在*AUTOSAR 自适应平台上肯定是有限的*，但仍有机会在所谓的Field的上下文中利用初始值的定义（参见 [ TPS\_MANI\_01034 ]）。

**3.3.2 应用数据类型**

*AUTOSAR经典平台*支持的ApplicationDataType的全系列建模也可以直接在*AUTOSAR自适应平台上使用*。

除了*AUTOSAR 经典平台*上支持的ApplicationDataType之外，还有其他ApplicationDataType ——虽然原则上在*AUTOSAR 经典平台上也可用*——主要用于*AUTOSAR 自适应平台并为其设计*。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***应用程序数据类型***（抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes | | | |
| ***笔记*** | ApplicationDataType 从应用程序的角度定义数据类型。尤其是当“物理”受到威胁时，应该使用它。  ApplicationDataType 表示在应用程序模型中看到的一组值，例如度量单位。它不考虑位大小、字节顺序等实现细节。  应该可以仅使用 ApplicationData 类型对 VFB 系统的应用程序级别方面进行建模。 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *AutosarDataType* ,  *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***子类*** | *ApplicationCompositeDataType* , ApplicationPrimitiveDataType | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.3：应用程序数据类型**

**3.3.2.1 字符串数据类型**

虽然表示文本字符串的数据类型的处理与*AUTOSAR 经典平台*和*AUTOSAR 自适应平台*上的ApplicationDataType定义非常相似，但在 AUTOSAR 自适应*平台*上的CppImplementationDataType级别适用特殊规定。

有关CppImplementationDataType级别的字符串数据类型建模的更多信息，请参阅第3.3.3.4节。

为了一致性起见，本章总结了ApplicationDataType的建模，用于就*AUTOSAR 自适应平台而言*表示文本字符串的数据类型的建模。

ApplicationPrimitiveDataType

SwTextProps

arraySizeSemantics: ArraySizeSemanticsEnum

[0..1]

+

swFillCharacter: Integer

[0..1]

+

«atpVariation»

[0..1]

swMaxTextSize: Integer

+

«atpVariation»

SwDataDefProps

*AtpBlueprint*

*AtpBlueprintable*

*ApplicationDataType*

*ARElement*

*AtpType*

*AutosarDataType*

*AtpBlueprint*

*AtpBlueprintable*

*BaseType*

SwBaseType

*ARElement*

SwRecordLayout

*CompositeRuleBasedValueArgument*

ApplicationValueSpecification

+

category: Identifier

[0..1]

*ValueSpecification*

shortLabel: Identifier

+

[0..1]

+

baseType

0..1

swDataDefProps

+

0..1

swRecordLayout

+

0..1

swTextProps

+

0..1

invalidValue

+

0..1

**图 3.1：文本字符串的规范**

图3.1总结了用于定义categorySTRING的ApplicationPrimitiveDataType的元类。

请注意，由于使用了比普通 C 更丰富的数据类型的编程语言， *AUTOSAR 自适应平台上*categorySTRING的ApplicationPrimitiveDataType的实现是为给定的*语言绑定预定义的*。

**[TPS\_MANI\_01047]** { DRAFT } **STRING 类别的ApplicationPrimitiveDataType的SwRecordLayout的存在**d对于*AUTOSAR 自适应平台*上类别 STRING的ApplicationPrimitiveDataType的使用， ApplicationPrimitiveDataType的存在。 swDataDefProps 。 swRecordLayout应被忽略。 c *( RS\_MANI\_00016 )*

请注意 [ TPS\_MANI\_01047 ] 故意不禁止存在

SwRecordLayout因为categorySTRING的相同ApplicationPrimitiveDataType可以正确地用于AUTOSAR**自适应***平台*和*AUTOSAR 经典平台*。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **ApplicationPrimitiveDataType** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes | | | |
| ***笔记*** | 原始数据类型定义了一组允许值。  **标签：** atp.recommendedPackage=ApplicationDataTypes | | | |
| ***根据*** | *ARElement* , *ARObject* , *ApplicationDataType* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* ,  *AutosarDataType* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.4：ApplicationPrimitiveDataType**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **SwTextProps** | | | |
| ***包裹*** | M2::MSR::DataDictionary::DataDefProperties | | | |
| ***笔记*** | 此元类表示适用于变量或校准参数中的字符串的特定属性。 | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| arraySize 语义 | 数组大小语义  枚举 | 0..1 | 属性 | 此属性控制表示实现数据类型中字符串的数组的数组大小的语义。  它支持ApplicationDatatype 和ImplementationDatatype 之间的安全转换，即使对于可变长度字符串（例如支持SAE J1939）也是如此。 |
| 基本类型 | SwBaseType | 0..1 | 参考 | 这是字符串中一个字符的基本类型。特别地，此 baseType 表示字符串中字符在 ApplicationData 类型级别上的预期编码。  **标签：** xml.sequenceOffset=30 |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **SwTextProps** |  |  |  |
| swFillCharacter | 整数 | 0..1 | 属性 | 用于填充最大长度 swMaxTextSize 的文本参数的填充字符。  该值将根据数据对象的相关基本类型中指定的编码进行解释，例如，0x30（十六进制）表示 ASCII 字符零作为填充字符，0（十进制）表示字符串的结尾作为填充字符。  填充字符的使用取决于 arraySize 语义。  **标签：** xml.sequenceOffset=40 |
| swMaxTextSize | 整数 | 0..1 | 属性 | 以字符为单位指定最大文本大小。请注意，字节大小取决于相应 baseType 中的编码。  **刻板印象：** atpVariation**标签：**  vh.latestBindingTime=preCompileTime xml.sequenceOffset=20 |

**表 3.5: SwTextProps**

|  |  |
| --- | --- |
| ***枚举*** | **ArraySizeSemanticsEnum** |
| ***包裹*** | M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes |
| ***笔记*** | 此类型控制如何解释有关 ApplicationArrayDataType 中元素数量的信息。 |
| ***文字*** | ***描述*** |
| 固定尺寸 | 这意味着 ApplicationArrayDataType 将始终具有固定数量的元素。  **标签：** atp.EnumerationLiteralIndex=0 |
| 可变尺寸 | 这意味着 ApplicationArrayDataType 中元素的实际数量可能会在运行时发生变化。 arraySize 的值表示数组中元素的最大数量。  **标签：** atp.EnumerationLiteralIndex=1 |

**表 3.6：ArraySizeSemanticsEnum**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **SwBaseType** | | | |
| ***包裹*** | M2::MSR::AsamHdo::BaseTypes | | | |
| ***笔记*** | 该元类表示 ECU 软件中使用的基本类型。  **标签：** atp.recommendedPackage=BaseTypes | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *BaseType* , *CollectableElement* , *Identifiable* ,  *MultilanguageReferable* , *PackageableElement* ,*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.7：SwBaseType**

**3.3.2.2 关联地图数据类型**

**[TPS\_MANI\_01027]** { DRAFT } **ApplicationAssocMapDataType** d的**语义**ApplicationAssocMapDataType表示关联数据结构，

即一种数据结构，其中所谓的键*s* （形式化为ApplicationAssocMapDataType .key依次由ApplicationDataType键入）与*值*s 相关联（形式化为ApplicationAssocMapDataType . value也依次由ApplicationDataType键入）。 c *( RS\_MANI\_00016 )*

[ **constr\_3349** *]* { DRAFT } **ApplicationAssocMapDataType**的**使用受限**\_ \_ \_ \_ \_ \_ \_ \_ . c *()*

[ constr\_3349 ] 是表示ApplicationAssocMapDataType只能在*AUTOSAR 自适应平台上使用的正式方法*。

**[TPS\_MANI\_01016]** { DRAFT } **ApplicationAssocMapDataType的类别**d值ApplicationAssocMapDataType 。 category应设置为属性的ASSOCIATIVE\_MAP 。 c *( RS\_MANI\_00016 )*

图3.2描述了一个ApplicationAssocMapDataType的结构示例。

ApplicationAssocMapDataType

Key

1

Value

1

Key

2

Value

2

Key

3

Value

3

Key

4

Value

4

Key

n

Value

n

......

......

***AUTOSAR 自适应平台*上的示例ApplicationAssocMapDataType**

3.2可以推断，应用程序的概念-

类别的DataType不应与ApplicationAssocMapDataType混淆[[2]](#footnote-2)。

在运行时使用关联数据结构有许多技术含义，例如每个*键的内容*在整个数据结构的上下文中应该是唯一的。

另一方面，如果 value 侧的内容包含重复项，则完全没有问题，例如两个唯一的key与具有完全相同内容的value关联。

但是，这些方面对ApplicationAssocMapDataType的正式模型没有影响，因此在本文档中不予考虑。

ApplicationAssocMapDataType的建模有点简约，主要是因为需要定义键和值的数据类型。

没有假设关联映射的实现结构可能是什么样子。例如，在 C++（目前是*AUTOSAR 自适应平台上唯一支持的语言绑定*）中，使用关联映射的直接方法是利用容器 ara::core::Map（其中实现对客户端程序员是不透明的）。

*AtpBlueprint*

*AtpBlueprintable*

*ApplicationDataType*

*ARElement*

*AtpType*

*AutosarDataType*

*ApplicationCompositeDataType*

ApplicationPrimitiveDataType

«atpVariation»

SwDataDefProps

ApplicationRecordDataType

ApplicationArrayDataType

dynamicArraySizeProfile: String

[0..1]

+

ApplicationAssocMapDataType

*DataPrototype*

*ApplicationCompositeElementDataPrototype*

ApplicationAssocMapElement

+

value

1

+

swDataDefProps

0..1

+

key

1

«isOfType»

+

type

0..1

{

redefines atpType

}

**图 3.3： ApplicationAssocMapDataType的正式模型**

图3.4包含ApplicationAssocMapDataType示例模型的图形表示。

MyMap: StdCppImplementationDataType

category = ASSOCIATIVE\_MAP

uint16\_t:

StdCppImplementationDataType

category = VALUE

MyAssociativeMap: ApplicationAssocMapDataType

category = ASSOCIATIVE\_MAP

key:

ApplicationAssocMapElement

value:

ApplicationAssocMapElement

myDataTypeMap

keyType:

ApplicationPrimitiveDataType

category = VALUE

valueType:

ApplicationPrimitiveDataType

category = VALUE

uint8\_t:

StdCppImplementationDataType

category = VALUE

valueDataTypeMap: DataTypeMap

keyDataTypeMap: DataTypeMap

CppTemplateArgument

:

CppTemplateArgument

:

+

implementationDataType

+

value

applicationDataType

+

+

templateType

applicationDataType

+

+

templateArgument

applicationDataType

+

+

implementationDataType

type

+

+

type

+

implementationDataType

key

+

templateArgument

+

+

templateType

**图 3.4：关联地图模型示例**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **ApplicationAssocMapDataType** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType | | | |
| ***笔记*** | 一种应用程序数据类型，它是一个映射，由一个键和一个值组成  **标签：**  atp.Status=草稿  atp.recommendedPackage=应用程序数据类型 | | | |
| ***根据*** | *ARElement* , *ARObject* , *ApplicationCompositeDataType* , *ApplicationDataType* , *AtpBlueprint* , *Atp*  *Blueprintable* , *AtpClassifier* , *AtpType* , *AutosarDataType* , *CollectableElement* , *Identifiable* , *Multilanguage Referrable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 钥匙 | 应用程序关联图  元素 | 1 | 聚合 | 映射的关键元素，用于唯一标识映射的值。  **标签：** atp.Status=draft |
| 价值 | 应用程序关联图  元素 | 1 | 聚合 | 存储与键关联的内容的映射的值元素。  **标签：** atp.Status=draft |

**表 3.8：ApplicationAssocMapDataType**

|  |  |
| --- | --- |
| ***班级*** | **ApplicationAssocMapElement** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType |
| ***笔记*** | 描述应用程序映射数据类型的元素的属性。  **标签：** atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **ApplicationAssocMapElement** | | | |
| ***根据*** | *ARObject* , *ApplicationCompositeElementDataPrototype* , *AtpFeature* , *AtpPrototype* , *DataPrototype* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.9：ApplicationAssocMapElement**

清单3.1提供了图3.4中描述的ApplicationAssocMapDataType示例模型的相应 ARXML 序列化。

**ApplicationAssocMapDataType的定义示例**

**<应用程序-关联-地图-数据-类型>**

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

**<键>**

**<SHORT-NAME>我的**密钥**</SHORT-NAME>**

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

**</KEY>**

**<值>**

**<SHORT-NAME>我的**值**</SHORT-NAME>**

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

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

**<应用程序原始数据类型>**

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

**<CATEGORY>**值**</CATEGORY>**

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

**<应用程序原始数据类型>**

**<SHORT-NAME>值类型</SHORT-NAME** >

**<CATEGORY>**值**</CATEGORY>**

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

但是，需要澄清ApplicationAssocMapDataType的初始化，因为它（使用RecordValueSpecification和ArrayValueSpecification的组合）通常在技术上可以定义许多在语义上相同的不同结构的ValueSpecification 。

为了将这种不确定性元素排除在 AUTOSAR 标准之外，由ApplicationAssocMapDataType类型化的 DataPrototype 的初始化通过 [ constr\_1488 ]进行了说明。

**[constr\_1488]** { DRAFT }**由ApplicationAssocMapDataType类型化的DataPrototype的初始化由**ApplicationAssocMapDataType类型化的DataPrototype只能由ApplicationAssocMapValueSpecification初始化。 c *()*

*ValueSpecification*

shortLabel: Identifier

+

[0..1]

*ARElement*

ConstantSpecification

*CompositeValueSpecification*

ApplicationAssocMapValueSpecification

ApplicationAssocMapElementValueSpecification

+

key

1

+

value

1

+

mapElementTuple

0..\* {

ordered

}

+

valueSpec

0..1

**ApplicationAssocMapDataType初始化的正式模型**

如前所述，有一个语义要求，即关联映射的*关键元素*需要在一个*关联映射*容器的上下文中是唯一的。

显然，模型对运行时发生的事情没有影响。另一方面，对ApplicationAssocMapDataType的初始化有影响，请参见 [ constr\_1489 ]。

**[constr\_1489]** { DRAFT } **ApplicationAssocMapValueSpecification的唯一性。地图元素元组。 key** d所有mapElementTuple的值。给定ApplicationAssocMapValueSpecification上下文中的关键元素应是唯一的。 c *()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **ApplicationAssocMapValueSpecification** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType | | | |
| ***笔记*** | 此元类表示定义 ApplicationAssocMapDataType 初始化的能力。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *CompositeValueSpecification* , *ValueSpecification* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 地图元素  元组（有序） | 应用程序关联图  元素值规范 | \* | 聚合 | 此聚合表示 ApplicationAssocMapValueSpecification 元素的初始值。  **标签：** atp.Status=draft |

**表 3.10：ApplicationAssocMapValueSpecification**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **ApplicationAssocMapElementValueSpecification** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType | | | |
| ***笔记*** | 此元类表示定义 ApplicationAssoc MapDataType 元素的初始化的能力。  **标签：** atp.Status=draft | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 钥匙 | 价值规范 | 1 | 聚合 | 此聚合表示 AssociativeElementValueSpecification 的关键部分的初始化。  **标签：** atp.Status=draft |
| 价值 | 价值规范 | 1 | 聚合 | 此聚合表示 AssociativeElementValueSpecification 的值部分的初始化。  **标签：** atp.Status=draft |

**表 3.11：ApplicationAssocMapElementValueSpecification**

**3.3.2.3 SwDataDefProps 的属性**

**[constr\_1478]** { DRAFT } **SwDataDefProps适用于*AUTOSAR 自适应平台*独有的ApplicationDataType** d表3.12显示了给定类别允许的SwDataDefProps和其他属性及其多重性的完整列表。 c *()*

[ constr\_1478 ] 的结果是表3.12仅显示了仅限于*AUTOSAR 自适应平台的*类别值。对于*AUTOSAR 经典平台*上也支持的所有其他类别值，请参阅软件组件模板 [ 1 ] 规范中包含的类似表格。

|  |  |  |  |
| --- | --- | --- | --- |
| **SwDataDefProps 的属性** | **根元素。** | | **每个类别的属性存在** |
|  | **ApplicationAssocMapDataType** | **ApplicationAssocMapElement** | **ASSOCIATIVE\_MAP** |
| **附加本机类型限定符** |  |  |  |
| **注解** | X | X | \* |
| **基本类型** |  |  |  |
| **计算方法** |  |  |  |
| **数据结构** |  |  |  |

5

4

|  |  |  |  |
| --- | --- | --- | --- |
| **显示格式** | X | X | 0..1 |
| **实现数据类型** |  |  |  |
| **无效值** |  |  |  |
| **步长** |  |  |  |
| **swAddr 方法** |  |  |  |
| **swAlignment** |  |  |  |
| **swBitRepresentation** |  |  |  |
| **swCalibrationAccess** |  |  |  |
| **swCalprmAxisSet** |  |  |  |
| **sw比较变量** |  |  |  |
| **swDataDependency** |  |  |  |
| **swHost 变量** |  |  |  |
| **swImplPolicy** |  |  |  |
| **swIntendedResolution** |  |  |  |
| **sw插值法** |  |  |  |
| **swIsVirtual** |  |  |  |
| **swPointerTargetProps** |  |  |  |
| **swRecordLayout** |  |  |  |
| **swRefreshTiming** |  |  |  |
| **swTextProps** |  |  |  |
| **swValueBlockSize** |  |  |  |
| **单元** |  |  |  |
| **值轴数据类型** |  |  |  |
| **根元素下的其他属性** | | | |
| **键： ApplicationAssocMapElement** | X |  | 1 |
| **值： ApplicationAssocMapElement** | X |  | 1 |

**表 3.12： ApplicationDataType的允许属性与类别**

**3.3.3 CppImplementationDataType**

**3.3.3.1 概述**

在 AUTOSAR 标准中，数据类型代表了整个开发方法中最重要的资产。

因此，AUTOSAR 实现[[3]](#footnote-3) 用于数据类型建模的多层次方法。所描述的级别之一，即所谓的*实现数据级别旨在在AUTOSAR 自适应平台*的客厅中可以描述为“语言绑定”的级别上进行建模。

对于*AUTOSAR 经典平台*，*实现数据级别*已通过创建实现数据类型来解决，该数据类型专门旨在涵盖 C 编程语言的数据类型行为。

与*AUTOSAR 经典平台相比*， *AUTOSAR 自适应平台*目前没有预见到 C 语言的使用，而是（至少在可预见的未来）定义了与 C++ 语言的语言绑定。

因此，有必要在*实现数据级别上提供一种建模方法，*并适当支持 C++ 语言的功能。

ImplementationDataType的语义以支持 C++ 语言绑定在技术上是可行的，但这将大大降低ImplementationDataType的清晰度和表现力[[4]](#footnote-4)。

因此，添加一个元类系统似乎是合理的，该系统专门支持使用预期绑定到 C++ 语言的数据类型。

**[TPS\_MANI\_01166]** { DRAFT } **CppImplementationDataType的语义**d抽象元类CppImplementationDataType支持专门为支持 C++ 语言绑定而定制的数据类型的建模。 c *( RS\_MANI\_00039 )*

**[TPS\_MANI\_03197]** { DRAFT } **StdCppImplementationDataType的语义**d元类StdCppImplementationDataType支持将映射到 C++ 语言绑定中的 C++ 标准库功能的数据类型的建模。 c *( RS\_MANI\_00039 )*

请注意结构 ( category = STRUCTURE ) 和类型别名 ( category

= TYPE\_REFERENCE ) 也被建模为StdCppImplementationDataType为简化原因。

**[TPS\_MANI\_03198]** { DRAFT } **CustomCppImplementationDataType的语义**d元类CustomCppImplementationDataType支持数据类型的建模，这些数据类型将映射到在headerFile中声明的 C++ 语言绑定中的自定义实现。 c *( RS\_MANI\_00039 )*

请注意，CustomCppImplementationDataType 的类别值受[ constr\_1578 ] 限制。

*AbstractImplementationDataType*

*CppImplementationDataTypeContextTarget*

*CppImplementationDataType*

+

headerFile: String

[0..1]

+

typeEmitter: NameToken

[0..1]

«atpVariation»

+

arraySize: PositiveInteger

[0..1]

CustomCppImplementationDataType

StdCppImplementationDataType

**图 3.6： CppImplementationDataType的特化**

这意味着只能使用StdCppImplementationDataType对原始数据类型和字符串进行建模。原因是AUTOSAR中为SOME/IP和DDS定义的序列化规则是基于标准库定义的类型。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***CppImplementationDataType*** （摘要） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType | | | |
| ***笔记*** | 此元类表示指定可重用数据类型定义的方式，该定义作为 C++ 语言绑定的基础**标签：** atp.Status=draft | | | |
| ***根据*** | *ARElement* , *ARObject* , *AbstractImplementationDataType* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* ,  *AtpType* ， *AutosarDataType* ， *CollectableElement* ， *CppImplementationDataTypeContextTarget* ，  *识别*，*多语言可引用*，可打包元素，*可引用* | | | |
| ***子类*** | CustomCppImplementationDataType , StdCppImplementationDataType | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 数组大小 | 正整数 | 0..1 | 属性 | 如果封闭的 CppImplementationDataType 具有数组语义，则此属性可用于指定数组大小。  **刻板印象：** atpVariation**标签：**  atp.Status=草稿  vh.latestBindingTime=preCompileTime |
| 头文件 | 细绳 | 0..1 | 属性 | 使用自定义类声明配置头文件。  **标签：** atp.Status=draft |
| 命名空间  （订购） | 符号道具 | \* | 聚合 | 此聚合允许为封闭的 CppImplementationData 类型定义自己的命名空间。  **标签：** atp.Status=draft |
| 子元素（有序） | Cpp实现  数据类型元素 | \* | 聚合 | 这表示封闭 CppImplementationDataType 的子元素的集合  **标签：** atp.Status=draft |
| 模板参数  （订购） | CppTemplateArgument | \* | 聚合 | 此聚合允许指定模板参数的属性**标签：** atp.Status=draft |
| 类型发射器 | 名称令牌 | 0..1 | 属性 | 此属性可用于控制相应的 CppImplementationDataType 对语言绑定的贡献方式。  **标签：** atp.Status=draft |
| 类型参考 | Cpp实现  数据类型 | 0..1 | 参考 | 应定义此引用以定义类型引用（又名 typedef）。  **标签：** atp.Status=draft |

**表 3.13：CppImplementationDataType**

|  |  |
| --- | --- |
| ***班级*** | **StdCppImplementationDataType** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType |
| ***笔记*** | 此元类表示指定数据类型定义的方法，该数据类型定义作为 C++ 语言绑定到 C++ 标准库功能的基础。  **标签：**  atp.Status=草稿  atp.recommendedPackage=CppImplementationDataTypes |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **StdCppImplementationDataType** | | | |
| ***根据*** | *ARElement* , *ARObject* , *AbstractImplementationDataType* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* ,  *AtpType* ， *AutosarDataType* ， *CollectableElement* ， *CppImplementationDataType* ， *CppImplementationData*  *TypeContextTarget* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.14：StdCppImplementationDataType**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **CustomCppImplementationDataType** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType | | | |
| ***笔记*** | 此元类表示指定数据类型定义的方式，该数据类型定义作为 C++ 语言绑定到已配置头文件中声明的自定义实现的基础。此 CustomCppImplementationDataType 的短名称定义了自定义实现的类名。  **标签：**  atp.Status=草稿  atp.recommendedPackage=CppImplementationDataTypes | | | |
| ***根据*** | *ARElement* , *ARObject* , *AbstractImplementationDataType* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* ,  *AtpType* ， *AutosarDataType* ， *CollectableElement* ， *CppImplementationDataType* ， *CppImplementationData*  *TypeContextTarget* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.15：CustomCppImplementationDataType**

**[constr\_1571]** { DRAFT } **CppImplementationDataType受限**d CppImplementationDataType的使用仅限于在Executable的上下文中定义的AdaptiveApplicationSwComponentType和CompositionSwComponentType的上下文。 c *()*

**[TPS\_MANI\_01167]** { DRAFT } **AbstractImplementationDataType** d元类CppImplementationDataType从抽象基类AbstractImplementationDataType继承，以便成为其他元类的特定引用的有效目标，这些元类想要引用“ ImplementationDataType in general”。 c *( RS\_MANI\_00039 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***AbstractImplementationDataType*** （抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes | | | |
| ***笔记*** | 此元类代表不同风格的 ImplementationDataType 的抽象基类。 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *AutosarDataType* ,  *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***子类*** | *CppImplementationDataType* , ImplementationDataType | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.16：AbstractImplementationDataType**

DataTypeMap中可以找到引用“ ImplementationDataType in general”的一个突出示例。 DataTypeMap存在的目的是将ApplicationDataType映射到ImplementationDataType或CppImplementationDataType 。

通过对参考DataTypeMap建模。 implementationDataType作为对AbstractImplementationDataType的引用，这两个选项都可能在一个角色中。

与 C 语言相比，C++ 支持在程序中定义名称空间。此功能也可在*AUTOSAR 自适应平台上进行开发*，因此需要在建模方法中进行表示。

**[TPS\_MANI\_01168]** { DRAFT } CppImplementationDataType**命名空间的规范**d为**CppImplementationDataType**定义命名空间的能力通过角色命名空间中CppImplementationDataType处的SymbolProps聚合来表达。 c *( RS\_MANI\_00039 )*

**[constr\_3443]** { DRAFT } StdCppImplementationDataType**的命名空间规范**d不允许为类别 VALUE的**StdCppImplementationDataType**定义命名空间。对于category的这个值，std 命名空间已经通过使用StdCppImplementation-

数据类型。 c *()*

*AbstractImplementationDataType*

*CppImplementationDataTypeContextTarget*

*CppImplementationDataType*

+

headerFile: String

[0..1]

typeEmitter: NameToken

[0..1]

+

«atpVariation»

+

arraySize: PositiveInteger

[0..1]

*AbstractImplementationDataTypeElement*

*CppImplementationDataTypeContextTarget*

CppImplementationDataTypeElement

[0..1]

isOptional: Boolean

+

SymbolProps

*Referrable*

*ImplementationProps*

+

[0..1]

symbol: CIdentifier

*ARElement*

Allocator

[0..1]

headerFile: String

+

CppTemplateArgument

+

[0..1]

category: CategoryString

inplace: Boolean

+

[0..1]

CppImplementationDataTypeElementQualifier

[0..1]

inplace: Boolean

+

allocator

+

0..1

+

typeReference

0..1

0..1

+

typeReference

+

templateType

0..1

+

templateArgument

0..\* {

ordered

}

namespace

+

0..\* {

}

ordered

subElement

+

0..\*

ordered

}

{

typeReference

+

1

+

namespace

}

ordered

0..\* {

**图 3.7：CppImplementationDataType 概览**

**[TPS\_MANI\_01309]** { DRAFT }**属性 CppImplementation的语义 -**

**数据类型。 headerFile** d属性CppImplementationDataType 。 headerFile用于指定对应头文件的名称有两种情况：

* CustomCppImplementationDataType应将属性的值设置为定义CustomCppImplementationDataType的 C++ 代码的头文件的名称。
* 平台数据类型（建模为StdCppImplementationDataType ）应将属性设置为来自 C++ 标准库的适用头文件的名称（例如“cstdint”）。

c *( RS\_MANI\_00016 )*

**[constr\_1743]** {草案} **CppImplementationDataType 。 headerFile与CppImplementationDataType 。 typeEmitter** d CppImplementationDataType的两个属性。 headerFile和CppImplementationDataType 。 typeEmitter应始终互斥使用。

换句话说， CppImplementationDataType的子类应使用headerFile或typeEmitter 。不支持同时使用这两个属性。 c *()*

**[TPS\_MANI\_01176]** { DRAFT }属性**CppImplementationDataType的标准化值。 typeEmitter** d AUTOSAR 标准为属性CppImplementationDataType保留以下值。类型发射器：

* TYPE\_EMITTER\_ARA
* FUNDAMENTAL\_TYPE：此值仅适用于平台类型 bool、float 和 double。

c *( RS\_MANI\_00039 )*

**[TPS\_MANI\_01177]** { DRAFT }**属性CppImplementationDataType的语义。 typeEmitter** d以下规则集适用于属性CppImplementationDataType的使用。类型发射器：

* 如果属性typeEmitter设置为值TYPE\_EMITTER\_ARA ，ARA 生成器应生成相应的数据类型定义。
* 如果属性typeEmitter设置为 TYPE\_EMITTER\_ARA 以外的任何值， ARA 生成器将**不**生成相应的数据类型定义。

c *( RS\_MANI\_00039 )*

在 [ TPS\_MANI\_01177 ] 的上下文中， [ TPS\_MANI\_01309 ] 和 [ constr\_1743 ] 适用。

**[TPS\_MANI\_01212]** { DRAFT }**在上下文中使用属性typeEmitter**

**CustomCppImplementationDataType** d属性typeEmitter不必在CustomCppImplementationDataType的上下文中使用。如果不考虑使用typeEmitter ，则属性的值应设置为包含相应CustomCppImplementationDataType的语言绑定的头文件的名称。 c *( RS\_MANI\_00039 )*

**[TPS\_MANI\_01169]** { DRAFT }**支持模板数据类型**d元类CppImplementationDataType通过引用CppImplementationDataType支持在 C++ 程序中使用模板来定义数据类型。模板参数。

模板中参数的顺序很重要，因此templateArgument被建模为**有序**集合。 c *( RS\_MANI\_00039 )*

**[TPS\_MANI\_01174]** { DRAFT }**角色CppTemplateArgument中的引用语义。 templateType** d属性CppTemplateArgument 。 templateType指定要填写在语言绑定中模板相应位置的数据类型。 c *( RS\_MANI\_00039 )*

**[TPS\_MANI\_01175]** { DRAFT }**角色CppTemplateArgument中的引用语义。分配器**d属性CppTemplateArgument 。 allocator指定要填充到语言绑定中模板的相应位置的分配器类的行为。 c *( RS\_MANI\_00039 )*

**[constr\_1576]** { DRAFT } **CppTemplateArgument的存在。 templateType与CppTemplateArgument 。 allocator** d对于任何给定的CppTemplateArgument ，**最多**引用一个

* CppTemplateArgument 。模板类型或
* CppTemplateArgument 。分配器

**可能**存在。 c *()*

**[TPS\_MANI\_01201]** { DRAFT }属性**CppTemplateArgument的标准化值。类别**d AUTOSAR 为属性CppTemplateArgument保留以下值。类别：

**ASSOC\_MAP\_KEY** ：特定的CppTemplateArgument表示关联映射的*键数据类型。*

**ASSOC\_MAP\_VALUE** ：特定的CppTemplateArgument表示关联映射的*值*数据类型。

c *( RS\_MANI\_00039 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **CppTemplateArgument** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType | | | |
| ***笔记*** | 这个元类能够定义模板参数的属性。  **标签：** atp.Status=draft | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **CppTemplateArgument** |  |  |  |
| 分配器 | 分配器 | 0..1 | 参考 | 该参考标识了适用的分配器。  **标签：** atp.Status=draft |
| 类别 | 类别字符串 | 0..1 | 属性 | 该属性将用于进一步阐明封闭 Cpp TemplateArgument 的语义。  **标签：** atp.Status=draft |
| 到位 | 布尔值 | 0..1 | 属性 | 此属性指定是否在代码生成中使用所引用模板类型的 shortName 并且在封闭 CppImplementationDataType 之外定义类型声明（true），或者是否将类型定义嵌入到封闭 CppImplementationDataType 内部并且忽略 shortName（false） .  **标签：** atp.Status=draft |
| 模板类型 | Cpp实现  数据类型 | 0..1 | 参考 | 此参考标识语言绑定所需的特定模板参数的数据类型。  **标签：** atp.Status=draft |

**表 3.17：CppTemplateArgument**

**[TPS\_MANI\_01171]** { DRAFT }**结构化数据类型建模**d元类CppImplementationDataType支持通过在角色subElement中聚合CppImplementationDataTypeElement来创建嵌套数据类型。

因为结构化数据类型中子元素的顺序很重要，所以聚合子元素被建模为**有序**集合。 c *( RS\_MANI\_00039 )*

请注意，虽然结构的建模是通过使用

CppImplementationDataType实际上只能为此目的使用StdCppImplementationDataType （参见 [ constr\_1578 ]）。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **CppImplementationDataTypeElement** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType | | | |
| ***笔记*** | 声明一个本地聚合的数据对象。这样的元素只能在它被聚合的范围内使用。 CppImplementationDataTypeElement 用于表示结构的元素，定义其类型。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *AbstractImplementationDataTypeElement* , *AtpClassifier* , *AtpFeature* , *AtpStructureElement* , *CppImplementationDataTypeContextTarget* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **CppImplementationDataTypeElement** | | |  |
| 是可选的 | 布尔值 | 0..1 | 属性 | 此属性表示将封闭的 CppImplementationDataTypeElement 声明为可选的能力。这意味着，在运行时，Cpp ImplementationDataTypeElement 可能有也可能没有有效值，因此应被忽略。  底层运行时软件提供了将 CppImplementationDataTypeElement 在通信的发送端设置为无效并在接收端确定其有效性的方法。  **标签：** atp.Status=draft |
| 类型参考 | Cpp实现  数据类型元素  预选赛 | 0..1 | 聚合 | 此聚合定义 Cpp ImplementationDataTypeElement 的类型并确定在 C++ 中 CppImplementationDataTypeElement  在封闭的 Cpp ImplementationDataType 内部或外部定义。  **标签：** atp.Status=draft |

**表 3.18：CppImplementationDataTypeElement**

请注意，无意支持结构化数据类型的“混合”建模，以便 C++ 级别的结果数据类型将由 C++ 本地的数据类型和来自 C 子系统的数据类型组成。

虽然这在技术上在代码级别上是可行的，但它将在建模级别上付出巨大的努力，并且共识是这种“混合”数据类型没有真正的用例。

*AUTOSAR 自适应平台*的实现而言，C++ 数据类型系统可以完全替代 C++ 中的“遗留”C 数据类型。

**[constr\_1572]** { DRAFT } **SwDataDefProps的用法。 CppImplementationDataType内的implementationDataType** d在CppImplementationDataType范围内，引用CppImplementationDataType 。 swDataDefProps 。 implementationDataType**不应存在**。 c *()*

这方面也由 [ constr\_1579 ] 以更一般的形式表示。

由于 [ constr\_1572 ]，类型引用必须在*AUTOSAR 自适应平台上以不同方式完成*。为此，提供了专门的参考资料。

**[TPS\_MANI\_01172]** { DRAFT }**范围内类型引用的描述**

**CppImplementationDataType** d参考CppImplementationDataType 。 typeReference可用于创建从封闭CppImplementationDataType到另一个CppImplementationDataType的类型引用。 c *( RS\_MANI\_00039 )*

**[TPS\_MANI\_01173]** { DRAFT }**范围内类型引用的描述**

**CppImplementationDataTypeElement** d CppImplementationDataTypeElement 。 typeReference可用于创建对CppImplementationDataType的引用，该引用应适用于封闭的CppImplementationDataType-

元素。 c *( RS\_MANI\_00039 )*

请注意CppImplementationDataTypeElement 。 typeReference被实现为一个关联类，它允许将就地属性添加到typeReference 。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **CppImplementationDataTypeElementQualifier** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType | | | |
| ***笔记*** | 此元素将 CppImplementationDataTypeElement 的 typeReference 限定为 Cpp ImplementationDataType。  **标签：** atp.Status=draft | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 到位 | 布尔值 | 0..1 | 属性 | 此属性定义 C++ 中 CppImplementationDataTypeElement 的成员类型是否是封闭结构内部的嵌入类型元素 (true)，或者类型声明是否定义在结构外部。  **标签：** atp.Status=draft |
| 类型参考 | Cpp实现  数据类型 | 1 | 参考 | 这个引用定义了一个类型引用。  **标签：** atp.Status=draft |

**表 3.19：CppImplementationDataTypeElementQualifier**

**[TPS\_MANI\_03196]** { DRAFT } **CppImplementationDataTypeElementQualifier的语义。就地属性**d CppImplementationDataTypeElementQualifier 。 \_ inplace属性定义C++ 语言绑定中CppImplementationDataTypeElement的数据类型是从引用的CppImplementationDataType的名称还是属性派生的。

具体应适用以下规则：

* 如果CppImplementationDataTypeElement 。类型参考。 inplace设置为 False，然后是角色CppImplementationDataTypeElement中引用的CppImplementationDataType的**shortName** 。类型参考。 typeReference应在 C++ 语言绑定中使用。
* 如果CppImplementationDataTypeElement 。类型参考。 inplace设置为 True 然后只有角色CppImplementationDataTypeElement中引用的CppImplementationDataType的**属性**。类型参考。 typeReference应在 C++ 语言绑定中使用，并且忽略shortName 。

c *( RS\_MANI\_00039 )*

请注意，图 3.13显示了一个结构的示例，其中一个子元素的typeReference被分类为inplace 。

**[constr\_1659]** { DRAFT } **CppImplementation的使用限制-**

**数据类型元素限定符。就地**d属性CppImplementationDataTypeElementQualifier 。仅当角色CppImplementationDataTypeElementQualifier中引用的目标时， inplace才应存在。 typeReference是一个StdCppImplementationDataType ，其属性类别设置为任一值

* 大批
* 向量
* ASSOCIATIVE\_MAP
* 变体
* 结构体
* 细绳
* TYPE\_REFERENCE ，如果CppImplementationDataType引用类别TEXTTABLE的CompuMethod

### c ()

constr\_1659 ]存在的理由：通过应用排除原则，属性CppImplementationDataTypeElementQualifier存在三种情况。 inplace不应存在：

* StdCppImplementationDataType of categoryVALUE
* CustomCppImplementationDataType
* categoryTYPE\_REFERENCE的CppImplementationDataType ，除非CppImplementationDataType引用类别TEXTTABLE的CompuMethod

它们都不能用作CppImplementationDataTypeElementQualifier的目标。 typeReference where CppImplementationDataTypeElementQualifier 。 inplace设置为 True 是因为在这些情况下，已经存在可直接用于语言绑定的有效名称，并且通过 using 子句的可能间接调用显然需要模型中不可用的附加名称。

毕竟，定义TYPE\_REFERENCE的动机与使用属性CppImplementationDataTypeElementQualifier背后的动机正好相反。就地控制语言绑定。因此，也排除了这种情况。

**[TPS\_MANI\_03201]** { DRAFT } **CppTemplateArgument的语义。就地属性**d CppTemplateArgument 。 \_ inplace属性定义 C++ 语言绑定中的templateType所引用的数据类型是派生自所引用的CppImplementationDataType的名称还是属性。

具体应适用以下规则：

* 如果CppTemplateArgument 。 inplace设置为 False，然后是角色CppTemplateArgument中引用的CppImplementationDataType的**shortName** 。 templateType应在 C++ 语言绑定中使用。
* 如果CppTemplateArgument 。 inplace设置为 True 然后只有角色CppTemplateArgument中引用的CppImplementationDataType的**属性**。 templateType应在 C++ 语言绑定中使用，而shortName被忽略。

c *( RS\_MANI\_00039 )*

**[constr\_1660]** { DRAFT } **CppTemplateArgument的使用限制。就地**d属性CppTemplateArgument 。仅当角色CppTemplateArgument中引用的目标时， inplace才应存在。 templateType是一个StdCppImplementationDataType ，其属性类别设置为任一值

* 大批
* 向量
* ASSOCIATIVE\_MAP
* 变体
* 结构体
* 细绳

### c ()

constr\_1660 ]存在的理由：通过应用排除原则，属性CppTemplateArgument存在三种情况。 inplace不应存在：

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

它们都不能用作CppTemplateArgument的目标。模板类型，其中CppTemplateArgument 。 inplace设置为 True 是因为在这些情况下，已经存在可直接用于语言绑定的有效名称，并且通过 using 子句的可能间接调用显然需要模型中不可用的附加名称。

毕竟，定义TYPE\_REFERENCE的动机与使用属性CppTemplateArgument背后的动机正好相反。就地控制语言绑定。因此，也排除了这种情况。

请注意属性CppTemplateArgument的值的问题。就地CppTemplateArgument的情况。引用STRUCTURE 类别的StdCppImplementationDataType的templateType由 [ constr\_3462 ] 规定。

**[constr\_1708]** { DRAFT } **CppImplementationDataTypeElement的组合。 isOptional和CppImplementationDataTypeElementQualifier 。 inplace** d如果CppImplementationDataTypeElement由categorySTRUCTURE的CppImplementationDataType类型，则属性CppImplementationDataTypeElement的组合。 isOptional设置为 True 和CppImplementationDataTypeElement 。类型参考。不允许就地设置为 True。 c *()*

constr\_1708 ]存在的理由：“可选”语义是通过模板实现的，不可能将“就地”结构作为模板参数传递。

**[constr\_3462]** {草案} **CppTemplateArgument 。模板类型参考**

**categorySTRUCTURE的StdCppImplementationDataType和就地标志**d CppTemplateArgument 。指向categorySTRUCTURE的StdCppImplementationDataType的templateType应将inplace属性设置为**false** 。 c *()*

constr\_3462 ]的原因是 ISO C++11/14/17 不允许使用未命名的结构作为模板参数。

**[constr\_3446]** { DRAFT } **CppTemplateArgument with allocator reference and the inplace flag** d一个CppTemplateArgument指向一个分配器引用的分配器不应将inplace标志设置为一个值。 c *()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **分配器** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType | | | |
| ***笔记*** | 这个元类代表了影响对象在内存中分配方式的能力，例如，它可以控制一个对象是在堆上分配还是在堆栈上分配。  **标签：**  atp.Status=草稿  atp.recommendedPackage=分配器 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 头文件 | 细绳 | 0..1 | 属性 | 使用自定义类声明配置头文件  **标签：** atp.Status=draft |
| 命名空间  （订购） | 符号道具 | \* | 聚合 | 此聚合允许定义分配器的命名空间。  **标签：** atp.Status=draft |

**表 3.20：分配器**

**[TPS\_MANI\_01100]** { DRAFT }**分配器语义**d元类分配器具有定义内存分配属性的能力。内存分配的一般方法是通过属性类别表示的。

Allocator的以下值。类别由 AUTOSAR 标准化：

* MAX\_SIZE\_HEAP：当使用这个分配器时，打算在堆上分配一个固定大小的块。这个分配器增加了定义最大元素数量到 ara::core::Vector 的默认分配器语义的能力。
* MAX\_SIZE\_STACK：当使用这个分配器时，打算在堆栈上分配一个固定大小的块。堆栈上的内存总是需要根据最大大小来限制。换句话说，几乎没有任何情况应该在堆栈上分配无限量的内存。
* MAX\_SIZE\_DATASEGMENT：使用此分配器时，意图在数据段中分配固定大小的块。

c *( RS\_MANI\_00016 )*

**[constr\_1578]** { DRAFT }**适用数据类别**d表3.21定义了适用类别s 与元类。 c *()*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **类别** | **适用于 ...** | | | | | | | | **描述** |
|  | **ApplicationArrayDataType** | **ApplicationRecordDataType** | **ApplicationPrimitiveDataType** | **ApplicationRecordElement** | **ApplicationArrayElement** | **ApplicationValueSpecification** | **StdCppImplementationDataType** | **CustomCppImplementationDataType** |  |
| **价值** |  |  | X | X | X | X | X |  | 包含单个值。另见 [ TPS\_MANI\_03192 ]。 |
| **TYPE\_REFERENCE** |  |  |  |  |  |  | X |  | 该元素是通过引用另一种数据类型（通过CppImplementationDataType . typeReference 。 |
| **结构体** |  | X |  | X | X |  | X |  | 包含一个或多个可以具有不同AutosarDataType的其他元素。另见 [ TPS\_MANI\_03180 ]。 |
| **变体** |  |  |  |  |  |  | X | X | 可以保存不同数据类型的值。它类似于结构  除了它的所有成员都从内存中的同一位置开始。  一个VARIANT数据原型一次只能包含一个元素，并表示一个类型安全的联合。 VARIANT的大小至少是最大成员的大小。另见 [ TPS\_MANI\_03189 ]。 |
| **大批** | X |  |  | X | X |  | X | X | 相同数据类型的固定大小的子元素数组。另见 [ TPS\_MANI\_03169 ]。 |
| **向量** |  |  |  |  |  |  | X | X | 能够在运行时增长的相同数据类型的元素数组。另见 [ TPS\_MANI\_03174 ]。 |
| **协会-**  **TIVE\_MAP** |  |  |  |  |  |  | X | X | 键值对的关联数组。另见 [ TPS\_MANI\_03183 ]。 |
| **细绳** |  |  | X | X | X | X | X |  | 包含一个文本字符串。另见 [ TPS\_MANI\_03178 ]。 |
| **布尔值** |  |  | X | X | X | X |  |  | 包含一个布尔状态。取决于 CPU，单个位的直接寻址可能不可用。  所以一个字节或一个字只能用于存储一个逻辑状态。 |

**表 3.21：数据类型的类别使用**

**3.3.3.2 SwDataDefProps 的属性**

**[constr\_1579]** { DRAFT } **SwDataDefProps适用于*AUTOSAR 自适应平台*独有的CppImplementationDataType** d表3.22中显示了给定类别允许的SwDataDefProps和其他属性及其多样性的完整列表。 c *()*

[ constr\_1578 ] 的结果是表3.22仅显示了仅限于*AUTOSAR 自适应平台的*类别值。对于*AUTOSAR 经典平台*上也支持的所有其他类别值，请参阅软件组件模板 [ 1 ] 规范中包含的类似表格。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SwDataDefProps 的属性** | **根**  **电-**  **更换** | **每个类别的属性存在** | | | | | | | |
|  | **CppImplementationDataType** | **VALUE** | **TYPE\_REFERENCE** | **STRUCTURE** | **VARIANT** | **ARRAY** | **VECTOR** | **ASSOCIATIVE\_MAP** | **STRING** |
| **附加本机类型限定符** |  |  |  |  |  |  |  |  |  |
| **注解** | X | \* | \* | \* | \* | \* | \* | \* | \* |
| **基本类型** |  |  |  |  |  |  |  |  |  |
| **计算方法** | X |  | 0..1 |  |  |  |  |  |  |
| **数据构造数据结构规则。物理常数** | X |  | 直流电 |  |  | 直流电 | 直流电 |  |  |
| **数据构造数据结构规则。内部构造** | X |  | 0..1 |  |  | 0..1 | 0..1 |  |  |
| **显示格式** | X | 0..1 | 0..1 | 0..1 | 0..1 | 0..1 | 0..1 | 0..1 | 0..1 |
| **实现数据类型** |  |  |  |  |  |  |  |  |  |
| **无效值** | X |  | 0..1 |  |  |  |  |  | 0..1 |
| **步长** |  |  |  |  |  |  |  |  |  |
| **swAddr 方法** |  |  |  |  |  |  |  |  |  |
| **swAlignment** |  |  |  |  |  |  |  |  |  |
| **swBitRepresentation** |  |  |  |  |  |  |  |  |  |
| **swCalibrationAccess** |  |  |  |  |  |  |  |  |  |
| **swCalprmAxisSet** |  |  |  |  |  |  |  |  |  |
| **sw比较变量** |  |  |  |  |  |  |  |  |  |
| **swDataDependency** |  |  |  |  |  |  |  |  |  |
| **swHost 变量** |  |  |  |  |  |  |  |  |  |
| **swImplPolicy** |  |  |  |  |  |  |  |  |  |
| **swIntendedResolution** |  |  |  |  |  |  |  |  |  |
| **sw插值法** |  |  |  |  |  |  |  |  |  |
| **swIsVirtual** |  |  |  |  |  |  |  |  |  |
| **swPointerTargetProps** |  |  |  |  |  |  |  |  |  |
| **swPointerTargetProps** 。 **swDataDefProps** |  |  |  |  |  |  |  |  |  |
| **swRecordLayout** |  |  |  |  |  |  |  |  |  |

5

4

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SwDataDefProps 的属性** | **根**  **电-**  **更换** |  | **每个类别的属性存在** | | | | | |  |
|  | **CppImplementationDataType** | **VALUE** | **TYPE\_REFERENCE** | **STRUCTURE** | **VARIANT** | **ARRAY** | **VECTOR** | **ASSOCIATIVE\_MAP** | **STRING** |
| **swRefreshTiming** | X | 0..1 | 0..1 | 0..1 | 0..1 | 0..1 | 0..1 | 0..1 | 0..1 |
| **swTextProps** |  |  |  |  |  |  |  |  |  |
| **swValueBlockSize** |  |  |  |  |  |  |  |  |  |
| **单元** |  |  |  |  |  |  |  |  |  |
| **值轴数据类型** |  |  |  |  |  |  |  |  |  |
| **其他属性** | | |  | | | | | |  |
| **元素： CppImplementationDataTypeElement** | X |  |  | 1..\* |  |  |  |  |  |
| **模板参数** | X |  |  |  | 1..\* | 1 | 1..\* | 2..\* | 0..1 |
| **类型参考** | X |  | 1 |  |  |  |  |  |  |

**表 3.22： CppImplementationDataType的允许属性与类别**

invalidValue适用于原始数据类型并定义一个特定值（在该原始数据类型的范围内），该值指示相应的值无效。

一个典型的用例是包含所有 4 个轮速值的复合数据类型。如果其中一个车轮速度传感器发生故障，并且不再能够提供有用的数据，则提供其他 3 个车轮速度值仍然有意义。

在这种情况下，一个车轮速度值将被设置为invalidValue 。接收器能够检查数据组合的每个单独元素的值是否对应于invalidValue并采取相应的操作。

**[constr\_3569]** { DRAFT }**属性invalidValue对TYPE\_REFERENCE类别的 CppImplementationDataType的适用性**d如果类别 TYPE\_REFERENCE的CppImplementationDataType定义了 invalidValue，则引用的CppImplementationDataType （通过typeReference ）最终应为categoryVALUE 。 c *()*

对于特定类别的数据类型支持invalidValue存在以下基本原理：

* *AUTOSAR自适应平台上*categoryVALUE的使用归结为标准类型的使用。没有为标准数据类型定义invalidValue的用例，因为标准数据类型的**所有用法**都以相同的invalidValue为特征。
* 容器的invalidValue定义（ STRING除外），因为没有支持invalidValue的已知用例。
* STRING 类别的数据类型上定义invalidValue是被接受的，因为它在*AUTOSAR 经典平台*上也受支持，并且有必要维持*AUTOSAR 经典平台*和*AUTOSAR 自适应平台之间的互操作性*。
* categoryTYPE\_REFERENCE的StdCppImplementationDataType上定义invalidValue表示定义invalidValue的主流用例。

**3.3.3.3 原始数据类型**

**[TPS\_MANI\_03192]** { DRAFT } **CppImplementationDataType of category VALUE** d基本数据类型，如布尔、固定宽度整数数据类型和浮点数据类型，被描述为categoryVALUE的CppImplementationDataType s 。 c *( RS\_MANI\_00039 )*

**[TPS\_MANI\_03193]** { DRAFT } **CppImplementationDataType类别 TYPE\_REFERENCE** d类别 TYPE\_REFERENCE**的**CppImplementationDataType的定义为typeReference引用的另一个CppImplementationDataType创建别名。 c *( RS\_MANI\_00039 )*

**3.3.3.4 字符串数据类型**

**[TPS\_MANI\_03178]** { DRAFT }**类别的StdCppImplementationDataType**

**STRING** d类别STRING的StdCppImplementationDataType表示字符序列的容器数据类型。

AUTOSAR 要求类别 STRING的StdCppImplementationDataType的C++ 绑定由 ara::core::String 实现。 c *( RS\_MANI\_00039 )*

**[constr\_1674]** { DRAFT }支持**的categorySTRING** d的**StdCppImplementationDataType编码**在元模型（以及扩展的语言绑定）级别上，唯一支持的categorySTRING的StdCppImplementationDataType编码是UTF-8 。 c *()*

请注意，仍然可以使用不同的编码，例如在 SOME/IP 消息级别上使用UTF-16 。可以通过ApSomeipTransformationProps配置此行为。因此，可能必须在网络上的字符串表示和软件中的字符串表示之间应用代码转换。

**[TPS\_MANI\_03179]** { DRAFT } **StdCppImplementation的 C++ 语言绑定-**

**类别的DataType**字符串d类别的CppImplementationDataType

STRING应实现为 ara::core::String。 c *( RS\_MANI\_00039 )*

[ TPS\_MANI\_03179 ] 的制定为以后可能的扩展以支持其他存储格式留下了空间。

图3.8中描述的示例包含ApplicationDataType的定义以及相应CppImplementationDataType的定义。

由于限制，就 C++ 语言绑定而言，categorySTRING 的 CppImplementationDataType 只能在 ara::core::String 的基础上实现，因此后者显然变得对模型更轻。

图 3.8中示例的另一个方面是它定义了在ApplicationPrimitiveDataType范围内建模数据类型的预期编码。

**[TPS\_MANI\_03188]** { DRAFT }**对categorySTRING**的**StdCppImplementationDataType** d的**分配器的使用允许**categorySTRING的StdCppImplementationDataType聚合一个CppTemplateArgument ，该 CppTemplateArgument 引用分配器引用的分配器。 c *( RS\_MANI\_00039 )*

MyStringApplicationDataType:

ApplicationPrimitiveDataType

category = STRING

dataTypeMap: DataTypeMap

string:

StdCppImplementationDataType

category = STRING

+

implementationDataType

+

applicationDataType

**UTF-8编码的字符串模型示例**

**3.3.3.5 数组数据类型**

**[TPS\_MANI\_03169] {** DRAFT }具有**固定大小数组语义**的CppImplementationDataType d categoryARRAY的CppImplementationDataType表示封装固定大小数组的容器数据类型。 c *( RS\_MANI\_00039 )*

**[TPS\_MANI\_03170]** { DRAFT } **CppImplementationDataType of category ARRAY** d对于 C++ 绑定，categoryARRAY 的CppImplementationDataType可以实现为

* 如果StdCppImplementationDataType子类用于建模或作为
* CustomCppImplementationDataType子类，则自定义命名空间中的数组类型（例如 my::array） （前提是自定义命名空间中的类型可以使用可用的建模功能进行配置）。

c *( RS\_MANI\_00039 )*

**[TPS\_MANI\_03171]** { DRAFT }**类别 ARRAY的CppImplementationDataType的值类型**d categoryARRAY的CppImplementationDataType中包含的元素的类型由聚合的templateArgument和定义CppTemplateArgument数据类型的相应templateType定义。 c *( RS\_MANI\_00039 )*

**[constr\_3433]** { DRAFT }**为ARRAY聚合templateArgument** d CppImplementationDataType of categoryARRAY归结为 ara::core::Array 应聚合一个templateArgument ，该 templateArgument定义包含在categoryARRAY的CppImplementationDataType中的元素的类型。 c *()*

**[TPS\_MANI\_03172]** { DRAFT }**类别 ARRAY的CppImplementationDataType的大小**d categoryARRAY的CppImplementationDataType的原始属性arraySize将用于定义数组的大小。 c *( RS\_MANI\_00039 )*

图 3.9显示了一个由 uint16 元素组成的一维数组的示例，其中arraySize = 5。

DataTypeMapping: DataTypeMap

ArrayLinear:

ApplicationArrayDataType

category = ARRAY

ArrayElementLinear:

ApplicationArrayElement

category = VALUE

maxNumberOfElements = 5

arraySizeSemantics = fixedSize

ArrayLinearElementDataType:

ApplicationPrimitiveDataType

category = VALUE

ArrayLinearImpl:

StdCppImplementationDataType

arraySize = 5

category = ARRAY

uint16\_t:

StdCppImplementationDataType

category = VALUE

CppTemplateArgument

:

templateArgument

+

+

type

applicationDataType

+

implementationDataType

+

+

element

+

templateType

**图 3.9：一维数组模型示例**

**[TPS\_MANI\_03173]** { DRAFT }**多维数组的定义**d类别 ARRAY的多维CppImplementationDataType包含categoryARRAY的嵌套CppImplementationDataType s 。

表示外部数组的 categoryARRAY 的 CppImplementationDataType 将通过聚合的templateArgument引用表示内部数组的categoryARRAY的CppImplementationDataType 。这样的定义描述了一个二维数组；因此，仅通过嵌套更多CppImplementationDataType的categoryARRAY来描述具有更多维度的类型。

数组元素本身由最里面的CppImplementationDataType指定，其类别与ARRAY不同。 c *( RS\_MANI\_00039 )*

图 3.10显示了一个多维数组的示例，其中arraySize = 5的categoryARRAY的CppImplementationDataType有一个templateArgument指向角色templateType中categoryARRAY的内部CppImplementationDataType 。

内部CppImplementationDataType有一个templateArgument ，它最终用templateType引用指向原始类型。

unit16\_t: StdCppImplementationDataType

category = VALUE

ArrayLinearImpl: StdCppImplementationDataType

arraySize = 10

category = ARRAY

ArrayLinearElementDataType: ApplicationPrimitiveDataType

category = VALUE

ArrayElementRectangular: ApplicationArrayElement

category = ARRAY

maxNumberOfElements = 5

arraySizeSemantics = fixedSize

ArrayRectangular: ApplicationArrayDataType

category = ARRAY

DataTypeMapping: DataTypeMap

ArrayLinear: ApplicationArrayDataType

category = ARRAY

ArrayLinearElement: ApplicationArrayElement

category = VALUE

maxNumberOfElements = 10

arraySizeSemantics = fixedSize

ArrayRectangularImpl: StdCppImplementationDataType

arraySize = 5

category = ARRAY

CppTemplateArgument

:

inplace = True

CppTemplateArgument

:

type

+

templateArgument

+

+

templateType

+

applicationDataType

+

element

templateType

+

+

implementationDataType

+

templateArgument

+

element

type

+

**图 3.10：多维数组模型示例**

**3.3.3.6 矢量数据类型**

**[TPS\_MANI\_03174] {** DRAFT }具有**可变大小数组语义**的CppImplementationDataType d categoryVECTOR的CppImplementationDataType表示封装可变大小数组的容器数据类型。 c *( RS\_MANI\_00039 )* **[TPS\_MANI\_03175]** { DRAFT } **CppImplementationDataType of categoryVEC-**

**TOR** d对于 C++ 绑定，categoryVECTOR 的CppImplementationDataType可以实现为

* StdCppImplementationDataType子类或作为ara::core::Vector
* CustomCppImplementationDataType子类，则自定义命名空间中的向量类型（例如 my::vector） （前提是可以使用可用的建模功能配置自定义命名空间中的类型）。

c *( RS\_MANI\_00039 )*

DynamicDataArray\_Linear: ApplicationArrayDataType

dynamicArraySizeProfile = VSA\_LINEAR

category = ARRAY

DynamicDataArrayDim1: ApplicationArrayElement

arraySizeSemantics = variableSize

maxNumberOfElements = 5

arraySizeHandling = AllIndicesSameArraySize

category = VALUE

PrimitiveDataElementType:

ApplicationPrimitiveDataType

category = VALUE

DynamicDataArrayImplLinear:

StdCppImplementationDataType

category = VECTOR

DataTypeMapping: DataTypeMap

uint16\_t: StdCppImplementationDataType

category = VALUE

:

CppTemplateArgument

element

+

implementationDataType

+

templateType

+

+

applicationDataType

+

templateArgument

+

type

**图 3.11：一维向量模型示例**

**[TPS\_MANI\_03176]** { DRAFT } **VECTOR** d**类别的CppImplementationDataType的值类型包含在**categoryVECTOR的CppImplementationDataType中的元素类型由聚合的templateArgument和定义CppTemplateArgument数据类型的相应templateType定义。 c *( RS\_MANI\_00039 )*

**[constr\_3434]** { DRAFT }**聚合templateArgument用于VECTOR** d CppImplementationDataType类别 VECTOR归结为 ara::core::Vector 应聚合

* 一个templateArgument ，它使用templateType引用定义categoryVECTOR的CppImplementationDataType中包含的元素的类型。
* 可选的一个额外的templateArgument ，它使用分配器引用定义分配器。

### c ()

**[TPS\_MANI\_03186]** { DRAFT }**在 Vector** d的情况下**使用arraySize**如果categoryVECTOR的CppImplementationDataType聚合了一个templateArgument ，该 templateArgument定义了分配器引用的分配器，则允许使用定义矢量最大大小的属性arraySize 。 c *( RS\_MANI\_00039 )*

图 3.11显示了 uint16 元素的一维向量示例。

DynamicDataArrayDim2Element:

ApplicationArrayElement

arraySizeSemantics = variableSize

maxNumberOfElements = 4

arraySizeHandling = AllIndicesSameArraySize

category = VALUE

PrimitiveDataElementType:

ApplicationPrimitiveDataType

category = VALUE

DynamicDataArrayImplRectangular:

StdCppImplementationDataType

category = VECTOR

templateArgumentDim1:

StdCppImplementationDataType

category = VECTOR

DynamicDataArrayDim1: ApplicationArrayElement

arraySizeSemantics = variableSize

arraySizeHandling = allIndicesDifferentArraySize

maxNumberOfElements = 3

category = ARRAY

DynamicDataArrayDim2: ApplicationArrayDataType

category = ARRAY

dynamicArraySizeProfile = VSA\_LINEAR

DataTypeMapping: DataTypeMap

uint16\_t: StdCppImplementationDataType

category = VALUE

DynamicDataArray\_Rectangular:

ApplicationArrayDataType

category = ARRAY

dynamicArraySizeProfile = VSA\_RECTANGULAR

:

CppTemplateArgument

CppTemplateArgument

:

element

+

+

implementationDataType

+

templateType

type

+

templateArgument

+

templateType

+

+

applicationDataType

+

type

+

templateArgument

+

element

**图 3.12：多维向量模型示例**

**[TPS\_MANI\_03177]** { DRAFT }**多维向量**d的定义categoryVECTOR的多维CppImplementationDataType包含categoryVECTOR的嵌套CppImplementationDataType 。

表示外部向量的 categoryVECTOR 的 CppImplementationDataType 将通过聚合的templateArgument引用表示内部向量的categoryVECTOR的CppImplementationDataType 。

这样的定义描述了一个二维向量；因此，仅通过嵌套更多CppImplementationDataType的categoryVECTOR来描述具有更多维度的类型。

向量元素本身由最里面的CppImplementationDataType指定，其类别与VECTOR不同。 c *( RS\_MANI\_00039 )*

图 3.12显示了一个多维向量示例，其中categoryVECTOR的CppImplementationDataType有一个templateArgument指向角色templateType中类别 VECTOR的内部CppImplementationDataType 。内部CppImplementationDataType有一个templateArgument ，它最终用templateType引用指向原始类型。

categoryVECTOR的CppImplementationDataType的特定元素（通过index标识）的引用。

但是，如果特定元素在相应的时间点不存在，这可能会导致运行时出现问题。任何使用此类数据类型的软件都需要为可能不存在向量元素做好准备。

或者，可以选择简单地避免categoryVECTOR的CppImplementationDataType的元素成为模型中引用的目标的情况。

**3.3.3.7 结构数据类型**

**[TPS\_MANI\_03180]** { DRAFT }结构的**定义**d类别STRUCTURE的StdCppImplementationDataType表示用于保存任意数据类型变量的有序集合的数据类型。 c *( RS\_MANI\_00039 )*

**[TPS\_MANI\_03181]** { DRAFT } **StdCppImplementation中成员的定义-**

**StdCppImplementation中类别 STRUCTURE** d成员的数据类型-

categorySTRUCTURE的DataType由有序CppImplementationDataTypeElement定义，这些有序 CppImplementationDataTypeElement由categorySTRUCTURE的封闭StdCppImplementationDataType在角色subElement中聚合。

每个成员的名称由CppImplementationDataTypeElement的shortName定义。

每个成员的类型由CppImplementationDataType的typeReference定义。 c *( RS\_MANI\_00039 )*

请注意，能够对CppImplementationDataTypeElement进行分类的就地标志。 typeReference记录在 [ TPS\_MANI\_03196 ] 中。

图3.13中描述的示例显示了一个名为 MyStruct 的结构的定义，它有两个成员。具有shortName ArrayElement的subElement的typeReference使用inplace = **True进行分类**。

如果数组的typeReference中的inplace属性设置为**False** ，则模型会导致在 MyStruct 外部定义的 ArrayDataType 的 using 声明。

MyStruct: CppImplementationDataType

category = STRUCTURE

PrimitiveElement:

CppImplementationDataTypeElement

ArrayElement:

CppImplementationDataTypeElement

ArrayDataType:

StdCppImplementationDataType

category = ARRAY

arraySize = 5

uint8\_t: StdCppImplementationDataType

category = VALUE

:

CppTemplateArgument

CppImplementationDataTypeElementQualifier

:

inplace = True

:

CppImplementationDataTypeElementQualifier

+

typeReference

templateArgument

+

+

templateType

subElement

+

typeReference

+

subElement

+

+

templateType

typeReference

+

**图 3.13：结构模型示例**

**3.3.3.8 枚举数据类型**

**[TPS\_MANI\_03187]** { DRAFT }**枚举类型的定义**d在 AUTOSAR 元模型中，枚举不是通过具有自己的类别的CppImplementationDataType来实现的。

相反，可以将一组离散的整数用作单个基本CppImplementationDataType的categoryTYPE\_REFERENCE的结构描述，归结为类别VALUE的CppImplementationDataType 。

整数到枚举定义范围内的标签的映射被认为是语义定义的一部分，通过带有类别TEXTTABLE 的附加CompuMethod ，而不是结构描述的一部分。 c *( RS\_MANI\_00039 )*

使用类别为 TEXTTABLE 的CompuMethod的规则与 AUTOSAR 经典平台中的规则相同，并在软件组件模板 [ 1 ] 中进行了描述。

总而言之，可以在 CompuConst 的vt中，在 CompuMethod 的适用 CompuScale 的shortLabel或符号中提供CompuMethod中具有类别TEXTTABLE的枚举值作为文本值。

每个CompuScale应在 CompuMethod 中定义为compuInternalToPhys计算，并应包含一个upperLimit和lowerLimit 。

以下示例说明了如何使用CompuMethod指定枚举。

**清单 3.2：枚举示例**

**<计算方法>**

**<SHORT-NAME>**气缸**</SHORT-NAME>**

**<CATEGORY>**可文本化**</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>**气缸 1 **</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>**气缸 2 **</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>**气缸 3 **</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>**气缸 4 **</VT>**

**</COMPU-CONST>**

**</COMPU-SCALE>**

**</COMPU-SCALES>**

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

**</计算方法>**

**3.3.3.9 地图数据类型**

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

**SOCIATIVE\_MAP** d类别 ASSOCIATIVE\_MAP的CppImplementationDataType表示包含具有唯一键的键值对的容器。 c *( RS\_MANI\_00039 )*

**[TPS\_MANI\_03184]** { DRAFT } **CppImplementationDataType of category ASSOCIATIVE\_MAP** d对于 C++ 绑定，categoryASSOCIATIVE\_MAP 的CppImplementationDataType可以实现为

* StdCppImplementationDataType子类或作为ara::core::Map
* CustomCppImplementationDataType子类，则自定义命名空间中的映射类型（例如 my::map） （前提是可以使用可用的建模功能配置自定义命名空间中的类型）。

c *( RS\_MANI\_00039 )*

MyVariant: StdCppImplementationDataType

category = ASSOCIATIVE\_MAP

CppTemplateArgument

:

category = ASSOC\_MAP\_KEY

CppTemplateArgument

:

category = ASSOC\_MAP\_VALUE

uint16\_t:

StdCppImplementationDataType

category = VALUE

uint8\_t:

StdCppImplementationDataType

category = VALUE

+

templateType

+

templateType

+

templateArgument

+

templateArgument

**ASSOCIATIVE\_MAP模型示例**

**[TPS\_MANI\_03185]** { DRAFT }**类别**ASSOCIATIVE\_MAP的**CppImplementationDataType的结构**d归结为 ara::core::Map 的类别 ASSOCIATIVE\_MAP 的 CppImplementationDataType 应聚合以下CppTemplateArgument s ：

* 一个CppTemplateArgument应具有类别ASSOC\_MAP\_KEY 并应使用模板类型引用引用CppImplementationDataType 。此CppTemplateArgument表示对应于ApplicationAssocMapDataType的角色。键并定义相应的数据类型详细信息。
* 一个CppTemplateArgument应具有ASSOC\_MAP\_VALUE类别，并应使用templateType引用引用CppImplementationDataType 。此CppTemplateArgument表示对应于ApplicationAssocMapDataType的角色。值并定义相应的数据类型详细信息。
* 一个额外的可选CppTemplateArgument允许使用分配器引用来引用分配器。

c *( RS\_MANI\_00039 )*

图3.14中描述的示例显示了ASSOCIATIVE\_MAP的定义，它有两个CppTemplateArgument ，一个用于键，一个用于值。

**3.3.3.10 变体数据类型**

**[TPS\_MANI\_03189]** { DRAFT } VARIANT类别的**CppImplementationDataType的定义VARIANT**类别的CppImplementationDataType

表示类型安全的联合。 c *( RS\_MANI\_00039 )*

**[TPS\_MANI\_03190]** { DRAFT }**类别的CppImplementationDataType**

**VARIANT** d对于 C++ 绑定， CppImplementationDataType类别

VARIANT可以实现为

* StdCppImplementationDataType子类或作为ara::core::Variant
* CustomCppImplementationDataType子类，则自定义命名空间中的变体类型（例如 my::variant） （前提是可以使用可用的建模功能配置自定义命名空间中的类型）。

c *( RS\_MANI\_00039 )*

**[TPS\_MANI\_03191]** { DRAFT }**存储在VARIANT** d中的替代类型的定义 存储在 VARIANT类别的CppImplementationDataType中的替代类型由聚合的templateArgument和定义CppTemplateArgument数据类型的相应templateType定义。 c *( RS\_MANI\_00039 )*

**[constr\_3429]** { DRAFT }**没有分配器用于**categoryVARIANT的CppImplementationDataType d **CppImplementationDataType的categoryVARIANT**不允许聚合指向角色 allocator 中的分配器的templateArgument 。 c *()*

MyVariant: StdCppImplementationDataType

category = VARIANT

CppTemplateArgument

:

CppTemplateArgument

:

float:

StdCppImplementationDataType

category = VALUE

uint8\_t:

StdCppImplementationDataType

category = VALUE

+

templateType

+

templateType

+

templateArgument

+

templateArgument

**VARIANT模型示例**

图3.15中描述的示例显示了具有两个CppTemplateArgument的VARIANT的定义。每个代表一种替代类型。请注意， CppImplementationDataType的CppTemplateArgument是在 ARXML 中排序的，并且此顺序在对象图中不可见。

**3.3.3.11 位域数据类型**

**[TPS\_MANI\_03202]** { DRAFT }**位域类型的定义**d在 AUTOSAR 元模型中，位域不是通过具有自己类别的CppImplementationDataType来实现的。

类别的原始StdCppImplementationDataType的上下文中定义，归结为categoryVALUE的StdCppImplementationDataType 。

BITFIELD\_TEXTTABLE类别的CompuMethod用于为原始StdCppImplementationDataType的每个位分配特殊含义。 c *( RS\_MANI\_00039 )*

BITFIELD\_TEXTTABLE类别的 CompuMethod内带有掩码的CompuScale定义了可以在与掩码匹配的数据语义方面相互独立的隔离部分。

使用类别为 BITFIELD\_TEXTTABLE 的CompuMethod的规则与 AUTOSAR 经典平台中的规则相同，并在软件组件模板 [ 1 ] 中进行了描述。

**3.3.4 ApplicationDataType和CppImplementationDataType的兼容性**

ApplicationDataType的使用意味着在某个时间点也存在相应的CppImplementationDataType 。需要在ServiceInterface中使用CppImplementationDataType作为生成 ara::com 代理和骨架的基础，以及作为网络绑定中有效负载序列化的基础。

**[TPS\_MANI\_03223]** { DRAFT } **CppImplementationDataType** d的存在 在为ServiceInterface生成服务头文件的方法步骤之前，不需要**存在CppImplementationDataType** 。在到达方法中的这一步之前，仅使用ApplicationDataType来描述ServiceInterface的语义是完全可行的。 c *( RS\_MANI\_00003 )*

因此，有必要定义兼容性规则，明确阐明ApplicationDataType与CppImplementationDataType的一致性，反之亦然。

几个规则取决于数据类型的类别：

1. 作为一般规则，如果CppImplementationDataType的categoryTYPE\_-

REFERENCE的目标是类型映射，下面给出的所有规则都适用于CppImplementationDataType ，它在解析所有此类引用后最终有效。

这在所有规则中都没有重复。例如，如果文档声明给定的

ApplicationDataType可以映射到categoryVALUE的CppImplementationDataType这应包括映射到 categoryTYPE\_REFERENCE 的 CppImplementationDataType 的可能性，该CppImplementationDataType引用categoryVALUE的另一个CppImplementationDataType 。

1. **[constr\_5033]** { DRAFT }**数据类型与categoryVALUE**的兼容性d categoryVALUE的ApplicationDataType只能映射到也具有categoryVALUE的CppImplementationDataType 。 c *()*

CppImplementationDataType产生的 C++ 数据类型应该能够表达ApplicationDataType所需的所有数值。

如果可以表示的数值范围满足此条件

C++ 数据类型至少涵盖了ApplicationDataType中的限制所定义的范围。 swDataDefProps 。 dataConstr （这是通过CompuMethod转换的内部限制或物理限制，也必须由ApplicationDataType提供）。

如果 C++ 数据类型覆盖CompuMethod中为枚举定义的范围，则也满足条件。

1. **[constr\_5034]** { DRAFT }**数据类型与categoryBOOLEAN的兼容性**d categoryBOOLEAN的ApplicationDataType只能映射到categoryVALUE的CppImplementationDataType 。 c *()*
2. **[constr\_5035]** { DRAFT }**数据类型与categorySTRING的兼容性**d categorySTRING的ApplicationDataType只能映射到categorySTRING的CppImplementationDataType 。 c *()*
3. **[constr\_5036]** { DRAFT }**数据类型与categoryARRAY** d的兼容性categoryARRAY的ApplicationDataType只能映射到
   * categoryARRAY的CppImplementationDataType**或**\_
   * categoryVECTOR的CppImplementationDataType 。 \_ c *()*

CppImplementationDataType的数组元素的数组大小和类型应能够按顺序 1:1 映射/传输到相应的应用程序数据，反之亦然。

1. **[constr\_5037]** { DRAFT }**数据类型与categoryARRAY和variableSize** d的兼容性一个ApplicationDataType的categoryARRAY包括一个ApplicationArrayElement ，其中arraySizeSemantics在定义的维度之一中设置为variableSize应映射到
   * 类别VECTOR的CppImplementationDataType c ( *)*
2. **[constr\_5038]** { DRAFT }**数据类型与categoryARRAY和fixedSize** d的兼容性类别 ARRAY的ApplicationDataType仅包括ApplicationArrayElement s 且arraySizeSemantics在所有定义的维度中设置为fixedSize应映射到
   * categoryARRAY c ( *)*的CppImplementationDataType
3. **[constr\_5039]** { DRAFT }**数据类型与categorySTRUC-的兼容性**

**TURE** d STRUCTURE 类别的ApplicationDataType只能映射到categorySTRUCTURE的CppImplementationDataType 。 c *()*

这意味着，相应的元素对也应具有兼容的类型。

1. **[constr\_5040]** { DRAFT }**都表示可选元素结构的ApplicationRecordDataType和CppImplementationDataType的兼容性**d如果相应的元素对具有相同的元素，则表示可选元素结构的ApplicationRecordDataType只能映射到表示可选元素结构的categorySTRUCTURE的CppImplementationDataType isOptional属性的值。 c *()*
2. **[constr\_5041]** { DRAFT }**数据类型与categoryASSO-的兼容性**

**CIATIVE\_MAP** d categoryASSOCIATIVE\_MAP的ApplicationDataType只能映射到categoryASSOCIATIVE\_MAP的CppImplementationDataType 。 c *()*

1. **[constr\_5042]** { DRAFT }没有**CppImplementation的数据类型映射-**

**categoryVARIANT的DataType** d ApplicationDataType永远不能映射到categoryVARIANT的CppImplementationDataType 。 c *()*

1. **[constr\_5043]** { DRAFT }**禁止映射到CppImplementation-**

**DataType** d自适应平台不支持 COM\_AXIS、RES\_AXIS、CURVE、MAP、CUBOID、CUBE\_4、CUBE\_5类别的ApplicationDataType ，因此不能映射到CppImplementationDataType 。 C

### ()

请注意，不支持 [ constr\_5043 ] 中列出的类别，因为在 Adaptive Platform 中没有使用案例。

在 AUTOSAR 经典平台上，不需要在DataTypeMap中考虑复合数据类型的元素。这一规定的动机是 AUTOSAR 经典平台上的复合数据类型的元素不一定具有对 ImplementationDataType 的引用。

在 AUTOSAR 自适应平台上情况有所不同。 CppImplementationDataTypeElement始终需要对正式的CppImplementationDataType的引用。

DataTypeMap中提及所有相关数据类型，则数据类型定义的处理会变得更加容易，因此会激发[ constr\_5044 ] 的存在。

**[constr\_5044]** { DRAFT }**复合数据类型**d的**DataTypeMap**在给定ServiceInterface的上下文中，应描述在事件、字段、方法的上下文中使用的ApplicationCompositeDataType定义的上下文中使用的所有ApplicationDataType和CppImplementationDataType对在DataTypeMap中，该 DataTypeMap 包含在 PortInterfaceToDataTypeMapping 中引用的DataTypeMappingSet之一中，该PortInterfaceToDataTypeMapping也引用了提到的ServiceInterface 。 c *()*

## 3.4 服务接口

**3.4.1 概述**

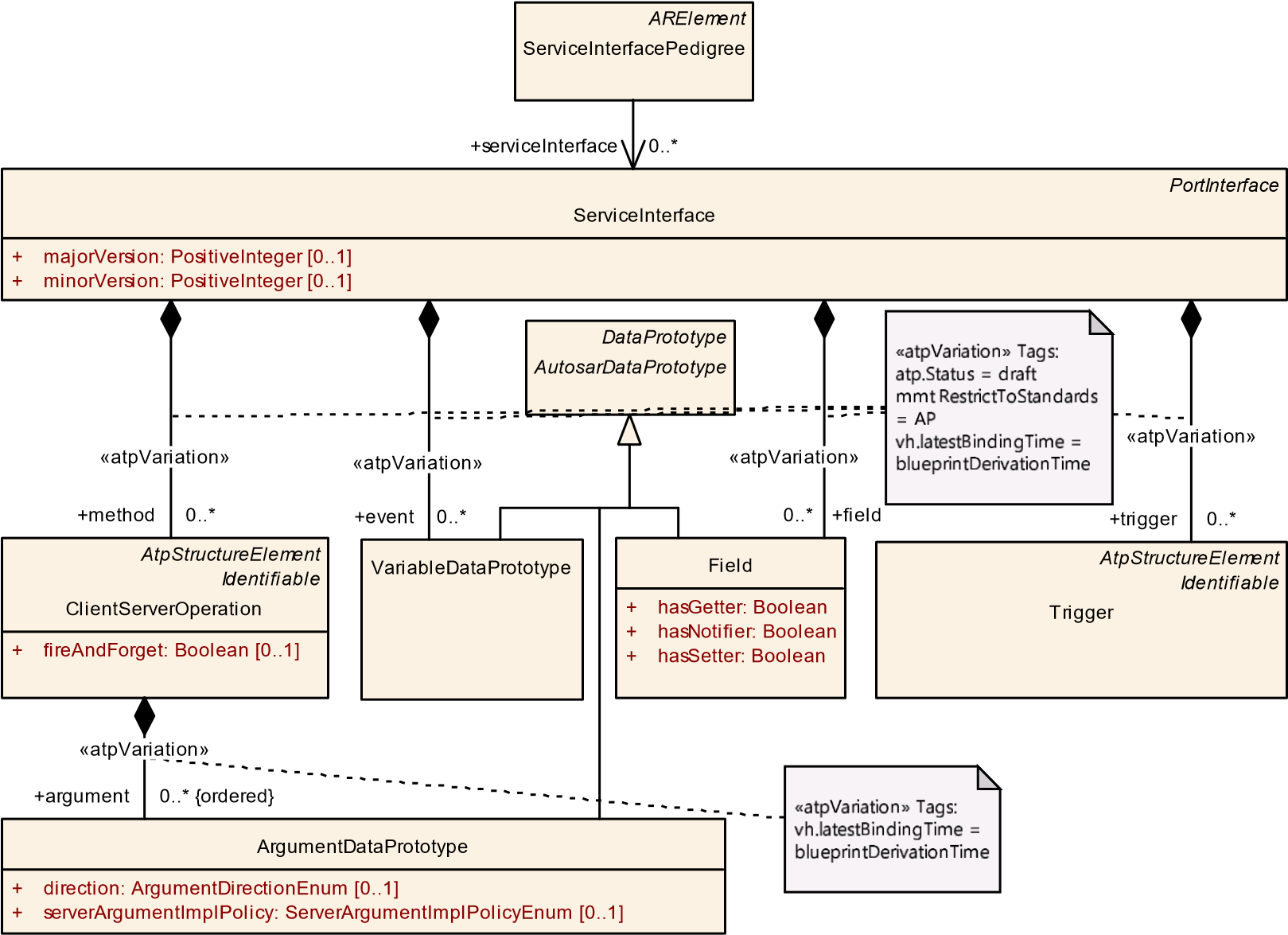
**[TPS\_MANI\_01001]** { DRAFT } **ServiceInterface的含义**d元类ServiceInterface继承自PortInterface并允许元素的异构聚合，即可以混合

* 角色事件中VariableDataPrototype的聚合
* 角色字段中元类字段的聚合
* 角色方法中ClientServerOperation的聚合
* Trigger在**同一**ServiceInterface内的角色触发器中的聚合。 c *( RS\_MANI\_00001 , RS\_MANI\_00003 )*

这种建模的目的是拥抱面向服务的通信的概念[ 3 ]，并更好地支持这种范式，以便在*AUTOSAR 自适应平台上进行通信*。

请注意，就语义而言， ApApplicationError代表ServiceInterface范围内的一种二等公民（仅在角色方法中存在ClientServerOperation时才有意义） 。

更多信息可以在第3.4.8节中找到。



**图 3.16： ServiceInterface的建模**

**[constr\_1483]** { DRAFT } ServiceInterface的**适用性**d **ServiceInterface**的适用性应仅限于*AUTOSAR 自适应平台*，即，如果PortPrototype由AdaptiveApplicationSwComponentType或在Executable的上下文。 c *()*

请注意，在*AUTOSAR 自适应平台上*，存在使用ServiceInterface的用例，但**不**存在相应的PortPrototype 。更多解释请参考[ TPS\_MANI\_01032 ]。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **服务接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 这代表了定义由方法、事件和字段的异构集合组成的 PortInterface 的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=服务接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *识别*，*多语言可引用*，*可打包元素*，端口接口，*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 事件 | 变量数据原型 | \* | 聚合 | 这表示在 ServiceInterface 的上下文中定义的事件集合。  **刻板印象：** atpVariation**标签：**  atp.Status=草稿  vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30 |
| 场地 | 场地 | \* | 聚合 | 这表示在 ServiceInterface 的上下文中定义的字段的集合。  **刻板印象：** atpVariation**标签：**  atp.Status=草稿  vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=40 |
| 主要版本 | 正整数 | 0..1 | 属性 | 服务合同的主要版本。  **标签：**  atp.Status=草稿 xml.sequenceOffset=10 |
| 方法 | 客户端服务器操作 | \* | 聚合 | 这表示在 ServiceInterface 的上下文中定义的方法的集合。  **刻板印象：** atpVariation**标签：**  atp.Status=草稿  vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=50 |
| 次要版本 | 正整数 | 0..1 | 属性 | 服务合同的次要版本。  **标签：**  atp.Status=草稿 xml.sequenceOffset=20 |
| 扳机 | 扳机 | \* | 聚合 | 这表示在 ServiceInterface 的上下文中定义的触发器集合。  **刻板印象：** atpVariation**标签：**  atp.Status=草稿  vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=60 |

**表 3.23：服务接口**

如 [TPS\_SWCT\_01844] 中所述，AUTOSAR 不支持ClientServerOperation中存在可选参数s 。

*AUTOSAR 经典平台*上 RTE 的 API 中缺乏对可选参数的支持。为了*经典平台*和*自适应平台之间的互操作性，在AUTOSAR自适应平台*上也观察到了这种限制。

**[TPS\_MANI\_01007]** { DRAFT }**服务发现的原子单元**d就应用层而言， *AUTOSAR 自适应平台*上**服务发现的原子单元**是ServiceInterface 。 c *( RS\_MANI\_00003 )*

请注意，没有义务在给定ServiceInterface的上下文中定义任何方法、事件、触发器或字段。换句话说， ServiceInterface本身的存在代表了一个有效的语义，它本身就有一个值。

例如，可能存在这样的用例，其中提供与此类ServiceInterface对应的给定服务实例的目的仅仅是为了表明提供服务实例的 ECU 正在为某些事情做好准备，例如被诊断。

然后，测试人员可以将报价的存在作为启动与相应 ECU 的连接的指示。

**3.4.2 事件**

**[TPS\_MANI\_01033]** {草案} **ServiceInterface的语义。 event** d一个事件表示对一条数据的更新。服务器决定何时发送此更新并确保事件完全控制该值。

事件的发生从服务器传输到一个或多个客户端。 c *( RS\_MANI\_00003 )*

**[constr\_1494]** { DRAFT }**事件**d的**初始值**一个ServiceInterface 。事件不应有initValue 。 **\_** c *()*

对于客户端，获取事件值的唯一方法是从服务器接收事件更新。

正如 [ constr\_1494 ] 中提到的，服务器始终可以完全控制事件的值以及何时将其发送给客户端。因此，不需要定义initValue 。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **变量数据原型** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes | | | |
| ***笔记*** | VariableDataPrototype 用于包含 ECU 应用程序中的值。这意味着很可能一个 VariableDataPrototype 在 ECU 上分配“静态”内存。在某些情况下，优化策略可能会导致可以避免内存分配的情况。  特别是，VariableDataPrototype 的值可能会随着使用它的 ECU 的执行而改变。 | | | |
| ***根据*** | *ARObject* , *AtpFeature* , *AtpPrototype* , *AutosarDataPrototype* , *DataPrototype* , *Identifiable* ,*多语言*  *可推荐*的,*可推荐的* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 初始值 | 价值规范 | 0..1 | 聚合 | 指定 VariableDataPrototype 的初始值 |

**表 3.24：变量数据原型**

**3.4.3 触发器**

**[TPS\_MANI\_03291]** {草案} **ServiceInterface的语义。触发器**d触发器代表一种特殊类型的事件，没有任何数据，它从服务器传输到一个或多个客户端，并且服务消费者应以特定方式做出反应。 c *( RS\_MANI\_00003 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **扳机** | | | |
| ***包裹*** | M2::AUTOSARTemplates::CommonStructure::TriggerDeclaration | | | |
| ***笔记*** | 触发器代表一种特殊类型的事件（无数据），服务消费者应在该事件发生时以特定方式作出反应。 | | | |
| ***根据*** | *ARObject* , *AtpClassifier* , *AtpFeature* , *AtpStructureElement* , *Identifiable* , *MultilanguageReferable* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.25：触发器**

请注意，触发器是按队列方式处理的，即触发器存储在一个队列中，按照“先进先出”的顺序进行处理。

**3.4.4 字段**

**[TPS\_MANI\_01034]** {草案} **ServiceInterface的语义。字段**d字段表示由服务器托管的一段数据，它向一个或多个客户端公开 get 访问器和/或 set mutator。

客户端可以选择接收字段值更改的通知。 c *( RS\_MANI\_00003 )*

与event相比，字段在任何时候都有一个具体的值。这种概念上的差异可以通过以下示例来解释：

事件语义的一个例子。交通标志的检测代表了一个离散的事件，只要检测到限速标志，服务组件就会触发该事件。

另一方面，让车载空调的温度预设作为在任何给定时间具有具体值的字段的示例。具体值可以由客户设定，也可以根据客户的要求获得，同时温度预设的变化本身就代表了相关信息。

总之，这意味着如果一个字段是用hasNotifier定义的，并且客户端订阅了它，那么一旦对该字段的订阅生效，该字段的当前值就会立即以类似事件的通知模式发送回订阅者.

字段的值更新时，都会向订阅者发送额外的更新通知。

在更专业的术语中，客户端可以检索当前字段值的 get() 访问器方法。通过调用 set() mutator 方法，字段值可以由客户端更新。

请注意，字段提供的所有功能都是可选的，只要满足 [ constr\_1673 ]。在服务接口中。字段描述 它定义了字段是否支持 on-change-notification ( hasNotifier )、get() 访问器 ( hasGetter ) 或 set() mutator ( hasSetter )。

诚然，字段的概念大致相当于一个事件的聚合与相关的 get()/set()方法。

就元模型而言，字段在任何时候都应具有具体值这一事实要求为字段定义初始**值**。 [ TPS\_MANI\_03212 ]阐明了这一方面。

元类字段作为ServiceInterface中的一等公民的存在表示除了单个事件和单个方法的存在之外，两个定义的访问器/更改器方法 get() 和 set() 应用于**相同的数据对象**，并且定义的字段通知器将此数据对象的每个值更改报告给订阅者。

换句话说，元类Field的语义完全由属性hasGetter 、 hasSetter和hasNotifier 决定。

因此，所有这些属性都设置为 False 的字段将没有任何有用的意义，因此不应存在。

**[constr\_1673]** { DRAFT }**属性的存在hasGetter 、 hasSetter和hasNotifier** d对于任何给定的Field ，所有属性

* 有吸气剂
* hasSetter
* hasNotifier

应存在且至少有一个属性应设置为 True。 c *()*

请注意，[ constr\_1673 ] 允许使用通知程序定义字段，但没有定义的两个方法 get() 和 set()。如上所述，字段通知器的订阅者将在订阅后立即获得字段的当前值。与仅在事件触发后才发送值的事件的使用相比，此功能使没有 get()/set() 方法的字段在某些功能情况下有用。

|  |  |
| --- | --- |
| ***班级*** | **场地** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **场地** | | | |
| ***笔记*** | 此元类表示定义可以通过读和/或写语义访问的一段数据的能力。如果数据的值发生变化，也可以生成通知。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *AtpFeature* , *AtpPrototype* , *AutosarDataPrototype* , *DataPrototype* , *Identifiable* ,*多语言*  *可推荐*的,*可推荐的* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 有吸气剂 | 布尔值 | 1 | 属性 | 此属性控制是否预见到对该字段的读取访问。  **标签：** atp.Status=draft |
| hasNotifier | 布尔值 | 1 | 属性 | 此属性控制是否为该字段预见通知语义。  **标签：** atp.Status=draft |
| hasSetter | 布尔值 | 1 | 属性 | 该属性控制是否预见到对该字段的写访问。  **标签：** atp.Status=draft |

**表 3.26：字段**

**3.4.5 方法**

**[TPS\_MANI\_01035]** {草案} **ServiceInterface的语义。方法**d方法表示一种功能，该功能在一个或多个客户端的请求下由服务器执行并在其范围内执行。 c *( RS\_MANI\_00003 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **客户端服务器操作** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::PortInterface | | | |
| ***笔记*** | 在客户端/服务器接口范围内声明的操作。 | | | |
| ***根据*** | *ARObject* , *AtpClassifier* , *AtpFeature* , *AtpStructureElement* , *Identifiable* , *MultilanguageReferable* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 论据（有序） | 参数数据原型 | \* | 聚合 | 此 ClientServerOperation 的参数  **刻板印象：** atpVariation  **标签：** vh.latestBindingTime=blueprintDerivationTime |
| 火与忘 | 布尔值 | 0..1 | 属性 | 此属性定义此方法是否为 fire&forget 方法 (true) 或不是 (false)。  **标签：** atp.Status=draft |
| 可能的ApError | 应用程序错误 | \* | 参考 | 此参考将 AdaptivePlatformApplication 错误标识为由封闭的 Client ServerOperation 引发的可能错误。  **标签：** atp.Status=draft |
| 可能的ApError 集 | 应用程序错误集 | \* | 参考 | 此引用表示将整个 ApApplicationError 组引用为一个模型元素的能力，而不必单独引用所有表示的 Ap ApplicationError。  **标签：** atp.Status=draft |

**表 3.27：客户端服务器操作**

**3.4.5.1 Fire and Forget 方法**

所谓的“即发即弃”方法代表一种特殊形式的方法，专门用于将信息从客户端传送到服务器。

不期望该方法的实现执行任何类型的算法。

从这个角度来看，“fire & forget”方法的语义与事件的语义相当，只是相反。

换句话说，“fire & forget”方法将数据和数据的发生**从客户端传送到服务器**。为了比较，事件用于结合信息从**服务器到客户端的发生来传递信息**。

该语句的*出现*方面的结果是，例如“fire & forget”调用的数量可以通过服务器的实现来计算，并且可以采用此元信息在实际数据之上传达额外的语义。

**[TPS\_MANI\_01064]** {草稿}**属性方法的语义。 fireAndForget** d给定方法的“fire & forget”语义的激活是通过设置属性method的值来实现的。 fireAndForget的值为 true。 c *( RS\_MANI\_00003 )*

**[TPS\_MANI\_03118]** {草案} **ServiceInterface的语义。 fireAndForget设置为 true的方法**d fireAndForget设置为 true的方法表示一个 void-return-method，其中客户端不期望来自服务器端的任何类型的确认或握手。 c *( RS\_MANI\_00003 )*

**[constr\_3374]** { DRAFT }**属性fireAndForget设置为 true 的方法不应有任何 inout 或 out 参数**d属性fireAndForget的值设置为 true的方法不允许有任何方向为 inout或out的参数s 。 c *()*

**[constr\_3375]** { DRAFT }**属性fireAndForget设置为 true**的**方法不得引用ApApplicationError** d属性fireAndForget的值设置为 true 的方法不允许引用

* 角色possibleApError和/或中的ApApplicationError
* 角色possibleApErrorSet中的ApApplicationErrorSet 。 c *()*

**[TPS\_MANI\_03119]** { DRAFT }**元类ClientServerOperation** d属性**fireAndForget的默认值**如果未定义属性fireAndForget ，则应假定不打算使用“fire & forget”语义。 c *( RS\_MANI\_00003 )*

**3.4.6 ServiceInterface的版本控制**

使用同一ServiceInterface的多个版本支持服务的独立生命周期，并允许在不影响现有消费者的情况下更改和增强ServiceInterface 。本章介绍如何对同一ServiceInterface的不同版本进行建模。

ServiceInterface的版本可以定义为例如具有自己的短名称的 ServiceInterface（例如 Service\_Version1、Service\_Version2）或位于自己的 ARPackage中的ServiceInterface （例如/Version1/Service、/Version2/Service）。

还允许为不同的ServiceInterface版本分配不同的*命名空间以*影响生成的代码，例如生成com::version1::Service 和com::version2::Service。

预计如果在一个Executable中使用相同ServiceInterface的不同版本，则每个ServiceInterface版本应使用不同的*命名空间。*

属性ServiceInterface 。主要版本和服务接口。 minorVersion提供了在ServiceInterface级别定义版本信息的可能性。

**[TPS\_MANI\_03616]** { DRAFT } **ServiceInterface的语义版本控制。主要版本和服务接口。 minorVersion** d服务合约版本控制规则：

* 对于向后不兼容的接口或行为更改， majorVersion编号应增加， minorVersion编号应设置为 0
* 对于向后兼容的接口或行为更改， majorVersion编号应保持不变， minorVersion编号应增加。 c *( RS\_MANI\_00064 )*

请注意，关于向后兼容性的决定应该由服务设计者做出。换句话说，AUTOSAR 没有为ServiceInterface的向后兼容性定义正式标准。

对于多个版本的 ServiceInterface 的建模， ServiceInterface的完全限定的shortName必须不同。 ServiceInterfacePedigree允许收集ServiceInterface的集合，这些集合形成同一*Service的不同版本的集合*。

|  |  |
| --- | --- |
| ***班级*** | **服务接口谱系** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **服务接口谱系** | | | |
| ***笔记*** | 属于同一版本的 ServiceInterfaces 的集合。  **标签：**  atp.Status=草稿  atp.recommendedPackage=ServiceInterfacePedigrees | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 服务接口 | 服务接口 | \* | 参考 | 对属于同一版本的 ServiceInterfaces 的引用。  **标签：** atp.Status=draft |

**表 3.28：ServiceInterfacePedigree**

此服务的其他使用者不需要切换到使用此ServiceInterface的最新版本，但可以继续使用他们设计和测试的旧版本的ServiceInterface 。

**3.4.6.1 传输层驱动的版本控制**

每个传输层机制（例如 SOME/IP）可以定义自己的兼容性规则。因此，对于每个单独的传输层，无论更改的服务接口在该传输层上是否具有不兼容的表示，都需要对兼容性进行自己的影响评估。

兼容性取决于传输层上使用的功能。例如，在 SOME/IP 中，放在结构前面的长度字段允许在反序列化期间跳过可扩展数据结构末尾的未知元素。

SOME/IP 中的一个附加选项是在可选结构成员前面使用数据 ID。使用这种方法，接收器可以跳过结构的未知成员，即数据 ID 未知的地方。

因此，在应用程序设计级别， ServiceInterface的所有更改都应谨慎处理，因为只有使用的传输层和传输层上使用的功能决定更改是否兼容。

如果要确保ServiceInterface中的两个AutosarDataPrototype是兼容的，那么两个AutosarDataPrototype都应由相同的AutosarDataType键入。

在ServiceInterfaceDeployment期间， ServiceInterface被映射到中间件传输层，在该中间件传输层执行必要的中间件传输层特定配置设置，如第10.2章所述。

例如，可以将相同的 SOME/IP serviceInterfaceId分配给相同ServiceInterface的不同版本，但分配不同的majorVersion或minorVersion 。

这种方法考虑到ServiceInterface的兼容性受到使用的传输绑定的严重影响。

Version1

ServiceInterface

MyInterface

Event1

Event2

Version2

ServiceInterface

MyInterface

Event1

Event2

Method

Version3

ServiceInterface

MyInterface

Event1

Method

SomeipSIDeployment

ServiceInterfaceId =

46

-

1

majorVersion =

-

0

minorVersion =

-

SomeipSIDeployment

ServiceInterfaceId =

46

-

1

majorVersion =

-

1

minorVersion =

-

SomeipSIDeployment

-

ServiceInterfaceId =

46

-

majorVersion =

2

-

minorVersion =

0

**图 3.17：同一ServiceInterface的不同版本的示例**

请注意，SOME/IP 的兼容性规则在 [ 7 ] 中有描述。

**3.4.7 命名空间**

ServiceInterface的定义直接影响*AUTOSAR 自适应平台上的应用程序代码*。

在这一点上无需过多介绍，有必要在ServiceInterface的上下文中支持*命名空间的定义*。

命名空间应用于封装与ServiceInterface相关的源代码，从而避免与ServiceInterface的其他定义的内容发生名称冲突。

原则上，围绕具体ServiceInterface的命名空间的定义可以从包含ServiceInterface定义的ARPackage的结构中导出。但是，这种方法对包结构施加了一些限制。

相同的ServiceInterface可以用于不同的项目，这些项目可能需要也可能不需要使用特定的*不同*包结构。

将相同的ServiceInterface放置在可能不同的包层次结构中会导致定义不同的命名空间，因此需要创建或生成表示ServiceInterface的代码**以及**一次又一次使用此定义的代码。

克服此潜在问题的一种方法是将专用命名空间定义附加到ServiceInterface本身的定义中。

这种方法记录在图3.18中。

SymbolProps

*Referrable*

*ImplementationProps*

+

symbol: CIdentifier

[0..1]

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

«atpSplitable»

+

namespace

0..\* {

ordered

}

**图 3.18： PortInterface中命名空间的规范**

**[TPS\_MANI\_01004]** {草稿} **ServiceInterface的语义。 namespace** d聚合ServiceInterface 。命名空间应用于定义用于与给定ServiceInterface对应的源代码的命名空间。 c *( RS\_MANI\_00003 )*

**[TPS\_MANI\_01005]** { DRAFT } **ServiceInterface的命名空间的定义可能遵循分层模式**d ServiceInterface的命名空间可能遵循许多现代编程语言所支持的分层模式。

分层命名空间定义的元素之间的分隔符取决于所使用的编程语言，并且未在模型中明确定义。

该模型只定义了分层命名空间模式的元素。 c *( RS\_-*

*MANI\_00003 )*

**清单 3.3：给定ServiceInterface的命名空间定义示例**

**<服务接口>**

**<SHORT-NAME>我的服务接口</SHORT-NAME** >

**<命名空间>**

**<符号-道具>**

**<SHORT-NAME>**第一个**</SHORT-NAME>**

**<符号>** com **</符号>**

**</符号-道具>**

**<符号-道具>**

**<SHORT-NAME>**第二个**</SHORT-NAME>**

**<SYMBOL>**我的公司**</SYMBOL>**

**</符号-道具>**

**<符号-道具>**

**<SHORT-NAME>**第三个**</SHORT-NAME>**

**<SYMBOL>**软件**</SYMBOL>**

**</符号-道具>**

**</命名空间>**

**</服务接口>**

由于能够定义分层命名空间，聚合ServiceInterface 。命名空间被限定为被订购。

这意味着单个元素对命名空间集合的顺序具有语义相关性[[5]](#footnote-5)。

**[TPS\_MANI\_01006]** { DRAFT } **ServiceInterface的有序定义。 namespace** d在ServiceInterface的分层定义中。命名空间命名空间片段的顺序应在命名空间到源代码的转换中保持。

换句话说，第一个命名空间片段应该首先出现，然后是第二个命名空间片段，以此类推。 c *( RS\_MANI\_00003 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***端口接口***（抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::PortInterface | | | |
| ***笔记*** | 由软件组件的端口提供或需要的接口的抽象基类。 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *识别*，*多语言可引用*，可打包元素，*可引用* | | | |
| ***子类*** | *AbstractRawDataStreamInterface* , *AbstractSynchronizedTimeBaseInterface* , ClientServerInterface ,  *CryptoInterface* , *DataInterface* , *DiagnosticPortInterface* , LogAndTraceInterface , ModeSwitchInterface ,  *PersistencyInterface* , *PlatformHealthManagementInterface* , SecurityEventReportInterface ,服务接口, TriggerInterface | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 命名空间  （订购） | 符号道具 | \* | 聚合 | 这表示用于定义分层命名空间的 SymbolProps，该命名空间适用于根据 ServiceInterface 的定义生成代码工件。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=namespace.shortName atp.Status=草稿 |

**表 3.29：端口接口**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **符号道具** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Components | | | |
| ***笔记*** | 此元类表示贡献名称空间的一部分的能力。 | | | |
| ***根据*** | *ARObject* , *ImplementationProps* ,*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.30：SymbolProps**

清单3.3举例说明了 [ TPS\_MANI\_01006 ] 的陈述，即在例如 C++ 中生成的名称空间看起来像清单3.4中的草图。

1. 命名空间com {
2. 命名空间我的公司 {
3. 命名空间软件{

4

1. }
2. }
3. }

**清单 3.4：示例ServiceInterface的结果命名空间**

**3.4.8 错误处理**

*AUTOSAR 自适应平台*上的错误处理建模与在*AUTOSAR 经典平台*上实现的方法略有不同。

特别是，方法执行期间错误的正式表示是在全局范围内完成的，即这样的定义可以被任何ServiceInterface任意重用。

**[TPS\_MANI\_01190]** { DRAFT } **ApApplicationError** d元类ApApplicationError的**语义表示在**方法执行期间定义错误存在的能力，而与ServiceInterface或ClientServerOperation的范围无关。 c *( RS\_MANI\_00002 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **应用程序错误** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 此元类表示在 AUTOSAR 自适应平台上正式指定应用程序错误语义的能力  **标签：**  atp.Status=草稿  atp.recommendedPackage=应用程序错误 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 错误代码 | 整数 | 1 | 属性 | 此属性能够在封闭的 AdaptivePlatformApplication 错误中指定错误代码值。  **标签：** atp.Status=draft |
| 错误域 | 应用程序错误  领域 | 1 | 参考 | 此引用表示 Ap ApplicationError 的错误域。  **标签：** atp.Status=draft |

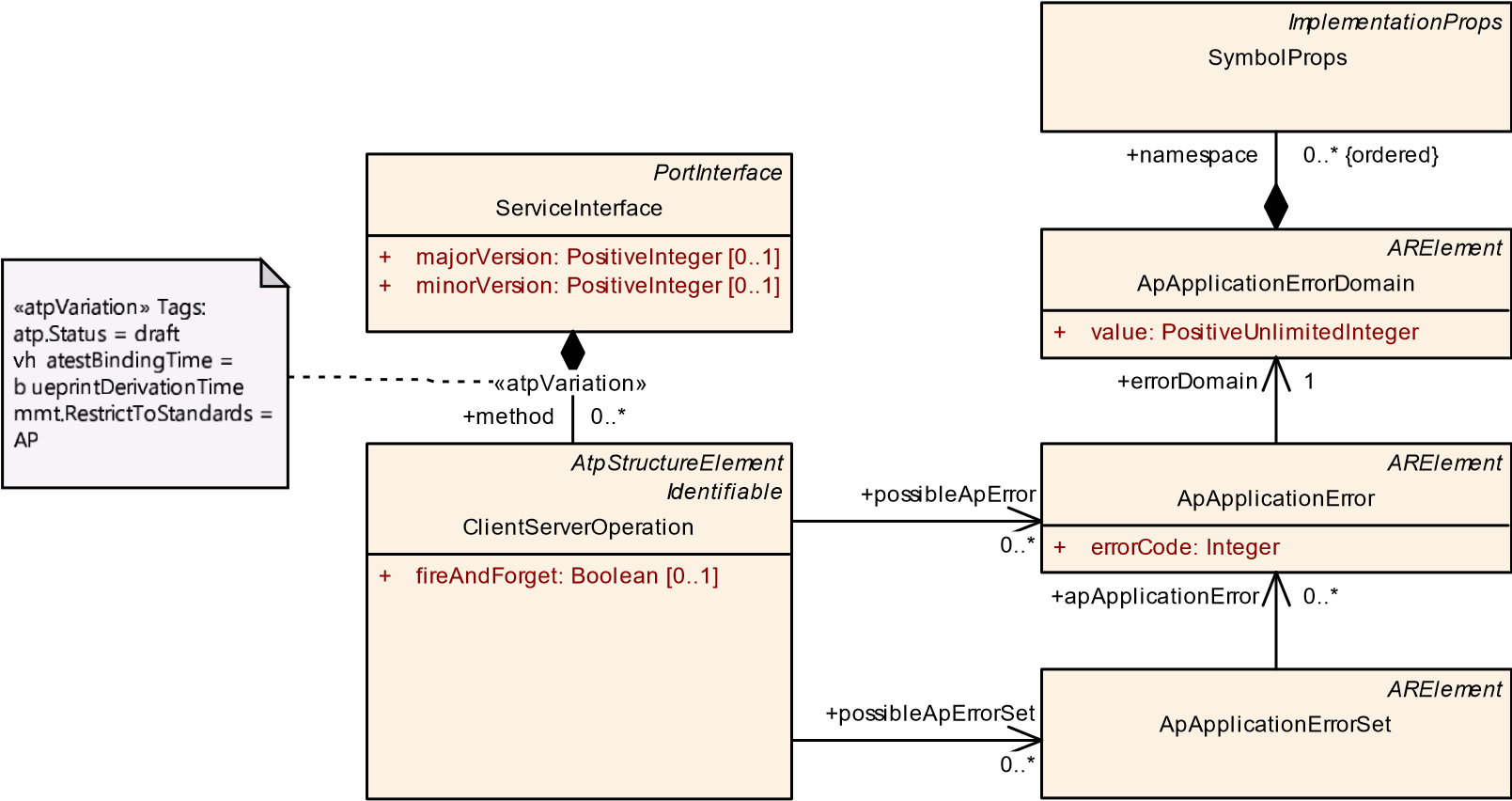
**表 3.31：ApApplicationError**

**[TPS\_MANI\_01198]** { DRAFT } **ApApplicationErrorSet** d元类的**语义**ApApplicationErrorSet能够对ApApplicationError的引用进行分组，因此代表这组对ClientServerOperation的引用的“代理” 。

这种建模能力的用例是一些ClientServerOperation可能必须引用相同数量的ApApplicationError 。

让每个ClientServerOperation重复对ApApplicationError的同一组引用被认为是不必要的，因此提供了引用组而不是单个引用的能力作为替代方法。 c *( RS\_MANI\_00026 )*

是否从特定ClientServerOperation定义和引用ApApplicationErrorSet的决定必须单独完成。 AUTOSAR 只是想让这项业务尽可能简单。



**图 3.19： ApApplicationError在*AUTOSAR 自适应平台上的建模***

请注意，也可以混合使用ClientServer-

操作。可能的ApError和ClientServerOperation 。可能的 ApErrorSet 。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **应用程序错误集** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 这个元类充当代表整个 APApplicationErrors 集合的参考目标。这减轻了引用大量 ApApplication 错误的 ClientServerOperations 的负担。  **标签：**  atp.Status=草稿  atp.recommendedPackage=ApplicationErrorSets | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| ap应用程序错误 | 应用程序错误 | \* | 参考 | 此引用代表 ApApplication 的集合  由封闭的 ApApplicationErrorSet 表示的错误  **标签：** atp.Status=draft |

**表 3.32：ApApplicationErrorSet**

由于ApApplicationError不再定义在ServiceInterface的范围内，因此无需通过ServiceInterfaceElementMapping的专用子类来定义两个ApApplicationError之间的映射。

**[TPS\_MANI\_01191]** { DRAFT }**可能错误的建模**d由角色方法中的ServiceInterface聚合的ClientServerOperation应参考

* possibleApError中的一个或多个ApApplicationError (s)
* 角色possibleApErrorSet中的一个或多个ApApplicationErrorSet

ClientServerOperation引发的可能错误的存在。 c *( RS\_MANI\_00026 )*

**[TPS\_MANI\_01192]** { DRAFT } **ApApplicationErrorDomain** d元类**的语义**ApApplicationErrorDomain应用于定义可能由 AUTOSAR 标准化的特定错误域。

因此，此类误差域的定义不在本标准的范围内定义。

ApApplicationError本身。相反， ApApplicationError通过角色errorDomain中的引用来标识适用的错误域。

可以将命名空间的定义附加到ApApplicationErrorDomain ，因为此信息与语言绑定相关。 c *( RS\_MANI\_00026 )*

**[constr\_1627]** { DRAFT }属性**ApApplication-支持的值范围**

**错误域。 value** d属性ApApplication支持的取值范围-

错误域。值限制在区间 [0..18446744073709551616] 内。 c *()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **应用程序错误域** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 此元类表示为 ApApplicationError 定义全局错误域的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=ApplicationErrorDomains | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 命名空间  （订购） | 符号道具 | \* | 聚合 | 这个聚合定义了 Ap ApplicationErrorDomain**标签的命名空间：** atp.Status=draft |
| 价值 | 正无限整数 | 1 | 属性 | 此属性标识错误类别。  **标签：** atp.Status=draft |

**表 3.33：ApApplicationErrorDomain**

**[constr\_1625]** { DRAFT }**引用ApApplicationError的存在。 errorDomain** d对于每个ApApplicationError ，应存在参考errorDomain 。

换言之， ApApplicationError与相应的ApApplicationErrorDomain的关联是强制性的。 c *()*

**[constr\_1664]** { DRAFT }**唯一的 ApApplicationError 。 shortName** d在引用角色errorDomain中的给定ApApplicationErrorDomain的所有ApApplicationError的集合中，属性ApApplicationError 。 shortName应具有唯一值。 c *()*

**[constr\_1665]** { DRAFT }**唯一的 ApApplicationError 。 errorCode** d在引用角色errorDomain中的给定ApApplicationErrorDomain的所有ApApplicationError的集合中，属性ApApplicationError 。 errorCode应具有唯一值。 c *()*

constr\_1664 ] 和 [ constr\_1665 ]存在的理由：C++ 的语言绑定预见到属性ApApplicationError的使用。 shortName和ApApplicationError 。 errorCode用于在ApApplicationErrorDomain的上下文中创建枚举。

枚举数标签或枚举数值的重复会导致编译时错误。

**3.4.9 服务接口数据类型映射**

*AUTOSAR 自适应平台*上实现软件的工作流程中的一个重要步骤是创建ServiceInterface的基于代码的表示，以使其可供应用程序代码访问。

这种基于代码的表示的创建通常是自动化的，并将由代码生成器执行。此代码生成器需要来自模型的输入。用于此目的的主要输入显然是ServiceInterface本身的定义。

然而，这还不够。 ServiceInterface的设计者可以自由地使用ApplicationDataType来规范ServiceInterface的细节。

因此，有必要为每个使用的ApplicationDataType提供AbstractImplementationDataType的定义。在元模型中，这种对应是通过元类DataTypeMappingSet实现的[[6]](#footnote-6)。

但是，从方法论的角度来看，让ServiceInterface直接引用一个或多个DataTypeMappingSet (s) 被认为是不合适的。

为澄清起见，这意味着ApplicationDataType到AbstractImplementationDataType的映射成为ServiceInterface定义的一个组成部分，尽管映射本身并不真正有助于ServiceInterface的实际语义。

因此，只要数据类型之间的映射发生变化，就必须更新ServiceInterface 。

但是由于ServiceInterface的定义通常被认为是非常稳定的，因此仅仅为了确认数据类型映射的变化而频繁更新是不可接受的。

在这个具体的例子中，所描述的问题可以通过定义一个引用ServiceInterface和DataTypeMappingSet的映射类来规避，因此无需更新ServiceInterface即可创建对应关系。

尽管本章的前奏表明存在将ServiceInterface映射到一个或多个DataTypeMappingSet (s) 的元类，但实际元模型的设计重点更广泛。

ServiceInterface旁边可能会有更多种类的PortInterface需要满足相同的用例。

因此，为此目的创建的元类的名称是PortInterfaceToDataTypeMapping 。

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

DataTypeMappingSet

*ARElement*

PortInterfaceToDataTypeMapping

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

+

dataTypeMappingSet

1..\*

+

portInterface

1

**图 3.20： PortInterfaceToDataTypeMapping的建模**

**[constr\_1507]** { DRAFT } **PortInterfaceToDataTypeMapping仅适用于ServiceInterface或PersistencyKeyValueStorageInterface** d PortInterfaceToDataTypeMapping 。 portInterface只能指ServiceInterface**或**PersistencyKeyValueStorageInterface 。 \_ **\_** c *()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **端口接口到数据类型映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 此元类表示将 PortInterface 与 DataTypeMappingSet 关联的能力。在单个 PortInterface 范围内生成头文件需要此关联。  该关联是有意在 PortInterface 本身的范围之外进行的，因为 PortInterface 的设计者很可能不想添加有关 ImplementationDataType 级别的详细信息。  **标签：**  atp.Status=草稿  atp.recommendedPackage=PortInterfaceToDataTypeMappings | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 数据类型  映射集 | 数据类型映射集 | 1..\* | 参考 | 这表示对适用数据 TypemappingSet 的引用  **标签：**  atp.Status=草稿  atp.StatusComment=为自适应平台保留 |
| 端口接口 | 端口接口 | 1 | 参考 | 这表示对适用端口接口的引用  **标签：**  atp.Status=草稿  atp.StatusComment=为自适应平台保留 |

**表 3.34：PortInterfaceToDataTypeMapping**

|  |  |
| --- | --- |
| ***班级*** | **数据类型映射集** |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **数据类型映射集** | | | |
| ***笔记*** | 此类表示 ApplicationDataTypes 和 ImplementationDataTypes 之间的映射列表。  此外，它还可以包含 ImplementationDataTypes 和 ModeDeclarationGroups 之间的映射。  **标签：** atp.recommendedPackage=DataTypeMappingSets | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *CollectableElement* , *Identifiable* , *Multilanguage Referrable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 数据类型映射 | 数据类型映射 | \* | 聚合 | 这是 Application DataType 与其 AbstractImplementationDataType 之间的一种特殊关联。 |
| 模式请求类型映射 | 模式请求类型映射 | \* | 聚合 | 这是 Mode DeclarationGroup 与其 AbstractImplementationData 类型之间的一种特殊关联。 |

**表 3.35：数据类型映射集**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **数据类型映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes | | | |
| ***笔记*** | 此类表示 ApplicationDataType 与其实现 Abstract ImplementationDataType 之间的关系。 | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 应用数据类型 | 应用程序数据类型 | 0..1 | 参考 | 这是对应的ApplicationDataType |
| 实现数据类型 | 抽象实现  数据类型 | 0..1 | 参考 | 这是对应的 AbstractImplementationData 类型。 |

**表 3.36：数据类型映射**

**3.4.10 通信组模式**

以双向方式定义了ServiceInterface使用的特定模式。

详细信息可在 SWS\_CommunicationManagement [ 8 ] 中找到。为了定义一个通信组几个ServiceInterface 。定义类别值。

**[TPS\_MANI\_03628]** { DRAFT } **ServiceInterface的标准化值。类别**d AUTOSAR 标准为属性ServiceInterface保留以下值。类别：

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

可以为属性ServiceInterface使用自定义的非标准化值。类别，但此选项有义务使用保证不会与标准化值集合的未来可能扩展发生冲突的值，例如在类别值中使用公司名称。 c *()*

通信组模式的一般思想是创建一个类别COMMUNICATION\_GROUP 的ServiceInterface来描述要传输的信息（msg 和responseMsg 数据类型）。系统中不会有categoryCOMMUNICATION\_GROUP这个ServiceInterface的任何实例，它只是一个设计神器。

在categoryCOMMUNICATION\_GROUP的ServiceInterface中，为服务器（ categoryCOMMUNICATION\_GROUP\_SERVER ）和客户端（ categoryCOMMUNICATION\_GROUP\_CLIENT ）角色创建了两个ServiceInterface 。 SWS\_CommunicationManagement [ 8 ]中定义了如何完成创建的规则。

**ServiceInterface**

category

:

**COMMUNICATION**

**\_**

**GROUP**

*-*

*method message*

*(*

*msg*

*)*

*-*

*event response*

*(*

*responseMsg*

*)*

**ServiceInterface**

category

:

**COMMUNICATION**

**\_**

**GROUP**

**\_**

**SERVER**

-

method broadcast

(

msg

)

-

method message

(

clientId

,

msg

)

-

event response

(

clientID

,

responseMsg

)

-

method

listClients

()

**ServiceInterface**

category

:

**COMMUNICATION**

**\_**

**GROUP**

**\_**

**CLIENT**

-

method message

(

msg

)

-

event response

(

responseMsg

)

<<

derive

>>

<<

derive

>>

**图 3.21：通信组类别示例**

## 3.5 服务接口映射

请注意，根据 [ TPS\_MANI\_01007 ]， ServiceInterface成为基于 VFB 和*外部*（即使用通信网络）通信的单一基础。

这个概念与*AUTOSAR 经典平台上的方法形成鲜明对比，*其中不同的模型元素用于 VFB 级别（ PortInterface ）和网络级别（ SystemSignal 、 ISignal和ISignalIPdu ）。

不同模型元素的使用最佳地支持基于 VFB 与基于网络的通信的不同粒度的存在。

换言之，网络层面的通信设计可能会受到不同的设计限制，例如通过定义粗粒度的通信包来保持由服务发现引起的总线负载可管理。

与此相反，VFB 级别的设计人员可能希望定义接口粒度以实现最大的可重用性。

ServiceInterfaceMapping

*PortInterface*

ServiceInterface

[0..1]

majorVersion: PositiveInteger

+

minorVersion: PositiveInteger

+

[0..1]

*AtpBlueprint*

*AtpBlueprintable*

*Identifiable*

*PortInterfaceMapping*

*AtpStructureElement*

*SwConnector*

PassThroughSwConnector

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

PortInterfaceMappingSet

+

compositeServiceInterface

1

+

sourceServiceInterface

1..\*

«atpVariation»

+

portInterfaceMapping

0..\*

+

mapping

0..1

**图 3.22： ServiceInterfaceMapping的建模**

**[TPS\_MANI\_01002]** { DRAFT }**元类ServiceInterfaceMapping的语义**d为了理清定义之间可能存在的不同动机

* ServiceInterface明确设计用于基于 VFB 的通信和
* ServiceInterface元类ServiceInterfaceMapping可用于映射
* （细粒度） ServiceInterface用于 VFB 通信
* （粗粒度） ServiceInterface用于网络通信。

c *( RS\_MANI\_00017 )*

**[TPS\_MANI\_01032]** { DRAFT } **ServiceInterfaceMapping的使用**d可以派生实现映射功能的专用AdaptiveApplicationSwComponentType 。从这个所谓的*外观*软件组件派生的SwComponentPrototype将为每个ServiceInterface公开PortPrototype 。

其他SwComponentPrototype可以“连接”到由角色sourceServiceInterface中引用的ServiceInterface键入的PortPrototype 。

这意味着由角色compositeServiceInterface中引用的ServiceInterface类型化的PortPrototype用于外部通信。

PassThroughSwConnector可用于在建模的*外观*CompositionSwComponentType中描述哪些“细粒度”端口组合为用于网络通信的“粗粒度”端口。 “细粒度”端口的服务接口元素到“粗粒度”端口的服务接口元素的映射使用ServiceInterfaceMapping或更确切地说是ServiceInterfaceElementMapping来描述。 c *( RS\_MANI\_00017 )*

请注意，*外观*SwComponentType的建模不会对实现和这种*外观*功能的实现做出任何假设。*外观*可以由自适应软件组件/应用程序实现，也可以由“网络守护程序”实现。 AUTOSAR 没有为实现此类功能定义任何指令，并且决定是特定于项目的。这种“门面”的行为方面（例如，何时提供粗粒度的 ServiceInstance）也是项目特定的，AUTOSAR 没有预定义。

Interface1

Event: event1

Interface2

Event: event2

Interface3

Event: event1, event2

SWC1

P

SWC2

P

SWC3

P

R

R

**图 3.23：外观软件组件的概念**

图3.23总结了创建*外观*软件组件背后的想法。后者能够“捆绑”由可能不同的 SwComponentType 拥有的不同PortPrototype的通信以进行外部通信。

换句话说，SWC1 拥有的元素 event1 和 SWC1 拥有的 event2 组合成一个ServiceInterface ，用于键入*外观*软件组件的一个PortPrototype 。

从与通信相关的外部角度来看，SWC3 就像 SWC1 和 SWC2 所创建的“内部结构”的一个门面，即通过 SWC3 的存在被抽象出来。

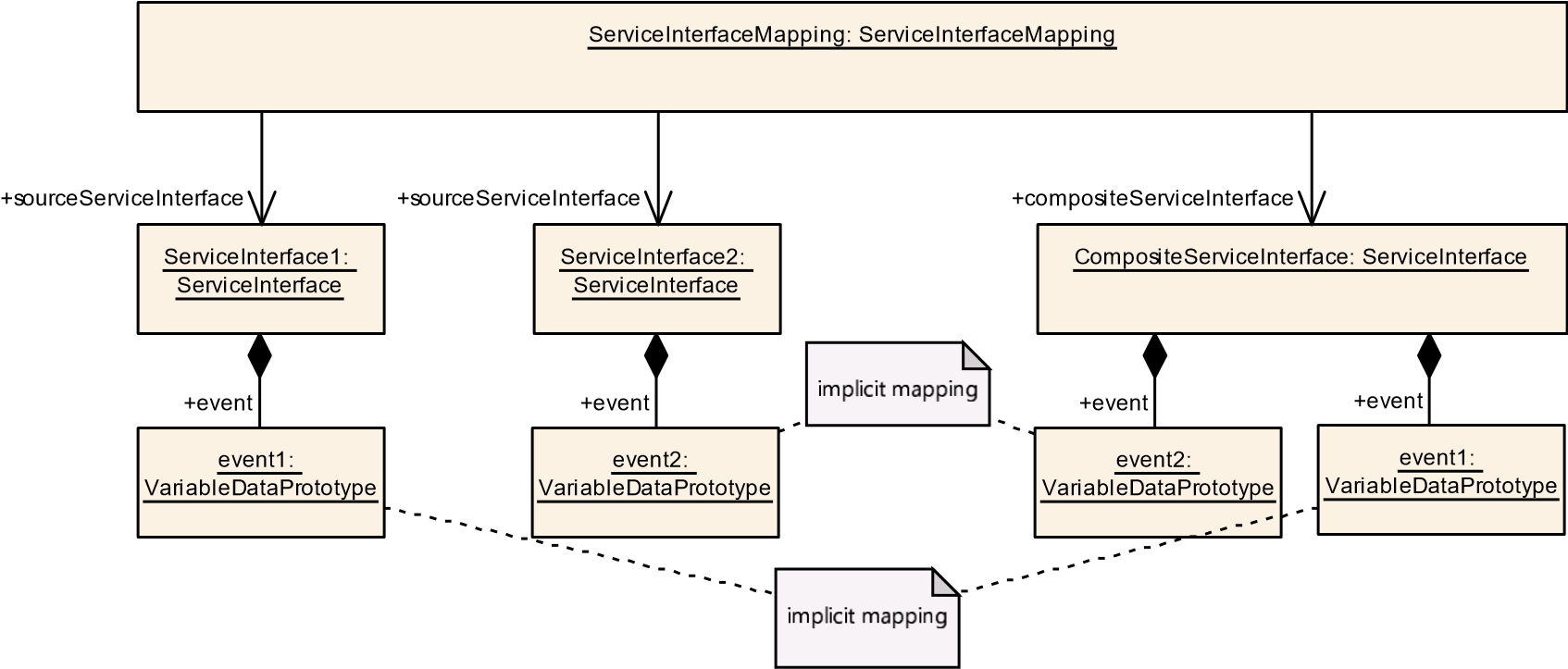
**[constr\_5056]** { DRAFT } **CompositionSwComponent子类的限制-**

**键入。连接器**d在CompositionSwComponentType的上下文中。由Executable引用的连接器（传递地） 。 rootSwComponentPrototype ，唯一支持的SwConnector子类是PassThroughSwConnector 。 c *()*

**[constr\_5057]** { DRAFT } **PassThroughSwConnector和ServiceInterfaceMapping** d如果在CompositionSwComponentType中的两个端口之间定义了PassThroughSwConnector ：

* 应定义这两个端口的ServiceInterface之间的ServiceInterfaceMapping ，并且PassThroughSwConnector应在角色映射中引用相关的ServiceInterfaceMapping或
* 应定义两个端口的 ServiceInterface 元素的 ServiceInterfaceElementMapping ，并且PassThroughSwConnector应引用角色serviceInterfaceElementMapping中的相关ServiceInterfaceElementMapping 。

### c ()



**图 3.24： ServiceInterfaceMapping应用示例**

**[TPS\_MANI\_01022]** { DRAFT } **ServiceInterfaceMapping 背后的概念 d ServiceInterfaceMapping定义**背后的概念是sourceServiceInterface的**所有元素**都必须具有相同**类型**的对应项（ ServiceInterface.event 、 ServiceInterface.field或ServiceInterface.method ）和具有相同的shortName 。 c *( RS\_MANI\_00017 )*

TPS\_MANI\_01022 ] 中所述的规定在图3.24中举例说明。

请注意， ServiceInterfaceMapping的创建被认为是一个原子步骤，这样的ServiceInterfaceMapping不太可能部分创建，然后由另一方完成。

毕竟，有相互排斥的方式来指定映射，并且ServiceInterface的部分映射的任何创建者都无法确定哪些替代方案适用于一个ServiceInterface与另一个 ServiceInterface 的特定配对，而无需知道另一个ServiceInterface （其中如果映射已经完成）。

因此，无需将对ServiceInterface的引用的较低多重性设置为 0。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **服务接口映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping | | | |
| ***笔记*** | 指定一个 ServiceInterfaceMapping，它允许定义一个 ServiceInterface 是几个其他 ServiceInterface 的组合。  **标签：**  atp.Status=草稿  atp.recommendedPackage=ServiceInterfaceMappings | | | |
| ***根据*** | *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *Identifiable* , *MultilanguageReferable* , *PortInterfaceMapping* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 合成的  服务接口 | 服务接口 | 1 | 参考 | 这表示复合服务接口。  **标签：** atp.Status=draft |
| 源服务接口 | 服务接口 | 1..\* | 参考 | 映射到复合 ServiceInterface 的 ServiceInterface。  **标签：** atp.Status=draft |

**表 3.37：服务接口映射**

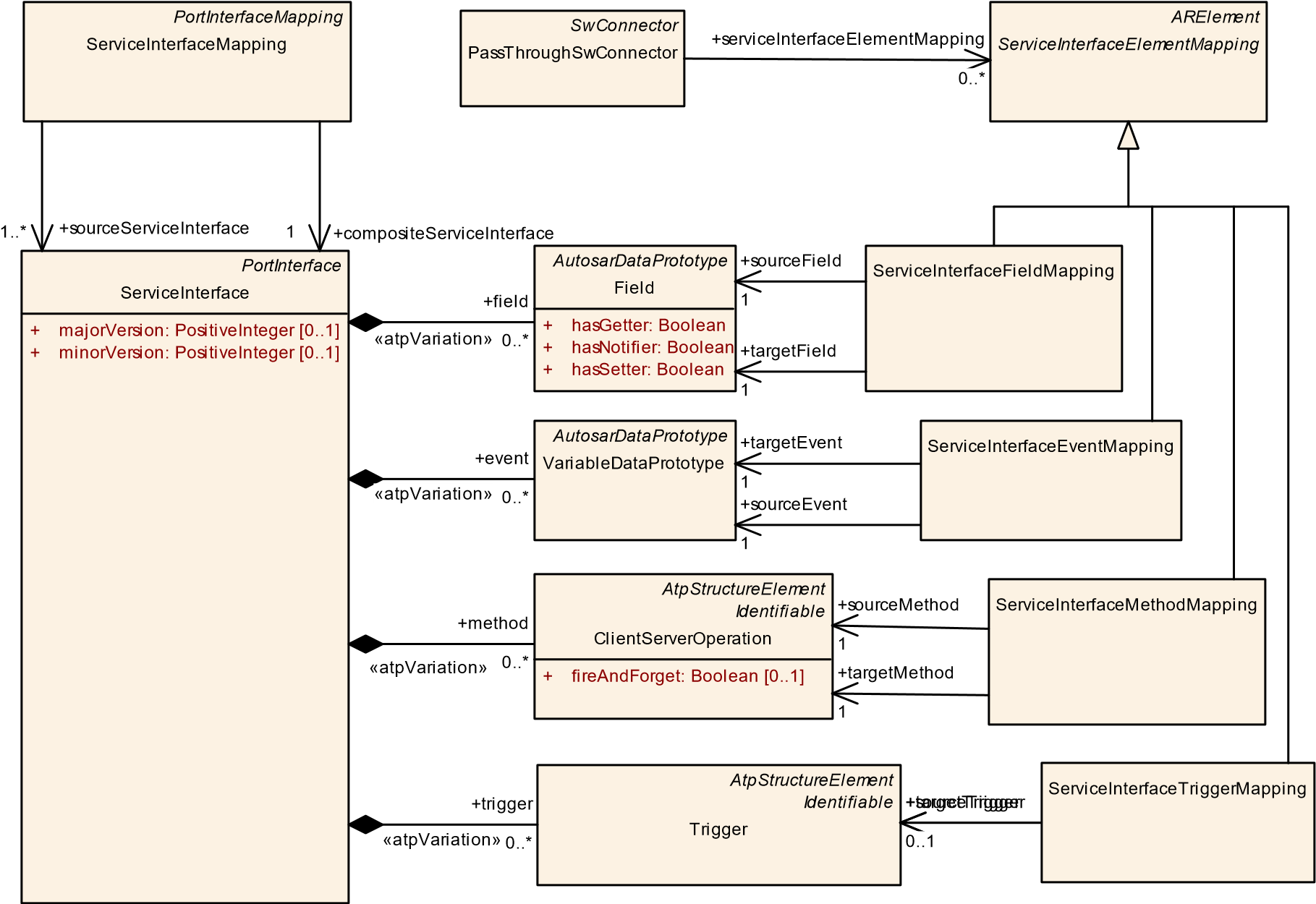
**[TPS\_MANI\_01003]** { DRAFT } ServiceInterfaceMapping的适用性**限制**d **ServiceInterfaceMapping**的适用性仅限于在compositeServiceInterface的上下文中complexServiceInterface的元素的shortName是**唯一的情况**。 c *( RS\_MANI\_00017 )*

请注意，在第3.4.6节的意义上， ServiceInterfaceMapping不是通信端之间的预先关联（通过SwConnector s） 。

如 [ TPS\_MANI\_01032 ] 中所述， ServiceInterfaceMapping允许派生外观软件组件或适当配置通信中间件。

sourceServiceInterface和compositeServiceInterface之间的兼容性是通过在外观软件组件或中间件配置中实现的适当转换来实现的。

因此，通过ServiceInterfaceMapping连接ServiceInterface （或其中的一部分）不受任何兼容性规则的限制，除了 [ TPS\_MANI\_01022 ] 中所述的规则。



**ServiceInterfaceMapping和ServiceInterfaceElementMapping建模概述**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **直通开关连接器** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Composition | | | |
| ***笔记*** | 这种 SwConnector 可以在 CompositionSwComponentType 内部使用，以连接两个委托 PortPrototype。 | | | |
| ***根据*** | *ARObject* , *AtpClassifier* , *AtpFeature* , *AtpStructureElement* , *Identifiable* , *MultilanguageReferable* , *Referrable* , *SwConnector* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 提供外端口 | 抽象提供端口  原型 | 0..1 | 参考 | 这表示提供的 PassThroughSwConnector 的外部委托端口原型。 |
| 需要外端口 | 抽象必需端口  原型 | 0..1 | 参考 | 这表示 PassThroughSwConnector 所需的外部委托端口原型。 |
| 服务接口  元素  映射 | 服务接口  元素映射 | \* | 参考 | 对 ServiceInterfaceElementMapping 的引用，指定了两个不同 ServiceInterfaces 的不等命名服务接口元素的映射，输入了 PassThroughSwConnector 引用的两个 PortPrototype。  **标签：** atp.Status=draft |

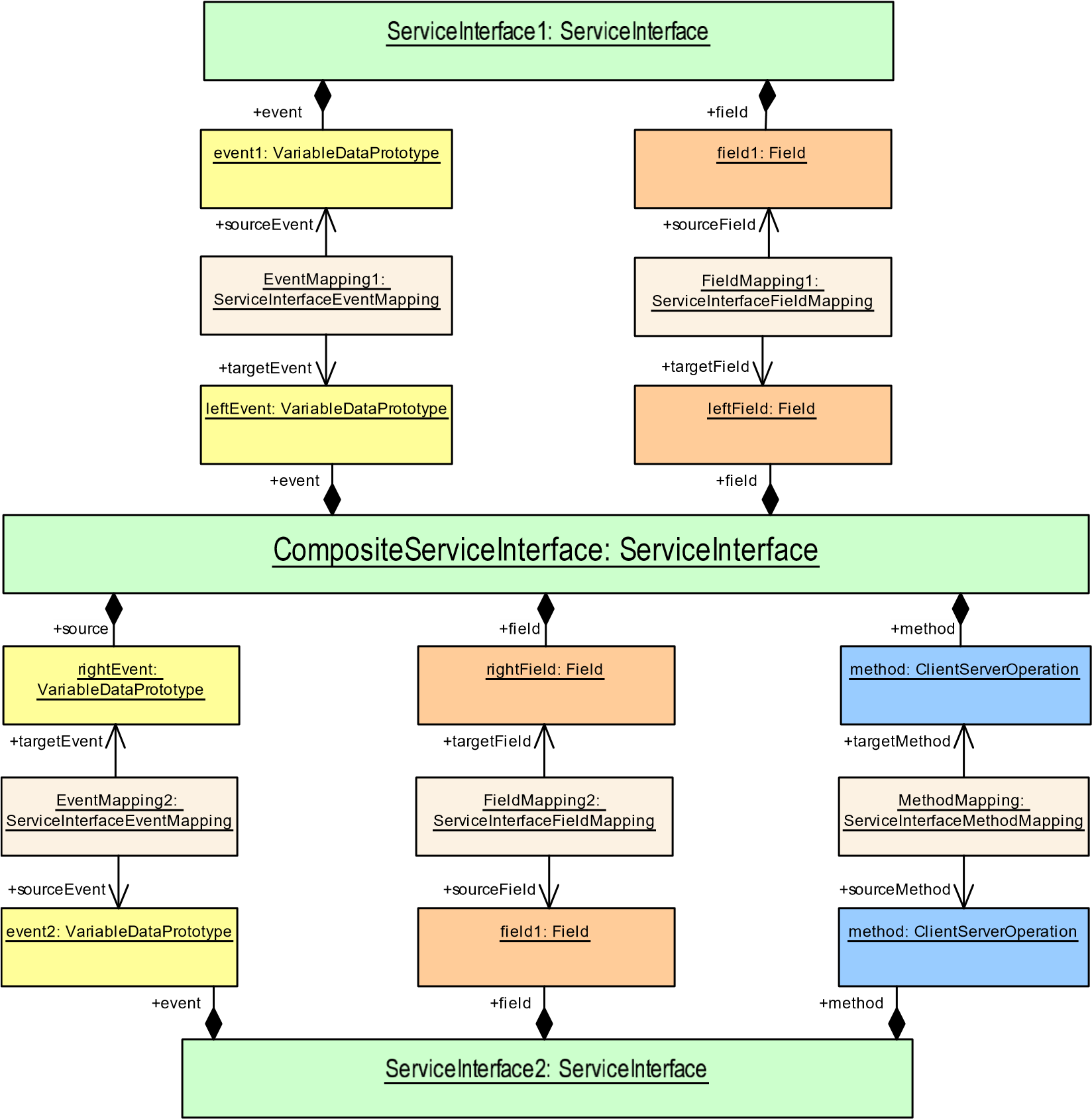
**表 3.38: PassThroughSwConnector**

## 3.6 服务接口元素映射

**3.6.1 概述**

ServiceInterfaceMapping的存在留下了关于如何映射元素具有不匹配shortName的ServiceInterface的问题。

这个问题的答案是通过创建同类元素的元素映射的能力来提供的。



**ServiceInterface元素的映射示例**

图3.26提供了一个基于元素的映射的示例。

请注意，在此示例中，ServiceInterface1 和 ServiceInterface2 都使用shortName field1聚合一个字段。

[ TPS\_MANI\_01003 ]开始，此配置取消了ServiceInterfaceMapping应用程序的场景资格。然而，逐元素映射能够很好地解决两个“源” ServiceInterface中存在的shortName字段1：

* ServiceInterface1.field1 映射到 CompositeServiceInterface.leftField
* ServiceInterface2.field1 映射到 CompositeServiceInterface.rightField

3.6节描述了各个映射的正式建模。

请注意，它**不打算**将 ServiceInterface 的映射与ServiceInterface的元素的映射混合。

换言之，只要两个ServiceInterface之间的映射存在，就不支持存在同一对ServiceInterface的元素之间的映射。这个重要的限制由 [ constr\_1482 ] 形式化。

**[constr\_1482]** { DRAFT }**服务接口的映射与服务接口元素的映射**d为了在给定的一对ServiceInterface之间建立映射，最多可以存在以下备选方案**之一：**

* 给定的一对ServiceInterface由ServiceInterfaceMapping引用，其中一个ServiceInterface在角色sourceServiceInterface中引用，另一个ServiceInterface在角色CompositeServiceInterface中引用。
* 存在以下选项的任意混合：
  + 由给定 ServiceInterface 之一聚合的事件由角色sourceEvent中的ServiceInterfaceEventMapping引用，由另一个给定ServiceInterface聚合的一个事件由角色targetEvent中的相同ServiceInterfaceEventMapping引用。
  + 由给定ServiceInterface之一聚合的触发器由角色sourceTrigger 中的ServiceInterfaceTriggerMapping 引用，由另一个给定 ServiceInterface聚合的一个触发器由角色 targetTrigger 中的相同ServiceInterfaceTriggerMapping引用。
  + 由给定 ServiceInterface 之一聚合的字段由角色sourceField中的ServiceInterfaceFieldMapping引用，由另一个给定ServiceInterface聚合的一个字段由角色targetField中的相同ServiceInterfaceFieldMapping引用。
  + 由给定ServiceInterface之一聚合的方法由角色sourceMethod中的ServiceInterfaceMethodMapping引用，并且由另一个给定ServiceInterface聚合的一个方法由角色targetMethod中的相同ServiceInterfaceMethodMapping引用。

### c ()

当然，相同的ServiceInterface可能被元素的映射和整个ServiceInterface的映射引用。 [ constr\_1482 ]中形式化的限制始终适用于一**对**ServiceInterface 。

ServiceInterface元素之间的映射通过抽象元类ServiceInterfaceElementMapping的子类建模。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***ServiceInterfaceElementMapping*** （抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping | | | |
| ***笔记*** | 此抽象元类充当映射 ServiceInterface 特定元素的基类。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***子类*** | 映射，服务接口字段映射，服务接口方法映射，服务  接口触发映射 | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.39：服务接口元素映射**

请注意， ServiceInterfaceElementMapping的创建被认为是一个原子步骤，即不太可能部分创建这样的ServiceInterfaceElementMapping ，将其移交给另一方，然后由该另一方完成。

毕竟，有相互排斥的方式来指定映射，并且ServiceInterface的部分映射的任何创建者都无法确定哪些替代方案适用于一个ServiceInterface与另一个 ServiceInterface 的特定配对，而无需知道另一个ServiceInterface （其中如果映射已经完成）。

因此，无需将对ServiceInterface元素的引用的较低多重性设置为 0。

**[TPS\_MANI\_03277]** { DRAFT } **ServiceInterfaceElementMapping用于单个ServiceInterface的元素子集**d如果源ServiceInterface的元素通过ServiceInterfaceElementMapping映射到目标ServiceInterface元素，则允许仅将这些源ServiceInterface元素的子集映射到目标服务接口元素。 c *( RS\_MANI\_00017 )*

使用 [ TPS\_MANI\_03277 ] 支持用例，如图 3.27所示，其中 AppM1\_2 提供的 ServiceInterface2 的 event4 和 method2 仅在 Machine1 上本地使用，并且同一ServiceInterface的 event3 与 ServiceInterface1 元素一起作为新的 SOME/IP 提供通过网络服务。

请注意，外观SwComponentType的建模不会对实现和这种外观功能的实现做出任何假设。

AUTOSAR 没有为实现此类功能定义任何指令，并且决定是特定于项目的。

这种“门面”的行为方面（例如，何时提供粗粒度的ProvidedApServiceInstance ）也是项目特定的，AUTOSAR 没有预定义。

AppM1\_1

P

AppM1\_2

P

AppM2\_1

R

Facade

R

R

P

Machine 1

Machine 2

SOME/IP Service with

Event1

Event2

Event3

Method1

Facade

R

P

AppM2\_2

R

P

AppM2\_3

R

Service 4 with

Event2

Service 3 with

Event1, Event3,

Method1

Service 1 with Event1,

Event2, Method1

Service 2 with Event3,

Event4, Method2

**图 3.27：提供者端单个ServiceInterface的元素子集的ServiceInterfaceElementMapping示例**

类似的用例适用于消费者端，如图 3.28所示，其中 Facade 组件是 SOME/IP 服务的服务消费者，具有 event1、event2 和 event3 和 method1，但软件仅使用 event1、event3 和 method1部署在 Machine2 上。

AppM11

P

AppM12

P

AppM22

R

Facade

R

R

P

Machine 1

Machine 2

SOME/IP Service with

Event1

Event2

Event3

Method1

Facade

R

P

Service 3 with

Event1, Event3,Method1

Service 2 with Event3,

Event4, Method2

Service 1 with

Event1, Event2,

Method1

**图 3.28：消费者端单个ServiceInterface的元素子集的ServiceInterfaceElementMapping示例**

**3.6.2 服务接口事件映射**

**[TPS\_MANI\_01024]** { DRAFT } **ServiceInterfaceEventMapping的语义**d元类ServiceInterfaceEventMapping具有映射ServiceInterface的能力。角色sourceEvent中引用的事件显式地指向另一个ServiceInterface 。角色targetEvent中引用的事件。 c *( RS\_MANI\_00017 )*

ServiceInterfaceEventMapping

*AutosarDataPrototype*

VariableDataPrototype

*ARElement*

*ServiceInterfaceElementMapping*

+

targetEvent

1

+

sourceEvent

1

**图 3.29： ServiceInterfaceEventMapping的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **服务接口事件映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping | | | |
| ***笔记*** | 此元类允许定义通过 ServiceInterfaceMapping 相互映射的 ServiceInterfaces 事件之间的映射。  **标签：**  atp.Status=草稿  atp.recommendedPackage=ServiceInterfaceElementMappings | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable*  *元素*,*可引用*, *ServiceInterfaceElementMapping* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 源事件 | 变量数据原型 | 1 | 参考 | 对源 ServiceInterface 中包含的事件的引用。  **标签：** atp.Status=draft |
| 目标事件 | 变量数据原型 | 1 | 参考 | 对包含在复合 ServiceInterface 中的事件的引用。  **标签：** atp.Status=draft |

**表 3.40：服务接口事件映射**

ServiceInterfaceEventMapping实现的显式映射不需要映射两边的**shortName**相等。

还可以将给定ServiceInterface的给定事件以不同角色多次映射到聚合targetEvent的ServiceInterface ，如图3.30所示。

请注意，一个sourceEvent到不同targetEvent的映射并不代表任何类型的**扇***出*。

这仅意味着sourceEvent将用于不同的角色，如部署中指定的那样。如需更多解释，请在图A.5中找到ServiceInterface元素的基于角色的映射如何工作的示例。

ServiceInterface1:

ServiceInterface

event1:

VariableDataPrototype

event11:

VariableDataPrototype

CompositeServiceInterface:

ServiceInterface

event10:

VariableDataPrototype

ServiceInterfaceEventMapping1:

ServiceInterfaceEventMapping

ServiceInterfaceEventMapping2:

ServiceInterfaceEventMapping

+

event

targetEvent

+

+

event

+

sourceEvent

targetEvent

+

+

sourceEvent

event

+

**图 3.30： ServiceInterfaceEventMapping应用示例**

**3.6.3 服务接口触发器映射**

**[TPS\_MANI\_03289]** { DRAFT } **ServiceInterfaceTriggerMapping的语义**d元类ServiceInterfaceTriggerMapping具有映射ServiceInterface的能力。角色sourceTrigger中引用的触发器显式地指向另一个ServiceInterface 。角色targetTrigger中引用的触发器。 c *( RS\_MANI\_00017 )*

*AtpStructureElement*

*Identifiable*

Trigger

ServiceInterfaceTriggerMapping

*ARElement*

*ServiceInterfaceElementMapping*

+

sourceTrigger

0..1

+

targetTrigger

0..1

**图 3.31： ServiceInterfaceTriggerMapping的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **服务接口触发器映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping | | | |
| ***笔记*** | 此元类允许定义通过 ServiceInterfaceMapping 相互映射的 ServiceInterfaces 触发器之间的映射。  **标签：**  atp.Status=草稿  atp.recommendedPackage=ServiceInterfaceElementMappings | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable*  *元素*,*可引用*, *ServiceInterfaceElementMapping* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 源触发器 | 扳机 | 0..1 | 参考 | 对源 ServiceInterface 中包含的触发器的引用。  **标签：** atp.Status=draft |
| 目标触发器 | 扳机 | 0..1 | 参考 | 对包含在目标 ServiceInterface 中的触发器的引用。  **标签：** atp.Status=draft |

**表 3.41：ServiceInterfaceTriggerMapping**

ServiceInterfaceTriggerMapping实现的显式映射不需要映射两边的**shortName**相等。

也可以将给定ServiceInterface的给定触发器以不同角色多次映射到聚合targetTrigger的ServiceInterface 。

请注意，一个sourceTrigger到不同targetTrigger的映射并不代表任何类型的**扇***出*。

这仅意味着sourceTrigger将用于不同的角色，如部署中指定的那样。

**3.6.4 服务接口字段映射**

**[TPS\_MANI\_01025]** { DRAFT } **ServiceInterfaceFieldMapping的语义**d元类ServiceInterfaceFieldMapping具有映射ServiceInterface的能力。角色sourceField中明确引用到另一个ServiceInterface的字段。角色targetField中引用的字段。 c *( RS\_MANI\_00017 )*

ServiceInterfaceFieldMapping

*AutosarDataPrototype*

Field

+

hasGetter: Boolean

+

hasNotifier: Boolean

+

hasSetter: Boolean

*ARElement*

*ServiceInterfaceElementMapping*

+

sourceField

1

+

targetField

1

**图 3.32： ServiceInterfaceFieldMapping的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **服务接口字段映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping | | | |
| ***笔记*** | 此元类允许定义通过 ServiceInterfaceMapping 相互映射的 ServiceInterfaces 字段之间的映射。  **标签：**  atp.Status=草稿  atp.recommendedPackage=ServiceInterfaceElementMappings | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable*  *元素*,*可引用*, *ServiceInterfaceElementMapping* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 源字段 | 场地 | 1 | 参考 | 对源 ServiceInterface 中包含的字段的引用。  **标签：** atp.Status=draft |
| 目标字段 | 场地 | 1 | 参考 | 对包含在复合 ServiceInterface 中的字段的引用。  **标签：** atp.Status=draft |

**表 3.42：ServiceInterfaceFieldMapping**

ServiceInterfaceFieldMapping实现的显式映射不需要映射两边的**shortName**相等。

也可以将给定ServiceInterface的给定字段以不同角色多次映射到聚合targetField的ServiceInterface ，如图3.33所示。

ServiceInterface1:

ServiceInterface

CompositeServiceInterface:

ServiceInterface

field1: Field

field10: Field

field11: Field

ServiceInterfaceFieldMapping1:

ServiceInterfaceFieldMapping

ServiceInterfaceFieldMapping2:

ServiceInterfaceFieldMapping

+

field

sourceField

+

+

targetField

targetField

+

+

sourceField

+

field

+

field

**图 3.33： ServiceInterfaceFieldMapping应用示例**

请注意，一个sourceField到不同targetField的映射并不代表任何类型的**扇***出*。

这仅意味着sourceField将用于不同的角色，如部署中指定的那样。如需更多解释，请在图A.5中找到ServiceInterface元素的基于角色的映射如何工作的示例。

**3.6.5 服务接口方法映射**

**[TPS\_MANI\_01026]** { DRAFT } **ServiceInterfaceMethodMapping的语义**d元类ServiceInterfaceMethodMapping具有映射ServiceInterface的能力。角色sourceMethod中明确引用到另一个ServiceInterface的方法。角色targetMethod中引用的方法。 c *( RS\_MANI\_-*

*00017 )*

ServiceInterfaceMethodMapping

*AtpStructureElement*

*Identifiable*

ClientServerOperation

+

fireAndForget: Boolean

[0..1]

*ARElement*

*ServiceInterfaceElementMapping*

+

sourceMethod

1

+

targetMethod

1

**图 3.34： ServiceInterfaceMethodMapping的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **服务接口方法映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping | | | |
| ***笔记*** | 此元类允许定义通过 ServiceInterfaceMapping 相互映射的 ServiceInterfaces 方法之间的映射。  **标签：**  atp.Status=草稿  atp.recommendedPackage=ServiceInterfaceElementMappings | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable*  *元素*,*可引用*, *ServiceInterfaceElementMapping* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 源方法 | 客户端服务器操作 | 1 | 参考 | 对源 ServiceInterface 中包含的方法的引用。  **标签：** atp.Status=draft |
| 目标方法 | 客户端服务器操作 | 1 | 参考 | 对包含在复合 ServiceInterface 中的方法的引用。  **标签：** atp.Status=draft |

**表 3.43：服务接口方法映射**

ServiceInterfaceMethodMapping实现的显式映射不需要映射两边的**shortName**相等。

也可以将给定ServiceInterface的给定方法以不同角色多次映射到聚合targetMethod的ServiceInterface ，如图3.35所示。

ServiceInterface1:

ServiceInterface

CompositeServiceInterface:

ServiceInterface

ServicInterfaceMethodMapping1:

ServiceInterfaceMethodMapping

ServicInterfaceMethodMapping2:

ServiceInterfaceMethodMapping

method1:

ClientServerOperation

method11:

ClientServerOperation

method10:

ClientServerOperation

+

method

+

method

sourceMethod

+

+

targetMethod

sourceMethod

+

targetMethod

+

+

method

**图 3.35： ServiceInterfaceMethodMapping应用示例**

请注意，一个sourceMethod到不同targetMethod的映射并不代表任何类型的**扇***出*。

这仅意味着sourceMethod将用于不同的角色，如部署中指定的那样。如需更多解释，请在图A.5中找到ServiceInterface元素的基于角色的映射如何工作的示例。

## 3.7 持久化接口

**3.7.1 概述**

**3.7.1.1 大局**

*AUTOSAR 自适应平台*预计支持通过例如应用软件访问持久数据。

ara::com

ara::per

ServiceInstance

Mapping

Mapping

Manifest

Design

Key

-

Value

Storage

(

Configuration on the target hard

-

and software)

Access to individual elements

Software

Component

Access to entire

Key

-

value storage

**图 3.36：持久性建模的一般方法**

PortPrototype的使用方面与通信模型有一些相似之处。

PortPrototype级别上访问整个键值存储）。

3.36解释了持久数据配置的部署方面。

为访问持久数据（即PersistencyKeyValueStorageInterface ）和访问文件系统上的文件的不同用例定义了两个单独的元类（更多详细信息，请参见图3.37 ），或者可能是模拟一个（通过PersistencyFileStorageInterface ）。

**3.7.1.2 持久性接口建模**

抽象元类PersistencyInterface已被创建为一种分类方式，即它允许轻松引用PortInterface通常专用于持久性。

PersistencyInterface的对应物，元类PersistencyInterfaceElement已被定义为PersistencyInterface元素的抽象基类。

*PersistencyInterface*

[0..1]

minimumSustainedSize: PositiveInteger

+

+

redundancy: PersistencyRedundancyEnum

[0..1]

[0..1]

updateStrategy: PersistencyCollectionLevelUpdateStrategyEnum

+

PersistencyKeyValueStorageInterface

PersistencyFileStorageInterface

maxNumberOfFiles: PositiveInteger

+

[0..1]

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

PersistencyDataElement

«enumeration»

PersistencyElementLevelUpdateStrategyEnum

overwrite

keepExisting

delete

*DataPrototype*

*AutosarDataPrototype*

*ARElement*

*AtpType*

*AutosarDataType*

«enumeration»

PersistencyCollectionLevelUpdateStrategyEnum

keepExisting

delete

PersistencyFileElement

contentUri: UriString

+

fileName: String

+

*AtpBlueprint*

*AtpBlueprintable*

*AbstractImplementationDataType*

«enumeration»

PersistencyRedundancyEnum

redundant

none

redundantPerElement

*Identifiable*

*PersistencyInterfaceElement*

updateStrategy: PersistencyElementLevelUpdateStrategyEnum

+

[0..1]

+

dataElement

0..\*

fileElement

+

0..\*

dataTypeForSerialization

+

0..\*

«isOfType»

+

type

0..1

{

}

redefines atpType

**图 3.37：持久性用例的PortInterface规范**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***PersistencyInterfaceElement*** （抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 此元类提供了定义 PortInterface 元素以支持持久性用例的抽象能力。**标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***子类*** | 持久性数据元素，持久性文件元素 | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***PersistencyInterfaceElement*** （抽象） | | |  |
| 更新策略 | 持久性元素  级别更新策略  枚举 | 0..1 | 属性 | 该属性可用于指定各个 PersistencyInterfaceElement 的更新策略。  **标签：** atp.Status=draft |

**表 3.44：PersistencyInterfaceElement**

**[TPS\_MANI\_01194]** {草稿} **PersistencyInterface的语义。最低限度-**

**SustainedSize** d属性PersistencyInterface 。 minimumSustainedSize可用于从应用程序设计者的角度定义PersistencyInterface需要分配的最小存储量。 c *( RS\_MANI\_00027 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***持久性接口***（抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 此元类提供了定义 PortInterface 以支持持久性用例的抽象能力。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *识别*，*多语言可引用*，*可打包元素*，端口接口，*可引用* | | | |
| ***子类*** | PersistencyFileStorageInterface , PersistencyKeyValueStorageInterface | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 最低限度  持续尺寸 | 正整数 | 0..1 | 属性 | 此属性的值表示封闭持久性接口在设计时所需的最小大小。  **标签：** atp.Status=draft |
| 冗余 | 持久性冗余  枚举 | 0..1 | 属性 | 此属性表示对存储冗余的要求。**标签：** atp.Status=draft |
| 冗余处理 | 持久性冗余  处理 | \* | 聚合 | 这种聚合代表了为子类实现的各种用例处理冗余的选择方法  **标签：** atp.Status=draft |
| 更新策略 | 持久性集合  级别更新策略  枚举 | 0..1 | 属性 | 该属性可用于指定各个 PersistencyInterface 整体的更新策略。  **标签：** atp.Status=draft |

**表 3.45：持久性接口**

**3.7.1.3 冗余处理**

**[TPS\_MANI\_01204]** { DRAFT }**持久数据冗余规范**d属性PersistencyInterface 。从软件组件的设计者的角度来看，冗余可以用来指定相应的键值存储或文件存储是否应该冗余存储数据。 c *( RS\_MANI\_00027 )*

细节留给集成商，他们也可能决定否决PersistencyInterface的值。如果有用例，则完全冗余。

|  |  |
| --- | --- |
| ***枚举*** | **持久性冗余枚举** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec |
| ***笔记*** | 这个元类提供了一种方法来指定在集合级别上应用冗余的方式。  **标签：** atp.Status=draft |
| ***文字*** | ***描述*** |
| 没有任何 | 该值表示冗余措施不适用于持久性存储级别的要求。  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |
| 多余的 | 此值表示在持久性存储级别上应用冗余措施的要求。  冗余持久存储的性质没有进一步限定，并受集成商决定的影响。  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |
| 冗余每元素 | 该值表示在键值存储的键值级别或文件存储的文件级别上应用冗余措施的要求。  持久存储上使用的冗余的性质没有进一步限定，并且受集成商决定的影响。  **标签：**  atp.EnumerationLiteralIndex=2 atp.Status=草稿 |

**表 3.46：PersistencyRedundancyEnum**

**[TPS\_MANI\_01319]** { DRAFT }在**PersistencyInterface** d**上下文中建模冗余作为使用**PersistencyInterface能力的替代方案。冗余为了宣布完全考虑冗余，持久性的设计级别还提供了通过在PersistencyInterface聚合PersistencyRedundancyHandling为键值存储和文件提供更详细的冗余行为定义的能力。

该建模附加到抽象基类PersistencyInterface以便让 AUTOSAR 自适应平台上的持久性的两个方面（即键值存储和文件存储）受益于元类PersistencyRedundancyHandling的存在。 c *( RS\_MANI\_00027 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***持久性冗余处理***（抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***笔记*** | 这个抽象基类代表了冗余的正式描述。  **标签：** atp.Status=draft | | | |
| ***根据*** | *AR对象* | | | |
| ***子类*** | *持久性*冗余校验和 | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***持久性冗余处理***（抽象） | | | |
| 范围 | 持久性冗余  处理范围枚举 | 0..1 | 属性 | 此属性控制应用冗余处理的范围。  **标签：** atp.Status=draft |

**表 3.47: PersistencyRedundancyHandling**

**[TPS\_MANI\_01320]** { DRAFT }**接口级别的冗余定义可能在部署中被否决**d通过PersistencyInterface对冗余建模。冗余处理代表PersistencyInterface设计者的意图。

虽然这肯定是部署阶段的宝贵输入，但可以明确预见，集成商可能会根据仅在部署时可用的高级知识否决有关持久性的设计决策。 c *( RS\_MANI\_00027 )*

*PortInterface*

*PersistencyInterface*

+

[0..1]

minimumSustainedSize: PositiveInteger

redundancy: PersistencyRedundancyEnum

+

[0..1]

[0..1]

updateStrategy: PersistencyCollectionLevelUpdateStrategyEnum

+

*PersistencyRedundancyHandling*

[0..1]

scope: PersistencyRedundancyHandlingScopeEnum

+

PersistencyRedundancyCrc

PersistencyRedundancyMOutOfN

m: PositiveInteger

+

n: PositiveInteger

+

*PersistencyRedundancyChecksum*

algorithmFamily: String

+

length: PositiveInteger

+

PersistencyRedundancyHash

initializationVectorLength: PositiveInteger

[0..1]

+

redundancyHandling

+

0..\*

**PersistencyInterface级别的冗余规范**

**[constr\_1746]** { DRAFT } **PersistencyInterface互斥存在。冗余和PersistencyInterface 。冗余处理**d对于每个PersistencyInterface ，属性冗余或聚合

PersistencyRedundancyHandling中角色redundancyHandling可能存在。 C

### ()

|  |  |
| --- | --- |
| ***枚举*** | **持久性冗余处理范围枚举** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency |

5

4

|  |  |
| --- | --- |
| ***枚举*** | **持久性冗余处理范围枚举** |
| ***笔记*** | 此元类提供值来控制持久性部署中冗余措施的范围  **标签：** atp.Status=draft |
| ***文字*** | ***描述*** |
| 持久性  冗余  处理范围  元素 | 冗余处理应应用于元素级别（键值对和文件）。  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |
| 持久性  冗余  处理范围  贮存 | 冗余处理应应用于存储（键值存储和文件存储）级别。  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |

**表 3.48：PersistencyRedundancyHandlingScopeEnum**

**[TPS\_MANI\_01207]** { DRAFT }属性**PersistencyRedundancyChecksum的标准化值。 algorithmFamily** d属性PersistencyRedundancyChecksum的以下值。 algorithmFamily由 AUTOSAR 标准化：

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

c *( RS\_MANI\_00027 )*

**[constr\_1668]** { DRAFT }**允许的PersistencyRedundancyChecksum组合。 length和algorithmFamily** d PersistencyRedundancyChecksum的允许组合。 length和algorithmFamily记录在

表3.49 。 c *()*

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

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CRC\_32P4** |  |  | X |  |

**表 3.49：允许的PersistencyRedundancy-组合**

**校验和。长度和算法系列**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***PersistencyRedundancyChecksum*** （抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***笔记*** | 定义冗余实现的公共属性的抽象类。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *PersistencyRedundancyHandling* | | | |
| ***子类*** | PersistencyRedundancyCrc , PersistencyRedundancyHash | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 算法家族 | 细绳 | 1 | 属性 | 此属性标识用于执行 CRC/Hash 的算法系列。  **标签：** atp.Status=draft |
| 长度 | 正整数 | 1 | 属性 | 该属性以单位比特描述 CRC/Hash 的长度。  **标签：** atp.Status=draft |

**表 3.50：PersistencyRedundancyChecksum**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **持久性冗余Crc** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***笔记*** | 这个元类正式描述了使用 CRC 来实现冗余。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *PersistencyRedundancyChecksum* , *PersistencyRedundancyHandling* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.51：PersistencyRedundancyCrc**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **持久性冗余哈希** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***笔记*** | 这个元类正式描述了使用 Hash 来实现冗余。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *PersistencyRedundancyChecksum* , *PersistencyRedundancyHandling* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 初始化  矢量长度 | 正整数 | 0..1 | 属性 | 初始化向量的长度。  **标签：** atp.Status=draft |

**表 3.52: PersistencyRedundancyHash**

**[constr\_10046]** { DRAFT } **PersistencyRedundancyMOutOfN的值。 n** d PersistencyRedundancyMOutOfN的值。 n应至少设置为 2，最多设置为 255，即允许的间隔为 [2..255]。 c *()*

**[constr\_1751]** { DRAFT } **PersistencyRedundancyMOutOfN的值。 m** d属性PersistencyRedundancyMOutOfN的值。 m应至少设置为 1，最多设置为属性PersistencyRedundancyMOutOfN的值。 n ，即允许的间隔是 [1.. PersistencyRedundancyMOutOfN 。 n ]。 c *()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **持久性冗余MOutOfN** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***笔记*** | 这个元类提供了通过“M out of N”方法来描述冗余的能力。在这种情况下，N 是创建的副本数，M 是最小的相同副本数，以证明对数据的可靠读取访问是合理的。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *PersistencyRedundancyHandling* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 米 | 正整数 | 1 | 属性 | 该属性表示“M out of N”方案中的“M”坐标。  **标签：** atp.Status=draft |
| n | 正整数 | 1 | 属性 | 该属性表示“M out of N”方案中的“N”坐标。  **标签：** atp.Status=draft |

**表 3.53：PersistencyRedundancyMOutOfN**

**3.7.1.4 更新处理**

**[TPS\_MANI\_01139]** {草案} **PersistencyInterface的语义。 updateStrategy** d属性PersistencyInterface 。 updateStrategy可用于指定更新在对应于PersistencyInterface的PersistencyDeployment上下文中使用的实际持久元素的策略。

定义自己的updateStrategy的显式建模PersistencyInterfaceElement之外，此更新策略应作为一个整体应用于PersistencyInterface 。 c *( RS\_MANI\_00027 )*

|  |  |
| --- | --- |
| ***枚举*** | **PersistencyCollectionLevelUpdateStrategyEnum** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |
| ***笔记*** | 此枚举为接口/存储级别的更新策略提供可能的值。  **标签：** atp.Status=draft |
| ***文字*** | ***描述*** |
| 删除 | 更新策略是删除相应集合级别上的所有值。  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |
| 保持现有 | 更新策略是将现有值保持在相应集合的级别。  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |

**表 3.54：PersistencyCollectionLevelUpdateStrategyEnum**

**[TPS\_MANI\_01140]** {草稿} **PersistencyInterfaceElement的语义。 updateStrategy** d属性PersistencyInterfaceElement 。 updateStrategy可用于指定更新与PersistencyInterfaceElement对应的实际持久元素的策略。 c *( RS\_MANI\_00027 )*

|  |  |
| --- | --- |
| ***枚举*** | **PersistencyElementLevelUpdateStrategyEnum** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |
| ***笔记*** | 此枚举为元素级别的更新策略提供可能的值。  **标签：** atp.Status=draft |
| ***文字*** | ***描述*** |
| 删除 | 更新策略是删除相应数据项的值。  **标签：**  atp.EnumerationLiteralIndex=2 atp.Status=草稿 |
| 保持现有 | 更新策略是保持各个数据项的现有值。  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |
| 覆盖 | 更新策略是覆盖相应的数据项。  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |

**表 3.55：PersistencyElementLevelUpdateStrategyEnum**

9 ]中详细解释了软件在应用更新策略方面的行为。

**3.7.2 持久化键值存储接口**

**[TPS\_MANI\_01065]** { DRAFT } PersistencyKeyValueStorageInterface的**用途**d **PersistencyKeyValueStorageInterface**的用途是支持对键值存储中数据的持久访问。 c *( RS\_MANI\_00027 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **持久性键值存储接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 此元类提供了实现 PortInterface 以支持数据持久性用例的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=PersistencyKeyValueStorageInterfaces | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *识别*的、*多语言可引用的*、 *PackageableElement* 、 *PersistencyInterface* 、 *PortInterface* 、  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 数据元素 | 持久性数据  元素 | \* | 聚合 | 此聚合表示封闭 Persistency KeyValueStorageInterface 上下文中的 Persistency DataElements 的集合。  **标签：** atp.Status=draft |

5

**表 3.56：PersistencyKeyValueStorageInterface**

**[TPS\_MANI\_01135]** { DRAFT } **PersistencyKeyValueStorageInterface的语义。 dataTypeForSerialization** d引用PersistencyKeyValueStorageInterface 。 dataTypeForSerialization可用于获取有关必须为其生成序列化算法的数据类型的信息，以支持此类数据类型对象的持久存储。 c *( RS\_MANI\_00027 )*

与其他类型的PortInterface相比，它**不需要**定义PersistencyKeyValueStorageInterface的元素。但是，如果这是有意的，则聚合PersistencyKeyValueStorageInterface 。 dataElement应用于此目的。

**[TPS\_MANI\_01138]** { DRAFT } **PersistencyKeyValueStorageInterface的语义。 dataElement** d通过在角色dataElement中聚合PersistencyDataElement ，可以显式地为应用软件在PersistencyKeyValueStorageInterface的上下文中可访问的键值对（及其一些属性）建模。 c *( RS\_MANI\_00027 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **持久性键值存储接口** | | |  |
| dataType用于序列化 | 抽象实现  数据类型 | \* | 参考 | 除了已经从 PersistencyDataElement 的聚合中确定的类型之外，该参考还标识了应支持存储在键值存储中的 AbstractImplementationData 类型。  **标签：** atp.Status=draft |

**[TPS\_MANI\_01180]** { DRAFT }**需要序列化支持**的数据类型集合d需要序列化支持的数据类型集合包括

* 角色PersistencyKeyValueStorageInterface中引用的AbstractImplementationDataType 。 dataTypeForSerialization
* 任何一个
  + AbstractImplementationDataType用于键入PersistencyKeyValueStorageInterface 。数据元素或
  + AbstractImplementationDataType映射到ApplicationDataType以键入PersistencyKeyValueStorageInterface 。 dataElement通过PortInterfaceToDataTypeMapping 。 dataTypeMappingSet也引用封闭的PersistencyKeyValueStorageInterface 。

c *( RS\_MANI\_00027 )*

|  |  |
| --- | --- |
| ***班级*** | **持久性数据元素** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **持久性数据元素** | | | |
| ***笔记*** | 这个元类代表了在封闭的 PersistencyKeyValueStorageInterface 的上下文中正式指定一段受持久性影响的数据的能力。  PersistencyDataElement 还表示部署的 PersistencyKeyValueStorage 的键值对并提供初始值。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *AtpFeature* , *AtpPrototype* , *AutosarDataPrototype* , *DataPrototype* , *Identifiable* , *Multilanguage Referrable* , *PersistencyInterfaceElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.57：持久性数据元素**

请注意， PersistencyDataElement可以由ApplicationDataType或CppImplementationDataType 键入。

**3.7.3 持久化文件存储接口**

**[TPS\_MANI\_01067]** { DRAFT } **PersistencyFileStorageInterface的目的**d元类PersistencyFileStorageInterface的目的是支持对文件存储的抽象表示的访问。 c *( RS\_MANI\_00027 )*

就 AUTOSAR 持久性而言，文件可以具有二进制或文本内容。如果它有文本内容，那么文件的内容应该被编码为带有 UNIX 行结尾的 UTF-8 编码。

**[TPS\_MANI\_01068]** { DRAFT } **PersistencyFileStorageInterface的语义。 maxNumberOfFiles** d由PersistencyFileStorageInterface键入的任何PortPrototype都可以访问多个文件。

PersistencyFileStorageInterface键入的给定PortPrototype表示的文件数的上限可以使用属性PersistencyFileStorageInterface进行配置。最大文件数。

属性PersistencyFileStorageInterface的值。 maxNumberOfFiles**包括**显式建模的PersistencyFileStorageInterface 。文件元素s。 c *( RS\_MANI\_00027 )*

请注意， PersistencyFileStorageInterface的存在不违反 IEEE1003.13 中定义的 POSIX 子集 PSE51 设置的限制

[ 10 ]。

|  |  |
| --- | --- |
| ***班级*** | **持久性文件存储接口** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **持久性文件存储接口** | | | |
| ***笔记*** | 此元类提供了实现 PortInterface 以支持文件持久性用例的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=PersistencyFileStorageInterfaces | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *识别*的、*多语言可引用的*、 *PackageableElement* 、 *PersistencyInterface* 、 *PortInterface* 、  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 文件元素 | 持久性文件元素 | \* | 聚合 | 此聚合表示封闭 Persistency FileStorageInterface 上下文中的 Persistency FileStorage 集合。  **标签：** atp.Status=draft |
| 最大数量  文件 | 正整数 | 0..1 | 属性 | 此属性表示在封闭 PersistencyFileStorageInterface 的上下文中在运行时处理文件的上限的定义。  **标签：** atp.Status=draft |

**表 3.58: PersistencyFileStorageInterface**

由PersistencyFileStorageInterface类型化的PortPrototype允许将对操作系统的实际调用从应用软件的范围抽象到*AUTOSAR 自适应平台的模块中*。

**[TPS\_MANI\_01142]** { DRAFT } **PersistencyFileElement的语义**d通过在角色fileElement中聚合PersistencyFileElement ，可以在PersistencyFileStorageInterface的上下文中显式地对应用软件可访问的文件（及其一些属性）进行建模。 c *( RS\_MANI\_00027 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **持久性文件元素** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 该元类具有在设计时表示文件的能力，以便可以配置在运行时访问表示的文件的行为。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *PersistencyInterfaceElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| contentUri | 字符串 | 1 | 属性 | 此属性表示标识 PersistencyFile 的初始内容的 URI。  **标签：** atp.Status=draft |
| 文件名 | 细绳 | 1 | 属性 | 该属性保存 PersistencyFileProxy 存储位置的文件名部分，例如文件系统上的文件。  **标签：** atp.Status=draft |

**表 3.59：PersistencyFileElement**

**[constr\_1581]** { DRAFT } **fileElement的值。 fileName** d在任何给定PersistencyFileStorageInterface范围内，所有fileElement的值。

文件名应该是唯一的。 c *()*

**[constr\_10080]** { DRAFT } **PersistencyFileElement** d**初始值的存在**对于每个PersistencyFileElement ，如果属性updateStrategy的值设置为值delete ，则属性PersistencyFileElement 。 contentUri不应存在。 c *()*

## 3.8 时间同步接口

自适应平台内的时间同步功能集群负责提供各种时基资源供应用程序读取或写入。

为了与时间同步基础软件交互，应用程序开发人员需要声明此应用程序将与哪种类型的时基资源交互。

时间同步的接口遵循PortPrototype和PortInterface的通用模式，这些模式适用于有关应用软件与平台软件交互的许多用例。

与基于服务的通信相比，使用PortPrototype和PortInterface的平台软件交互建模不太详细。 PortPrototype是用于与平台软件交互的占位符，它不模拟可用于交互的实际使用的 API。要使用的 API 在平台软件 SWS 文档中正式指定，即 SWS\_TimeSync [ 11 ]。

**[TPS\_MANI\_03535]** { DRAFT }**时间同步交互的定义**d元类AbstractSynchronizedTimeBaseInterface及其子类用于定义应用软件与时间同步时基的交互。 c *( RS\_MANI\_00040 )*

更多信息，请参见图3.39 。

通过定义由AbstractSynchronizedTimeBaseInterface子类之一键入的RPortPrototype ，应用程序指示它将访问特定的时基。

**[TPS\_MANI\_03549] { DRAFT } PortPrototype**用于与时间**同步交互的使用**d根据用例，由AbstractSynchronizedTimeBaseInterface的子类之一类型化的RPortPrototype或PPortPrototype的使用应用于与时间同步的交互。 c *( RS\_MANI\_00040 )*

应用软件在与功能集群的交互中可能扮演主动或被动的角色，因此从应用软件的角度来看，应该使用RPortPrototype或PPortPrototype来表示这种交互。 TimeBase 资源实例使用相应PortPrototype的 InstanceSpecifier 来标识。

**[TPS\_MANI\_03536]** { DRAFT }**提供者角色中的时间同步交互**d元类SynchronizedTimeBaseProviderInterface用于指示与*提供者*角色中的同步全局时基的预期交互。 c *( RS\_-*

*MANI\_00040 )*

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*AbstractSynchronizedTimeBaseInterface*

SynchronizedTimeBaseProviderInterface

+

timeBaseKind: TimeSynchronizationKindEnum

SynchronizedTimeBaseConsumerInterface

«enumeration»

TimeSynchronizationKindEnum

offset

synchronized

**图 3.39：时间同步接口的建模**

当以*提供者*角色与同步的全局时基进行交互时，应用程序能够*设置*（和*获取*）同步的全局时基的值，然后将其传播到网络上的时间值。

**[TPS\_MANI\_03537]** { DRAFT }**消费者角色中的时间同步交互**d元类SynchronizedTimeBaseConsumerInterface用于指示与*消费者*角色中的同步全局时基的预期交互。 c *( RS\_MANI\_00040 )*

当以*消费者*角色与同步的全局时基进行交互时，应用程序只能*获取*同步的全局时基的值，该值是从来自网络的时间值同步的。

**[TPS\_MANI\_03551]** { DRAFT }**时基类型的定义**d属性SynchronizedTimeBaseProviderInterface 。 timeBaseKind定义时基是同步时基还是偏移时基。 c *( RS\_MANI\_00040 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **SynchronizedTimeBaseProvider接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 这个元类提供了为与时间同步提供者交互定义端口接口的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=时间同步接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AbstractSynchronizedTimeBaseInterface* , *AtpBlueprint* , *AtpBlueprintable* , *Atp*  *分类器*, *AtpType* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* ,  *端口接口*,*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **SynchronizedTimeBaseProvider接口** | | |  |
| 时基种类 | 时间同步  种类枚举 | 1 | 属性 | 定义在此接口上请求哪种时基。  **标签：** atp.Status=draft |

**表 3.60：SynchronizedTimeBaseProviderInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **SynchronizedTimeBaseConsumerInterface** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 这个元类提供了为与时间同步消费者交互定义 PortInterface 的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=时间同步接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AbstractSynchronizedTimeBaseInterface* , *AtpBlueprint* , *AtpBlueprintable* , *Atp*  *分类器*, *AtpType* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* ,  *端口接口*,*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.61: SynchronizedTimeBaseConsumerInterface**

|  |  |
| --- | --- |
| ***枚举*** | **TimeSynchronizationKindEnum** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |
| ***笔记*** | 定义可能的 TimeSynchronizationInterfaces 类型。  **标签：** atp.Status=draft |
| ***文字*** | ***描述*** |
| 抵消 | 定义请求的时基应基于偏移时间。  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |
| 同步的 | 定义请求的时基应是基于同步时间的。  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |

**表 3.62：TimeSynchronizationKindEnum**

在图3.40的示例中，说明了一个应用程序与多个时间同步方面的交互。

Application

1

tb

P

tb

C1

SynchronizedTimeBase

ProviderInterface

P

tb

C2

SynchronizedTimeBase

ConsumerInterface

C

**图 3.40：示例应用程序和时间同步交互**

交互方法是，对于由AbstractSynchronizedTimeBaseInterface的子类键入的每个PortPrototype ，应用程序开发人员可以访问相应类型的时基资源。

在应用程序代码中，相应的 Time Base 类（如 [ 11 ] 中定义）是使用表示PortPrototype名称的 InstanceSpecifier 构造的。

在应用程序部署期间，那些PortPrototype被映射到时间同步管理中的实际 TimeBase 资源（见图9.24 ）。

## 3.9 平台健康管理界面

**3.9.1 概述**

自适应平台内的平台健康管理功能集群负责监督应用程序的执行，监控它们的状态，并触发相应动作的状态管理。

为了与平台健康管理基础软件接口，应用程序开发者需要声明应用软件提供哪些监督和状态信息，并应由平台健康管理观察。

面向平台健康管理的接口遵循PortPrototype和PortInterface的通用模式，这些模式适用于有关应用软件与平台软件交互的许多用例。

与基于服务的通信相比，使用PortPrototype和PortInterface的平台软件交互建模不太详细。 PortPrototype是用于与平台软件交互的占位符，它不模拟可用于交互的实际使用的 API。要使用的 API 在平台软件 SWS 文档 [ 12 ] 中正式指定。

**3.9.2 受 监管实体和检查点**

监督与平台健康管理的交互由PhmSupervisedEntityInterface和PhmCheckpoint定义。

**[TPS\_MANI\_03500]** { DRAFT }**平台健康管理监督和检查点的定义**d元类PhmSupervisedEntityInterface与聚合的PhmCheckpoint一起用于定义一个受监督实体与平台健康管理监督的交互。 c *( RS\_MANI\_00032 )*

通过定义一个由PhmSupervisedEntityInterface输入的RPortPrototype ，应用程序表明它想要报告这个PhmSupervisedEntityInterface的检查点。

|  |
| --- |
| **图例** /u/v/t/Swc2 = AUTOSAR 模型中模型元素的短名称路径  = 与创建 InstanceSpecifier 相关 |

Executable

„/

a/b/c/

**Exe**

“

RootSwComponentPrototype

„/

a/b/c/Exe/

**Root**

“

SwComponentPrototype

„/

u/v/t/Swc2/

**C2**

"

SwComponentPrototype

„/

q/r/s/Swc1/

**C1**

"

CompositionSwComponent

Type

„/

q/r/s/Swc1"

CompositionSwComponent

Type

„/

u/v/t/Swc2"

AdaptiveApplicationSw

ComponentType

„/

u/v/t/Swc3"

RPortPrototype

„/

u/v/t/Swc3/

**RP**

“

«isOfType»

«isOfType»

InstanceSpecifier: Exe/Root/C1/C2/RP

InstanceRef

phmCheckpoint

contextRootSwComponentPrototype

ContextComponentPrototype[0]

contextComponentPrototype[1]

contextRPortPrototype

PhmSupervisedEntityInterface

„/

f/h/MyPhmInt

“

«isOfType»

PhmCheckpoint

„/

f/h/MyPhmInt/CP

“

targetPhmCheckpoint

«isOfType»

**图 3.41：创建*SupervisedEntity的 InstanceSpecifier 的示例***

**[TPS\_MANI\_03623]** { DRAFT }**在应用程序代码中使用checkpointId** d应用程序代码只能使用那些PhmCheckpoint 。 checkpointId值被定义为PhmSupervisedEntityInterface的成员。检查点。 c *( RS\_MANI\_00032 )*

**[constr\_1727]** { DRAFT } **PortPrototype和PhmSupervisedEntityInterface在应用软件级别**d上的**合格组合 在**APPLICATION\_LEVEL 类别的可执行文件的上下文中，**仅**RPortPrototype支持PhmSupervisedEntityInterface的使用。 c *()*

应用软件在与基础平台软件的交互中扮演着积极的角色，因此从应用软件的角度使用RPortPrototype来表示这种交互。 *SupervisedEntity*实例是使用相应RPortPrototype的 InstanceSpecifier 构造的。

然后应用程序代码调用 SupervisedEntity 的*ReportCheckpoint API* （在 [ 12 ] 中定义） （已在相应RPort-

PhmSupervisedEntityInterface键入的原型）以通知平台健康管理程序流中已到达特定的PhmCheckpoint 。

**[constr\_3530]** { DRAFT } **checkpointId**的**强制定义**d应为每个PhmCheckpoint元素定义checkpointId 。 c *()*

checkpointId在调用*ReportCheckpoint* API 期间用作 PhmCheckpoint 的表示。

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*PlatformHealthManagementInterface*

PhmSupervisedEntityInterface

*AtpFeature*

PhmCheckpoint

+

checkpointId: PositiveInteger

+

checkpoint

0..\*

**图 3.42：受监管实体和检查点的建模**

请注意，从应用程序设计的角度来看，检查点之间没有定义任何关系（以指示报告中的特定观察顺序）。检查点之间的可能转换及其时间方面在PlatformHealthManagementContribution的上下文中定义，并在第9.3.3章中描述。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **PhmSupervisedEntity接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 此元类提供了实现 PortInterface 以与平台健康管理受监督实体交互的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=平台健康管理接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *识别*的、*多语言可引用的*、 *PackageableElement* 、 *PlatformHealthManagementInterface* 、*端口*  *接口*,*可参考* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 检查点 | Phm检查点 | \* | 聚合 | 定义可以在此受监督实体上报告的检查点集。  **标签：** atp.Status=draft |

**表 3.63: PhmSupervisedEntityInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **Phm检查点** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 这个元类提供了实现与平台健康管理受监督实体交互的检查点的能力。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *AtpFeature* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 检查点 ID | 正整数 | 1 | 属性 | 定义用于指示将此检查点报告给 Phm 的数值。  **标签：** atp.Status=draft |

**表 3.64：PhmCheckpoint**

**3.9.3 健康渠道**

健康通道与平台健康管理的交互由PhmHealthChannelInterface和PhmHealthChannelStatus状态定义。

**[TPS\_MANI\_03534]** { DRAFT }**平台健康管理健康定义**

**通道**d元类PhmHealthChannelInterface与聚合的PhmHealthChannelStatus一起用于定义一个健康通道与平台健康管理的交互。 c *( RS\_MANI\_00032 )*

通过定义一个由PhmHealthChannelInterface输入的RPortPrototype （参见 [ constr\_1728 ]），应用程序表明它想要报告这个PhmHealthChannelInterface的状态。

应用软件在与基础平台软件的交互中扮演着积极的角色，因此从应用软件的角度使用RPortPrototype来表示这种交互。 *HealthChannel*实例是使用相应RPortPrototype的 InstanceSpecifier 构造的。

然后应用程序代码调用*HealthChannel （在 PhmHealthChannelInterface*类型的相应RPortPrototype的上下文中构建）的*ReportHealthStatus* API（在 [ 12 ] 中定义），以便通知平台健康管理由RPortPrototype已更改其状态。

**[constr\_3532]** { DRAFT } **statusId**的**强制定义**d应为每个PhmHealthChannelStatus元素定义statusId 。 c *()*

**[TPS\_MANI\_03624]** { DRAFT }**在应用程序代码**d中**使用statusId应用**程序代码只能使用那些PhmHealthChannelStatus 。 statusId值，定义为PhmHealthChannelInterface的成员。状态。 c *( RS\_MANI\_00032 )*

**[TPS\_MANI\_03630]** { DRAFT } **triggersRecoveryNotification的语义**d属性triggersRecoveryNotification定义 PHM 是否应将此特定PhmHealthChannelStatus视为触发恢复通知。 c *( RS\_MANI\_00032 )*

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*PlatformHealthManagementInterface*

PhmHealthChannelInterface

*AtpFeature*

PhmHealthChannelStatus

+

statusId: PositiveInteger

[0..1]

+

triggersRecoveryNotification: Boolean

[0..1]

+

status

0..\*

**图 3.43：健康通道建模**

**[constr\_1728]** { DRAFT } **PortPrototype s 和PhmHealthChannelInterface在应用软件级别**d上的**合格组合 在**APPLICATION\_LEVEL类别的可执行文件的上下文中，仅**RPortPrototype**支持使用PhmHealthChannelInterface 。 c *()*

|  |  |
| --- | --- |
| ***班级*** | **PhmHealthChannel接口** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |
| ***笔记*** | 此元类提供了实现 PortInterface 以与平台健康管理健康通道交互的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=平台健康管理接口 |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **PhmHealthChannel接口** | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *识别*的、*多语言可引用的*、 *PackageableElement* 、 *PlatformHealthManagementInterface* 、*端口*  *接口*,*可参考* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 地位 | 健康频道  地位 | \* | 聚合 | 定义可用于健康通道的可能状态信息集。  **标签：** atp.Status=draft |

**表 3.65：PhmHealthChannelInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **PhmHealthChannelStatus** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | PhmHealthChannelStatus 指定健康通道的一种可能状态。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *AtpFeature* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 状态标识 | 正整数 | 0..1 | 属性 | 定义用于指示 Phm 状态指示的数值。  **标签：** atp.Status=draft |
| 触发器  恢复  通知 | 布尔值 | 0..1 | 属性 | 定义此 PhmHealthChannelStatus 是否应导致 Phm 触发健康通道恢复通知。  True：表示不健康状态。 Phm 用于在 Health 通道状态更改为该状态时触发 Health Channel 恢复通知。  False：表示健康状态。当 Health 通道状态更改为该状态时 Phm 不会触发 Health Channel 恢复通知。  **标签：** atp.Status=draft |

**表 3.66：PhmHealthChannelStatus**

**3.9.4 状态管理恢复通知**

Phm 监控受监管实体和检查点的报告以及报告的健康通道状态信息。如果发生违规，Phm 可以配置为将违规报告给状态管理，并让状态管理处理恢复活动。

图3.44中的示例说明了应用程序 1 和 2 对受监管实体的报告。Phm 被配置为执行对这些报告元素的监管。在违规情况下，Phm 被配置为通知状态管理应用程序来处理这种情况。

**[TPS\_MANI\_01280]** { DRAFT }**元类PhmSupervisionRecoveryNotificationInterface的语义**d PHM 监督失败的恢复通知确实是在状态管理软件级别调用一段代码。

在状态管理软件级别激活代码的机制是对由PhmSupervisionRecoveryNotificationInterface键入的PPortPrototype 建模。 c *( RS\_MANI\_00032 )*

**[TPS\_MANI\_03631]** { DRAFT }**元类PhmHealthChannelRecoveryNotificationInterface的语义**d PHM 对失败的 HealthChannel 监控的恢复通知确实是调用状态管理软件级别的一段代码。

在状态管理软件级别激活代码的机制是对由PhmHealthChannelRecoveryNotificationInterface键入的PPortPrototype 建模。 c *( RS\_MANI\_00032 )*

TPS\_MANI\_01280 ] 和 [ TPS\_MANI\_03631 ]的上下文中调用的操作在平台健康管理规范文档 [ 12 ] 中定义。

如前所述，状态管理应该执行恢复操作。这意味着PhmSupervisionRecoveryNotificationInterface和PhmHealthChannelRecoveryNotificationInterface只能与 PPortPrototype 结合使用。 [ constr\_1729 ]阐明了这一方面。

Platform

Health

Manager

Application

2

SE

A

Phm

SupervisedEntity

Interface

Application

1

SE

D

State

Management

Global

Supervision

Supervision

U

Supervision

V

Supervision

W

Supv

Notf

PhmSupervision

RecoveryNotification

Interface

SE

A

**图 3.44：Phm 监控和恢复设置示例**

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*PlatformHealthManagementInterface*

*PhmAbstractRecoveryNotificationInterface*

PhmSupervisionRecoveryNotificationInterface

PhmHealthChannelRecoveryNotificationInterface

**图 3.45： PhmAbstractRecoveryNotificationInterface的建模**

**[constr\_1729]** { DRAFT } **PortPrototype和PhmSupervisionRecoveryNotificationInterface / PhmHealthChannelRecoveryNotificationInterface在状态管理软件级别**d的**合格组合 在**类别APPLICATION\_LEVEL 的可执行文件的上下文中， Phm-

SupervisionRecoveryNotificationInterface和PhmHealthChannelRecoveryNotificationInterface仅支持PPortPrototype 。 **\_** c *()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **PhmSupervisionRecoveryNotificationInterface** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 此元类表示可用于实现 PHM 监督通知的 PortInterface。  **标签：**  atp.Status=草稿  atp.recommendedPackage=平台健康管理接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PhmAbstractRecoveryNotificationInterface* , *PlatformHealthManagementInterface* , *PortInterface* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.67：PhmSupervisionRecoveryNotificationInterface**

|  |  |
| --- | --- |
| ***班级*** | **PhmHealthChannelRecoveryNotificationInterface** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface |
| ***笔记*** | 此元类表示一个 PortInterface，可用于实现 PHM HealthChannel 通知。  **标签：**  atp.Status=草稿  atp.recommendedPackage=平台健康管理接口 |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **PhmHealthChannelRecoveryNotificationInterface** | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PhmAbstractRecoveryNotificationInterface* , *PlatformHealthManagementInterface* , *PortInterface* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.68：PhmHealthChannelRecoveryNotificationInterface**

## 3.10 诊断接口

**3.10.1 概述**

在*AUTOSAR 自适应平台上，*为应用层软件与 AUTOSAR 诊断管理器的交互定义了专用的PortInterface 。

与 AUTOSAR 经典平台上的约定相比，这些PortInterface以及扩展的标准化 ara::diag API**仅用**于此通信关系的应用程序端。

AUTOSAR 诊断管理器一侧的接口（以及因此面向 AUTOSAR 诊断管理器的PortPrototype实现部分）是**完全专有的**。这方面在图3.46中进行了描述。

Diagnostic Manager

Application Software

Standardized

ara::diag

proprietary

proprietary

PortPrototype

DiagnosticPortInterface

«isOfType»

**图 3.46：ara::diag 实现中的标准化与专有部分**

这种安排试图从应用程序的角度为应用程序程序员提供最简单的 API。同时，它在坚实的抽象层后面隐藏了应用程序和诊断管理器之间交互的许多复杂性。

*PortInterface*

*DiagnosticPortInterface*

DiagnosticServiceValidationInterface

DiagnosticSecurityLevelInterface

DiagnosticOperationCycleInterface

*DiagnosticAbstractDataIdentifierInterface*

*DiagnosticAbstractRoutineInterface*

DiagnosticConditionInterface

DiagnosticDTCInformationInterface

DiagnosticDoIPGroupIdentificationInterface

DiagnosticDoIPPowerModeInterface

DiagnosticEventInterface

DiagnosticGenericUdsInterface

DiagnosticIndicatorInterface

DiagnosticMonitorInterface

DiagnosticDownloadInterface

DiagnosticUploadInterface

DiagnosticDoIPActivationLineInterface

DiagnosticAuthenticationInterface

DiagnosticExternalAuthenticationInterface

**图 3.47：用于诊断目的的PortInterface建模**

**[TPS\_MANI\_01242]** { DRAFT } **PortInterface用于与 AUTOSAR 诊断管理器进行通信**d用于此目的的所有PortInterface都派生自抽象元类DiagnosticPortInterface 。 DiagnosticPortInterface没有实现面向服务的通信模式，特别是在所涉及的 API 级别上没有明确的服务发现。 c *( RS\_MANI\_00061 )*

DiagnosticPortInterface的专业化涵盖了诊断通信的各个方面，例如诊断例程的实施、诊断事件的报告或对诊断数据标识符 (DID) 的访问。

图3.47描述了所有直接继承自DiagnosticPortInterface的元类。

|  |  |
| --- | --- |
| ***班级*** | ***诊断端口接口***（抽象） |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface |
| ***笔记*** | 此元类充当所有与诊断相关的 PortInterfaces 的抽象基类。  **标签：** atp.Status=draft |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *识别*，*多语言可引用*，*可打包元素*，端口接口，*可引用* |
| ***子类*** | *DiagnosticAbstractDataIdentifierInterface* , *DiagnosticAbstractRoutineInterface* , DiagnosticAuthentication  接口， DiagnosticComControlInterface ， DiagnosticConditionInterface ， DiagnosticDTCInformation  接口, DiagnosticDoIPActivationLineInterface , DiagnosticDoIPGroupIdentificationInterface ,诊断  DoIPPowerModeInterface , DiagnosticDoIPTriggerVehicleAnnouncementInterface , DiagnosticDownload 接口, DiagnosticEcuResetInterface , DiagnosticEventInterface , DiagnosticExternalAuthentication  接口, DiagnosticGenericUdsInterface , DiagnosticIndicatorInterface , DiagnosticMonitorInterface , DiagnosticOperationCycleInterface , DiagnosticSecurityLevelInterface , DiagnosticServiceValidation 接口, DiagnosticUploadInterface |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***诊断端口接口***（抽象） | | |  |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.69：诊断端口接口**

**3.10.2 诊断程序接口**

创建诊断例程的惯例是为每个诊断例程建立最多三种方法：

* 开始执行例程。
* 停止例程的执行。
* 请求例程的执行结果。

为了响应这个约定， DiagnosticRoutineInterface被建模为以三个专用角色聚合ClientServerOperation ： start 、 stop和requestResult 。

**[constr\_1696]** { DRAFT }由DiagnosticRoutineInterface聚合的**ClientServerOperation** d**由DiagnosticRoutineInterface**聚合的任何ClientServerOperation都不应定义以下属性：

* 火与忘
* 可能的ApError
* 可能的ApErrorSet

### c ()

诊断例程的参数应建模为在角色start 、 stop和requestResult中聚合的各个ClientServerOperation的参数。

*PortInterface*

*DiagnosticPortInterface*

DiagnosticRoutineInterface

*AtpStructureElement*

*Identifiable*

ClientServerOperation

[0..1]

fireAndForget: Boolean

+

ArgumentDataPrototype

[0..1]

direction: ArgumentDirectionEnum

+

[0..1]

serverArgumentImplPolicy: ServerArgumentImplPolicyEnum

+

*DataPrototype*

*AutosarDataPrototype*

DiagnosticRoutineGenericInterface

*DiagnosticAbstractRoutineInterface*

start

+

0..1

stop

+

0..1

«atpVariation»

+

argument

ordered

}

0..\* {

+

requestResult

0..1

**图 3.48： DiagnosticRoutineInterface的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断例程接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 这个元类代表了在自适应平台上实现以例程为中心的 PortInterface 诊断的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticAbstractRoutineInterface* , *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 请求结果 | 客户端服务器操作 | 0..1 | 聚合 | 这表示诊断例程的请求结果方法。  **标签：** atp.Status=draft |
| 开始 | 客户端服务器操作 | 0..1 | 聚合 | 这表示诊断例程的启动方法。  **标签：** atp.Status=draft |
| 停止 | 客户端服务器操作 | 0..1 | 聚合 | 这代表了诊断程序的停止方法。  **标签：** atp.Status=draft |

**表 3.70：诊断例程接口**

除了使用 DiagnosticRoutineInterface 对“类型化”诊断例程进行建模之外，还可以使用DiagnosticRoutineGenericInterface来定义不提供进一步形式化的诊断例程。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断例程通用接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 这个元类代表了在自适应平台上实现一个以常规为中心的通用端口接口以进行诊断的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* , *DiagnosticAbstractRoutineInterface* , *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.71：诊断例程通用接口**

这意味着仍然隐含地为已经提到的诊断例程角色定义了多达三种方法。

然而，在这种通用诊断例程的上下文中的方法总是使用纯字节数组作为参数，因此 AUTOSAR 元模型中的形式化不再有意义。

元类DiagnosticAbstractRoutineInterface作为*AUTOSAR 自适应平台*上所有例程相关的DiagnosticPortInterface的抽象基类。

**[constr\_10031]** { DRAFT }**存在DiagnosticRoutineInterface 。启动**d属性DiagnosticRoutineInterface 。 start应**在清单创建完成时存在**。 c *()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***DiagnosticAbstractRoutineInterface*** （摘要） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 此元类用作 PortInterfaces 的抽象基类，专用于 AUTOSAR 自适应平台上的例程执行。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***子类*** | 诊断例程通用接口 | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.72：DiagnosticAbstractRoutineInterface**

**3.10.3 数据标识符接口和数据标识符元素**

中访问与诊断相关的**数据的能力在**DiagnosticPortInterface的另一个抽象子类中形式化： DiagnosticAbstractDataIdentifierInterface 。

反过来，元类DiagnosticAbstractDataIdentifierInterface定义了三个具体的子类，它们表示访问应用程序软件中与诊断相关的数据的具体能力。

**[TPS\_MANI\_01243]** { DRAFT } **DiagnosticDataIdentifierInterface** d DiagnosticDataIdentifierInterface的**语义**用于一次访问整个 DID 的内容。

为此，根据特定DiagnosticDataIdentifierInterface的具体用例，最多可以在读取和写入角色中聚合两个ClientServerOperation 。 c *( RS\_MANI\_00061 )*

**[constr\_10030]** { DRAFT }**存在DiagnosticDataIdentifierInterface 。读取**d属性DiagnosticDataIdentifierInterface 。 read应**在清单创建完成时存在**。 c *()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断数据标识符接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 这个元类代表了在自适应平台上实现以 DID 为中心的 PortInterface 诊断的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticAbstractDataIdentifierInterface* , *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 读 | 客户端服务器操作 | 0..1 | 聚合 | 这表示读取诊断数据标识符内容的方法。**标签：** atp.Status=draft |
| 写 | 客户端服务器操作 | 0..1 | 聚合 | 这表示写入诊断数据标识符的内容的方法。**标签：** atp.Status=draft |

**表 3.73：诊断数据标识符接口**

*PortInterface*

*DiagnosticPortInterface*

DiagnosticDataElementInterface

DiagnosticDataIdentifierGenericInterface

DiagnosticDataIdentifierInterface

*AtpStructureElement*

*Identifiable*

ClientServerOperation

fireAndForget: Boolean

+

[0..1]

ArgumentDataPrototype

+

direction: ArgumentDirectionEnum

[0..1]

[0..1]

serverArgumentImplPolicy: ServerArgumentImplPolicyEnum

+

*DataPrototype*

*AutosarDataPrototype*

*DiagnosticAbstractDataIdentifierInterface*

read

+

0..1

+

write

0..1

read

+

0..1

«atpVariation»

+

argument

0..\* {

ordered

}

**图 3.49： DiagnosticDataIdentifierInterface的建模**

**[TPS\_MANI\_01244]** { DRAFT } **DiagnosticDataElementInterface** d DiagnosticDataElementInterface的**语义**用于访问给定 DID 内元素的内容。

为此，在角色read中聚合了一个ClientServerOperation 。 c *( RS\_MANI\_00061 )*

请注意， DiagnosticDataElementInterface*故意*不支持写入操作，因为如果将数据分段发送到应用软件，原则上无法保证数据的一致性。可以配置不同的进程来接收数据，但不能保证在传输数据时所有进程都启动并运行。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断数据元素接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 这个元类代表了在自适应平台上实现以 DID 为中心的 PortInterface 进行诊断的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticAbstractDataIdentifierInterface* , *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 读 | 客户端服务器操作 | 0..1 | 聚合 | 这表示读取诊断数据标识符的元素内容的方法。  **标签：** atp.Status=draft |

**表 3.74：诊断数据元素接口**

**[TPS\_MANI\_01245]** { DRAFT } **DiagnosticDataIdentifierGenericInterface** d DiagnosticDataIdentifierInterface的**语义**用于一次访问整个 DID 的内容。

为此，将使用读写语义定义方法，但这些方法将始终仅提供字节数组的参数。

因此，在 AUTOSAR 元模型的上下文中对这些用于读取和写入数据的方法进行进一步形式化是没有意义的，因此被省略。 c *( RS\_MANI\_00061 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断数据标识符通用接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 这个元类代表了在自适应平台上实现一个以 DID 为中心的通用 PortInterface 诊断的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticAbstractDataIdentifierInterface* , *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.75：诊断数据标识符通用接口**

**除非**给定的DiagnosticDataIdentifierInterface或DiagnosticDataElementInterface在角色read**或**write中仅聚合**一个**ClientServerOperation ，**否则有**必要对参数进行一些限制。

**[constr\_1697]** { DRAFT }对**由DiagnosticDataIdentifierInterface或DiagnosticDataElementInterface聚合的ClientServerOperation**的**限制**如果元类DiagnosticDataIdentifierInterface或DiagnosticDataElementInterface聚合两个ClientServerOperation则

* 两个ClientServerOperation应该有相同数量的参数。
* 参数s 集合中第n个位置的参数s 应具有相同的属性，方向除外。特别是，关于属性方向，应满足以下条件：
  + 由ClientServerOperation聚合的任何ArgumentDataPrototype本身在角色DiagnosticDataIdentifierInterface中聚合。读取或DiagnosticDataElementInterface 。 read应将属性方向设置为out 。
  + 由ClientServerOperation聚合的任何ArgumentDataPrototype本身聚合在角色DiagnosticDataIdentifierInterface中。 write应将属性方向设置为in 。

### c ()

**3.10.4 诊断事件接口**

AUTOSAR 定义了几个专用于处理诊断事件的DiagnosticPortInterface子类。

**[TPS\_MANI\_01246]** { DRAFT } **DiagnosticMonitorInterface的语义**d元类DiagnosticMonitorInterface表示向 AUTOSAR 诊断管理器报告诊断事件的能力。 c *( RS\_MANI\_00061 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断监视器接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 此元类表示在自适应平台上实现以监视器为中心的 PortInterface 以进行诊断的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.76：诊断监视器接口**

*PortInterface*

*DiagnosticPortInterface*

DiagnosticEventInterface

DiagnosticDTCInformationInterface

DiagnosticMonitorInterface

**图 3.50： DiagnosticEventInterface的建模**

**[TPS\_MANI\_01247]** { DRAFT } **DiagnosticDTCInformationInterface的语义**d元类DiagnosticDTCInformationInterface表示检索有关给定诊断故障代码的信息的能力。 c *( RS\_MANI\_00061 )*

|  |  |
| --- | --- |
| ***班级*** | **诊断DTC信息接口** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface |
| ***笔记*** | 此元类表示实现 PortInterface 以访问自适应平台上的 DTC 属性的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断DTC信息接口** | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.77：诊断 DTCInformation 接口**

**[TPS\_MANI\_01248]** { DRAFT } **DiagnosticEventInterface的语义**d元类DiagnosticEventInterface表示检索有关给定诊断事件的信息的能力。 c *( RS\_MANI\_00061 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断事件接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 此元类表示实现 PortInterface 以访问自适应平台上诊断事件的属性的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.78：诊断事件接口**

**3.10.5 诊断条件接口**

**[TPS\_MANI\_01249]** { DRAFT } **DiagnosticConditionInterface** d AUTOSAR**的语义**支持不同的诊断条件，即启用条件和清除条件。这方面在 AUTOSAR 自适应平台的 DiagnosticConditionInterface 的定义*中*表示。 c *( RS\_MANI\_00061 )*

DiagnosticConditionInterface在其形式化中不需要任何进一步的细节。

*PortInterface*

*DiagnosticPortInterface*

DiagnosticConditionInterface

**图 3.51： DiagnosticConditionInterface的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断条件接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 此元类表示实现 PortInterface 以在自适应平台上处理诊断条件请求的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.79：诊断条件接口**

**3.10.6 指标界面**

**[TPS\_MANI\_01250]** { DRAFT } DiagnosticIndicatorInterface的**语义**d **DiagnosticIndicatorInterface**的使用预计用于实现诊断指示器（即仪表板上的警告灯）的软件。 c *( RS\_MANI\_-*

*00061 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断指示器接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 此元类表示实现 PortInterface 以在自适应平台上实现指标功能的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.80：诊断指标接口**

*PortInterface*

*DiagnosticPortInterface*

DiagnosticIndicatorInterface

**图 3.52： DiagnosticIndicatorInterface的建模**

DiagnosticIndicatorInterface在其形式化中不需要任何进一步的细节。

**3.10.7 安全级别接口**

**[TPS\_MANI\_01251]** { DRAFT } **DiagnosticSecurityLevelInterface的语义**d可以预见， DiagnosticSecurityLevelInterface的使用将用于执行检查给定安全级别的许可的软件。 c *( RS\_-*

*MANI\_00061 )*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticSecurityLevelInterface

**图 3.53： DiagnosticSecurityLevelInterface的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断安全级别接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 此元类表示在自适应平台上实现以安全级别为重点的 PortInterface 进行诊断的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.81：诊断安全级别接口**

DiagnosticSecurityLevelInterface在其形式化中不需要任何进一步的细节。

**3.10.8 服务验证接口**

**[TPS\_MANI\_01252]** { DRAFT } **DiagnosticServiceValidationInterface的语义**d预计DiagnosticServiceValidationInterface的使用将用于在制造商或供应商层面实施许可检查的软件。 c *( RS\_MANI\_00061 )*

DiagnosticServiceValidationInterface在其形式化中不需要任何进一步的细节。

*PortInterface*

*DiagnosticPortInterface*

DiagnosticServiceValidationInterface

**图 3.54： DiagnosticsServiceValidationInterface的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断服务验证接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 此元类表示实现 PortInterface 以处理自适应平台上的服务验证请求的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.82：诊断服务验证接口**

**3.10.9 运行周期界面**

**[TPS\_MANI\_01253]** { DRAFT } **DiagnosticOperationCycleInterface的语义**d对于实现管理操作周期的软件，可以预见到DiagnosticOperationCycleInterface的使用。 c *( RS\_MANI\_00061 )*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticOperationCycleInterface

**图 3.55： DiagnosticsOperationCycleInterface的建模**

|  |  |
| --- | --- |
| ***班级*** | **诊断操作循环接口** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface |
| ***笔记*** | 此元类表示实现 PortInterface 以处理自适应平台上的操作周期请求的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断操作循环接口** | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.83：诊断操作循环接口**

DiagnosticOperationCycleInterface在其形式化中不需要任何进一步的细节。

**3.10.10 通用 UDS 接口**

**[TPS\_MANI\_01254]** { DRAFT } **DiagnosticGenericUdsInterface的语义**d AUTOSAR 诊断通信 API 还预见到存在一个支持 UDS 服务的完全通用处理程序实现的DiagnosticPortInterface 。 c *( RS\_MANI\_00061 )*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticGenericUdsInterface

**图 3.56： DiagnosticsGenericUdsInterface的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断通用 Uds 接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 此元类表示实现通用 UDS PortInterface 以在自适应平台上进行诊断的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.84：DiagnosticsGenericUdsInterface**

DiagnosticGenericUdsInterface在其形式化中不需要任何进一步的细节。

**3.10.11 DoIP 接口**

**[TPS\_MANI\_01255]** { DRAFT } **DoIP 的语义DiagnosticPortInterface s** d AUTOSAR 诊断通信 API 还预见到 DiagnosticPortInterface 的存在，以在 DoIP 操作的上下文中实现功能。

DiagnosticPortInterface的具体子类以支持在 DoIP 上下文中实现功能：

* 诊断DoIPGroupIdentificationInterface
* 诊断DoIPPowerModeInterface
* 诊断DoIPActivationLineInterface
* 诊断DoIPTriggerVehicleAnnouncementInterface c *( RS\_MANI\_00061 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断DoIPGroupIdentificationInterface** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 此元类表示实现 PortInterface 以在自适应平台上实现 DoIP 组标识的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.85：DiagnosticDoIPGroupIdentificationInterface**

*PortInterface*

*DiagnosticPortInterface*

DiagnosticDoIPGroupIdentificationInterface

DiagnosticDoIPPowerModeInterface

DiagnosticDoIPActivationLineInterface

DiagnosticDoIPTriggerVehicleAnnouncementInterface

**图 3.57：DoIP DiagnosticPortInterface的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断DoIPPowerModeInterface** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 此元类表示实现 PortInterface 以在自适应平台上实现 DoIP 电源模式的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.86：DiagnosticsDoIPPowerModeInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断DoIPActivationLineInterface** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 此元类表示实现 PortInterface 以在自适应平台上实现 DoIPActivationLine 的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.87：DiagnosticDoIPActivationLineInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **DiagnosticDoIPTriggerVehicleAnnouncementInterface** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 此元类表示实现 PortInterface 以在自适应平台上实现 DoIPTriggerVehicle Announcement 的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.88：DiagnosticDoIPTriggerVehicleAnnouncementInterface**

DiagnosticDoIPGroupIdentificationInterface 、 DiagnosticDoIPPowerModeInterface 、 DiagnosticDoIPActivationLineInterface和DiagnosticDoIPTriggerVehicleAnnouncementInterface在其形式化中不需要任何进一步的细节。

**3.10.12 上传和下载诊断接口**

**[TPS\_MANI\_01265]** { DRAFT } **DiagnosticDownloadInterface和DiagnosticDownloadInterface** d的**语义**AUTOSAR 诊断通信 API 还预见到DiagnosticPortInterface的存在，以通过诊断通道实现上传和下载。

DiagnosticPortInterface的具体子类来支持上传和下载的实现：

* 诊断上传接口
* 诊断下载接口c *( RS\_MANI\_00061 )*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticDownloadInterface

DiagnosticUploadInterface

**图 3.58： DiagnosticUploadInterface和DiagnosticDownloadInterface的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断上传接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 此元类表示实现 PortInterface 以处理使用自适应平台上的诊断通道上传数据的请求的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.89：诊断上传接口**

|  |  |
| --- | --- |
| ***班级*** | **诊断下载接口** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface |
| ***笔记*** | 此元类表示实现 PortInterface 以处理使用自适应平台上的诊断通道下载数据的请求的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断下载接口** | |  |  |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.90：诊断下载接口**

DiagnosticUploadInterface和DiagnosticDownloadInterface在其形式化中不需要任何进一步的细节。

**3.10.13 支持管理 EcuReset 的接口**

**[TPS\_MANI\_01332]** { DRAFT } **DiagnosticEcuResetInterface的语义**d元类DiagnosticEcuResetInterface表示支持处理重置机器的请求的能力。

此接口通常由 AUTOSAR 自适应平台上的状态管理器使用。 c *( RS\_MANI\_00061 )*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticEcuResetInterface

**图 3.59： DiagnosticEcuResetInterface的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断EcuReset接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 此元类表示实现一个集中的 PortInterface 以在自适应平台上处理诊断服务 EcuReset 的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.91：DiagnosticEcuResetInterface**

**3.10.14 诊断认证接口**

**[TPS\_MANI\_01359]** { DRAFT } **DiagnosticAuthenticationInterface的语义**d在应用软件中支持诊断认证的能力在DiagnosticPortInterface的另一个子类中正式化： DiagnosticAuthenticationInterface 。 c *( RS\_MANI\_00061 )*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticAuthenticationInterface

**图 3.60： DiagnosticAuthenticationInterface的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断认证接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 此元类表示实现一个集中的 PortInterface 以处理自适应平台上的诊断服务通信控制的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.92：诊断验证接口**

**3.10.15 诊断外部认证接口**

**[TPS\_MANI\_01353]** { DRAFT } **DiagnosticExternalAuthenticationInterface的语义**d在应用软件中支持诊断客户端身份验证的能力在DiagnosticPortInterface的另一个子类中正式化： DiagnosticExternalAuthenticationInterface 。 c *( RS\_MANI\_00061 )*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticExternalAuthenticationInterface

**图 3.61： DiagnosticExternalAuthenticationInterface的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断外部身份验证接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 此元类表示实现一个集中的 PortInterface 以处理自适应平台上的诊断客户端身份验证的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.93：诊断外部身份验证接口**

**3.10.16 诊断通讯控制接口**

**[TPS\_MANI\_01363]** { DRAFT } **DiagnosticComControlInterface的语义**d支持与诊断客户端通信的激活和停用的能力在 DiagnosticPortInterface 的另一个子类中正式化： DiagnosticComControlInterface 。 c *( RS\_MANI\_00061 )*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticComControlInterface

**图 3.62： DiagnosticComControlInterface的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断通信控制接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface | | | |
| ***笔记*** | 此元类表示实现一个集中的 PortInterface 以处理自适应平台上的诊断服务通信控制的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *DiagnosticPortInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.94：诊断通信控制接口**

## 3.11 加密接口

**3.11.1 与加密软件的交互**

**[TPS\_MANI\_03253]** { DRAFT }**与加密软件的交互**d与应用软件实例上的加密软件的交互应基于从抽象元类CryptoInterface派生的PortInterface类型化的RPortPrototype的存在建模。 c *( RS\_MANI\_00031 )*

与 AUTOSAR 经典平台上的约定不同，这些CryptoInterface仅用于此通信关系的应用程序端。

Crypto API 在 [ 13 ] 中有描述。引用CryptoInterface的RPortPrototype的模型路径由定义应用程序开发人员在 API 调用中使用的逻辑本地名称的 ara::core::InstanceSpecifier 提供。此本地 ara::core::InstanceSpecifier 在运行时与来自部署模型的信息一起转换为特定的加密对象，例如 CryptoKeyStorage 中的 CryptoKeySlot。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***加密接口***（摘要） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***笔记*** | 此元类提供了定义 PortInterface 以支持加密用例的抽象能力。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *识别*，*多语言可引用*，*可打包元素*，端口接口，*可引用* | | | |
| ***子类*** | CryptoCertificateInterface , CryptoKeySlotInterface , CryptoProviderInterface , CryptoTrustMasterInterface | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.95：加密接口**

*AbstractRequiredPortPrototype*

RPortPrototype

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

CryptoKeySlotInterface

+

allocateShadowCopy: Boolean

[0..1]

[0..1]

+

cryptoAlgId: String

+

cryptoObjectType: CryptoObjectTypeEnum

[0..1]

slotCapacity: PositiveInteger

[0..1]

+

+

slotType: CryptoKeySlotTypeEnum

[0..1]

CryptoProviderInterface

CryptoCertificateInterface

isPrivate: Boolean

[0..1]

+

+

writeAccess: Boolean

[0..1]

CryptoTrustMasterInterface

«enumeration»

CryptoObjectTypeEnum

Undefined

SymmetricKey

PrivateKey

PublicKey

Signature

SecretSeed

CryptoKeySlotAllowedModification

+

allowContentTypeChange: Boolean

[0..1]

exportability: Boolean

+

[0..1]

[0..1]

+

maxNumberOfAllowedUpdates: PositiveInteger

[0..1]

restrictUpdate: Boolean

+

CryptoKeySlotContentAllowedUsage

[0..1]

+

allowedKeyslotUsage: String

*CryptoInterface*

«enumeration»

CryptoKeySlotTypeEnum

machine

application

«enumeration»

CryptoKeySlotUsageEnum

verification

encryption

+

keySlotContentAllowedUsage

0..\*

«isOfType»

requiredInterface

+

0..1

}

redefines atpType

{

keySlotAllowedModification

+

0..1

**图 3.63: CryptoInterface用于建模应用程序与 Crypto 软件的交互**

图 3.63描述了所有直接继承自CryptoInterface的元类。

**3.11.2 加密密钥槽接口**

**[TPS\_MANI\_03254]** { DRAFT }**使用和修改加密密钥的应用程序建模**d使用和修改加密密钥的应用程序软件被建模为具有RPortPrototype的AdaptiveApplicationSwComponentType ，该 RPortPrototype由具有将slotType值设置为application的CryptoKeySlotInterface键入。 c *( RS\_MANI\_00031 )*

**[TPS\_MANI\_03255]** { DRAFT }**管理堆栈服务使用的加密密钥的密钥管理器应用程序的建模**d管理堆栈服务（如 COM、持久性或诊断）使用的加密密钥的密钥管理器应用程序建模为AdaptiveApplicationSwComponentType由CryptoKeySlotInterface键入的RPortPrototype ，其slotType值设置为machine 。 c *( RS\_MANI\_00031 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **CryptoKeySlot接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***笔记*** | 这个元类提供了为 Crypto Key Slots 定义 PortInterface 的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=加密接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *CryptoInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| allocateShadow Copy | 布尔值 | 0..1 | 属性 | 此属性定义是否应分配此 Key Slot 的卷影副本以启用失败的 Key Slot 更新活动的回滚（参见接口 BeginTransaction）。  **标签：** atp.Status=draft |
| 加密算法 ID | 细绳 | 0..1 | 属性 | 该属性定义了加密算法限制（kAlgId Any 表示没有限制）。该算法可以部分指定：家庭和长度，模式，填充。  未来的加密提供者可以支持一些目前还不为人所知/标准化的加密算法，因此 AUTOSAR 不提供加密算法标识符的具体列表，也不假设使用数字标识符。取而代之的是，提供者供应商应在随附文档中提供支持算法的字符串名称。密码算法的名称应遵循自适应平台密码规范中定义的规则。  **标签：** atp.Status=draft |
| 加密对象类型 | 加密对象类型枚举 | 0..1 | 属性 | 可以存储在插槽中的对象类型。如果此字段包含“未定义”，则必须提供 mSlotCapacity 且大于 0  **标签：** atp.Status=draft |
| keySlotAllowed 修改 | CryptoKeySlotAllowed  修改 | 0..1 | 聚合 | 限制如何使用这个 keySlot  **标签：** atp.Status=draft |
| keySlotContent AllowedUsage | CryptoKeySlot 内容  允许使用 | \* | 聚合 | 存储到槽的密钥的允许使用限制。  **标签：** atp.Status=draft |
| 槽容量 | 正整数 | 0..1 | 属性 | 堆栈供应商要保留的插槽容量（以字节为单位）。一个用例是在未定义cryptoObjectType 并且无法从cryptoObjectType 和cryptoAlgId 推导出槽大小的情况下定义此值。  “0”表示可以从cryptoObject Type 和cryptoAlgId 推导出槽大小。**标签：** atp.Status=draft |
| 插槽类型 | CryptoKeySlotType  枚举 | 0..1 | 属性 | 该属性定义keySlot是否被Application独占使用；或者它是否由堆栈服务使用并由密钥管理器应用程序管理。  **标签：** atp.Status=draft |

**表 3.96：CryptoKeySlot 接口**

|  |  |
| --- | --- |
| ***枚举*** | **CryptoKeySlotTypeEnum** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign |
| ***笔记*** | 此枚举定义了在平台中使用 Key Slot 的选项。  **标签：** atp.Status=draft |

5

4

|  |  |
| --- | --- |
| ***枚举*** | **CryptoKeySlotTypeEnum** |
| ***文字*** | ***描述*** |
| 应用 | KeySlot 仅由应用程序使用和修改。  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |
| 机器 | 钥匙槽仅供平台模块使用。应用程序管理密钥，但不能使用密钥。  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |

**表 3.97: CryptoKeySlotTypeEnum**

|  |  |
| --- | --- |
| ***枚举*** | **加密对象类型枚举** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign |
| ***笔记*** | 枚举所有类型的加密对象，即可以存储到密钥槽的内容类型。  **标签：** atp.Status=draft |
| ***文字*** | ***描述*** |
| 私钥 | 密码::PrivateKey 对象  **标签：**  atp.EnumerationLiteralIndex=2 atp.Status=草稿 |
| 公钥 | 密码::PublicKey 对象  **标签：**  atp.EnumerationLiteralIndex=3 atp.Status=草稿 |
| 秘密种子 | crypt::SecretSeed 对象  **标签：**  atp.EnumerationLiteralIndex=5 atp.Status=草稿 |
| 签名 | crypt::Signature 对象（非对称数字签名或对称 MAC/HMAC）  **标签：**  atp.EnumerationLiteralIndex=4 atp.Status=草稿 |
| 对称密钥 | crypt::SymmetricKey 对象  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |
| 不明确的 | 对象类型未知  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |

**表 3.98: CryptoObjectTypeEnum**

请注意，部署模型（机器清单）中描述了将CryptoKeySlot分配给CryptoProvider 。通过这种映射，还建立了CryptoKeySlot到 CryptoProvider 的CryptoPrimitive的分配。

但是应用程序开发人员能够将 CryptoKeySlot 的使用限制为具有属性cryptoAlgId的特定加密算法。

为了支持当今不为人所知/标准化的加密算法，AUTOSAR 不提供加密算法标识符的具体列表，也不假设使用数字标识符。

取而代之的是，提供者供应商应在随附文档中提供支持算法的字符串名称。密码算法的名称应遵循自适应平台密码规范中定义的规则。

此外，应用程序开发人员能够为CryptoKeySlot的使用定义进一步的要求。使用属性cryptoObjectType可以指定允许存储在密钥槽中的加密对象。

密钥槽的允许修改可以通过keySlotAllowedModification指定。可以通过keySlotContentAllowedUsage指定键槽内容的允许使用。

执行对加密存储中的CryptoKeySlot的分配，集成商需要考虑应用程序设计模型中定义的设置。请注意，应用程序设计模型设置被转移到部署模型中，因此在运行时也可用，如第9.10.2章所述。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **CryptoKeySlotAllowedModification** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***笔记*** | 这个元类限制了对存储在密钥槽中的密钥的允许修改。  **标签：** atp.Status=draft | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 允许内容类型更改 | 布尔值 | 0..1 | 属性 | 该属性描述了密钥内容类型是否可以更改（真）或不可以（假），例如将密钥从对称更改为 RSA。  **标签：** atp.Status=draft |
| 可出口性 | 布尔值 | 0..1 | 属性 | 该属性描述是否允许导出键槽内容。  **标签：** atp.Status=draft |
| 最大数量  允许更新 | 正整数 | 0..1 | 属性 | 此属性描述了允许对插槽进行的最大更新。  **标签：** atp.Status=draft |
| 限制更新 | 布尔值 | 0..1 | 属性 | 该属性定义是否定义了对更新次数的限制。  False：不限制更新次数。 True：使用属性 maxNumberOfAllowedUpdates 限制更新次数。  **标签：** atp.Status=draft |

**表 3.99：CryptoKeySlotAllowedModification**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **CryptoKeySlotContentAllowedUsage** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***笔记*** | 此元类限制存储在密钥槽中的密钥的允许使用。  **标签：** atp.Status=draft | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| allowedKeyslot 用法 | 细绳 | 0..1 | 属性 | 此属性定义 KeySlot 可用于哪些操作。  **标签：** atp.Status=draft |

**表 3.100：CryptoKeySlotContentAllowedUsage**

**[constr\_5238]** {草案} **CryptoKeySlotAllowedModification 。 restrictUpdate和 maxNumberOfAllowedUpdates 的关系**d如果CryptoKeySlotAllowedModification 。然后将restrictUpdate设置为true

CryptoKeySlotAllowedModification 。 maxNumberOfAllowedUpdates应设置为一个值。 c *()*

**[constr\_5239]** { DRAFT } **CryptoKeySlotContentAllowedUsage的预定义值。 allowedKeyslotUsage** d CryptoKeySlotContentAllowedUsage的以下值。 allowedKeyslotUsage由 AUTOSAR 预定义：

* 允许数据加密，
* 允许数据解密，
* 允许签名，
* 允许验证，
* 允许密钥协议，
* 允许键多样化，
* 允许-DRNG-初始化，
* 允许-KDF-材料，
* 允许密钥导出，
* 允许密钥导入，
* 仅允许精确模式，
* 允许派生数据加密，
* 允许派生数据解密，
* 允许的签名，
* 允许派生验证，
* 允许衍生的多样化，
* 允许派生的 DRNG 初始化，
* 允许衍生的 KDF 材料，
* 允许导出密钥，
* 允许派生密钥导入，
* 仅允许派生的精确模式

### c ()

**3.11.3 加密证书接口**

**[TPS\_MANI\_03256]** { DRAFT }**访问加密证书的应用程序建模**d访问加密证书的应用程序软件被建模为

AdaptiveApplicationSwComponentType具有由CryptoCertificateInterface键入的RPortPrototype 。 c *( RS\_MANI\_00031 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **加密证书接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***笔记*** | 这个元类提供了为 CryptoCertificate 定义 PortInterface 的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=加密接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *CryptoInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 是私人的 | 布尔值 | 0..1 | 属性 | 此属性控制通过 X509 提供程序的 Find() 接口访问 CryptoCertificateSlot 内容的可能性。  **标签：** atp.Status=draft |
| 写访问 | 布尔值 | 0..1 | 属性 | 此属性定义应用程序是否具有对 CryptoCertificate 的写访问权限 (True) 或只有读访问权限 (False)。  **标签：** atp.Status=draft |

**表 3.101：加密证书接口**

**3.11.4 加密提供者接口**

**[TPS\_MANI\_03257]** { DRAFT }**访问 Crypto Provider 的应用程序建模**d访问 Crypto Provider的应用程序软件被建模为AdaptiveApplicationSwComponentType ，它具有由 a 类型化的RPortPrototype

加密提供者接口。 c *( RS\_MANI\_00031 )*

请注意，如果应用程序需要访问 Crypto Provider 以执行无密钥操作，例如散列、随机数生成，则应使用CryptoProviderInterface 。对于需要密钥的加密转换，可以使用CryptoKeySlotInterface 。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **CryptoProvider接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***笔记*** | 这个元类提供了为 CryptoProvider 定义 PortInterface 的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=加密接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *CryptoInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.102：CryptoProvider 接口**

**3.11.5 Crypto TrustMaster 接口**

**[TPS\_MANI\_03258]** { DRAFT }**设计为 trust-master 的应用程序建模**d设计为 trust-master的应用程序软件被建模为AdaptiveApplicationSwComponentType ，其RPortPrototype由CryptoTrustMasterInterface键入。 c *( RS\_MANI\_00031 )*

应用程序需要 TrustMaster 权限来设置全局（机器范围的）信任根证书。注意：这样的证书可能不是私有的。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **CryptoTrustMaster 接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***笔记*** | 这个元类提供了为 TrustMaster 定义 PortInterface 的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=加密接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *CryptoInterface* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.103：CryptoTrustMaster 接口**

**3.11.6 加密证书与加密密钥槽的链接**

可以对加密证书和加密密钥槽之间的链接建模

具有元类SwcServiceDependency的应用程序设计，它聚合角色serviceNeeds和RoleBasedPortAssignment中的CryptoCertificateKeySlotNeeds ，它们引用由CryptoCertificateInterface键入的RPortPrototype和由CryptoKeySlotInterface键入的RPortPrototype 。

*Identifiable*

*ServiceNeeds*

*Identifiable*

AdaptiveSwcInternalBehavior

*AtpStructureElement*

*Identifiable*

*ServiceDependency*

SwcServiceDependency

*SwComponentType*

AdaptiveApplicationSwComponentType

*CryptoNeeds*

CryptoCertificateKeySlotNeeds

RoleBasedPortAssignment

+

[0..1]

role: Identifier

*AtpBlueprintable*

*AtpPrototype*

*PortPrototype*

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

CryptoKeySlotInterface

allocateShadowCopy: Boolean

[0..1]

+

+

cryptoAlgId: String

[0..1]

[0..1]

+

cryptoObjectType: CryptoObjectTypeEnum

[0..1]

+

slotCapacity: PositiveInteger

slotType: CryptoKeySlotTypeEnum

[0..1]

+

CryptoCertificateInterface

+

isPrivate: Boolean

[0..1]

[0..1]

+

writeAccess: Boolean

RPortPrototype

*AbstractRequiredPortPrototype*

*CryptoInterface*

portPrototype

+

0..1

serviceNeeds

+

0..1

«atpVariation,atpSplitable»

assignedPort

+

0..\*

«isOfType»

+

requiredInterface

0..1

redefines atpType

}

{

«atpVariation,atpSplitable»

internalBehavior

+

0..1

serviceDependency

+

0..\*

**图 3.64：应用程序设计中加密证书与加密密钥槽的链接**

**[TPS\_MANI\_03259]** { DRAFT }**将加密证书链接到加密密钥槽**d

**ServiceNeeds 种类**CryptoCertificateKeySlotNeeds **RoleBasedPortAssignment**有效角色：

* CryptoKeySlot接口[1]
* 加密证书接口[1]

**基于角色的数据分配**

不适用

**RepresentedPortGroups** N/A c *( RS\_MANI\_00031 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***CryptoNeeds*** （摘要） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***笔记*** | 指定 Crypto 配置的抽象需求。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* , *ServiceNeeds* | | | |
| ***子类*** | CryptoCertificateKeySlot 需要 | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.104：加密需求**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **CryptoCertificateKeySlot 需要** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign | | | |
| ***笔记*** | 这个元类应该被用来表示用这种 ServiceNeeds 建模的 SwcServiceDependecy 定义了 CryptoKeySlot 和 CryptoCertificate 之间的关系。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *CryptoNeeds* , *Identifiable* , *MultilanguageReferable* , *Referrable* , *ServiceNeeds* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.105：CryptoCertificateKeySlotNeeds**

下图3.65显示了如何使用SwcServiceDependency来创建 Crypto Certificate 和 Crypto KeySlot 之间的关系的示例。

MyApplication: AdaptiveApplicationSwComponentType

MyCryptoKeyPort:

RPortPrototype

MyCryptoKeyInterface:

CryptoKeySlotInterface

objectType = SymmetricKey

slotType = application

AdaptiveSwcInternalBehavior

:

SwcServiceDependency

:

CryptoCertificateKeySlotNeeds

:

RoleBasedPortAssignment

:

role = CryptoCertificateInterface

RoleBasedPortAssignment

:

role = CryptoKeySlotInterface

MyCryptoCertificateInterface:

CryptoCertificateInterface

MyCryptoCertificatePort:

RPortPrototype

port

+

+

requiredInterface

port

+

+

requiredInterface

**图 3.65：显示由 CryptoKeySlotInterface 键入的端口和由CryptoCertificateInterface键入的端口之间的链接的示例**

## 3.12 原始数据流接口

在某些情况下，应用软件必须能够处理通过通信通道发送的原始二进制数据流。显然，SOME/IP 序列化在这种情况下没有意义，就像AutosarDataType的建模一样，即ServiceInterface的创建。

*AUTOSAR 自适应平台*上提供了一种主动支持原始数据流传输要求的不同机制。

就应用软件而言，与原始数据流的交互基于使用由RawDataStreamClientInterface或RawDataStreamServerInterface键入的RPortPrototype 。

这种PortInterface既不支持也不要求任何具有建模数据类型的元素，即AutosarDataType 。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***AbstractRawDataStreamInterface*** （抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 此元类用作与原始数据流相关的 PortInterfaces 的抽象基类。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *识别*，*多语言可引用*，*可打包元素*，端口接口，*可引用* | | | |
| ***子类*** | 原始数据流客户端接口，原始数据流服务器接口 | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.106：AbstractRawDataStreamInterface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **原始数据流客户端接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 这个元类代表了客户端原始数据流的必要能力，即不经过任何序列化的数据流。每个 RawDataStreamClientInterface 都支持以下功能，无需进一步建模：   * connect：建立通讯通道 * shutdown：关闭通讯通道 * write：通过通信通道发送数据 * 读取：访问通信通道上的传入数据   **标签：**  atp.Status=草稿  atp.recommendedPackage=RawDataStreamInterfaces | | | |
| ***根据*** | *ARElement* , *ARObject* , *AbstractRawDataStreamInterface* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* ,  *AtpType* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.107：RawDataStreamClient 接口**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **原始数据流服务器接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 这个元类代表了服务器端原始数据流的必要能力，即不经过任何序列化的数据流。  每个 RawDataStreamServerInterface 都支持以下功能，无需进一步建模：   * waitForConnection：等待通信通道建立。 * shutdown：关闭通讯通道 * write：通过通信通道发送数据 * 读取：访问通信通道上的传入数据   **标签：**  atp.Status=草稿  atp.recommendedPackage=RawDataStreamInterfaces | | | |
| ***根据*** | *ARElement* , *ARObject* , *AbstractRawDataStreamInterface* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* ,  *AtpType* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *PortInterface* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.108：原始数据流服务器接口**

## 3.13 安全事件上报界面

在*AUTOSAR自适应平台上*，定义了一个专用的PortInterface ，用于应用层软件与AUTOSAR入侵检测系统管理器的交互。

这个抽象元类PortInterface的子类的名称是SecurityEventReportInterface 。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **安全事件报告接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 这个元类提供了定义一个端口接口的能力，用于在入侵检测系统的上下文中报告安全事件。  **标签：**  atp.Status=草稿  atp.recommendedPackage=SecurityEventReportInterfaces | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *识别*，*多语言可引用*，*可打包元素*，端口接口，*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.109：SecurityEventReportInterface**

**[TPS\_MANI\_01340]** { DRAFT } **SecurityEventReportInterface的语义**d每个由SecurityEventReportInterface键入的RPortPrototype都能够准确报告一个安全事件。 c *( RS\_MANI\_00068 )*

**[TPS\_MANI\_01338]** { DRAFT } **SecurityEventReportToSecurityEventDefinitionMapping的语义**d通过SecurityEventReportToSecurityEventDefinitionMapping创建特定安全事件与由 SecurityEventReportInterface 类型化的相应 RPortPrototype 之间的关联建模。 c *( RS\_MANI\_00068 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **安全事件报告到安全事件定义映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 该元类表示将用于报告安全事件的 PortPrototype 映射到应由该 PortPrototype 报告的实际安全事件的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=SecurityEventReportToSecurityEventDefinitionMappings | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 报道  安全事件 | 抽象必需端口  原型 | 0..1 | 参考 | 这标识了映射的安全事件。  **标签：** atp.Status=draft  **InstanceRef 由：** RPortInComposition InstanceRef实现 |
| 安全事件定义 | 安全事件定义 | 0..1 | 参考 | 该参考标识了安全事件的定义。  **标签：** atp.Status=draft |

**表 3.110：SecurityEventReportToSecurityEventDefinitionMapping**

这个元类将RPortPrototype映射到一个SecurityEventDefinition ，它本身就是所谓的安全提取的一部分。

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

SecurityEventReportInterface

SecurityEventDefinition

id: PositiveInteger

+

[0..1]

*ARElement*

IdsDesign

*ARElement*

*IdsCommonElement*

*ARElement*

SecurityEventReportToSecurityEventDefinitionMapping

RPortPrototype

*PortInCompositionTypeInstanceRef*

RPortInCompositionInstanceRef

*PortPrototype*

*AbstractRequiredPortPrototype*

reportedSecurityEvent

+

0..1

+

securityEventDefinition

0..1

+

targetRPort

0..1

{

redefines targetPort

}

«isOfType»

+

requiredInterface

0..1

}

redefines atpType

{

«atpVariation,atpSplitable»

element

+

0..\*

«instanceRef»

+

reportedSecurityEvent

0..1

**图 3.66： SecurityEventReportInterface和SecurityEventReportToSecurityEventDefinitionMapping的规范**

**[TPS\_MANI\_01339]** { DRAFT } **SecurityEventReportToSecurityEventDefinitionMapping的存在是由 AUTOSAR 方法驱动的**d SecurityEventReportToSecurityEventDefinitionMapping的存在是由 AUTOSAR 方法驱动的。在定义给定SecurityEventReportInterface的时间点，可能尚未定义相应的SecurityEventDefinition 。

所以以后可以添加这个关联。映射类存在的另一个原因是，一个特定的应用软件可能会报告不同 OEM 定义的不同的特定安全事件，这取决于应用软件的部署。

当然，安全事件的语义总是相同或至少具有可比性，安全事件的 Id 可能会根据特定项目或仅仅是因为不同的 OEM 对语义相同的安全事件使用不同的 Id 而发生变化。 . c *( RS\_MANI\_00068 )*

## 3.14 日志和跟踪接口

在*AUTOSAR 自适应平台上，定义了一个*名为LogAndTraceInterface的专用PortInterface ，用于应用层软件与 AUTOSAR 日志和跟踪功能集群的交互。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **LogAndTrace 接口** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface | | | |
| ***笔记*** | 此元类提供了实现 PortInterface 以支持 Logging 或 Tracing 的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=端口接口 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *识别*，*多语言可引用*，*可打包元素*，端口接口，*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.111：LogAndTrace 接口**

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

LogAndTraceInterface

**图 3.67： LogAndTraceInterface规范**

**[TPS\_MANI\_03284]** { DRAFT } **LogAndTraceInterface** d Each**的语义**

LogAndTraceInterface键入的RPortPrototype能够将日志信息转发到外部 Dlt 日志查看器。 c *( RS\_MANI\_00037 )* **[constr\_5290]** { DRAFT } **PPortPrototype不允许由LogAndTraceInterface键入**d PPortPrototype不允许在角色providedInterface中引用LogAndTraceInterface 。 c *()*

请注意，由LogAndTraceInterface键入的RPortPrototype到DltLogSink的映射在第9.6.2章的部署模型中进行了描述。

**[constr\_5291]** { DRAFT }**允许使用PortPrototype 。 logAndTraceMessageCollectionSet** d仅允许由LogAndTraceInterface键入的PortPrototype引用角色 logAndTraceMessageCollectionSet 中的LogAndTraceMessageCollectionSet 。 c *()*

**[TPS\_MANI\_03285]** {草案} **PortPrototype的语义。 logAndTraceMessageCollectionSet** d使用PortPrototype 。 logAndTraceMessageCollectionSet参考 可以在应用程序设计时描述应用程序将使用的日志或跟踪消息的集合。 c *( RS\_MANI\_00023 )*

## 3.15 应用交互端点

软件组件与外部世界的交互可以采取多种形式，例如面向服务的通信或与持久数据存储的交互。

需要将交互的正式表示描述为添加各种附加配置属性的锚点，这些配置属性在此上下文中有意义，但在PortInterface的上下文中没有意义。

AUTOSAR 元模型中有一个模型元素已经具有长期的传统，正是为了描述的目的： PortPrototype 。

以下子章节讨论通过PortPrototype与给定软件组件“外部”的软件的交互，重点关注需要不同方式进一步贡献模型元素以进行配置的不同类型的交互。

**3.15.1 面向服务的通信**

通过PortPrototype**进行**的面向服务的通信不支持**同时**需要和提供的通信端点的概念。这激发了 [ constr\_1473 ] 的存在。

**[constr\_1473]** { DRAFT }**不支持PRPortPrototype** d A ServiceInterface不应被PRPortPrototype在角色providedRequiredInterface中引用。 c *()*

**[TPS\_MANI\_01039]** { DRAFT }**提供服务的表示**d提供的**服务**应通过由ServiceInterface类型化的PPortPrototype建模。 c *( RS\_MANI\_00002 )*

**[TPS\_MANI\_01040]** { DRAFT }**所需服务**的表示d**所需服务**应通过由ServiceInterface类型化的RPortPrototype建模。 c *( RS\_MANI\_00002 )*

有关 [ constr\_1473 ] 基本原理的更多背景信息，请参阅 [ 1 ]。

请注意，在*AUTOSAR 自适应平台上使用服务发现*意味着相反的通信端**在设计上是预先未知的**。

因此，通常不可能使用AssemblySwConnector来建模两个通信端点之间的预定义关系，这些通信端点被建模为PortPrototype 。

与上述问题无关，仍然需要提供在不同级别上配置给定PortPrototype的方法：

* PortPrototype本身（即作为一个整体）可能需要定制，与相应的ServiceInterface聚合的元素的种类或数量无关。这方面在第3.15.4节中讨论。
* 可能需要为给定的PortPrototype配置相应ServiceInterface元素的使用。这方面在第3.15.5节中讨论。

**3.15.2 与持久键值存储的交互**

， PortPrototype与*持久键值存储交互的使用*受到的限制更少。换句话说，在适用的情况下使用PRPortPrototype是完全有效的。

**[TPS\_MANI\_01073]** { DRAFT } PersistencyKeyValueStorageInterface**类型的PortPrototype的语义**d **PersistencyKeyValueStorageInterface**类型的PortPrototype的特定子类的使用表明交互的预期语义：

* 使用RPortPrototype表示只能**从**持久存储中读取持久数据。
* PPortPrototype的使用表明持久数据只能**写入**持久存储。
* PRPortPrototype的使用表明可以**从持久存储读取持久数据，也可以将**其**写入**持久存储。

c *( RS\_MANI\_00027 )*

请注意PersistencyKeyValueStorageInterface在章节3.7.2中描述。

**3.15.3 与持久文件存储的交互**

**持久文件存储的**交互可能涉及由同一应用程序读取和写入文件的能力。因此，应支持由PersistencyFileStorageInterface类型化的PRPortPrototype的存在。

**[TPS\_MANI\_01081]** { DRAFT } PersistencyFileStorageInterface**类型的PortPrototype的语义**d **PersistencyFileStorageInterface**类型的PortPrototype的特定子类的使用表明交互的预期语义：

* RPortPrototype的使用表明可以**打开相应的文件进行读取访问**。
* PPortPrototype的使用表明可以**打开或创建相应的文件以进行写访问**。此外，还可以**删除**文件。
* PRPortPrototype的使用表明可以**打开或创建相应的文件以进行读写访问**。此外，还可以**删除**文件。

c *( RS\_MANI\_00027 )*

请注意PersistencyFileStorageInterface在章节中描述

3.7.3 .

**3.15.4 端口原型道具**

如前所述，在某些情况下，需要对PortPrototype的语义进行限定。为此，AUTOSAR 通常会定义某种类型的*props*类。同样的方法也适用于这种情况。

特别是， PortPrototype聚合了抽象元类PortPrototypeProps ，进而启动派生元类的继承树，这些元类能够相应地限定PortPrototype的子类。

**[constr\_3642]** { DRAFT }将**PortPrototypeProps聚合限制到自适应平台**d PortPrototypeProps的聚合仅在由rootSwComponentPrototype （可传递地）引用的SwComponentType的上下文中受支持。 c *()*

这种方法的一个例子是元类RPortPrototypeProps的定义，如图3.68所示。

**[constr\_3359]** { DRAFT } **RPortPrototypeProps仅与RPortPrototype相关**d **RPortPrototypeProps**应仅由角色portPrototypeProps中的RPortPrototype聚合。 c *()*

**[TPS\_MANI\_01057]** { DRAFT } **RPortPrototypeProps的语义。 searchIntention** d属性RPortPrototypeProps的值。 searchIntention明确了对相应报价的搜索是否应作为对 all 的搜索进行，或者作为对特定 ID 的搜索进行。

通常，搜索任何结果会产生一组报价，而搜索给定 id 只会产生一个报价。 c *( RS\_MANI\_00002 )*

*AtpBlueprintable*

*AtpPrototype*

*PortPrototype*

RPortPrototypeProps

searchIntention: SearchIntentionEnum

+

[0..1]

*PortPrototypeProps*

*AbstractRequiredPortPrototype*

RPortPrototype

«enumeration»

SearchIntentionEnum

searchForAllInstances

searchForSpecificInstance

+

portPrototypeProps

0..1

**图 3.68： RPortPrototype的RPortPrototypeProps建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***PortPrototypeProps*** （抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure | | | |
| ***笔记*** | 该元类表示定义端口原型子类语义的进一步限定的能力。  **标签：** atp.Status=draft | | | |
| ***根据*** | *AR对象* | | | |
| ***子类*** | RPortPrototypeProps | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.112：PortPrototypeProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **RPortPrototypeProps** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure | | | |
| ***笔记*** | RPort 的 PortPrototypeProps。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *PortPrototypeProps* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 搜索意图 | 搜索意图枚举 | 0..1 | 属性 | 该属性用于指定封闭软件组件的开发人员的意图，即是否应使用相应的 PortPrototype 来搜索特定服务实例或给定服务的所有实例。  请注意，此属性的值不会创建具有约束力的合同。实际的搜索行为被定义为服务实例清单的一部分。  **标签：** atp.Status=draft |

**表 3.113：RPortPrototypeProps**

|  |  |
| --- | --- |
| ***枚举*** | **搜索意图枚举** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign |
| ***笔记*** | 此元类允许从应用程序的角度定义专用搜索意图。  **标签：** atp.Status=draft |
| ***文字*** | ***描述*** |
| 搜索所有实例 | 此值表示搜索给定服务的所有实例的意图。  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |
| 搜索特定实例 | 此值表示搜索给定服务的特定实例的意图。  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |

**表 3.114：SearchIntentionEnum**

**3.15.5 端口原型 ComSpec**

**[TPS\_MANI\_01053]** { DRAFT } **ComSpecs 在*AUTOSAR 自适应平台上的使用***d用于输入给定PortPrototype的ServiceInterface元素的进一步限定方面是通过 ComSpecs 实现的，即抽象元类RPortComSpec和端口通信规范。

*AUTOSAR 自适应平台*上对 ComSpecs 的支持仅涵盖特定 ComSpec的**有限属性选择。** c *( RS\_MANI\_00002 )*

RPortComSpec或PPortComSpec支持的属性的详细信息。

通过元类TransformationComSpecProps的子类，可以在 ComSpec 的上下文中配置转换功能。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***TransformationComSpecProps*** （摘要） | | | |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Communication | | | |
| ***笔记*** | TransformationComSpecProps 包含特定于端口的转换器的所有属性。 | | | |
| ***根据*** | *可描述*的*AR对象* | | | |
| ***子类*** | EndToEndTransformationComSpecProps , UserDefinedTransformationComSpecProps | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.115：TransformationComSpecProps**

|  |  |
| --- | --- |
| ***班级*** | **UserDefinedTransformationComSpecProps** |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Communication |
| ***笔记*** | UserDefinedTransformationComSpecProps 用于为自定义转换器指定端口特定的配置属性。 |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **UserDefinedTransformationComSpecProps** | | | |
| ***根据*** | *ARObject* ,*可描述*的 , *TransformationComSpecProps* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.116: UserDefinedTransformationComSpecProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **EndToEndTransformationComSpecProps** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Transformer | | | |
| ***笔记*** | EndToEndTransformationIComSpecProps 类为 EndToEnd 转换器属性指定端口特定的配置属性。 | | | |
| ***根据*** | *ARObject* ,*可描述*的 , *TransformationComSpecProps* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| clearFromValid 到无效 | 布尔值 | 0..1 | 属性 | 清除从状态有效到状态无效转换的监视窗口。 |
| disableEndTo EndCheck | 布尔值 | 0..1 | 属性 | 禁用/启用 E2E 检查。 E2Eheader 从有效负载中移除，与该属性的设置无关。 |
| 禁用EndTo  结束状态  机器 | 布尔值 | 0..1 | 属性 | 禁用 E2EStateMachine（仅执行 E2E 检查功能） |
| e2e简介  兼容性  道具 | E2E 配置文件兼容性  道具 | 0..1 | 参考 | 参考 E2E 状态机的附加设置。 |
| maxDelta 计数器 | 正整数 | 0..1 | 属性 | 两个连续接收的有效消息的两个计数器值之间允许的最大差异。例如，如果接收器使用计数器 1 获取数据并且 Max DeltaCounter 为 3，那么在下一次接收时，接收器可以接受值为 2、3 或 4 的计数器。 |
| 最大错误状态  在里面 | 正整数 | 0..1 | 属性 | 在最后一次窗口大小检查中，为状态 E2E\_SM\_INIT 确定 ProfileStatus 等于 E2E\_P\_ERROR 的最大检查次数。  最小值为 0。 |
| maxErrorState 无效 | 正整数 | 0..1 | 属性 | 在最后一次窗口大小检查中，针对状态 E2E\_SM\_INVALID 确定 ProfileStatus 等于 E2E\_P\_ERROR 的最大检查次数。  最小值为 0。 |
| maxErrorState 有效 | 正整数 | 0..1 | 属性 | 在最后一次窗口大小检查中，为状态 E2E\_SM\_VALID 确定 ProfileStatus 等于 E2E\_P\_ERROR 的最大检查次数。  最小值为 0。 |
| minOkStateInit | 正整数 | 0..1 | 属性 | 在最后一次 WindowSize 检查中，针对状态 E2E\_SM\_INIT 确定 ProfileStatus 等于 E2E\_P\_OK 的最小检查次数。  最小值为 1。 |
| 最小正常状态  无效的 | 正整数 | 0..1 | 属性 | 在最后一次 WindowSize 检查中，针对状态 E2E\_SM\_INVALID 确定 ProfileStatus 等于 E2E\_P\_OK 的最小检查次数。  最小值为 1。 |
| minOkState 有效 | 正整数 | 0..1 | 属性 | 在最后一次 WindowSize 检查中，针对状态 E2E\_SM\_VALID 确定 ProfileStatus 等于 E2E\_P\_OK 的最小检查次数。  最小值为 1。 |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **EndToEndTransformationComSpecProps** | | |  |
| 窗口大小初始化 | 正整数 | 0..1 | 属性 | E2E状态机的状态Init监控窗口大小。 |
| 窗口大小  无效的 | 正整数 | 0..1 | 属性 | 状态监控窗口大小 对E2E状态机无效。 |
| 窗口大小有效 | 正整数 | 0..1 | 属性 | 状态监控窗口大小 对E2E状态机有效。 |

**表 3.117：EndToEndTransformationComSpecProps**

**[TPS\_MANI\_01327]** { DRAFT } **EndToEndTransformationComSpecProps的值。 disableEndToEndCheck与EndToEndTransformationComSpecProps的值。 disableEndToEndStateMachine** d如果属性的值

EndToEndTransformationComSpecProps 。 disableEndToEndCheck设置为 True，然后是属性EndToEndTransformationComSpecProps的值。

disableEndToEndStateMachine应被忽略。 c *( RS\_MANI\_00028 )*

**3.15.5.1 服务接口类型的端口原型**

**3.15.5.1.1 接收机 ComSpec**

ReceiverComSpec需要一个属性来指示封闭的AdaptiveApplicationSwComponentType是否有意图实际访问引用的dataElement 。该属性表示与身份和访问管理相关的安全特性 [ 14 ]。

具体来说，这方面通常被概括为软件的能力，即

AdaptiveApplicationSwComponentType表达了它相对于特定dataElement的能力。术语“能力”是身份和访问管理领域行话的一个组成部分。

然而，在身份和访问管理领域之外，这个术语有时很难被激发。可能的动机是AdaptiveApplicationSwComponentType表达了其实际访问dataElement的*意图*。

从这个角度来看，将事件添加到ServiceInterface的过程增加了使用dataElement的能力。但是使用ServiceInterface的软件是否真的打算访问dataElement可以通过ReceiverComSpec中名为receiverIntent的属性来表示。

*RPortComSpec*

*ReceiverComSpec*

[0..1]

+

receiverIntent: ReceiverIntentEnum

QueuedReceiverComSpec

VariableDataPrototype

*PortPrototype*

*AbstractRequiredPortPrototype*

*Describable*

*TransformationComSpecProps*

«enumeration»

ReceiverIntentEnum

willReceive

wontReceive

EndToEndTransformationComSpecProps

clearFromValidToInvalid: Boolean

[0..1]

+

+

disableEndToEndCheck: Boolean

[0..1]

disableEndToEndStateMachine: Boolean

[0..1]

+

+

maxDeltaCounter: PositiveInteger

[0..1]

maxErrorStateInit: PositiveInteger

+

[0..1]

+

maxErrorStateInvalid: PositiveInteger

[0..1]

maxErrorStateValid: PositiveInteger

[0..1]

+

+

minOkStateInit: PositiveInteger

[0..1]

+

minOkStateInvalid: PositiveInteger

[0..1]

+

minOkStateValid: PositiveInteger

[0..1]

[0..1]

+

windowSizeInit: PositiveInteger

+

windowSizeInvalid: PositiveInteger

[0..1]

+

windowSizeValid: PositiveInteger

[0..1]

*DataPrototype*

*AutosarDataPrototype*

UserDefinedTransformationComSpecProps

ReceptionComSpecProps

[0..1]

dataUpdatePeriod: TimeValue

+

timeout: TimeValue

[0..1]

+

+

transformationComSpecProps

0..\*

+

requiredComSpec

0..\*

+

receptionProps

0..1

dataElement

+

0..1

**图 3.69： *AUTOSAR 自适应平台*上的ReceiverComSpec建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***ReceiverComSpec*** （摘要） | | | |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Communication | | | |
| ***笔记*** | 与事件和字段通知器相关的接收器特定的通信属性（由 ServiceInterface 键入的 RPortPrototype）。 | | | |
| ***根据*** | *ARObject* , *RPortComSpec* | | | |
| ***子类*** | NonqueuedReceiverComSpec, QueuedReceiverComSpec | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 数据元素 | AutosarDataPrototype | 0..1 | 参考 | 这些属性所属的数据元素。 |
| 接收者意图 | 接收者意图枚举 | 0..1 | 属性 | 该属性表示接收者表达的意图。接收方可以决定声明该特定接收方明确不使用 ServiceInterface 的现有资源。这种说法的概念背景可能是由安全、安全等驱动的。  **标签：** atp.Status=draft |
| 接待道具 | 接待委员会规范  道具 | 0..1 | 聚合 | “这个聚合代表了在封闭的 ReceiverComSpec 上下文中的定义传输道具。 |
| 转型  ComSpecProps | 转型委员会  规格道具 | \* | 聚合 | 这引用了为数据转换定义端口特定配置的 TransformationComSpecProps。 |

**表 3.118：ReceiverComSpec**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **QueuedReceiverComSpec** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Communication | | | |
| ***笔记*** | 特定于排队接收的通信属性。 | | | |
| ***根据*** | *ARObject* , *RPortComSpec* , *ReceiverComSpec* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.119：QueuedReceiverComSpec**

**[TPS\_MANI\_01106]** { DRAFT }**事件接收者或字段通知者**d**的意图规范**ReceiverComSpec属性。 receiverIntent可用于指定软件是否真正打算访问引用的事件或字段通知器，或者它是否明确声明它对该值不感兴趣。 c *( RS\_MANI\_00034 )*

|  |  |
| --- | --- |
| ***枚举*** | **接收者意图枚举** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec |
| ***笔记*** | 此元类表示从给定事件接收器的角度指定如何使用给定 ServiceInterface 的意图。  **标签：** atp.Status=draft |
| ***文字*** | ***描述*** |
| 会收到 | 接收者将收到事件或字段通知。  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |
| 不会收到 | 接收者不会收到事件或字段通知。  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |

**表 3.120：ReceiverIntentEnum**

**[TPS\_MANI\_03132]** { DRAFT } **ReceiverComSpec中 E2E 属性的语义**d EndToEndTransformationComSpecProps应用于RPortPrototype 的规范-与事件或字段通知的端到端保护相关的特定配置选项。 c *( RS\_MANI\_00028 )*

**3.15.5.1.2 发件人 ComSpec**

SenderComSpec的建模方式与软件组件模板 [ 1 ]中描述的方式相同。

*PPortComSpec*

*SenderComSpec*

senderIntent: SenderIntentEnum

+

[0..1]

«atpVariation»

[0..1]

usesEndToEndProtection: Boolean

+

*ValueSpecification*

shortLabel: Identifier

+

[0..1]

VariableDataPrototype

*PortPrototype*

*AbstractProvidedPortPrototype*

QueuedSenderComSpec

«enumeration»

SenderIntentEnum

willSend

wontSend

*DataPrototype*

*AutosarDataPrototype*

FieldSenderComSpec

TransmissionComSpecProps

+

[0..1]

dataUpdatePeriod: TimeValue

[0..1]

minimumSendInterval: TimeValue

+

[0..1]

transmissionMode: TransmissionModeDefinitionEnum

+

+

transmissionProps

0..1

+

dataElement

0..1

+

initValue

1

providedComSpec

+

0..\*

.

**图 3.70： *AUTOSAR 自适应平台*上的SenderComSpec建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***SenderComSpec*** （摘要） | | | |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Communication | | | |
| ***笔记*** | 与事件和字段通知器相关的发送方端口（由 ServiceInterface 键入的 PPortPrototype）的通信属性。 | | | |
| ***根据*** | *ARObject* , *PPortComSpec* | | | |
| ***子类*** | FieldSenderComSpec , NonqueuedSenderComSpec, QueuedSenderComSpec | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 数据元素 | AutosarDataPrototype | 0..1 | 参考 | 这些服务质量属性适用的数据元素。 |
| 发件人意图 | 发件人意图枚举 | 0..1 | 属性 | 该属性代表发送者表达的意图。发送方可以决定声明该特定发送方明确不使用 ServiceInterface 的现有资源。这种说法的概念背景可能是由安全、安全等驱动的。  **标签：** atp.Status=draft |
| 传输道具 | 传输通信规范  道具 | 0..1 | 聚合 | 此聚合表示封闭 SenderComSpec 上下文中的定义传输道具。 |
| 使用EndToEnd  保护 | 布尔值 | 0..1 | 属性 | 这表明相应的数据元素是否应使用端到端保护进行传输。  **刻板印象：** atpVariation  **标签：** vh.latestBindingTime=preCompileTime |

**表 3.121：SenderComSpec**

**[TPS\_MANI\_03210]** { DRAFT }**事件特定通信属性的规范**d元类QueuedSenderComSpec可用于指定与发送方事件相关的通信属性。 c *( RS\_MANI\_00002 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **QueuedSenderComSpec** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Communication | | | |
| ***笔记*** | 特定于事件分发的通信属性（PPortPrototype、SenderReceiverInterface 和 dataElement 携带一个“事件”）。 | | | |
| ***根据*** | *ARObject* , *PPortComSpec* , *SenderComSpec* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.122：QueuedSenderComSpec**

**[TPS\_MANI\_03211]** { DRAFT }**字段特定通信属性的规范**d元类FieldSenderComSpec可用于指定与发送方字段相关的通信属性。 c *( RS\_MANI\_00002 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **FieldSenderComSpec** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec | | | |
| ***笔记*** | ServiceInterface 中定义的字段的端口特定通信属性。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *PPortComSpec* , *SenderComSpec* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 初始值 | 价值规范 | 1 | 聚合 | 在提供服务接口之前设置的字段的初始值。  **标签：** atp.Status=draft |

**表 3.123：FieldSenderComSpec**

**[TPS\_MANI\_03212]** { DRAFT }**字段初始值的规范**d属性FieldSenderComSpec 。 initValue可用于指定字段的初始值。 c *( RS\_MANI\_00002 )*

第 3.4.4 小节所述，字段在任何时候都具有有效值。 ara::com 确保提供字段的服务实现在字段对潜在消费者可见之前具有字段值。

15 ] 中有更详细的解释，其中定义了在调用 OfferService() 之前，应用程序代码应通过 Update() 至少设置一次初始字段值。

自定义代码（例如 ara::com 之上的组件模型）可以使用定义的initValue来调用 Field.Update(initValue)。

**[TPS\_MANI\_01107]** { DRAFT }**事件发送者或字段通知者的意图规范**d属性SenderComSpec 。 senderIntent可用于指定软件是否真正打算发送引用的事件或字段通知器。 c *( RS\_MANI\_00034 )*

|  |  |
| --- | --- |
| ***枚举*** | **发件人意图枚举** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec |
| ***笔记*** | 此元类表示从给定事件发送者的角度指定如何使用给定 ServiceInterface 的意图。  **标签：** atp.Status=draft |
| ***文字*** | ***描述*** |
| 将发送 | 发送者将发送事件或字段通知器。  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |
| 不会发送 | 发件人不会发送事件或字段通知器。  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |

**表 3.124：SenderIntentEnum**

**3.15.5.1.3 客户端 ComSpec**

ClientComSpec对*AUTOSAR 自适应平台*进行了扩展，即能够引用字段的 getter 和 setter 方法以及意图的定义。

*RPortComSpec*

ClientComSpec

[0..1]

clientIntent: ClientIntentEnum

+

+

[0..1]

endToEndCallResponseTimeout: TimeValue

*AtpStructureElement*

*Identifiable*

ClientServerOperation

+

[0..1]

fireAndForget: Boolean

*PortPrototype*

*AbstractRequiredPortPrototype*

*Describable*

*TransformationComSpecProps*

«enumeration»

ClientIntentEnum

willCall

wontCall

*AutosarDataPrototype*

Field

+

hasGetter: Boolean

hasNotifier: Boolean

+

hasSetter: Boolean

+

EndToEndTransformationComSpecProps

clearFromValidToInvalid: Boolean

+

[0..1]

+

disableEndToEndCheck: Boolean

[0..1]

[0..1]

+

disableEndToEndStateMachine: Boolean

+

maxDeltaCounter: PositiveInteger

[0..1]

+

maxErrorStateInit: PositiveInteger

[0..1]

+

maxErrorStateInvalid: PositiveInteger

[0..1]

+

maxErrorStateValid: PositiveInteger

[0..1]

minOkStateInit: PositiveInteger

[0..1]

+

[0..1]

+

minOkStateInvalid: PositiveInteger

[0..1]

minOkStateValid: PositiveInteger

+

[0..1]

windowSizeInit: PositiveInteger

+

+

[0..1]

windowSizeInvalid: PositiveInteger

[0..1]

windowSizeValid: PositiveInteger

+

UserDefinedTransformationComSpecProps

+

getter

0..1

transformationComSpecProps

+

0..\*

setter

+

0..1

+

requiredComSpec

0..\*

+

operation

0..1

**图 3.71：在*AUTOSAR 自适应平台上的*ClientComSpec建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **客户通讯规范** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Communication | | | |
| ***笔记*** | 与方法和字段 getter 和 setter 相关的特定于客户端的通信属性（由 ServiceInterface 键入的 RPortPrototype）。 | | | |
| ***根据*** | *ARObject* , *RPortComSpec* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 客户意图 | 客户端意图枚举 | 0..1 | 属性 | 该属性代表客户表达的意图。客户端可以决定声明该特定客户端明确不使用 ServiceInterface 的现有资源。这种说法的概念背景可能是由安全、安全等驱动的。  **标签：** atp.Status=draft |
| endToEndCall  回复  暂停 | 时间价值 | 0..1 | 属性 | 该属性定义了应用程序期望服务器响应的最大时间间隔（从发送调用调用到服务器响应到达之间的时间）。 |
| 吸气剂 | 场地 | 0..1 | 参考 | 此引用的存在表明 Client ComSpec 引用了 Field 的 getter。  **标签：** atp.Status=draft |
| 手术 | 客户端服务器操作 | 0..1 | 参考 | 这代表了相应的 ClientServerOperation。 |
| 二传手 | 场地 | 0..1 | 参考 | 此引用的存在表明 Client ComSpec 引用了 Field 的 setter。  **标签：** atp.Status=draft |
| 转型  ComSpecProps | 转型委员会  规格道具 | \* | 聚合 | 这引用了为数据转换定义端口特定配置的 TransformationComSpecProps。 |

**表 3.125：ClientComSpec**

**[TPS\_MANI\_01108]** { DRAFT }**方法s 或字段setter/getter** d属性ClientComSpec**的调用者的意图规范**。 clientIntent可用于指定软件是否真正打算调用被引用的方法或被引用字段的 getter/setter 。 c *( RS\_MANI\_00034 )*

|  |  |
| --- | --- |
| ***枚举*** | **客户端意图枚举** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec |
| ***笔记*** | 此元类表示从给定客户端的角度指定如何使用给定 ServiceInterface 的意图。  **标签：** atp.Status=draft |
| ***文字*** | ***描述*** |
| 将会通知 | 客户端将调用此方法。  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |
| 不会打电话 | 客户端不会调用此方法。  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |

**表 3.126：ClientIntentEnum**

**[TPS\_MANI\_01324]** { DRAFT } **ClientComSpec** d**中 E2E 属性的语义**EndToEndTransformationComSpecProps应用于与方法s的端到端保护相关的RPortPrototype特定配置选项的规范。 c *( RS\_MANI\_00028 )*

**3.15.5.1.4 服务器 ComSpec**

ServerComSpec对AUTOSAR 自适应平台进行了扩展，即能够引用字段的 getter 和 setter方法以及意图的定义。

**[TPS\_MANI\_01325]** { DRAFT } **ServerComSpec** d**中 E2E 属性的语义**EndToEndTransformationComSpecProps应用于与方法s的端到端保护相关的PPortPrototype特定配置选项的规范。 c *( RS\_MANI\_00028 )*

ServerComSpec

[0..1]

queueLength: PositiveInteger

+

*PPortComSpec*

*PortPrototype*

*AbstractProvidedPortPrototype*

*AutosarDataPrototype*

Field

+

hasGetter: Boolean

+

hasNotifier: Boolean

+

hasSetter: Boolean

*Describable*

*TransformationComSpecProps*

EndToEndTransformationComSpecProps

+

clearFromValidToInvalid: Boolean

[0..1]

disableEndToEndCheck: Boolean

[0..1]

+

+

disableEndToEndStateMachine: Boolean

[0..1]

+

maxDeltaCounter: PositiveInteger

[0..1]

+

maxErrorStateInit: PositiveInteger

[0..1]

maxErrorStateInvalid: PositiveInteger

[0..1]

+

+

maxErrorStateValid: PositiveInteger

[0..1]

+

minOkStateInit: PositiveInteger

[0..1]

minOkStateInvalid: PositiveInteger

+

[0..1]

[0..1]

minOkStateValid: PositiveInteger

+

windowSizeInit: PositiveInteger

+

[0..1]

[0..1]

windowSizeInvalid: PositiveInteger

+

[0..1]

+

windowSizeValid: PositiveInteger

UserDefinedTransformationComSpecProps

+

transformationComSpecProps

0..\*

+

providedComSpec

0..\*

getter

+

0..1

setter

+

0..1

**图 3.72：在*AUTOSAR 自适应平台上的*ServerComSpec建模**

|  |  |
| --- | --- |
| ***班级*** | **服务器通信规范** |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Communication |
| ***笔记*** | 与方法和字段 getter 和 setter 相关的特定于服务器的通信属性（由 ServiceInterface 键入的 PPortPrototype）。 |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **服务器通信规范** | |  |  |
| ***根据*** | *ARObject* , *PPortComSpec* | |  |  |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 吸气剂 | 场地 | 0..1 | 参考 | 此引用的存在表明 Server ComSpec 引用了 Field 的 getter。  **标签：** atp.Status=draft |
| 手术 | 客户端服务器操作 | 0..1 | 参考 | 这些通信属性适用于操作。 |
| 队列长度 | 正整数 | 0..1 | 属性 | 服务器端的呼叫队列长度。 |
| 二传手 | 场地 | 0..1 | 参考 | 这个引用的存在表明Server ComSpec 引用了一个Field 的setter。  **标签：** atp.Status=draft |
| 转型  ComSpecProps | 转型委员会  规格道具 | \* | 聚合 | 这引用了为数据转换定义端口特定配置的 TransformationComSpecProps。 |

**表 3.127：ServerComSpec**

**3.15.5.1.5 应用软件实现的通信行为**

为了支持周期性的数据收发（主要是为了能够周期性地调用EndToEnd变换），应用软件需要在其程序流程中实现周期。但也可能有其他原因导致应用程序需要处理周期性。

可以使用SenderComSpec和ReceiverComSpec提供的TransmissionComSpecProps和ReceptionComSpecProps定义传输和接收 API 的调用周期。

由于TransmissionComSpecProps和ReceptionComSpecProps定义了预期的通信行为，通信（网络）测量工具也可以利用这些值来验证应用程序代码是否真正正确地实现了属性。

属性ReceptionComSpecProps 。 dataUpdatePeriod定义接收应用程序应调用接收 API 以检查新数据的时间段。

属性ReceptionComSpecProps 。 timeout定义了应用程序应假定要接收的数据接收已超时的时间。

属性TransmissionComSpecProps 。 dataUpdatePeriod定义发送应用程序调用发送 API 的时间段。

属性TransmissionComSpecProps 。 minimumSendInterval和TransmissionComSpecProps 。传输模式定义影响传输行为的值，由应用程序代码实现。

属性End2EndEventProtectionProps 。 dataUpdatePeriod还定义了应用软件为 EndToEnd 保护实施的预期周期。

更具体地说，属性End2EndEventProtectionProps 。 dataUpdatePeriod表示网络透视图，尤其是当尚未定义应用程序软件且因此没有SenderComSpec和ReceiverComSpec可用时。

**3.15.5.2 持久性数据接口类型的端口原型**

**[TPS\_MANI\_01314]** { DRAFT }**由 PersistencyKeyValueStorageInterface 类型的 PortPrototype 属性的进一步限定 对于由**PersistencyKeyValueStorageInterface类型**的**PortPrototype s **，**可以**为所需**侧定义进一步的限定属性。

为此，提供了元类PersistencyDataRequiredComSpec 。 C

*( RS\_MANI\_00027 )*

PersistencyDataRequiredComSpec

*RPortComSpec*

*PortPrototype*

*AbstractRequiredPortPrototype*

*AutosarDataPrototype*

*PersistencyInterfaceElement*

PersistencyDataElement

*ValueSpecification*

+

shortLabel: Identifier

[0..1]

+

dataElement

1

+

requiredComSpec

0..\*

+

initValue

0..1

**图 3.73：ComSpec 的持久性建模**

**[TPS\_MANI\_01160]** { DRAFT } PersistencyDataElement**初始值的定义**d **PersistencyDataElement的初始值**的定义可以通过 PersistencyDataRequiredComSpec 在PortPrototype级别上完成。*初始值*c *(* RS\_MANI\_00027 *)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **PersistencyDataRequiredComSpec** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec | | | |
| ***笔记*** | 此元类表示定义端口特定属性以支持所需端数据持久性用例的能力。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *RPortComSpec* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 数据元素 | 持久性数据  元素 | 1 | 参考 | 该引用表示 PersistencyDataRequiredComSpec 适用的 PersistencyDataElement。  **标签：** atp.Status=draft |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **PersistencyDataRequiredComSpec** | | |  |
| 初始值 | 价值规范 | 0..1 | 聚合 | 此聚合表示由封闭 PersistencyDataRequiredComSpec 引用的 PersistencyDataElement 的初始值的定义  **标签：** atp.Status=draft |

**表 3.128：PersistencyDataRequiredComSpec**

**[constr\_10081]** { DRAFT } **PersistencyDataRequiredComSpec的定义中存在初始值**对于每个PersistencyDataRequiredComSpec ，如果属性dataElement的值。 updateStrategy设置为值delete ，然后属性PersistencyDataRequiredComSpec 。 initValue不得

存在。 c *()*

## 3.16 可执行文件

**[TPS\_MANI\_01010]** { DRAFT }**分层软件组件的根元素**d可执行文件在角色rootSwComponentPrototype中聚合元类RootSwComponentPrototype以为由参考RootSwComponentPrototype表示的任意嵌套的软件组件层次结构提供根元素。应用类型。 c *( RS\_MANI\_00001 , RS\_MANI\_00004 )*

请注意RootSwComponentPrototype通过Executable聚合是instanceRef 适用性的基础 对代表Executable功能的软件组件层次结构的引用。

这种建模方法类似于*AUTOSAR 经典平台*上的系统建模。

**[TPS\_MANI\_01279]** {草稿}**可执行文件的语义。报告行为**d属性可执行文件。报告行为应用于控制向执行管理报告封闭可执行文件的执行状态。如果该属性不存在，可执行文件应将其执行状态报告给执行管理。 c *( RS\_MANI\_00023 )*

**[constr\_1605]** { DRAFT }属性**Executable的标准化值。类别**d属性Executable的以下值。类别由 AUTOSAR 标准化：

* PLATFORM\_LEVEL： Executable代表平台级别的软件（即概念上位于中间件*级别）。*
* APPLICATION\_LEVEL：可执行文件代表应用程序级别的软件（即概念上位于中间件*之上）。*

### c ()

|  |  |  |
| --- | --- | --- |
| «枚举»  记录行为枚举 |  | «枚举»  ExecutionStateReportingBehaviorEnum |
| 使用记录不使用记录 | 报告执行状态不报告执行状态 |

*ARElement*

*AtpClassifier*

Executable

+

buildType: BuildTypeEnum

[0..1]

+

loggingBehavior: LoggingBehaviorEnum

[0..1]

[0..1]

+

minimumTimerGranularity: TimeValue

[0..1]

reportingBehavior: ExecutionStateReportingBehaviorEnum

+

[0..1]

version: StrongRevisionLabelString

+

*AtpPrototype*

*Identifiable*

RootSwComponentPrototype

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*SwComponentType*

«isOfType»

+

applicationType

1

{

}

redefines atpType

+

rootSwComponentPrototype

0..1

**图 3.74：可执行文件的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **可执行文件** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure | | | |
| ***笔记*** | 这个元类代表一个可执行程序。  **标签：**  atp.Status=草稿  atp.recommendedPackage=可执行文件 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpClassifier* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 构建类型 | 构建类型枚举 | 0..1 | 属性 | 此属性描述模块和/或平台实现的 buildType。**标签：** atp.Status=draft |
| 日志记录行为 | 记录行为枚举 | 0..1 | 属性 | 此属性指示封闭可执行文件的预期日志记录行为。**标签：** atp.Status=draft |
| minimumTimer 粒度 | 时间价值 | 0..1 | 属性 | 此属性描述可执行文件所需的最小计时器分辨率（一个刻度的 TimeValue）。  **标签：** atp.Status=draft |
| 举报行为 | 执行状态  报告行为  枚举 | 0..1 | 属性 | 此属性控制封闭可执行文件的执行状态报告行为。  **标签：** atp.Status=draft |
| rootSw 组件  原型 | RootSwComponent  原型 | 0..1 | 聚合 | 这表示 Executable 的根 SwCompositionPrototype。需要此聚合（与 SwComponentType 的直接引用相反）以支持在 Executable 上下文中定义 instanceRefs。  **标签：** atp.Status=draft |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **可执行文件** |  |  |  |
| 版本 | 强修订标签  细绳 | 0..1 | 属性 | 可执行文件的版本。  **标签：** atp.Status=draft |

**表 3.129：可执行文件**

|  |  |
| --- | --- |
| ***枚举*** | **构建类型枚举** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule 实现 |
| ***笔记*** | 这个枚举定义了一个软件模块可以实现的可能的构建类型。  **标签：** atp.Status=draft |
| ***文字*** | ***描述*** |
| buildTypeDebug | 用于调试。  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |
| buildTypeRelease | 用于释放。  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |

**表 3.130：BuildTypeEnum**

|  |  |
| --- | --- |
| ***枚举*** | **记录行为枚举** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure |
| ***笔记*** | 此枚举提供了用于控制 Executable 是否使用日志记录的选项。  **标签：** atp.Status=draft |
| ***文字*** | ***描述*** |
| 不使用日志记录 | 可执行文件表明其不使用日志记录的意图。  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |
| 使用记录 | 可执行文件表明它打算使用日志记录  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |

**表 3.131：LoggingBehaviorEnum**

|  |  |
| --- | --- |
| ***枚举*** | **ExecutionStateReportingBehaviorEnum** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure |
| ***笔记*** | 此枚举提供用于控制可执行文件如何向执行管理报告其执行状态的选项  **标签：** atp.Status=draft |
| ***文字*** | ***描述*** |
| 不报告执行状态 | 可执行文件不应向执行管理报告其执行状态。  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |

5

4

|  |  |
| --- | --- |
| ***枚举*** | **ExecutionStateReportingBehaviorEnum** |
| 报告执行状态 | 可执行文件应向执行管理报告其执行状态。  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |

**表 3.132：ExecutionStateReportingBehaviorEnum**

**[TPS\_MANI\_01271]** {草稿}**可执行文件的语义。 loggingBehavior** d属性可执行文件。 loggingBehavior应用于指示封闭的可执行文件是否使用日志记录。

如果该属性不存在，则Executable指示它不使用日志记录。 c *( RS\_MANI\_00023 , RS\_MANI\_00037 )*

**[TPS\_MANI\_03056]** { DRAFT }**可执行文件的可选性。 rootSwComponentPrototype** d聚合Executable 。 rootSwComponentPrototype已成为可选，以支持不使用任何面向服务的通信且不需要任何进一步形式化的*平台模块的实现。* c *( RS\_MANI\_00023 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **RootSwComponentPrototype** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure | | | |
| ***笔记*** | RootSwCompositionPrototype 表示可执行文件中软件组件的顶级组合。  包含的 SwComponentPrototypes 完全由它们的 SwComponentTypes 指定（包括 Port Prototypes、PortInterfaces、VariableDataPrototypes 等）。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *AtpFeature* , *AtpPrototype* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 申请类型 | 组件类型 | 1 | 特雷夫 | 此 SwComponentType 充当 RootSw ComponentPrototype 的类型。  **刻板印象：** isOfType**标签：** atp.Status=draft |

**表 3.133：RootSwComponentPrototype**

**[constr\_1492]** { DRAFT } **SwComponentType在角色Executable中引用。 rootSwComponentPrototype 。 applicationType** d角色Executable中引用的任何SwComponentType 。 rootSwComponentPrototype 。 applicationType或用于键入嵌套在角色Executable中引用的SwComponentType内的SwComponentPrototype 。 rootSwComponentPrototype 。 applicationType**只能**是CompositionSwComponentType或AdaptiveApplicationSwComponentType 。 \_ c *()*

图3.75中描述的示例举例说明了 [ constr\_1492 ] 的语句。该示例显示了由SwComponentPrototype组成的组件层次结构，这些 SwComponentPrototype由CompositionSwComponentType或AdaptiveApplicationSwComponentType进行类型化。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***SwComponentType*** （抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Components | | | |
| ***笔记*** | AUTOSAR 软件组件的基类。 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *识别*，*多语言可引用*，可打包元素，*可引用* | | | |
| ***子类*** | AdaptiveApplicationSwComponentType , *AtomicSwComponentType* , CompositionSwComponentType , ParameterSwComponentType | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 港口 | 端口原型 | \* | 聚合 | 此 SwComponent 类型可以通过其进行通信的 PortPrototypes。  PortPrototype 的聚合受可变性的影响，目的是支持 PortPrototype 的条件存在。  **刻板印象：** atpSplitable； atpVariation**标签：**  atp.Splitkey=port.shortName, port.variationPoint.short 标签  vh.latestBindingTime=preCompileTime |
| 港口集团 | 端口组 | \* | 聚合 | 作为此组件一部分的端口组。  **刻板印象：** atpVariation  **标签：** vh.latestBindingTime=preCompileTime |
| sw组件  文档 | 组件  文档 | 0..1 | 聚合 | 这会向 SwComponentType 添加一个文档。  **刻板印象：** atpSplitable； atpVariation**标签：**  atp.Splitkey=swComponentDocumentation, sw ComponentDocumentation.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=-10 |

**表 3.134: SwComponentType**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **CompositionSwComponentType** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Composition | | | |
| ***笔记*** | CompositionSwComponentType 聚合了 SwComponentPrototypes（反过来又由 Sw ComponentTypes 类型化）以及 SwConnector，用于主要将 SwComponentPrototypes 相互连接并连接到 CompositionSwComponentType 的表面。通过这种方式，可以创建软件组件的层次结构。  **标签：** atp.recommendedPackage=SwComponentTypes | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *识别*、*多语言可引用*、可*打包元素*、*可引用*、 *SwComponentType* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 零件 | 组件  原型 | \* | 聚合 | 作为此组合一部分的实例化组件。  **刻板印象：** atpSplitable； atpVariation**标签：**  atp.Splitkey=component.shortName, component.variation Point.shortLabel  vh.latestBindingTime=postBuild |
| 连接器 | 开关连接器 | \* | 聚合 | SwConnector 具有在 PortPrototype 之间建立连接的主要能力。它们可以在 CompositionSwComponentType 的上下文中扮演许多角色。细节由子类细化。  5 |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **CompositionSwComponentType** | |  |  |
|  |  |  |  | 4  SwConnector 的聚合受可变性的影响，目的是支持变体数据流。  该聚合被标记为 atpSplitable，以便在 ECU 集成期间允许在 ApplicationSwComponentTypes 和 ServiceSwComponentTypes 之间使用 AssemblySw 连接器扩展 ECU 提取。  **刻板印象：** atpSplitable； atpVariation**标签：**  atp.Splitkey=connector.shortName，connector.variation Point.shortLabel  vh.latestBindingTime=postBuild |
| 常量值映射 | 常量规范  映射集 | \* | 参考 | 引用将应用于 PPortComSpecs 和 RPortCom Spec 的 initValues 的 ConstantSpecificationMapping。  **刻板印象：** atpSplitable  **标签：** atp.Splitkey=constantValueMapping |
| 数据类型映射 | 数据类型映射集 | \* | 参考 | 对 ServiceInterfaces 中使用的 ApplicationDataTypes 应用的 DataTypeMapping 的引用。  **刻板印象：** atpSplitable  **标签：** atp.Splitkey=dataTypeMapping |

**表 3.135：CompositionSwComponentType**

3.75的左侧部分类似于元模型中的建模，但右侧部分使用简化的符号来说明软件组件的嵌套定义的外观。

constr\_1492 ]的一个明显后果是在*AUTOSAR 自适应平台上*（即在Executable的上下文中）不允许使用可以在*AUTOSAR 经典平台*上使用的软件组件。 rootSwComponentPrototype 。应用类型。

*AUTOSAR 自适应平台*上的软件组件主要通过ServiceInterface类型的PortPrototype与外界交互来定义。没有预见到内部行为的定义，除了一个小例外。

这种内部结构的缺乏，再加上有关头文件生成范围的决定，导致在源代码中实现软件组件的情况（与*AUTOSAR 经典平台上的情况相比*）少得多受到严格的隔离。

换句话说，没有真正的动机将软件组件彼此分开实现。尽管不鼓励，但可以直接在程序的 Main() 函数中实现给定可执行程序的所有软件组件。

Executable

CompositionSwComponentType

RootSwComponent

Prototype

rootSwComponentPrototype

ComponentPrototype

1

(

CompositionSwComponentType

)

ComponentPrototype

2

(

CompositionSwComponentType

)

ComponentPrototype

3

(

AdaptiveApplicationSwComponentType

)

ComponentPrototype

4

(

AdaptiveApplicationSwComponentType

)

ComponentPrototype

5

(

AdaptiveApplicationSwComponentType

)

applicationType

P

R

R

R

R

R

P

**可执行文件的可能结构示例**

## 3.17 复杂数据结构中的可选成员

**3.17.1 背景**

*AUTOSAR 自适应平台*支持使用[[7]](#footnote-7)TLVSOME/IP 传输层上的数据编码。 TLV 通常用于至少部分传输数据仅*可选*存在并填充有意义的值的情况。

换言之：数据结构的可选部分可能存在并在数据传输的一个实例中携带有意义的值，而在数据传输的另一个实例中完全缺失。

接收软件需要能够识别可选部分是否存在并相应地读取其值。

如果特定通信实例中不存在这种数据结构的可选部分，接收软件还需要能够有意义地执行。

因此，有必要能够准确地识别数据结构中对于特定数据传输实例可能变得可选的部分。

就 AUTOSAR 元模型而言，标识原则上可以附加到各个抽象级别：

**AutosarDataType**在这种情况下，主要仅用于通信目的的可选性仍然存在于数据类型的所有其他用途中。 AUTOSAR 仍然看到实现此选项的用例，尤其是在*AUTOSAR 经典平台的上下文中*。

诚然，对同一数据类型定义不同的可选配置可能会导致存在一堆结构相同的数据类型，这些数据类型只是在可选方面有所不同。不过，变异点的存在可能有助于减轻这种影响。

**ServiceInterface**在这种情况下，可选性是在实际需要的地方定义的。但是，原则上可以为由相同 AutosarDataType 键入的 DataPrototype 定义不同的可选性。

这将导致在同一ServiceInterface的上下文中定义 C++ 数据类型的工作量增加。在*AUTOSAR 经典平台*的上下文中已经确定了其他限制，最终使该选项不可行。

**ComSpec在这种情况下，与**ServiceInterface级别上的可选性定义相比，可选性的定义甚至会更加具体。

最重要的是，在绝大多数情况下定义可选性的任务是由 OEM 完成的，而 ComSpec 级别的模型定义需要SwComponentType的存在，并且这个定义在许多情况下是在供应商的域中.

由于这种考虑，AUTOSAR 选择了实现在AutosarDataType级别定义可选性的概念。

**3.17.2 可选性的定义**

如前所述，在Autosar级别上对可选性的具体定义-

DataType是通过复合AutosarDataType的各个元素的指示来完成的。

更具体地说，可选性的定义需要支持

AutosarDataType ，即ApplicationDataType级别以及CppImplementationDataType级别。

换句话说，如果使用带有可选元素的ApplicationDataType来定义ServiceInterface ，那么仍然需要将可选性向下传递到直接影响 AUTOSAR 模型的语言绑定的数据类型定义级别。

图3.76显示了ApplicationDataType级别的可选性建模。

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | «枚举»  ArraySizeSemanticsEnum | | 固定大小变量大小 | |  |
| |  | | --- | | «枚举»  ArraySizeHandlingEnum | | allIndicesSameArraySize allIndicesDifferentArraySize inheritFromArrayElementTypeSize | |

**ApplicationDataType级别的可选性建模**

**[TPS\_MANI\_01184]** { DRAFT }在ApplicationDataType**级别定义可选元素**d在**ApplicationDataType级别定义可选元素**的建模方法是设置属性ApplicationRecordElement 。 isOptional值 True。

如果该属性未设置或设置为值 False，则相应的Application-

RecordElement**应被认为是强制性**的。 c *( RS\_MANI\_00030 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **应用记录数据类型** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes | | | |
| ***笔记*** | 一种应用程序数据类型，可以分解为其他应用程序数据类型的原型。  **标签：** atp.recommendedPackage=ApplicationDataTypes | | | |
| ***根据*** | *ARElement* , *ARObject* , *ApplicationCompositeDataType* , *ApplicationDataType* , *AtpBlueprint* , *Atp*  *Blueprintable* , *AtpClassifier* , *AtpType* , *AutosarDataType* , *CollectableElement* , *Identifiable* , *Multilanguage Referrable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 元素  （订购） | 申请记录  元素 | \* | 聚合 | 指定记录的元素。  ApplicationRecordElement 的聚合受可变性的影响，目的是支持 ApplicationrecordData 类型内元素的条件存在。  **刻板印象：** atpVariation  **标签：** vh.latestBindingTime=preCompileTime |

**表 3.136：ApplicationRecordDataType**

|  |  |
| --- | --- |
| ***班级*** | **应用记录元素** |
| ***包裹*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes |
| ***笔记*** | 描述应用程序记录数据类型的一个特定元素的属性。 |
| ***根据*** | *ARObject* , *ApplicationCompositeElementDataPrototype* , *AtpFeature* , *AtpPrototype* , *DataPrototype* , *Identifiable* , *MultilanguageReferable* , *Referrable* |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **应用记录元素** | |  |  |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 是可选的 | 布尔值 | 0..1 | 属性 | 此属性表示将封闭的 ApplicationRecordElement 声明为可选的能力。这意味着，在运行时，ApplicationRecord 元素可能有也可能没有有效值，因此应被忽略。  底层运行时软件提供了在通信的发送端将 ApplicationRecordElement 设置为无效并在接收端确定其有效性的方法。 |

**表 3.137：ApplicationRecordElement**

最重要的是，仍然可以直接使用CppImplementationDataType来定义ServiceInterface 。

**[TPS\_MANI\_01185]** { DRAFT }**在 CppImplementationDataType 级别定义可选元素 d 在 CppImplementationDataType 级别定义可选元素**的建模**方法**是设置属性CppImplementationDataTypeElement 。 isOptional值 True。

如果该属性未设置或设置为值 False，则相应的CppImplementationDataTypeElement**应被认为是强制性**的。 c *( RS\_MANI\_00030 )*

属性NotAvailableValueSpecification 。 defaultPattern对*AUTOSAR 自适应平台上*DataPrototype的初始化没有任何意义。 [ TPS\_MANI\_01333 ]涵盖了这方面：

**[TPS\_MANI\_01333]** {**草稿**}**属性**NotAvailableValueSpecification **。 defaultPattern不适用**d属性NotAvailableValueSpecification 。自适应平台应忽略defaultPattern （如果已定义）。

忽略defaultPattern的基本原理是，从技术上讲，如果尚未收到可选数据，则应用程序代码无法访问它。 c *( RS\_MANI\_00030 )*

## 3.18 序列化属性

在 Adaptive AUTOSAR 中，序列化代码是根据服务描述生成的，并在应用程序上下文中编译和执行。

元类TransformationPropsToServiceInterfaceElementMapping定义ServiceInterface元素的序列化，并使用TransformationProps元素提供必要的序列化设置。

TransformationPropsToServiceInterfaceElementMapping的存在要求存在与应用程序组件目标文件链接到应用程序二进制文件的序列化代码。

SOME/IP 的序列化基于ServiceInterface规范。如果在ServiceInterface中使用的AutosarDataPrototype是复合的，例如结构、联合或数组，那么 SOME/IP 支持将放在序列化数据前面的长度字段的配置。

AUTOSAR 支持在两个不同级别上配置此类序列化设置：

* ServiceInterface元素级别上建模，该级别对ServiceInterface元素中所有可用的DataPrototype均有效。这种情况在第3.18.1章中有详细描述。
* 本章描述的DataPrototype级别的细粒度建模。这种情况在第3.18.2章中有详细描述。

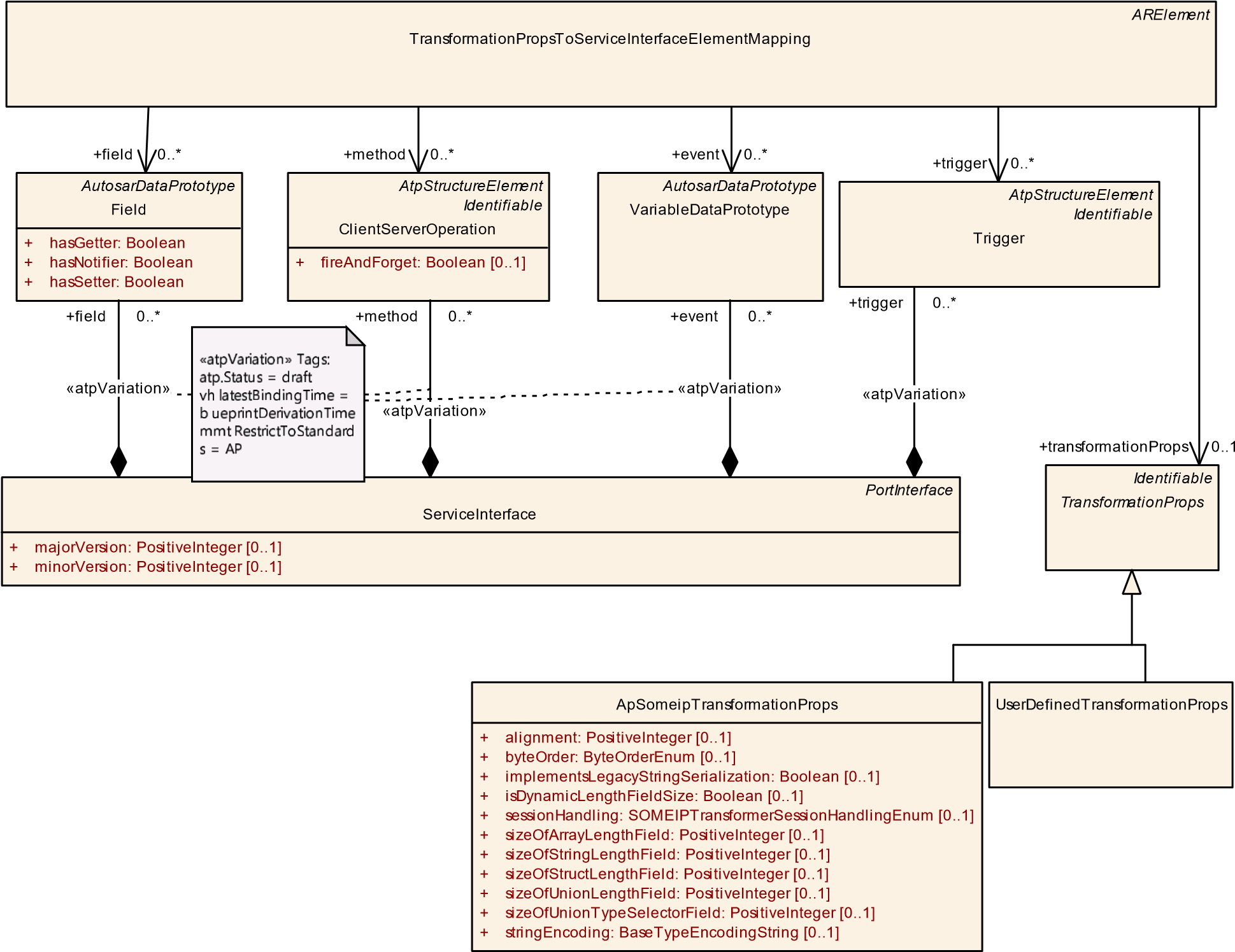
**3.18.1 序列化属性的默认值**

**[TPS\_MANI\_03101]** { DRAFT } **SOME/IP 序列化**d角色transformationProps中的TransformationPropsToServiceInterfaceElementMapping引用的ApSomeipTransformationProps元类提供了为角色事件中的TransformationPropsToServiceInterfaceElementMapping引用的ServiceInterface元素定义 SOME/IP 序列化设置的能力、触发器、方法或字段。 c *( RS\_MANI\_00008 , RS\_MANI\_00025 )*

**[constr\_3395]** { DRAFT } **TransformationPropsToServiceInterfaceElementMapping仅限于一个单一的ServiceInterface** d角色event 、 trigger 、 method或field中TransformationPropsToServiceInterfaceElementMapping引用的所有ServiceInterface元素应由角色event 、 trigger 、 method或field中的相同ServiceInterface聚合. c *()*

**[TPS\_MANI\_03288]** { DRAFT } **ApSomeipTransformationProps for trigger s** d如果将 ApSomeipTransformationProps分配给触发器，则只有属性sessionHandling与 SOME/IP 转换相关。 ApSomeipTransformationProps的所有其他属性与触发器无关，将被 SOME/IP 转换忽略。 c *( RS\_MANI\_00008 , RS\_MANI\_00025 )*

**[TPS\_MANI\_03103]** { DRAFT }**所有数组和映射长度字段的默认大小**d角色transformationProps中TransformationPropsToServiceInterfaceElementMapping引用的ApSomeipTransformationProps的属性sizeOfArrayLengthField定义了 SOME/IP 在所有可用可变大小数组（向量）前面生成的长度字段的大小)、固定大小的数组和关联映射定义在ServiceInterface元素中，这些元素由角色事件、方法或字段中的TransformationPropsToServiceInterfaceElementMapping引用。 c *( RS\_MANI\_00008 , RS\_MANI\_00025 )*



**图 3.77：序列化属性与 ServiceInterface 的关联**

**[TPS\_MANI\_03104]** { DRAFT }**所有结构长度字段的默认大小**d角色transformationProps中TransformationPropsToServiceInterfaceElementMapping引用的ApSomeipTransformationProps的属性sizeOfStructLengthField定义了 SOME/IP 在ServiceInterface元素中定义的所有可用结构之前生成的长度字段的大小由角色事件、方法或字段中的TransformationPropsToServiceInterfaceElementMapping引用。 c *( RS\_MANI\_00008 , RS\_MANI\_00025 )*

**[TPS\_MANI\_03117]** { DRAFT }**所有字符串长度字段的默认大小**d角色transformationProps中TransformationPropsToServiceInterfaceElementMapping引用的ApSomeipTransformationProps的属性sizeOfStringLengthField定义了 SOME/IP 在ServiceInterface元素中定义的所有可用字符串前面生成的长度字段的大小，这些字符串是由角色事件、方法或字段中的TransformationPropsToServiceInterfaceElementMapping引用。 c *( RS\_MANI\_00008 , RS\_MANI\_00025 )*

**[TPS\_MANI\_03105]** { DRAFT }**所有联合长度字段的默认大小**d角色transformationProps中TransformationPropsToServiceInterfaceElementMapping引用的ApSomeipTransformationProps的属性sizeOfUnionLengthField定义了 SOME/IP 在ServiceInterface元素中定义的所有可用联合之前生成的长度字段的大小由角色事件、方法或字段中的TransformationPropsToServiceInterfaceElementMapping引用。 c *( RS\_MANI\_00008 , RS\_MANI\_00025 )*

**[TPS\_MANI\_03106]** { DRAFT }**所有联合类型选择器字段的默认大小**d角色transformationProps中TransformationPropsToServiceInterfaceElementMapping引用的ApSomeipTransformationProps的属性sizeOfUnionTypeSelectorField定义了 SOME/IP 在ServiceInterface元素中定义的所有可用联合之前生成的类型字段的大小由角色事件、方法或字段中的TransformationPropsToServiceInterfaceElementMapping引用。 c *( RS\_MANI\_00008 , RS\_MANI\_00025 )*

**[TPS\_MANI\_03107]** { DRAFT }**所有动态 DataPrototype 的默认对齐d角色**transformationProps中TransformationPropsToServiceInterfaceElementMapping引用的ApSomeipTransformationProps的属性对齐定义了对齐目的的填充，将由 SOME/IP 在所有可变数据长度数据的序列化数据之后添加由角色事件、方法或字段中的TransformationPropsToServiceInterfaceElementMapping引用的ServiceInterface元素中定义的元素。 c *( RS\_MANI\_00008 , RS\_MANI\_00025 )*

**[TPS\_MANI\_03108]** { DRAFT }所有**DataPrototype的默认字节顺序**d角色transformationProps中TransformationPropsToServiceInterfaceElementMapping引用的ApSomeipTransformationProps的属性byteOrder定义了由角色事件中的TransformationPropsToServiceInterfaceElementMapping引用的ServiceInterface元素产生的序列化数据流中的字节顺序，方法或领域。 c *( RS\_MANI\_00008 , RS\_MANI\_00025 )*

**[constr\_1614]** { DRAFT }**属性TransformationPropsToServiceInterfaceElementMapping的存在。转换道具。 sessionHandling** d属性ApSomeipTransformationProps 。 sessionHandling仅在**引用**角色transformationProps中的相应ApSomeipTransformationProps的TransformationPropsToServiceInterfaceElementMapping不引用角色方法中的ClientServerOperation时才存在。 C

### ()

**[TPS\_MANI\_01210]** { DRAFT }**由类别 STRING** d的**CppImplementationDataType键入的所有**DataPrototype的**默认编码**角色transformationProps 中 TransformationPropsToServiceInterfaceElementMapping引用的ApSomeipTransformationProps**的**属性stringEncoding定义了由ServiceInterface元素引用的序列化数据流中的字符串编码TransformationPropsToServiceInterfaceElementMapping中的角色事件、方法或字段。 c *( RS\_MANI\_00008 , RS\_MANI\_00025 )*

**[constr\_1675]** { DRAFT }**存在属性ApSomeipTransformationProps 。 stringEncoding** d属性TransformationPropsToServiceInterfaceElementMapping 。转换道具。 stringEncoding应仅存在于由categorySTRING的CppImplementationDataType类型化的DataPrototype组成或包含的事件、方法或字段（由相同的TransformationPropsToServiceInterfaceElementMapping引用） 。 c *()*

请注意，有关ApSomeipTransformationProps的更多详细信息可以在第3.18.2章中找到。

**[constr\_1678]** { DRAFT }属性**ApSomeipTransformationProps的允许值。 stringEncoding** d由于 SOME/IP 消息格式 [ 7 ] 的定义中的技术限制，属性ApSomeipTransformationProps只有两个可能的值。允许字符串编码：

* UTF-8：UCS 转换格式 8
* 基于 16 位*代码单元的 Unicode代码点*字符编码[ 16 ]

### c ()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **ApSomeipTransformationProps** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::SerializationProperties | | | |
| ***笔记*** | 一些/IP 序列化属性。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* , *TransformationProps* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 结盟 | 正整数 | 0..1 | 属性 | 定义用于对齐目的的填充，将由 SOME/IP 转换器在可变数据长度数据元素的序列化数据之后添加。对齐应在位中指定。  **标签：** atp.Status=draft |
| 字节顺序 | 字节顺序枚举 | 0..1 | 属性 | 指定序列化数据流中数据的字节顺序。  **标签：** atp.Status=draft |
| 实现 LegacyString  序列化 | 布尔值 | 0..1 | 属性 | 该属性指示 SOME/IP 消息中的字符串不应根据字符串的 SOME/IP 规范进行序列化。  如果此属性设置为 true，则不应在有效负载中字符串的序列化中添加 BOM 和空终止。  如果此属性设置为 false（或未设置），则 BOM 和空终止应根据字符串的 SOME/IP 规范添加到有效负载中字符串的序列化中。  笔记！此属性在未来不安全，将在即将发布的 AUTOSAR 版本中删除！  **标签：** atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **ApSomeipTransformationProps** | | | |
| 是动态的  长度字段大小 | 布尔值 | 0..1 | 属性 | 此属性表示控制 TLV 编码的线路类型设置的能力。  如果该属性设置为 True，则应使用接线类型 5-7。  如果该属性不存在或设置为 False，则应使用导线类型 4。  **标签：** atp.Status=draft |
| 会议  处理 | SOMEIP变压器  会话处理枚举 | 0..1 | 属性 | 定义 SOME/IP 转换器是否应使用会话处理来进行发送方/接收方通信。  **标签：** atp.Status=draft |
| sizeOfArray  长度字段 | 正整数 | 0..1 | 属性 | 在可变大小数组（向量）、固定大小数组或 Associative\_Map 的情况下，为引用的 dataPrototype 配置 SOME/IP 序列化。它描述了将放在 SOME/IP 消息中的 Array 或 Associative\_Map 前面的长度字段的大小（以字节为单位）。  **标签：** atp.Status=draft |
| sizeOfString 长度字段 | 正整数 | 0..1 | 属性 | 在 String 的情况下为引用的 dataPrototype 配置 SOME/IP 序列化。它描述了将放在 SOME/IP 消息中字符串前面的长度字段的大小（以字节为单位）。  **标签：** atp.Status=draft |
| sizeOfStruct 长度字段 | 正整数 | 0..1 | 属性 | 在 Struct 的情况下为引用的 dataPrototype 配置 SOME/IP 序列化。它描述了将放在 SOME/IP 消息中结构前面的长度字段的大小（以字节为单位）。  **标签：** atp.Status=draft |
| sizeOfUnion 长度字段 | 正整数 | 0..1 | 属性 | 在 Union 的情况下为引用的 dataPrototype 配置 SOME/IP 序列化。它描述了将在 SOME/IP 消息中放在联合前面的长度字段的大小（以字节为单位）。  **标签：** atp.Status=draft |
| sizeOfUnion 类型选择器  场地 | 正整数 | 0..1 | 属性 | 在 Union 的情况下为引用的 dataPrototype 配置 SOME/IP 序列化。它描述了将在 SOME/IP 消息中放在联合前面的类型选择器字段的大小（以字节为单位）。  **标签：** atp.Status=draft |
| 字符串编码 | 基本类型编码  细绳 | 0..1 | 属性 | 如果是字符串，则为引用的 dataPrototype 配置 SOME/IP 序列化的编码。  **标签：** atp.Status=draft |

**表 3.138：ApSomeipTransformationProps**

**[TPS\_MANI\_03102]** { DRAFT } **UserDefined 序列**化d UserDefined-

Transformation- 引用的 TransformationProps元类

角色transformationProps中的PropsToServiceInterfaceElementMapping提供了为角色事件、方法或字段中的TransformationPropsToServiceInterfaceElementMapping引用的ServiceInterface元素定义用户定义序列化的能力。 c *( RS\_MANI\_00014 , RS\_MANI\_00025 )*

请注意， UserDefinedTransformationProps派生自元类

识别，因此具有描述特殊数据 ( sdg ) 的能力，通过这些数据可以以通用方式定义 AUTOSAR 模型的自定义结构扩展。有关特殊数据的更多信息，请参阅[ 6 ]。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **TransformationPropsToServiceInterfaceElementMapping** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure | | | |
| ***笔记*** | 此元类表示将 ServiceInterface 元素与 TransformationProps 关联的能力。服务接口的引用元素将根据 TransformationProps 中定义的设置进行序列化。  **标签：** atp.Status=draft  atp.recommendedPackage=TransformationPropsToServiceInterfaceElementMappings | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 事件 | 变量数据原型 | \* | 参考 | 这表示对一个 ServiceInterface 的一个或多个事件的引用。  **标签：** atp.Status=draft |
| 场地 | 场地 | \* | 参考 | 这表示对一个 ServiceInterface 的一个或多个字段的引用。  **标签：** atp.Status=draft |
| 方法 | 客户端服务器操作 | \* | 参考 | 这表示对一个 ServiceInterface 的一个或多个方法的引用。**标签：** atp.Status=draft |
| tlvDataId 定义 | TlvDataIdDefinitionSet | \* | 参考 | 此参考标识与封闭的 TransformationPropsToServiceInterface 映射相关的 TlvDataIdDefinitions。  **标签：** atp.Status=draft |
| 改造道具 | 转换道具 | 0..1 | 参考 | 这表示对适用的序列化属性的引用。  **标签：** atp.Status=draft |
| 扳机 | 扳机 | \* | 参考 | 这表示对一个 ServiceInterface 的一个或多个触发器的引用。  **标签：** atp.Status=draft |

**表 3.139：TransformationPropsToServiceInterfaceElementMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **UserDefinedTransformationProps** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Transformer | | | |
| ***笔记*** | UserDefinedTransformationProps 类指定用户定义的序列化程序的特定配置属性。 | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* , *TransformationProps* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.140: UserDefinedTransformationProps**

**3.18.2 序列化属性的单独定义**

**[TPS\_MANI\_03109]** { DRAFT } **DataPrototype级别上的TransformationProps覆盖了ServiceInterface级别上的TransformationProps设置**DataPrototype级别上的TransformationProps的细粒度建模覆盖了使用TransformationPropsToServiceInterfaceElementMapping描述的ServiceInterface级别上定义的TransformationProps设置. c *( RS\_MANI\_00025 )*

**[constr\_3361]** { DRAFT }**序列化设置的选择性定义**d如果为ServiceInterface （方法、字段、事件）的元素的复合 DataPrototype 定义了SomeipDataPrototypeTransformationProps ，并且如果存在引用someipTransformationProps ，则应定义定义引用someipTransformationProps的 SomeipDataPrototypeTransformationProps也适用于ServiceInterface元素的所有其他复合DataPrototype 。 c *()*

«enumeration»

SOMEIPTransformerSessionHandlingEnum

sessionHandlingActive

sessionHandlingInactive

ApSomeipTransformationProps

[0..1]

alignment: PositiveInteger

+

+

byteOrder: ByteOrderEnum

[0..1]

+

implementsLegacyStringSerialization: Boolean

[0..1]

[0..1]

isDynamicLengthFieldSize: Boolean

+

[0..1]

sessionHandling: SOMEIPTransformerSessionHandlingEnum

+

sizeOfArrayLengthField: PositiveInteger

+

[0..1]

[0..1]

sizeOfStringLengthField: PositiveInteger

+

sizeOfStructLengthField: PositiveInteger

+

[0..1]

sizeOfUnionLengthField: PositiveInteger

+

[0..1]

+

sizeOfUnionTypeSelectorField: PositiveInteger

[0..1]

+

stringEncoding: BaseTypeEncodingString

[0..1]

*ARElement*

SomeipDataPrototypeTransformationProps

*ARElement*

TransformationPropsSet

*Identifiable*

*TransformationProps*

«atpVariation»

SwDataDefProps

UserDefinedTransformationProps

DataPrototypeInServiceInterfaceRef

transformationProps

+

0..\*

+

dataPrototype

0..\*

+

someipTransformationProps

0..1

+

networkRepresentation

0..1

**图 3.78：关于 SOME/IP 序列化属性的概述**

|  |  |
| --- | --- |
| ***班级*** | **TransformationPropsSet** |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Transformer |
| ***笔记*** | TransformationProps 的集合。  **标签：** atp.recommendedPackage=TransformationPropsSets |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **TransformationPropsSet** | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 改造道具 | 转换道具 | \* | 聚合 | 变压器特定的配置属性。 |

**表 3.141：TransformationPropsSet**

|  |  |
| --- | --- |
| ***枚举*** | **SOMEIPTransformerSessionHandlingEnum** |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Transformer |
| ***笔记*** | 启用或禁用 SOME/IP 转换器的会话处理 |
| ***文字*** | ***描述*** |
| 会话处理活动 | SOME/IP Transformer 应使用会话处理  **标签：** atp.EnumerationLiteralIndex=0 |
| sessionHandling 不活动 | SOME/IP 转换器不使用会话处理  **标签：** atp.EnumerationLiteralIndex=1 |

**表 3.142：SOMEIPTransformerSessionHandlingEnum**

**[TPS\_MANI\_03070]** { DRAFT }**所选数组或映射的长度字段的大小**d ApSomeipTransformationProps 的属性 sizeOfArrayLengthField定义由SOME /IP 在可变大小数组（向量）、固定大小数组或associative\_map 为其定义了SomeipDataPrototypeTransformationProps ，即在聚合的 DataPrototypeInServiceInterfaceRef 中引用的可变大小数组（向量）、固定大小数组或associative\_map 。 c *( RS\_MANI\_00008 , RS\_MANI\_00024 )*

**[constr\_3353]** { DRAFT }**限制ApSomeipTransformationProps的使用。 sizeOfArrayLengthField** d属性sizeOfArrayLengthField的值应为 0、1、2 或 4。 c *()*

**[constr\_3447]** {草案} **ApSomeipTransformationProps 。 sizeOfArrayLengthField等于 0** d只有在聚合DataPrototypeInServiceInterfaceRef中引用了为其定义了SomeipDataPrototypeTransformationProps的固定大小数组时，才允许使用sizeOfArrayLengthField值0 。 c *()*

sizeOfArrayLengthField设置支持使用附加数组元素对此类数组进行向后兼容扩展。

**[TPS\_MANI\_03071]** { DRAFT }**所选结构的长度字段的大小**d ApSomeipTransformationProps 的属性 sizeOfStructLengthField定义了由SOME/IP 在结构前面生成的长度字段的大小，

SomeipDataPrototypeTransformationProps已定义，即在聚合的 DataPrototypeInServiceInterfaceRef 中引用的结构。 c *( RS\_MANI\_00008 , RS\_MANI\_00024 )*

**[constr\_3354]** { DRAFT }**限制ApSomeipTransformationProps的使用。 sizeOfStructLengthField** d属性sizeOfStructLengthField的值应为 0、1、2 或 4。 c *()*

**[TPS\_MANI\_03116]** { DRAFT }**所选字符串的长度字段的大小**d ApSomeipTransformationProps 的属性 sizeOfStringLengthField定义了SOME /IP 在为其定义SomeipDataPrototypeTransformationProps的字符串前面生成的长度字段的大小，即在聚合的DataPrototypeInServiceInterfaceRef中引用。 c *( RS\_MANI\_00008 , RS\_MANI\_00024 )*

**[constr\_3372]** { DRAFT }**限制ApSomeipTransformationProps的使用。 sizeOfStringLengthField** d属性sizeOfStringLengthField的值应为 0、1、2 或 4。 c *()*

**[TPS\_MANI\_03217]** { DRAFT }**所选字符串的在线编码**d ApSomeipTransformationProps的属性stringEncoding定义了为其定义 SomeipDataPrototypeTransformationProps 的字符串的在线编码，即在聚合的 DataPrototypeInServiceInterfaceRef 中引用的字符串。 c *( RS\_MANI\_00008 , RS\_MANI\_00024 )*

**[TPS\_MANI\_03072]** { DRAFT }**所选联合**d的长度字段大小ApSomeipTransformationProps 的属性 sizeOfUnionLengthField定义了由SOME/IP 在联合前面生成的长度字段的大小，

SomeipDataPrototypeTransformationProps已定义，即在聚合的 DataPrototypeInServiceInterfaceRef 中引用的联合。 c *( RS\_MANI\_00008 , RS\_MANI\_00024 )*

**[constr\_3355]** { DRAFT } **ApSomeipTransformation的使用限制-**

**道具。 sizeOfUnionLengthField** d属性sizeOfUnionLengthField的值应为 0、1、2 或 4。 c *()*

**[TPS\_MANI\_03073]** { DRAFT }**动态 DataPrototype的**对齐d ApSomeipTransformationProps的属性对齐定义了用于对齐目的的填充，它将由 SOME/IP 在定义了SomeipDataPrototypeTransformationProps的可变数据长度数据元素的序列化数据之后添加，即在聚合的 DataPrototypeInServiceInterfaceRef 中引用的可变数据长度DataPrototype 。 c *( RS\_MANI\_00008 , RS\_MANI\_00024 )*

**[constr\_3356]** { DRAFT }**限制ApSomeipTransformationProps的使用。对齐**d属性对齐的值应为 8、16、32、64、128 或 256。 c *()*

**[TPS\_MANI\_03074]** { DRAFT }**所选联合的类型选择器字段的大小**d ApSomeipTransformationProps 的属性 sizeOfUnionTypeSelectorField定义了由SOME/IP 在定义了SomeipDataPrototypeTransformationProps的联合之前生成的类型选择器字段的大小，即联合在聚合的DataPrototypeInServiceInterfaceRef中引用。 c *( RS\_MANI\_00008 , RS\_MANI\_00024 )*

**[constr\_3357]** { DRAFT }**限制ApSomeipTransformationProps的使用。 sizeOfUnionTypeSelectorField** d属性sizeOfUnionTypeSelectorField的值应为 1、2 或 4。 c *()*

**[TPS\_MANI\_03235]** { DRAFT } **ApSomeipTransformationProps的使用。 sessionHandling** d在 SomeipDataPrototypeTransformationProps引用的ApSomeipTransformationProps中定义的sessionHandling属性与SomeipDataPrototypeTransformationProps中引用的DataPrototype无关。 c *( RS\_MANI\_00008 , RS\_MANI\_00024 )*

**[TPS\_MANI\_03278]** { DRAFT } **ApSomeipTransformationProps的使用。 byteOrder** d在 SomeipDataPrototypeTransformationProps引用的 ApSomeipTransformationProps 中定义的 byteOrder 属性对于SomeipDataPrototypeTransformationProps适用的dataPrototype应被忽略。 c *( RS\_MANI\_00008 , RS\_MANI\_00024 )*

byteOrder属性定义了 SOME/IP 消息中完整有效负载的字节顺序，因此通过TransformationPropsToServiceInterfaceElementMapping进行的配置是定义ServiceInterface元素的字节顺序的唯一有效选项。请注意，根据 SOME/IP，标头以网络字节顺序（Big Endian）编码。

sessionHandling属性用于激活/停用事件/通知程序的 SessionHandling，因此通过TransformationPropsToServiceInterfaceElementMapping使用是唯一有效的配置选项。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **SomeipDataPrototypeTransformationProps** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::SerializationProperties | | | |
| ***笔记*** | 此元类表示为给定 DataPrototype 专门为 SOME/IP 序列化定义数据转换道具的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=SomeipDataPrototypeTransformationPropss | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 数据原型 | DataPrototypeInService  接口参考 | \* | 聚合 | SomeipDataPrototypeTransformationProps 中的设置对其有效的 DataPrototype 集合。出于重用的原因，SomeipDataPrototypeTransformation Props 能够聚合多个 DataPrototype。  **标签：** atp.Status=draft |
| 网络  表示 | SwDataDefProps | 0..1 | 聚合 | 引用的原语 DataPrototype 的实际网络表示的可选规范。如果提供了网络表示，则 SwDataDefProps 中可用的 baseType 应用作序列化/反序列化的输入。如果网络  没有提供表示，则 AbstractImplementationDataType 的 baseType 将用于序列化/反序列化。  **标签：** atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **SomeipDataPrototypeTransformationProps** | | | |
| someip 转换  道具 | 阿普索普  转换道具 | 0..1 | 参考 | 此参考表示专门为 SOME/IP 序列化定义数据转换道具的能力。  **标签：** atp.Status=draft |

**表 3.143：SomeipDataPrototypeTransformationProps**

图10.17描述了在由ApplicationDataType或CppImplementationDataType类型化的PortInterface上下文中对DataPrototype的引用的建模。

**[TPS\_MANI\_01136]** { DRAFT } **AutosarDataPrototype是**DataPrototypeInServiceInterfaceRef的目标d如果 DataPrototypeInServiceInterfaceRef 的目标是**AutosarDataPrototype**角色DataPrototypeInServiceInterfaceRef 。 dataPrototype应用于**独立**于AutosarDataPrototype是由ApplicationDataType还是CppImplementationDataType类型化，甚至**独立**于AutosarDataPrototype的AutosarDataType是否表示复合数据类型来描述引用。 c *( RS\_MANI\_00008 , RS\_MANI\_00024 )*

**[TPS\_MANI\_01137]** { DRAFT }**适用于 DataPrototypeInServiceInterfaceRef 的用例d**表3.144包含使用 DataPrototypeInServiceInterfaceRef 的完整用例列表。 c *( RS\_MANI\_00008 , RS\_MANI\_00024 )*

|  |  |
| --- | --- |
| **用例** | **角色** |
| 应用程序键入的AutosarDataPrototype-  数据类型 | 数据原型 |
| ApplicationCompositeDataType键入的AutosarDataPrototype中的DataPrototype | 数据原型 |
| CppImplementationDataType类型化的AutosarDataPrototype | 数据原型 |
| CppImplementationDataType键入的AutosarDataPrototype中的DataPrototype | elementInImplDatatype |

**表 3.144：使用DataPrototypeInServiceInterfaceRef的可能用例**

仔细观察表3.144应该清楚，在同一个 DataPrototypeInServiceInterfaceRef 的上下文中，不存在同时使用两个角色dataPrototype和 elementInImplDatatype 的有效用例。

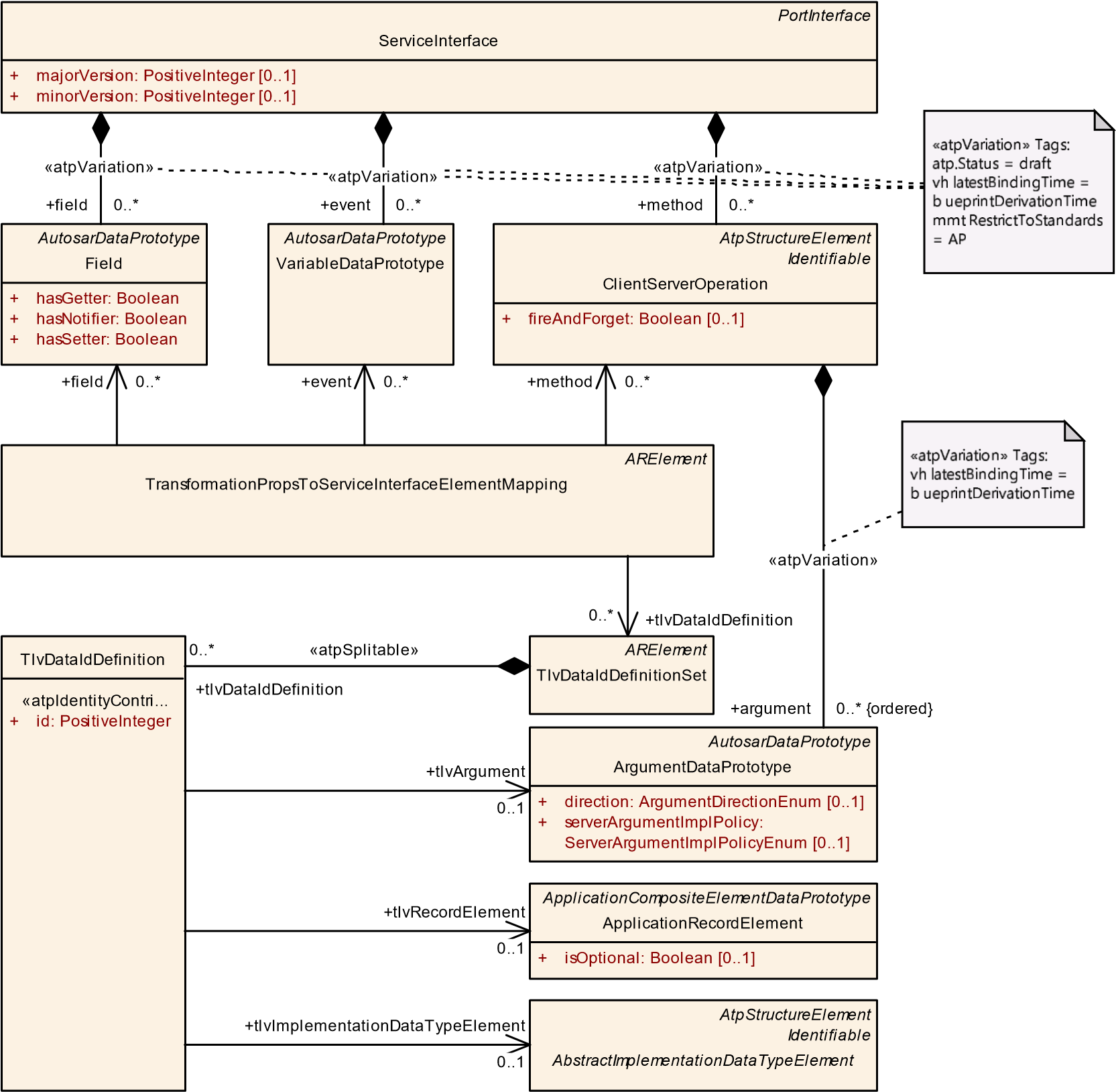
**[constr\_1551]** { DRAFT } **DataPrototypeInServiceInterfaceRef的存在。 dataPrototype与DataPrototypeInServiceInterfaceRef 。 elementInImplDatatype** d对于每个给定的DataPrototypeInServiceInterfaceRef ，聚合DataPrototypeInServiceInterfaceRef 。 dataPrototype或DataPrototypeInServiceInterfaceRef 。 elementInImplDatatype应存在。 c *()*

SomeipDataPrototypeTransformationProps的使用。 networkRepresentation在 [TPS\_SYST\_02136] 和 [TPS\_SYST\_02137]的系统模板 [ 17 ] 中有更详细的解释。

**3.18.3 TLV 属性的分配**

**3.18.3.1 TLV 数据 ID 的分配**

**[TPS\_MANI\_01097]** { DRAFT } **TLV 数据 id的分配**d TLV 数据 id 的分配是在TransformationPropsToServiceInterfaceElementMapping规范的上下文中完成的，即通过属性TransformationPropsToServiceInterfaceElementMapping 。 tlvDataIdDefinition 。身份证。 c *( RS\_MANI\_00030 )*



**图 3.79：TLV 数据 id 的建模**

这种方法受益于TlvDataIdDefinition能够创建对相关模型元素的引用这一事实。

因此，TLV 数据 id 的分配是通过创建此类引用并通过属性TlvDataIdDefinition 为其分配 TLV 数据 id 来完成的。身份证。

请注意，对于具有至少一个可选成员的整个数据结构，TLV 数据 ID 的分配是强制性的。简而言之，这个结论（也得到 [PRS\_SOMEIP\_00230] 的支持，参见 [ 7 ]）是 [ constr\_1594 ] 和 [ constr\_1595 ] 存在的动机。

请进一步注意，TLV 数据 ID 的分配不限于具有可选成员的数据结构。还有一个用例支持以任意顺序发送特定数据结构的元素，即使没有一个元素被认为是可选的。

**[TPS\_MANI\_01270]** { DRAFT }**从 TransformationPropsToServiceInterfaceElementMapping 到 TlvDataIdDefinitionSet 的引用 d 从**TransformationPropsToServiceInterfaceElementMapping到TlvDataIdDefinitionSet 的引用意味着**模型的创建**者可以决定是否应该假设全局范围或是否需要为具体情况。 c *( RS\_MANI\_00030 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **TlvDataIdDefinitionSet** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Transformer | | | |
| ***笔记*** | 此元类充当 TlvDataIdDefinitions 的容器，以在给定上下文中使用  **标签：** atp.recommendedPackage=TlvDataDefinitionSets | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| tlvDataId 定义 | TlvDataId 定义 | \* | 聚合 | 此聚合表示由 TlvDataIdDefinitionSet 聚合的 TlVDataTid 定义的集合  **刻板印象：** atpSplitable  **标签：** atp.Splitkey=tlvDataIdDefinition.id |

**表 3.145：TlvDataIdDefinitionSet**

**[constr\_1594]** { DRAFT }**将 TLV 数据 ID 一致分配给应用程序-**

**RecordDataType** d对于直接成员设置属性 ApplicationRecordElement 的每个ApplicationRecordDataType 。 isOptional值 True 对ApplicationRecordDataType的**所有直接成员**的引用应基于TlvDataIdDefinition的定义创建。 c *()*

**[constr\_1595]** { DRAFT }**将 TLV 数据 ID 一致分配给CppImplementationDataType或CppImplementationDataTypeElement** d对于categorySTRUCTURE的每个CppImplementationDataType ，其中直接成员设置属性CppImplementationDataTypeElement 。 isOptional值 True 对 CppImplementationDataType 的**所有直接成员**的引用应基于 TlvDataIdDefinition 的定义创建。 c *()*

TlvDataIdDefinition的定义不限于定义可选元素的场景。也可以为任意方法或数据结构定义TlvDataIdDefinition 。

一个典型的用例可能是为将来的扩展准备参数列表或子元素。但是，如果引用了一个参数或子元素，则有必要定义从TlvDataIdDefinition到所有其他参数或子元素的引用。

**[constr\_1593]** { DRAFT }**一组TlvDataId-存在的完整性**

**定义。 tlvArgument s** d如果引用TlvDataIdDefinition 。 tlvArgument存在于给定ClientServerOperation的一个参数，然后是TlvDataIdDefinition 。 tlvArgument应存在于给定ClientServerOperation**的所有**参数s 中，并且所有受影响的TlvDataIdDefinition应由相同的TransformationPropsToServiceInterfaceElementMapping聚合。 c *()*

尽管可以在 TLV 配置的定义中应用优化，以便 TLV 配置可以定义为方向特定的[[8]](#footnote-8)，但 AUTOSAR 定义不应使用这种混合 TLV 配置。

**[constr\_1603]** { DRAFT }**一组TlvDataIdDefinition存在的完整性。 tlvRecordElement s** d如果引用TlvDataIdDefinition 。 tlvRecordElement存在于给定ApplicationRecordDataType的一个元素，然后是TlvDataIdDefinition 。 tlvRecordElement应存在于给定ApplicationRecordDataType**的所有**元素中，并且所有受影响的TlvDataIdDefinition应由相同的TransformationPropsToServiceInterfaceElementMapping聚合。 c *()*

**[constr\_1604]** { DRAFT }**一组Tlv-存在的完整性**

**数据标识定义。 tlvImplementationDataTypeElement s** d如果引用TlvDataIdDefinition 。 tlvImplementationDataTypeElement存在于给定CppImplementationDataType或CppImplementationDataTypeElement的一个子元素，然后是 TlvDataIdDefinition 。 tlvImplementationDataTypeElement应存在**于**给定CppImplementationDataType或CppImplementationDataTypeElement的所有subElement中，并且所有受影响的TlvDataIdDefinition应由相同的TransformationPropsToServiceInterfaceElementMapping聚合。 c *()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **TlvDataId 定义** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Transformer | | | |
| ***笔记*** | 此元类表示定义 tlvDataId 的能力。 | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| ID | 正整数 | 1 | 属性 | 该属性代表定义了 TlvDataId 的值  **刻板印象：** atpIdentityContributor |
| tlv参数 | 参数数据原型 | 0..1 | 参考 | 此引用将 tlvDataId 分配给 ClientServerOperation 的给定参数。 |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **TlvDataId 定义** |  |  |  |
| 电视  执行  数据类型  元素 | 抽象实现  数据类型元素 | 0..1 | 参考 | 此引用将 TLV 数据 id 的定义与给定的 AbstractImplementationDataTypeElement 相关联。 |
| tlvRecord 元素 | 申请记录  元素 | 0..1 | 参考 | 此参考将 TLV 数据 id 的定义与给定的 ApplicationRecordElement 相关联。 |

**表 3.146：TlvDataIdDefinition**

TlvDataIdDefinition的定义。 id的目的是提供明确识别参数或子元素的方法。为此， id的值需要在各自的上下文中是唯一的。

**[constr\_1596]** { DRAFT } **TlvDataId值的唯一性范围-**

**定义。 id用于引用ArgumentDataPrototype** d对于 所有Tlv-

引用给定ArgumentDataPrototype的DataIdDefinition

ClientServerOperation中的角色tlvArgument ，属性TlvDataIdDefinition 。 id**应该存在并且每个通信方向都有一个唯一的值**，即在所有的集合的上下文中。

* 参数s，其中属性方向设置为in或inout
* 参数s，其中属性方向设置为out或inout
* 各个封闭ClientServerOperation的参数s ，其中属性方向设置为inout （如果该方法**只有**参数s ，其中属性方向设置为inout ） 。 c *()*

constr\_1596 ]存在的理由：属性方向设置为in或inout的参数s 永远不会在与属性方向设置为out或inout的参数s相同的 SOME/IP 消息中发送。

**[constr\_1597]** { DRAFT } **TlvDataIdDefinition值的唯一性范围。对ApplicationRecordElement的引用的id** d对于所有在角色tlvRecordElement中引用给定ApplicationDataType的ApplicationRecordElement的TlvDataIdDefinition属性TlvDataIdDefinition 。 id应存在并在各自封闭的ApplicationRecordDataType的上下文中**具有唯一值**。 c *()*

**[constr\_1598]** { DRAFT } **TlvDataId值的唯一性范围-**

**定义。对CppImplementationDataTypeElement的引用的id**对于所有在角色tlvImplementationDataTypeElement中引用给定CppImplementationDataTypeElement的CppImplementationDataTypeElement的TlvDataIdDefinition属性TlvDataIdDefinition 。 id应存在并在各自封闭的CppImplementationDataType或CppImplementationDataTypeElement的上下文中**具有唯一值**。 c *()*

显然，对于 TLV 数据 id 的定义，有必要避免歧义。可以分配这样一个 id 的每个模型元素应该只分配一个 id。

**[constr\_1599]** { DRAFT } **TlvDataIdDefinition引用ArgumentDataPrototype** d每个ArgumentDataPrototype应在同一TransformationPropsToServiceInterfaceElementMapping的上下文中的角色tlvArgument中**最多引用一次**。 c *()*

**[constr\_1600]** { DRAFT } **TlvDataIdDefinition引用ApplicationRecordElement** d每个ApplicationRecordElement在同一TransformationPropsToServiceInterfaceElementMapping的上下文中的角色tlvRecordElement中最多被引用**一次**。 c *()*

**[constr\_1601]** { DRAFT } **TlvDataIdDefinition引用CppImplementationDataTypeElement** d每个CppImplementationDataTypeElement在同一TransformationPropsToServiceInterfaceElementMapping的上下文中的角色tlvImplementationDataTypeElement中最多被引用**一次**。 c *()*

**[constr\_1748]** { DRAFT }**引用TlvDataIdDefinition的存在。 tlvArgument ， TlvDataIdDefinition 。 tlvRecordElement和TlvDataIdDefinition 。 tlvImplementationDataTypeElement** d对于每个TlvDataIdDefinition ，应仅存在以下引用之一：

* 角色tlvArgument中的ArgumentDataPrototype的引用
* 引用角色tlvRecordElement中的ApplicationRecordElement
* 角色tlvImplementationDataTypeElement中的AbstractImplementationDataTypeElement的引用。

### c ()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***AbstractImplementationDataTypeElement*** （抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes | | | |
| ***笔记*** | 此元类表示充当特定派生元类的抽象基类的能力，这些派生元类支持特定语言绑定的 ImplementationDataTypes 建模。 | | | |
| ***根据*** | *ARObject* , *AtpClassifier* , *AtpFeature* , *AtpStructureElement* , *Identifiable* , *MultilanguageReferable* ,  *可推荐* | | | |
| ***子类*** | CppImplementationDataTypeElement ， ImplementationDataTypeElement | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.147：AbstractImplementationDataTypeElement**

**[constr\_1628]** { DRAFT }**在使用 TLV 的情况下定义静态长度字段大小**d如果给定的Transformation存在聚合tlvDataIdDefinition-

PropsToServiceInterfaceElementMapping然后是属性

* sizeOfArrayLengthField ,
* sizeOfStringLengthField ,
* sizeOfStructLengthField和
* sizeOfUnionLengthField

应具有大于 0 的值。 c *()*

constr\_1628 ]存在的理由：TLV 序列化需要使用长度字段：

* 如果使用 4类线（更多详细信息，请参阅 [ TPS\_MANI\_01186 ]），则长度字段大小应静态配置。
* 如果使用 5-7线类型（参见 [ TPS\_MANI\_01186 ]），则长度字段大小的静态配置也应存在，因为并非所有长度字段都以标签开头，例如包含在数组或顶层中的结构SOME/IP 事件中包含的结构。

在这种情况下，如果不要求存在长度字段，则序列化的结果可能是不明确的，即使反序列化器无法确定数据布局[[9]](#footnote-9)。

**[constr\_1629]** { DRAFT }**在使用 TLV 的情况下，长度字段的大小相同**d如果给定TransformationPropsToServiceInterfaceElementMapping的聚合tlvDataIdDefinition存在，则属性

* sizeOfArrayLengthField ,
* sizeOfStringLengthField ,
* sizeOfStructLengthField和
* sizeOfUnionLengthField

应具有相同的值。 c *()*

constr\_1629 ]存在的理由：如果使用了线类型4（有关详细信息，请参阅 [ TPS\_MANI\_01186 ]）并且如果接收器遇到结构成员或带有未知标签的参数，则反序列化程序无法确定结构或参数成员的实际数据类型。

**[constr\_1630]** { DRAFT }**在使用 TLV 的情况下，没有在DataPrototype级别上定义长度字段大小**d如果给定TransformationPropsToServiceInterfaceElementMapping角色中的引用存在tlvDataIdDefinition

然后属性

* sizeOfArrayLengthField ,
* sizeOfStringLengthField ,
* sizeOfStructLengthField和
* sizeOfUnionLengthField

不应在 DataPrototype 级别上单独定义（即通过引用SomeipDataPrototypeTransformationProps . someipTransformationProps ），而只能在ServiceInterface级别上定义（即通过引用TransformationPropsToServiceInterfaceElementMapping . transformationProps ）。 c *()*

constr\_1630 ]存在的理由：如果使用线类型4（更多详细信息，请参阅 [ TPS\_MANI\_01186 ]）并且如果接收器遇到带有未知标签的成员或参数，则反序列化器需要知道长度字段。

实现这一点最可靠的方法是要求在ServiceInterface级别定义长度字段的大小。

**3.18.3.2 线型选择的分配**

TLV 编码支持所谓的线类型的定义，该类型控制如何解释有关长度字段长度的信息。

电线类型的具体设置含义在[ 7 , PRS SOME/IP 协议]中定义。

**[TPS\_MANI\_01186]** { DRAFT }**适用电线类型**d属性的定义

ApSomeipTransformationProps 。 isDynamicLengthFieldSize应用于定义适用的电线类型。

如果属性ApSomeipTransformationProps的值。 isDynamicLengthFieldSize设置为 True ，则应使用 5-7线类型。

如果属性ApSomeipTransformationProps的值。 isDynamicLength-

FieldSize不存在或设置为 False ，则应使用导线类型4。 C

*( RS\_MANI\_00030 )*

## 3.19 工艺设计

在例如诊断映射的定义中，对过程的分配通常在方法步骤中完成[[10]](#footnote-10) 当所有诊断映射[[11]](#footnote-11) 已经完成。

因此，最好为实际的Process实现代理，该代理可以在设计时作为与Process的关系的目标。这种语义是由元类ProcessDesign实现的。

**[TPS\_MANI\_01228]** { DRAFT }元类**ProcessDesign的语义**d元类ProcessDesign应在需要设计时表示时用于在工作流的**后续**步骤中作为部署规范的一部分设计的流程。 c *( RS\_MANI\_00004 )*

集成商必须注意实际流程引用相应的流程设计，以便通过此引用， AUTOSAR软件工具能够找出诊断映射和流程之间的关系，前提是每个流程设计**仅**由单个进程。

**[constr\_1550]** { DRAFT }**从Process引用ProcessDesign** d每个ProcessDesign只能从单个Process引用。 c *()*

请注意，从Process到ProcessDesign的引用承认了Process通常是稍后创建的事实[[12]](#footnote-12)。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **流程设计** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ProcessDesign | | | |
| ***笔记*** | 这个元类有能力在进程还不存在的时候代替进程。但它的未来存在已经需要在设计阶段考虑，为此需要一个专用的模型元素。  **标签：**  atp.Status=草稿  atp.recommendedPackage=工艺设计 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 确定性的  客户资源  需求 | 确定性客户端  资源需求 | \* | 聚合 | 此聚合表示确定性客户端设计的适用资源需求的集合。  **标签：** atp.Status=draft |
| 可执行的 | 可执行文件 | 0..1 | 参考 | 对在进程中执行的可执行文件的引用。  **标签：** atp.Status=draft |

**表 3.148：流程设计**

可以想象，诊断映射与元类ProcessDesign的关联可能仍会作为创建诊断映射的活动的最后一步进行。为了适应这种潜在的建模，从诊断映射到ProcessDesign的引用已由原型 atpSplitable 修饰。

有关此原型的语义的更多信息，请参阅 AUTOSAR 通用结构模板 [ 6 ] 的规范。

**[constr\_1693]** { DRAFT } **Executable 、 ProcessDesign和Process的关系**d ProcessDesign引用的任何Executable也应被引用ProcessDesign的每个Process引用。 c *()*

*AtpStructureElement*

*FibexElement*

MachineDesign

+

accessControl: AccessControlEnum

[0..1]

+

pncPrepareSleepTimer: TimeValue

[0..1]

+

pnResetTimer: TimeValue

[0..1]

*ARElement*

ProcessDesign

*ARElement*

ProcessDesignToMachineDesignMapping

machineDesign

+

0..1

+

processDesign

1

.

**图 3.80： ProcessDesignToMachineDesignMapping的建模**

**[TPS\_MANI\_01229]** { DRAFT }**在特定**MachineDesign上**预分配给定ProcessDesign** d也可以在特定**MachineDesign上预分配给定**ProcessDesign 。为此，存在元类ProcessDesignToMachineDesignMapping 。 c *( RS\_MANI\_00004 )*

元类MachineDesign的语义在第5节中解释。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **过程设计到机器设计映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign | | | |
| ***笔记*** | 该元素在设计阶段用于预定义流程到机器的映射。这种映射可能在部署阶段被否决。  **标签：**  atp.Status=草稿  atp.recommendedPackage=ProcessDesignToMachineDesignMappings | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 机器设计 | 机器设计 | 0..1 | 参考 | 此参考在 ProcessDesignToMachineDesignMapping 的上下文中标识 MachineDesign。  **标签：** atp.Status=draft |
| 流程设计 | 流程设计 | 1 | 参考 | 此参考在 ProcessDesignToMachineDesignMapping 的上下文中标识了 ProcessDesign。  **标签：** atp.Status=draft |

**表 3.149：ProcessDesignToMachineDesignMapping**

请注意，预期的ProcessDesignToMachineDesignMapping可能无法用于目标机器，因此可能会在部署阶段创建不同的ProcessToMachineMapping 。

**3.19.1 确定性客户端资源**

元类ProcessDesign也可用于添加对所谓的确定性客户端的支持。

请注意，术语确定性客户端的具体含义的解释超出了本文档的范围。详细解释可以在 SWS 执行管理 [ 18 ] 中找到。

为了正式支持 Deterministic Client，在ProcessDesign中聚合了元类DeterministicClientResourceNeeds 。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **确定性客户端资源需求** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ProcessDesign | | | |
| ***笔记*** | 此元类指定 DeterministicClient 库函数的进程和周期特定计算资源需求。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 硬件平台 | 细绳 | 0..1 | 属性 | 此属性表示目标平台的文本标识。  **标签：** atp.Status=draft |
| 初始化资源 | 确定性客户端  资源 | 0..1 | 聚合 | 这代表了一个计算资源的需求  DeterministicClient::WaitForNextActivation kInit 周期。  **标签：** atp.Status=draft |
| 运行资源 | 确定性客户端  资源 | 0..1 | 聚合 | 这代表了一个计算资源的需求  DeterministicClient::WaitForNextActivation kRun 循环。  **标签：** atp.Status=draft |

**表 3.150：DeterministicClientResourceNeeds**

*ARElement*

ProcessDesign

*Identifiable*

DeterministicClientResourceNeeds

+

hardwarePlatform: String

[0..1]

DeterministicClientResource

[0..1]

numberOfInstructions: NormalizedInstruction

+

+

sequentialInstructionsBegin: NormalizedInstruction

[0..1]

+

sequentialInstructionsEnd: NormalizedInstruction

[0..1]

[0..1]

+

speedup: Float

+

deterministicClientResourceNeeds

0..\*

runResource

+

0..1

+

initResource

0..1

**图 3.81： DeterministicClientResourceNeeds的建模**

**[TPS\_MANI\_01199]** { DRAFT } **DeterministicClientResourceNeeds的语义**d元类DeterministicClientResourceNeeds以两个角色聚合DeterministicClientResource ，以便能够在 Deterministic Client 执行的两个不同上下文中指定资源需求。 c *( RS\_MANI\_00050 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **确定性客户端资源** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ProcessDesign | | | |
| ***笔记*** | 此元类指定 DeterministicClient 库函数的计算资源需求。  **标签：** atp.Status=draft | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 数量  指示 | 规范化指令 | 0..1 | 属性 | 该属性表示目标系统上的标准化运行时消耗在一个  DeterministicClient::WaitForNextActivation 循环，假设工人将按顺序执行的“最坏情况”运行时。  **标签：** atp.Status=draft |
| 顺序指令  开始 | 规范化指令 | 0..1 | 属性 | 开始时的规范化顺序运行时  DeterministicClient::WaitForNextActivation 循环（大多数情况下无法并行化），在工作池的主要使用开始之前。**标签：** atp.Status=draft |
| 顺序的  说明结束 | 规范化指令 | 0..1 | 属性 | WaitForNextActivation 循环（大多数情况下无法并行化），在工作池的主要使用结束后。  **标签：** atp.Status=draft |
| 加速 | 漂浮 | 0..1 | 属性 | 该属性定义了在 numberOfWorkers 物理可用的情况下，一个 DeterministicClient::WaitForNextActivation 周期内的计算可以以多快的速度完成，即机器上是否有足够的内核来执行所有工作程序的并行执行（顺序运行时/并行运行时）。  **标签：** atp.Status=draft |

**表 3.151：DeterministicClientResource**

**[TPS\_MANI\_01200]** { DRAFT }元类**DeterministicClientResource的语义**d元类DeterministicClientResource定义了几个属性，这些属性提供了有关工作线程执行性质的信息。这些属性的值被赋予一个无量纲的NormalizedInstruction 。

然而，属性的值

* 指令数
* 顺序指令开始
* 顺序指令结束

仅对特定硬件平台有效。使用NormalizedInstruction的目的是在集成时调整不同Process es（可能来自不同供应商）的资源使用情况。 c *( RS\_MANI\_00050 )*

|  |  |
| --- | --- |
| ***原始*** | **规范化指令** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ProcessDesign |

5

4

|  |  |
| --- | --- |
| ***原始*** | **规范化指令** |
| ***笔记*** | 此元类用于描述 Deterministic Client::WaitForNextActivation 周期内目标系统上的运行时预算需求。 NormalizedInstructions 不反映代码指令的实际数量，但允许描述比较资源需求。 NormalizedInstructions 用于在集成时配置计算资源。 NormalizedInstruction = 运行时间（秒） \* 时钟频率（Hz）  **标签：**  atp.Status=草稿  xml.xsd.customType=NORMALIZED-INSTRUCTION xml.xsd.pattern=[1-9][0-9]\* xml.xsd.type=string |

**表 3.152：规范化指令**

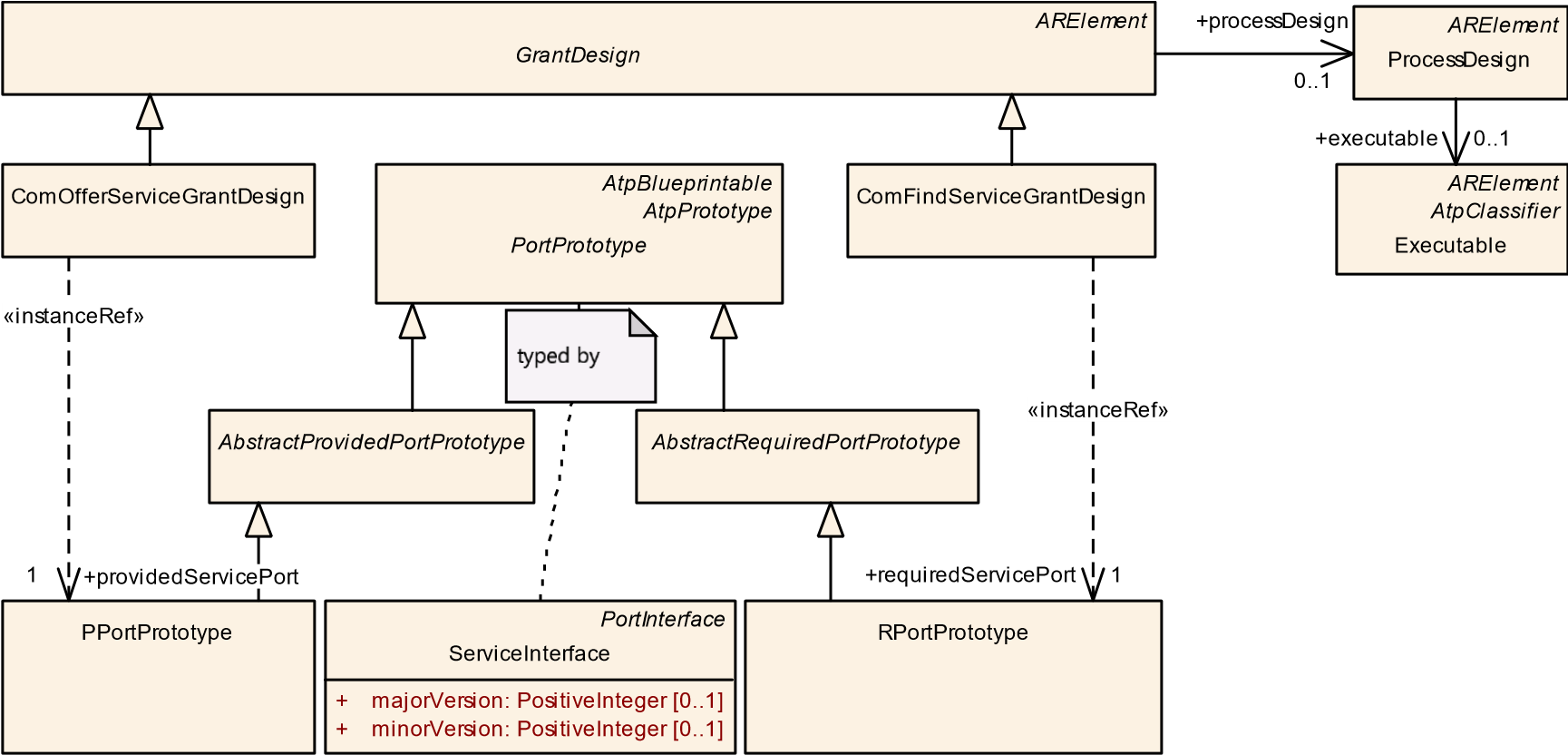
## 3.20 赠款设计

第3.15.5.1.3章中描述的意图定义（例如： ClientComSpec.clientIntent ）用于表达软件设计者在应用软件中使用（或不使用）特定 API 的意图。

意图的定义代表身份和访问管理 (IAM) 的一个方面。 IAM 配置的另一个方面是平台软件授予的实际权限的定义。

此类赠款的建模在两个层面上完成：

* GrantDesign的定义允许预先指定设计级别的授权。本章描述了GrantDesign的建模。
* Grant的定义允许从平台软件的角度对 Grants 进行实际和最终的规范。 Grant 的建模在第9.9章中描述。



**图 3.82：服务发现的授权设计建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***格兰特设计***（摘要） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign | | | |
| ***笔记*** | 该元类用作描述设计级别授权的抽象基类。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***子类*** | ComFindServiceGrantDesign , *ComGrantDesign* , ComOfferServiceGrantDesign , RawDataStreamGrant  设计 | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 流程设计 | 流程设计 | 0..1 | 参考 | 该参考确定了为 GrantDesing 提供上下文的相应流程设计。  **标签：** atp.Status=draft |

**表 3.153：GrantDesign**

抽象元类GrantDesign充当设计级别授权定义的基类。

授予特定于给定的Process 。换句话说，从同一个Executable创建的两个Process可以被分配不同的授权集。该特定关系也应在设计级别上可用。

**[TPS\_MANI\_01231]** { DRAFT } **GrantDesign引用ProcessDesign** d元类GrantDesign引用ProcessDesign作为为给定Process设计一组Grant的方法。 c *( RS\_MANI\_00060 )*

**3.20.1 Com Grant 设计**

GrantDesign的子类以涵盖在*AUTOSAR 自适应平台*上进行通信的授权的特定方面。

**[TPS\_MANI\_01232]** { DRAFT }**元类ComOfferServiceGrantDesign的语义**d ComOfferServiceGrantDesign在角色提供服务端口中引用特定AbstractProvidedPortPrototype的存在表明设计预见到所引用的AbstractProvidedPortPrototype应被授予提供相应服务的权利。 c *( RS\_MANI\_00060 )*

ComOfferServiceGrantDesign的存在相对应的显式建模意图。理解是，仅存在由ServiceInterface键入的AbstractProvidedPortPrototype就表明了提供服务的意图。

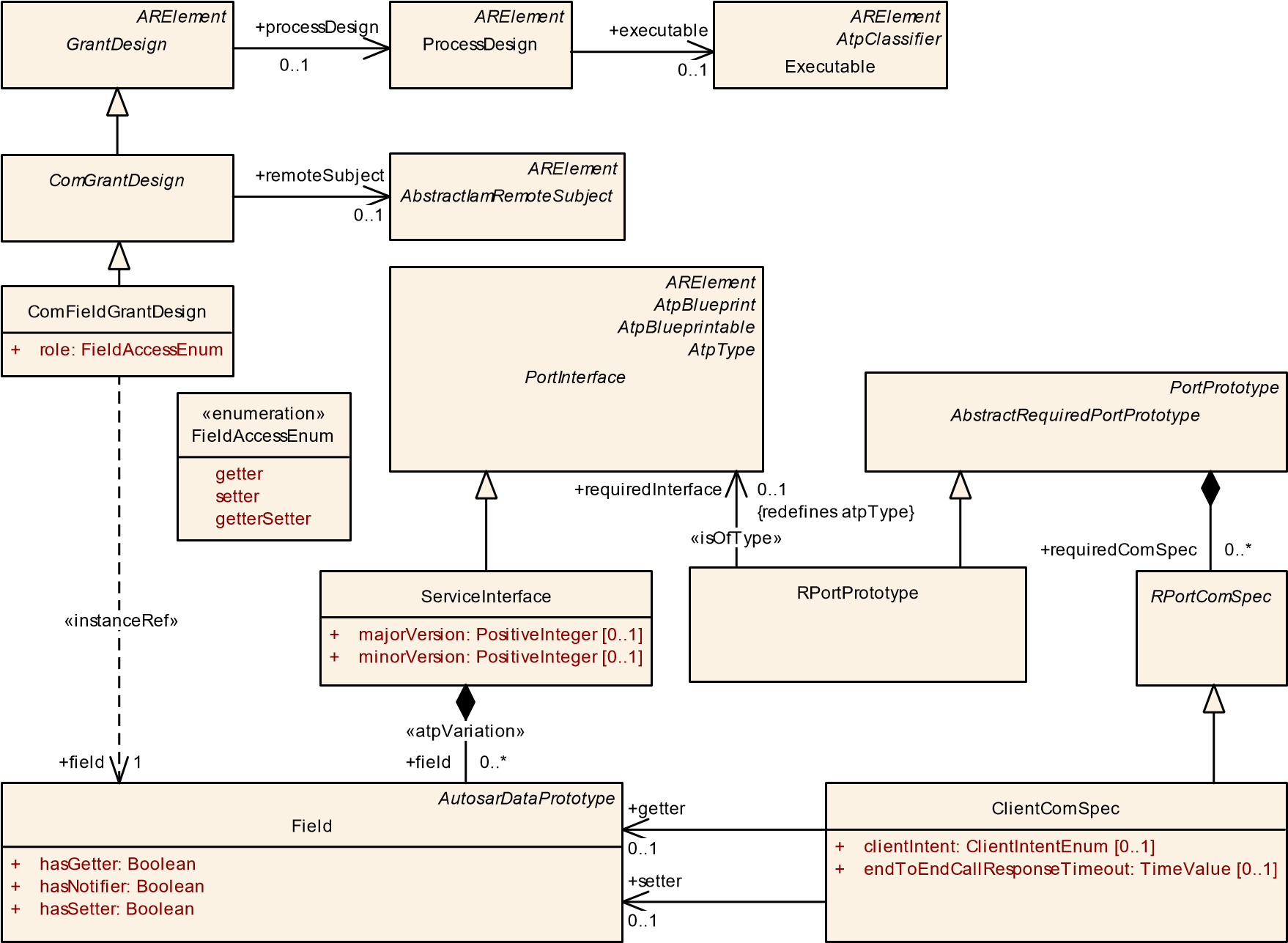
**[TPS\_MANI\_01233]** { DRAFT }**元类ComFindServiceGrantDesign的语义**d引用角色requiredServicePort中特定AbstractRequiredPortPrototype的ComFindServiceGrantDesign的存在表明设计预见到所引用的AbstractRequiredPortPrototype应被授予查找相应服务的权利。 c *( RS\_MANI\_00060 )*

ComFindServiceGrantDesign的存在相对应的显式建模意图。

理解是，仅存在由ServiceInterface键入的AbstractRequiredPortPrototype就表明了查找服务的意图。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **ComOfferServiceGrantDesign** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant | | | |
| ***笔记*** | 这个元类代表了定义一个提供服务的授权的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=GrantDesigns | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *GrantDesign* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 提供的服务端口 | PPort原型 | 1 | 参考 | 此 instanceRef 标识应提供服务的 PPortPrototype。  **标签：** atp.Status=draft  **InstanceRef 由：** PPortPrototypeIn实现  可执行实例引用 |

**表 3.154：ComOfferServiceGrantDesign**



**田间拨款设计建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **ComFindServiceGrantDesign** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant | | | |
| ***笔记*** | 此元类表示定义用于查找服务的 Grant 的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=GrantDesigns | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *GrantDesign* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 需要服务端口 | RPort原型 | 1 | 参考 | 这个 instanceRef 标识了应该在其上找到服务的 RPortPrototype。  **标签：** atp.Status=draft  **InstanceRef 由：** RPortPrototypeIn实现  可执行实例引用 |

**表 3.155：ComFindServiceGrantDesign**

**[TPS\_MANI\_01234]** { DRAFT } **ComFieldGrantDesign的语义**d ComFieldGrantDesign在角色字段中引用特定字段的存在表明该设计预见到应用软件应被授予访问相应字段的权限。访问的性质，即get vs. set，是通过属性角色指定的。 c *( RS\_MANI\_00060 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **ComFieldGrantDesign** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant | | | |
| ***笔记*** | 此元类表示为 ServiceInterface.field 定义 Grant 的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=GrantDesigns | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *ComGrantDesign* , *GrantDesign* , *Identifiable* , *Multilanguage Referrable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 场地 | 场地 | 1 | 参考 | 在可执行文件的上下文中引用受影响的字段。  **标签：** atp.Status=draft  **InstanceRef 实现者：** FieldInExecutableInstance Ref |
| 角色 | 字段访问枚举 | 1 | 属性 | 此属性提供了从设计角度进一步指定对 ServiceInterface.field 的访问的能力。  **标签：** atp.Status=draft |

**表 3.156：ComFieldGrantDesign**

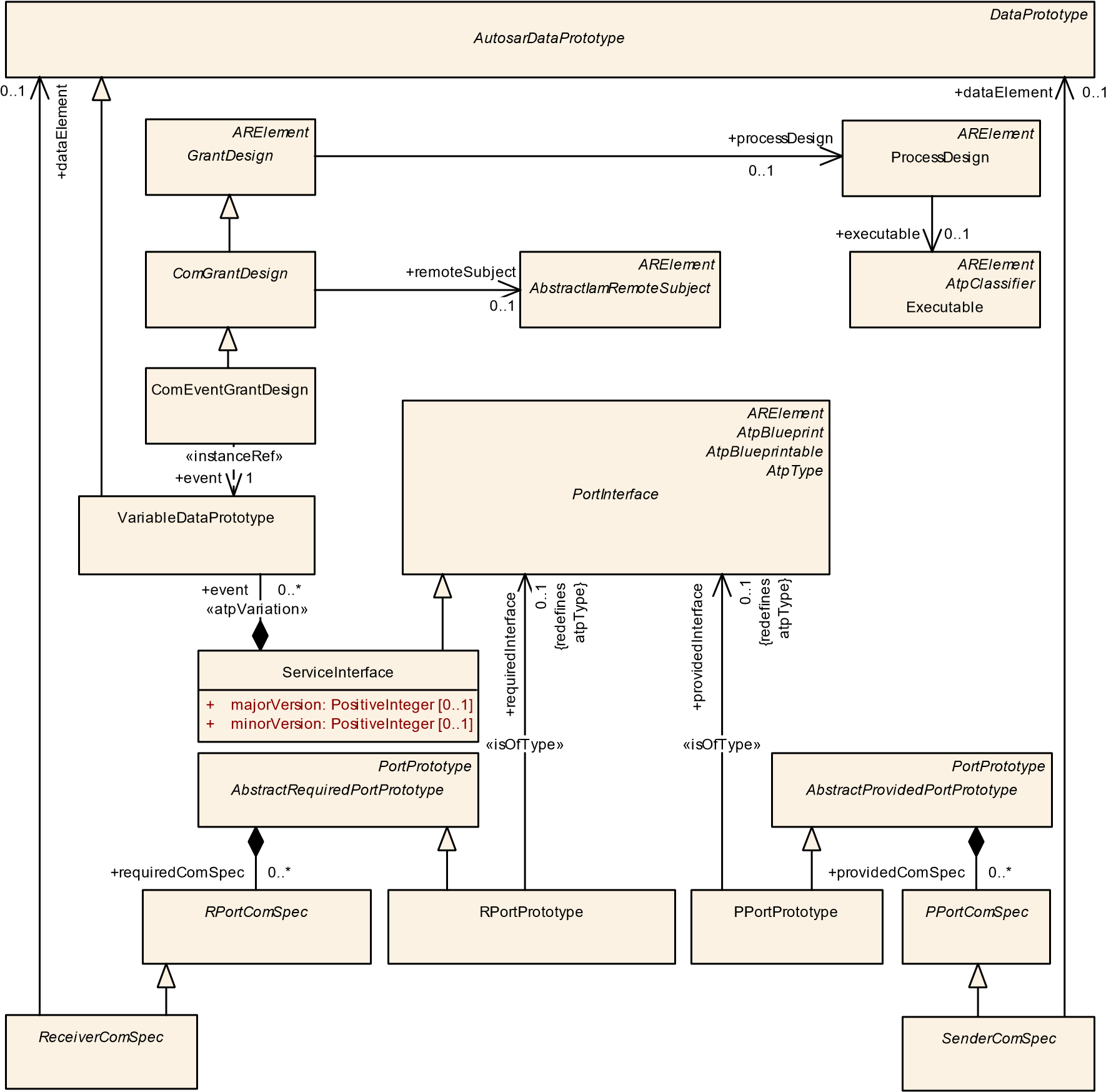
|  |  |
| --- | --- |
| ***枚举*** | **字段访问枚举** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant |
| ***笔记*** | 这个元类提供了限定访问字段的值。  **标签：** atp.Status=draft |
| ***文字*** | ***描述*** |

5

4

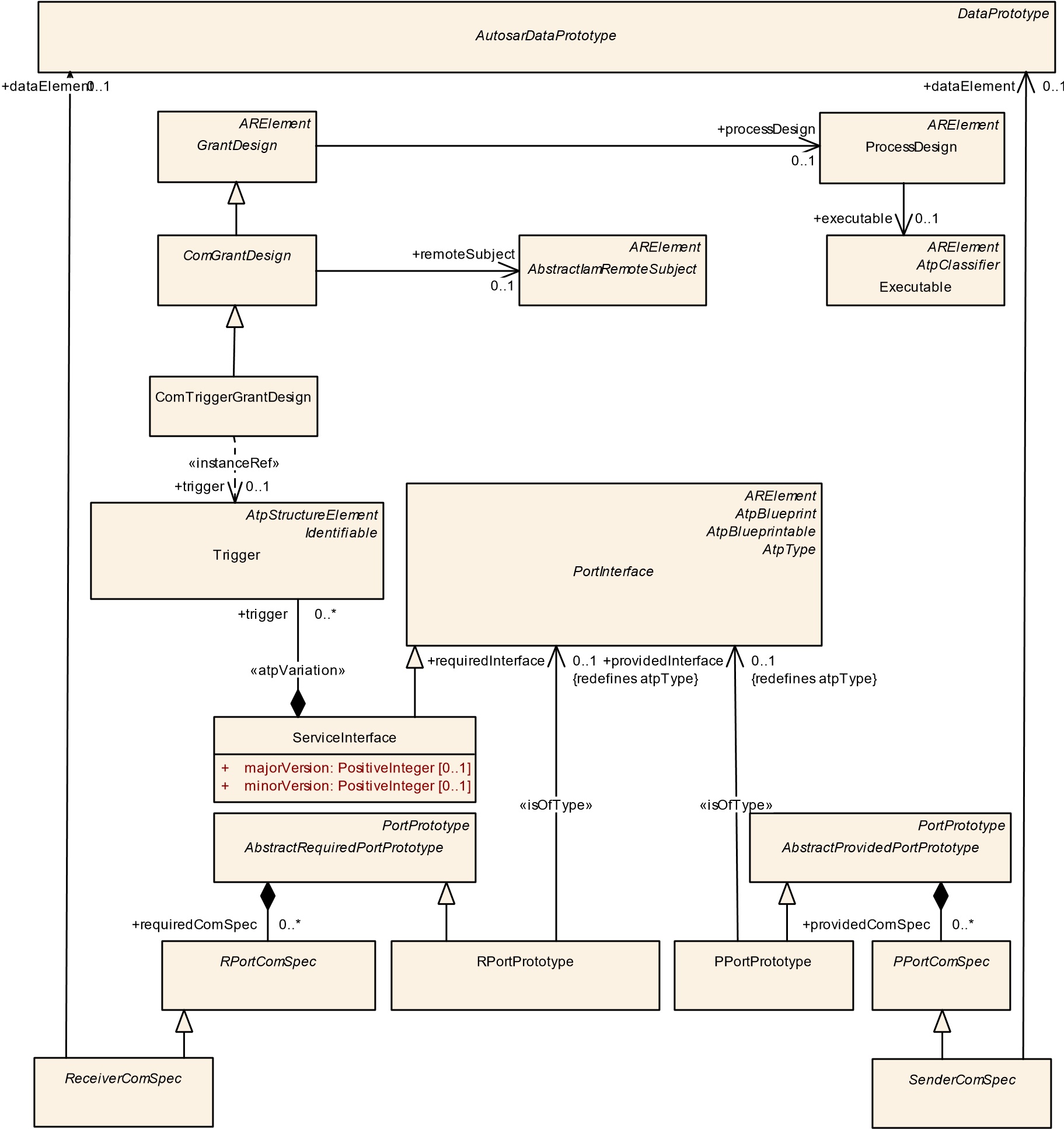
|  |  |
| --- | --- |
| ***枚举*** | **字段访问枚举** |
| 吸气剂 | 访问 Field 的 getter。  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |
| getterSetter | 访问字段的 getter 和 setter  **标签：**  atp.EnumerationLiteralIndex=2 atp.Status=草稿 |
| 二传手 | 访问字段的设置器。  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |

**表 3.157：FieldAccessEnum**



**活动资助设计建模**

**[TPS\_MANI\_01235]** { DRAFT } ComEventGrantDesign**的语义**d **ComEventGrantDesign**的存在引用了由封闭的ServiceInterface在角色事件中聚合的特定VariableDataPrototype ，这表明该设计预见到应用程序软件应被授予访问相应事件的权限。 c *( RS\_MANI\_00060 )*



**触发器的授权设计建模**

|  |  |
| --- | --- |
| ***班级*** | **ComEventGrantDesign** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **ComEventGrantDesign** | | | |
| ***笔记*** | 此元类表示为 ServiceInterface.event 定义 Grant 的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=GrantDesigns | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *ComGrantDesign* , *GrantDesign* , *Identifiable* , *Multilanguage Referrable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 事件 | 变量数据原型 | 1 | 参考 | 此引用表示受影响的事件。  **标签：** atp.Status=draft  **InstanceRef 实现者：** EventInExecutable InstanceRef |

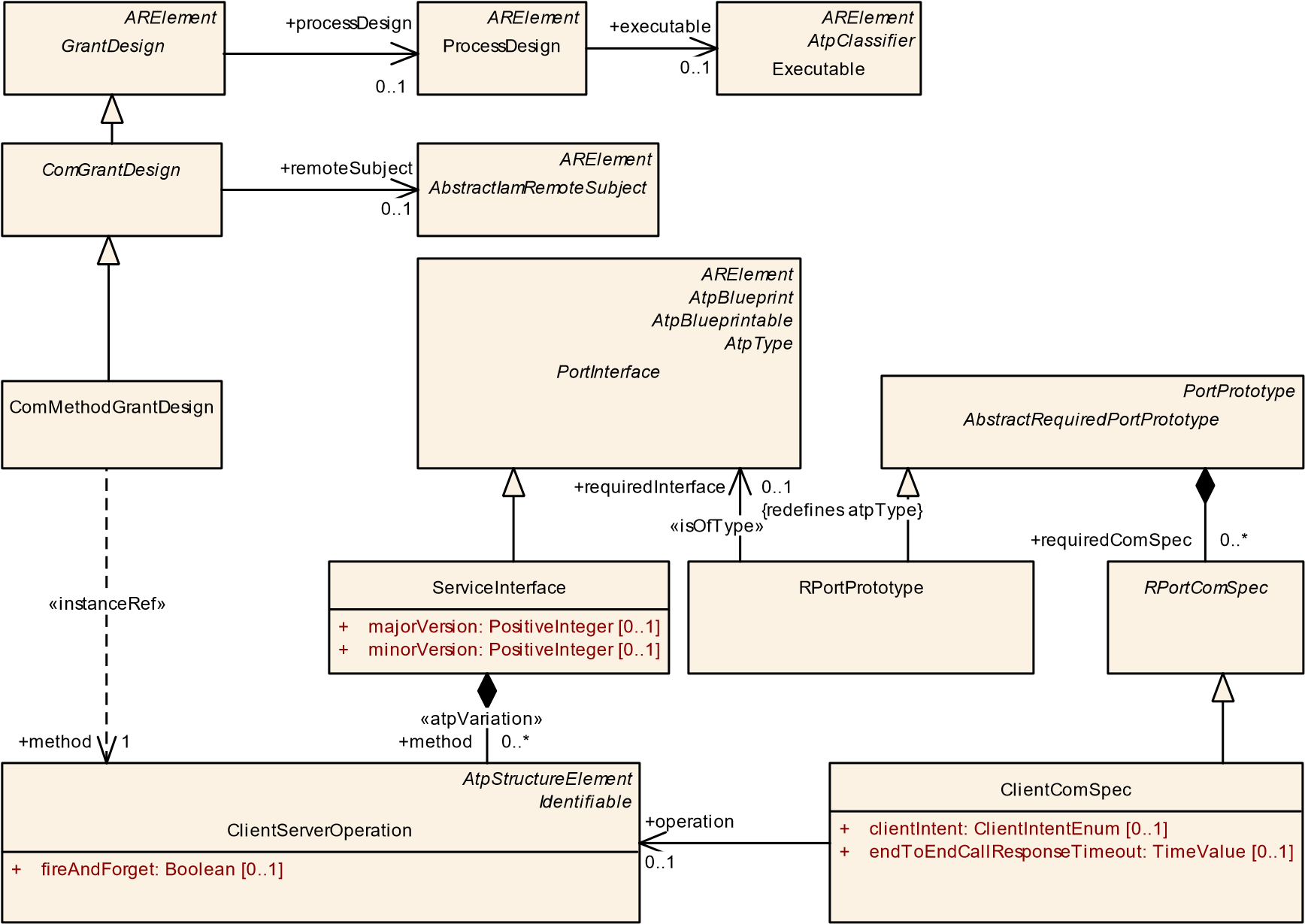
**表 3.158：ComEventGrantDesign**

**[TPS\_MANI\_03290]** { DRAFT } ComTriggerGrantDesign的**语义**d **ComTriggerGrantDesign**的存在引用了由封闭的ServiceInterface在角色触发器中聚合的特定触发器，这表明该设计预见到应用程序软件应被授予访问相应触发器的权限。 c *( RS\_MANI\_00060 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **ComTriggerGrantDesign** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant | | | |
| ***笔记*** | 此元类表示为 ServiceInterface.trigger 定义 Grant 的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=GrantDesigns | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *ComGrantDesign* , *GrantDesign* , *Identifiable* , *Multilanguage Referrable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 扳机 | 扳机 | 1 | 参考 | 此引用表示受影响的触发器。  **标签：** atp.Status=draft  **InstanceRef 实现者：** TriggerInExecutable InstanceRef |

**表 3.159：ComTriggerGrantDesign**

**[TPS\_MANI\_01236]** { DRAFT } ComMethodGrantDesign的**语义**d **ComMethodGrantDesign**的存在引用了通过封闭的ServiceInterface在角色方法中聚合的特定ClientServerOperation ，这表明该设计预见到应用程序软件应被授予调用相应方法的权利。 c *( RS\_MANI\_00060 )*



**方法的资助设计建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **ComMethodGrant设计** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant | | | |
| ***笔记*** | 此元类表示为 ServiceInterface.method 定义 Grant 的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=GrantDesigns | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *ComGrantDesign* , *GrantDesign* , *Identifiable* , *Multilanguage Referrable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 方法 | 客户端服务器操作 | 1 | 参考 | 此参考标识了相应的方法。  **标签：** atp.Status=draft  **InstanceRef 由：** RequiredMethodIn ExecutableInstanceRef实现 |

**表 3.160：ComMethodGrantDesign**

**3.20.2 原始流数据的授权设计**

原始数据流的使用受 IAM 施加的限制。因此，存在元类RawDataStreamGrantDesign以支持此用例。

**[TPS\_MANI\_01284]** { DRAFT }**元类RawDataStreamGrantDesign的粒度**d RawDataStreamGrantDesign的粒度是整个AbstractRawDataStreamInterface 。预计 IAM 策略的定义在较小级别上没有意义，即在由AbstractRawDataStreamInterface聚合的ClientServerOperation级别上。 c *( RS\_MANI\_00060 )*

*ARElement*

ProcessDesign

*ARElement*

*GrantDesign*

RawDataStreamGrantDesign

*AbstractRawDataStreamInterface*

RawDataStreamClientInterface

+

processDesign

0..1

+

rawDataStream

0..1

**图 3.87： RawDataStreamGrantDesign的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **RawDataStreamGrantDesign** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::RawDataStreamGrant | | | |
| ***笔记*** | 此元类表示在设计级别为 RawDataStream 定义 IAM 配置的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=GrantDesigns | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *GrantDesign* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 原始数据流 | 原始数据流客户端  界面 | 0..1 | 参考 | 该参考标识了适用的 RawDataStream 接口。  **标签：** atp.Status=draft |

**表 3.161：RawDataStreamGrantDesign**

**3.20.3 远程访问控制**

*Identity and Access Manager*部署的定义和授权的定义依赖于本地实施的身份和访问权限。换句话说，例如可以定义允许本地Machine上的本地Process调用ServiceInterface的特定方法。但是不能限制允许调用该方法的远程Machine 。

机器无法对传入请求进行额外检查，这一事实使得健康的远程机器能够处理错误发出的请求，以及攻击者通过向服务发出任意请求来提升权限。被攻陷的远程机器。

机器不可能识别出其通信对等方已受到威胁，因为攻击者可以访问该机器的所有资源并且可以在隐身模式下运行。将受损远程机器的损害降至最低的一种有效方法是在接收方对传入请求进行额外检查，以确保远程机器不能超出它们在健康状态下可以请求的内容。

|  |  |  |
| --- | --- | --- |
| 网络信息 -> SubjectId |  | SubjectId x Object -> {允许，拒绝} |

Subject

Authentication

Authorization

Object

**图 3.88：基于来自网络绑定的主题 ID 的访问策略实施**

访问控制过程旨在对“主题”和“对象”之间的关系执行策略。在远程机器调用服务接口的示例中，远程机器是主体，服务接口的方法是对象。

访问控制过程包括两个主要操作，即认证和授权，它们大多是独立的。在认证过程中，主体的身份得到验证并解析出一个真实的标识符。身份验证是链条的重要组成部分，以确保不同的主体不能互相冒充。

下一步，在授权期间，根据为访问对象定义的规则和策略检查主体的身份，以验证主体的请求是否合法。这些策略应由系统或资源所有者定义。

远程主体的认证基于网络绑定。建立安全通道后，远程对等方已经通过了身份验证协议。因此，身份信息可以转发到 IAM 以应用为来自该通道的请求定义的相应定义策略，如图 3.88 所示。

远程主题被建模为AbstractIamRemoteSubject的特化。

以下部分将介绍不同的专业。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***AbstractIamRemoteSubject*** （抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM | | | |
| ***笔记*** | 这个抽象元类定义了远程节点的代理信息。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***子类*** | IPSecIamRemoteSubject , IpIamRemoteSubject , TlsIamRemoteSubject | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 3.162：AbstractIamRemoteSubject**

ComGrantDesign的建模，定义了平台软件授予的权限。作为一个选项， ComGrantDesign能够在角色remoteSubject中引用AbstractIamRemoteSubject 。 **[TPS\_MANI\_03238]** { DRAFT } **ComMethodGrantDesign的定义。 remoteSubject** d如果ComMethodGrantDesign引用角色remoteSubject中的一个或多个AbstractIamRemoteSubject则设计预见只有定义的remoteSubject应被授予访问相同ComMethodGrantDesign在角色方法中引用的ClientServerOperation的权限。 c *( RS\_MANI\_00060 )*

**[TPS\_MANI\_03239]** {草案} **ComEventGrantDesign的定义。 remoteSubject** d如果ComEventGrantDesign引用角色 remoteSubject 中的一个或多个AbstractIamRemoteSubject则设计预见只有定义的remoteSubject应被授予访问同一ComEventGrantDesign在角色事件中引用的VariableDataPrototype的权限。 c *( RS\_MANI\_00060 )*

**[TPS\_MANI\_03251]** {草案} **ComFieldGrantDesign的定义。 remoteSubject** d如果ComFieldGrantDesign引用角色remoteSubject中的一个或多个AbstractIamRemoteSubject则设计预见只有定义的remoteSubject应被授予访问同一ComFieldGrantDesign在角色字段中引用的字段的权限。 c *( RS\_MANI\_00060 )*

**3.20.3.1 TLS 情况下的远程主体**

本章定义了在基于 TLS 的安全通道的情况下如何对AbstractIamRemoteSubject建模。

**[TPS\_MANI\_03240]** { DRAFT }**在基于 TLS 的安全通道的情况下远程对等点的建模**d在基于 TLS 的安全通道的情况下，远程对等点被建模为

TlsIamRemoteSubject由以下方式标识

* 一个CryptoServiceCertificate被TlsIamRemoteSubject在角色acceptedRemoteCertificate中引用，
* TlsIamRemoteSubject通过引用的预共享密钥

TlsCryptoCipherSuite在角色中接受了CryptoCipherSuiteWithPsk 。

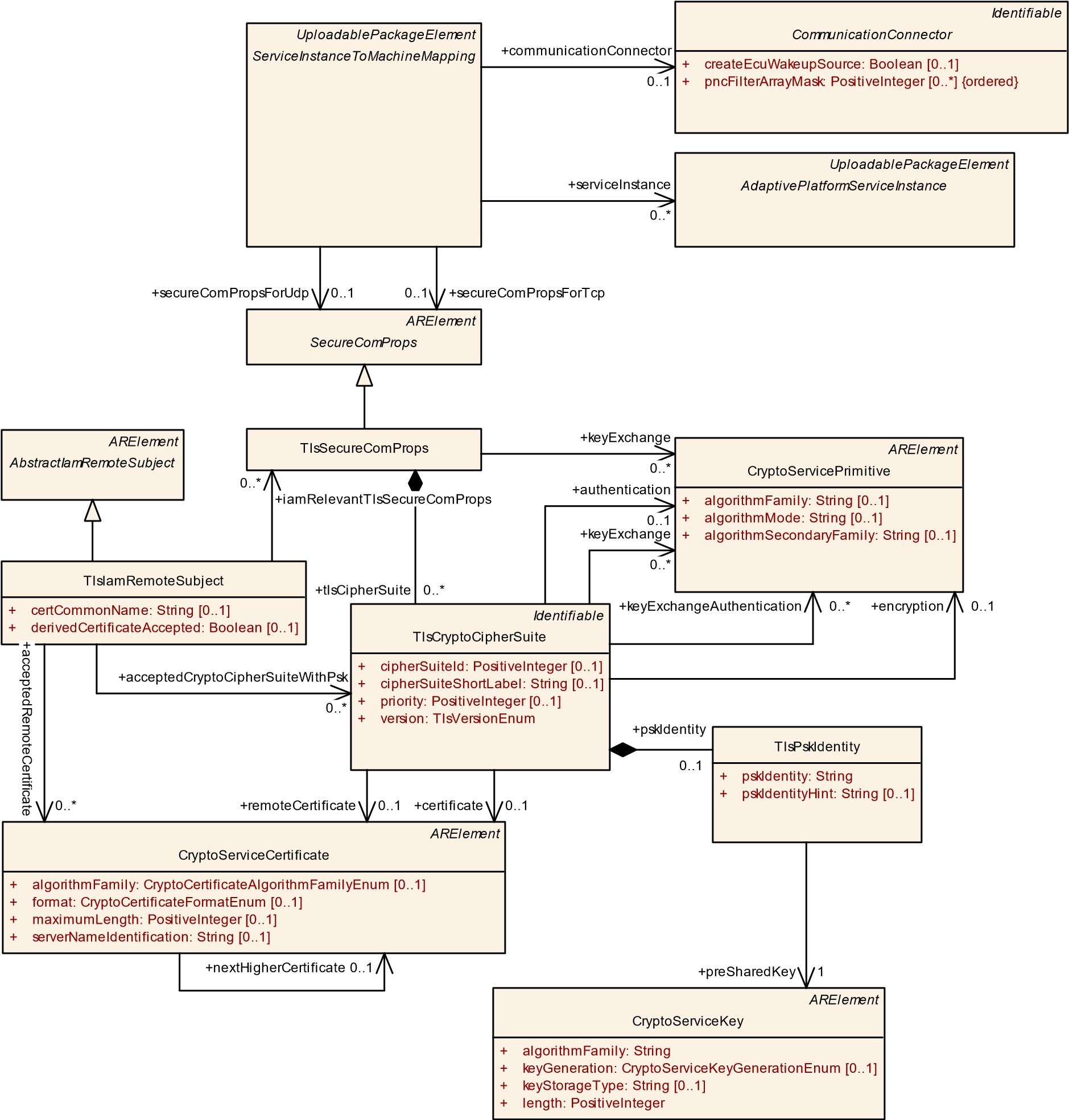
c *( RS\_MANI\_00036 , RS\_MANI\_00060 )*

请注意，在 TLS 协议中作为身份验证的预共享密钥的安全性取决于共享同一密钥的实体数量。如果多个Machine使用相同的共享密钥，则无法可靠地区分这些Machine ，因为它们中的任何一个都可以冒充其他的。

只能确保不知道预共享密钥的其他机器不能建立安全通道。

TlsIamRemoteSubject可以通过两种方式使用证书来识别。首先，可以通过引用它们并将 derivedCertificateAccepted 设置为 false来直接指定TlsIamRemoteSubject可以使用的证书。

这种方法要求本地机器上存在远程证书。其次，通过将 derivedCertificateAccepted 设置为 true，可以指定TlsIamRemoteSubject的通用名称（在 X509 证书中给出） 。



**图 3.89：TLS 情况下远程节点的代理信息**

在这种情况下， acceptedRemoteCertificate为TlsIamRemoteSubject提供的证书定义了一组允许的根证书。

较高的多重性的原因是 OEM 可能有多个Machine供应商，并且应允许定义在这种情况下所有这些Machine都允许连接，即使它们具有不同的证书链。 **[TPS\_MANI\_03241]** { DRAFT }**相关TlsSecureComProps的建模**

**TlsIamRemoteSubject** d与TlsIamRemoteSubject 。 iamRelevantTlsSecureComProps参考可以定义所有TlsSecureComProps

TlsIamRemoteSubject支持建立安全通道。 c *( RS\_MANI\_00036 ,*

*RS\_MANI\_00060 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **TlsIamRemoteSubject** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM | | | |
| ***笔记*** | 在 TLS 的情况下，此元类定义有关远程节点的代理信息。  **标签：**  atp.Status=草稿  atp.recommendedPackage=IamRemoteSubjects | | | |
| ***根据*** | *ARElement* , *ARObject* , *AbstractIamRemoteSubject* , *CollectableElement* , *Identifiable* , *Multilanguage Referrable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 接受Crypto CipherSuiteWith  普斯克 | TLSCryptoCipherSuite | \* | 参考 | 该参考用于通过预共享密钥识别远程节点。  **标签：** atp.Status=draft |
| 接受远程  证书 | 加密服务  证书 | \* | 参考 | 此引用用于通过证书识别远程节点。**标签：** atp.Status=draft |
| certCommon Name | 细绳 | 0..1 | 属性 | 此属性定义远程对等点证书的通用名称 (CN)。  **标签：** atp.Status=draft |
| 派生证书  公认 | 布尔值 | 0..1 | 属性 | 此属性定义是否接受 (true) 或不接受 (false) 派生证书。  **标签：** atp.Status=draft |
| iamRelevantTls  安全通信  道具 | TlsSecureComProps | \* | 参考 | 此参考定义了与 IAM 相关的本地 TlsSecureComProps。  **标签：** atp.Status=draft |

**表 3.163：TlsIamRemoteSubject**

**3.20.3.2 IPsec 情况下的远程主体**

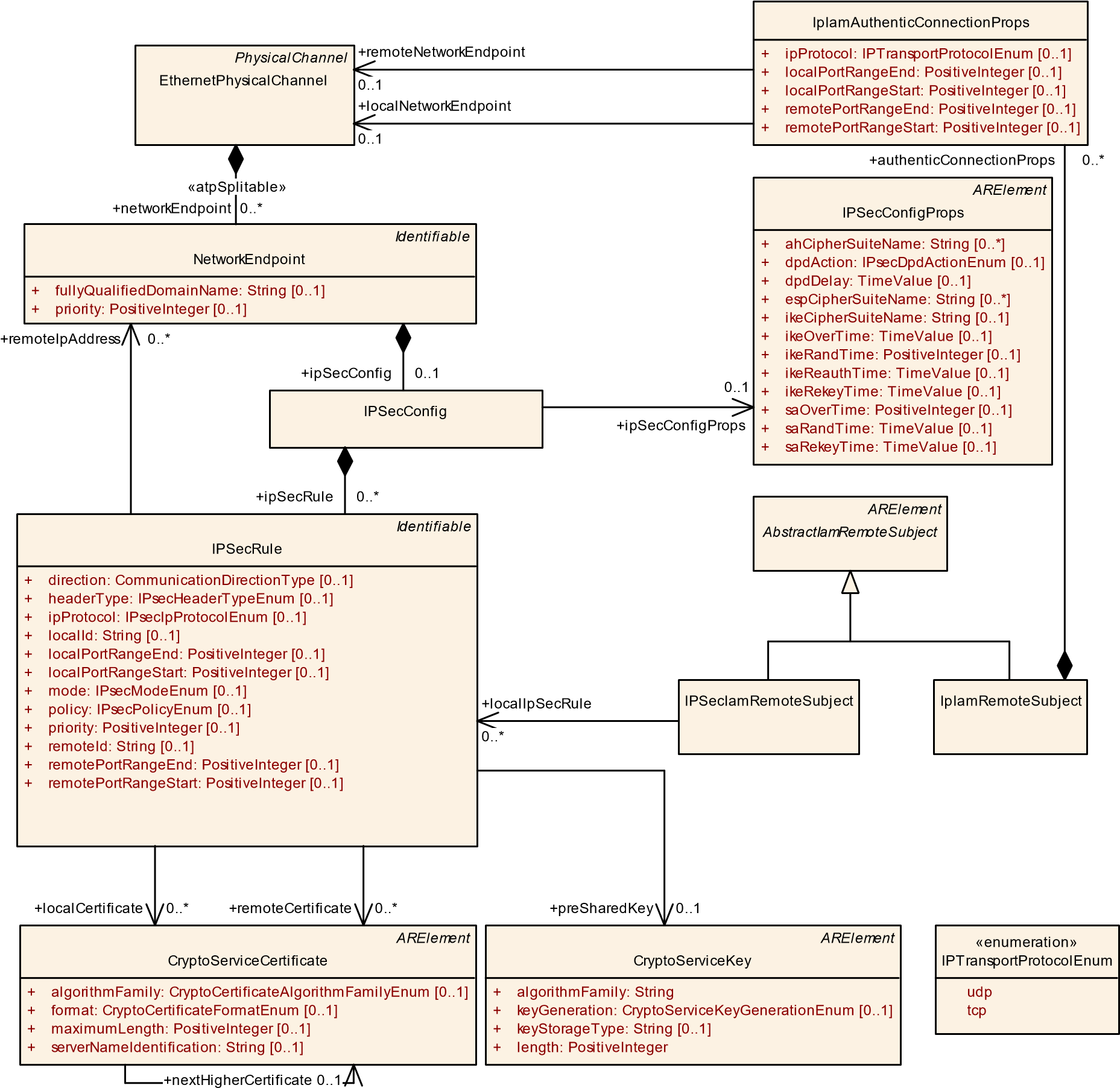
本章定义了在基于 IPsec 的安全通道的情况下如何对AbstractIamRemoteSubject建模。

**[TPS\_MANI\_03242]** { DRAFT }**在基于 IPsec 的安全通道的情况下远程对等方的建模**d在基于 IPsec 的安全通道的情况下，远程对等方被建模为IPSecIamRemoteSubject ，由localIpSecRule引用的IPSecRule 标识。 IPSecRule定义了允许远程对等方建立的所有安全连接。 c *( RS\_MANI\_00036 , RS\_MANI\_00060 )*

请注意，远程对等方的本地 IP 地址由聚合IPSecRule的NetworkEndpoint定义。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **IPSecIamRemoteSubject** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM | | | |
| ***笔记*** | 在 IPsec 的情况下，此元类定义有关远程节点的代理信息。  **标签：**  atp.Status=草稿  atp.recommendedPackage=IamRemoteSubjects | | | |
| ***根据*** | *ARElement* , *ARObject* , *AbstractIamRemoteSubject* , *CollectableElement* , *Identifiable* , *Multilanguage Referrable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| localIpSecRule | IPSecRule | \* | 参考 | 此参考用于描述 RemoteSubjects 本地 IPSec 规则。  **标签：** atp.Status=draft |

**表 3.164：IPSecIamRemoteSubject**



**图 3.90：IPsec 情况下远程节点的代理信息**

**3.20.3.3 IP通信情况下的远程主体**

请注意，可以定义基于通用 IP 通信的AbstractIamRemoteSubject 。在这种情况下，没有给出关于如何保护通信的细节，并且实际上保护通信（例如，加密、通过硬件机制或适当的网络和交换机设计）不是模型的一部分。 IpIamRemoteSubject由本地和远程 IP 地址、本地和远程端口范围以及传输协议的组合标识。

**[TPS\_MANI\_03244]** { DRAFT }**在通用 IP 通信的情况下远程对等点的建模**d在通用 IP 通信的情况下，远程对等点被建模为IpIamRemoteSubject ，由localNetworkEndpoint引用所引用的NetworkEndpoint 标识。允许定义的远程对等点通过ipProtocol和定义的本地端口范围和远程端口范围建立到remoteNetworkEndpoint的 IP 连接。 c *( RS\_MANI\_00060 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **IpIamRemoteSubject** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM | | | |
| ***笔记*** | 在一般 IP 通信的情况下，此元类定义有关远程节点的代理信息。  **标签：**  atp.Status=草稿  atp.recommendedPackage=IamRemoteSubjects | | | |
| ***根据*** | *ARElement* , *ARObject* , *AbstractIamRemoteSubject* , *CollectableElement* , *Identifiable* , *Multilanguage Referrable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 真正的  联系  道具 | IpIamAuthentic  连接道具 | \* | 聚合 | 分配给 IpIamRemote 主题的 IP 规则的定义。  **标签：** atp.Status=draft |

**表 3.165：IpIamRemoteSubject**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **IpIamAuthenticConnectionProps** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM | | | |
| ***笔记*** | 此元类在 IAM 配置的上下文中为 IP 连接定义一组属性。  **标签：** atp.Status=draft | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| ip协议 | IP传输协议  枚举 | 0..1 | 属性 | 该属性定义了相关的 IP 协议。  **标签：** atp.Status=draft |
| 本地网络端点 | 以太网物理  渠道 | 0..1 | 参考 | 此参考根据 IAM 配置定义了一个真实的本地网络端点。  **标签：** atp.Status=draft |
| localPortRange 结束 | 正整数 | 0..1 | 属性 | 此属性限制流量监控并定义本地端口范围的结束值。  **标签：** atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **IpIamAuthenticConnectionProps** | |  |  |
| localPortRange 开始 | 正整数 | 0..1 | 属性 | 此属性限制流量监控并定义本地端口范围的起始值。  **标签：** atp.Status=draft |
| 远程网络端点 | 以太网物理  渠道 | 0..1 | 参考 | 此参考根据 IAM 配置定义了一个真实的远程网络端点。  **标签：** atp.Status=draft |
| 远程端口范围结束 | 正整数 | 0..1 | 属性 | 此属性限制流量监控并定义远程端口范围的结束值。  **标签：** atp.Status=draft |
| 远程端口范围开始 | 正整数 | 0..1 | 属性 | 此属性限制流量监控并定义远程端口范围的起始值。  **标签：** atp.Status=draft |

**表 3.166：IpIamAuthenticConnectionProps**

**3.20.3.4 SecOC 通信中的远程主体**

在 SecOC 的情况下，身份信息取决于共享相同加密密钥的Machine组。

换句话说，如果使用给定密钥接收到有效的 SecOC 消息，则认为只有“知道密钥”的远程Machine能够发送消息。密钥与 DataId 相关联，并在访问控制模型中定义“对象”。如果无法验证为给定 DataID 收到的消息，则将丢弃该消息。因此，远程主体和本地客体之间的访问控制正在发生。

总而言之，在 SecOC 的情况下，远程主体的建模既不能通过增加主体识别的粒度来提供额外的好处，也不能通过对对象提供新的规则执行来提供额外的好处。

# 4 诊断设计

## 4.1 诊断映射

**4.1.1 概述**

*AUTOSAR 自适应平台*上的诊断配置通常通过在*AUTOSAR 经典平台上使用的诊断提取模板 [* 19 ]创建一个诊断提取来完成。

因此，Diagnostic Extract 中的概念应该同样适用于两个平台上的模型。

甚至可以有把握地预期，给定的诊断提取可以分为适用于构建在*AUTOSAR 经典平台之上的*ECU 的部分和适用于构建在*AUTOSAR 自适应平台之上且*都属于同一车辆的 ECU 的部分。

就本文档的适用性而言，与此上下文相关的诊断摘录部分是与诊断协议内容相关的信息定义与应用软件之间的映射。

为了举例说明该方法，图4.1中描绘的图表描述了一个非常简单的情况，其中两个不同的PPortPrototype可能由AdaptiveApplicationSwComponentType公开的两个不同的DiagnosticDataIdentifierInterface类型由*AUTOSAR 自适应平台*上的 AUTOSAR 自适应诊断管理访问，目的是访问整个 DID。

P



R

Application Software

Component

Diagnostic Manager

DiagnosticDataIdentifierInterface

P

DiagnosticDataPortMapping

DiagnosticDataIdentifier

Data

Element

Data

Element

Data

Element

Data

Element

**图 4.1：用于诊断目的的数据交换示例**

特别是， DiagnosticSwMapping的一个子类（在本例中为： DiagnosticDataPortMapping ）形式化了通信两端之间的“连接”。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***DiagnosticSwMapping*** （摘要） | | | |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping::ServiceMapping | | | |
| ***笔记*** | 这表示定义诊断信息（此时无法更具体地了解语义）到软件组件之间的映射的能力。 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMapping* ,  *识别*，*多语言可引用*，可打包元素，*可引用* | | | |
| ***子类*** | 诊断验证端口映射、诊断清除条件端口映射、诊断数据端口  映射、诊断启用条件端口映射、诊断事件端口映射、诊断外部  AuthenticationPortMapping ， DiagnosticFimFunctionMapping ， DiagnosticIndicatorPortMapping ，  DiagnosticMemoryDestinationPortMapping , DiagnosticMonitorPortMapping , DiagnosticOperationCycle  端口映射、诊断安全级别端口映射、诊断服务通用映射、诊断服务SwMapping、诊断服务验证映射 | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 4.1：DiagnosticsSwMapping**

*AUTOSAR 自适应平台*上启用诊断所需的另一种映射会稍微复杂一些。

在这种情况下，实现的用例可能涉及也可能不涉及多个通信端（以PortPrototype的形式）。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断数据标识符** | | | |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics | | | |
| ***笔记*** | 此元类表示对在配置时完全指定的关于有效负载的诊断数据标识符 (DID) 进行建模的能力。  **标签：** atp.recommendedPackage=DiagnosticDataIdentifiers | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticAbstractDataIdentifier* , *DiagnosticCommon*  *元素*,可*识别*, *MultilanguageReferable* , *PackageableElement* ,*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 数据元素 | 诊断参数 | \* | 聚合 | 这是与诊断关联的数据元素  数据标识符。  **刻板印象：** atpSplitable； atpVariation**标签：**  atp.Splitkey=dataElement.bitOffset, dataElement.variation Point.shortLabel  vh.latestBindingTime=postBuild |
| 大小 | 正整数 | 0..1 | 属性 | 此属性指示诊断数据标识符的字节大小。 |
| 代表Vin | 布尔值 | 0..1 | 属性 | 该属性指示特定的诊断数据标识符是否代表车辆标识。 |
| 支持信息字节 | 诊断支持信息  字节 | 0..1 | 聚合 | 此属性表示与 DiagnosticDataIdentifier 关联的受支持信息。 |

**表 4.2：诊断数据标识符**

*AUTOSAR 经典平台*上对这种情况的响应是SwcServiceDependency的定义，它允许将特定角色中的多个PortPrototype与给定用例相关联。

另一方面，在*AUTOSAR 经典平台*（使用通用ClientServerInterface 、 SenderReceiverInterface和ModeSwitchInterface ）上用于诊断用途的PortInterface设计与*AUTOSAR 自适应平台*（使用PortInterface专门针对相应的诊断用例量身定制）。

换句话说，如果使用了专门定制的PortInterface ，对定义给定服务用例建模范围的“保护伞”的需求将大大减少。

最重要的是，使用 InstanceSpecifier 作为标识元素严格绑定到PortPrototype而不是SwcServiceDependency 。换句话说，诊断管理使用instanceRefs 来识别应用软件中的“端点”。

instanceRef 和对应的 InstanceSpecifier 是强相关的。如果一个是已知的，则可以推断出另一个。该结论的结果是，可以在存在到应用软件模型的映射的情况下配置诊断管理，该应用软件模型本身在创建诊断管理配置时不必可访问。

这意味着一个 instanceRef 源自DiagnosticSwMapping的子类需要以构造型 atpUriDef 的形式进行额外装饰。

请注意映射目标[[13]](#footnote-13) 在一组诊断映射中可能存在于运行时的多个实例中。

这种多重实例化由元类Process的存在形式化（在设计层面又由元类ProcessDesign表示），见第3.19章。

很典型的是，一个应用软件的不同实例可能需要不同的诊断映射，而建模需要适应这个要求，即需要建立诊断映射和ProcessDesign之间的关系。

**[constr\_10002]** { DRAFT }每个**PortPrototype只有一个映射**d如果DiagnosticSwMapping的以下子类的一个实例引用PortPrototype ，则DiagnosticSwMapping的其他实例不应引用相同的PortPrototype ：

* DiagnosticMonitorInterface或DiagnosticEventInterface键入的RPortPrototype关联的DiagnosticEventPortMapping 。
* DiagnosticOperationCycleInterface键入的RPortPrototype关联的DiagnosticOperationCyclePortMapping 。
* DiagnosticConditionInterface键入的RPortPrototype关联的DiagnosticEnableConditionPortMapping 。
* DiagnosticConditionInterface键入的RPortPrototype关联的DiagnosticClearConditionPortMapping 。
* DiagnosticIndicatorInterface键入的RPortPrototype关联的DiagnosticIndicatorPortMapping 。
* DiagnosticDTCInformationInterface键入的RPortPrototype关联的DiagnosticMemoryDe stinationPortMapping 。
* DiagnosticSecurityLevelInterface键入的PPortPrototype关联的DiagnosticSecurityLevelPortMapping 。
* DiagnosticDataIdentifierInterface键入的PPortPrototype关联的DiagnosticDataPortMapping 。
* DiagnosticSecurityLevelInterface键入的PPortPrototype关联的DiagnosticSecurityLevelPortMapping 。
* DiagnosticServiceValidationInterface键入的PPortPrototype关联的DiagnosticServiceValidationMapping 。

### c ()

[ constr\_10002 ] 存在的基本原理是各个PortPrototype具有明确定义的功能。例如，它只能提供一个 DID 的内容，而不能提供任意数量的 DID 的内容。

对于这种情况，应应用DiagnosticServiceGenericMapping （参见第4.1.12节）。

请注意 [ constr\_10002 ] 不适用于DiagnosticServiceGenericMapping ，即不受 [ constr\_10002 ] 约束的PortPrototype可以被多个DiagnosticServiceGenericMapping引用。

换句话说，几个DiagnosticServiceGenericMapping引用同一个PortPrototype的能力是使DiagnosticServiceGenericMapping*通用的原因*。

此外，[ constr\_10002 ] 故意不应用于由DiagnosticDataElementInterface键入的PPortPrototype ，因为在这种上下文中执行的操作是无状态的，因此可以不受限制地从多个客户端调用。

**4.1.2 诊断监视器到端口映射**

**[TPS\_MANI\_01351]** { DRAFT }**在AUTOSAR 自适应平台上报告DiagnosticEvent的状态**d为了在***AUTOSAR 自适应平台****上报告诊断事件的状态，* DiagnosticEvent和一个RPortPrototype之间的关系是通过使用DiagnosticMonitorPortMapping创建的那是指

* 角色diagnosticEvent中的DiagnosticEvent ，
* 角色过程中的ProcessDesign ，以及
* 由角色 rPortPrototypeInExecutable 中的DiagnosticMonitorInterface键入的RPortPrototype 。此引用的目标不一定必须在配置 DM 时存在。然而，引用本身是构造PortPrototype将在运行时使用的相应 InstanceSpecifier 所必需的。

c *( RS\_MANI\_00005 , RS\_MANI\_00061 )*

支持报告诊断事件的用例如图4.2所示。

**[constr\_10047]** { DRAFT }对**DiagnosticMonitorPortMapping的适用性的限制**d如果RPortPrototype由DiagnosticMonitorPortMapping引用，则RPortPrototype应由DiagnosticMonitorInterface键入。 c *()*

**[constr\_10048]** { DRAFT }**存在从DiagnosticMonitorPortMapping到DiagnosticEvent** d的引用每个DiagnosticEvent只能由一个DiagnosticMonitorPortMapping引用。 c *()*

DiagnosticEvent

*AbstractExecutionContext*

Process

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

*ARElement*

ProcessDesign

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

*AbstractRequiredPortPrototype*

RPortPrototype

DiagnosticMonitorInterface

DiagnosticMonitorPortMapping

+

diagnosticEvent

0..1

«instanceRef,atpUriDef»

+

rPortPrototypeInExecutable

0..1

«isOfType»

requiredInterface

+

0..1

}

{

redefines atpType

+

design

0..1

«atpSplitable»

+

process

0..1

**图 4.2: DiagnosticMonitorPortMapping建模，用于报告*AUTOSAR 自适应平台上的诊断事件状态***

|  |  |
| --- | --- |
| ***班级*** | **诊断监视器端口映射** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断监视器端口映射** | | | |
| ***笔记*** | 定义诊断监视器映射到哪个 SWC 服务端口。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口映射 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMapping* ,  *DiagnosticSwMapping* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 诊断事件 | 诊断事件 | 0..1 | 参考 | 对分配给 SWC 服务端口的 DiagnosticEvent 的引用。  **标签：** atp.Status=draft |
| 过程 | 流程设计 | 0..1 | 参考 | 引用所需的流程表示，因为对于引用特定可执行文件的不同流程，映射可能不同。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=处理 atp.Status=草稿 |
| rPortPrototype InExecutable | RPort原型 | 0..1 | 参考 | 此聚合允许在 AUTOSAR 自适应平台上使用诊断 MonitorPortMapping。  **刻板印象：** atpUriDef  **标签：** atp.Status=draft  **InstanceRef 由：** RPortPrototypeIn实现  可执行实例引用 |

**表 4.3：诊断监视器端口映射**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断事件** | | | |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticEvent | | | |
| ***笔记*** | 此元素用于配置诊断事件。  **标签：** atp.recommendedPackage=DiagnosticEvents | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *Identifiable* , *Multilanguage Referrable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 相关事件  鉴别 | 正整数 | 0..1 | 属性 | 此属性表示与封闭的 DiagnosticEvent 相关联的标识号，并允许在放入快照记录或扩展数据记录存储时对其进行标识。  该值可以报告为快照记录或扩展数据记录中的内部数据元素。 |
| 允许清除事件  行为 | 诊断清除事件  AllowedBehaviorEnum | 0..1 | 属性 | 该属性定义了相关事件的 UDS 状态字节，不应根据 ClearEventAllowed 回调清除 |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断事件** | | | |
| 确认阈值 | 正整数 | 0..1 | 属性 | 此属性定义在确认的 DTC 设置为 1 之前具有失败结果的操作周期数。此属性的语义是与 ISO 14229 中提到的“行程计数器”的确认阈值相比增加了“1”的值-图 D.4 中的 1。值“1”定义 DTC 的立即确认以及第一次报告的失败。这有时也称为“零行程 DTC”。 “2”值定义第一次发生失败后在操作周期中的 DTC 确认。在美国，值“2”通常用于 OBD DTC 确认。  **刻板印象：** atpVariation  **标签：** vh.latestBindingTime=preCompileTime |
| 连接指示器 | 诊断连接  指标 | \* | 聚合 | 指标的事件特定描述。  **刻板印象：** atpSplitable； atpVariation**标签：**  atp.Splitkey=connectedIndicator.shortName，连接 Indicator.variationPoint.shortLabel vh.latestBindingTime=postBuild |
| 事件清除允许 | 诊断事件清除  允许枚举 | 0..1 | 属性 | 此属性定义 Dem 是否有权访问“ClearEventAllowed”回调。 |
| 预存  冻结帧 | 布尔值 | 0..1 | 属性 | 该属性描述了分配的事件是否支持冻结帧的预存储。  True: 支持 FreezeFrames 的 Prestorage  False：不支持冻结帧的预存储 |
| 预存冻结帧  StoredInNvm | 布尔值 | 0..1 | 属性 | 如果事件使用预先存储的冻结帧（使用服务接口 DiagnosticMonitor 的操作 PrestoreFreezeFrame 和 ClearPrestored FreezeFrame），则此属性指示事件是否需要将数据存储在非易失性存储器中。 TRUE = Dem 应将预先存储的数据存储在非易失性存储器中，FALSE = 关机时数据可能丢失（不存储在 Nvm 中） |
| 可恢复的  相同的操作  循环 | 布尔值 | 0..1 | 属性 | 如果该属性设置为 true，则报告 PASSED 将重置当前操作周期中测试失败的指示。如果该属性设置为 false 则报告  PASSED 将被忽略并且不会导致测试失败指示的重置。 |

**表 4.4：诊断事件**

**4.1.3 诊断事件到端口映射**

**[TPS\_MANI\_01048]** { DRAFT }将**DiagnosticEvent的状态检索到*AUTOSAR 自适应平台上的*PortPrototype (s)** d为了报告*AUTOSAR 自适应平台上诊断事件的状态，* DiagnosticEvent和一个RPortPrototype之间的关系是通过使用创建

DiagnosticEventPortMapping指的是

* 角色diagnosticEvent中的DiagnosticEvent ，
* 角色过程中的ProcessDesign ，以及
* 由角色 rPortPrototypeInExecutable 中的DiagnosticEventInterface键入的RPortPrototype 。此引用的目标不一定必须在配置 DM 时存在。然而，引用本身是构造PortPrototype将在运行时使用的相应 InstanceSpecifier 所必需的。

c *( RS\_MANI\_00005 , RS\_MANI\_00061 )*

**[constr\_10049]** { DRAFT }对**DiagnosticEvent的适用性的限制-**

**PortMapping** d如果RPortPrototype由DiagnosticEventPortMapping引用，则RPortPrototype应由DiagnosticEventInterface键入。 c *()*

4.3描述了支持检索有关诊断事件的信息的用例。

DiagnosticEventPortMapping

DiagnosticEvent

*AbstractExecutionContext*

Process

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

*ARElement*

ProcessDesign

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

*AbstractRequiredPortPrototype*

RPortPrototype

DiagnosticEventInterface

+

design

0..1

«isOfType»

requiredInterface

+

0..1

}

redefines atpType

{

«instanceRef,atpUriDef»

+

rPortPrototypeInExecutable

0..1

+

diagnosticEvent

0..1

«atpSplitable»

+

process

0..1

**图 4.3：在*AUTOSAR 自适应平台*上检索诊断事件信息的DiagnosticEventPortMapping建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断事件端口映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping | | | |
| ***笔记*** | 使用 DiagnosticEventInfoNeeds 定义 DiagnosticEvent 映射到哪些 SWC 服务端口。  **标签：** atp.recommendedPackage=DiagnosticMappings | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMapping* ,  *DiagnosticSwMapping* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 诊断事件 | 诊断事件 | 0..1 | 参考 | 对通过 DiagnosticEventInfoNeeds 分配给 SWC 服务端口的 DiagnosticEvent 的引用。 |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断事件端口映射** | | | |
| 过程 | 流程设计 | 0..1 | 参考 | 引用所需的流程表示，因为对于引用特定可执行文件的不同流程，映射可能不同。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=处理 atp.Status=草稿 |
| rPortPrototype InExecutable | RPort原型 | 0..1 | 参考 | 此聚合允许在 AUTOSAR 自适应平台上使用诊断 EventPortMapping。  **刻板印象：** atpUriDef  **标签：** atp.Status=draft  **InstanceRef 由：** RPortPrototypeIn实现  可执行实例引用 |

**表 4.5：诊断事件端口映射**

**4.1.4 诊断操作周期到端口映射**

[ **TPS\_MANI\_01049** ] { DRAFT } **DiagnosticOperationCycle映射到***AUTOSAR*自适应***平台***上的**PortPrototype** ( **s** ) \_ \_

* 角色operationCycle中的DiagnosticOperationCycle ，
* 角色过程中的ProcessDesign ，以及
* 由角色 rPortPrototypeInExecutable 中的DiagnosticOperationCycleInterface键入的RPortPrototype 。此引用的目标不一定必须在配置 DM 时存在。然而，引用本身是构造PortPrototype将在运行时使用的相应 InstanceSpecifier 所必需的。

c *( RS\_MANI\_00005 , RS\_MANI\_00061 )*

**[constr\_10050]** { DRAFT }对**DiagnosticOperationCyclePortMapping的适用性的限制**d如果RPortPrototype被DiagnosticOperationCyclePortMapping引用，则RPortPrototype应由DiagnosticOperationCycleInterface键入。 c *()*

**[constr\_10051]** { DRAFT }**从DiagnosticOperationCyclePortMapping到DiagnosticOperationCycle** d的引用存在每个DiagnosticOperationCycle只能被一个DiagnosticOperationCyclePortMapping c *()引用*

DiagnosticOperationCyclePortMapping

DiagnosticOperationCycle

*AbstractExecutionContext*

Process

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

*ARElement*

ProcessDesign

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticOperationCycleInterface

*AbstractRequiredPortPrototype*

RPortPrototype

«atpSplitable»

+

process

0..1

«isOfType»

requiredInterface

+

0..1

}

redefines atpType

{

+

operationCycle

0..1

«instanceRef,atpUriDef»

+

rPortPrototypeInExecutable

0..1

design

+

0..1

**图 4.4：在*AUTOSAR 自适应平台*上使用的DiagnosticOperationCyclePortMapping建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断操作周期** | | | |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticOperationCycle | | | |
| ***笔记*** | 操作周期的定义，它是事件限定和 Dem 调度的基础。  **标签：** atp.recommendedPackage=DiagnosticOperationCycles | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *Identifiable* , *Multilanguage Referrable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 循环状态存储 | 布尔值 | 0..1 | 属性 | 定义操作循环状态在电源循环期间是否可用（存储的非易失性）。   * true：操作周期状态存储为非易失性 * false：操作循环状态仅存储 volatile   此属性仅与 AUTOSAR 自适应平台相关。它在 AUTOSAR 经典平台上不再具有意义。 |
| 类型 | 诊断操作  循环类型枚举 | 0..1 | 属性 | Dem 的操作周期类型。 |

**表 4.6：诊断操作周期**

|  |  |
| --- | --- |
| ***班级*** | **诊断操作循环端口映射** |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping |
| ***笔记*** | 定义 DiagnosticOperationCycle 映射到具有 DiagnosticOperationCycleNeeds 的 SWC 服务端口。  **标签：** atp.recommendedPackage=DiagnosticMappings |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断操作循环端口映射** | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMapping* ,  *DiagnosticSwMapping* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 操作周期 | 诊断操作  循环 | 0..1 | 参考 | 对通过 DiagnosticOperation CycleNeeds 分配给 SWC 服务端口的 DiagnosticOperationCycle 的引用。 |
| 过程 | 流程设计 | 0..1 | 参考 | 引用所需的流程表示，因为对于引用特定可执行文件的不同流程，映射可能不同。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=处理 atp.Status=草稿 |
| rPortPrototype InExecutable | RPort原型 | 0..1 | 参考 | 此聚合允许在 AUTOSAR 自适应平台上使用诊断 OperationCyclePortMapping。  **刻板印象：** atpUriDef  **标签：** atp.Status=draft  **InstanceRef 由：** RPortPrototypeIn实现  可执行实例引用 |
| swcFlatService 依赖 | swc服务  依赖 | 0..1 | 参考 | 对将 ServiceNeeds 链接到 SWC 服务端口的 SwcServiceDependencyType 的引用。 |
| swc服务  依赖项  系统 | swc服务  依赖 | 0..1 | 参考 | 对将 ServiceNeeds 链接到 SWC 服务端口的 SwcServiceDependency 的实例引用。  **InstanceRef 实现者：** SwcServiceDependency InSystemInstanceRef |

**表 4.7：诊断操作循环端口映射**

**4.1.5 端口映射的诊断启用条件**

**[TPS\_MANI\_01050]** { DRAFT }**在*AUTOSAR 自适应平台上将*DiagnosticEnableCondition映射到PortPrototype** d在*AUTOSAR 自适应平台上，* DiagnosticEnableCondition和一个RPortPrototype之间的关系是通过使用DiagnosticEnableCondition-创建的

PortMapping指的是

* 角色enableCondition中的DiagnosticEnableCondition ，
* 角色过程中的ProcessDesign ，以及
* 由角色 rPortPrototypeInExecutable 中的DiagnosticConditionInterface键入的RPortPrototype 。此引用的目标不一定必须在配置 DM 时存在。然而，引用本身是构造PortPrototype将在运行时使用的相应 InstanceSpecifier 所必需的。

c *( RS\_MANI\_00005 , RS\_MANI\_00061 )*

**[constr\_10052]** { DRAFT }对**DiagnosticEnable的适用性的限制-**

**ConditionPortMapping** d如果RPortPrototype由DiagnosticEnableConditionPortMapping引用，则RPortPrototype应由DiagnosticConditionInterface键入。 c *()*

**[constr\_10053]** { DRAFT }**从DiagnosticEnableConditionPortMapping到DiagnosticEnableCondition** d的引用存在每个DiagnosticEnableCondition最多只能被一个DiagnosticEnableConditionPortMapping c *()引用*

DiagnosticEnableConditionPortMapping

DiagnosticEnableCondition

*AbstractExecutionContext*

Process

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

*DiagnosticCondition*

*ARElement*

ProcessDesign

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticConditionInterface

*AbstractRequiredPortPrototype*

RPortPrototype

+

design

0..1

«isOfType»

requiredInterface

+

0..1

}

{

redefines atpType

+

enableCondition

0..1

«instanceRef,atpUriDef»

+

rPortPrototypeInExecutable

0..1

«atpSplitable»

process

+

0..1

**图 4.5：在*AUTOSAR 自适应平台*上使用的DiagnosticEnableConditionPortMapping建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断启用条件** | | | |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticCondition | | | |
| ***笔记*** | 启用条件的规范。  **标签：** atp.recommendedPackage=DiagnosticConditions | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticCondition* ,  *识别*，*多语言可引用*，可打包元素，*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 4.8：诊断启用条件**

|  |  |
| --- | --- |
| ***班级*** | **诊断启用条件端口映射** |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断启用条件端口映射** | | | |
| ***笔记*** | 定义将 DiagnosticEnable Condition 映射到具有 DiagnosticEnableCondition 的 SWC 服务端口。  **标签：** atp.recommendedPackage=DiagnosticMappings | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMapping* ,  *DiagnosticSwMapping* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 启用条件 | 诊断启用  健康）状况 | 0..1 | 参考 | 对映射到的 EnableCondition 的引用  带有 DiagnosticEnableConditionNeeds 的 SWC 服务端口。 |
| 过程 | 流程设计 | 0..1 | 参考 | 引用所需的流程表示，因为对于引用特定可执行文件的不同流程，映射可能不同。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=处理 atp.Status=草稿 |
| rPortPrototype InExecutable | RPort原型 | 0..1 | 参考 | 此聚合允许在 AUTOSAR 自适应平台上使用诊断 EnableConditionPortMapping。  **刻板印象：** atpUriDef  **标签：** atp.Status=draft  **InstanceRef 由：** RPortPrototypeIn实现  可执行实例引用 |
| swcFlatService 依赖 | swc服务  依赖 | 0..1 | 参考 | 对将 ServiceNeeds 链接到 SWC 服务端口的 SwcServiceDependencyType 的引用。此参考可用于开发的早期阶段，以便在没有完整系统上下文的情况下识别 SwcServiceDependency。 |
| swc服务  依赖项  系统 | swc服务  依赖 | 0..1 | 参考 | 对将 ServiceNeeds 链接到 SWC 服务端口的 SwcServiceDependency 的实例引用。  **InstanceRef 实现者：** SwcServiceDependency InSystemInstanceRef |

**表 4.9：诊断启用条件端口映射**

**4.1.6 诊断清除条件到端口映射**

**[TPS\_MANI\_01259]** { DRAFT } **DiagnosticClearCondition到*AUTOSAR 自适应平台上的*PortPrototype (s) 的映射**在*AUTOSAR 自适应平台上，* DiagnosticClearCondition和一个RPortPrototype之间的关系是通过使用 DiagnosticClearConditionPortMapping 创建的，该关系指

* 角色clearCondition中的DiagnosticClearCondition ，
* 角色过程中的ProcessDesign ，以及
* 由角色 rPortPrototypeInExecutable 中的DiagnosticConditionInterface键入的RPortPrototype 。此引用的目标不一定必须在配置 DM 时存在。然而，引用本身是构造PortPrototype将在运行时使用的相应 InstanceSpecifier 所必需的。

c *( RS\_MANI\_00005 , RS\_MANI\_00061 )*

**[constr\_10054] {** DRAFT }诊断**的适用性限制-**

**ClearConditionPortMapping** d如果RPortPrototype由DiagnosticClearConditionPortMapping引用，则RPortPrototype应由DiagnosticConditionInterface键入。 c *()*

**[constr\_10055]** { DRAFT }**从DiagnosticClearConditionPortMapping到DiagnosticClearCondition** d的引用存在每个DiagnosticClearCondition最多只能被一个DiagnosticClearConditionPortMapping c *()引用*

*AbstractExecutionContext*

Process

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

*DiagnosticCondition*

*ARElement*

ProcessDesign

DiagnosticClearConditionPortMapping

DiagnosticClearCondition

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticConditionInterface

*AbstractRequiredPortPrototype*

RPortPrototype

design

+

0..1

«instanceRef,atpUriDef»

+

rPortPrototypeInExecutable

0..1

clearCondition

+

1

«isOfType»

+

requiredInterface

0..1

}

{

redefines atpType

«atpSplitable»

+

process

0..1

**图 4.6：在*AUTOSAR 自适应平台*上使用的DiagnosticClearConditionPortMapping建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断清除条件端口映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***笔记*** | 定义 DiagnosticsClearConditionNeeds 映射到哪些 SWC 服务端口。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断映射 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMapping* ,  *DiagnosticSwMapping* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 清除条件 | 诊断清除  健康）状况 | 1 | 参考 | 使用 DiagnosticClearConditionNeeds 对映射到 SWC 服务端口的 ClearCondition 的引用。  **标签：** atp.Status=draft |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断清除条件端口映射** | | | |
| 过程 | 流程设计 | 0..1 | 参考 | 引用所需的流程表示，因为对于引用特定可执行文件的不同流程，映射可能不同。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=处理 atp.Status=草稿 |
| rPortPrototype InExecutable | RPort原型 | 0..1 | 参考 | 此聚合允许在 AUTOSAR 自适应平台上使用诊断 ClearConditionMapping。  **刻板印象：** atpUriDef  **标签：** atp.Status=draft  **InstanceRef 由：** RPortPrototypeIn实现  可执行实例引用 |

**表 4.10：诊断清除条件端口映射**

**4.1.7 诊断指示灯到端口映射**

**[TPS\_MANI\_01260]** { DRAFT } ***AUTOSAR 自适应平台上*DiagnosticIndicator到PortPrototype (s) 的映射**d在*AUTOSAR 自适应平台上，* DiagnosticIndicator和一个RPortPrototype之间的关系是通过使用DiagnosticIndicatorPortMapping创建的，该关系指

* 角色指示器中的诊断指示器，
* 角色过程中的ProcessDesign ，以及
* 由角色 rPortPrototypeInExecutable 中的DiagnosticIndicatorInterface键入的RPortPrototype 。此引用的目标不一定必须在配置 DM 时存在。然而，引用本身是构造PortPrototype将在运行时使用的相应 InstanceSpecifier 所必需的。

c *( RS\_MANI\_00005 , RS\_MANI\_00061 )*

**[constr\_10056]** { DRAFT }对**DiagnosticIndicatorPortMapping的适用性的限制**d如果RPortPrototype由DiagnosticIndicatorPortMapping引用，则RPortPrototype应由DiagnosticIndicatorInterface键入。 c *()*

DiagnosticIndicator

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

DiagnosticIndicatorPortMapping

*AbstractExecutionContext*

Process

*ARElement*

ProcessDesign

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticIndicatorInterface

*AbstractRequiredPortPrototype*

RPortPrototype

«instanceRef,atpUriDef»

+

rPortPrototypeInExecutable

0..1

+

indicator

1

+

design

0..1

«isOfType»

requiredInterface

+

0..1

}

{

redefines atpType

«atpSplitable»

process

+

0..1

**图 4.7： DiagnosticIndicatorPortMapping的建模用于在**

***AUTOSAR自适应平台***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断指标端口映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***笔记*** | 定义将 DiagnosticIndicator 映射到具有 DiagnosticsIndicatorNeeds 的 SWC 服务端口。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断映射 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMapping* ,  *DiagnosticSwMapping* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 指标 | 诊断指标 | 1 | 参考 | 对通过 DiagnosticIndicatorNeeds 映射到 SWC 服务端口的 DiagnosticIndicator 的引用。  **标签：** atp.Status=draft |
| 过程 | 流程设计 | 0..1 | 参考 | 引用所需的流程表示，因为对于引用特定可执行文件的不同流程，映射可能不同。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=处理 atp.Status=草稿 |
| rPortPrototype InExecutable | RPort原型 | 0..1 | 参考 | 此聚合允许在 AUTOSAR 自适应平台上使用诊断指标映射。  **刻板印象：** atpUriDef  **标签：** atp.Status=draft  **InstanceRef 由：** RPortPrototypeIn实现  可执行实例引用 |

**表 4.11：诊断指标端口映射**

**4.1.8 诊断内存目的地到端口映射**

**[TPS\_MANI\_01261]** { DRAFT } ***AUTOSAR 自适应平台上*DiagnosticMemoryDe** stination**到PortPrototype (s) 的映射**在*AUTOSAR 自适应平台上，* DiagnosticMemoryDe stination和一个RPortPrototype之间的关系是通过使用

PortMapping指的是

* 角色 memoryDestination 中的DiagnosticMemoryDe stination ，
* 角色过程中的ProcessDesign ，以及
* 由角色 rPortPrototypeInExecutable 中的DiagnosticDTCInformationInterface键入的RPortPrototype 。此引用的目标不一定必须在配置 DM 时存在。然而，引用本身是构造PortPrototype将在运行时使用的相应 InstanceSpecifier 所必需的。

c *( RS\_MANI\_00005 , RS\_MANI\_00061 )*

**[constr\_10057]** { DRAFT } **DiagnosticMemory的适用性限制-**

**DestinationPortMapping** d如果RPortPrototype由DiagnosticMemoryDe stinationPortMapping 引用，则RPortPrototype应由DiagnosticDTCInformationInterface键入。 c *()*

*DiagnosticMemoryDestination*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticMapping*

*DiagnosticSwMapping*

DiagnosticMemoryDestinationPortMapping

*AbstractExecutionContext*

Process

*ARElement*

ProcessDesign

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticDTCInformationInterface

*AbstractRequiredPortPrototype*

RPortPrototype

«instanceRef,atpUriDef»

+

rPortPrototypeInExecutable

0..1

«atpSplitable»

+

process

0..1

«isOfType»

+

requiredInterface

0..1

{

redefines atpType

}

+

design

0..1

+

memoryDestination

1

**图 4.8：在*AUTOSAR 自适应平台*上使用的DiagnosticMemoryDe stinationPortMapping 建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断内存目标端口映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***笔记*** | 使用 DiagnosticsEventInfoNeeds 定义将 DiagnosticMemoryDestination 映射到哪些 SWC 服务端口。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断映射 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMapping* ,  *DiagnosticSwMapping* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 记忆  目的地 | 诊断记忆  目的地 | 1 | 参考 | 对 MemoryDestination 的引用，该 MemoryDestination 使用 DiagnosticEventInfoNeeds 映射到 SWC 服务端口。  **标签：** atp.Status=draft |
| 过程 | 流程设计 | 0..1 | 参考 | 引用所需的流程表示，因为对于引用特定可执行文件的不同流程，映射可能不同。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=处理 atp.Status=草稿 |
| rPortPrototype InExecutable | RPort原型 | 0..1 | 参考 | 此聚合允许在 AUTOSAR 自适应平台上使用诊断 MemoryDestinationMapping。  **刻板印象：** atpUriDef  **标签：** atp.Status=draft  **InstanceRef 由：** RPortPrototypeIn实现  可执行实例引用 |

**表 4.12：诊断内存目的地端口映射**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***DiagnosticMemoryDestination*** （摘要） | | | |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode | | | |
| ***笔记*** | 此抽象元类表示诊断事件的可能内存目的地。 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *Identifiable* , *Multilanguage Referrable* , *PackageableElement* , *Referrable* | | | |
| ***子类*** | DiagnosticMemoryDestinationPrimary、DiagnosticMemoryDestinationUserDefined | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 老化需要TestedCycle | 布尔值 | 0..1 | 属性 | 定义是否每个老化周期都处理老化周期计数器，否则只考虑测试的老化周期。  如果该属性设置为 TRUE：老化周期计数器只考虑测试的老化周期。  如果该属性设置为 FALSE：每个老化周期都会处理老化周期计数器。 |
| 清除Dtc  局限性 | 诊断清除Dtc  限制枚举 | 0..1 | 属性 | 定义 DEM\_ClearDTC Api 的范围。 |
| dtc状态  可用性掩码 | 正整数 | 0..1 | 属性 | Dem 支持的 DTC 状态位的掩码。 |
| 事件  移位  战略 | 诊断事件  置换策略  枚举 | 0..1 | 属性 | 此属性定义是否启用对事件置换的支持，以及遵循哪种置换策略。 |
| 最大数量  事件条目 | 正整数 | 0..1 | 属性 | 此属性固定故障存储器中的最大事件条目数。 |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***DiagnosticMemoryDestination*** （摘要） | | |  |
| 记忆入口  存储触发器 | 诊断内存条目  存储触发器枚举 | 0..1 | 属性 | 描述分配事件内存条目的触发器。 |
| 状态位  处理测试  失败自上次  清除 | 诊断状态位  处理测试失败  自上次清除枚举 | 0..1 | 属性 | 该属性定义是否将老化和置换机制应用于“Test FailedSinceLastClear”状态位。 |
| 状态位存储测试  失败的 | 布尔值 | 0..1 | 属性 | 此参数用于激活/停用“TestFailed”状态位的永久存储。  true：存储已激活 false：存储已停用 |
| typeOfFreeze FrameRecord  记数法 | 诊断类型  冻结帧记录  枚举枚举 | 0..1 | 属性 | 此属性定义为特定事件的冻结帧记录分配冻结帧记录编号的类型。 |

**表 4.13：诊断内存目标**

**4.1.9 端口映射的诊断安全性**

**[TPS\_MANI\_01262]** { DRAFT } **DiagnosticSecurityLevel映射到*AUTOSAR 自适应平台上的*PortPrototype (s) 在**AUTOSAR自适应*平台上，* DiagnosticSecurityLevel和一个PPortPrototype之间的关系是通过使用DiagnosticSecurityLevelPortMapping创建的

* 角色securityLevel中的DiagnosticSecurityLevel ，
* 角色过程中的ProcessDesign ，以及
* 由角色 pPortPrototypeInExecutable 中的DiagnosticSecurityLevelInterface键入的PPortPrototype 。此引用的目标不一定必须在配置 DM 时存在。然而，引用本身对于构造应用层上的PortPrototype将在运行时使用的相应 InstanceSpecifier 是必需的。

c *( RS\_MANI\_00005 , RS\_MANI\_00061 )*

**[constr\_10058]** { DRAFT }对**DiagnosticSecurityLevelPortMapping的适用性的限制**d如果PPortPrototype被DiagnosticSecurityLevelPortMapping引用，则PPortPrototype应由DiagnosticSecurityLevelInterface键入。 c *()*

**[constr\_10059]** { DRAFT }**从DiagnosticSecurityLevelPortMapping到DiagnosticSecurityLevel的引用存在**d每个DiagnosticSecurityLevel只能由一个DiagnosticSecurityLevelPortMapping引用。 c *()*

DiagnosticSecurityLevelPortMapping

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

DiagnosticSecurityLevel

*AbstractExecutionContext*

Process

*ARElement*

ProcessDesign

*AbstractProvidedPortPrototype*

PPortPrototype

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticSecurityLevelInterface

+

securityLevel

0..1

«instanceRef,atpUriDef»

+

pPortPrototypeInExecutable

0..1

design

+

0..1

«isOfType»

providedInterface

+

0..1

}

{

redefines atpType

«atpSplitable»

+

process

0..1

**图 4.9： DiagnosticSecurityLevelPortMapping的建模用于在**

***AUTOSAR自适应平台***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断安全级别端口映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***笔记*** | 定义映射到具有 DiagnosticsCommunicationSecurityNeeds 诊断安全级别的 SWC 服务端口。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断映射 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMapping* ,  *DiagnosticSwMapping* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| pPortPrototype InExecutable | PPort原型 | 0..1 | 参考 | 此聚合允许在 AUTOSAR 自适应平台上使用诊断安全级别映射。  **刻板印象：** atpUriDef  **标签：** atp.Status=draft  **InstanceRef 由：** PPortPrototypeIn实现  可执行实例引用 |
| 过程 | 流程设计 | 0..1 | 参考 | 引用所需的流程表示，因为对于引用特定可执行文件的不同流程，映射可能不同。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=处理 atp.Status=草稿 |
| 安全级别 | 诊断安全级别 | 0..1 | 参考 | 对映射到具有 DiagnosticCommunicationSecurity 需要的 SWC 服务端口的 SecurityLevel 的引用。  **标签：** atp.Status=draft |

**表 4.14：诊断安全级别端口映射**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断安全级别** | | | |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::Dcm | | | |
| ***笔记*** | 此元类表示定义考虑用于诊断目的的安全级别的能力。  **标签：** atp.recommendedPackage=DiagnosticSecurityLevels | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *Identifiable* , *Multilanguage Referrable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 访问数据记录大小 | 正整数 | 0..1 | 属性 | 这表示 GetSeed 中使用的 AccessDataRecord 的大小。单位：字节。 |
| 键大小 | 正整数 | 0..1 | 属性 | 这表示安全密钥的大小。单位：字节。 |
| num失败  安全访问 | 正整数 | 0..1 | 属性 | 这表示在激活延迟时间之后安全访问失败的次数。 |
| 安全延迟时间 | 时间价值 | 0..1 | 属性 | 这表示安全访问失败后的延迟时间。单位：秒。 |
| 种子大小 | 正整数 | 0..1 | 属性 | 这表示安全种子的大小。单位：字节。 |

**表 4.15：诊断安全级别**

**4.1.10 诊断数据标识符到端口映射**

*AUTOSAR 自适应平台*上的 DM具有一次访问整个DiagnosticDataIdentifier的能力。为了支持这种能力，引入了一个名为DiagnosticDataPortMapping的专用映射类。

[ **TPS\_MANI\_01263** ] { DRAFT } *AUTOSAR*自适应***平台*上DiagnosticDataIdentifier或DiagnosticDataElement到PortPrototype**的**映射**\_ \_ DiagnosticDataElement和一个PPortPrototype是通过使用DiagnosticDataPortMapping创建的，它指的是

* 角色diagnosticDataIdentifier中的DiagnosticDataIdentifier或
* 角色diagnosticDataElement中的DiagnosticDataElement 。 c *( RS\_MANI\_00005 , RS\_MANI\_00061 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断数据元素** | | | |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics | | | |
| ***笔记*** | 该元类表示描述要考虑用于诊断目的的具体数据的能力。 | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| arraySize 语义 | 数组大小语义  枚举 | 0..1 | 属性 | 该属性控制数组大小值的含义。 |
| maxNumberOf 元素 | 正整数 | 0..1 | 属性 | 该属性的存在将数据实例转换为数据数组。该属性根据数组可以采用多少元素来确定数组的大小。 |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断数据元素** |  |  |  |
| 缩放信息大小 | 正整数 | 0..1 | 属性 | DiagnosticData 的缩放信息的大小（以字节为单位）  元素（如果与 DiagnosticReadScalingDataBy 一起使用）  标识符 |
| swDataDef 道具 | SwDataDefProps | 0..1 | 聚合 | 该属性允许指定数据定义属性以支持例如计算公式和数据约束的定义。 |

**表 4.16：诊断数据元素**

如图4.10所示， DiagnosticDataPortMapping能够处理对整个 DID**或**仅 DID 元素的**访问。**

因此，需要 [ constr\_10003 ] 的存在来强制只有一个引用实际用于任何给定的DiagnosticDataPortMapping 。

**[constr\_10003]** { DRAFT } **DiagnosticDataPortMapping存在的限制。诊断数据标识符与诊断数据端口映射。 diagnosticDataElement** d对于每个DiagnosticDataPortMapping ，必须存在角色diagnosticDataIdentifier**或**diagnosticDataElement**中的引用。** c *()*

**[constr\_10060]** { DRAFT }**由DiagnosticDataPortMapping引用的PPortPrototype的PortInterface** d在角色DiagnosticDataPortMapping中引用的任何特定PPortPrototype 。 pPortPrototypeInExecutable应由以下任一类型键入

* 诊断数据标识符接口
* 诊断数据元素接口
* 诊断数据标识符通用接口c *()*

**[constr\_10061]** { DRAFT }**映射到DiagnosticDataIdentifierInterface ，**

**DiagnosticDataElementInterface或DiagnosticDataIdentifierGenericInterface** d所有PPortPrototype由以下任一类型键入

* 诊断数据标识符接口
* 诊断数据元素接口
* 诊断数据标识符通用接口

只能由**DiagnosticDataPortMapping**引用。没有其他DiagnosticSwMapping子类适用于此目的。 c *()*

**[TPS\_MANI\_01347]** { DRAFT }**在 DID 上下文中使用的DiagnosticDataElement的定义是从由DiagnosticDataElementInterface类型化的PPortPrototype获得的**d如果DiagnosticDataElement由DiagnosticParameter聚合，而 DiagnosticParameter 又由DiagnosticDataIdentifier聚合，并且DiagnosticDataElement也被DiagnosticDataPortMapping然后引用的DiagnosticDataElement （通过SwDataDefProps ）应引用属性baseTypeSize设置为的SwBaseType

* 8、
* 16，或
* 32

并且属性baseTypeEncoding设置为

* NONE（适用于 8、16 或 32 位）或
* 2C（适用于 8、16 或 32 位）或
* IEEE754（用于 32 位）。

在这种情况下，只能将DiagnosticDataElement定义为标量或数组（参见 [TPS\_DEXT\_01001]、[TPS\_DEXT\_01002]）。定义一个能力

可变大小数组只能用于 DID 的最后一个元素。 c *( RS\_MANI\_00061 )*

*DiagnosticMapping*

*DiagnosticSwMapping*

*ARElement*

*DiagnosticCommonElement*

DiagnosticDataPortMapping

DiagnosticDataIdentifier

*AbstractExecutionContext*

Process

*ARElement*

ProcessDesign

*DiagnosticAbstractDataIdentifier*

*AbstractProvidedPortPrototype*

PPortPrototype

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticDataIdentifierInterface

*DiagnosticAbstractDataIdentifierInterface*

*Identifiable*

DiagnosticDataElement

DiagnosticDataElementInterface

«instanceRef,atpUriDef»

pPortPrototypeInExecutable

+

0..1

+

design

0..1

diagnosticDataElement

+

0..1

«atpSplitable»

+

process

«isOfType»

providedInterface

+

0..1

{

}

redefines atpType

+

diagnosticDataIdentifier

0..1

**图 4.10：在*AUTOSAR 自适应平台*上使用的DiagnosticDataPortMapping建模**

**[TPS\_MANI\_01348]** { DRAFT }**在 DID 上下文中使用的DiagnosticDataElement的定义是从由DiagnosticDataIdentifierInterface类型化的PPortPrototype获得的**d如果DiagnosticDataIdentifier由DiagnosticDataPortMapping引用，则聚合的DiagnosticDataIdentifier 。数据元素。 dataElement （通过SwDataDefProps ）应指

SwBaseType属性baseTypeSize设置为

* 8、
* 16，或
* 32

并且属性baseTypeEncoding设置为

* NONE（适用于 8、16 或 32 位）或
* 2C（适用于 8、16 或 32 位）或
* IEEE754（用于 32 位）。

在这种情况下，只能将DiagnosticDataElement定义为标量或数组（参见 [TPS\_DEXT\_01001]、[TPS\_DEXT\_01002]）。定义一个能力

可变大小数组只能用于 DID 的最后一个元素。 c *( RS\_-*

*MANI\_00061 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断数据端口映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***笔记*** | 这个元类提供了定义对整个 DID 的诊断访问的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=DiagnosticServiceMappings | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMapping* ,  *DiagnosticSwMapping* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 诊断数据元素 | 诊断数据元素 | 0..1 | 参考 | 此参考代表适用的诊断数据元素。  **标签：** atp.Status=draft |
| 诊断数据标识符 | 诊断数据标识符 | 0..1 | 参考 | 此参考代表适用的诊断数据  标识符。  **标签：** atp.Status=draft |
| pPortPrototype InExecutable | PPort原型 | 0..1 | 参考 | 该参考标识了从中获取该数据的适用 PPortPrototype。该引用能够指向组件层次结构（在可能考虑 rootSoftwareComposition 的情况下）。  **刻板印象：** atpUriDef  **标签：** atp.Status=draft  **InstanceRef 由：** PPortPrototypeIn实现  可执行实例引用 |
| 过程 | 流程设计 | 1 | 参考 | 引用所需的流程表示，因为对于引用特定可执行文件的不同流程，映射可能不同。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=处理 atp.Status=草稿 |

**表 4.17：诊断数据端口映射**

**4.1.11 诊断通用 UDS 服务处理程序到端口映射**

可以将 UDS 服务的集合与给定的PPortPrototype相关联，目的是PPortPrototype可以处理相关联的服务。

通过在通用 UDS 处理程序和它们可以使用的服务之间创建专用关联，可以使用多个通用 UDS 处理程序并让每个处理程序只使用关联的服务。

从技术上讲，通用 UDS 处理的文档化建模的一个可能替代方法是完全避免映射，并预见到存在一个包罗万象的通用 UDS 处理程序。

这在很大程度上与基于 SoftwareCluster 定义的模块化软件安装的想法相矛盾（参见第14.2节）。

**4.1.12 诊断通用映射**

**[TPS\_MANI\_01326]** { DRAFT }**通用映射到 AUTOSAR 自适应平台上的DiagnosticServiceInstance d**在AUTOSAR 自适应平台上， DiagnosticServiceInstance和一个用于选择诊断服务的PPortPrototype之间的关系是通过使用元类DiagnosticServiceGenericMapping创建的，该元类引用

* 角色diagnosticServiceInstance中的DiagnosticServiceInstance ，
* 角色 pPortPrototypeInExecutable 中的PPortPrototype ，以及
* 角色过程中的一个ProcessDesign 。

c *( RS\_MANI\_00005 , RS\_MANI\_00061 )*

**[constr\_10062]** { DRAFT }可以由**DiagnosticServiceGenericMapping** d DiagnosticServiceGenericMapping**映射的DiagnosticServiceInstance**只能用于以下DiagnosticServiceInstance列表：

* 诊断EcuReset
* 诊断通信控制
* 诊断例程控制
* 诊断自定义服务实例
* 诊断请求上传
* 诊断请求下载c *()*

*DiagnosticServiceInstance*

*DiagnosticSwMapping*

DiagnosticServiceGenericMapping

*ARElement*

ProcessDesign

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*AbstractExecutionContext*

Process

*AbstractProvidedPortPrototype*

PPortPrototype

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

+

design

0..1

«instanceRef,atpUriDef»

pPortPrototypeInExecutable

+

0..1

«isOfType»

+

providedInterface

0..1

{

redefines atpType

}

diagnosticServiceInstance

+

0..1

«atpSplitable»

+

process

0..1

**图 4.11：诊断通用映射的建模**

请注意，服务DiagnosticDataTransfer和DiagnosticTransferExit是根据上传和下载的 UDS 请求顺序动态映射的。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断EcuReset** | | | |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EcuReset | | | |
| ***笔记*** | 这代表“ECU 重置”诊断服务的一个实例。  **标签：** atp.recommendedPackage=DiagnosticEcuResets | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticServiceInstance* ,  *识别*，*多语言可引用*，可打包元素，*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 自定义子  功能  数字 | 正整数 | 0..1 | 属性 | 如果不使用类别的标准化值，则该属性将用于定义自定义子功能编号。 |
| ecuResetClass | 诊断EcuReset  班级 | 0..1 | 参考 | 此引用证实了该特定具体类的角色 serviceClass 中的抽象引用。  因此，引用表示访问给定上下文中所有 DiagnosticEcuReset 之间共享属性的能力。 |

**表 4.18：DiagnosticEcuReset**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断通信控制** | | | |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommunicationControl | | | |
| ***笔记*** | 这代表“通信控制”诊断服务的一个实例。  **标签：** atp.recommendedPackage=DiagnosticCommunicationControls | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticServiceInstance* ,  *识别*，*多语言可引用*，可打包元素，*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断通信控制** |  |  |  |
| comControl 类 | 诊断通信控制  班级 | 0..1 | 参考 | 此引用证实了该特定具体类的角色 serviceClass 中的抽象引用。  因此，引用表示访问给定上下文中所有 DiagnosticComControl 之间共享属性的能力。 |
| 自定义子  功能  数字 | 正整数 | 0..1 | 属性 | 如果不使用类别的标准化值，则该属性将用于定义自定义子功能编号。 |

**表 4.19：DiagnosticComControl**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断例程控制** | | | |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::RoutineControl | | | |
| ***笔记*** | 这代表“常规控制”诊断服务的一个实例。  **标签：** atp.recommendedPackage=DiagnosticRoutineControls | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticServiceInstance* ,  *识别*，*多语言可引用*，可打包元素，*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 常规 | 诊断程序 | 1 | 参考 | 这是指适用的诊断程序。 |
| 例程控制类 | 诊断程序  控制类 | 0..1 | 参考 | 此引用证实了该特定具体类的角色 serviceClass 中的抽象引用。  因此，引用表示访问给定上下文中所有 DiagnosticRoutineControl 之间共享属性的能力。 |

**表 4.20：诊断程序控制**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断自定义服务实例** | | | |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CustomServiceInstance | | | |
| ***笔记*** | 此元类能够定义自定义诊断服务的实例。  **标签：** atp.recommendedPackage=DiagnosticCustomInstances | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticServiceInstance* ,  *识别*，*多语言可引用*，可打包元素，*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 自定义服务类 | 诊断定制  服务类 | 0..1 | 参考 | 引用相应的 DiagnosticCustom ServiceClass。 |

**表 4.21：诊断自定义服务实例**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断请求上传** | | | |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::MemoryByAddress | | | |
| ***笔记*** | 这表示“请求上传”诊断服务的一个实例。  **标签：** atp.recommendedPackage=DiagnosticMemoryByAdresss | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMemory*  *AddressableRangeAccess* , *DiagnosticMemoryByAddress* , *DiagnosticServiceInstance* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断请求上传** | |  |  |
| 请求上传类 | 诊断请求  上传类 | 0..1 | 参考 | 此引用证实了该特定具体类的角色 serviceClass 中的抽象引用。  因此，引用表示访问给定上下文中所有 DiagnosticRequestUpload 之间共享属性的能力。 |

**表 4.22：诊断请求上传**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断请求下载** | | | |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::MemoryByAddress | | | |
| ***笔记*** | 这表示“请求下载”诊断服务的一个实例。  **标签：** atp.recommendedPackage=DiagnosticMemoryByAdresss | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMemory*  *AddressableRangeAccess* , *DiagnosticMemoryByAddress* , *DiagnosticServiceInstance* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 要求  下载类 | 诊断请求  下载类 | 0..1 | 参考 | 此引用证实了该特定具体类的角色 serviceClass 中的抽象引用。  因此，引用表示访问给定上下文中所有 DiagnosticRequestDownload 之间共享属性的能力。 |

**表 4.23：诊断请求下载**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断服务通用映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***笔记*** | 这个元类代表了在自适应平台上为选择的诊断服务实现通用映射的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=DiagnosticServiceMappings | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMapping* ,  *DiagnosticSwMapping* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 诊断  服务实例 | 诊断服务  实例 | 0..1 | 参考 | 对映射到 SWC 服务端口的 ServiceInstance 的引用。  **标签：** atp.Status=draft |
| pPortPrototype InExecutable | PPort原型 | 0..1 | 参考 | 此聚合允许在 AUTOSAR 自适应平台上使用诊断服务通用映射。  **刻板印象：** atpUriDef  **标签：** atp.Status=draft  **InstanceRef 由：** PPortPrototypeIn实现  可执行实例引用 |
| 过程 | 流程设计 | 0..1 | 参考 | 引用所需的流程表示，因为对于引用特定可执行文件的不同流程，映射可能不同。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=处理 atp.Status=草稿 |

**表 4.24：诊断服务通用映射**

**4.1.13 诊断服务验证映射**

**[TPS\_MANI\_01352]** { DRAFT } **DiagnosticServiceValidation的定义-**

**映射**d在*AUTOSAR 自适应平台上*，存在元类DiagnosticServiceValidationMapping以指示引用的PPortPrototype用于在执行诊断服务之前执行特定于制造商或特定于供应商的检查。

PPortPrototype的数量是任意的，并且执行各个检查的顺序取决于部署级别的决定。

这是DiagnosticServiceValidationMapping （除了对ProcessDesign的引用）在 DEXT 上下文中没有对另一个设计对象的引用的主要原因。 c *( RS\_MANI\_00005 , RS\_MANI\_00061 )*

DiagnosticServiceValidationMapping

*AbstractProvidedPortPrototype*

PPortPrototype

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

*AbstractExecutionContext*

Process

functionClusterAffiliation: String

+

[0..1]

[0..1]

numberOfRestartAttempts: PositiveInteger

+

+

preMapping: Boolean

[0..1]

*ARElement*

ProcessDesign

*DiagnosticMapping*

*DiagnosticSwMapping*

DiagnosticServiceValidationInterface

«isOfType»

providedInterface

+

0..1

}

redefines atpType

{

«atpSplitable»

+

process

0..1

«instanceRef,atpUriDef»

+

pPortPrototypeInExecutable

0..1

design

+

0..1

**图 4.12：诊断验证映射的建模**

**[constr\_10063]** { DRAFT } **DiagnosticServiceValidationMapping的可能值。类别**d属性DiagnosticServiceValidationMapping的值。类别仅限于以下值：

**MANUFACTURER\_VALIDATION**封闭的DiagnosticServiceValidationMapping表示由制造商定义的验证。

**SUPPLIER\_VALIDATION**封闭的DiagnosticServiceValidationMapping表示由供应商定义的验证。

### c ()

**[constr\_10064]** { DRAFT }**存在DiagnosticServiceValidationMapping 。 pPortPrototypeInExecutable** d角色中引用的PPortPrototype

诊断服务验证映射。 pPortPrototypeInExecutable应由DiagnosticServiceValidationInterface键入。 c *()*

请注意，关于执行验证的顺序的最终决定是在部署阶段决定的，请参阅第14.2.2.4节。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断服务验证映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***笔记*** | 此元类提供了指定在处理诊断服务之前要执行的制造商/供应商检查的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口映射 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMapping* ,  *DiagnosticSwMapping* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| pPortPrototype InExecutable | PPort原型 | 0..1 | 参考 | 此映射标识由 DiagnosticValidationInterface 键入的 PortPrototype，其中执行特定于制造商/供应商的检查。  **刻板印象：** atpUriDef  **标签：** atp.Status=draft  **InstanceRef 由：** PPortPrototypeIn实现  可执行实例引用 |
| 过程 | 流程设计 | 0..1 | 参考 | 引用所需的流程表示，因为对于引用特定可执行文件的不同流程，映射可能不同。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=处理 atp.Status=草稿 |

**表 4.25：诊断服务验证映射**

**4.1.14 诊断认证映射**

**[TPS\_MANI\_01360]** { DRAFT }**创建两个诊断映射，在验证诊断客户端的上下文中**履行不同*的*角色在验证诊断客户端的上下文中：

* 元类DiagnosticAuthenticationPortMapping （在本节中描述）用于配置将身份验证请求从 DM 转发到充当身份验证管理器的某些应用程序软件。
* 元类DiagnosticExternalAuthenticationPortMapping用于配置将认证管理器软件的响应传回DM。 c *( RS\_MANI\_00005 , RS\_MANI\_00061 )*

请注意， 4.1.15节中描述了元类DiagnosticExternalAuthenticationPortMapping 。

图4.13描述了 [ TPS\_MANI\_01360 ]中提到的用于诊断客户端身份验证的两种不同角色。



R

Application Software

Component

Diagnostic Manager

DiagnosticAuthenti

-

cationInterface

P



P

R

DiagnosticExternal

-

AuthenticationInterface

Authentication Request

Authentication Confirm/Deny

DiagnosticAuthenti

-

cationMapping

DiagnosticExternalAuthenti

-

cationMapping

**图 4.13：诊断验证的示例数据交换**

**[TPS\_MANI\_01361]** { DRAFT }**支持诊断客户端的身份验证请求**d为了在*AUTOSAR 自适应平台上支持诊断客户端的身份验证请求，* DiagnosticAuthentication和一个RPortPrototype之间的关系是通过使用DiagnosticAuthenticationPortMapping创建的，该关系指

* 角色diagnosticAuthentication中的DiagnosticAuthentication ，
* 角色过程中的ProcessDesign ，以及
* 由角色 pPortPrototypeInExecutable 中的DiagnosticAuthenticationInterface键入的PPortPrototype 。此引用的目标不一定必须在配置 DM 时存在。然而，引用本身是构造PortPrototype将在运行时使用的相应 InstanceSpecifier 所必需的。

c *( RS\_MANI\_00005 , RS\_MANI\_00061 )*

**[constr\_10092]** { DRAFT }对**DiagnosticAuthenticationPortMapping的适用性的限制**d如果PPortPrototype被DiagnosticAuthenticationPortMapping引用，则PPortPrototype应由DiagnosticAuthenticationInterface c *()键入*

**[constr\_10093]** { DRAFT }**从DiagnosticAuthenticationPortMapping到DiagnosticAuthentication的引用存在**d每个DiagnosticAuthentication只能由一个DiagnosticAuthenticationPortMapping引用。 c *()*

*AbstractExecutionContext*

Process

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

*ARElement*

ProcessDesign

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

DiagnosticExternalAuthenticationInterface

DiagnosticAuthenticationPortMapping

*DiagnosticAuthentication*

*DiagnosticServiceInstance*

*AbstractProvidedPortPrototype*

PPortPrototype

+

diagnosticAuthentication

0..1

«instanceRef,atpUriDef»

+

pPortPrototypeInExecutable

0..1

«isOfType»

providedInterface

+

0..1

}

redefines atpType

{

«atpSplitable»

+

process

0..1

+

design

0..1

**图 4.14： DiagnosticAuthenticationPortMapping的建模，用于将诊断客户端的身份验证请求转发到身份验证管理器**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断认证端口映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***笔记*** | 此映射类标识处理客户端身份验证的应用程序软件中的 PortPrototype。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口映射 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMapping* ,  *DiagnosticSwMapping* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 诊断  验证 | 诊断  验证 | 0..1 | 参考 | 对分配给 SWC 服务端口的 DiagnosticAuthentication 的引用。  **标签：** atp.Status=draft |
| pPortPrototype InExecutable | PPort原型 | 0..1 | 参考 | 此聚合允许在 AUTOSAR 自适应平台上使用诊断 AuthenticationPortMapping。  **标签：** atp.Status=draft  **InstanceRef 由：** PPortPrototypeIn实现  可执行实例引用 |
| 过程 | 流程设计 | 0..1 | 参考 | 引用所需的流程表示，因为对于引用特定可执行文件的不同流程，映射可能不同。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=处理 atp.Status=草稿 |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断认证端口映射** | | |  |
| rPortPrototype InExecutable | RPort原型 | 0..1 | 参考 | 此聚合允许在 AUTOSAR 自适应平台上使用诊断 AuthenticationPortMapping。  **标签：** atp.Status=draft  **InstanceRef 由：** RPortPrototypeIn实现  可执行实例引用 |

**表 4.26：诊断验证端口映射**

**4.1.15 诊断外部认证映射**

**[TPS\_MANI\_01362]** { DRAFT }**支持响应诊断客户端的身份验证请求**d为了支持在*AUTOSAR 自适应平台上传输对诊断客户端的身份验证请求的响应，* DiagnosticAuthentication和一个RPortPrototype之间的关系是通过使用引用的DiagnosticExternalAuthenticationPortMapping创建的

* 的DiagnosticAuthentication ，
* 的ProcessDesign ，以及
* 由角色 rPortPrototypeInExecutable 中的DiagnosticExternalAuthenticationInterface键入的RPortPrototype 。此引用的目标不一定必须在配置 DM 时存在。然而，引用本身是构造PortPrototype将在运行时使用的相应 InstanceSpecifier 所必需的。

c *( RS\_MANI\_00005 , RS\_MANI\_00061 )*

**[constr\_10094]** { DRAFT }对**DiagnosticExternalAuthenticationPortMapping的适用性的限制**d如果一个RPortPrototype由DiagnosticExternalAuthenticationPortMapping引用，则RPortPrototype应由DiagnosticExternalAuthenticationInterface c *()键入*

**[constr\_10095]** { DRAFT }**从DiagnosticExternalAuthenticationPortMapping到DiagnosticAuthentication的引用存在**d每个DiagnosticAuthentication只能由一个DiagnosticExternalAuthenticationPortMapping引用。 c *()*

*AbstractExecutionContext*

Process

*DiagnosticMapping*

*ARElement*

*DiagnosticCommonElement*

*DiagnosticSwMapping*

*ARElement*

ProcessDesign

*ARElement*

*AtpBlueprint*

*AtpBlueprintable*

*AtpType*

*PortInterface*

*DiagnosticPortInterface*

*AbstractRequiredPortPrototype*

RPortPrototype

DiagnosticExternalAuthenticationPortMapping

DiagnosticExternalAuthenticationInterface

*DiagnosticAuthentication*

*DiagnosticServiceInstance*

+

diagnosticAuthentication

0..1

design

+

0..1

«instanceRef,atpUriDef»

+

rPortPrototypeInExecutable

0..1

«isOfType»

requiredInterface

+

0..1

redefines atpType

{

}

«atpSplitable»

+

process

0..1

**图 4.15： DiagnosticsExternalAuthenticationPortMapping的建模，用于将身份验证请求的响应发送回 DM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断外部身份验证端口映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping | | | |
| ***笔记*** | 此映射类标识处理外部身份验证的应用程序软件中的 PortPrototype。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断端口映射 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMapping* ,  *DiagnosticSwMapping* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 诊断  验证 | 诊断  验证 | 0..1 | 参考 | 对分配给 SWC 服务端口的 DiagnosticAuthentication 的引用。  **标签：** atp.Status=draft |
| 过程 | 流程设计 | 0..1 | 参考 | 引用所需的流程表示，因为对于引用特定可执行文件的不同流程，映射可能不同。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=处理 atp.Status=草稿 |
| rPortPrototype InExecutable | RPort原型 | 0..1 | 参考 | 此聚合允许在 AUTOSAR 自适应平台上使用诊断 ClientAuthenticationPortMapping。  **标签：** atp.Status=draft  **InstanceRef 由：** RPortPrototypeIn实现  可执行实例引用 |

**表 4.27：诊断外部身份验证端口映射**

## 4.2 诊断清除条件

在*AUTOSAR自适应平台上*，引入了一个类似DiagnosticEnableCondition的新模型元素： DiagnosticClearCondition 。

*DiagnosticTroubleCode*

DiagnosticTroubleCodeUds

*DiagnosticCommonElement*

DiagnosticEvent

*DiagnosticMapping*

DiagnosticEventToTroubleCodeUdsMapping

*DiagnosticConditionGroup*

DiagnosticClearConditionGroup

*DiagnosticMapping*

DiagnosticTroubleCodeUdsToClearConditionGroupMapping

*DiagnosticCondition*

DiagnosticClearCondition

+

troubleCodeUds

0..1

«atpSplitable»

+

clearCondition

0..\*

+

clearConditionGroup

0..1

diagnosticEvent

+

0..1

+

troubleCodeUds

0..1

**图 4.16：诊断清除条件建模**

与DiagnosticEnableCondition相比， DiagnosticClearCondition不映射到DiagnosticEvent ，而是（通过DiagnosticClearConditionGroup的聚合）映射到DiagnosticTroubleCodeUds 。

为此，已经定义了元类DiagnosticTroubleCodeUdsToClearConditionGroupMapping 。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断清除条件** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition | | | |
| ***笔记*** | 此元类描述了用于诊断目的的明确条件。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断条件 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticCondition* ,  *识别*，*多语言可引用*，可打包元素，*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 4.28：诊断清除条件**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断清除条件组** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition | | | |
| ***笔记*** | 清除条件组，包括一个或多个清除条件。  **标签：**  atp.Status=草稿  atp.recommendedPackage=DiagnosticClearConditionGroups | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticConditionGroup* ,  *识别*，*多语言可引用*，可打包元素，*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 清除条件 | 诊断清除  健康）状况 | \* | 参考 | 此聚合表示属于 DiagnosticClear ConditionGroup 的 Diagnostic ClearCondition 的集合。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=clearCondition atp.Status=草稿 |

**表 4.29：诊断清除条件组**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **DiagnosticTroubleCodeUdsToClearConditionGroupMapping** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition | | | |
| ***笔记*** | 此元类提供将 DiagnosticClearConditionGroup 映射到 Diagnostic TroubleCodeUds 集合的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=诊断映射 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMapping* ,  *识别*，*多语言可引用*，可打包元素，*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| clearCondition 组 | 诊断清除  条件组 | 0..1 | 参考 | 该参考标识了适用的诊断清除条件组。  **标签：** atp.Status=draft |
| 麻烦代码Uds | 诊断故障码  UD | 0..1 | 参考 | 此参考标识与映射相关的 DiagnosticTroubleCodeUds。  **标签：** atp.Status=draft |

**表 4.30：DiagnosticTroubleCodeUdsToClearConditionGroupMapping**

**[constr\_1658]** { DRAFT }每个DiagnosticTroubleCodeUds的**DiagnosticTroubleCodeUdsToClearConditionGroupMapping元素的数量**d每个**DiagnosticTroubleCodeUds**映射元素DiagnosticTroubleCodeUdsToClearConditionGroupMapping的创建不应超过一次。

如果定义了几个引用相同DiagnosticTroubleCodeUds的DiagnosticTroubleCodeUdsToClearConditionGroupMapping元素，则清除条件组映射应被视为有缺陷。 c *()*

## 4.3 安全访问

在自适应平台上实现诊断管理器需要对元类DiagnosticSecurityAccessClass进行精细建模。

名为sharedTimer的新属性，该属性控制单个计时器是否用于所有安全访问级别，或者各个级别是否分别使用单独的计时器。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断安全访问类** | | | |
| ***包裹*** | M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::SecurityAccess | | | |
| ***笔记*** | 此元类包含由“安全访问”诊断服务的所有实例共享的属性。  **标签：** atp.recommendedPackage=DiagnosticSecurityAccesss | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticServiceClass* ,  *识别*，*多语言可引用*，可打包元素，*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断安全访问类** | |  |  |
| 共享定时器 | 布尔值 | 0..1 | 属性 | 在单独或单个共享计时器实例和计时器值之间切换。   * True：对所有安全访问级别组合使用共享计时器实例和计时器值。 * False：为每个安全级别使用单独的计时器实例和计时器值。   **标签：** atp.Status=draft |

**表 4.31：诊断安全访问类**

*DiagnosticCommonElement*

*DiagnosticServiceInstance*

DiagnosticSecurityAccess

[0..1]

requestSeedId: PositiveInteger

+

[0..1]

securityDelayTimeOnBoot: TimeValue

+

*DiagnosticCommonElement*

DiagnosticSecurityLevel

accessDataRecordSize: PositiveInteger

+

[0..1]

+

[0..1]

keySize: PositiveInteger

[0..1]

numFailedSecurityAccess: PositiveInteger

+

[0..1]

securityDelayTime: TimeValue

+

+

seedSize: PositiveInteger

[0..1]

*DiagnosticCommonElement*

DiagnosticAccessPermission

*DiagnosticCommonElement*

*DiagnosticServiceClass*

DiagnosticSecurityAccessClass

+

sharedTimer: Boolean

[0..1]

*DiagnosticCommonElement*

DiagnosticAuthRole

+

bitPosition: PositiveInteger

[0..1]

+

[0..1]

isDefault: Boolean

authenticationRole

+

0..\*

+

securityLevel

0..\*

+

accessPermission

0..1

«atpAbstract»

+

serviceClass

0..1

«atpSplitable»

securityLevel

+

0..1

+

securityAccessClass

0..1

}

redefines serviceClass

{

**图 4.17：诊断安全访问的改进建模**

## 4.4 诊断提供的数据映射

**[TPS\_MANI\_01230]** { DRAFT } **DiagnosticProvidedDataMapping的语义**d元类DiagnosticProvidedDataMapping似乎不满足表示映射类的条件，因为它在角色dataElement中只有一个对DiagnosticDataElement的引用。

但是，此映射的特定性质是，应该在映射中发生的第二个元素（ DiagnosticProvidedDataMapping.dataProvider ）不能精确地建模为单个元类。

因此，没有比对DiagnosticProvidedDataMapping建模更好的方法了。 DataProvider通过NameToken 。 c *( RS\_MANI\_00005 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **诊断提供的数据映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticProvidedDataMapping | | | |
| ***笔记*** | 这表示能够基于无法显式建模的数据提供者为 DiagnosticDataElement 定义数据访问的性质。  **标签：**  atp.Status=草稿  atp.recommendedPackage=数据映射 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *DiagnosticCommonElement* , *DiagnosticMapping* ,  *识别*，*多语言可引用*，可打包元素，*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 数据元素 | 诊断数据元素 | 0..1 | 参考 | 这表示 DiagnosticDataElement 的访问权限由 DiagnosticProvidedData Mapping.dataProvider 进一步限定。  **标签：** atp.Status=draft |
| 数据提供者 | 名称令牌 | 1 | 属性 | 这代表了进一步指定数据提供者的能力。  **标签：** atp.Status=draft |

**表 4.32：DiagnosticProvidedDataMapping**

请注意属性DiagnosticProvidedDataMapping的标准化值列表。 dataProvider在 SWS 诊断 [ 20 ] 中定义。

*DiagnosticMapping*

DiagnosticProvidedDataMapping

+

dataProvider: NameToken

*Identifiable*

DiagnosticDataElement

+

arraySizeSemantics: ArraySizeSemanticsEnum

[0..1]

+

maxNumberOfElements: PositiveInteger

[0..1]

+

scalingInfoSize: PositiveInteger

[0..1]

+

dataElement

0..1

**图 4.18： DiagnosticProvidedDataMapping的建模**

## 4.5 诊断杂项

**[TPS\_MANI\_01358]** { DRAFT }**诊断去抖动配置的限制**d诊断去抖动配置的限制，如[TPS\_DEXT\_03004] 中所述，**不适**用于*AUTOSAR 自适应平台*。 c *( RS\_MANI\_00005 )*

# 5 系统设计

## 5.1 概述

典型的车辆很可能配备在 AUTOSAR 经典平台上开发的 ECU 和在 AUTOSAR 自适应平台上开发的机器。因此，整个车辆的系统设计必须涵盖两种平台部署类型。

AUTOSAR 支持系统设计描述，可以描述将在系统中使用的两个 AUTOSAR 平台的软件组件，如果可能的话，甚至允许指示它们之间的面向服务的通信。

尤其是在以协调方式描述 AUTOSAR 经典和自适应 ECU 的通信行为时，系统设计的概念成为一个特殊的焦点。

所有系统设计方面的共同点是它们必须同时应对 AUTOSAR 经典 ECU 和自适应机器。这种跨学科系统的基本设计方面必须已经在 AUTOSAR 经典建模方法中可用，否则它们将不适用于两个平台。

因此，直接以现有的元类System作为对此类混合系统建模的起点。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **系统** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate | | | |
| ***笔记*** | 系统描述的顶级元素。  **标签：** atp.recommendedPackage=系统 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpClassifier* , *AtpFeature* , *AtpStructureElement* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 纤维元素 | 纤维元素 | \* | 参考 | 参考指定通信和拓扑的 ASAM FIBEX 元素。  系统描述中使用的所有 Fibex 元素都应从系统元素中引用。  atpVariation：为了描述产品线，所有 Fibex 元素都可以是可选的。  **刻板印象：** atpVariation  **标签：** vh.latestBindingTime=postBuild |
| 插值程序  映射集 | 插值程序  映射集 | \* | 参考 | 该参考标识了在封闭系统的上下文中相关的 InterpolationRoutineMapping 集。 |
| 映射 | 系统映射 | \* | 聚合 | 系统描述中相关的所有映射方面的聚合。  **刻板印象：** atpSplitable； atpVariation**标签：**  atp.Splitkey=mapping.shortName，mapping.variation Point.shortLabel  vh.latestBindingTime=postBuild |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **系统** |  |  |  |
| pnc向量长度 | 正整数 | 0..1 | 属性 | 部分联网请求释放信息向量的长度（以字节为单位）。 |
| pncVectorOffset | 正整数 | 0..1 | 属性 | 部分联网请求释放信息向量的绝对偏移量（相对于 NM-PDU），以字节为单位定义为从 0 开始的索引。 |
| root软件构成 | RootSw组合  原型 | 0..1 | 聚合 | 根软件组合的聚合，在层次结构中包含系统中的所有软件组件。当系统描述用于仅网络用例时，不需要此元素。  atpVariation：RootSwCompositionPrototype 可以变化。  **刻板印象：** atpSplitable； atpVariation**标签：**  atp.Splitkey=rootSoftwareComposition.shortName，根 SoftwareComposition.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime |
| 系统版本 | 修订标签字符串 | 1 | 属性 | 系统描述的版本号。 |

**表 5.1：系统**

有一些用例可以在不同的方法阶段和不同的开发方之间交换这样的SYSTEM\_DESIGN\_DESCRIPTION的一部分，因此 AUTOSAR 支持进一步的系统类别。

例如，一种常见的方法是 OEM 为设计 ECU 提供基础，然后由供应商推进。因此 Classic AUTOSAR 支持

系统类别，如 ECU\_EXTRACT 或 ECU\_SYSTEM\_DESCRIPTION，范围内只有一个 ECU。经典平台系统。类别在 [TPS\_SYST\_01003] 中定义。

AUTOSAR 自适应平台使用相同的方法。如果 OEM 想要提供与单个机器的配置相关的设计工件，所有不必要的信息都将从系统中剥离，并使用categorySYSTEM\_DESIGN\_DESCRIPTION并提供子系统的定义。

**[TPS\_MANI\_01274]** { DRAFT }**的标准化系统类别定义**

**Adaptive Platform** d标准化系统类别定义见表5.2 c *()*

|  |  |
| --- | --- |
| 类别 | 意义 |
| SYSTEM\_DESIGN\_DESCRIPTION | 包含与自适应平台和经典平台相关的设计工件的系统描述。 |
| MACHINE\_DESIGN\_EXTRACT | 包含与单个自适应平台机器相关的设计工件的系统描述。 |

**表 5.2：标准化系统类别定义**

**[constr\_3421]** { DRAFT } **Fibex 元素适用于类别MA-的系统**

**CHINE\_DESIGN\_EXTRACT** d具有类别MACHINE\_DESIGN\_EXTRACT的系统可以引用以下fibexElement ：

* 通信集群
* 机器设计
* 全球时域
* NmConfig
* 允许仅包含PncMapping的SystemMapping

c *()*

## 5.2 通信系统结构规范

当为车辆系统设计通信交互时，重点放在网络和连接的 ECU 上。连接到网络的特定 ECU 是使用 AUTOSAR 经典还是 AUTOSAR 自适应来实现，不会影响主要的通信设计。

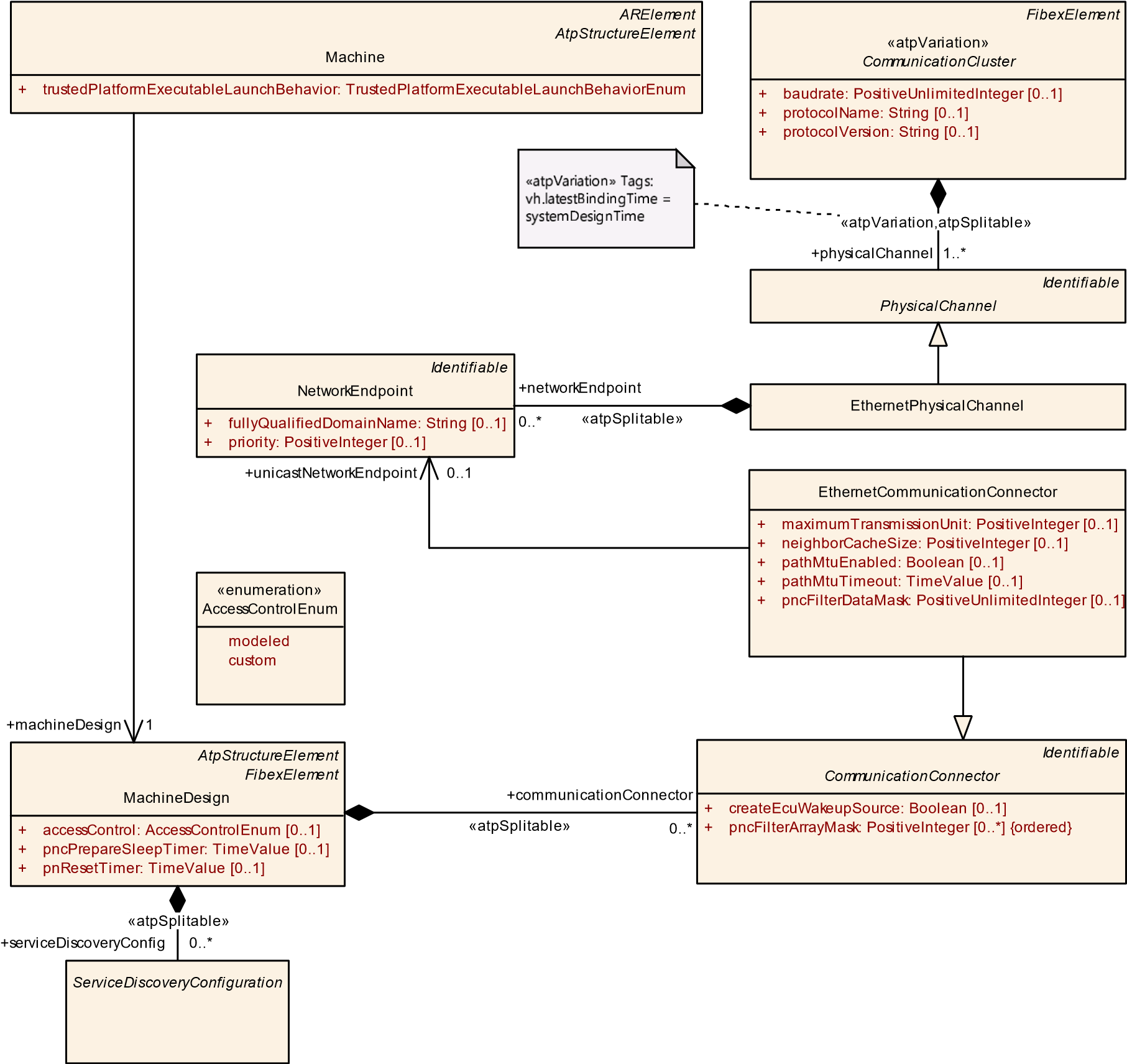
但当然，从汽车制造商的角度来看，一个特定的 ECU 是使用 AUTOSAR 经典版还是自适应版来实现是至关重要的。因此，已经在系统设计级别需要指定用于实现 ECU 的 AUTOSAR 平台类型。

在 AUTOSAR 经典中，元素EcuInstance用于在系统设计中定义一个 ECU。

在 AUTOSAR 自适应中， Machine元素是一个实体，它已经代表了具有专用配置的特定 ECU 实现，例如Processor s、 ProcessorCore s。

机器是一种模型实体，它不是通信设计人员关注的重点，不应在系统设计期间使用。

因此，引入MachineDesign是为了让通信系统设计人员能够在系统范围内为自适应 ECU 定义占位符（ MachineDesign对应于 AUTOSAR 经典的EcuInstance ）。



**图 5.1：机器设计**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **机器设计** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign | | | |
| ***笔记*** | 这个元类代表了在设计系统的上下文中定义机器需求的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=机器设计 | | | |
| ***根据*** | *ARObject* , *AtpClassifier* , *AtpFeature* , *AtpStructureElement* , *CollectableElement* , *FibexElement* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 访问控制 | 访问控制枚举 | 0..1 | 属性 | 此属性定义如何定义对服务实例的访问限制。  **标签：** atp.Status=draft |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **机器设计** | | | |
| 通讯连接器 | 沟通  连接器 | \* | 聚合 | 这个聚合定义了机器的网络连接。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=communicationConnector.shortName，communicationConnector.variationPoint.shortLabel atp.Status=draft |
| ethIpProps | EthIpProps | 0..1 | 参考 | 机器特定的 IP 属性。  **标签：** atp.Status=draft |
| pnc准备睡眠定时器 | 时间价值 | 0..1 | 属性 | PNC 状态机应在 PNC\_PREPARE\_SLEEP 中等待的时间（以秒为单位）。  **标签：** atp.Status=draft |
| pn复位定时器 | 时间价值 | 0..1 | 属性 | 以秒为单位指定重置计时器的运行时间。此复位时间对 PN 请求的复位有效。  **标签：** atp.Status=draft |
| 服务发现  配置 | 服务发现  配置 | \* | 聚合 | 在机器上为各个通信连接器定义的一组服务发现配置设置。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=serviceDiscoveryConfig atp.Status=draft |
| tcpIpIcmpProps | EthTcpIpIcmpProps | 0..1 | 参考 | 机器特定的 ICMP（Internet 控制消息协议）属性  **标签：** atp.Status=draft |
| tcpIpProps | EthTcpIpProps | 0..1 | 参考 | 机器特定的 TcpIp 堆栈属性。  **标签：** atp.Status=draft |

**表 5.3：机器设计**

**[TPS\_MANI\_03209]** { DRAFT } **MachineDesign的含义。**访问控制d **MachineDesign** 。 accessControl定义访问控制是由应用程序设计中的 AUTOSAR 方式定义的，带有receiverIntent （参见 [ TPS\_MANI\_01106 ]）和senderIntent （参见 [ TPS\_MANI\_01107 ]），还是由非 AUTOSAR 进程创建的自定义列表定义。 c *( RS\_MANI\_00034 )*

|  |  |
| --- | --- |
| ***枚举*** | **访问控制枚举** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment |
| ***笔记*** | 此枚举描述了用于定义对资源的访问限制的选项。  **标签：** atp.Status=draft |
| ***文字*** | ***描述*** |
| 风俗 | 对资源的访问限制由非 AUTOSAR 进程定义。  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |

5

**表 5.4：访问控制枚举**

**5.2.1 网络连接**

MachineDesign的上下文中定义的最突出的信息之一是网络连接。由于*AUTOSAR 自适应平台*专注于使用以太网进行通信，这归结为 IP 地址的规范。

具体来说， MachineDesign连接性的基本定义是通过在角色communicationConnector中聚合抽象基类CommunicationConnector来创建的。在此上下文中使用的CommunicationConnector的特定子类是EthernetCommunicationConnector 。

|  |  |
| --- | --- |
| ***枚举*** | **访问控制枚举** |
| 建模 | 对资源的访问限制在 AUTOSAR 应用程序设计模型或 AUTOSAR 部署模型中建模。  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |

EthernetCommunicationConnector用于将MachineDesign与VLAN连接，该VLAN在 AUTOSAR 中由作为EthernetCluster一部分的EthernetPhysicalChannel表示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***PhysicalChannel*** （抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology | | | |
| ***笔记*** | 此元素表示通信设备之间的物理连接（在 CAN、FlexRay、LIN 的情况下）或逻辑连接（在以太网的情况下的 VLAN）。 | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***子类*** | *AbstractCanPhysicalChannel* , EthernetPhysicalChannel , FlexrayPhysicalChannel , UserDefinedPhysical  渠道 | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 5.5：物理通道**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | <<atpVariation>>**以太网集群** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | 特定于以太网的集群属性。  **标签：** atp.recommendedPackage=CommunicationClusters | | | |
| ***根据*** | *ARObject* , *CollectableElement* , *CommunicationCluster* , *FibexElement* , *Identifiable* , *Multilanguage Referrable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | <<atpVariation>>**以太网集群** | | | |
| 耦合端口连接 | 耦合端口  联系 | \* | 聚合 | CouplingElements 和 EcuInstances 之间的连接规范。  注意：由于 atpVariation (PropertySetPattern)，此 atpSplitable 属性没有 atp.Splitkey。  **刻板印象：** atpSplitable； atp变体  **标签：** vh.latestBindingTime=postBuild |
| 耦合端口  启动活动  时间 | 时间价值 | 0..1 | 属性 | 该属性以秒为单位指定打开耦合端口以使主机 ECU（维护以太网交换机的 ECU）能够侦听网络以获取潜在的网络管理请求。 |
| 耦合端口  关断延迟 | 时间价值 | 0..1 | 属性 | 以秒为单位关闭 CouplingPorts 的延迟。它表示在发出关闭耦合端口的请求后关闭耦合端口的延迟。 （例如关闭以太网交换机端口）。 |
| mac组播组 | MacMulticastGroup | \* | 聚合 | 为子网 (EthernetCluster) 定义的 MacMulticastGroup。 |

**表 5.6：以太网集群**

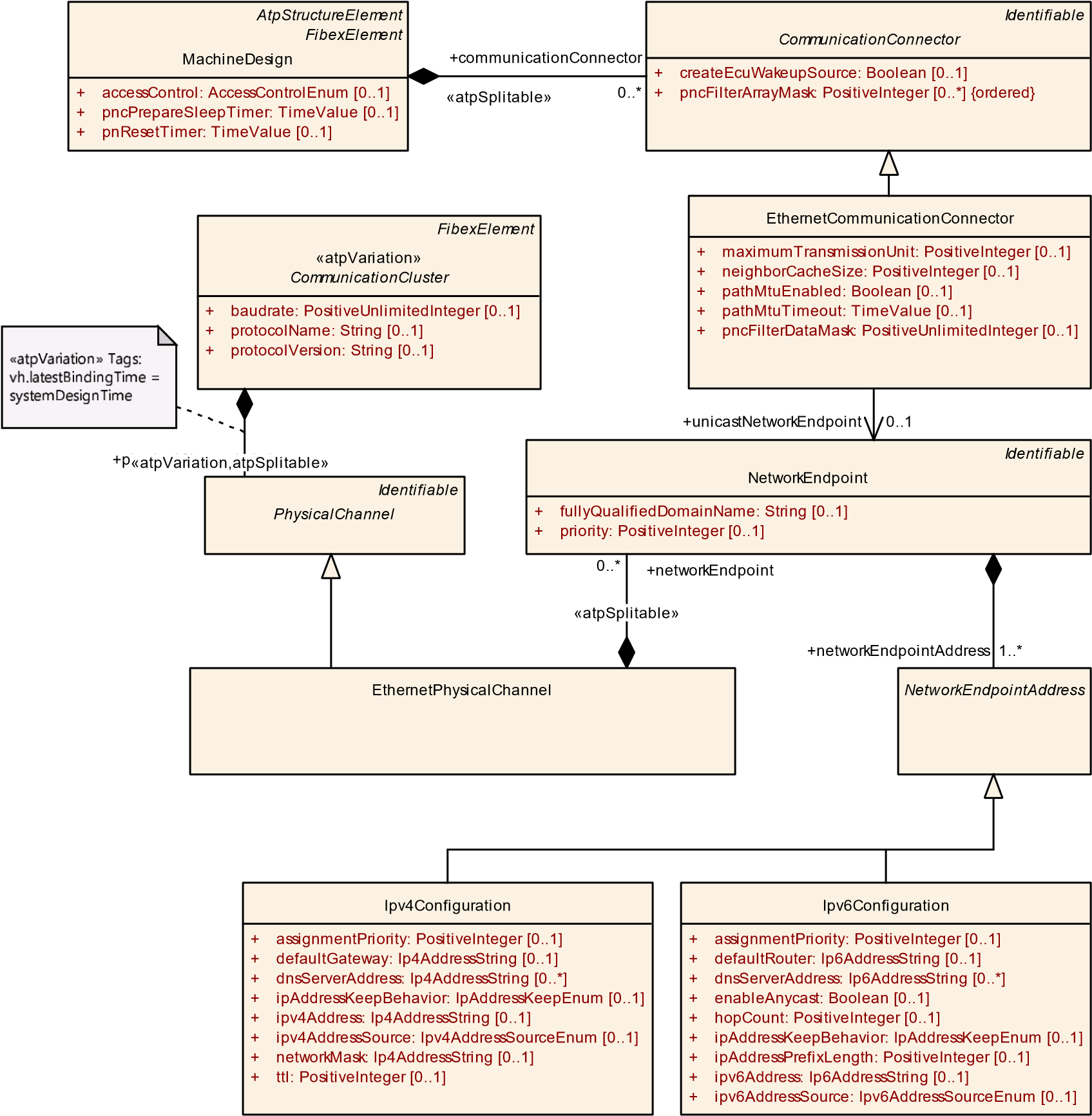
**[constr\_3320]** { DRAFT } **MachineDesign** d元类MachineDesign**聚合CommunicationConnector应**仅聚合角色communicationConnector中的EthernetCommunicationConnector 。 CommunicationConnector的其他子类不得出现在此聚合中。 c *()*

指定 IP 地址的规范方法是 NetworkEndpoint 的建模，从由MachineDesign在角色communicationConnector中聚合的EthernetCommunicationConnector引用。

除了 IP 地址， NetworkEndpoint可能有一个*完全限定的域名*和一个优先级。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **网络端点** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | 网络端点定义网络地址（例如 IP 地址或 MAC 多播地址）。 | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 完全合格  域名 | 细绳 | 0..1 | 属性 | 定义完全限定域名 (FQDN)，例如  一些.example.host。 |
| ipSecConfig | IPSecConfig | 0..1 | 聚合 | 为 IP 数据包提供安全服务的可选 IPSec 配置。 |
| 网络端点地址 | 网络端点  地址 | 1..\* | 聚合 | 网络地址的定义。  **标签：** xml.name  复数=网络-端点-地址 |
| 优先 | 正整数 | 0..1 | 属性 | 定义允许从 0（尽力而为）到 7（最高）的值的帧优先级。 |

**表 5.7：网络端点**



**图 5.2：MachineDesign 的网络连接**

更准确地说，特定的 IP 地址是通过聚合配置的

networkEndpointAddress中的Ipv4Configuration或Ipv6Configuration 。

NetworkEndpoint由 EthernetPhysicalChannel 聚合，而EthernetPhysicalChannel又由EthernetCluster聚合。

**[TPS\_MANI\_03052]** { DRAFT }**静态 IPv4 配置**d如果元类Ipv4Configuration的属性ipv4AddressSource的值设置为Ipv4AddressSourceEnum 。固定然后ipv4Address定义静态 IPv4

地址。 c *( RS\_MANI\_00018 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **IPv4配置** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | Internet 协议版本 4 (IPv4) 配置。 | | | |
| ***根据*** | *ARObject* ,*网络端点地址* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 任务  优先 | 正整数 | 0..1 | 属性 | 分配的优先级（1 最高）。如果具有较高优先级的分配方法的新地址可用，它将覆盖先前由具有较低优先级的分配方法分配的 IP 地址。 |
| 默认网关 | Ip4地址字符串 | 0..1 | 属性 | 默认网关的 IP 地址。 |
| dns服务器地址 | Ip4地址字符串 | \* | 属性 | 预配置 DNS 服务器的 IP 地址。  **标签：** xml.namePlural=DNS-SERVER-ADDRESSES |
| ipAddressKeep 行为 | IpAddressKeepEnum | 0..1 | 属性 | 定义动态获取的 IP 地址的生命周期。 |
| ipv4地址 | Ip4地址字符串 | 0..1 | 属性 | IPv4 地址。符号：255.255.255.255。如果 ipv4AddressSource 是 FIXED 并且因此不使用自动配置机制，则应声明 IP 地址。 |
| ipv4地址源 | Ipv4AddressSource  枚举 | 0..1 | 属性 | 定义节点如何获取其 IP 地址。 |
| 网络掩码 | Ip4地址字符串 | 0..1 | 属性 | 网络掩码。符号 255.255.255.255 |
| ttl | 正整数 | 0..1 | 属性 | 数据的生命周期 (0..255)。 TimeToLive 字段的目的是避免无法传递的数据报在系统中不断循环的情况。 |

**表 5.8：IPv4 配置**

|  |  |
| --- | --- |
| ***枚举*** | **Ipv4AddressSourceEnum** |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology |
| ***笔记*** | 定义节点如何获取其 IPv4 地址。 |
| ***文字*** | ***描述*** |
| 自动IP | AutoIP 用于在设备启动时动态分配 IP 地址。  **标签：** atp.EnumerationLiteralIndex=0 |
| autoIp\_doip | 使用 DoIP 参数的 Linklocal IPv4 地址分配  **标签：** atp.EnumerationLiteralIndex=2 |
| dhcpv4 | DHCP 是一种用于客户端自动 IP 配置的服务。  **标签：** atp.EnumerationLiteralIndex=3 |
| 固定的 | IP 地址应手动声明。  **标签：** atp.EnumerationLiteralIndex=4 |

**表 5.9：Ipv4AddressSourceEnum**

**[TPS\_MANI\_03053]** { DRAFT }**静态 IPv6 配置**d如果元类Ipv6Configuration的属性ipv6AddressSource的值设置为Ipv6AddressSourceEnum 。固定然后ipv6Address定义静态 IPv6

地址。 c *( RS\_MANI\_00018 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **IPv6配置** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | Internet 协议版本 6 (IPv6) 配置。 | | | |
| ***根据*** | *ARObject* ,*网络端点地址* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 任务  优先 | 正整数 | 0..1 | 属性 | 分配的优先级（1 最高）。如果具有较高优先级的分配方法的新地址可用，它将覆盖先前由具有较低优先级的分配方法分配的 IP 地址。 |
| 默认路由器 | Ip6地址字符串 | 0..1 | 属性 | 默认路由器的 IP 地址。 |
| dns服务器地址 | Ip6地址字符串 | \* | 属性 | 预配置 DNS 服务器的 IP 地址。  **标签：** xml.namePlural=DNS-SERVER-ADDRESSES |
| 启用任播 | 布尔值 | 0..1 | 属性 | 该属性用于启用任播寻址（即到多个接收器之一）。 |
| 跳数 | 正整数 | 0..1 | 属性 | 两台主机之间的距离。跳数 n 表示有 n 个网关将源主机与目标主机分开（范围 0..255） |
| ipAddressKeep 行为 | IpAddressKeepEnum | 0..1 | 属性 | 定义动态获取的 IP 地址的生命周期。 |
| ipAddressPrefix 长度 | 正整数 | 0..1 | 属性 | IPv6 前缀长度定义作为网络前缀的 IPv6 地址部分。 |
| ipv6地址 | Ip6地址字符串 | 0..1 | 属性 | IPv6 地址。符号：FFFF:...:FFFF。如果 ipv6AddressSource 是 FIXED 并且因此不使用自动配置机制，则应声明 IP 地址。 |
| ipv6地址源 | IPv6地址源  枚举 | 0..1 | 属性 | 定义节点如何获取其 IP 地址。 |

**表 5.10：IPv6 配置**

|  |  |
| --- | --- |
| ***枚举*** | **Ipv6AddressSourceEnum** |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology |
| ***笔记*** | 定义节点如何获取其 IPv6 地址。 |
| ***文字*** | ***描述*** |
| dhcpv6 | DHCP 是一种用于客户端自动 IP 配置的服务。  **标签：** atp.EnumerationLiteralIndex=0 |
| 固定的 | IP 地址应手动声明。  **标签：** atp.EnumerationLiteralIndex=1 |
| 链接本地 | LinkLocal 仅用于本地网络（链接）的网段或主机连接的点对点连接内的通信。  **标签：** atp.EnumerationLiteralIndex=2 |
| 链接Local\_doip | 使用 DoIP 参数的链路本地 IPv6 地址分配  **标签：** atp.EnumerationLiteralIndex=3 |
| 路由器  广告 | IPv6 无状态自动配置。  **标签：** atp.EnumerationLiteralIndex=4 |

**表 5.11：Ipv6AddressSourceEnum**

|  |  |
| --- | --- |
| ***枚举*** | **IpAddressKeepEnum** |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology |
| ***笔记*** | 定义分配动态 IP 地址后的行为。 |
| ***文字*** | ***描述*** |
| 忘记 | 分配动态 IP 地址后，只需将其用于此会话。  **标签：** atp.EnumerationLiteralIndex=0 |
| 持久存储 | 分配动态 IP 地址后，永久存储该地址。  **标签：** atp.EnumerationLiteralIndex=1 |

**表 5.12: IpAddressKeepEnum**

请注意，除了Ipv4Configuration和Ipv6Configuration在系统拓扑描述中，还有可能将MacMulticastConfiguration描述为NetworkEndpointAddress 。这对于描述不基于 IP 的端点可能很有用，例如，对于像 AVB（音频视频桥接）这样的流协议。但请注意，Adaptive Autosar 中没有基础软件或 ara::com 支持此类MacMulticastConfigurationNetworkEndpoint 。对于 SOME/IP 通信，[ constr\_3288 ]排除了此类NetworkEndpoint 。

**5.2.1.1 支持 10BASE-T1S 网络拓扑**

请注意系统支持 10BASE-T1S 网络拓扑描述

CouplingPortConnection进行设计，该连接将nodePort引用指向CouplingPort ，代表连接到网络的 10Base-T1S PHY。

有关 10BASE-T1S 网络建模的更多详细信息，请参见系统模板 [ 17 ]。由于在经典平台和自适应平台中使用了相同的建模方法，因此在本规范中不再重复对系统模板中可用的元模型的详细描述。

**5.2.2 TcpIp 栈配置属性**

MachineDesign引用以下元素并允许在系统设计中设置机器特定的 TcpIp 堆栈配置选项：

* EthIpProps - 用于配置 IPv4 和 IPv6
* EthTcpIpProps - 用于配置 TCP 和 UDP
* EthTcpIpIcmpProps - 用于配置 ICMP

请注意，系统模板 [ 17 ] 定义了EthIpProps 、 EthTcpIpProps 、 EthTcpIpIcmpProps的使用限制。如果MachineDesign引用这些元素，这些约束也是有效的。

**5.2.2.1 IP 配置属性**

Ipv4Props

Ipv6Props

*ARElement*

EthIpProps

Ipv4ArpProps

[0..1]

tcpIpArpNumGratuitousArpOnStartup: PositiveInteger

+

tcpIpArpPacketQueueEnabled: Boolean

[0..1]

+

tcpIpArpRequestTimeout: TimeValue

+

[0..1]

tcpIpArpTableEntryTimeout: TimeValue

[0..1]

+

Ipv4AutoIpProps

tcpIpAutoIpInitTimeout: TimeValue

+

[0..1]

Ipv4FragmentationProps

[0..1]

tcpIpIpFragmentationRxEnabled: Boolean

+

tcpIpIpNumFragments: PositiveInteger

[0..1]

+

tcpIpIpNumReassDgrams: PositiveInteger

+

[0..1]

+

tcpIpIpReassTimeout: TimeValue

[0..1]

Dhcpv6Props

+

tcpIpDhcpV6CnfDelayMax: TimeValue

[0..1]

+

tcpIpDhcpV6CnfDelayMin: TimeValue

[0..1]

[0..1]

tcpIpDhcpV6InfDelayMax: TimeValue

+

+

tcpIpDhcpV6InfDelayMin: TimeValue

[0..1]

tcpIpDhcpV6SolDelayMax: TimeValue

[0..1]

+

+

[0..1]

tcpIpDhcpV6SolDelayMin: TimeValue

Ipv6FragmentationProps

tcpIpIpReassemblyBufferCount: PositiveInteger

+

[0..1]

tcpIpIpReassemblyBufferSize: PositiveInteger

[0..1]

+

[0..1]

+

tcpIpIpReassemblySegmentCount: PositiveInteger

+

tcpIpIpReassemblyTimeout: TimeValue

[0..1]

+

tcpIpIpTxFragmentBufferCount: PositiveInteger

[0..1]

tcpIpIpTxFragmentBufferSize: PositiveInteger

+

[0..1]

Ipv6NdpProps

tcpIpNdpDefaultReachableTime: TimeValue

[0..1]

+

tcpIpNdpDefaultRetransTimer: TimeValue

+

+

[0..1]

tcpIpNdpDefaultRouterListSize: PositiveInteger

[0..1]

tcpIpNdpDefensiveProcessing: Boolean

+

tcpIpNdpDelayFirstProbeTime: PositiveInteger

[0..1]

+

[0..1]

tcpIpNdpDestinationCacheSize: PositiveInteger

+

tcpIpNdpDynamicHopLimitEnabled: Boolean

[0..1]

+

[0..1]

tcpIpNdpDynamicMtuEnabled: Boolean

+

tcpIpNdpDynamicReachableTimeEnabled: Boolean

+

[0..1]

[0..1]

+

tcpIpNdpDynamicRetransTimeEnabled: Boolean

+

tcpIpNdpMaxRandomFactor: PositiveInteger

[0..1]

tcpIpNdpMaxRtrSolicitationDelay: TimeValue

+

[0..1]

+

tcpIpNdpMaxRtrSolicitations: PositiveInteger

[0..1]

+

tcpIpNdpMinRandomFactor: PositiveInteger

[0..1]

+

tcpIpNdpNeighborUnreachabilityDetectionEnabled: Boolean

[0..1]

tcpIpNdpNumMulticastSolicitations: PositiveInteger

[0..1]

+

tcpIpNdpNumUnicastSolicitations: PositiveInteger

[0..1]

+

+

tcpIpNdpPacketQueueEnabled: Boolean

[0..1]

+

tcpIpNdpPrefixListSize: PositiveInteger

[0..1]

+

tcpIpNdpRandomReachableTimeEnabled: Boolean

[0..1]

[0..1]

tcpIpNdpRndRtrSolicitationDelayEnabled: Boolean

+

[0..1]

tcpIpNdpRtrSolicitationInterval: TimeValue

+

+

[0..1]

tcpIpNdpSlaacDadNumberOfTransmissions: PositiveInteger

tcpIpNdpSlaacDadRetransmissionDelay: TimeValue

[0..1]

+

[0..1]

tcpIpNdpSlaacDelayEnabled: Boolean

+

[0..1]

tcpIpNdpSlaacOptimisticDadEnabled: Boolean

+

*AtpStructureElement*

*FibexElement*

MachineDesign

+

accessControl: AccessControlEnum

[0..1]

[0..1]

+

pncPrepareSleepTimer: TimeValue

pnResetTimer: TimeValue

[0..1]

+

+

fragmentationProps

0..1

ipv4Props

+

0..1

ethIpProps

+

0..1

+

dhcpProps

0..1

+

fragmentationProps

0..1

+

ipv6Props

0..1

+

ndpProps

0..1

+

arpProps

0..1

autoIpProps

+

0..1

**图 5.3：机器特定的 IP 配置选项**

|  |  |
| --- | --- |
| ***班级*** | **EthIpProps** |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **EthIpProps** | | | |
| ***笔记*** | 此元类用于配置机器特定的 IP 属性。  **标签：** atp.recommendedPackage=EthIpProps | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| ipv4Props | IPv4Props | 0..1 | 聚合 | IPv4 的配置选项。 |
| ipv6Props | IPv6Props | 0..1 | 聚合 | IPv6 的配置选项。 |

**表 5.13：EthIpProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **IPv4Props** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | 此元类指定 IPv4 的配置选项。 | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| arp道具 | Ipv4ArpProps | 0..1 | 聚合 | ARP（地址解析协议）的配置属性。 |
| autoIpProps | Ipv4AutoIpProps | 0..1 | 聚合 | Auto-IP（自动专用 IP 寻址）的配置选项。 |
| 分片道具 | IPv4分片  道具 | 0..1 | 聚合 | IPv4 数据包分段/重组的配置选项。 |

**表 5.14：Ipv4Props**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **Ipv4ArpProps** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | 指定 ARP（地址解析协议）的配置选项。 | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| tcpIpArpNum  GratuitousArp OnStartup | 正整数 | 0..1 | 属性 | 该属性指定在分配新 IP 地址时应发送的无偿 ARP 回复的数量。 |
| tcpIpArpPacket QueueEnabled | 布尔值 | 0..1 | 属性 | 根据 IETF RFC 1122 第 2.3.2.2 节，此属性启用 (TRUE) 或禁用 (FALSE) 对 ARP 数据包队列的支持。 |
| tcpIpArp 请求  暂停 | 时间价值 | 0..1 | 属性 | 此属性指定 ARP 请求有效性的超时时间（以秒为单位）。在发送 ARP 请求后，TcpIp 应在 tcpIpArpRequestTimeout 秒的持续时间内跳过向同一目的地发送任何进一步的 ARP 请求。 （IETF RFC 1122，第 2.3.2.1 节）。 |
| tcpIpArpTable 入口超时 | 时间价值 | 0..1 | 属性 | 此属性指定删除未使用的 ARP 条目后的超时时间（以秒为单位）。 |

**表 5.15：Ipv4ArpProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **Ipv4AutoIpProps** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | 指定 Auto-IP（自动专用 IP 寻址）的配置选项。 | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| tcpIpAutoIpInit 超时 | 时间价值 | 0..1 | 属性 | 此属性以秒为单位指定 Auto-IP 在启动时等待开始 ARP 探测之前的时间。此延迟用于让 DHCP 有时间在 DHCP 服务器存在的情况下获取租约。 |

**表 5.16：Ipv4AutoIpProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **Ipv4FragmentationProps** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | 指定 IPv4 数据包分段/重组的配置选项。 | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| tcpIpIp  碎片化  启用 Rx | 布尔值 | 0..1 | 属性 | 启用 (TRUE) 或禁用 (FALSE) 支持重新组装根据 IETF RFC 815（IP 数据报重组）分段的传入数据报  算法）。 |
| tcpIpIpNum 片段 | 正整数 | 0..1 | 属性 | 指定每个数据报的最大 IP 片段数。 |
| tcpIpIpNum  重估图 | 正整数 | 0..1 | 属性 | 指定可以并行重组的分段 IP 数据报的最大数量。 |
| tcpIpIpReass 超时 | 时间价值 | 0..1 | 属性 | 以 [s] 为单位指定超时，在此之后不完整的数据报将被丢弃。 |

**表 5.17：Ipv4FragmentationProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **IPv6Props** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | 此元类指定 IPv6 的配置选项。 | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| dhcpProps | dhcpv6Props | 0..1 | 聚合 | DHCPv6 的配置属性。 |
| 分片道具 | IPv6分片  道具 | 0..1 | 聚合 | IPv6 数据包分段/重组的配置属性。 |
| ndpProps | Ipv6NdpProps | 0..1 | 聚合 | IPv6 的邻居发现协议的配置属性。 |

**表 5.18：Ipv6Props**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **Ipv6FragmentationProps** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | 此元类指定 IPv6 数据包分段/重组的配置选项。 | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **Ipv6FragmentationProps** | | | |
| tcpIpIp  重新组装  缓冲区计数 | 正整数 | 0..1 | 属性 | 可用于片段重组的缓冲区数。如果发生重组错误或者如果没有及时接收到所有片段，则此缓冲区将被阻塞，直到超过指定的“片段重组超时”。  值 0 禁用片段重组。 |
| tcpIpIp  重新组装  缓冲区大小 | 正整数 | 0..1 | 属性 | 每个片段 tx 缓冲区的大小（以字节为单位）。 |
| tcpIpIp  重新组装  段数 | 正整数 | 0..1 | 属性 | 指定可以在每个重组缓冲区中管理的最大连续数据段数。如果按顺序接收所有片段，则只需要一个片段。  为了处理乱序接收的片段，该值应配置为大于 1。 |
| tcpIpIp 重组  暂停 | 时间价值 | 0..1 | 属性 | 指定丢弃不完整数据报之前的超时时间（以秒为单位）。 |
| tcpIpIpTx  片段缓冲区  数数 | 正整数 | 0..1 | 属性 | 如果 IpV6 从上层接收到不适合 MTU 并因此必须分段的数据包，则将使用这些缓冲区。  值 0 禁用 tx 碎片。 |
| tcpIpIpTx  片段缓冲区  尺寸 | 正整数 | 0..1 | 属性 | 每个片段 tx 缓冲区的大小（以字节为单位）。 |

**表 5.19：Ipv6FragmentationProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **dhcpv6Props** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | 此元类指定 DHCPv6 的配置选项。 | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| tcpIpDhcp  V6CnfDelayMax | 时间价值 | 0..1 | 属性 | 发送第一条确认消息之前的最大延迟（以秒为单位）。如果此值大于先前的最小延迟值，则将从间隔中选择随机延迟。 |
| tcpIpDhcp  V6CnfDelayMin | 时间价值 | 0..1 | 属性 | 发送第一条确认消息之前的最小延迟（以秒为单位）。 |
| tcpIpDhcpV6Inf 延迟最大值 | 时间价值 | 0..1 | 属性 | 发送第一个信息请求消息之前的最大延迟（以秒为单位）。如果此值大于先前的最小延迟值，则将从间隔中选择随机延迟。 |
| tcpIpDhcpV6Inf 延迟最小值 | 时间价值 | 0..1 | 属性 | 发送第一个信息请求消息之前的最小延迟（秒）。 |
| tcpIpDhcpV6Sol DelayMax | 时间价值 | 0..1 | 属性 | 发送第一条 Solicit 消息之前的最大延迟（以秒为单位）。如果此值大于先前的最小延迟值，则将从间隔中选择随机延迟。 |
| tcpIpDhcpV6Sol DelayMin | 时间价值 | 0..1 | 属性 | 发送第一条 Solicit 消息之前的最小延迟（秒）。 |

**表 5.20：Dhcpv6Props**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **Ipv6NdpProps** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | 此元类指定 IPv6 的邻居发现协议的配置选项。 | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| tcpIpNdpDefault ReachableTime | 时间价值 | 0..1 | 属性 | [RFC4861 6.3.2 中指定的 ReachableTime (s) 的配置。主变量]。 |
| tcpIpNdpDefault RetransTimer | 时间价值 | 1 | 属性 | 为 [RFC4861 6.3.2. 中指定的 RetransTimer 变量配置默认值 (s)。主变量]。 |
| tcpIpNdpDefault RouterListSize | 正整数 | 0..1 | 属性 | 默认路由器条目的最大数量。 |
| tcpIpNdp 防御型  加工 | 布尔值 | 0..1 | 属性 | 如果启用，NDP 将仅处理响应先前发送的邻居请求而接收的邻居广告，并根据收到的邻居请求跳过对邻居缓存的更新。如果禁用所有邻居广告和请求，则应按照 RFC4861 中的规定进行处理。 |
| tcpIpNdpDelay FirstProbeTime | 正整数 | 0..1 | 属性 | 在 (s) 中发送第一个 NUD 探测之前的延迟。 |
| tcpIpNdp 目标  缓存大小 | 正整数 | 0..1 | 属性 | 目标缓存中的最大条目数。 |
| tcpIpNdp  动态跳跃  限制启用 | 布尔值 | 0..1 | 属性 | 如果启用，则可以根据收到的路由器通告重新配置默认跳数限制。 |
| tcpIpNdp  动态Mtu  启用 | 布尔值 | 0..1 | 属性 | 允许通过路由器通告动态重新配置链路 MTU。 |
| tcpIpNdp  动态的  可达时间  启用 | 布尔值 | 0..1 | 属性 | 如果启用，则可以根据收到的路由器通告重新配置默认的可达时间值。 |
| tcpIpNdp 动态  重传时间  启用 | 布尔值 | 0..1 | 属性 | 如果启用，则可以根据收到的路由器广告重新配置默认的重传计时器值。 |
| tcpIpNdpMax  随机因子 | 正整数 | 0..1 | 属性 | 用于随机化的最大随机因子 |
| tcpIpNdpMaxRtr  征集  延迟 | 时间价值 | 0..1 | 属性 | 第一个Router Solicitation 之前的最大延迟将在(s) 中接口初始化之后发送。 |
| tcpIpNdpMaxRtr  征集 | 正整数 | 0..1 | 属性 | 在收到第一个路由器广告之前将发送的最大路由器请求数。 |
| tcpIpNdpMin  随机因子 | 正整数 | 0..1 | 属性 | 用于随机化的最小随机因子 |
| tcpIpNdp  邻居  不可达性  启用检测 | 布尔值 | 0..1 | 属性 | 邻居不可达检测用于从邻居缓存中删除未使用的条目。此功能是 NDP 的基本功能，应开启。 |
| tcpIpNdpNum  组播  征集 | 正整数 | 0..1 | 属性 | 执行地址解析时将发送的最大多播请求数。 |
| tcpIpNdpNum  单播  征集 | 正整数 | 0..1 | 属性 | 执行邻居不可达检测时将发送的最大单播请求数。 |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **Ipv6NdpProps** | | | |
| tcpIpNdpPacket QueueEnabled | 布尔值 | 0..1 | 属性 | 启用 (TRUE) 或禁用 (FALSE) 对 NDP 的支持  符合 IETF RFC 4861 的数据包队列，章节  7.2.2. |
| tcpIpNdpPrefix ListSize | 正整数 | 0..1 | 属性 | 在线前缀列表中的最大条目数。 |
| tcpIpNdp  随机的  可达时间  启用 | 布尔值 | 0..1 | 属性 | 如果启用，ReachableTime 的值将与 MIN\_RANDOM\_FACTOR 和 MAX\_RANDOM\_FACTOR 之间的随机值相乘，以防止多个节点同时传输。 |
| tcpIpNdpRndRtr  征集  延迟启用 | 布尔值 | 0..1 | 属性 | 如果启用，第一个路由器请求将从 [0...MAX\_RTR\_SOLICITATION\_DELAY] 随机延迟。否则，第一个路由器请求将在 MAX\_RTR\_SOLICITATION\_DELAY 毫秒后发送。 |
| tcpIpNdpRtr  征集  间隔 | 时间价值 | 0..1 | 属性 | (s) 中连续路由器请求之间的间隔。 |
| tcpIpNdpSlaac DadNumberOf  传输 | 正整数 | 0..1 | 属性 | 为了将自动配置的地址设置为 PREFERRED（可用）状态而必须回答的邻居请求数。 |
| tcpIpNdpSlaac  爸爸  重传  延迟 | 时间价值 | 0..1 | 属性 | 设置地址配置延迟的最大值（秒）。 |
| tcpIpNdpSlaac 延迟启用 | 布尔值 | 0..1 | 属性 | 如果启用第一个 DAD 邻居请求的传输，将延迟 [0...MAX\_DAD\_DELAY] 中的随机值。 |
| tcpIpNdpSlaac  乐观的爸爸  启用 | 布尔值 | 0..1 | 属性 | 根据 RFC4429 启用乐观重复地址检测 (DAD)。 |

**表 5.21：Ipv6NdpProps**

**5.2.2.2 TCP 和 UDP 配置属性**

*ARElement*

EthTcpIpProps

TcpProps

[0..1]

tcpCongestionAvoidanceEnabled: Boolean

+

[0..1]

tcpDelayedAckTimeout: TimeValue

+

[0..1]

tcpFastRecoveryEnabled: Boolean

+

[0..1]

tcpFastRetransmitEnabled: Boolean

+

tcpFinWait2Timeout: TimeValue

+

[0..1]

[0..1]

tcpKeepAliveEnabled: Boolean

+

tcpKeepAliveInterval: TimeValue

+

[0..1]

+

tcpKeepAliveProbesMax: PositiveInteger

[0..1]

+

tcpKeepAliveTime: TimeValue

[0..1]

+

tcpMaxRtx: PositiveInteger

[0..1]

+

tcpMsl: TimeValue

[0..1]

+

tcpNagleEnabled: Boolean

[0..1]

+

[0..1]

tcpReceiveWindowMax: PositiveInteger

[0..1]

tcpRetransmissionTimeout: TimeValue

+

+

[0..1]

tcpSlowStartEnabled: Boolean

tcpSynMaxRtx: PositiveInteger

+

[0..1]

tcpSynReceivedTimeout: TimeValue

+

[0..1]

[0..1]

+

tcpTtl: PositiveInteger

UdpProps

+

udpTtl: PositiveInteger

[0..1]

TcpIpIcmpv6Props

[0..1]

tcpIpIcmpV6EchoReplyAvoidFragmentation: Boolean

+

[0..1]

+

tcpIpIcmpV6EchoReplyEnabled: Boolean

+

tcpIpIcmpV6HopLimit: PositiveInteger

[0..1]

+

tcpIpIcmpV6MsgDestinationUnreachableEnabled: Boolean

[0..1]

[0..1]

+

tcpIpIcmpV6MsgParameterProblemEnabled: Boolean

TcpIpIcmpv4Props

+

tcpIpIcmpV4EchoReplyEnabled: Boolean

[0..1]

tcpIpIcmpV4Ttl: PositiveInteger

[0..1]

+

*ARElement*

EthTcpIpIcmpProps

*AtpStructureElement*

*FibexElement*

MachineDesign

+

accessControl: AccessControlEnum

[0..1]

+

pncPrepareSleepTimer: TimeValue

[0..1]

pnResetTimer: TimeValue

[0..1]

+

+

tcpIpProps

0..1

udpProps

+

0..1

tcpProps

+

0..1

tcpIpIcmpProps

+

0..1

+

icmpV6Props

0..1

+

icmpV4Props

0..1

**图 5.4：机器特定的 TCP/UDP 和 ICMP 配置选项**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **EthTcpIpProps** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | 此元类用于配置机器特定的 TcpIp 堆栈属性。  **标签：** atp.recommendedPackage=EthTcpIpProps | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| tcpProps | TcpProps | 0..1 | 聚合 | TCP 配置属性 |
| udpProps | udp道具 | 0..1 | 聚合 | UDP 配置属性 |

**表 5.22：EthTcpIpProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **udp道具** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | 此元类指定 UDP（用户数据报协议）的配置选项。 | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| udpTtl | 正整数 | 0..1 | 属性 | 传出 UDP 数据包的默认生存时间值。 |

**表 5.23：UdpProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **TcpProps** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | 此元类指定 TCP（传输控制协议）的配置选项。 | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| tcp拥塞  回避  启用 | 布尔值 | 0..1 | 属性 | 根据 IETF RFC 5681 启用 (TRUE) 或禁用 (FALSE) 支持 TCP 拥塞避免算法。 |
| tcpDelayedAck 超时 | 时间价值 | 0..1 | 属性 | 确认延迟传输的最大时间（以秒为单位）。 |
| tcp快速恢复  启用 | 布尔值 | 0..1 | 属性 | 根据 IETF RFC 5681 启用 (TRUE) 或禁用 (FALSE) 对 TCP 快速恢复的支持。 |
| tcpFast 重传  启用 | 布尔值 | 0..1 | 属性 | 根据 IETF RFC 5681 启用 (TRUE) 或禁用 (FALSE) 对 TCP 快速重传的支持。 |
| tcpFin  等待2超时 | 时间价值 | 0..1 | 属性 | 超时 [s] 从远程节点接收 FIN  （在该节点发起连接终止之后），即在 FINWAIT-2 中等待来自远程 TCP 的连接终止请求的最长时间。 |
| tcpKeepAlive 启用 | 布尔值 | 0..1 | 属性 | 根据 IETF RFC 1122 第 4.2.3.6 章启用 (TRUE) 或禁用 (FALSE) TCP Keep Alive Probes。 |
| tcpKeepAlive 间隔 | 时间价值 | 0..1 | 属性 | 指定后续 keepalive 探测之间的间隔（以秒为单位）。 |
| tcpKeepAlive ProbesMax | 正整数 | 0..1 | 属性 | 在连接关闭之前重新传输 TCP Keep Alive 的最大次数。 |
| tcpKeepAlive 时间 | 时间价值 | 0..1 | 属性 | 指定发送的最后一个数据包（简单的 ACK 不被视为数据）和第一个 keepalive 探测之间的时间（以 [s] 为单位）。 |
| tcpMaxRtx | 正整数 | 0..1 | 属性 | 在关闭 TCP 连接之前重新传输 TCP 段的最大次数。该参数仅在配置了 tcpRetransmissionTimeout 时有效。注意：此参数也适用于 FIN 重传。 |
| tcpMsl | 时间价值 | 0..1 | 属性 | 以 [s] 为单位的最大段生命周期。 |
| tcpNagle 已启用 | 布尔值 | 0..1 | 属性 | 根据 IETF RFC 1122（第 4.2.3.4 章何时发送数据）启用 (TRUE) 或禁用 (FALSE) 对 Nagle 算法的支持。如果启用，默认情况下会为所有 TCP 套接字激活 Nagle 算法，但可以禁用每个套接字（使用属性 TcpTp.nagle 算法）。 |
| tcpReceive  窗口最大值 | 正整数 | 0..1 | 属性 | 最大接收窗口的默认值（以字节为单位）。 |
| tcp  重传  暂停 | 时间价值 | 0..1 | 属性 | 在再次发送未确认的 TCP 段之前超时 [s]。如果禁用超时，则不应重传任何 TCP 段。 |
| tcpSlowStart 已启用 | 布尔值 | 0..1 | 属性 | 根据 IETF RFC 5681 启用 (TRUE) 或禁用 (FALSE) 对 TCP 慢启动算法的支持。 |
| tcpSynMaxRtx | 正整数 | 0..1 | 属性 | 重传 TCP SYN 的最大次数。 |
| tcpSynReceived 超时 | 时间价值 | 0..1 | 属性 | 完成远程发起的 TCP 连接建立的超时 [s]，即在接收和发送连接请求后，在 SYN-RECEIVED 中等待确认连接请求确认的最长时间。 |
| tcpTtl | 正整数 | 0..1 | 属性 | 传出 TCP 数据包的默认生存时间值。 |

**表 5.24：TcpProps**

**5.2.2.3 ICMP 配置属性**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **EthTcpIpIcmpProps** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | 此元类用于配置机器特定的 ICMP（Internet 控制消息协议）属性  **标签：** atp.recommendedPackage=EthTcpIcmpProps | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| icmpV4Props | TcpIpIcmpv4Props | 0..1 | 聚合 | ICMPv4 配置属性 |
| icmpV6道具 | TcpIpIcmpv6Props | 0..1 | 聚合 | ICMPv6 配置属性 |

**表 5.25：EthTcpIpIcmpProps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **TcpIpIcmpv4Props** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | 此元类指定 ICMPv4（Internet 控制消息协议）的配置选项。 | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| tcpIpIcmp V4EchoReply  启用 | 布尔值 | 0..1 | 属性 | 在收到 ICMP 回显的情况下，此属性启用或禁用 ICMP 回显应答消息的传输。 |
| tcpIpIcmpV4Ttl | 正整数 | 0..1 | 属性 | 该属性仅在使用 ICMP（Internet 控制消息协议）的情况下才相关。它指定传出 ICMP 数据包的默认生存时间值。 |

**表 5.26：TcpIpIcmpv4Props**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **TcpIpIcmpv6Props** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology | | | |
| ***笔记*** | 此元类指定 ICMPv6（Internet 控制消息协议）的配置选项。 | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| tcpIpIcmp  V6Echo回复  避免  碎片化 | 布尔值 | 0..1 | 属性 | 该属性定义是否仅在传入的 ICMPv6 回显请求 (Ping) 符合相应接口的 MTU 时才传输回显回复，即可以在没有 IPv6 分段的情况下传输。 |
| tcpIpIcmp V6EchoReply  启用 | 布尔值 | 0..1 | 属性 | 在收到 ICMP 回显的情况下，此属性启用或禁用 ICMP 回显应答消息的传输。 |
| tcpIpIcmp  V6HopLimit | 正整数 | 0..1 | 属性 | 传出 ICMPv6 数据包的默认 Hop-Limit 值。 |
| tcpIpIcmp  V6消息  目的地  无法到达  启用 | 布尔值 | 0..1 | 属性 | 此属性启用/禁用目标不可达消息的传输。 |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **TcpIpIcmpv6Props** | |  |  |
| tcpIpIcmp  V6消息  范围  问题  启用 | 布尔值 | 0..1 | 属性 | 如果启用，如果收到的数据包由于在数据包中发现未知选项或标头而被丢弃，则将发送 ICMPv6 参数问题消息。 |

**表 5.27：TcpIpIcmpv6Props**

**5.2.3 使用 IPsec 保护通信**

IPsec 是一个协议套件，它为 IPv4 和 IPv6 网络数据包中的 IP 数据报提供加密保护。

IPsec 使用安全关联来指定通信方之间共享的安全属性。安全关联定义了两方或多方之间的关系，并确定将使用哪些安全服务进行安全通信。换句话说，安全关联充当不同设备之间的“合同”。

单个安全关联在一个通信方向上保护数据。应存在两个安全关联以保护双向的流量。每个安全关联都可以提供加密、数据完整性和数据认证。

此外，IP 数据报的发送者和接收者可以根据 IPsec 安全策略确定 IP 数据包所需的保护。这些规则定义了如何处理设备接收的数据报。例如，安全策略用于决定特定数据包是否需要丢弃或需要由 IPsec 处理。

**[TPS\_MANI\_03203]** { DRAFT } **IPsec 的配置由**角色 ipSecConfig 中的NetworkEndpoint聚合的IPSecConfig元类提供了定义配置 IPsec 安全关联和 IPsec 安全策略所必需的 IPsec 设置的能力。 c *( RS\_MANI\_00036 )*

**[TPS\_MANI\_03204]** { DRAFT } **IPSecRule的定义**IPSecConfig元类可能包含一个或多个IPSecRule 。每个IPSecRule通过定义本地端点和远程端点来定义由 IPsec 监视的网络连接。每个端点由 IP 地址和 Tcp/Udp 端口定义。 IPSecRule有效的通信方向由方向属性定义。 c *( RS\_MANI\_00036 )*

*PhysicalChannel*

EthernetPhysicalChannel

*Identifiable*

NetworkEndpoint

+

fullyQualifiedDomainName: String

[0..1]

+

priority: PositiveInteger

[0..1]

*NetworkEndpointAddress*

Ipv6Configuration

+

[0..1]

assignmentPriority: PositiveInteger

+

defaultRouter: Ip6AddressString

[0..1]

+

[0..\*]

dnsServerAddress: Ip6AddressString

[0..1]

enableAnycast: Boolean

+

+

hopCount: PositiveInteger

[0..1]

+

ipAddressKeepBehavior: IpAddressKeepEnum

[0..1]

[0..1]

+

ipAddressPrefixLength: PositiveInteger

+

ipv6Address: Ip6AddressString

[0..1]

+

ipv6AddressSource: Ipv6AddressSourceEnum

[0..1]

Ipv4Configuration

assignmentPriority: PositiveInteger

+

[0..1]

[0..1]

defaultGateway: Ip4AddressString

+

dnsServerAddress: Ip4AddressString

[0..\*]

+

ipAddressKeepBehavior: IpAddressKeepEnum

+

[0..1]

ipv4Address: Ip4AddressString

[0..1]

+

[0..1]

ipv4AddressSource: Ipv4AddressSourceEnum

+

+

networkMask: Ip4AddressString

[0..1]

[0..1]

ttl: PositiveInteger

+

*ARElement*

IPSecConfigProps

+

ahCipherSuiteName: String

[0..\*]

[0..1]

+

dpdAction: IPsecDpdActionEnum

[0..1]

+

dpdDelay: TimeValue

[0..\*]

+

espCipherSuiteName: String

+

ikeCipherSuiteName: String

[0..1]

+

ikeOverTime: TimeValue

[0..1]

+

ikeRandTime: PositiveInteger

[0..1]

ikeReauthTime: TimeValue

[0..1]

+

+

ikeRekeyTime: TimeValue

[0..1]

+

saOverTime: PositiveInteger

[0..1]

+

saRandTime: TimeValue

[0..1]

+

saRekeyTime: TimeValue

[0..1]

*Identifiable*

IPSecRule

+

direction: CommunicationDirectionType

[0..1]

[0..1]

+

headerType: IPsecHeaderTypeEnum

ipProtocol: IPsecIpProtocolEnum

+

[0..1]

[0..1]

+

localId: String

+

localPortRangeEnd: PositiveInteger

[0..1]

+

localPortRangeStart: PositiveInteger

[0..1]

+

mode: IPsecModeEnum

[0..1]

+

policy: IPsecPolicyEnum

[0..1]

priority: PositiveInteger

[0..1]

+

+

remoteId: String

[0..1]

remotePortRangeEnd: PositiveInteger

+

[0..1]

[0..1]

remotePortRangeStart: PositiveInteger

+

IPSecConfig

*ARElement*

CryptoServiceCertificate

[0..1]

algorithmFamily: CryptoCertificateAlgorithmFamilyEnum

+

+

format: CryptoCertificateFormatEnum

[0..1]

[0..1]

+

maximumLength: PositiveInteger

[0..1]

serverNameIdentification: String

+

*ARElement*

CryptoServiceKey

algorithmFamily: String

+

[0..1]

keyGeneration: CryptoServiceKeyGenerationEnum

+

keyStorageType: String

[0..1]

+

length: PositiveInteger

+

networkEndpointAddress

+

1..\*

ipSecConfig

+

0..1

+

localCertificate

0..\*

0..1

+

nextHigherCertificate

ipSecRule

+

0..\*

preSharedKey

+

0..1

+

remoteCertificate

0..\*

+

remoteIpAddress

0..\*

+

networkEndpoint

0..\*

«atpSplitable»

ipSecConfigProps

+

0..1

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | «枚举»  IPsecIpProtocolEnum | | udp tcp 任何 icmp | | |  | | --- | | «枚举»  通信方向类型 | | 进出\_ | | |  | | --- | | «枚举... IPsecPolicyEnum | | ipsec 直通丢弃  拒绝 | | |  | | --- | | «枚举»  IPsecModeEnum | | 隧道运输 | |
| |  | | --- | | «枚举»  IPsecHeaderTypeEnum | | 啊特别是没有 | |
| |  | | --- | | «枚举»  IPsecDpdActionEnum | | 清除陷阱  重新开始 | |

**图 5.5：IPsec 配置模型**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **IPSecConfig** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::SecureCommunication | | | |
| ***笔记*** | IPsec 是一种协议，旨在为 IP 网络连接提供“端到端”的基于加密的安全性。 | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| ipSecConfig 道具 | IPSecConfigProps | 0..1 | 参考 | 对 NetworkEndpoint 上定义的所有 IPSecRules 有效的全局 IPsec 配置设置。 |
| ipSecRule | IPSecRule | \* | 聚合 | 在 IPSecConfig 中为特定 NetworkEndpoint 定义的 IPSec 规则和过滤器。 |

**表 5.28：IPSecConfig**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **IPSecRule** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::SecureCommunication | | | |
| ***笔记*** | 此元素定义了一个 IPsec 规则，该规则描述了受监控、保护和过滤的通信流量。 | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 方向 | 沟通  方向类型 | 0..1 | 属性 | 该属性定义了监控流量的方向。如果未设置此属性，则假定为双向流量监控。 |
| 标头类型 | IPsecHeaderTypeEnum | 0..1 | 属性 | 指定 IPsec 安全机制的标头类型。 |
| ip协议 | IPsecIpProtocolEnum | 0..1 | 属性 | 此属性定义安全策略数据库 (SPD) 条目中使用的相关 IP 协议。 |
| 本地证书 | 加密服务  证书 | \* | 参考 | 此参考标识用于本地身份验证的适用证书。 |
| 本地标识 | 细绳 | 0..1 | 属性 | 该属性定义本地参与者应如何被识别以进行身份验证。 |
| localPortRange 结束 | 正整数 | 0..1 | 属性 | 此属性限制流量监控并定义本地端口范围的结束值。  如果未设置该属性，则该规则对所有本地端口均有效。  请注意，AUTOSAR AP 的操作系统后端当前不支持端口范围。如果涉及 AP 系统，则每个 IPsec 规则可能只包含一个端口。 |
| localPortRange 开始 | 正整数 | 0..1 | 属性 | 此属性限制流量监控并定义本地端口范围的起始值。  如果未设置该属性，则该规则对所有本地端口均有效。  请注意，AUTOSAR AP 的操作系统后端当前不支持端口范围。如果涉及 AP 系统，则每个 IPsec 规则可能只包含一个端口。 |
| 模式 | IPsecModeEnum | 0..1 | 属性 | 此属性定义连接的类型。 |
| 政策 | IPsecPolicyEnum | 0..1 | 属性 | IPsec 策略定义了确定需要使用 IPsec 保护哪种类型的 IP 流量以及如何保护该流量的规则。 |
| 预共享密钥 | 加密服务密钥 | 0..1 | 参考 | 此参考标识用于身份验证的适用加密密钥。 |
| 优先 | 正整数 | 0..1 | 属性 | 此属性定义 IPSecRule（SPD 条目）的优先级。条目的处理基于优先级，从最高优先级“0”开始。 |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **IPSecRule** |  | | |
| 偏僻的  证书 | 加密服务  证书 | \* | 参考 | 此参考标识用于远程身份验证的适用证书。 |
| 远程ID | 细绳 | 0..1 | 属性 | 此属性定义应如何识别远程参与者以进行身份验证。 |
| 远程IP地址 | 网络端点 | \* | 参考 | 远程 NetworkEndpoint 的定义。通过此参考，描述了本地网络端点和远程网络端点之间的连接，在该连接上监控流量。 |
| 远程端口范围结束 | 正整数 | 0..1 | 属性 | 此属性限制流量监控并定义远程端口范围的结束值。  如果未设置该属性，则该规则对所有本地端口均有效。  请注意，AUTOSAR AP 的操作系统后端当前不支持端口范围。如果涉及 AP 系统，则每个 IPsec 规则可能只包含一个端口。 |
| 远程端口范围开始 | 正整数 | 0..1 | 属性 | 此属性限制流量监控并定义远程端口范围的起始值。  如果未设置该属性，则该规则对所有本地端口均有效。  请注意，AUTOSAR AP 的操作系统后端当前不支持端口范围。如果涉及 AP 系统，则每个 IPsec 规则可能只包含一个端口。 |

**表 5.29：IPSecRule**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **IPSecConfigProps** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::SecureCommunication | | | |
| ***笔记*** | 此元素包含独立于特定 IPsec 规则的 IPsec 配置的所有属性。  **标签：** atp.recommendedPackage=IPSecConfigProps | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| ahCipherSuite  姓名 | 细绳 | \* | 属性 | 用于连接的 AH（Authentication Header）算法，例如 HMAC/SHA2-256 |
| dpd动作 | IPsecDpdActionEnum | 0..1 | 属性 | 该属性定义了当对等体被认为死亡时要做什么。  如果未配置，则应假定“重新启动”。 |
| dpd延迟 | 时间价值 | 0..1 | 属性 | 该属性描述了使用 IKEv2 INFORMATIONAL 交换主动检查对等点活跃度的时间间隔。只有在配置的 DPD 延迟未收到任何 IKE 或 ESP/AH 数据包时，才会强制执行主动 DPD 检查。  在未配置的情况下，应假定值“5 分钟”。 |
| espCipherSuite  姓名 | 细绳 | \* | 属性 | ESP（封装安全有效负载）算法，为连接提供加密和可选身份验证，例如 AES-128+SHA2-256。 |
| ikeCipherSuite  姓名 | 细绳 | 0..1 | 属性 | 用于连接的 IKE 加密/身份验证算法。 |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **IPSecConfigProps** | | | |
| ikeOverTime | 时间价值 | 0..1 | 属性 | 此属性以百分比形式描述 SA 变为无效时的硬期限。  示例： ikeOverTime of max(ikeReauthTime, ikeRekey Time)。  默认值：10 % |
| ikeRandTime | 正整数 | 0..1 | 属性 | 此属性以百分比形式定义在 ikeReauthTime 和 ikeRekeyTime 到期前多久将被重新加密/重新认证。  默认值：10% |
| ikeReauthTime | 时间价值 | 0..1 | 属性 | 该属性定义了重新认证 IKE SA 的绝对时间。  0 表示禁用重新身份验证。 |
| ikeRekeyTime | 时间价值 | 0..1 | 属性 | 该属性定义了一个 IKE SA 将被重新加密的绝对时间。  0 表示禁用重新生成密钥。 |
| saOverTime | 正整数 | 0..1 | 属性 | 此属性以百分比形式描述 IPsec SA 变为无效时的硬期限。  示例：saOverTime \* saRekeyTime。  默认值：110% |
| 沙兰德时间 | 时间价值 | 0..1 | 属性 | 此属性定义在 saRekeyTime 到期前多久重新输入密钥。 |
| 密钥时间 | 时间价值 | 0..1 | 属性 | 此属性定义 IPsec SA 将被重新加密的绝对时间。  0 表示禁用重新生成密钥。 |

**表 5.30：IPSecConfigProps**

|  |  |
| --- | --- |
| ***枚举*** | **IPsecIpProtocolEnum** |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::SecureCommunication |
| ***笔记*** | IPSec 配置中的安全策略数据库 (SPD) 条目中支持的受支持 TcpIp 协议的定义。 |
| ***文字*** | ***描述*** |
| 任何 | 任何协议  **标签：** atp.EnumerationLiteralIndex=3 |
| icmp | 互联网控制消息协议 (ICMP)  **标签：** atp.EnumerationLiteralIndex=2 |
| tcp | TCP 协议  **标签：** atp.EnumerationLiteralIndex=1 |
| UDP | UDP协议  **标签：** atp.EnumerationLiteralIndex=0 |

**表 5.31：IPsecIpProtocolEnum**

**[constr\_5102]** { DRAFT } **IPSecRule中不允许使用远程端口范围**d IPSecRule 。 remotePortRangeStart和IPSecRule 。 remotePortRangeEnd应始终设置为相同的值。 c *()*

**[constr\_5103]** { DRAFT } **IPSecRule中不允许使用本地端口范围**d IPSecRule 。 localPortRangeStart和IPSecRule 。 localPortRangeEnd应始终设置为相同的值。 c *()*

[ constr\_5102 ] 和 [ constr\_5103 ] 的原因是 AUTOSAR Adaptive Platform 操作系统后端当前不支持端口范围，并且每个IPSecRule只允许定义一个本地端口和一个远程端口。

**[TPS\_MANI\_03232]** { DRAFT }**一般 IPsec 配置设置的定义**d独立于特定IPSecRule的一般配置属性收集在IPSecConfigProps元素中，该元素从角色 ipSecConfigProps 中的IPSecConfig引用。 c *( RS\_MANI\_00036 )*

**[TPS\_MANI\_03205]** { DRAFT } IPsec**策略**d IPSecRule 。策略属性定义如何处理通过IPSecRule定义的网络连接的 IP 数据包。具体来说，它定义了 IP 数据包是否由 IPsec 处理。 c *( RS\_MANI\_00036 )*

|  |  |
| --- | --- |
| ***枚举*** | **IPsecPolicyEnum** |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::SecureCommunication |
| ***笔记*** | 定义 IPsec 支持的过滤器操作。 |
| ***文字*** | ***描述*** |
| 降低 | 表示应该丢弃数据包  **标签：** atp.EnumerationLiteralIndex=3 |
| ipsec | 表示应保护数据包。  **标签：** atp.EnumerationLiteralIndex=1 |
| 直通 | 表示根本不应该进行任何 IPsec 处理。  **标签：** atp.EnumerationLiteralIndex=2 |
| 拒绝 | 表示应丢弃数据包并返回诊断 ICMP。  **标签：** atp.EnumerationLiteralIndex=4 |

**表 5.32：IPsecPolicyEnum**

IPsec 可以配置为在两种不同的模式下运行，即隧道模式和传输模式。使用隧道模式，整个 IP 数据包都受 IPsec 保护。 IPsec 包装原始数据包，对其进行加密并向其添加新的 IP 标头。

隧道模式最常用于 VPN 网关之间，新添加的外层 IP 头的 IP 地址是 VPN 网关的 IP 地址。换句话说，两个 VPN 网关之间的流量受到保护，每个网关充当其背后主机的代理。

传输模式提供对带有 AH 或 ESP 报头的 IP 数据报的数据有效负载的保护。 IP 标头保持不变，IPsec 将其标头插入 IP 标头和上层标头之间。

在保护两台主机之间或主机与 VPN 网关之间的流量时，可以使用 IPsec 传输模式。

**[TPS\_MANI\_03233]** { DRAFT } **IPsec**模式d IPSecRule 。 mode属性定义IP数据包是在传输模式还是隧道模式下处理。 c *( RS\_MANI\_00036 )*

请注意，AUTOSAR 目前仅支持传输模式作为配置选项。

|  |  |
| --- | --- |
| ***枚举*** | **IPsecModeEnum** |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::SecureCommunication |
| ***笔记*** | 此枚举描述了支持的 IPSec 模式。 |
| ***文字*** | ***描述*** |
| 运输 | 表示使用 IPSec 传输模式。在传输模式下，原始 IP 标头被保留，只有 IP 有效负载和 ESP 尾端被加密。  **标签：** atp.EnumerationLiteralIndex=1 |
| 隧道 | 表示使用 IPSec 隧道模式。在隧道模式下，整个原始 IP 数据包都受到 IPSec 的保护。这意味着 IPSec 包装原始数据包，对其进行加密，添加新的 IP 标头并将其发送到另一端。  **标签：** atp.EnumerationLiteralIndex=0 |

**表 5.33：IPsecModeEnum**

IPsec 使用两种协议：

* AH - 身份验证标头
* ESP - 封装安全负载

AH 协议提供了一种仅用于身份验证的机制，并对整个 IP 数据包进行身份验证，包括外部 IP 报头。

ESP 协议提供数据机密性（加密）和/或身份验证（数据完整性、数据源身份验证和重放保护）。

当 ESP 用于传输模式时，IP 有效负载被加密，原始 IP 标头被移动到消息的前面。 ESP 标头插入 IP 标头之后，并与 IP 有效负载一起签名。原始 IP 标头仍然不受保护。

在隧道模式下使用 ESP 时，会创建一个新的 IP 报头，并将 ESP 报头添加到原始 IP 数据包的前面。整个原始 IP 数据包在这种模式下被加密和签名。

**[TPS\_MANI\_03206]** { DRAFT } **IPsec AH 和 ESP 协议配置**d在IPSecRule中，可以定义 IPsec 协议，用于保护通过定义的网络连接的 IP 数据包。属性headerType定义是否使用 AH、ESP 或两者都不使用。 c *( RS\_MANI\_00036 )*

|  |  |
| --- | --- |
| ***枚举*** | **IPsecHeaderTypeEnum** |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::SecureCommunication |
| ***笔记*** | IPsec 标头类型选项 |
| ***文字*** | ***描述*** |
| 啊 | 身份验证标头 (AH)  **标签：** atp.EnumerationLiteralIndex=0 |

5

4

|  |  |
| --- | --- |
| ***枚举*** | **IPsecHeaderTypeEnum** |
| 特别是 | 封装安全负载 (ESP)  **标签：** atp.EnumerationLiteralIndex=1 |
| 没有任何 | 没有标题  **标签：** atp.EnumerationLiteralIndex=2 |

**表 5.34：IPsecHeaderTypeEnum**

**[TPS\_MANI\_03234]** { DRAFT } **IPsec AH 和 ESP CipherSuites** d属性ah-

CipherSuiteName和espCipherSuiteName定义了支持的 AH 和 ESP 算法。 c *( RS\_MANI\_00036 )*

ahCipherSuiteName 、 espCipherSuiteName和IPSecConfigProps的命名约定。 ikeCipherSuiteName应遵循 [ 13 ]中定义的密码原语的命名约定。

**[TPS\_MANI\_03207]** { DRAFT } **IPsec Internet 密钥交换协议配置**d在IPSecRule中，可以定义 IKE 协议如何验证远程方以及本地方如何向远程方验证自身。换句话说，双方使用相同的方法。 IPSecRule的使用。 preSharedKey参考定义使用预共享密钥。 IPSecRule的使用。当地的-

证书和IPSecRule 。 remoteCertificate定义使用数字签名认证。 c *( RS\_MANI\_00036 )*

请注意，支持的 IKE CipherSuites 配置了IPSec-

配置属性。 ikeCipherSuiteName 。 IPSecConfigProps包含附加的 IKE 特定配置设置。

|  |  |
| --- | --- |
| ***枚举*** | **IPsecDpdActionEnum** |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::SecureCommunication |
| ***笔记*** | 潜在的死亡对等检测 (Dpd) 操作 |
| ***文字*** | ***描述*** |
| 清除 | 删除 SA。  **标签：** atp.EnumerationLiteralIndex=0 |
| 重新开始 | 立即尝试建立连接。  **标签：** atp.EnumerationLiteralIndex=2 |
| 陷阱 | 尝试在流量发送到对等方后建立连接。  **标签：** atp.EnumerationLiteralIndex=1 |

**表 5.35：IPsecDpdActionEnum**

**[TPS\_MANI\_03208]** { DRAFT } **IPsec对AdaptivePlatformServiceInstance的保护**d为了描述IPsec 对AdaptivePlatformServiceInstance的保护，需要通过ServiceInstanceToMachineMapping将AdaptivePlatformServiceInstance映射到一个EthernetCommunicationConnector ，该 EthernetCommunicationConnector 使用unicastNetworkEndpoint指向一个NetworkEndpoint ，该 NetworkEndpoint聚合了IPSecConfig ，而 IPSecConfig 又描述了 IPsec

安全协会。 c *( RS\_MANI\_00036 )*

请注意，不支持 IPsec 的 IP 多播保护。由于 IP 多播地址在SomeipProvidedEventGroup中由两个属性ipv4MulticastIpAddress和ipv6MulticastIpAddress定义，因此不可能对 IP 多播通信的 IPsec 保护进行建模。 NetworkEndpoint元素仅用于描述 IP 单播端点。这意味着只有根据 [ TPS\_MANI\_03208 ] 描述的AdaptivePlatformServiceInstance的 IP 单播通信将受 IPsec 保护。

**5.2.4 服务发现配置**

服务发现消息在网络节点之间交换以宣布和发现可用的服务实例。本章描述了为支持的中间件传输层交换服务发现消息所需的配置。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***ServiceDiscoveryConfiguration*** （抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest | | | |
| ***笔记*** | 中间件传输层的服务发现配置设置。  **标签：** atp.Status=draft | | | |
| ***根据*** | *AR对象* | | | |
| ***子类*** | SomeipServiceDiscovery | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 5.36：服务发现配置**

**5.2.4.1 SOME/IP 服务发现配置**

**[TPS\_MANI\_03064]** { DRAFT } **SOME/IP Service Discovery 消息交换配置**d ProvidedServiceInstance由服务器在 SOME/IP 中通过VLAN上的多播寻址向特定 UDP 端口号的特定指定 IP 多播地址 ( SomeipServiceDiscovery.multicastSdIpAddress )宣布( SomeipServiceDiscovery . someipServiceDiscoveryPort )。 c *( RS\_MANI\_00019 )*

**[constr\_5045]** { DRAFT }**每个 VLAN 只允许一个SomeipServiceDiscovery配置**d只允许EthernetPhysicalChannel (VLAN)上的单个NetworkEndpoint被角色multicastSdIpAddress中的SomeipServiceDiscovery元素引用。 c *()*

SomeipServiceDiscovery能够引用SecureCompProps来定义和配置为服务发现消息提供通信安全的安全协议。

对于将传输到指定多播 IP 地址的服务发现消息，保护由角色multicastSecureComProps中引用的SecureComProps定义。对于单播服务发现消息，不同的凭证可用于不同的 ECU 对。

因此， SecureCompProps列表聚合在角色unicastSecureCompProps中。

*PhysicalChannel*

EthernetPhysicalChannel

*Identifiable*

NetworkEndpoint

fullyQualifiedDomainName: String

[0..1]

+

[0..1]

priority: PositiveInteger

+

SomeipServiceDiscovery

+

someipServiceDiscoveryPort: PositiveInteger

*ServiceDiscoveryConfiguration*

*ARElement*

*AtpStructureElement*

Machine

trustedPlatformExecutableLaunchBehavior

+

:

TrustedPlatformExecutableLaunchBehaviorEnum

*ARElement*

*SecureComProps*

*AtpStructureElement*

*FibexElement*

MachineDesign

accessControl: AccessControlEnum

+

[0..1]

+

[0..1]

pncPrepareSleepTimer: TimeValue

+

pnResetTimer: TimeValue

[0..1]

«atpSplitable»

+

serviceDiscoveryConfig

0..\*

+

networkEndpoint

0..\*

«atpSplitable»

+

multicastSecureComProps

0..1

machineDesign

+

1

+

multicastSdIpAddress

0..1

+

unicastSecureComProps

0..\*

**图 5.6：SOME/IP 服务发现配置**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **SomeipServiceDiscovery** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment | | | |
| ***笔记*** | 这个元类代表了针对 SOME/IP 案例的通用服务发现的专门化。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* ,*服务发现配置* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 多播SdIp地址 | 网络端点 | 0..1 | 参考 | 此参考标识用于服务发现的多播 IP 地址。  **标签：** atp.Status=draft |
| 多播安全 CompProps | SecureComProps | 0..1 | 参考 | 引用通信安全协议及其配置设置，这些设置将为使用多播传输的服务发现消息（例如 FindService 消息）提供通信安全。  **标签：** atp.Status=draft |
| someipService DiscoveryPort | 正整数 | 1 | 属性 | 该属性表示为服务发现保留的端口号。  **标签：** atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **SomeipServiceDiscover** | **是的** |  |  |
| 单播Secure CompProps | SecureComProps | \* | 参考 | 对通信安全协议及其配置设置的引用，该协议将为使用单播传输的服务发现消息提供通信安全，例如作为对 FindService 消息的应答的 OfferService。  .  **标签：** atp.Status=draft |

**表 5.37：SomeipServiceDiscovery**

**5.2.5 部分网络**

AUTOSAR 通过部分联网机制支持车辆运行期间的省电。该机制允许在正常总线通信期间关闭和启动 ECU 组（部分网络集群）的总线通信接口。

在 VFB 级别上，部分网络由虚拟功能集群表示，并使用PortGroup进行描述。虚拟功能集群对实现一个或多个车辆功能所需的通信进行分组，这些功能可以在正常车辆操作期间被激活/停用。虚拟功能集群映射到部分网络集群。

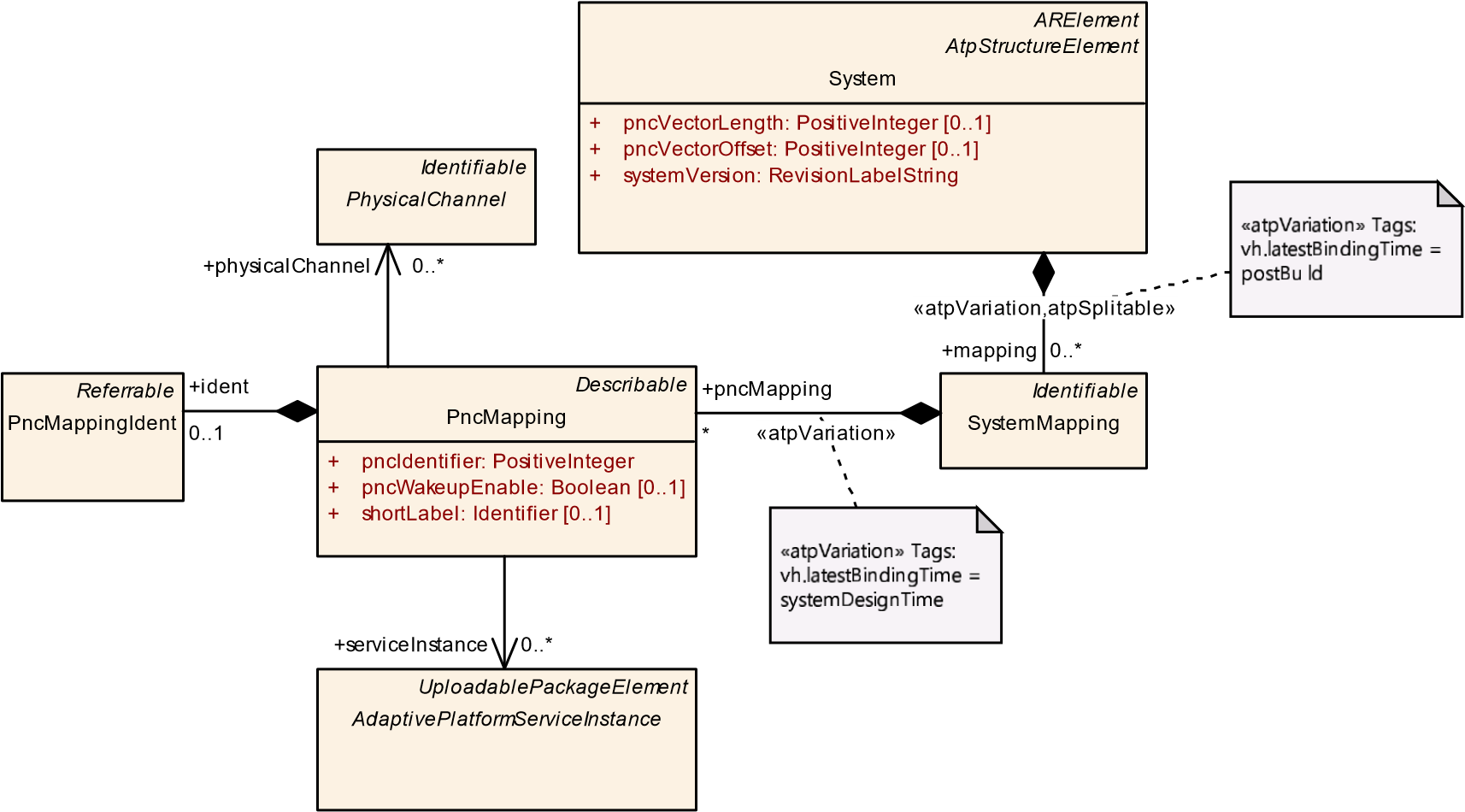
**[TPS\_MANI\_03224]** { DRAFT }**部分网络集群的建模**d部分网络集群使用PncMapping元素建模并由pncIdentifier标识。 PncMapping定义了使用PncMapping参与部分网络的AdaptivePlatformServiceInstance的集合。服务实例参考。 c *( RS\_MANI\_00062 )*

**[TPS\_MANI\_03225]** { DRAFT } PncMapping**中对 VLAN 的引用**d PncMapping与 VLAN 或未标记通道相关的方式有两种：

* 通过从PncMapping到角色physicalChannel中的PhysicalChannel的引用，
* 通过映射AdaptivePlatformServiceInstance的ServiceInstanceToMachineMapping并且还引用了一个EthernetCommunicationConnector ，后者又通过unicastNetworkEndpoint连接到EthernetPhysicalChannel 。

这两个选项不是相互排斥的，它们可以同时存在。 c *( RS\_-*

*MANI\_00062 )*



**图 5.7：PncMapping 与参与的 ServiceInstances 集合**

**部分网络集群**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **系统映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate | | | |
| ***笔记*** | 系统映射汇总了系统描述中相关的所有映射方面。 | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| pnc映射 | Pnc映射 | \* | 聚合 | 虚拟功能集群和部分网络集群之间的映射。  **刻板印象：** atpVariation  **标签：** vh.latestBindingTime=systemDesignTime |

**表 5.38：系统映射**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **Pnc映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate::PncMapping | | | |
| ***笔记*** | 描述一个或多个虚拟功能集群到部分网络集群之间的映射。虚拟功能集群由 PortGroup 实现。一个Partial Network Cluster由一个或多个ServiceInstances实现。 | | | |
| ***根据*** | *可描述*的*AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 识别 | PncMappingIdent | 0..1 | 聚合 | 这增加了可参考 PncMapping 的能力。 |
| 身体的  渠道 | 物理频道 | \* | 参考 | 该参考将部分网络映射到通信信道。 |
| pnc消费  假如  服务实例  团体 | 消费提供  服务实例组 | \* | 参考 | 在部分网络集群中使用的 ConsumedProvidedServiceInstanceGroup。此引用是可选的，因为它可用于根据请求的部分网络启动和停止 Consumed ProvidedServiceInstanceGroup，但不一定需要。  **刻板印象：** atpVariation  **标签：** vh.latestBindingTime=postBuild |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **Pnc映射** |  |  |  |
| pnc标识符 | 正整数 | 1 | 属性 | 部分网络集群的标识符。这个数字代表这个部分网络簇在 NM Pdu 中的绝对位位置。 |
| pncWakeup 启用 | 布尔值 | 0..1 | 属性 | 如果此参数可用并设置为 true，则一旦在分配此 PNC 的通道上发生通道唤醒，此 PNC 将被唤醒。这是通过在上行映射期间将此 PNC 添加到相应的通道唤醒源来确保的。 |
| 服务实例 | 自适应平台  服务实例 | \* | 参考 | 对参与部分网络集群的 ServiceInstance 的引用。  **标签：** atp.Status=draft |
| 短标签 | 标识符 | 0..1 | 属性 | 此属性指定 PncMapping 的标识短名称。它在系统范围内应该是唯一的。 |
| vfc | 端口组 | \* | 参考 | 要映射到部分网络集群的虚拟功能集群。如果系统描述不使用完整的软件组件描述（VFB 视图），则此参考是可选的。这支持包含遗留系统。  **InstanceRef 实现者：** PortGroupInSystem InstanceRef |

**表 5.39：PncMapping**

## 5.3 应用软件系统结构规范

系统设计模型的根元素是AUTOSAR 经典平台中已知的系统元素。 System聚合RootSwCompositionPrototype ，它表示给定系统中可用的所有软件组件的顶级组合。

**[TPS\_MANI\_03110]** { DRAFT }**系统描述中允许的组件，类别为 SYSTEM\_DESIGN\_DESCRIPTION 。** d SwComponentPrototype嵌套在由具有类别 SYSTEM\_DESIGN\_DESCRIPTION的系统的RootSwCompositionPrototype引用的CompositionSwComponentType中，允许属于 Classic 或 Adaptive AUTOSAR 支持的任何SwComponentType 。 c *( RS\_MANI\_00026 )*

|  |  |
| --- | --- |
| ***班级*** | **RootSwCompositionPrototype** |
| ***包裹*** | M2::AUTOSARTemplates::SystemTemplate |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **RootSwCompositionPrototype** | | | |
| ***笔记*** | RootSwCompositionPrototype 表示给定系统内软件组件的顶级组合。  根据系统的用例，这可能是或多或少完整的 VFB 描述、系统提取的软件或仅具有原子 SWC 的平面 ECU 提取的软件。  因此，RootSwComposition 只会偶尔包含在完整 VFB 系统中使用的所有原子软件组件。 OEM 主要对定义将软件组件集成到系统中所需的功能和接口感兴趣。这种组件的内部结构通常包含供应商的大量知识产权。因此，顶级软件组合通常包含代表子系统的空组合。  包含的 SwComponentPrototypes 完全由其 SwComponentTypes（包括 Port Prototypes、PortInterfaces、VariableDataPrototypes、SwcInternalBehavior 等）指定，并且它们的端口使用 SwConnectorPrototypes 互连。 | | | |
| ***根据*** | *ARObject* , *AtpFeature* , *AtpPrototype* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 软件  作品 | 组合开关  组件类型 | 1 | 特雷夫 | 我们假设只有一个顶级组合包含系统的所有 Component 实例。  **刻板印象：** isOfType |

**表 5.40：RootSwCompositionPrototype**

如果软件组件通过面向服务的通信进行通信并提供或需要ServiceInterface ，则相反的通信端并不总是预先知道。在具有SYSTEM\_DESIGN\_DESCRIPTION 类别的系统中，如果在系统设计时已经知道端点之间的面向服务的通信，则系统设计者可能希望指示它。

AA1

P

AA2

R

AA3

R

IF\_A

IF\_A

IF\_A

searchForSpecificInstance

searchForAllInstances

**图 5.8：系统设计模型中的装配连接器示例**

**[TPS\_MANI\_03114]** { DRAFT }**在系统设计模型中使用AssemblySwConnector** d在具有类别SYSTEM\_DESIGN\_DESCRIPTION 的系统中，如果所需的RPortPrototype正在搜索特定的服务实例，则允许通过AssemblySwConnector指示两个通信端点之间的面向服务的通信，即如果RPortPrototypeProps 。 searchIntention设置为searchForSpecificInstance 。

如果searchIntention设置为searchForAllInstances ，则不得使用AssemblySwConnector连接此RPortPrototype 。 c *( RS\_MANI\_00026 )*

**5.4 Classic之间面向服务通信的建模**

## 和自适应平台

AUTOSAR 经典平台尚不支持ServiceInterface ，但提供了通过 SOME/IP 以面向服务的方式进行通信的可能性。为了模仿经典平台中的ServiceInterface ，可以使用ClientServerInterface 、 SenderReceiverInterface或TriggerInterface的任意组合来描述稍后分配 SOME/IP 服务 ID 的服务。

为了简化系统设计模型中经典和自适应软件组件之间面向服务的通信的描述，引入了InterfaceMapping ，它允许将经典平台的PortInterface的元素映射到自适应平台的单个ServiceInterface 。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **接口映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign | | | |
| ***笔记*** | 此元类收集单个 ServiceInterface 的元素到 AUTOSAR Classic Platform 的 PortInterface 元素的映射。  **标签：**  atp.Status=草稿  atp.recommendedPackage=接口映射 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 事件映射 | 事件映射 | \* | 聚合 | 将 SenderReceiver 接口中的 VariableDataPrototype 映射到 ServiceInterface 中的事件。  **标签：** atp.Status=draft |
| 字段映射 | 场映射 | \* | 聚合 | 将 ServiceInterface 中的字段映射到表示 getter 和 setter 方法的 ClientServer 操作以及表示字段中的通知程序的 VariableDataPrototype。  **标签：** atp.Status=draft |
| fireAndForget 映射 | FireAndForgetMapping | \* | 聚合 | Fire&Forget 方法的映射，它位于  ServiceInterface 到 Sender ReceiverInterface 中的 VariableDataPrototype 或到 TriggerInterface 中的触发器。  **标签：** atp.Status=draft |
| 方法映射 | 方法映射 | \* | 聚合 | 将 ClientServer 接口中的 ClientServerOperation 映射到 ServiceInterface 中的方法。  **标签：** atp.Status=draft |

**表 5.41：接口映射**

**[constr\_3370]** { DRAFT } **InterfaceMapping应映射单个ServiceInterface的所有元素**d InterfaceMapping中包含的映射应将单个ServiceInterface的所有元素（即字段s、事件s、方法s）映射到经典平台的PortInterface元素. c *()*

*Identifiable*

FieldMapping

*Identifiable*

EventMapping

*Identifiable*

MethodMapping

*Identifiable*

FireAndForgetMapping

*ARElement*

InterfaceMapping

+

eventMapping

0..\*

+

fieldMapping

0..\*

+

methodMapping

0..\*

+

fireAndForgetMapping

0..\*

**图 5.9：接口映射概述**

图5.10显示了一种可能的系统设计建模方法，其中一个应用软件以面向服务的方式通过 SOME/IP 与经典软件组件进行通信。 SWC\_1 需要一个带有ClientServerOperation的ClientServerInterface IF\_Y和一个带有VariableDataPrototype的SenderReceiverInterface IF\_X 。 SWC\_2 需要带有VariableDataPrototype的SenderReceiverInterface IF\_X 。

AA1

P

SWC\_1

R

SWC\_2

R

IF\_A

P

R

IF\_Y

R

IF\_X

IF\_X

IF\_Y

IF\_A

R

IF\_X

Interface

Mapping

Network / SOME/IP Protocol

***AUTOSAR 自适应平台上的应用软件*与经典平台的软件组件之间的面向服务通信的建模示例**

两个PortInterface的 IF\_X 和 IF\_Y使用InterfaceMapping映射到单个ServiceInterface IF\_A 。

另一方面，应用软件AA1提供ServiceInterface IF\_A。

请注意，这是PortInterface级别的映射。如果每个PortInterface在网络中只使用一次，则实际通信可以直接从InterfaceMapping派生出来。如果在网络上多次使用PortInterface ，则需要考虑网络配置，以便能够模拟服务发现在网络上的行为方式。从这些信息中可以推断出软件级别的实际通信关系。

**5.4.1 方法映射**

**[TPS\_MANI\_03111]** { DRAFT }位于**ClientServerInterface中的方法和操作之间**的映射d位于ServiceInterface中的方法和位于ClientServerInterface中的操作之间的映射由MethodMapping类提供。 c *( RS\_MANI\_00026 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **方法映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign | | | |
| ***笔记*** | 将位于 ClientServerInterface 中的 ClientServerOperation 映射到位于 ServiceInterface 中的 Method。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 客户端服务器操作 | 客户端服务器操作 | 0..1 | 参考 | 对位于 ClientSeverInterface 中的 ClientSeverOperation 的引用。  **标签：** atp.Status=draft |
| 方法 | 客户端服务器操作 | 0..1 | 参考 | 对位于服务中的方法的引用  界面。  **标签：** atp.Status=draft |

**表 5.42：方法映射**

|  |  |  |
| --- | --- | --- |
| |  | | --- | | *端口接口*  客户端服务器接口 | |  |

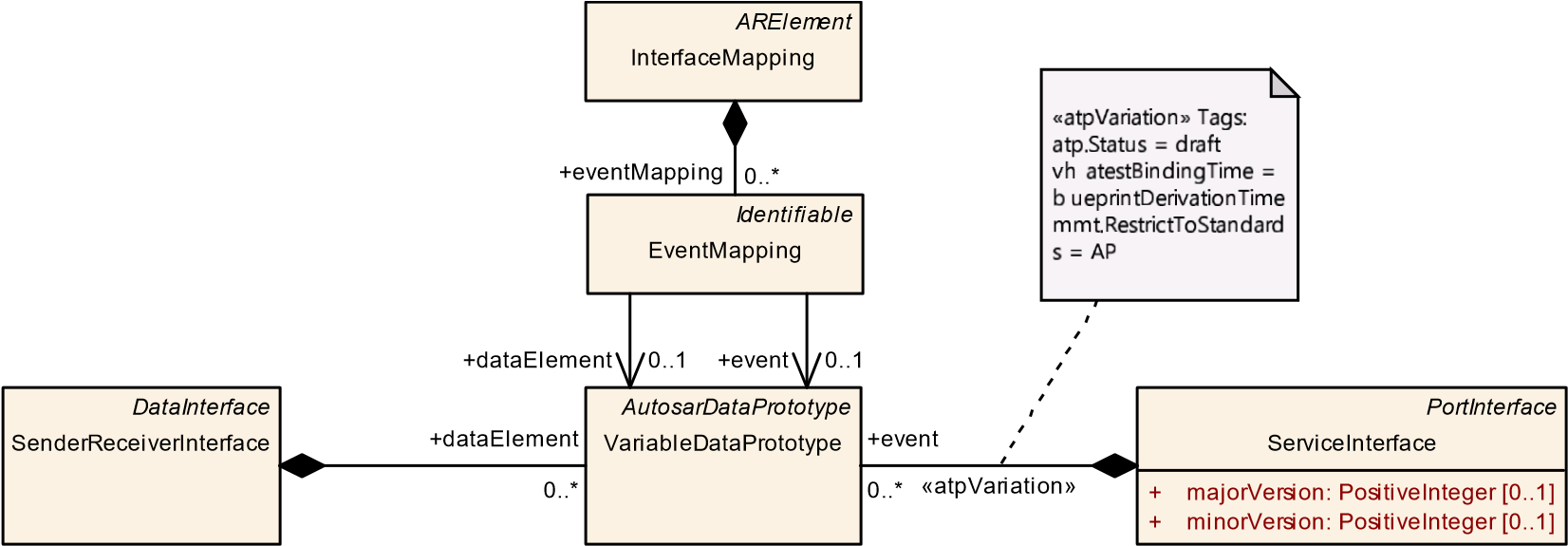
**图 5.11：方法到 ClientServerOperation 的映射**

**5.4.2 事件映射**

**[TPS\_MANI\_03112]** { DRAFT }**事件与数据元素之间**的映射d位于ServiceInterface中的事件与位于SenderReceiverInterface中的数据元素之间的映射由EventMapping类提供。 c *( RS\_MANI\_00026 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **事件映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign | | | |
| ***笔记*** | 将位于 SenderReceiverInterface 中的 VariableDataPrototype 映射到位于 ServiceInterface 中的事件。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 数据元素 | 变量数据原型 | 0..1 | 参考 | 对位于 SenderReceiverInterface 中的 VariableDataPrototype 的引用。  **标签：** atp.Status=draft |
| 事件 | 变量数据原型 | 0..1 | 参考 | 对位于服务中的事件的引用  界面。  **标签：** atp.Status=draft |

**表 5.43：事件映射**



**图 5.12：事件和数据元素之间的映射**

**5.4.3 字段映射**

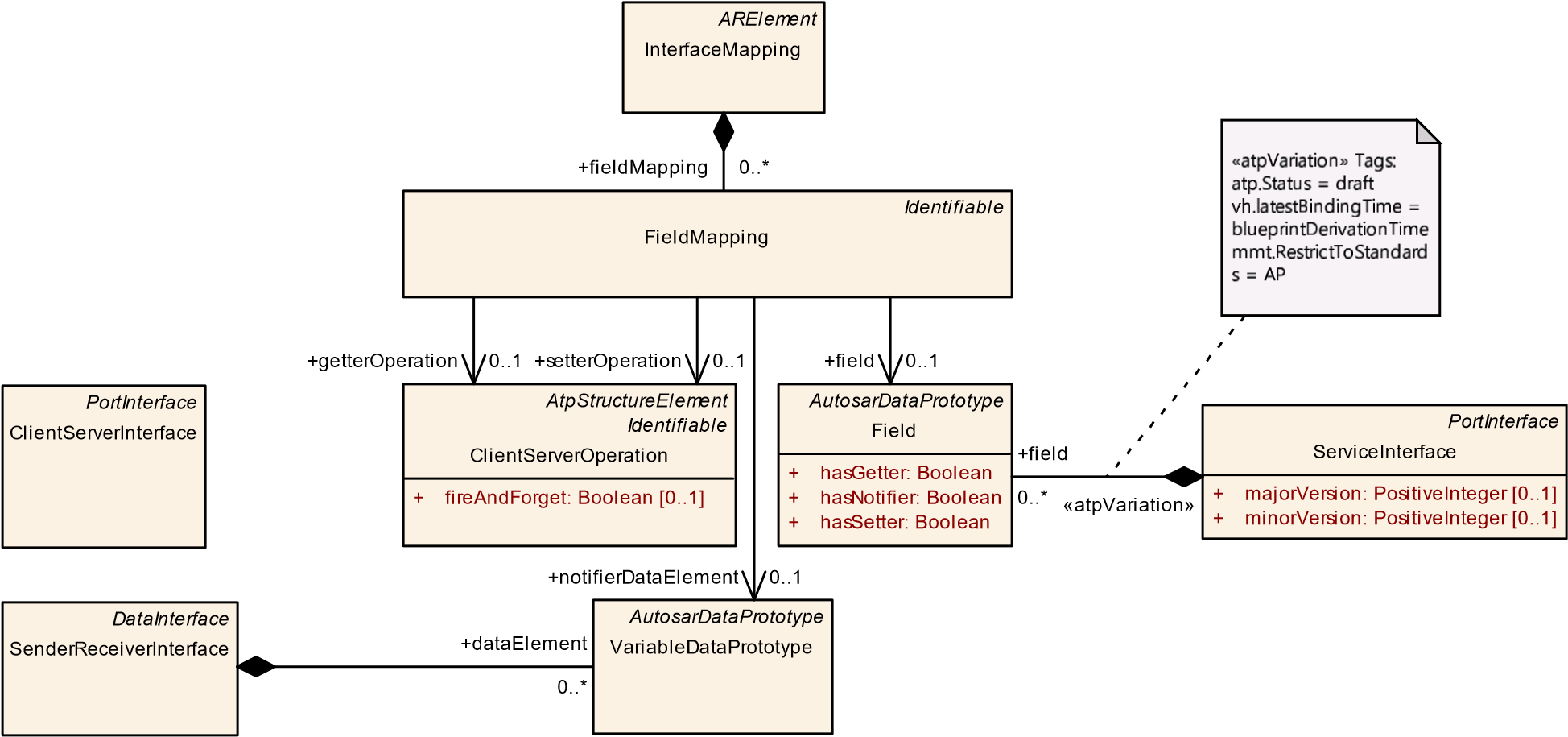
**[TPS\_MANI\_03113]** { DRAFT }字段**与 Classic Platform PortInterface s** d元素**之间**的映射位于ServiceInterface中的**字段与**Classic Platform PortInterface s 元素之间的映射由FieldMapping类提供。经典平台中的字段通知器由位于SenderReceiverInterface中的dataElement表示。经典平台中的 getter 和 setter 方法由位于ClientServerInterface中的操作表示。 c *( RS\_MANI\_00026 )*

**[constr\_3367]** {草稿}**字段映射。 notifierDataElement引用**d \_

字段中的hasNotifier属性设置为 true ，则FieldMapping应仅包含notifierDataElement引用。 c *()*

**[constr\_3368]** {草稿}**字段映射。 getterOperation引用**d如果引用字段中的hasGetter属性设置为 true ，则FieldMapping应仅包含getterOperation引用。 c *()*

**[constr\_3369]** {草稿}**字段映射。 setterOperation引用**d如果引用字段中的hasSetter属性设置为 true ，则FieldMapping应仅包含setterOperation引用。 c *()*



**图 5.13： Classic Platform PortInterface的字段和元素之间的映射**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **场映射** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign | | | |
| ***笔记*** | 将位于 ServiceInterface 中的字段映射到表示 getter 和 setter 方法的 ClientServerOperations 以及表示字段中的通知程序的 VariableDataPrototype。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 场地 | 场地 | 0..1 | 参考 | 对位于 ServiceInterface 中的字段的引用。  **标签：** atp.Status=draft |
| getter操作 | 客户端服务器操作 | 0..1 | 参考 | 对表示字段中的 getter 方法的 ClientServerOperation 的引用。  **标签：** atp.Status=draft |
| notifierData 元素 | 变量数据原型 | 0..1 | 参考 | 对表示字段中通知程序的 VariableDataPrototype 的引用。  **标签：** atp.Status=draft |
| 设置器操作 | 客户端服务器操作 | 0..1 | 参考 | 对表示字段中的 setter 方法的 ClientServerOperation 的引用。  **标签：** atp.Status=draft |

**表 5.44：字段映射**

**5.4.4 FireAndForget 映射**

在“一劳永逸”的消息交换模式中，消费者向提供者发送消息，不期望得到响应，如第3.4.5.1章所述。

在 Adaptive AUTOSAR 中，fire and forget 方法是用属性 method的值来描述的。 fireAndForget设置为 [ TPS\_MANI\_01064 ]定义的 true 。

在经典的 AUTOSAR 中，不能用ClientServerOperation描述即发即弃的方法，因为客户端-服务器调用总是有响应。因此，如果 fire and forget 方法包含输入参数，则使用VariableDataPrototype 。

如果 fire and forget 方法包含多个输入参数，则VariableDataPrototype需要是 Structure 类型，它为 fire and forget 方法的每个参数托管一个元素。重要的是，结构中元素的顺序与ClientServerOperation中的ArgumentDataPrototype的顺序相同。

这种表示可确保 SOME/IP 序列化产生与 Adaptive Platform 中相同的字节流，其中具有方向 in的所有参数s 根据ClientServerOperation中的ArgumentDataPrototype s的顺序进行序列化。

如果 fire and forget 方法没有任何参数，则使用Trigger来描述经典 AUTOSAR 中的此类方法。

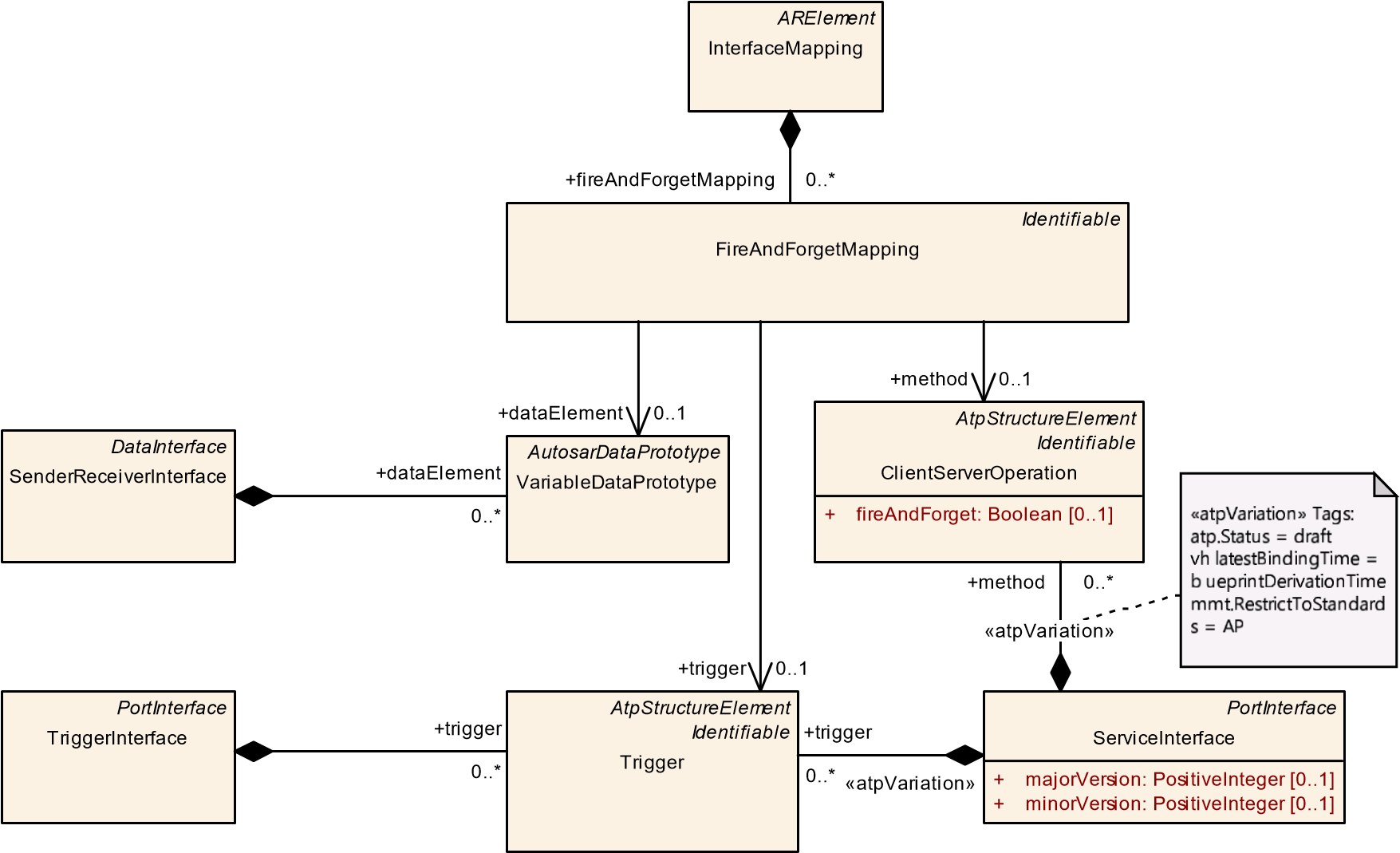
如果通过 SOME/IP 传输一个即发即弃的方法，那么将 SOME/IP MessageType 设置为 REQUEST\_NO\_RETURN 很重要。

**[TPS\_MANI\_03115]** { DRAFT } **Fire and forget方法与 Classic Platform PortInterface s** d元素之间的映射属性method的值的方法之间的映射。 fireAndForget设置为 true，Classic Platform PortInterface的元素由FireAndForgetMapping类提供。

如果在经典平台中通过VariableDataPrototype表示即发即弃方法，则此dataElement将映射到位于ServiceInterface中的方法。如果在经典平台中触发并忘记方法由Trigger表示，那么此触发器将映射到位于ServiceInterface中的方法。 c *( RS\_MANI\_00026 )*

**[constr\_3371]** { DRAFT } **FireAndForgetMapping的互斥存在。 dataElement参考和FireAndForgetMapping 。触发器引用**d FireAndForgetMapping决不能同时引用数据元素和触发器。 c *()*

[ **constr\_3376** ] { DRAFT } **FireAndForgetMapping**应该只引用触发和**忘记方法\_** \_ \_ \_ fireAndForget设置为 true。 c *()*



**方法与元素之间的映射**

**端口接口**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **FireAndForgetMapping** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign | | | |
| ***笔记*** | 将位于 ServiceInterface 中的 Fire&Forget 方法映射到 SenderReceiverInterface 中的 VariableDataPrototype 或映射到 TriggerInterface 中的触发器。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 数据元素 | 变量数据原型 | 0..1 | 参考 | 对位于  如果 Fire&Forget 方法由此 VariableDataPrototype 表示，则为 SenderReceiverInterface。  **标签：** atp.Status=draft |
| 方法 | 客户端服务器操作 | 0..1 | 参考 | 对位于 ServiceInterface 中的 Fire&Forget 方法的引用。  **标签：** atp.Status=draft |
| 扳机 | 扳机 | 0..1 | 参考 | 如果 Fire&Forget 方法由此触发器表示，则引用位于 TriggerInterface 中的触发器。  **标签：** atp.Status=draft |

**表 5.45：FireAndForgetMapping**

# 6 子系统设计

## 6.1 概述

*AUTOSAR 自适应平台作为在现场部署软件单元的平台*的性质意味着可以在现场安装的软件单元需要一些设计支持。

更具体地说，可以在现场部署的软件单元通常代表某种或多或少的独立驾驶功能。

换句话说，为此目的的设计支持需要进行定制，以促进与其他应用程序级软件通信的应用程序级软件的设计。

假设这种设计的第一步是定义正在开发的驱动功能所提供的服务和所需的服务。

这种对所需和提供的服务的定义可以用作其他这种驾驶功能设计的输入，并且随着时间的推移，呈现驾驶功能级别的通信视图。

进一步假设 OEM 最感兴趣的是驾驶功能的通信视图，并且可以将各个驾驶功能分包给一级供应商。

这意味着对于一级供应商而言，提供的和所需的驱动功能服务列表代表了开发该功能的技术合同。

在设计层面，元类SoftwareClusterDesign用于对可能代表这种驱动功能的软件进行形式化。换句话说，假设存在一个工作流，其中 AUTOSAR 自适应平台上的特定功能的设计从SoftwareClusterDesign的创建开始。

在这种情况下，进一步假设为相应功能定义所需和提供的服务实例是开发的良好起点。

请注意， SoftwareClusterDesign支持任意复杂的软件，因此不受单一驱动功能等设计的约束。

## 6.2 软件集群设计

**免责声明：本章已过时，将在以后的版本中更新。**

**[TPS\_MANI\_01112]** { DRAFT } SoftwareClusterDesign的**语义**d **SoftwareClusterDesign**的存在代表了对最初由 OEM 制定的需求的正式响应，并且随着软件开发的进展可能会得到丰富。

最后，集成应将SoftwareClusterDesign作为集成步骤结果定义的进一步输入： SoftwareCluster的定义。 c *( RS\_MANI\_00035 )*

可以肯定的是， SoftwareClusterDesign并不打算上传到目标平台。它只是最终被上传的软件集群的早期形式。 SoftwareClusterDesign的存在是从方法论的角度出发的。

**[constr\_1557]** { DRAFT } **SoftwareClusterDesign的标准化值。类别和SoftwareCluster 。类别**d AUTOSAR 标准保留以下属性SoftwareClusterDesign的值。类别和SoftwareCluster 。类别：

* ROOT\_SOFTWARE\_CLUSTER
* SUB\_SOFTWARE\_CLUSTER c *()*

**[TPS\_MANI\_01161]** { DRAFT }**类别值对语义的影响**

**SoftwareClusterDesign** d类别 ROOT\_SOFTWARE\_CLUSTER的SoftwareClusterDesign可以引用角色subSoftwareCluster中类别 SUB\_SOFTWARE\_CLUSTER的其他SoftwareClusterDesign ，从而提供进一步分解SoftwareClusterDesign的创建的方法。 c *( RS\_MANI\_00035 )*

**[constr\_1558]** { DRAFT } **SoftwareClusterDesign的存在。 diagnosticAddress** d SoftwareClusterDiagnosticAddress在SoftwareClusterDesign中的角色diagnosticAddress的聚合仅在SoftwareClusterDesign的值存在时才存在。类别设置为ROOT\_SOFTWARE\_CLUSTER 。 c *()*

**[constr\_1559]** { DRAFT } **SoftwareClusterDesign的存在。 subSoftwareCluster** d角色subSoftwareCluster中从SoftwareClusterDesign对其自身的引用仅在SoftwareClusterDesign的值存在时才存在。类别设置为ROOT\_SOFTWARE\_CLUSTER 。 c *()*

**[constr\_1560]** { DRAFT } **SoftwareClusterDesign的使用。 requiredARElement** d参考SoftwareClusterDesign 。 requiredARElement不得用于指代另一个SoftwareClusterDesign甚至SoftwareCluster 。 c *()*

constr\_1560 ]存在的理由：为引用SoftwareClusterDesign定义了专用引用。

**[TPS\_MANI\_01211]** { DRAFT } **SoftwareClusterDesign中可执行软件的规范**d可上传软件包最突出的内容之一是对可执行软件的引用。

在SoftwareClusterDesign的定义中，此引用通过引用SoftwareCluster隐式给出。包含进程。

SoftwareClusterDesign的目标。 containsProcess是一个ProcessDesign ，表示相应可执行程序（软件映像）的实例（形式化为Process ）的设计级表示，形式化为Executable c *( RS\_MANI\_00035 )*

*AtpClassifier*

SoftwareClusterDesign

DiagnosticContributionSet

*UploadablePackageElement*

*PackageableElement*

*ARElement*

*SoftwareClusterDiagnosticAddress*

addressSemantics: SoftwareClusterDiagnosticAddressSemanticsEnum

+

SoftwareClusterDoipDiagnosticAddress

+

diagnosticAddress: PositiveInteger

[0..1]

«enumeration»

SoftwareClusterDiagnosticAddressSemanticsEnum

physicalAddress

functionalAddress

*PackageableElement*

*FibexElement*

*AtpStructureElement*

MachineDesign

+

[0..1]

accessControl: AccessControlEnum

pncPrepareSleepTimer: TimeValue

+

[0..1]

[0..1]

+

pnResetTimer: TimeValue

ProcessDesign

«atpSplitable»

diagnosticContribution

+

0..\*

«atpSplitable»

+

requiredARElement

0..\*

«atpSplitable»

diagnosticAddress

+

0..\*

«atpSplitable»

+

requiredPackageElement

0..\*

«atpUriDef»

intendedTargetMachine

+

0..1

0..\*

dependsOn

+

«atpSplitable»

+

subSoftwareCluster

0..\*

«atpSplitable»

requiredFibexElement

+

0..\*

«atpSplitable»

+

containedProcess

0..\*

**图 6.1： SoftwareClusterDesign的建模**

**[TPS\_MANI\_01113]** {草案} **SoftwareClusterDesign的语义。 diagnosticAddress** d属性SoftwareClusterDesign的存在。即使在开发的早期阶段，也可以使用诊断地址来表达有关诊断地址分布的信息，即这通常由 OEM 完成。

这包括指定多个（即几个功能加一个物理）诊断地址的能力，因此diagnosticAddress的多重性设置为 0..\*。 c *( RS\_MANI\_00035 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **软件集群设计** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign | | | |
| ***笔记*** | 此元类代表 OEM 设计可上传到特定目标机器的软件分组的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=软件集群设计 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpClassifier* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 包含过程 | 流程设计 | \* | 参考 | 此参考代表包含在封闭 SoftwareCluster 中的 ProcessDesign。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=containedProcess  atp.Status=草稿 |
| 取决于 | 软件集群设计 | \* | 参考 | 所有者 SoftwareClusterDesign 依赖于引用的 SoftwareClusterDesign  **标签：** atp.Status=draft |
| 诊断地址 | 软件集群  诊断地址 | \* | 聚合 | 此聚合用于指定诊断地址。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=诊断地址 atp.Status=草稿 |
| 诊断  贡献 | 诊断贡献  放 | \* | 参考 | 此参考标识了相应的 DiagnosticContributionSet 集合。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=diagnosticContribution atp.Status=草稿 |
| 预期目标机 | 机器设计 | 0..1 | 参考 | 此参考可用于识别最终 SoftwareCluster 应为其开发的机器设计。  **刻板印象：** atpUriDef**标签：** atp.Status=draft |
| 必需的  AR元素 | AR元素 | \* | 参考 | 该参考代表了完整定义 SoftwareCluster 所需的 ARElement 集合。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=requiredARElement atp.Status=草稿 |
| requiredFibex 元素 | 纤维元素 | \* | 参考 | 此引用表示完成 SoftwareCluster 定义所需的 fibexElements 的集合。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=requiredFibexElement atp.Status=draft |

5 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **软件集群设计** |  |  |  |
| 所需包  元素 | 可上传包  元素 | \* | 参考 | 此参考指向已在封闭 SoftwareClusterDesign 的上下文中标识为相关的可上传元素。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=requiredPackageElement atp.Status=草稿 |
| 根  作品 | RootSwClusterDesign  组件原型 | 0..1 | 聚合 | 此聚合表示通信端点的 SwClusterDesign 术语中的软件设计。  **标签：** atp.Status=draft |
| 子软件集群 | 软件集群设计 | \* | 参考 | 此参考用于识别“伞形”SoftwareClusterDesign 的子 SoftwareCluster 设计。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=subSoftwareCluster atp.Status=草稿 |

**表 6.1：软件集群设计**

**[TPS\_MANI\_01117]** {草案} **SoftwareClusterDesign的语义。 expectedTargetMachine** d SoftwareClusterDesign的规范。 expectedTargetMachine 允许将可上传软件包的规范从开发项目的早期阶段集中到特定的MachineDesign 。 c *( RS\_MANI\_00035 )*

请注意， SoftwareCluster没有对目标Machine的专用引用。

这种关系通过对 Process 的引用来表达，而Process又可以通过ProcessToMachineMapping映射到专用机器。在这种情况下，[ constr\_1536 ] 适用。

**[TPS\_MANI\_01118]** { DRAFT } **SoftwareClusterDesign和DiagnosticContributionSet** d之间的**关系**SoftwareClusterDesign定义的一个重要方面是什么诊断提取应与SoftwareClusterDesign相关联的问题。

为此，提供了从SoftwareClusterDesign到角色diagnosticContribution中的DiagnosticContributionSet的引用。

在开发过程的早期阶段，有意使引用多个DiagnosticContributionSet成为可能，以支持诊断堆栈的分散式（例如，部分由 OEM 完成，部分由供应商完成）配置。 c *( RS\_MANI\_00035 )*

**[TPS\_MANI\_01189]** { DRAFT }**软件集群和诊断贡献-**

**设置。类别**d在上下文中使用 的诊断贡献集

SoftwareCluster应将 属性类别的值设置为 DIAGNOSTICS\_SWCL\_EXTRACT。 c *( RS\_MANI\_00035 )*

**[constr\_1562]** { DRAFT } **SoftwareClusterDesign的存在。 diagnosticContribution** d引用SoftwareClusterDesign的存在。 diagnosticContribution仅限于SoftwareClusterDesign ，其中属性类别设置为值ROOT\_SOFTWARE\_CLUSTER 。 c *()*

constr\_1562 ]存在的理由：诊断行为的定义仅限于SoftwareClusterDesign结构的根级别，其精神与导致 [ constr\_1558 ] 存在的精神相同。

SoftwareCluster和SoftwareClusterDesign在与DiagnosticContributionSet的关系方面故意引入的差异。

换句话说，对DiagnosticContributionSet的引用的多样性是有意不同的。

如前所述， SoftwareClusterDesign应支持DiagnosticContributionSet的分散配置，而SoftwareCluster需要存在最终（合并）的DiagnosticContributionSet 。

**[TPS\_MANI\_01119]** { DRAFT }**对来自SoftwareClusterDesign的模型元素的引用**d SoftwareClusterDesign能够定义对与定义可上传软件包相关的模型元素的以下引用：

* UploadablePackageElement派生的元类的引用通过SoftwareClusterDesign形式化。必需的PackageElement 。
* ARElement派生的元类的引用通过SoftwareClusterDesign形式化。必需的ARElement 。
* FibexElement派生的元类的引用通过SoftwareClusterDesign形式化。必需的FibexElement 。

c *( RS\_MANI\_00035 )*

请注意，将 SoftwareClusterDesign 转换为SoftwareCluster不是由AUTOSAR 形式化的。此步骤可以由集成商自行决定的工具完成。

换句话说，在某些情况下，在开发项目中相对较早地进行此转换可能是适用的，而其他项目可能需要将SoftwareClusterDesign保留更长的时间。

**[TPS\_MANI\_01310]** {草案} **SoftwareClusterDesign的语义。 dependsOn** d参考SoftwareClusterDesign 。 dependsOn可用于准备在设计级别上已经存在的SoftwareCluster之间存在的依赖关系的定义。 c *( RS\_MANI\_00035 )*

换句话说， SoftwareClusterDesign的定义。构建SoftwareClusterDesign的一致模型当然不需要dependsOn 。该引用只能用于“预先加载”依赖关系的形式化，这些依赖关系以后可能会在SoftwareCluster的设计级别上发生。

## 6.3 软件集群设计提供和需要的服务

为了在SoftwareCluster设计的早期支持定义所需和提供的服务[[14]](#footnote-14)，AUTOSAR 支持在给定SoftwareClusterDesign的上下文中定义RootSwClusterDesignComponentPrototype 。

RootSwClusterDesignComponentPrototype本身是指一个SwComponentType ，它反过来将PortPrototype暴露给外界。

请注意，对于 RootSwClusterDesignComponentPrototype 的特定情况，预计所引用的SwComponentType表示一个CompositionSwComponentType ，无需任何进一步的详细说明。详细说明显然是不必要的，因为唯一的目的是公开可以映射AdaptivePlatformServiceInstance的PortPrototype 。

一个专用映射类ServiceInstanceToSwClusterDesignPortPrototypeMapping以支持在PortPrototype和AdaptivePlatformServiceInstance之间创建所描述的关系。

*ARElement*

*ServiceInstanceToSwClusterDesignPortPrototypeMapping*

*UploadablePackageElement*

*AdaptivePlatformServiceInstance*

*ProvidedApServiceInstance*

PPortPrototype

*PortPrototype*

*AbstractProvidedPortPrototype*

ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping

«instanceRef»

+

providedPortPrototype

0..1

+

providedServiceInstance

0..1

**图 6.2： ProvidedServiceInstanceToSwClusterDesignPPort的建模-**

**原型映射**

**[TPS\_MANI\_01275]** { DRAFT }**元类ServiceInstanceToSwClusterDesignPortPrototypeMapping的语义**d用于键入RootSwClusterDesignComponentPrototype的软件组件通常会向外界公开一组PortPrototype 。

这些PortPrototype可用于指定所需和提供的服务实例。为此，使用了元类ServiceInstanceToSwClusterDesignPortPrototypeMapping 。 c *( RS\_MANI\_00011 )*

在图6.4中， ServiceInstanceToSwClusterDesignPortPrototypeMapping由标记为“mapping”且带有圆圈 1 的块表示。标记为“mapping”且带有圆圈 2 的块表示CompositionPortToExecutablePortMapping ，如第6.4节所述。

*ARElement*

*ServiceInstanceToSwClusterDesignPortPrototypeMapping*

*UploadablePackageElement*

*AdaptivePlatformServiceInstance*

RequiredServiceInstanceToSwClusterDesignRPortPrototypeMapping

RPortPrototype

*PortPrototype*

*AbstractRequiredPortPrototype*

*RequiredApServiceInstance*

«instanceRef»

+

requiredPortPrototype

0..1

+

requiredServiceInstance

0..1

**图 6.3： RequiredServiceInstanceToSwClusterDesignRPort的建模-**

**原型映射**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **RootSwClusterDesignComponentPrototype** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution | | | |
| ***笔记*** | 此元类表示在 SwClusterDesign 范围内定义服务端点的能力。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *AtpFeature* , *AtpPrototype* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 申请类型 | 组件类型 | 1 | 参考 | 此 SwComponentType 充当 RootSw ClusterDesignComponentPrototype 的类型。  **标签：** atp.Status=draft |

**表 6.2：RootSwClusterDesignComponentPrototype**

SoftwareClusterDesign

SwComponentType (referenced by

RootSwClusterDesignComponentPrototype)

ProvidedServiceInstance 1

RequiredServiceInstance 45

Executable A

SwComponentType (referenced by

RootSwComponentPrototype)

Mapping

Mapping

Mapping

1

1

2

Executable B

SwComponentType (referenced by

RootSwComponentPrototype)

The position (left/right) of PortPrototypes does

not make any implications on their directions.

Mapping

2

ProcessDesign

Mapping

2

ProcessDesign

ProcessDesign

**图 6.4： SoftwareClusterDesign上下文中的映射建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***ServiceInstanceToSwClusterDesignPortPrototypeMapping*** （抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow | | | |
| ***笔记*** | 此抽象元类表示将依赖于传输层的 ServiceInstance 分配给 SoftwareClusterDesign 上下文中的 PortPrototype 的能力。使用此映射，可以在 SoftwareCluster 设计的范围内定义提供的和需要的 AdaptivePlatformServiceInstances 的列表。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***子类*** | 提供ServiceInstanceToSwClusterDesignPPortPrototypeMapping ， RequiredServiceInstanceToSw ClusterDesignRPortPrototypeMapping | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 6.3：ServiceInstanceToSwClusterDesignPortPrototypeMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **必需的ServiceInstanceToSwClusterDesignRPortPrototypeMapping** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow | | | |
| ***笔记*** | 这个具体的元类表示在 SoftwareClusterDesign 的上下文中将依赖于传输层的 RequiredService 实例分配给 RPortPrototype 的能力。使用此映射，可以在 SoftwareClusterDesign 的范围内定义提供的和需要的 AdaptivePlatformServiceInstances 的列表。  **标签：** atp.Status=draft  atp.recommendedPackage=ServiceInstanceToSwClusterDesignPortPrototypeMappings | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable*  *元素*,*可引用*, *ServiceInstanceToSwClusterDesignPortPrototypeMapping* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 所需端口原型 | RPort原型 | 0..1 | 参考 | 该参考标识了 SwClusterDesign 范围内的适用 PortPrototype。  **标签：** atp.Status=draft  **InstanceRef 实现者：** RPortPrototypeInSoftware ClusterDesignInstanceRef |
| 所需服务实例 | 必需的应用服务  实例 | 0..1 | 参考 | 对映射到 SwCluster 设计范围内给定 RPortPrototype 的 RequiredServiceInstance 的引用。  **标签：** atp.Status=draft |

**表 6.4：RequiredServiceInstanceToSwClusterDesignRPortPrototypeMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **提供的ServiceInstanceToSwClusterDesignPPortPrototypeMapping** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow | | | |
| ***笔记*** | 这个具体的元类表示在 SoftwareClusterDesign 的上下文中将依赖于传输层的 ProvidedService 实例分配给 PPortPrototype 的能力。使用此映射，可以在 Software ClusterDesign 的范围内定义提供的和需要的 AdaptivePlatformServiceInstances 的列表。  **标签：**  atp.Status=草稿  atp.recommendedPackage=ServiceInstanceToSwClusterDesignPortPrototypeMappings | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable*  *元素*,*可引用*, *ServiceInstanceToSwClusterDesignPortPrototypeMapping* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |

5

4

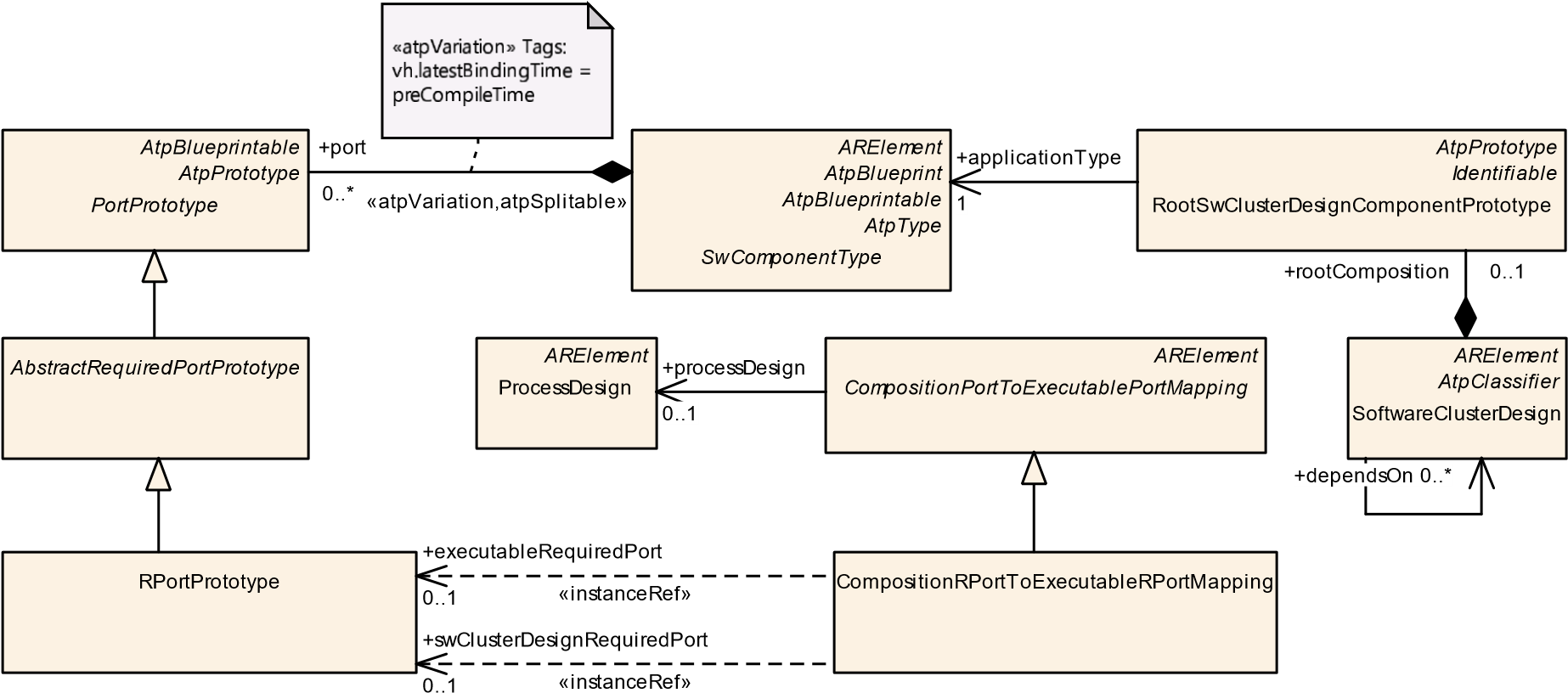
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **提供的ServiceInstanceToSwClusterDesignPPortPrototypeMapping** | | | |
| 提供的端口原型 | PPort原型 | 0..1 | 参考 | 该参考标识了 SwClusterDesign 范围内的适用 PortPrototype。  **标签：** atp.Status=draft  **InstanceRef 实现者：** PPortPrototypeInSoftware ClusterDesignInstanceRef |
| 提供的服务实例 | 提供的ApService  实例 | 0..1 | 参考 | 对映射到 SwCluster 设计范围内给定 PPortPrototype 的 ProvidedServiceInstance 的引用。  **标签：** atp.Status=draft |

**表 6.5：ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping**

## 6.4 服务到可执行文件的映射

设计工作流程中典型的下一步可能是决定

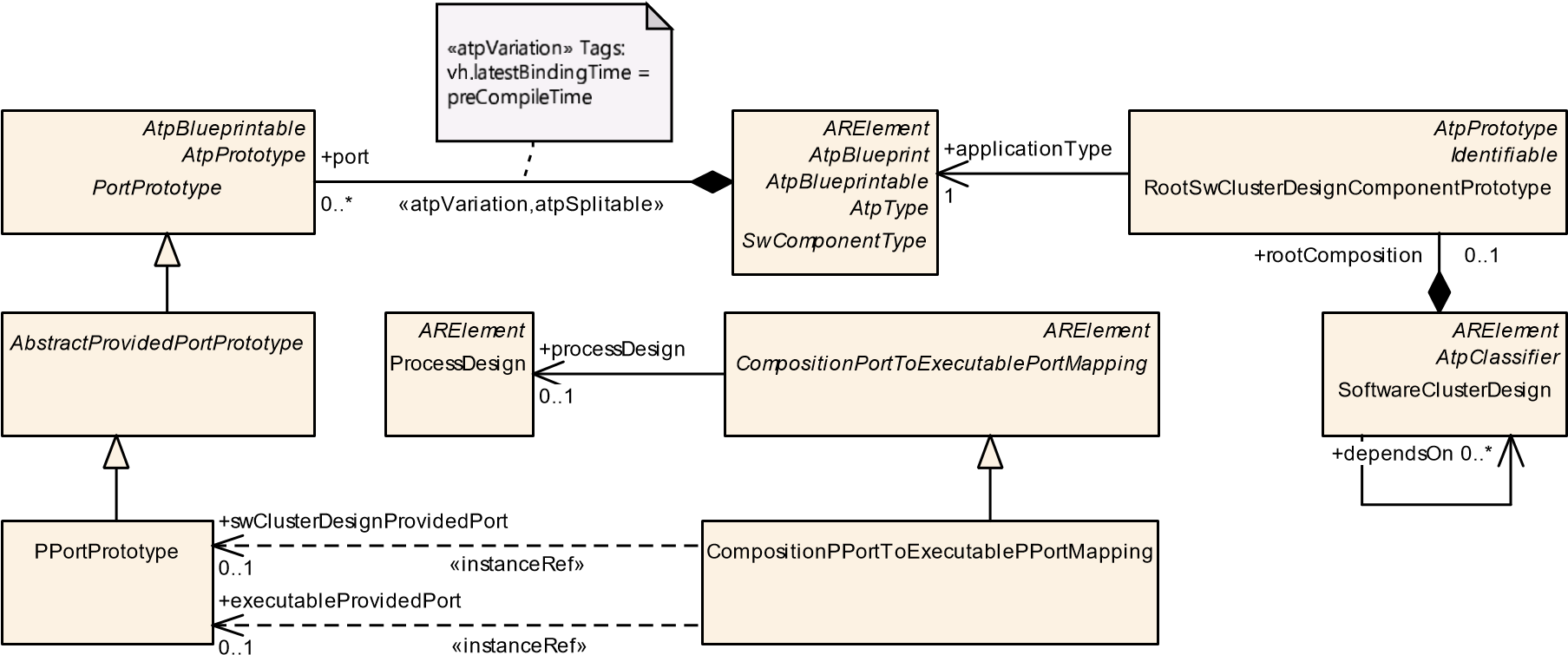
SoftwareClusterDesign中的可执行文件。可执行文件建模中使用的PortPrototype实际实现了所需和提供的服务实例应映射到的端点。



**图 6.5： RootSwClusterDesignComponentPrototype和CompositionRPortToExecutableRPortMapping的建模**

**[TPS\_MANI\_01276]** { DRAFT } **CompositionRPortToExecutableRPortMapping和CompositionPPortToExecutablePPortMapping** d的**语义 在**创建SoftwareClusterDesign的上下文中，不可能已经定义PortPrototype到AdaptivePlatformServiceInstance的实际映射。

为了解决这个问题，并且作为稍后创建实际ServiceInstanceToPortPrototypeMapping的附加指导，可以在SoftwareClusterDesign范围内创建另一个映射，将在RootSwClusterDesignComponentPrototype的上下文中定义的PortPrototype映射到定义的细化PortPrototype在Executable的上下文中。 c *( RS\_MANI\_00011 )*



**图 6.6： RootSwClusterDesignComponentPrototype和CompositionPPortToExecutablePPortMapping的建模**

这样，可以更深入地追溯 RootSwClusterDesignComponentPrototype级别的设计决策，并为创建ServiceInstanceToPortPrototypeMapping提供指导，如第10.3节所述。

**[TPS\_MANI\_01282]** { DRAFT }**参考CompositionPortToExecutablePortMapping的语义。 processDesign** d参考CompositionPortToExecutablePortMapping 。 processDesign标识映射的适用ProcessDesign 。因此，此引用消除了在Executable上下文中引用完全相同的PortPrototype的多个CompositionPortToExecutablePortMapping的存在。 c *( RS\_MANI\_00011 )*

TPS\_MANI\_01282 ]的陈述在图6.4中进一步解释。二

CompositionPortToExecutablePortMapping指的是Executable B表面上的同一个PortPrototype 。

重要的是要了解这些CompositionPortToExecutablePortMapping中的每一个都引用不同的ProcessDesign 。

这意味着，在运行时，两个CompositionPortToExecutablePortMapping应用于作为不同Process启动的Executable B 的不同实例（每个依次引用Executable B引用的ProcessDesign之一）。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***CompositionPortToExecutablePortMapping*** （抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow | | | |
| ***笔记*** | 此抽象元类充当基类，用于规范 RootSwClusterDesignComponentPrototype 拥有的 PortPrototype 与 Executable.rootSwComponentType 内的 Component Prototype 拥有的 PortPrototype 之间的映射。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***子类*** | CompositionPPortToExecutablePPortMapping , CompositionRPortToExecutableRPortMapping | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 流程设计 | 流程设计 | 0..1 | 参考 | 此参考标识了此映射受影响的 ProcessDesign。这允许通过引用不同的 ProcessDesign 将多个服务映射到 Executable 上的相同 PortPrototype。  **标签：** atp.Status=draft |

**表 6.6：CompositionPortToExecutablePortMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **组合RPortToExecutableRPortMapping** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow | | | |
| ***笔记*** | 此元类能够将在 SwCluster 设计的上下文中定义的 RPortPrototype 与在可执行文件的上下文中的 RPortPrototype 相关联。  **标签：**  atp.Status=草稿  atp.recommendedPackage=CompositionPortToExecutablePortMappings | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *CompositionPortToExecutablePortMapping* , *Identifiable* ,  *MultilanguageReferable* , *PackageableElement* ,*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 可执行的  必需端口 | RPort原型 | 0..1 | 参考 | 该参考在可执行文件的上下文中标识了适用的 PortPrototype。  **标签：** atp.Status=draft  **InstanceRef 由：** RPortPrototypeIn实现  可执行实例引用 |
| swCluster  设计必填  港口 | RPort原型 | 0..1 | 参考 | 该参考在 SwClusterDesign 的上下文中标识了适用的 RPortPrototype。  **标签：** atp.Status=draft  **InstanceRef 实现者：** RPortPrototypeInSoftware ClusterDesignInstanceRef |

**表 6.7：CompositionRPortToExecutableRPortMapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **组合PPortToExecutablePPortMapping** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow | | | |
| ***笔记*** | 该元类能够将在 SwClusterDesign 的上下文中定义的 PPortPrototype 与在 Executable 的上下文中的 PPortPrototype 相关联。  **标签：**  atp.Status=草稿  atp.recommendedPackage=CompositionPortToExecutablePortMappings | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *CompositionPortToExecutablePortMapping* , *Identifiable* ,  *MultilanguageReferable* , *PackageableElement* ,*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **组合PPortToExecutablePPortMapping** | | | |
| 可执行的  提供端口 | PPort原型 | 0..1 | 参考 | 该参考在可执行文件的上下文中标识了适用的 PortPrototype。  **标签：** atp.Status=draft  **InstanceRef 由：** PPortPrototypeIn实现  可执行实例引用 |
| swCluster  设计提供  港口 | PPort原型 | 0..1 | 参考 | 该参考标识了 SwClusterDesign 上下文中适用的 PPortPrototype。  **标签：** atp.Status=draft  **InstanceRef 实现者：** PPortPrototypeInSoftware ClusterDesignInstanceRef |

**表 6.8：CompositionPPortToExecutablePPortMapping**

# 7 机器清单

Machine元类定义了一个*Adaptive AUTOSAR 软件堆栈*与操作系统一起运行的实体。机器可以是物理的或虚拟的。

机器的某些方面已经可以从MachineDesign的系统设计（见第5.2章）中获得。 MachineDesign中定义的信息也可用于Machine ，因为Machine在角色machineDesign中引用了MachineDesign （参见图5.1 ）。

该机器能够聚合一个或多个处理器。每个处理器由一个或多个ProcessorCore组成。

元类ProcessorCore提供属性coreId ，例如可以在位掩码中使用以更好地控制处理资源的使用。

**[constr\_1549]** { DRAFT } **ProcessorCore的值。 coreId** d ProcessorCore的值。 coreId在封闭处理器的上下文中应该是唯一的。 c *()* Machine元类的概述如图 7.1 所示。

*ARElement*

*AtpStructureElement*

Machine

trustedPlatformExecutableLaunchBehavior: TrustedPlatformExecutableLaunchBehaviorEnum

+

*ServiceDiscoveryConfiguration*

*Identifiable*

*CommunicationConnector*

[0..1]

createEcuWakeupSource: Boolean

+

pncFilterArrayMask: PositiveInteger [0..\*] {ordered

+

}

*Identifiable*

*AdaptiveModuleInstantiation*

*Identifiable*

ProcessorCore

coreId: PositiveInteger

+

*Identifiable*

Processor

*AtpStructureElement*

*FibexElement*

MachineDesign

accessControl: AccessControlEnum

+

[0..1]

[0..1]

pncPrepareSleepTimer: TimeValue

+

[0..1]

pnResetTimer: TimeValue

+

TagWithOptionalValue

key: String

+

sequenceOffset: Integer

+

[0..1]

value: String

[0..1]

+

«enumeration»

TrustedPlatformExecutableLaunchBehaviorEnum

strictMode

monitorMode

noTrustedPlatformSupport

«atpSplitable»

+

moduleInstantiation

0..\*

«atpSplitable»

+

communicationConnector

0..\*

+

processor

1..\*

+

core

1..\*

«atpSplitable»

+

environmentVariable

0..\*

+

machineDesign

1

«atpSplitable»

+

serviceDiscoveryConfig

0..\*

**图 7.1：机器配置内容概览**

**[TPS\_MANI\_03035]** { DRAFT }**机器配置的内容**d机器的目的是提供机器特定的配置设置。 c *（ RS\_MANI\_00020 、 RS\_MANI\_00021 、 RS\_MANI\_00022 、 RS\_MANI\_00023 ）*

**[TPS\_MANI\_01208]** { DRAFT }**在Machine范围内定义环境变量**d可以在整个Machine范围内定义环境变量。

为此目的， TagWithOptionalValue在角色Machine中的聚合。环境变量存在。

环境变量的名称应通过属性TagWithOptionalValue指定。 key ，值可以通过TagWithOptionalValue建模。值。

这包括使用空值定义环境变量的能力。为此，属性TagWithOptionalValue 。值应简单地省略。 c *( RS\_MANI\_00022 , RS\_MANI\_00023 )*

请注意聚合机器。 environmentVariable已使用构造型 atpSplitable 定义。这种建模的结果是可以为来自**不同来源的环境变量的定义做出贡献**。

例如，假设有两个部分模型（如清单7.1和7.2所示），它们都将文件夹添加到在自适应平台 ECU 上运行的机器的搜索路径中。

**<机器>**

**<SHORT-NAME>**机器**</SHORT-NAME>**

**<环境变量>**

**<带有可选值的标签>**

**<KEY>**路径**</KEY>**

**<序列偏移>** 10 **</序列偏移>**

**<VALUE>** /usr/application-x/bin **</VALUE>**

**</TAG-WITH-OPTIONAL-VALUE>**

**</环境变量>**

**</机器>**

**清单 7.1： environmentVariable 定义示例（文件 1）**

**<机器>**

**<SHORT-NAME>**机器**</SHORT-NAME>**

**<环境变量>**

**<带有可选值的标签>**

**<KEY>**路径**</KEY>**

**<序列偏移>** 20 **</序列偏移>**

**<VALUE>** /usr/application-y/bin **</VALUE>**

**</TAG-WITH-OPTIONAL-VALUE>**

**</环境变量>**

**</机器>**

**清单 7.2：环境变量定义示例（文件 2）**

合并部分模型中的可拆分元素会产生以下内容。请注意，合并模型仅存在于 AUTOSAR 工具内部。因此，清单7.3**仅用于说明**。

**<机器>**

**<SHORT-NAME>**机器**</SHORT-NAME>**

**<环境变量>**

**<带有可选值的标签>**

**<KEY>**路径**</KEY>**

**<序列偏移>** 10 **</序列偏移>**

**<VALUE>** /usr/application-x/bin **</VALUE>**

**</TAG-WITH-OPTIONAL-VALUE>**

**<带有可选值的标签>**

**<KEY>**路径**</KEY>**

**<序列偏移>** 20 **</序列偏移>**

**<VALUE>** /usr/application-y/bin **</VALUE>**

**</TAG-WITH-OPTIONAL-VALUE>**

**</环境变量>**

**</机器>**

**清单 7.3: environmentVariable 定义示例（合并）**

目标机器配置的生成器可以使用此信息来创建以下环境变量：

PATH=/usr/application-x/bin;/usr/application-y/bin;

**[TPS\_MANI\_01273]** { DRAFT }**支持受信任的平台**d If 属性Machine 。 trustedPlatformExecutableLaunchBehavior设置为与noTrustedPlatformSupport不同的值，然后根据Machine的具体值激活“受信任平台”的功能。受信任的平台可执行启动行为。 c *( RS\_MANI\_00022 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **机器** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest | | | |
| ***笔记*** | 代表自适应 Autosar 软件堆栈的机器。  **标签：**  atp.Status=草稿  atp.recommendedPackage=机器 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpClassifier* , *AtpFeature* , *AtpStructureElement* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 默认应用  暂停 | 进入退出超时 | 0..1 | 聚合 | 这个聚合在给定机器的上下文中定义了一个关于应用程序启动和终止的默认超时。  **标签：** atp.Status=draft |
| 环境变量 | TagWithOptionalValue | \* | 聚合 | 此聚合表示应添加到在封闭机器级别上定义的环境中的环境变量的集合。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=environmentVariable, 环境变量.variationPoint.shortLabel atp.Status=draft |
| 机器设计 | 机器设计 | 1 | 参考 | 参考此机器正在实施的 MachineDesign。  **标签：** atp.Status=draft |
| 模块  实例化 | 自适应模块  实例化 | \* | 聚合 | 在机器上运行的自适应 Autosar 模块实例的配置。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=moduleInstantiation.shortName atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **机器** |  |  |  |
| 处理器 | 处理器 | 1..\* | 聚合 | 这表示封闭机器拥有的处理器的集合。  **标签：** atp.Status=draft |
| 安全的  沟通  部署 | 安全通讯  部署 | \* | 聚合 | 将安全通信协议配置设置部署到加密模块实体。  **刻板印象：** atpSplitable**标签：** atp.Splitkey=secureCommunicationDeployment.short  姓名  atp.Status=草稿 |
| 可信平台可执行文件  启动行为 | 可信平台  可执行启动  行为枚举 | 1 | 属性 | 此属性控制身份验证如何影响每个可执行文件的启动能力的行为。  **标签：** atp.Status=draft |

**表 7.1：机器**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **处理器** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest | | | |
| ***笔记*** | 这表示用于执行 AUTOSAR 自适应平台的处理器  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 核 | 处理器核心 | 1..\* | 聚合 | 这表示封闭处理器拥有的核心集合。  **标签：** atp.Status=draft |

**表 7.2：处理器**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **处理器核心** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest | | | |
| ***笔记*** | 这个元类代表了为执行 AUTOSAR 自适应平台的处理器内核建模的能力。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 核心标识 | 正整数 | 1 | 属性 | 该属性表示分配给特定核心的数值。例如，该值可以用于位掩码。  **标签：** atp.Status=draft |

**表 7.3：处理器核心**

|  |  |
| --- | --- |
| ***枚举*** | **TrustedPlatformExecutableLaunchBehaviorEnum** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest |
| ***笔记*** | 此枚举提供了用于控制身份验证如何影响启动可执行文件的行为的选项。**标签：** atp.Status=draft |
| ***文字*** | ***描述*** |

5

|  |  |
| --- | --- |
| ***枚举*** | **TrustedPlatformExecutableLaunchBehaviorEnum** |
| 监控模式 | 即使相应的身份验证失败，可执行文件也应始终启动  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |
| 不信任平台支持 | 如果机器上没有 TrustedPlatform 支持，则应使用此值  **标签：**  atp.EnumerationLiteralIndex=2 atp.Status=草稿 |
| 严格模式 | 如果相应的身份验证失败，可执行文件将不会启动。  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |

**表 7.4：TrustedPlatformExecutableLaunchBehaviorEnum**

## 7.1 过程到机器的映射

**7.1.1 一般建模方法**

**[TPS\_MANI\_03147]** { DRAFT }**进程到机器的映射**d元类ProcessToMachineMapping提供了将进程映射到机器的能力。 c *( RS\_MANI\_00006 )*

**[constr\_1553]** { DRAFT } **ProcessToMachineMapping的限制**d以下限制适用于ProcessToMachineMapping的使用：

1. Process和Machine的每个组合只能由角色process或machine中的一个ProcessToMachineMapping引用。
2. 每个进程只能由角色进程中的单个ProcessToMachineMapping引用。

### c ()

请注意，[ constr\_1553 ] 并不意味着给定的机器只能由单个ProcessToMachineMapping引用。它只是说一个Process只能映射一次，恰好映射到一台 Machine 。

**[constr\_5004]** { DRAFT }**进程到机器的映射在执行清单中是强制性的**d每个进程都应通过ProcessToMachineMapping映射到一台机器。 c *()*

[ constr\_5004 ] 表示应在执行清单中提供进程分配给机器的正式描述，即使清单将与清单适用的其他工件一起上传到机器。引入正式的ProcessToMachineMapping是因为它在许多情况下对模型的处理很有用。

请注意，根据 AUTOSAR 方法，执行清单是在现有机器清单的基础上创建的，因此始终可以在执行清单中创建到机器的链接。

*ARElement*

*AtpStructureElement*

Machine

trustedPlatformExecutableLaunchBehavior

+

:

TrustedPlatformExecutableLaunchBehaviorEnum

*AbstractExecutionContext*

Process

+

functionClusterAffiliation: String

[0..1]

numberOfRestartAttempts: PositiveInteger

[0..1]

+

[0..1]

preMapping: Boolean

+

*UploadablePackageElement*

ProcessToMachineMappingSet

*Identifiable*

ProcessToMachineMapping

+

[0..1]

persistencyCentralStorageURI: UriString

EnterExitTimeout

:

enterTimeoutValue

+

TimeValue [0..1]

+

:

exitTimeoutValue

TimeValue [0..1]

*Identifiable*

ProcessorCore

coreId: PositiveInteger

+

*Identifiable*

Processor

*ARElement*

ProcessDesignToMachineDesignMapping

processor

+

1..\*

+

core

1..\*

+

shallRunOn

0..\*

design

+

0..1

+

shallNotRunOn

0..\*

+

processToMachineMapping

0..\*

process

+

1

+

defaultApplicationTimeout

0..1

+

machine

0..1

**图 7.2：流程到机器的映射**

**[constr\_10090]** { DRAFT } **ProcessToMachineMapping的存在。 persistencyCentralStorageURI** d属性ProcessToMachineMapping 。如果角色ProcessToMachineMapping中引用的Process ，则应存在persistencyCentralStorageURI 。**在清单完成时，**角色进程中也至少有一个PersistencyPortPrototypeToDeploymentMapping引用了进程。 c *()*

如果在以后的 AUTOSAR 版本中， FunctionalClusterInteractsWithFunctionalClusterMapping的用法是在提供者端使用持久性功能集群开发的，那么必须扩展[ constr\_10090 ] 以涵盖此类情况。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **ProcessToMachineMappingSet** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest | | | |
| ***笔记*** | 这个元类充当收集 ProcessToMachineMappings 的桶。  **标签：**  atp.Status=草稿  atp.recommendedPackage=ProcessToMachineMappings | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable*  *元素*,可*引用*, *UploadablePackageElement* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **ProcessToMachineMappingSet** | |  |  |
| 加工到机器  映射 | 加工到机器  映射 | \* | 聚合 | 这表示封闭 ProcessToMachineMapping Set 的 ProcessToMachine Mappings 的集合。  **标签：** atp.Status=draft |

**表 7.5：ProcessToMachineMappingSet**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **ProcessToMachineMapping** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest | | | |
| ***笔记*** | 这个元类具有将进程与机器关联的能力。这种关系涉及更多属性的定义，例如超时。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 设计 | 工艺设计到  机器设计制图 | 0..1 | 参考 | 此引用表示拥有该引用的 ProcessToMachine Mapping 的设计时表示的标识。  **标签：** atp.Status=draft |
| 机器 | 机器 | 0..1 | 参考 | 此参考在 ProcessToMachineMapping 的上下文中标识机器。  **标签：** atp.Status=draft |
| 非操作系统模块  实例化 | 非操作系统模块  实例化 | 0..1 | 参考 | 这支持进程代表平台模块的可选情况。  **标签：** atp.Status=draft |
| 持久性  中央存储  URI | 字符串 | 0..1 | 属性 | 此属性标识映射进程存储可用存储列表和版本信息的中心位置。  **标签：** atp.Status=draft |
| 过程 | 过程 | 1 | 参考 | 此参考在 ProcessToMachineMapping 的上下文中标识了 Process。  **标签：** atp.Status=draft |
| shallNotRunOn | 处理器核心 | \* | 参考 | 此引用指示不应在其上执行映射进程的核心集合。  **标签：** atp.Status=draft |
| 应运行 | 处理器核心 | \* | 参考 | 此引用指示映射进程应在其上执行的核心集合。  **标签：** atp.Status=draft |

**表 7.6：ProcessToMachineMapping**

**7.1.2 核心亲和力**

**[TPS\_MANI\_03148]** { DRAFT }**核心亲和性描述**d元类ProcessToMachineMapping提供了将进程分配限制到具有两个引用shallRunOn和shallNotRunOn的选定ProcessorCore的能力。 c *( RS\_MANI\_00020 )*

**[constr\_3393]** { DRAFT } **shallRunOn和shallNotRunOn的使用引用**d角色shallRunOn或shallNotRunOn中的ProcessToMachineMapping引用的ProcessorCore应由同一ProcessToMachineMapping在角色machine中引用的Machine聚合。 c *()*

**[constr\_1676]** { DRAFT }**引用的一致性shallRunOn和shall-**

**NotRunOn** d在一个ProcessToMachineMapping的上下文中，角色shallRunOn或shallNotRunOn中引用的所有ProcessorCore应由同一个Processor聚合。 c *()*

如果模型定义给定Process应在一组选定的ProcessorCore上运行，则几乎没有用例（此外）还指定相反的情况，即Process不应在另一组ProcessorCore上运行，反之亦然反之亦然。

换句话说，要么有动机去识别应该在其上运行 Process 的ProcessorCore ，要么有动机去做完全相反的事情并指定不应该运行Process的ProcessorCore 。

constr\_1677 ]的存在提供了动机。

**[constr\_1677]** { DRAFT }**引用的互斥存在shallRunOn和shallNotRunOn** d对于任何给定的ProcessToMachineMapping ，角色中的引用shallRunOn或角色中的引用shallNotRunOn可能

存在。 c *()*

**7.1.3 默认启动和终止超时**

**[TPS\_MANI\_03151]** { DRAFT }**终止超时的默认值**d元类Machine提供了在Machine的上下文中定义应用程序终止超时的默认值的能力，该属性在EnterExitTimeout元类中可用的属性exitTimeoutValue是聚合的由机器角色defaultApplicationTimeout 。 c *( RS\_MANI\_00007 )*

**[constr\_3394]** { DRAFT } **Machine上启动超时的默认值不可配置**d如果EnterExitTimeout由Machine在角色defaultApplicationTimeout中聚合，则不允许使用EnterExitTimeout中可用的属性enterTimeoutValue 。 c *()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **进入退出超时** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest | | | |
| ***笔记*** | 此元类表示指定一对超时的能力，一个用于进入，一个用于退出。  **标签：** atp.Status=draft | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **进入退出超时** |  |  |  |
| 输入超时值 | 时间价值 | 0..1 | 属性 | 该属性以秒为单位表示输入超时的值。  **标签：** atp.Status=draft |
| 退出超时值 | 时间价值 | 0..1 | 属性 | 此属性表示退出超时的值（以秒为单位）。  **标签：** atp.Status=draft |

**表 7.7：EnterExitTimeout**

# 8 执行清单

## 8.1 概述

执行清单的目的是提供将应用程序（正式建模为SwComponentType ）实际部署到 AUTOSAR 自适应平台上所需的信息。

部署信息的一个方面是提供原则上可以作为应用软件代码的一部分提供但会使应用软件代码变得非常受特定使用场景约束的信息。

总体思路是让应用软件代码尽可能地独立于部署场景，以增加应用软件在不同部署场景中被重用的几率。

特别是，使用PortPrototype作为与应用软件“外部”进行通信的一种方式，可以抽象出服务配置的细节（具体的服务实例标识）。就模型而言，应用程序和中间件之间的 API 由PortPrototype表示。

应用程序代码不使用特定的服务实例，而是将PortPrototype作为此信息的符号替换。这个建模方面的细节在第10节中描述。

执行清单定义的顶级元素是Process ，这是因为*AUTOSAR 自适应平台*上的部署单元是一个二进制文件，在运行时会生成一个 POSIX 进程。

**[TPS\_MANI\_01308]** { DRAFT }**流程不是为可重用而设计的**d元类流程的创建**不是**为了在不同的机器上重用它。

Process的定义中，有*一些*重用配置方面的潜力。 stateDependentStartupConfig 。启动配置。 c *( RS\_MANI\_00006 )*

|  |  |  |  |
| --- | --- | --- | --- |
| *AbstractExecutionContext*  Process  [0..1]  functionClusterAffiliation: String  +  +  [0..1]  numberOfRestartAttempts: PositiveInteger  [0..1]  preMapping: Boolean  +  *ARElement*  *AtpClassifier*  Executable  [0..1]  buildType: BuildTypeEnum  +  [0..1]  loggingBehavior: LoggingBehaviorEnum  +  minimumTimerGranularity: TimeValue  +  [0..1]  +  reportingBehavior: ExecutionStateReportingBehaviorEnum  [0..1]  +  version: StrongRevisionLabelString  [0..1]  «atpUriDef»  +  executable  0..1 | |  | | --- | | «枚举»  记录行为枚举 | | 使用记录不使用记录 | |
| |  | | --- | | «枚举»  LogTraceDefaultLogLevelEnum | | 致命错误警告信息调试详细关闭 | |

**图 8.1：元类Executable和Process的关系**

**[TPS\_MANI\_01011]** { DRAFT }**应用程序设计和应用程序部署**之间的连接d应用程序*设计*和应用程序*部署之间的连接*是通过从元类Process到角色executable中的元类Executable的引用来实现的。

通过在这个方向上对参考进行建模，可以保持设计级别独立于部署级别，同时将部署绑定到特定设计。 c *( RS\_MANI\_00006 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **过程** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest | | | |
| ***笔记*** | 此元类提供执行引用的可执行文件所需的信息。  **标签：**  atp.Status=草稿  atp.recommendedPackage=进程 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AbstractExecutionContext* , *AtpClassifier* , *CollectableElement* , *Identifiable* ,  *MultilanguageReferable* , *PackageableElement* , *Referrable* , *UploadablePackageElement* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 设计 | 流程设计 | 0..1 | 参考 | 此引用表示拥有该引用的流程的设计时表示的标识。  **标签：** atp.Status=draft |
| 确定性客户端 | 确定性客户端 | 0..1 | 参考 | 此参考为确定性客户端添加了进一步的执行特性。  **标签：** atp.Status=draft |
| 可执行的 | 可执行文件 | 0..1 | 参考 | 对在进程中执行的可执行文件的引用。  **刻板印象：** atpUriDef  **标签：** atp.Status=draft |
| 功能集群隶属关系 | 细绳 | 0..1 | 属性 | 此属性指定进程隶属于哪个功能集群。**标签：** atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **过程** |  |  |  |
| 数量  重启尝试 | 正整数 | 0..1 | 属性 | 这个属性定义了一个进程在启动失败时应该多久重新启动一次。  numberOfRestartAttempts = "0" 或属性不存在，开始一次  numberOfRestartAttempts = "1"，第二次开始  **标签：** atp.Status=draft |
| 预映射 | 布尔值 | 0..1 | 属性 | 该属性描述可执行文件是否预加载到内存中。  **标签：** atp.Status=draft |
| 进程状态机 | 模式声明组  原型 | 0..1 | 聚合 | 为流程定义的一组流程状态。  **标签：** atp.Status=draft |
| 安全事件 | 安全事件定义 | \* | 参考 | 该参考标识可以由封闭的 SoftwareCluster 报告的 SecurityEvents 的集合。  **刻板印象：** atpSplitable； atpUriDef**标签：**  atp.Splitkey=securityEvent atp.Status=草稿 |
| stateDependent 启动配置 | 状态依赖启动  配置 | \* | 聚合 | 适用的启动配置。  **标签：** atp.Status=draft |

**表 8.1：流程**

**[TPS\_MANI\_01337]** { DRAFT }属性**Process的标准化值。 functionClusterAffiliation** d属性Process的以下值。 functionClusterAffiliation由 AUTOSAR 标准化：

* **STATE\_MANAGEMENT**
* **PLATFORM\_HEALTH\_MANAGEMENT** c *( RS\_MANI\_00006 )*

Process中标准化集合以外的值。函数集群关联。

但是，重要的是，该属性的专有值的制定方式可以避免与未来标准化值的潜在冲突。

可以通过使用特定于公司或特定于项目的前缀、中缀或后缀来实现避免冲突。

在 SWS 执行管理 [ 18 ] 的 [ SWS\_EM\_02109 ] 中更详细地描述了流程的预映射方法。

|  |  |
| --- | --- |
| ***班级*** | ***AbstractExecutionContext*** （抽象） |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest |
| ***笔记*** | 该元类充当在不同级别（例如容器、进程、线程、光纤）上执行代码的实体的基类。**标签：** atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***AbstractExecutionContext*** （抽象） | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpClassifier* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* ,  *PackageableElement* ,*可引用*, *UploadablePackageElement* | | | |
| ***子类*** | 过程 | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 8.2：AbstractExecutionContext**

请注意，元模型，如图8.1所示，支持存在两个或多个引用相同Executable的Process 。

Application1



ManifestOfInstance1.arxml



ManifestOfInstance2.arxml



Executable1

**Executable与两个 ARXML 文件捆绑在一起的示例部署，每个文件都包含一个Process的描述**

这表明特定的可执行文件应该在同一平台上的多个实例中（即以 POSIX 进程的形式）执行。这种情况如图8.2所示

Process的启动条件和启动参数可能会有所不同（为了实现Executable功能的变化）。

因此，有必要允许在每个进程的基础上定义启动配置。

这方面在第8.2节中进行了描述。

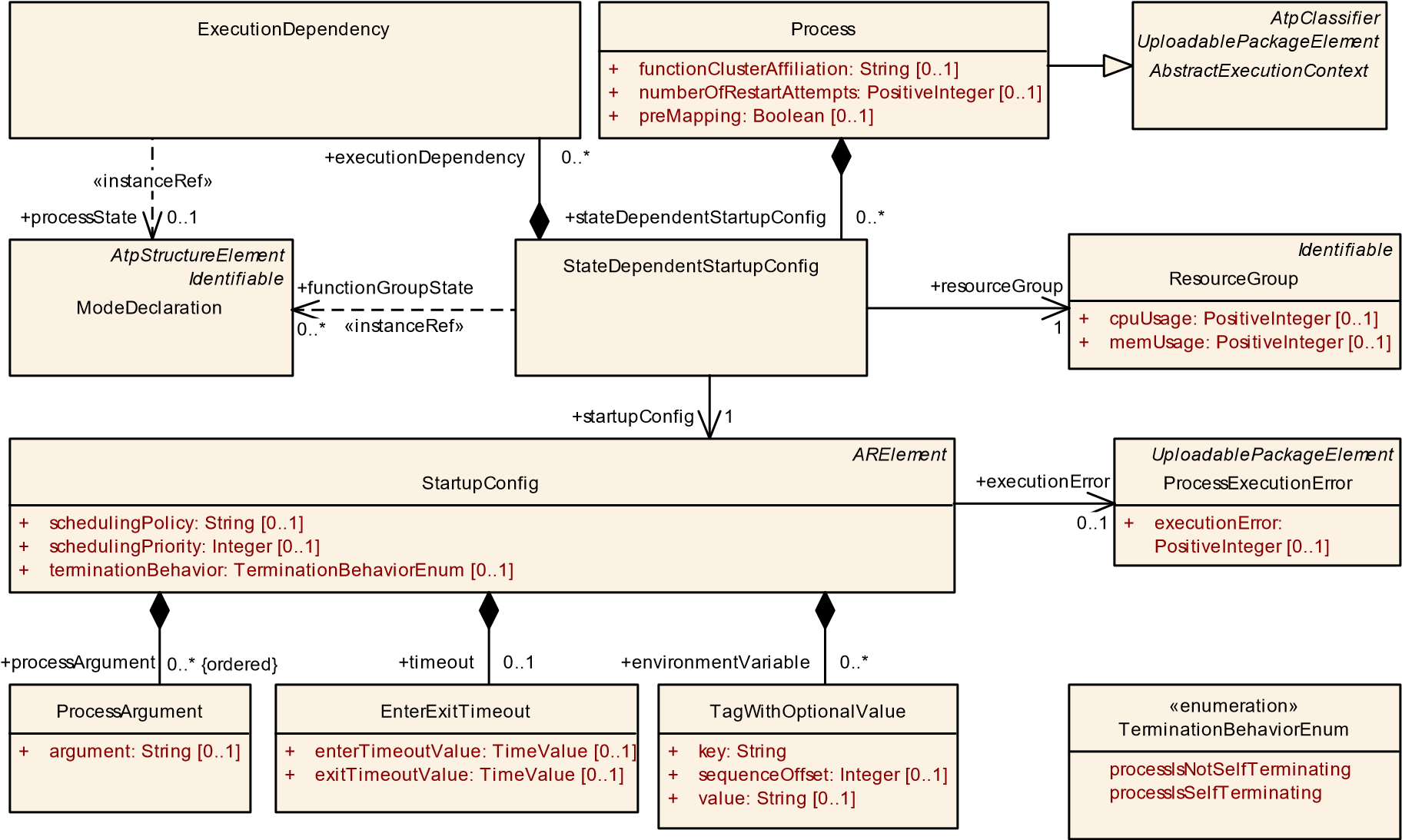
Process中定义的受支持的流程状态。 processStateMachine在 [ 18 ]中有更详细的描述。

## 8.2 启动配置

启动行为的配置是执行清单的重要组成部分。

**[TPS\_MANI\_01012]** { DRAFT }**应用程序启动行为的形式化建模**d应用程序启动行为的形式化建模是通过角色Process中元类StateDependentStartupConfig的聚合来实现的。

stateDependentStartupConfig 。 c *( RS\_MANI\_00007 )*



**图 8.3：流程的内容**

**8.2.1 状态相关启动配置**

**[TPS\_MANI\_01013]** { DRAFT }元类**StateDependentStartupConfig** d的**语义 元类**StateDependentStartupConfig的目的是为特定的ModeDeclaration限定由元类StartupConfig表示的启动配置。

换句话说，如果控制启动的状态机处于由角色StateDependentStartupConfig中引用的ModeDeclaration表示的状态，则意图表示StartupConfig适用。函数组状态。 c *( RS\_MANI\_00007 )*

作为从StateDependentStartupConfig到ModeDeclaration的引用的结果，执行清单是为部署二进制文件和清单的特定机器定义的。

**[constr\_3423]** { DRAFT }**进程的StateDependentStartupConfig应引用functionGroupState** d进程的每个StateDependentStartupConfig应引用角色functionGroupState中的至少一个ModeDeclaration 。 c *()*

但是，在一个进程的上下文中对功能组状态的引用应仅指**同一功能组的功能组状态**。这方面由 [ constr\_1688 ] 形式化。

**[constr\_1688]** { DRAFT } **StateDependentStartupConfig应仅指同一功能组的功能组状态**d对于角色Process中聚合的所有StateDependentStartupConfig 。 stateDependentStartupConfig ，角色functionGroupState中对ModeDeclaration的引用应仅引用由相同ModeDeclarationGroup在相同ModeDeclarationGroupPrototype （表示实际功能组）的上下文中聚合的ModeDeclaration 。 c *()*

有必要指定约束 [ constr\_3396 ] 来规范在一个Process的上下文中引用相同ModeDeclaration的StateDependentStartupConfig的数量，因为生成的启动配置将是模棱两可的。

**[constr\_3396]** { DRAFT }**进程数。 stateDependentStartupConfig引用相同的functionGroupState** d在给定Process的上下文中，没有两个StateDependentStartupConfig应该引用角色functionGroupState中的相同ModeDeclaration 。 c *()*

**[TPS\_MANI\_01046]** { DRAFT } **StateDependentStartupConfig的语义。 functionGroupState** d角色StateDependentStartupConfig中引用的ModeDeclaration 。 functionGroupState应以这样的方式考虑，即如果**任何**引用的ModeDeclaration处于活动状态，则StateDependentStartupConfig应用。

换句话说， ModeDeclaration s 是 or-ed 以确定是否

StateDependentStartupConfig适用。 c *( RS\_MANI\_00007 )*

**[constr\_3424]** { DRAFT } **StateDependentStartupConfig不得引用functionGroupState Off** d StateDependentStartupConfig不得引用在角色functionGroupState中具有短名称 Off的ModeDeclaration 。请注意，Off ModeDeclaration是 [ TPS\_MANI\_03195 ]定义的功能组中的一种特殊状态。 c *()*

**[constr\_1618]** { DRAFT }**关闭能力**d在一台 Machine的上下文中，至少一个Process应具有stateDependentStartupConfig 。 functionGroupState具有shortNameShutdown 。 c *()*

**[constr\_1619]** { DRAFT }**重启能力**d在一台 Machine的上下文中，至少一个Process应具有stateDependentStartupConfig 。 functionGroupState具有shortNameRestart 。 c *()*

**[TPS\_MANI\_01209]** { DRAFT }**进程范围内环境变量的定义**d可以在任何给定Process的范围内定义环境变量。

为此目的，在角色StartupConfig中聚合TagWithOptionalValue 。环境变量存在。

环境变量的名称应通过属性TagWithOptionalValue指定。 key ，值可以通过TagWithOptionalValue建模。值。

这包括使用空值定义环境变量的能力。为此，属性TagWithOptionalValue 。值应简单地省略。 c *( RS\_MANI\_00007 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **StateDependentStartupConfig** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest | | | |
| ***笔记*** | 这个元类根据机器状态的集合定义进程的启动配置。  **标签：** atp.Status=draft | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 执行  依赖 | 执行依赖 | \* | 聚合 | 该属性定义了所有通过 ExecutionDependency 引用的进程都应该被启动并且在引用进程启动之前达到某个 ProcessState。  **标签：** atp.Status=draft |
| 功能组状态 | 模式声明 | \* | 参考 | 这表示适用的函数GroupMode。  **标签：** atp.Status=draft  **InstanceRef 实现者：** FunctionGroupStateIn  FunctionGroupSetInstanceRef |
| 资源  消耗 | 资源消耗 | 0..1 | 聚合 | 这种聚合提供了在每个进程启动配置的基础上定义资源消耗边界的能力。  **标签：** atp.Status=draft |
| 资源组 | 资源组 | 1 | 参考 | 引用适用的资源组。  **标签：** atp.Status=draft |
| 启动配置 | 启动配置 | 1 | 参考 | 使用启动参数引用可重用的启动配置。  **标签：** atp.Status=draft |

**表 8.3：StateDependentStartupConfig**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **启动配置** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest | | | |
| ***笔记*** | 此元类表示进程的可重用启动配置。  **标签：**  atp.Status=草稿  atp.recommendedPackage=启动配置 | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable Element* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 环境变量 | TagWithOptionalValue | \* | 聚合 | 此聚合表示在启动之前应添加到相应进程环境中的环境变量的集合。  **标签：** atp.Status=draft |
| 执行错误 | 进程执行错误 | 0..1 | 参考 | 此参考用于识别适用的执行错误  **标签：** atp.Status=draft |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **启动配置** |  |  |  |
| 过程参数  （订购） | 过程参数 | \* | 聚合 | 此聚合表示适用于封闭 StartupConfig 的命令行参数的集合。  **标签：** atp.Status=draft |
| 调度策略 | 细绳 | 0..1 | 属性 | 该属性表示为应用程序的初始线程定义调度策略的能力。  **标签：** atp.Status=draft |
| 调度优先级 | 整数 | 0..1 | 属性 | 这是应用程序本身请求的调度优先级。  **标签：** atp.Status=draft |
| 终止行为 | 终止行为  枚举 | 0..1 | 属性 | 该属性定义进程的终止行为。  **标签：** atp.Status=draft |
| 暂停 | 进入退出超时 | 0..1 | 聚合 | 此聚合可用于根据 StartupConfig 指定启动和终止进程的超时时间。  **标签：** atp.Status=draft |

**表 8.4：启动配置**

**[TPS\_MANI\_01277]** { DRAFT }**为进程的StartupConfig定义启动超时**d元类StartupConfig提供了通过由元类EnterExitTimeout聚合的属性enterTimeoutValue为进程定义启动超时的能力由角色timeout中的StartupConfig聚合。 c *( RS\_MANI\_00007 )*

**[TPS\_MANI\_01278]** { DRAFT }**为进程的StartupConfig定义终止超时**d元类StartupConfig提供了通过属性exitTimeoutValue为进程定义终止超时的能力，该属性由聚合的元类EnterExitTimeout聚合通过StartupConfig中的角色timeout 。 c *( RS\_MANI\_00007 )*

**8.2.2 调度**

**[TPS\_MANI\_01061 ]** { DRAFT }调度**要求** d属性

启动配置。 schedulePolicy和StartupConfig 。 schedulePriority 对由启动相应的Executable创建的进程的主线程的调度提出要求。 c *( RS\_MANI\_00007 )*

**[TPS\_MANI\_01328]** { DRAFT }属性**StartupConfig的标准化值。 schedulePolicy** d以下值是为属性StartupConfig标准化的。调度策略：

* **SCHED\_RR**
* **SCHED\_FIFO**
* **SCHED\_OTHER**

c *( RS\_MANI\_00007 )*

可以为属性StartupConfig使用自定义的非标准化值。 schedulePolicy但此选项附带使用一个值的义务，该值保证不会与标准化值集合的未来可能扩展发生冲突。

**[TPS\_MANI\_01188]** { DRAFT }属性**schedulingPriority的语义**d属性StartupConfig的值。 schedulePriority应被解释为，较高的值表示较高的调度优先级。 c *( RS\_MANI\_00007 )*

**[constr\_1692]** { DRAFT } **schedulingPriority** d的**值**属性的值

启动配置。 schedulePriority应设置为正整数值。 c *()*

**8.2.3 过程参数**

在 SWS 执行清单 [ 18 ]中找到有关ProcessArgument解释的更多信息。

**[constr\_1769]** { DRAFT } **ProcessArgument的存在。参数**d对于每个ProcessArgument ，属性参数应**在清单创建完成时存在**。 c *()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **过程参数** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest | | | |
| ***笔记*** | 该元类能够定义命令行参数以供 Main 函数处理。  **标签：** atp.Status=draft | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 争论 | 细绳 | 0..1 | 属性 | 这表示要由可执行软件处理的一个命令行参数。  **标签：** atp.Status=draft |

**表 8.5: ProcessArgument**

**8.2.4 与资源组的关联**

元类StateDependentStartupConfig还支持与资源组的关系规范。

**[TPS\_MANI\_01017]** { DRAFT }**启动配置与资源组**d的关系 通过机器的OsModuleInstantiation中的元类ResourceGroup可以对资源组进行建模，并且从关联支持将进程分配给ResourceGroup StateDependentStartupConfig到ResourceGroup中的角色resourceGroup 。 c *( RS\_MANI\_00007 )*

Process

[0..1]

functionClusterAffiliation: String

+

[0..1]

+

numberOfRestartAttempts: PositiveInteger

+

preMapping: Boolean

[0..1]

*Identifiable*

ResourceGroup

[0..1]

cpuUsage: PositiveInteger

+

[0..1]

memUsage: PositiveInteger

+

StateDependentStartupConfig

*AtpClassifier*

*UploadablePackageElement*

*AbstractExecutionContext*

+

stateDependentStartupConfig

0..\*

+

resourceGroup

1

**图 8.4： Process与ResourceGroup关联的建模**

**[constr\_3413]** { DRAFT }**进程的StateDependentStartupConfig恰好映射到一个ResourceGroup** d进程的每个StateDependentStartupConfig应分配给机器清单中定义的恰好一个ResourceGroup 。 c *()*

**8.2.5 执行依赖**

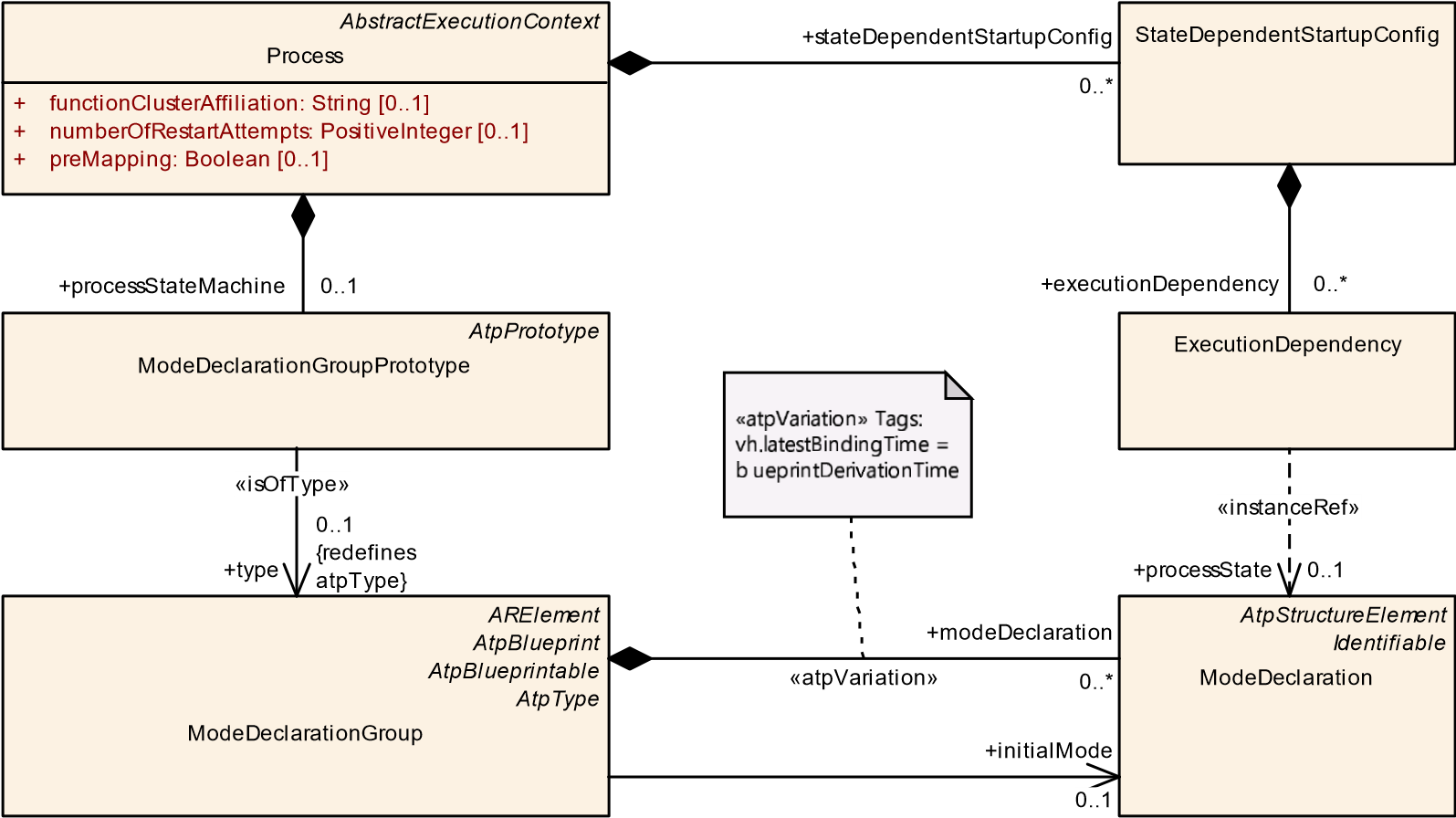
执行依赖的建模使两个Process通过定义ExecutionDependency相互关联。

但是由于定义执行依赖的引用被建模为instanceRef，所以需要从instanceRef中的上下文引用中提取出被引用的Process 。

一旦两个Process被识别，启动依赖的有效性就需要它们引用相同的函数组。

**[TPS\_MANI\_01041]** { DRAFT }**启动配置支持定义启动序列依赖**d启动配置的建模还支持定义启动序列依赖，由StateDependentStartupConfig在角色executionDependency中聚合的元类ExecutionDependency形式化。

ExecutionDependency允许在启动通过StateDependentStartupConfig聚合ExecutionDependency的进程之前定义对需要处于特定进程状态的进程的依赖关系。 c *( RS\_MANI\_00007 )*



**图 8.5： Process与另一个拥有的ModeDeclaration相关的建模**

**过程**

**[constr\_1689]** { DRAFT }**建模不同Process es之间的启动依赖关系**d属性Process的存在。 stateDependentStartupConfig 。 executionDependency仅在stateDependentStartupConfig的所有者时有效。 executionDependency （换句话说：引用Process ）和引用 stateDependentStartupConfig中角色contextModeDeclarationGroupPrototype中引用的ModeDeclarationGroupPrototype的所有者。执行依赖。 processState （即引用的Process ）指的是相同的功能组状态，形式化为

模式声明。 c *()*

图8.6提供了 [ constr\_1689 ] 的示例性解释。在此示例中，进程“B”（从[ constr\_1689 ] 开始的引用进程）定义了进程“ A”的executionDependency 。

此executionDependency仅在进程“A”和进程“B”都聚合一个StateDependentStartupConfig 时有效，该配置引用函数组“FG”中的相同函数组状态“MD”。

ExecutionDependency （特别是contextModeDeclarationGroupPrototype ）和 instanceRef可以找到进程“A” 从ExecutionDependency到进程状态“PS”。

的**所有者**是进程“B”，如果“B”指的是功能组“FG”中的功能组状态“MD”，如果“A”指的是“FG”中的功能组状态“MD”，则约束 [ constr\_1689 ] 得到满足。

Process "B"

Process "A"

StateDependentStartup

Config

ExecutionDependency

ProcessState "PS"

ModeDeclarationGroup

(

Prototype)

FunctionGroup "FG"

ModeDeclarationGroup

(

Prototype)

StateDependentStartup

Config

context

context

context

ModeDeclaration "MD"

ModeDeclarationGroup

type

target

target

**图 8.6：解释从一个进程到另一个进程的依赖关系**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **执行依赖** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest | | | |
| ***笔记*** | 此元素定义了一个 ProcessState，在启动聚合 ExecutionDependency 元素的进程之前，依赖进程需要处于其中。  **标签：** atp.Status=draft | | | |
| ***根据*** | *AR对象* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 进程状态 | 模式声明 | 0..1 | 参考 | 这表示表示 ProcessState 的适用模式声明。  **标签：** atp.Status=draft  **InstanceRef 实现者：** ModeInProcessInstance Ref |

**表 8.6：执行依赖**

**[constr\_1606]** { DRAFT }**具有相互ExecutionDependency s** d A的**进程**

过程。 stateDependentStartupConfig 。 executionDependency不应引用第二个Process拥有的任何ModeDeclaration ，而后者又通过stateDependentStartupConfig 引用。对第一个Process拥有的任何ModeDeclaration的executionDependency 。 c *()*

**8.2.6 进程到功能组状态的分配**

有些用例需要启动和终止各个进程组。这在 AUTOSAR 中由将进程分组在一起的功能组支持。

一个功能组可以具有多个功能组状态，例如正在运行、空闲、终止。 Process的StateDependentStartupConfig可以分配给功能组状态，然后 Process 的启动将取决于此分配。

功能组的建模及其功能组状态在第 8.4 节中有更详细的描述。功能组的使用在[ 18 ]中有更详细的描述。

**[TPS\_MANI\_03152]** { DRAFT }将**StateDependentStartupConfig分配给功能组状态**d StateDependentStartupConfig分配给具有functionGroupState引用的功能组状态。 c *( RS\_MANI\_00041 )*

**8.2.7 资源消耗边界**

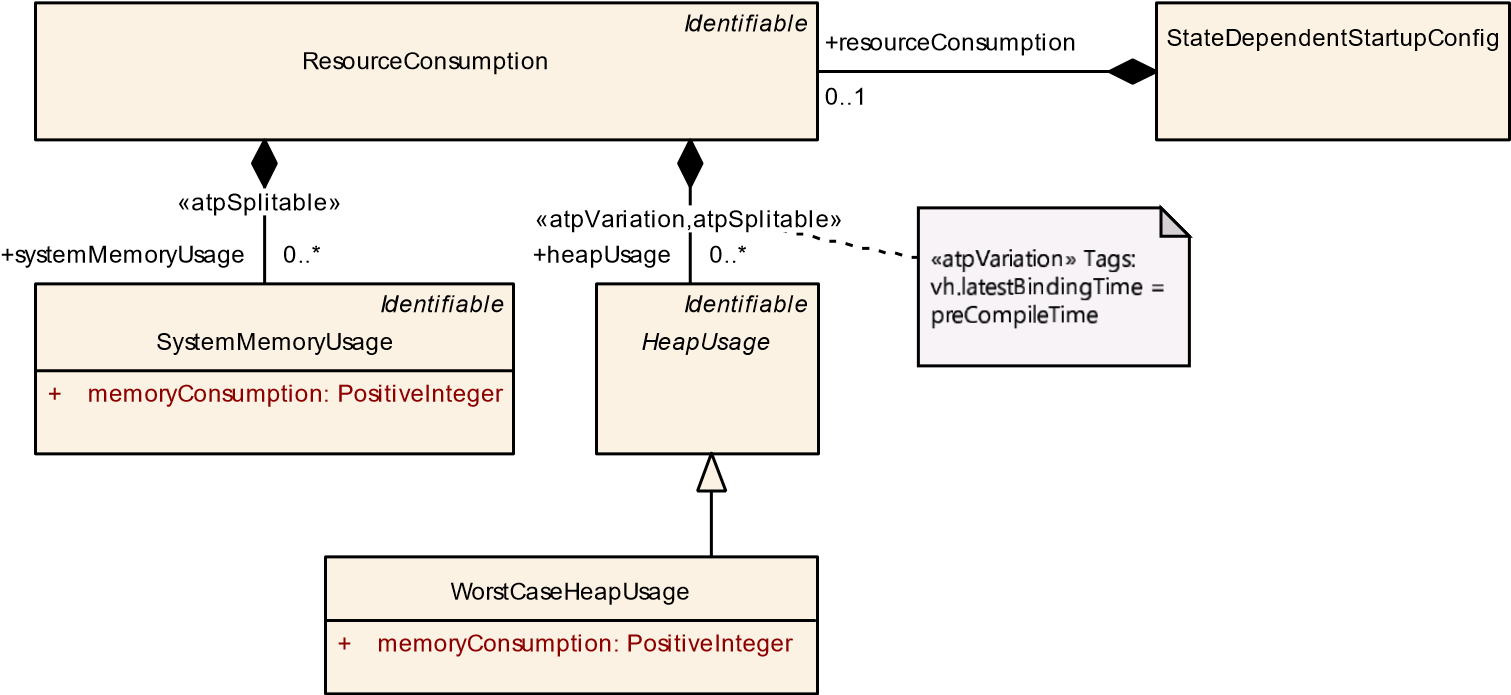
**[TPS\_MANI\_01269]** { DRAFT }**资源消耗边界的规范**d可以指定资源消耗的边界，特别是在给定的进程启动配置的系统内存和堆内存消耗方面：

* 堆使用的形式化由元类HeapUsage表示，通过StateDependentStartupConfig的元类ResourceConsumption聚合。堆使用的实际值是根据所有聚合ResourceConsumption的总和计算的。堆使用。
* 系统（即内核空间）内存使用的形式化由元类SystemMemoryUsage表示，通过StateDependentStartupConfig的元类ResourceConsumption聚合。系统内存使用的实际值是根据所有聚合ResourceConsumption的总和计算得出的。系统内存使用。

c *( RS\_MANI\_00020 )*

请注意为单个Process定义资源消耗边界的能力与将Process与具有定义资源消耗边界的能力的ResourceGroup相关联的能力之间的区别，尽管是在更粗粒度的级别上。

与此相反， StateDependentStartupConfig 。 resourceConsumption允许细粒度的定义，甚至可以观察不同启动配置的资源消耗差异。



**图 8.7：给定流程的资源消耗边界建模**

**[constr\_1707]** { DRAFT }在**StateDependentStartupConfig上下文中HeapUsage的合格子类。 resourceConsumption** d StateDependentStartupConfig的定义。资源消耗。 heapUsage只能通过具体的子类WorstCaseHeapUsage来完成。 c *()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **资源消耗** | | | |
| ***包裹*** | M2::AUTOSARTemplates::CommonStructure::ResourceConsumption | | | |
| ***笔记*** | 对软件的一种实现所消耗资源的描述。 | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 堆使用 | 堆使用 | \* | 聚合 | 此实现分配的堆内存的集合。  **刻板印象：** atpSplitable； atpVariation**标签：**  atp.Splitkey=heapUsage.shortName, heap Usage.variationPoint.shortLabel vh.latestBindingTime=preCompileTime |
| 系统内存使用 | 系统内存使用 | \* | 聚合 | 所有者分配的系统内存的集合。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=systemMemoryUsage.shortName atp.Status=草稿 |

**表 8.7：资源消耗**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***堆使用***（抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::CommonStructure::ResourceConsumption::HeapUsage | | | |
| ***笔记*** | 描述 SW 组件的堆内存使用情况。 | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***子类*** | MeasuredHeapUsage, RoughEstimateHeapUsage, WorstCaseHeapUsage | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 8.8：堆使用**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **最坏情况堆使用** | | | |
| ***包裹*** | M2::AUTOSARTemplates::CommonStructure::ResourceConsumption::HeapUsage | | | |
| ***笔记*** | 提供正式的最坏情况堆使用情况。 | | | |
| ***根据*** | *ARObject* , *HeapUsage* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 记忆  消耗 | 正整数 | 1 | 属性 | 最坏情况下的堆消耗。单位：字节。 |

**表 8.9：WorstCaseHeapUsage**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **系统内存使用** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest | | | |
| ***笔记*** | 描述系统内存（即内核空间）的消耗。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 记忆  消耗 | 正整数 | 1 | 属性 | 提供正式的最坏情况系统用法。单位是字节。  **标签：** atp.Status=draft |

**表 8.10: SystemMemoryUsage**

**8.2.8 错误和终止行为**

**[TPS\_MANI\_01334]** {草稿} **StartupConfig的语义。终止行为**d属性StartupConfig 。 TerminationBehavior根据是否（或不）引用角色stateDependentStartupConfig中的封闭StartupConfig的Process来定义Process的终止行为。 startupConfig配置为自终止。 c *( RS\_MANI\_00007 )*

|  |  |
| --- | --- |
| ***枚举*** | **终止行为枚举** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest |
| ***笔记*** | 此枚举提供了用于控制进程如何终止的选项。  **标签：** atp.Status=draft |
| ***文字*** | ***描述*** |
| processIsNotSelf 终止 | 该流程仅应执行管理的请求而终止。  **标签：**  atp.EnumerationLiteralIndex=0 atp.Status=草稿 |
| processIsSelf 终止 | 进程可以在没有执行管理请求的情况下终止。  **标签：**  atp.EnumerationLiteralIndex=1 atp.Status=草稿 |

**表 8.11：终止行为枚举**

**[constr\_10007]** { DRAFT }**存在ProcessExecutionError 。 executionError** d对于每个ProcessExecutionError ，属性executionError应**在清单创建完成时存在**。 c *()*

**[constr\_10008]** { DRAFT } **ProcessExecutionError的值。 executionError** d属性ProcessExecutionError的值。 executionError至少应设置为 1（或更高）。 c *()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **进程执行错误** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest | | | |
| ***笔记*** | 该元类能够描述执行错误的值及其语义文档。  **标签：**  atp.Status=草稿  atp.recommendedPackage=ProcessExecutionErrors | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable*  *元素*,可*引用*, *UploadablePackageElement* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 执行错误 | 正整数 | 0..1 | 属性 | 如果进程意外终止或违反其监督，此属性定义执行管理和平台健康管理向状态管理报告的数值。它应为错误恢复提供进一步的错误信息。  **标签：** atp.Status=draft |

**表 8.12：ProcessExecutionError**

## 8.3 确定性客户端

如第3.19.1节所述，有一个用例支持*AUTOSAR 自适应平台*上所谓的确定性客户端的概念。确定性客户端的概念背景在 SWS 执行管理 [ 18 ] 中进行了解释。

对这一概念的支持包括两个方面。*设计方面*已经在第 3.19.1 节中解释过，而部署方面*则*在本章中讨论。

**[TPS\_MANI\_01203]** { DRAFT } **DeterministicClient的语义**d引用Process的存在。 deterministicClient意味着封闭的Process实现了 Deterministic Client 的概念。

可以从角色Process中引用的ProcessDesign获得有关 Deterministic Client 配置的更多信息。设计。 c *( RS\_MANI\_00050 )*

图8.8显示了对 Deterministic Client 支持的详细信息。

*UploadablePackageElement*

DeterministicClient

+

[0..1]

cycleTimeValue: TimeValue

[0..1]

numberOfWorkers: PositiveInteger

+

*AbstractExecutionContext*

Process

[0..1]

functionClusterAffiliation: String

+

[0..1]

numberOfRestartAttempts: PositiveInteger

+

+

[0..1]

preMapping: Boolean

*ARElement*

ProcessDesign

*Identifiable*

DeterministicClientResourceNeeds

+

hardwarePlatform: String

[0..1]

DeterministicClientResource

[0..1]

numberOfInstructions: NormalizedInstruction

+

[0..1]

sequentialInstructionsBegin: NormalizedInstruction

+

[0..1]

sequentialInstructionsEnd: NormalizedInstruction

+

[0..1]

speedup: Float

+

+

runResource

0..1

design

+

0..1

+

initResource

0..1

+

deterministicClient

0..1

+

deterministicClientResourceNeeds

0..\*

**图 8.8：部署中对 Deterministic Client 的支持建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **确定性客户端** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest | | | |
| ***笔记*** | 元类 DeterministicClient 提供了支持一个或多个进程的确定性执行的能力，该进程具有 DeterministicClient 库函数的特定配置参数。  **标签：**  atp.Status=草稿  atp.recommendedPackage=DeterministicClients | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable*  *元素*,可*引用*, *UploadablePackageElement* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 周期时间值 | 时间价值 | 0..1 | 属性 | 此属性表示执行 DeterministicClient 激活周期的周期时间。  **标签：** atp.Status=draft |
| 工人数 | 正整数 | 0..1 | 属性 | 处理数据集的独立工作人员的数量。工作池的大小应根据处理器内核或内存等资源的可用性来决定。  **标签：** atp.Status=draft |

**表 8.13：确定性客户端**

## 8.4 功能组

**8.4.1 功能组语义**

具有功能组状态的功能组单独控制功能一致的应用程序进程组。如果StateDependentStartupConfig使用functionGroupState引用来引用功能组状态，则进程状态可能取决于功能组中定义的模式。

功能组的使用在[ 18 ]中有更详细的描述。 **[TPS\_MANI\_03145]** { DRAFT }**功能组的描述**d通过定义一个

ModeDeclarationGroupPrototype聚合在角色FunctionGroupSet中。 functionGroup可以定义一个具有短名称和一组模式（状态）的功能组。

ModeDeclarationGroupPrototype指向角色类型中的可重用ModeDeclarationGroup ，其中包含不同的模式，如ModeDeclaration和指定的initialMode 。 c *( RS\_MANI\_00041 )*

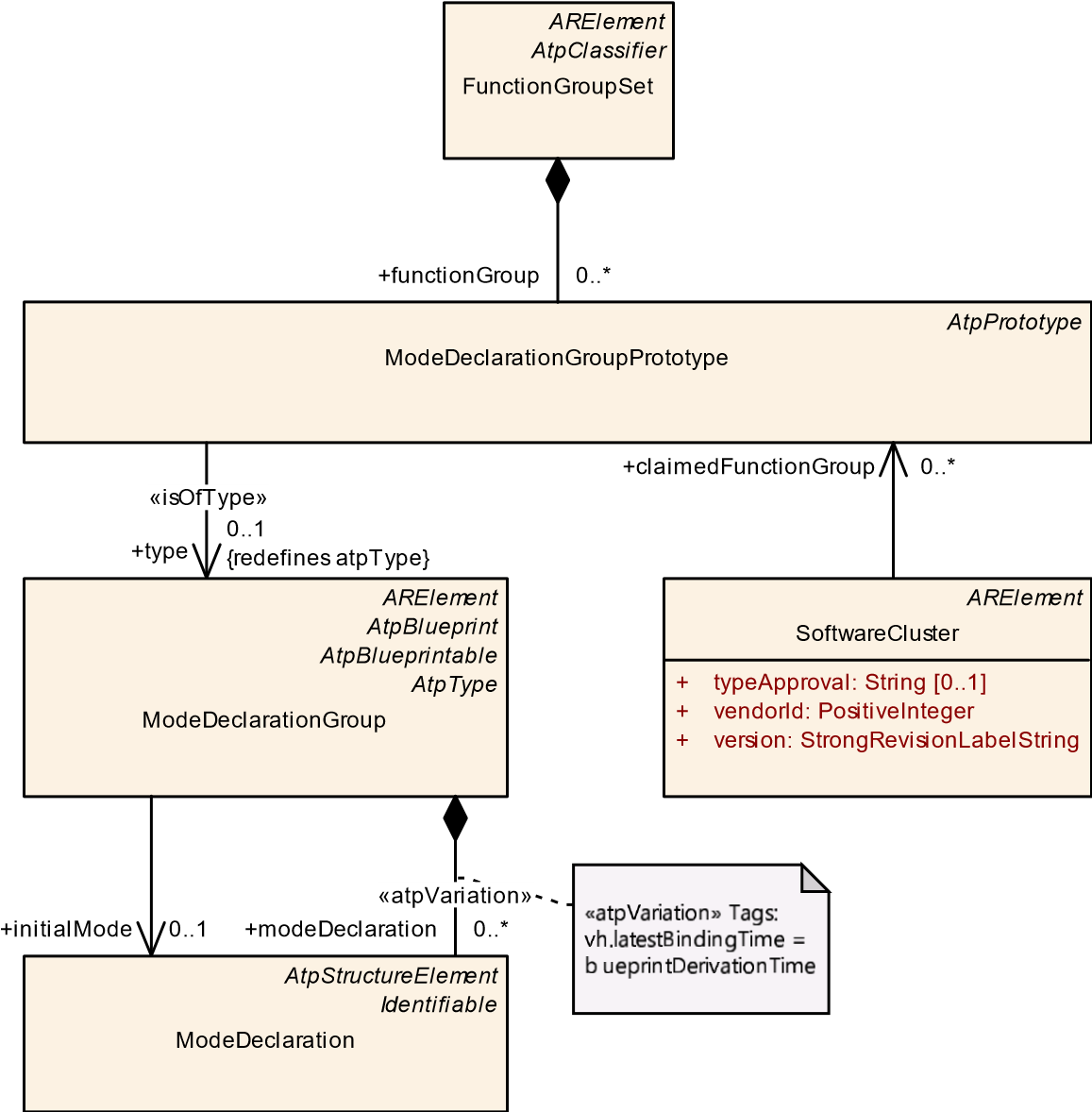
**[TPS\_MANI\_03194]** { DRAFT }**功能组状态**d功能组状态由ModeDeclarationGroup中的ModeDeclaration描述，ModeDeclarationGroupPrototype在角色类型中引用该ModeDeclarationGroupPrototype聚合为functionGroup的FunctionGroupSet 。功能组状态由其shortName标识。 c *( RS\_MANI\_00041 )*

TPS\_MANI\_03145 ] 和 [ TPS\_MANI\_03194 ] 中描述的建模如图8.9所示。

**[TPS\_MANI\_03195]** { DRAFT }功能组d**中的关闭状态**每个功能组应定义一个带有短名称关闭的ModeDeclaration 。此ModeDeclaration还应在角色initialMode中由ModeDeclarationGroup引用，该模式键入相应的functionGroup 。 c *( RS\_MANI\_00041 )*

**[constr\_1786]** { DRAFT }**根据SoftwareCluster** d**使用functionGroup的限制**每个functionGroup只能在声明的角色中引用 -

FunctionGroup最多由**一个**SoftwareCluster组成。 c *()*



**图 8.9：功能组配置**

**[constr\_1787]** { DRAFT }**在 a 的上下文中限制使用函数组**

**SoftwareCluster** d角色所包含的SoftwareCluster引用的所有Process仅应聚合StateDependentStartupConfig ，其中引用functionGroupState指的是ModeDeclarationGroupPrototype （作为上下文），该模型也由角色中的同一SoftwareCluster引用。 c *()*

SoftwareCluster的描述可以在第14节中找到。

**[constr\_10023]** { DRAFT }任何**functionGroup** d**的强制性内容由**角色functionGroup中的FunctionGroupSet聚合的所有ModeDeclarationGroupPrototype应指包含一个ModeDeclaration的ModeDeclarationGroup ，该ModeDeclaration具有短名称**Verify** 。 c *()*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **函数组集** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::General | | | |
| ***笔记*** | 这个元类提供了创建任意函数组集合的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=功能组集 | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpClassifier* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 功能组 | 模式声明组  原型 | \* | 聚合 | 此聚合表示功能组的集合。  **标签：** atp.Status=draft |

**表 8.14：函数组集**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **模式声明组原型** | | | |
| ***包裹*** | M2::AUTOSARTemplates::CommonStructure::ModeDeclaration | | | |
| ***笔记*** | ModeDeclarationGroupPrototype 指定一组在给定上下文中提供或要求的模式 (ModeDeclarationGroup)。 | | | |
| ***根据*** | *ARObject* , *AtpFeature* , *AtpPrototype* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 类型 | 模式声明组 | 0..1 | 特雷夫 | “模式声明的集合”（= ModeDeclaration  组）由组件支持  **刻板印象：** isOfType |

**表 8.15：ModeDeclarationGroupPrototype**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **模式声明组** | | | |
| ***包裹*** | M2::AUTOSARTemplates::CommonStructure::ModeDeclaration | | | |
| ***笔记*** | 模式声明的集合。此外，初始模式是明确标识的。  **标签：** atp.recommendedPackage=ModeDeclarationGroups | | | |
| ***根据*** | *ARElement* , *ARObject* , *AtpBlueprint* , *AtpBlueprintable* , *AtpClassifier* , *AtpType* , *CollectableElement* ,  *识别*，*多语言可引用*，可打包元素，*可引用* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |

5

4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **模式声明组** |  |  |  |
| 初始模式 | 模式声明 | 0..1 | 参考 | ModeDeclarationGroup 的初始模式。此模式在发生任何模式切换之前处于活动状态。 |
| 模式  宣言 | 模式声明 | \* | 聚合 | 在此 ModeDeclaration Group 中收集的 ModeDeclarations。  **刻板印象：** atpVariation  **标签：** vh.latestBindingTime=blueprintDerivationTime |
| 模式转换 | 模式转换 | \* | 聚合 | 这代表了可用的 ModeTransitions  模式声明组 |

**表 8.16：ModeDeclarationGroup**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **模式声明** | | | |
| ***包裹*** | M2::AUTOSARTemplates::CommonStructure::ModeDeclaration | | | |
| ***笔记*** | 一种模式的声明。特定模式的名称和语义未在元模型中定义。 | | | |
| ***根据*** | *ARObject* , *AtpClassifier* , *AtpFeature* , *AtpStructureElement* , *Identifiable* , *MultilanguageReferable* ,  *可推荐* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 8.17：模式声明**

**8.4.2 机器功能组**

请注意，类别PLATFORM\_CORE 的一个SoftwareCluster 声明的一个functionGroup扮演“机器功能组”的角色。

此functionGroup需要有一个专用的shortName并且还需要定义一组最小但可扩展的ModeDeclaration ，它们也具有标准化的shortName 。

**[TPS\_MANI\_01330]** { DRAFT }机器功能**组**d的定义应该存在一个具有短名称“ MachineFG ”并且由ModeDeclarationGroup类型的功能组，该ModeDeclarationGroup至少定义以下具有短名称的ModeDeclaration列表

* **关闭**,
* **验证**，
* **启动**，
* **关机**，和
* **重启**。 c *( RS\_MANI\_00041 )*请注意，进程的启动可能取决于在类别PLATFORM\_CORE 的SoftwareCluster 上下文中定义的模式。 StateDependentStartupConfig在第8.2章中描述。

**[constr\_1789]** { DRAFT }**机器功能组的范围**d表示机器功能组的 functionGroup（参见 [ TPS\_MANI\_01330 ] ）只能在角色claimFunctionGroup中由PLATFORM\_CORE 类别的SoftwareCluster引用。 c *()*

## 8.5 安全事件的报告

Process的上下文中报告所谓的安全事件（由元类SecurityEventDefinition形式化） 。

这种方法适用于应用程序级软件以及功能集群，但执行管理器除外（因为执行管理器本身并未建模为流程）。

*IdsCommonElement*

SecurityEventDefinition

+

id: PositiveInteger

[0..1]

*AbstractExecutionContext*

Process

+

functionClusterAffiliation: String

[0..1]

+

numberOfRestartAttempts: PositiveInteger

[0..1]

+

preMapping: Boolean

[0..1]

«atpSplitable,atpUriDef»

+

securityEvent

0..\*

**图 8.10：支持在部署级别报告SecurityEventDefinition的建模**

21 ]中找到有关安全事件的语义和使用的更多信息。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **安全事件定义** | | | |
| ***包裹*** | M2::AUTOSARTemplates::SecurityExtractTemplate | | | |
| ***笔记*** | 该元类将安全相关事件定义为入侵检测系统的一部分。  **标签：**  atp.Status=草稿  atp.recommendedPackage=SecurityEventDefinitions | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *IdsCommonElement* , *MultilanguageReferable* , *PackageableElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 事件符号名称 | 符号道具 | 0..1 | 聚合 | 此聚合可选地为 SecurityEventDefinition 定义一个替代事件名称，以防短名称发生冲突。  **刻板印象：** atpSplitable**标签：**  atp.Splitkey=eventSymbolName.shortName atp.Status=草稿 |
| ID | 正整数 | 0..1 | 属性 | 该属性表示已定义安全事件的数字标识。该标识在 IDS 范围内应是唯一的。  **标签：** atp.Status=draft |

**表 8.18：安全事件定义**

# 9 平台模块开发

平台模块模型及其实例化有两个主要用例：

* 提供专用属性来配置平台模块
* 将模块可执行文件的潜在开始定义为进程。

这两个用例组合在一种建模方法中：机器。模块-

实例化，它收集了AdaptiveModuleInstantiation的子类。这种建模方法归结为本章中发现的各种平台模块模型。

OsModuleInstantiation定义了要为 Os 配置的几个属性，但是OsModuleInstantiation是唯一无法将其映射到Process模型元素的AdaptiveModuleInstantiation 。

当然，无论如何都会有进程在机器上运行操作系统，但是，这些进程没有被建模。

NonOsModuleInstantiation的专用子类的场景。在这里，每个子类单独提供特定属性，例如

NmInstantiation或LogAndTraceInstantiation 。

这些NonOsModuleInstantiation独立于启动行为实现。如果堆栈实现决定在专用Process中实现特定功能集群，则特定NonOsModuleInstantiation也将成为ProcessToMachineMapping的一部分。

但是，如果堆栈实现决定将特定功能集群实现为库（或使功能成为另一个功能集群的一部分），则特定的NonOsModuleInstantiation仅定义该功能的配置值，而不明确参与ProcessToMachineMapping 。

另一种场景是功能集群的分布式特性，不需要提供集中配置手段。这适用于例如PersistencyDeployment或PlatformHealthManagementContribution 。

功能集群的功能行为由几个贡献的总和决定。没有提供单一的配置实体。

然而，如果堆栈实现决定将这样的分布式功能集群实现为单个Executable ，则可以使用GenericModuleInstantiation来定义特定机器的启动行为。

AdaptiveModuleInstantiation的特化涵盖了各个 Adaptive Autosar 模块的配置设置。

*ARElement*

*AtpStructureElement*

Machine

trustedPlatformExecutableLaunchBehavior: TrustedPlatformExecutableLaunchBehaviorEnum

+

*Identifiable*

*AdaptiveModuleInstantiation*

OsModuleInstantiation

[0..1]

supportedTimerGranularity: TimeValue

+

«enumeration»

BuildTypeEnum

buildTypeDebug

buildTypeRelease

*Identifiable*

ResourceGroup

[0..1]

+

cpuUsage: PositiveInteger

+

memUsage: PositiveInteger

[0..1]

GenericModuleInstantiation

*AbstractExecutionContext*

Process

+

functionClusterAffiliation: String

[0..1]

+

numberOfRestartAttempts: PositiveInteger

[0..1]

preMapping: Boolean

[0..1]

+

*AtpPrototype*

*Identifiable*

RootSwComponentPrototype

*ARElement*

*AtpClassifier*

Executable

buildType: BuildTypeEnum

+

[0..1]

[0..1]

loggingBehavior: LoggingBehaviorEnum

+

[0..1]

minimumTimerGranularity: TimeValue

+

[0..1]

reportingBehavior: ExecutionStateReportingBehaviorEnum

+

version: StrongRevisionLabelString

[0..1]

+

*NonOsModuleInstantiation*

*Identifiable*

ProcessToMachineMapping

[0..1]

+

persistencyCentralStorageURI: UriString

«atpSplitable»

+

moduleInstantiation

0..\*

+

nonOsModuleInstantiation

0..1

process

+

1

+

machine

0..1

+

rootSwComponentPrototype

0..1

«atpUriDef»

+

executable

0..1

+

resourceGroup

0..\*

**图 9.1：自适应 Autosar 模块配置**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***AdaptiveModuleInstantiation*** （抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule 实现 | | | |
| ***笔记*** | 这个元类定义了用于在特定机器上配置自适应 autosar 模块实例的抽象属性。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***子类*** | 非*OsModuleInstantiation* \_ | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 9.1：自适应模块实例化**

除了 OS 之外的每个 Adaptive Autosar 模块都可以通过 ProcessToMachineMapping 分配给一个进程。

**[constr\_1490]** { DRAFT } **Executable的允许值。 category如果ProcessToMachineMapping引用NonOsModuleInstantiation** d如果ProcessToMachineMapping引用NonOsModuleInstantiation ，则ProcessToMachineMapping角色中引用的Process 。进程仅应（在角色Process.executable中）引用Executable where 属性Executable 。类别设置为PLATFORM\_LEVEL （参见 [ constr\_1605 ]）。 c *()*

请注意，[ constr\_1490 ] 中描述的模型关系如图9.1所示。

元类GenericModuleInstantiation可用于定义通用模块和未被 AUTOSAR 标准化的模块的配置设置。不同的模块可以通过类别属性来区分。

请注意，这两个元素都是可识别的，因此能够描述特殊数据 ( sdg )，这意味着可以定义标准模型未表示的通用自定义设置。有关更多信息，请参阅 AUTOSAR 通用结构模板 [ 6 ]。

**[TPS\_MANI\_03096]** { DRAFT }通用模块的**机器特定配置设置**d**通用模块的**机器特定配置设置收集在GenericModuleInstantiation中，其中属性类别值的值表示模块。 c *( RS\_MANI\_00023 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **通用模块实例化** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule 实现 | | | |
| ***笔记*** | 此元类定义特定机器上通用模块配置的属性。不同的模块可以通过类别属性来区分。此元素还可用于描述 AUTOSAR 未标准化的模块。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *AdaptiveModuleInstantiation* , *Identifiable* , *MultilanguageReferable* , *NonOsModule 实例化*, *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 9.2: GenericModuleInstantiation**

## 9.1 操作系统模块配置

**[TPS\_MANI\_03098]** { DRAFT } OS 模块的**机器特定配置设置**d **OS 模块**的机器特定配置设置收集在OsModuleInstantiation中。 c *( RS\_MANI\_00023 )*

|  |  |
| --- | --- |
| ***班级*** | **OsModule 实例化** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule 实现 |
| ***笔记*** | 此元类定义特定机器上操作系统配置的属性。  **标签：** atp.Status=draft |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **OsModule 实例化** | | | |
| ***根据*** | *ARObject* , *AdaptiveModuleInstantiation* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 资源组 | 资源组 | \* | 聚合 | 这表示封闭的 OsModuleImplementation 拥有的 ResourceGroups 的集合。  **标签：** atp.Status=draft |
| 支持的定时器粒度 | 时间价值 | 0..1 | 属性 | 此属性描述了支持的计时器粒度（一个刻度的 TimeValue）。  **标签：** atp.Status=draft |

**表 9.3：OsModuleInstantiation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***NonOsModuleInstantiation*** （抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule 实现 | | | |
| ***笔记*** | 此元类定义了用于配置除 OS 模块之外的自适应 autosar 模块的抽象属性。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *AdaptiveModuleInstantiation* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***子类*** | CryptoModuleInstantiation , DoIpInstantiation , GenericModuleInstantiation , IamModuleInstantiation , *Ids*  *平台实例化*， LogAndTrace实例化， Nm实例化，时间同步模块实例化， Ucm  模块实例化 | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| – | – | – | – | – |

**表 9.4: NonOsModuleInstantiation**

AUTOSAR 支持在机器的OsModuleInstantiation中配置ResourceGroup ，例如对应于 Linux 中的 cgroup（又名控制组）。 ResourceGroup提供了一种机制来管理系统资源，方法是将cpuUsage和memUsage等约束划分为限制进程集合的资源使用的组（另请参见 [ TPS\_MANI\_01017 ]）。

**[constr\_1661]** { DRAFT } **OsModuleInstantiation的多重性。 resourceGroup** d任何给定的 OsModuleInstantiation都应始终定义至少一个resourceGroup 。 c *()*

[ constr\_1661 ] 的基本原理是StateDependentStartupConfig需要对ResourceGroup的引用。

ResourceGroup语义的更多信息可以在 [SWS\_OSI\_02001] 中找到。

|  |  |
| --- | --- |
| ***班级*** | **资源组** |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule 实现 |
| ***笔记*** | 此元类表示限制进程集合的资源使用的资源组。  **标签：** atp.Status=draft |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **资源组** | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| CPU使用率 | 正整数 | 0..1 | 属性 | CPU 资源限制，以计算机上总 CPU 容量的百分比表示。**标签：** atp.Status=draft |
| 内存使用 | 正整数 | 0..1 | 属性 | 以字节为单位的内存限制。  **标签：** atp.Status=draft |

**表 9.5：资源组**

## 9.2 持久化部署

**9.2.1 概述**

本章从具体的存储模型映射到应用软件的相应部分来解释对持久存储的支持部分。

**[TPS\_MANI\_01205]** { DRAFT }元类**PersistencyDeployment的语义**d抽象元类PersistencyDeployment为更具体的专业化提供共享属性。 c *( RS\_MANI\_00027 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***持久性部署***（抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***笔记*** | 这个抽象元类作为代表持久性不同方面的具体类的基类。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable*  *元素*,*可引用*, *UploadableExclusivePackageElement* , *UploadablePackageElement* | | | |
| ***子类*** | 持久性文件存储，持久性键值存储 | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 最大  允许大小 | 正无限整数 | 0..1 | 属性 | 此属性的值表示封闭持久性部署在部署时允许的最大大小。  **标签：** atp.Status=draft |
| 最低限度  持续尺寸 | 正整数 | 0..1 | 属性 | 此属性的值表示在部署时为封闭的 PersistencyDeployment 保证的最小大小。  **标签：** atp.Status=draft |
| 冗余处理 | 持久性冗余  处理 | \* | 聚合 | 这种聚合代表了处理冗余的选择方法。  **标签：** atp.Status=draft |
| 更新策略 | 持久性集合  级别更新策略  枚举 | 1 | 属性 | 该属性用于指定各个 PersistencyDeployment 整体的更新策略。  **标签：** atp.Status=draft |

5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***持久性部署***（抽象） | |  |  |
| 版本 | 强修订标签  细绳 | 0..1 | 属性 | 该属性表示 PersistencyFile Storage 或 PersistencyKeyValueStorage 的版本。  **标签：** atp.Status=draft |

**表 9.6：持久性部署**

*UploadableExclusivePackageElement*

*PersistencyDeployment*

+

[0..1]

maximumAllowedSize: PositiveUnlimitedInteger

minimumSustainedSize: PositiveInteger

[0..1]

+

+

updateStrategy: PersistencyCollectionLevelUpdateStrategyEnum

version: StrongRevisionLabelString

+

[0..1]

*PersistencyRedundancyHandling*

[0..1]

scope: PersistencyRedundancyHandlingScopeEnum

+

PersistencyRedundancyCrc

PersistencyRedundancyMOutOfN

+

m: PositiveInteger

n: PositiveInteger

+

«enumeration»

PersistencyRedundancyHandlingScopeEnum

persistencyRedundancyHandlingScopeElement

persistencyRedundancyHandlingScopeStorage

*PersistencyRedundancyChecksum*

algorithmFamily: String

+

length: PositiveInteger

+

PersistencyRedundancyHash

[0..1]

+

initializationVectorLength: PositiveInteger

redundancyHandling

+

0..\*

**图 9.2：抽象基类PersistencyDeployment的建模**

**[constr\_10035]** { DRAFT } **PersistencyDeployment的完整性。版本**d PersistencyDeployment 。 \_版本应包含以下所有部分：

* 主要版本
* 次要版本
* 补丁版本
* **清单完成时**预发布版本和构建元数据的附加标签。 c *()*

版本用于确定是否需要更新或回滚持久数据。当应用程序打开PersistencyFileStorage或PersistencyKeyValueStorage时，Persistency 模块将检查PersistencyDeployment 。清单中的版本对照存储的版本信息。

如果存储的版本低于清单版本，则持久性模块将在创建数据备份后更新持久性数据（有关详细信息，请参阅 [SWS\_PER\_00386]）。

如果存储的版本高于清单版本，则需要从备份中回滚持久数据（有关详细信息，请参阅 [SWS\_PER\_00396]）。

**[TPS\_MANI\_01321]** { DRAFT }元类**PersistencyDeploymentElement的语义**d元类PersistencyDeploymentElement表示一个抽象基类，用于在元素级别对持久性的不同方面进行建模。 C

*( RS\_MANI\_00027 )*

*Identifiable*

*PersistencyDeploymentElement*

+

updateStrategy: PersistencyElementLevelUpdateStrategyEnum

[0..1]

PersistencyFile

+

contentUri: UriString

[0..1]

+

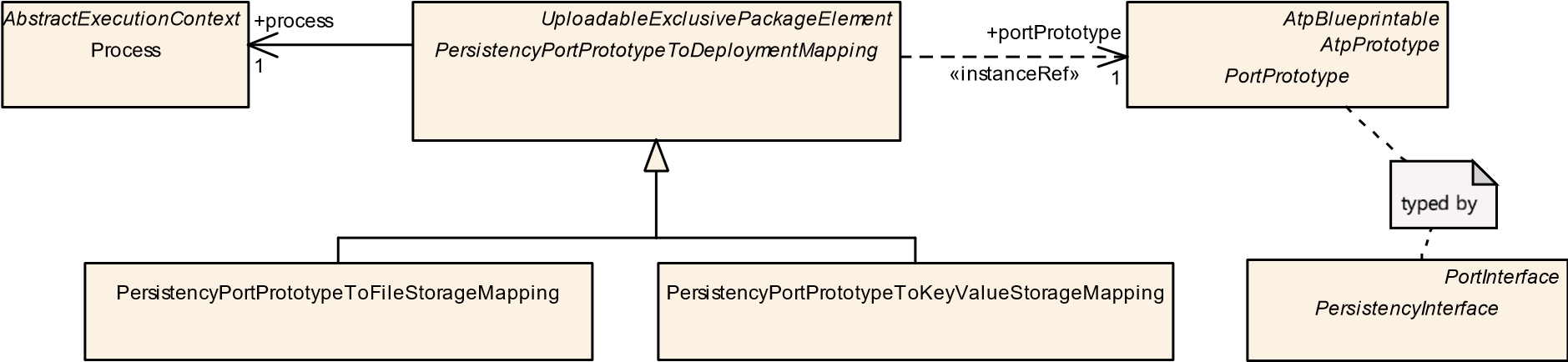
fileName: String

PersistencyKeyValuePair

**图 9.3：抽象基类PersistencyDeploymentElement的建模**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***持久性部署元素***（抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***笔记*** | 这个抽象元类用作表示 PersistencyDeployment 元素的不同方面的具体类的基类。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *Referrable* | | | |
| ***子类*** | 持久性文件,持久性键值对 | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 更新策略 | 持久性元素  级别更新策略  枚举 | 0..1 | 属性 | 该属性可用于指定各个 PersistencyDeploymentElement 的更新策略。  **标签：** atp.Status=draft |

**表 9.7：PersistencyDeploymentElement**



**图 9.4：抽象基类PersistencyPortPrototypeToDeploymentMapping的建模**

**[TPS\_MANI\_01322]** { DRAFT }元类**PersistencyPortPrototypeToDeploymentMapping的语义**d元类PersistencyPortPrototypeToDeploymentMapping表示一个抽象基类，用于对具体持久性案例（键值存储、文件存储）到PortPrototype和Process的映射进行建模。 c *( RS\_MANI\_00027 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | ***PersistencyPortPrototypeToDeploymentMapping*** （抽象） | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***笔记*** | 这个抽象的基类实现了 PortPrototype、Process 和 PersistencyDeployment 的特定子类之间的所有映射的共享功能。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *Packageable*  *元素*,*可引用*, *UploadableExclusivePackageElement* , *UploadablePackageElement* | | | |
| ***子类*** | PersistencyPortPrototypeToFileStorageMapping , PersistencyPortPrototypeToKeyValueStorageMapping | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 端口原型 | 端口原型 | 0..1 | 参考 | 此引用表示映射的 PortPrototype。  **标签：** atp.Status=draft  **InstanceRef 实现者：** PortPrototypeIn  可执行实例引用 |
| 过程 | 过程 | 1 | 参考 | 此引用表示作为映射上下文所需的过程。  **标签：** atp.Status=draft |

**表 9.8：PersistencyPortPrototypeToDeploymentMapping**

**9.2.1.1 冗余处理**

**[TPS\_MANI\_01206]** { DRAFT } **PersistencyDeployment上下文中的冗余建模**d部署级别能够为键值存储和文件存储提供更详细的冗余行为定义。

此建模附加到抽象基类PersistencyDeployment以便让*AUTOSAR 自适应平台上的持久性的两个方面都*受益于元类PersistencyRedundancyHandling的存在。 c *( RS\_MANI\_00027 )*

**[constr\_1710]** { DRAFT }**属性值的一致性PersistencyInterface 。冗余和PersistencyRedundancyHandling 。范围**d如果属性PersistencyInterface 。冗余设置为值PersistencyRedundancyEnum 。冗余PerElement然后属性PersistencyRedundancyHandling 。范围应设置为PersistencyRedundancyHandlingScopeEnum 。由对应的PersistencyDeploy聚合的至少一个PersistencyRedundancyHandling的persistencyRedundancyHandlingScopeElement-

换货。 c *()*

**9.2.1.2 更新处理**

**[TPS\_MANI\_01155]** {草案}**持久性部署。 updateStrategy覆盖PersistencyInterface 。 updateStrategy** d属性PersistencyDeployment的值。 updateStrategy应否决PersistencyInterface的值。通过PersistencyPortPrototypeToDeploymentMapping映射到PersistencyDeployment的任何PersistencyInterface组合的updateStrategy 。 c *( RS\_MANI\_00027 )*

**[TPS\_MANI\_01147]** { DRAFT } **PersistencyDeployment的语义。 updateStrategy** d属性PersistencyDeployment 。 updateStrategy用于指定更新实际持久元素的策略。

此更新策略应作为一个整体应用于PersistencyDeployment ，但明确建模的PersistencyDeploymentElement定义了自己的updateStrategy 。 c *( RS\_MANI\_00027 )*

**[TPS\_MANI\_01157]** { DRAFT } **updateStrategy在集合级别的语义**d属性updateStrategy在集合级别的语义在表中指定

9.9 . c *( RS\_MANI\_00027 )*

|  |  |  |
| --- | --- | --- |
| **更新策略** | **用例：安装** | **用例：更新** |
| 删除 | 无关紧要的 | 删除当前清单中未包含的所有元素 |
| 保持现有 | 无关紧要的 | 保留当前清单中未包含的所有元素 |

**表 9.9：集合级别的updateStrategy语义**

**[TPS\_MANI\_01313]** { DRAFT }**元素级别更新策略的定义 d 元素级别更新**策略的定义通过抽象基类PersistencyDeploymentElement （及其属性**updateStrategy ）进行**建模，持久性元素的具体子类从中派生. c *( RS\_MANI\_00027 )*

**[TPS\_MANI\_01159]** { DRAFT } **updateStrategy在元素级别的语义**d属性updateStrategy在元素级别的语义在表中指定

9.10 。 c *( RS\_MANI\_00027 )*

|  |  |  |
| --- | --- | --- |
| **更新策略** | **用例：安装** | **用例：更新** |
| 删除 | 不要创建 | 消除 |
| 保持现有 | 创造 | 没做什么 |
| 覆盖 | 创造 | 代替 |

**表 9.10：元素级别的updateStrategy语义**

**[TPS\_MANI\_01148]** {草稿} **PersistencyDeploymentElement的语义。 updateStrategy** d属性PersistencyDeploymentElement 。 updateStrategy可用于指定更新与PersistencyDeploymentElement对应的实际持久元素的策略。 c *( RS\_MANI\_00027 )* **[TPS\_MANI\_01156]** {草案} **PersistencyDeploymentElement 。 updateStrategy覆盖PersistencyDeployment 。 updateStrategy** d为PersistencyDeploymentElement指定的值。 updateStrategy覆盖PersistencyDeployment的值。此特定PersistencyDeploymentElement的updateStrategy 。 c *( RS\_MANI\_00027 )*

**[TPS\_MANI\_01182]** { DRAFT } **PersistencyDeploymentElement的值。 updateStrategy覆盖PersistencyInterfaceElement 。 updateStrategy** d属性PersistencyDeploymentElement的值。 updateStrategy覆盖属性PersistencyInterfaceElement的值。*更新策略*c *(* RS\_MANI\_00027 *)*

这意味着软件集成商有权同意设计者的观点，或者基于对集成策略的高级知识推翻设计者的决定。

**9.2.1.3 尺寸处理**

**[TPS\_MANI\_01196]** {草稿} **PersistencyDeployment的语义。 minimumSustainedSize** d属性PersistencyDeployment 。 minimumSustainedSize可用于从集成商的角度定义PersistencyDeployment需要分配的**最小存储量**。

底层平台有责任确保此最小存储量随时可用。 c *( RS\_MANI\_00027 )*

**[TPS\_MANI\_01197]** {草案} **PersistencyDeployment的语义。 maximumAllowedSize** d属性PersistencyDeployment 。从集成商的角度来看， maximumAllowedSize可用于定义PersistencyDeployment在运行时可以分配的**最大存储量**。

PersistencyDeployment的存在。 maximumAllowedSize不构成对平台的绑定要求，即该存储量应随时可用。 c *( RS\_MANI\_00027 )*

为了解释，可用存储量至少应为minimumSustainedSize值的总和。

也就是说，如果所有键值存储同时分配的存储量超过最小存储量（更不用说最大量），那么存储可能会被超出是合理的。

**9.2.1.4 安全处理**

存储在 Key-Value Storage 或 File Storage 中的数据的加密和/或身份验证在清单中由PersistencyDeploymentToCryptoKeySlotMapping或PersistencyDeploymentElementToCryptoKeySlotMapping进行描述，在第13.4章和第 13.4 章中有更详细的描述

13.5 .

如果PersistencyDeploymentToCryptoKeySlotMapping 。 keySlotUsage或

PersistencyDeploymentElementToCryptoKeySlotMapping 。 keySlotUsage设置为加密，持久性集群将在将数据存储到持久性内存之前对其进行加密，或者在从持久性内存中读取数据后对其进行解密。

如果PersistencyDeploymentToCryptoKeySlotMapping 。 keySlotUsage或

PersistencyDeploymentElementToCryptoKeySlotMapping 。 keySlotUsage设置为verify ，Persistency 集群将在将数据存储到持久内存之前对数据进行签名，或者在从持久内存读取数据后验证数据的签名。

请注意， PersistencyDeploymentToCryptoKeySlotMapping能够定义一个验证哈希，PersistencyCluster 将使用该验证哈希来验证数据。 PersistencyDeploymentElementToCryptoKeySlotMapping也是如此。验证哈希。

**9.2.2 持久键值存储部署**

**[TPS\_MANI\_01079]** { DRAFT } **PersistencyKeyValueStorage的语义**d元类PersistencyKeyValueStorage表示用于持久存储数据的实际键值存储。 c *( RS\_MANI\_00027 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **持久性键值存储** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***笔记*** | 此元类表示在部署级别上对键值存储进行建模的能力。  **标签：**  atp.Status=草稿  atp.recommendedPackage=PersistencyKeyValueStorages | | | |
| ***根据*** | *ARElement* , *ARObject* , *CollectableElement* , *Identifiable* , *MultilanguageReferable* , *PackageableElement* ,  *PersistencyDeployment* ,*可引用*, *UploadableExclusivePackageElement* , *UploadablePackageElement* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 键值对 | 持久性键值  一对 | \* | 聚合 | 此聚合表示封闭 PersistencyKeyValueStorage 拥有的键值对。  **标签：** atp.Status=draft |
| 乌里 | 字符串 | 0..1 | 属性 | 该属性保存存储位置  PersistencyKeyValueStorage，例如文件系统上的文件。  **标签：** atp.Status=draft |

**表 9.11: PersistencyKeyValueStorage**

**[TPS\_MANI\_01144]** { DRAFT } **PersistencyKeyValuePair** d Metaclass PersistencyKeyValuePair的**语义**表示用于持久存储数据的键值存储（由PersistencyKeyValueStorage正式化）的**条目。** c *( RS\_MANI\_00027 )*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***班级*** | **持久性键值对** | | | |
| ***包裹*** | M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency | | | |
| ***笔记*** | 这个元类代表了在持久性部署的上下文中对键值对进行形式化建模的能力。  **标签：** atp.Status=draft | | | |
| ***根据*** | *ARObject* , *Identifiable* , *MultilanguageReferable* , *PersistencyDeploymentElement* , *Referrable* | | | |
| ***属性*** | ***类型*** | ***多。*** | ***种类*** | ***笔记*** |
| 初始值 | 价值规范 | 0..1 | 聚合 | 此聚合表示为键值对的值侧定义初始值的能力。请注意，如果 PersistencyDeploymentElement.updateStrategy 设置为值 delete，则配置初始值没有意义。  **标签：** atp.Status=draft |
| 值数据类型 | 抽象实现  数据类型 | 1 | 参考 | 此引用表示适用于键值对值的数据类型。  **标签：** atp.Status=draft |

**表 9.12: PersistencyKeyValuePair**

**[constr\_10083]** { DRAFT } **PersistencyKeyValuePair的初始值的存在**d对于每个PersistencyKeyValuePair ，如果属性updateStrategy的值设置为值delete ，则属性PersistencyKeyValuePair 。 initValue不应存在。 c *()*

PersistencyKeyValuePair的建模聚合在角色PersistencyKeyValueStorage中。 keyValuePair是可选的。无论keyValuePair是否存在，都可以使用持久性功能。

然而， keyValuePair的存在为行为的定制提供了更多的自由和方式。

**[TPS\_MANI\_01078]** { DRAFT } **PersistencyPortPrototype的语义 -**

**ToKeyValueStorageMapping** d元类PersistencyPortPrototypeToKeyValueStorageMapping能够将角色portPrototype中引用的特定PortPrototype映射到角色keyValueStorage中引用的PersistencyKeyValueStorage 。

该映射还包括对元类进程的引用，以适应这样一个事实，即keyValueStorage和portPrototype的相同组合可能会或可能不会应用于在运行时表示封闭可执行文件的给定进程。 c *( RS\_MANI\_00027 )*

1. 未来可能会将设计元素的描述移至其他与模型相关的文档中。但就目前而言，本文档中存在清单相关和设计相关的模型元素并存的情况。 [↑](#footnote-ref-1)
2. 另一方面，“地图”的两个概念在各自的“社区”中都是合理的，并且选择将其中一个命名为非常不同的名称以减少整体潜在的混淆可能不适用 [↑](#footnote-ref-2)
3. 如 [ 1 ]中所述 [↑](#footnote-ref-3)
4. 即使可以将ImplementationDataType扩展到或多或少地对 C++ 的干净支持，也可能会在*AUTOSAR 自适应平台中添加更多的语言绑定，这*将需要对ImplementationDataType进行进一步的扩展。 [↑](#footnote-ref-4)
5. 这意味着命名空间 a::b 的定义在语义上不同于命名空间 b::a 的定义。 [↑](#footnote-ref-5)
6. 有关元类DataTypeMappingSet的定义和使用的更多背景信息，请参阅 [ 1 ]。 [↑](#footnote-ref-6)
7. 这个缩写代表标签长度值 [↑](#footnote-ref-7)
8. 例如，对于属性方向设置为in的参数s 使用 TLV 编码，但对于属性方向设置为out的参数s**不使用**。 [↑](#footnote-ref-8)
9. 如果一个结构仅由可选元素组成，则很难检测到数组元素携带的这种结构碰巧将所有元素设置为不可用的情况。 [↑](#footnote-ref-9)
10. 即在创建执行清单期间 [↑](#footnote-ref-10)
11. 从方法论的角度来看，诊断映射的创建通常被认为是设计时活动。 [↑](#footnote-ref-11)
12. 换句话说，如果设计相关元类和部署相关元类之间需要引用，那么这些引用的方向应始终指向从部署到设计。 [↑](#footnote-ref-12)
13. 在应用软件的末端 [↑](#footnote-ref-13)
14. 有关详细信息，请参阅第14.2节。 [↑](#footnote-ref-14)